CONSTRUCCIÓ MODULAR D'AGENTS BASATS EN COMPONENTS. ESTUDI I DESENVOLUPAMENT.

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Carles Martínez García. i dirigit per Álvaro Moratalla Anaya.

Bellaterra, juny de 2007

El firmant, Álvaro Moratalla Anaya , professor del Departament d'Enginyeria de la Informació i de les Comunicacions de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Carles Martínez García

Bellaterra, juny de 2007

Firmat: Álvaro Moratalla Anaya

A tota la gent que m'ha recolzat durant aquests últims cinc anys

Agraïments

M’agradaria donar les gràcies a totes les persones que m’han fet costat aquests últims cinc anys. Als companys i amics des del primer dia de la carrera especialment a l’Abraham Martín per tots el moments que hem compartit. De la mateixa manera vull agrair de forma especial al projectistes del SeNDA. A Miriam, la meva parella i a tota la meva família. Gràcies a tots per fer això possible.
Menció especial al meu director, Álvaro, a qui dec part d’aquest treball.

Índex

Índex de figures

Capítol 1

Introducció.

A finals del segle XX neix el concepte de societat de la informació, societat en la que actualment vivim. Gràcies a l'expansió de les tecnologies de la informació i la comunicació (TIC) abandonem un model de societat on el coneixement i la informació romanen de forma centralitzada, en petits nuclis, per endinsar-nos en un model de societat on la informació es troba de forma totalment distribuïda. Informació, ara accessible a gairebé tothom des de qualsevol lloc a través d'Internet, no obstant, sotmesa a possibles restriccions i limitacions en la nostra capacitat de tractament.

Des del punt de vista informàtic, es podria dir que en els últims anys han aparegut escenaris on la informació queda distribuïda de forma heterogènia en un ampli conjunt de sistemes. Aquests escenaris han estat batejats sota el nom Mar-De-Dades. Així doncs, actualment ens trobem amb models de grans sistemes on la informació es troba totalment descentralitzada i on, a més, sovint no es coneix a priori la seva localització.

Com s'ha citat prèviament, la informació pot estar sotmesa a restriccions. Aquestes restriccions poden aparèixer en l'autorització d'accés a la pròpia dada. A més, centralitzar la informació com a pas previ al seu tractament pot ser inviable en termes d'ample de banda en les comunicacions o quota de disc per l'emmagatzematge local. També també es pot considerar una restricció el fet que, a priori, no es coneix la localització de la totalitat de la informació.

Les limitacions en l'accés poden venir donades, per exemple, per restriccions legals. Podem pensar en bases de dades que contenen informació sobre persones físiques identificades o identificables i que, per tant, queda sotmesa a la llei orgànica de protecció de dades. La informació, per exemple, pot ser tractada per part de terceres persones amb l'objectiu de fer *data mining* sempre i quan no és reveli cap dada personal. Una mostra d'aquesta situació pot ser el tractament permès d'expedients de diversos hospitals per tal de realitzar un estudi epidemiològic. Les limitacions en l'ample de banda i/o emmagatzemament local responen al fet que la quantitat de dades pot ser tan gran que no és viable concentrar-les en un sol punt per tractar-les a posteriori.

Juntament amb els escenaris Mar-De-Dades, ha sorgit la necessitat de trobar solucions als problemes que presenta el tractament d'informació localitzada de forma distribuïda. Fruits d'aquesta necessitat sorgeix un nou paradigma de desenvolupament; els agents mòbils.

Els agents son entitats autònomes que tracten la informació actuant en base a un objectiu. Si a més son mòbils tindran la capacitat de desplaçar-se i ampliar els seus horitzons, arribant a major quantitat d'informació. Un agent mòbil pot ser vist com un navegant en un mar de dades.

Aquest nou paradigma soluciona els problemes anteriorment citats. La idea bàsica és no apropar les dades al algorisme sinó apropar l'algorisme a les dades. D'aquesta manera l'ample de banda en les comunicacions requerit per tal d'ajuntar algorisme i dades és molt més petit. De la mateixa manera, els requisits en termes de memòria per l'emmagatzemament de les dades també es veuen reduïts. A més, una entitat mòbil, de forma recorrent, pot anar construint un itinerari dinàmic a mesura que troba informació referent a la localització de més informació.

Des de l'aparició del paradigma dels agents mòbils s'han creat i perfeccionat fluxos de disseny, així com eines de desenvolupament i mecanismes de seguretat. No obstant, encara queda un llarg camí a recórrer fins la consolidació dels agents mòbils com un solució viable. Actualment, les principals mancances son la dificultat a l'hora de construir un agent robust i la falta d'implantacions importants i conegudes que demostrin la viabilitat del paradigma i que estableixin una

senda a seguir. No obstant, hi ha línies de recerca actives que pretenen pal·liar aquestes mancances. A [VRCCNM525] es proposa una solució pel tractament d'informació mèdica als hospitals mitjançant agents mòbils.

La construcció d'un agent mòbil presenta greus dificultats si s'enfoca des d'un punt de vista monolític. El procés d'anàlisi, disseny i implementació pot imbricar diverses doctrines tan diverses com la seguretat o la tolerància a fallades. Una primera solució és abandonar l'arquitectura monolítica per enfocar el disseny d'una manera modular. D'aquesta manera no tan sols es facilitaria la reutilització de codi sinó que a més, cada mòdul pot ser dissenyat i programat per desenvolupadors experts en el seu àmbit. Aquest projecte gira entorn a la construcció modular d'agents mòbils.

A l'hora de programar un agent mòbil, no només s'han de tenir en compte les tasques que s'encarreguen del problema a resoldre, sinó que s'han de tenir presents molts més factors. La seguretat en totes les seves dimensions, la localització de l'agent en cada moment i la tolerància a fallades son clars exemples. Un programador d'agents requereix un alt grau de coneixement en diverses doctrines per tal d'implementar un agent robust. A més, aquest problema pot implicar que un programador es vegi forçat a dedicar més temps desenvolupant les tasques secundàries de l'agent que no pas les tasques que defineixen la feina a realitzar per part d'aquest. La idea defensada al grup de recerca SeNDA [senda] és lliurar al programador d'agents de tot allò que no estigui directament implicat amb la tasca que ha de realitzar l'agent. De la resta de funcionalitats s'encarreguen experts. En ultima instància, el programador simplement haurà d'afegir les funcionalitats que desitgi al seu agent seleccionant-les d'una llista.

Es poden diferenciar dos tipus de característiques comunes a tots els agents. El primer tipus és la seguretat a nivell d'arquitectura. Dit d'una altra manera; la forma en que l'agent s'auto-protegeix [ARO347]. Un dels grans reptes dels agents mòbils ha estat mitigar el problema que suposa que els agents s'executin en entorns aliens al control de l'usuari i potencialment insegurs. El segon tipus de característica fa referència a funcionalitats de rerefons que hauria de presentar un agent: informar sobre la seva posició, tolerància a fallades, capacitat de desplaçar-

se, etcètera. A aquest últim tipus els anomenarem components i seran la pedra angular del nostre projecte.

En definitiva, podem concloure amb un símil matemàtic. El desenvolupament d'agents és una expressió molt complexa. Si traiem factor comú, estem fraccionant la complexitat a subproblemes més petits. Si a més, fruit de aplicar factor comú, obtenim equacions que prèviament han resolt experts, la feina de total de resolució de l'expressió ha quedat minimitzada.

Arribats a aquest punt, hem introduït el context del problema i s'han citat les motivacions per a la realització d'aquest treball. Tot seguit es citen tots i cadascun dels objectius inicials:

- Identificar els elements bàsics imbricats en el disseny modular d'agents.
- Estudiar els requisits de la construcció d'agents basada en components.
- Analitzar i desenvolupar dos components d'exemple. Els components triats afegiran als agents mòbils la funcionalitat d'informar sobre la seva posició i establiran un mecanisme d'entrega de resultats a l'usuari.
- Integrar el projecte amb les eines del grup de recerca. El desenvolupament s'ha d'adequar a les eines existents per tal de contribuir amb el compromís del departament.
- Integració amb el projecte que es desenvolupa en paral·lel "Desenvolupament d'un servei criptogràfic per a la plataforma JADE" afegint així els requisits de seguretat exigits a qualsevol aplicació amb agents mòbils.

Un cop presentats els objectius es mostra com s'estructura la resta de la memòria:

Capítol 2: Fonaments. Amb els continguts expressats en aquest capítol és pretén aportar al lector els conceptes necessaris per situar-lo dins del context d'aquest projecte. S'exposarà una definició del concepte "agents mòbils", es mostrarà la necessitat dels components, com interaccionen els agents, l'entorn d'execució, la plataforma de desenvolupament, les eines del departament, el concepte d'auto-protecció i finalment es citaran els estàndards als quals s'ha de cenyir el treball realitzat.

Capítol 3: Anàlisi. Al llarg del capítol d'anàlisi es mostraran tots els elements que conformen el projecte així com els seus requisits que presenten inicialment.

Capítol 4: Disseny i implementació. Al llarg del capítol 4 s'exposaran els detalls de disseny i implementació dels mòduls citats al capítol d'anàlisi. Veurem les implicacions dels requisits presentats al capítol previ i com finalment han estat cobertes.

Capítol 5: Proves, execució i integració. A mesura que es desenvolupa el capítol 5 es mostraran els elements necessaris per a l'execució dels elements implementats. Acte seguit es presentaran els resultats de les proves realitzades i un petit anàlisi dels resultats d'integració amb el projecte "Desenvolupament d'un servei criptogràfic per a la plataforma JADE" i la resta d'eines del departament.

Capítol 6: Conclusions i línies d'ampliació. Al capítol final s'esposaran les conclusions que s'han extret del treball així com un llistat dels objectius acomplits i pendents. Per finalitzar es citaran les línies de recerca que s'han considerat vàlides per a l'ampliació del treball realitzat.

Capítol 2

Fonaments.

Un cop presentades quines son les motivacions d'aquest projecte, ens endinsem en el capítol de fonaments. Al llarg d'aquest capítol es presentaran totes les qüestions de rerefons que envolten la feina a realitzar amb l'objectiu de situar al lector dins del marc de treball.

2.1 Agents mòbils

Un agent és una entitat autònoma que percep i actua en base a uns objectius. Si a més aquest agent presenta l'habilitat de desplaçar-se físicament dins del món on es troba, l'anomenarem agent mòbil. Aquesta és una definició del terme agent mòbil que rau d'una vessant més propera a l'intel·ligència artificial. A baix nivell, i des del punt de vista purament informàtic, un agent mòbil és un algorisme capaç de desplaçar-se autònomament entre diferents entorns d'execució albergats en diferents màquines. És de vital importància remarcar la paraula "algorisme" ja que ens ajudarà a comprendre quines son les motivacions i la raó de ser dels agents mòbils.

Programa = Algorismes + Estructures de dades. Aquesta és la definició clàssica del terme "programa". Els puristes d'enginyeria del software afegirien a aquesta definició la documentació del producte, de fet, és el tema que ens ocupa ara. No obstant, cal remarcar "Algorismes + Estructures de dades". Hem vist que els

agents (mòbils, o no) són algorismes però el punt clau per entendre la seva necessitat son les dades sobre les que actuen. No obstant, i abans d'entrar en profunditat en el tema, cal esclarir que en el fons al referir-nos al concepte "agent mòbil" ens referim a una manera de enfocar i solucionar problemes. Si es vol, ens referim a un paradigma de programació.

"Si la muntanya no va a Mahoma... Mahoma anirà a la muntanya". Aquesta frase reflexa de forma molt clara la raó de ser dels agents mòbils. Des del punt de vista del marc on ens trobem, la podem reescriure com: "Si les dades no van a l'algorisme, l'algorisme anirà a les dades". Però aturem-nos de nou, i només per un instant, en la proverbial primera frase per veure que els agents mòbils neixen de l'aplicació al món de la informàtica d'una vella estratègia humana per solucionar problemes .

Entrem ja en profunditat en la necessitat dels agent mòbils vista des de la vessant informàtica. Imaginem un entorn informàtic on hi ha milers de ordinadors connectats entre ells i on es localitza infinitat de informació distribuïda. Si es volgués tractar les dades caldria, bé tenir N executables replicats a cada màquina o bé concentrar totes les dades en una màquina per poder-les tractar. Impensable d'ambdues maneres, bé per que per qüestions d'escalabilitat, no podem tenir a cada màquina un executable per cada aplicació que poguéssim pensar, o bé per que concentrar les dades en una sola màquina pot ser inviable a nivell d'emmagatzematge i/o comunicacions. La solució que proposa el paradigma dels agents mòbils és tenir una sola aplicació comuna a cada màquina que possibiliti la recepció i execució d'agents. D'aquesta manera son els algorismes qui es mouen a la recerca de dades. Un altre entorn d'aplicació dels agents mòbils és, per exemple, un entorn de dades distribuïdes on les dades estan sotmeses a impediments de distribució com pot ser la llei orgànica de protecció de dades. Traient factor comú, podem concloure que els entorns d'aplicació ideals dels agents mòbils son tots aquells entorns en que surt més a compte moure l'algorisme que moure les dades.

2.2 Agències

A la secció anterior s'ha comentat que els agents mòbils requerien d'una aplicació a cada màquina del seu itinerari. Aquesta aplicació rep el nom d'agència (o plataforma) i és l'entorn d'execució dels agents. Aquest entorn actua de contenidor per tal de restringir i controlar les accions que efectuen els agents sobre la màquina física on s'executen. D'aquesta manera s'ofereix protecció al sistema contra agents que vulguin executar accions no permeses. A més, les agències proporcionen als agents serveis com poden ser una interfície per l'enviament i recepció de missatges o un servei de migració inter-plataforma.

Una plataforma que té control total sobre l'entorn on s'està executant un agent pot realitzar atacs de tota mena. Des de manipular les dades que porta l'agent a sobre fins a denegar-li el servei, passant per destruir l'agent, manipular-ho, duplicar-lo, etcètera. No obstant, aquest panorama que pot semblar tan pessimista a la pràctica no ho és tant. En primer lloc la majoria de les agències no tenen per que ser malicioses. A més, existeixen línies de recerca [ARO347] que tracten de auto-protegir els agents per tal suprimir o mitigar els efectes de possibles atacs realitzats per part d'una agència amb males intencions. En ultima instància, però, és un problema a tenir present a l'hora de desenvolupar un agent mòbil per actuar en conseqüència.

2.3 Divideix i venceràs: Els components.

Arribats a aquest punt, hem vist que un agent mòbil és un algorisme que actuant de forma autònoma és capaç de desplaçar-se i executar-se entre diferents agències. De fet, no és rigorosament del tot cert; un agent mòbil pot ser un conjunt d'algorismes. Al llarg d'aquesta secció tractarem de desglossar l'estructura d'un agent mòbil per veure com podem aplicar una altra clàssica estratègia de resolució de problemes: "divideix i venceràs", que ens servirà per introduir les necessitats de creació de components.

La creació d'un agent mòbil es complica perillosament a mesura que hi volem

afegir funcionalitats. Fins i tot l'agent més simple es complica molt quan volem afegir requeriments de seguretat bàsics per garantir la confidencialitat de les dades que hi generarà. Però la complicació no es fa notar no només en l'estructura de l'agent, sinó també en el grau de coneixement que es requereix al programador per afegir a l'agent certes funcionalitats. Hi ha dues tendències a l'hora d'enfocar aquest problema. La primera d'elles defensa oferir totes aquelles funcionalitats secundàries a un agent mòbil com serveis de l'entorn d'execució. D'aquesta manera el programador d'agents únicament s'hauria de limitar a fer servir aquestes funcionalitats. El problema de base que presenta aquest enfocament és que l'actualització o inclusió d'un nou servei a una agència implica repetir el mateix procés a totes les agencies del sistema. L'altra tendència és la reflectida a [M06] i és en la que es basa aquest projecte. La proposta és dividir l'agent mòbil en mòduls i oferir una llibreria de components, a disposició del programador, preparats per ser inclosos a l'hora de la construcció de l'agent. En resum, el que es pretén és agilitzar la construcció dels agents i reduir els coneixements requerits al programador en diferents matèries oferint funcionalitats ja creades que es pressuposen factor comú a la majoria dels agents mòbils. L'esquema es pot observar a la figura 2.1.

A partir d'ara haurem de veure un agent mòbil com una col·lecció de comportaments (algorismes) que s'executen en un ordre definit. Un altre concepte a tenir en compte és el de "component". Un component és una funcionalitat comuna a un ampli nombre d'agents. El programador haurà de generar els algorismes que adoptarà l'agent en cada estació del seu itinerari i incloure els components que requereix el seu agent en temps de construcció. L'ordre d'execució dels components no queda totalment oberta al programador sinó que l'eina de construcció proposta a [MR] s'encarregarà de garantir la correcta execució dels mòduls. Mes endavant es fara un estudi de diferents components i posteriorment els seus requeriments d'implementació.

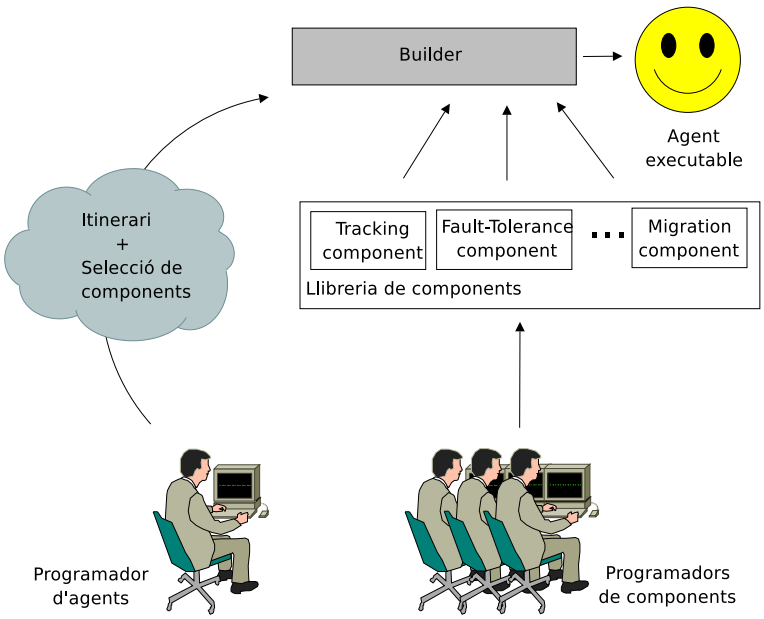

Figura 2.1: Entorn de desenvolupament d'agents basats en components.

2.4 La plataforma Jade

El marc de desenvolupament d'aquest projecte es centra en l'entorn JADE [jade] (Java Agent DEvelopment framework). Jade és un projecte de codi obert i sotmès als termes de la llicència LGPL (Lesser General Public License). Sense entrar en massa profunditat bàsicament el que ofereix Jade és una llibreria per al desenvolupament d'agents així com un entorn d'execució per aquests, és a dir; una agència.

En aquesta secció veurem l'estructura dels agent que es proposa a Jade i tot seguit veurem una breu descripció dels serveis utilitzats per la realització d'aquest projecte.

2.4.1 Estructura d'un agent Jade

Jade proposa l'estructura dels seus agents com una entitat principal *agent* que conté una serie de comportaments (o *behaviours*). L'entitat *agent* és la unitat mínima d'execució en l'entorn Jade i realitza les tasques pròpies que se li pressuposen a l'agent, com ara enviar i rebre missatges, migrar, etcètera. Els *behaviours* conte-

nen les accions que realitzarà l'agent dins d'una plataforma, per exemple, realitzar consultes a una base de dades. La figura 2.2 mostra l'estructura d'un agent Jade.

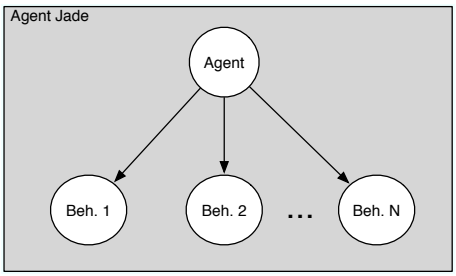

Figura 2.2: Esquema dels agents Jade.

2.4.2 Serveis

Els serveis en Jade son funcionalitats afegides a les agències. Per la realització d'aquest projecte hem fet servir el servei de migració i el servei de criptografia.

Servei de mobilitat inter-plataforma

El servei de migració [ipmp] ha estat implementat al grup SeNDA i La seva funcionalitat principal és la de habilitar que els agent puguin migrar d'una agència a una altra. Sense aquest servei funcionant, els agent de Jade no es poden considerar mòbils entre plataformes.

Servei de criptografia

El servei criptogràfic ha estat desenvolupat de forma paral·lela a aquest projecte. Bàsicament proporciona els mecanismes de xifratge i signatura digital indispensables per afegir característiques de seguretat als agents mòbils.

2.5 Eines de desenvolupament

Un dels objectius d'aquest projecte és integrar-se amb el conjunt d'eines de desenvolupament [smard] que s'estan construint al grup de recerca SeNDA. En aquesta secció farem un esbós d'aquestes eines per entendre millor com interaccionen amb el nostre projecte.

2.5.1 IDT

L'eina IDT és una interfície gràfica que permet la construcció de l'itinerari d'un agent mòbil així com les tasques que ha d'executar a cada agència que visitarà l'agent. Des de la interfície el programador d'agents pot iniciar el procés de construcció mitjançant el *Agent Builder*, eina que veurem tot seguit. L'objectiu d'integració amb aquesta eina és que des de la IDT es puguin seleccionar els components que s'afegiran a l'agent mòbil.

2.5.2 Agent Builder

L'eina *Agent Builder* rep les especificacions de l'agent des de la IDT i genera una estructura segura amb el codi a executar a cada plataforma. A més, es generarà una part de control que contindrà els components inclosos a l'agent.

2.5.3 Agent Launcher

El *Agent Launcher* és l'eina encarregada de posar en marxa l'agent mòbil. Mitjançant el servei de migració envia l'agent a la primera agència on començarà la seva execució

2.6 Agents auto-protegits

A la secció anterior s'han mostrat les eines de desenvolupament d'agents on s'emmarca aquest projecte. En aquest apartat mostrarem quina és l'estructura dels agents que generen les eines anteriors i quins son els seus atributs de seguretat.

La idea bàsica dels agents auto-protegits és que cada agència només serà capaç d'accedir a la part de codi dels agents generat per ser executat en aquella agència. Els components, però, no es protegeixen d'aquesta manera ja que s'executaran a cada plataforma. La figura 2.3 mostra l'estructura citada.

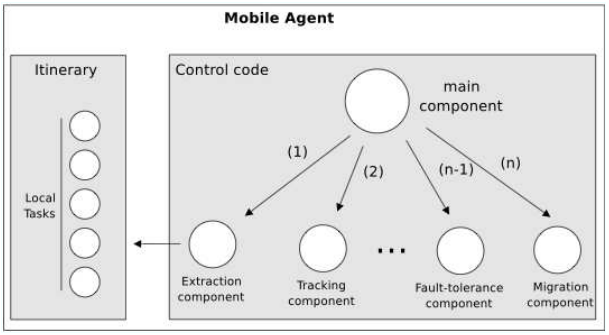

Figura 2.3: Estructura dels agents auto-protegits.

Mitjançant la clau pública de cada plataforma el codi es protegeix per restringir l'accés a ell. Construint els agents d'aquesta manera s'evita que les agències puguin accedir i alterar al codi generat per ser executat en altres agències.

Un component fonamental d'aquest esquema és el component d'extracció. Aquest s'encarregarà d'accedir a la part de codi de l'itinerari corresponent a l'agència actual. A continuació s'executen els components afegits per l'usuari i finalment s'executarà el component de migració.

2.7 FIPA

FIPA [FIPA] (Foundation for Intelligent Physical Agents) és una organització estandaitzadora acceptada al IEEE que promou les tecnologies basades en agents i la interoperabilitat dels seus estàndards amb altres tecnologies. Dels estàndards de FIPA ens centrarem en dos que especifiquen els mecanismes de comunicació entre agents: els missatges ACL i les ontologies.

2.7.1 ACL

ACL prové de l'acrònim Agent Communication Language. Aquest estàndard defineix el format dels missatges que s'intercanvien entre els agents. El format del missatge és el següent:

Performative: És l'únic camp de tipus obligatori. Especifica el tipus d'acte comunicatiu.

Sender: Identifica l'emissor del missatge.

Receiver: Identifica els receptors del missatge.

Reply-to: Identifica l'agent a qui ha d'anar dirigida la resposta a un cert missatge.

Content: Contingut del missatge.

Language: Especifica el llenguatge amb el qual ha estat expressat el contingut del missatge.

Encoding: Especifica la codificació del contingut del missatge.

Ontology: Especifica les ontologies utilitzades per classificar el contingut del missatge.

Protocol: Especifica el nom del protocol imbricat en l'acte comunicatiu.

Conversation-Id: Especifica l'identificador únic de la conversa.

Reply-with: Expressió que indica l'expressió utilitzada per respondre al missatge.

In-reply-to: Expressió que especifica la resposta al missatge amb el mateix contingut al camp reply-with.

Reply-by: Especifica el termini en el qual serà acceptada la resposta al missatge.

2.7.2 Ontologies

Sovint una habilitat molt comuna entre els agents és la capacitat de comunicar-se i interactuar entre ells per tal de col·laborar en la resolució d'un o varis problemes. La comunicació entre agents, però, no és un problema trivial. Al llarg d'aquesta secció veurem quina és la problemàtica i quins mecanismes estandarditza FIPA per solucionar-la.

Els elements bàsics en un procés comunicatiu son l'emissor, el receptor, el canal, el llenguatge i una forma de representar el coneixement. A més, s'ha de tenir present que tot procés comunicatiu té un objectiu. En el nostre cas, un agent pot actuar d'emissor i receptor alhora. Podem considerar el canal com la xarxa de comunicació sobre la qual circulen els missatges de forma transparent al programador. La plataforma sobre la qual treballarem ens proporciona els mecanismes de comunicació necessaris per accedir al canal. No obstant, però ens manquen encara uns últims elements fonamentals per dur a terme el procés comunicatiu: el llenguatge i la representació del coneixement.

Sembla evident que si dos agents no parlen el mateix llenguatge no és podran comunicar. El llenguatge estableix els símbols atòmics presents en l'acte comunicatiu i les regles per la construcció de conjunts de símbols amb significat propi. No obstant, pot no ser suficient compartir un mateix llenguatge tant si emissor com receptor no comparteixen una base comuna de coneixement i una forma comuna de representar-ho. Podem definir "ontologia" com un model de dades que representa un conjunt de conceptes (definits per símbols) i la seva relació dins d'un domini.

Quan dos agents es comuniquen poden fer-ho mitjançant missatges simples de text. Si l'agent A vol enviar a l'agent B informació sobre la persona P haurà de codificar en una cadena de caràcters el seu nom, dni, telèfon, etc. L'agent B en rebre el missatge haurà d'analitzar la cadena i classificar la informació continguda per poder-la fer servir. Son evidents les limitacions d'aquest mètode a mesura que la mida dels missatges creix i fa referència a entitats més complexes.

En l'acte comunicatiu entre agents podem distingir entre termes i predicats. Un predicat és una expressió que dona informació veritable o falsa sobre l'estat del món. Un terme és una expressió que identifica entitats que existeixen al món. Els termes es poden classificar en:

Conceptes: Expressions que representen entitats amb una estructura complexa.
Accions d'agent: Accions que poden dur a terme certs agents sobre una entitat.
Primitives: Expressions que indiquen entitats atòmiques.
Agregacions: Grups d'entitats.

Expressions identificatives referencials: Expressions que identifiquen el conjunt d'entitats que compleixen una certa característica.

Variables: Expressions que identifiquen un element genèric no conegut a priori.

En el fons, una ontologia és un model de representació d'informació compartit entre els agents implicats en un acte comunicatiu. A més baix nivell podem definir ontologia com una col·lecció de predicats, conceptes i accions d'agent.

Veurem ara com les ontologies faciliten l'acte comunicatiu entre agents reprenent l'exemple anterior. Abans veiem dos agents que compartien informació explícitament col·locada en cadenes de caràcters. Imaginem ara que els agents A i B comparteixen la mateixa manera de representar la informació sobre les persones en general. Així doncs, si l'agent A vol enviar informació sobre la persona P a l'agent B només cal que A expressi la informació de la persona en el format de dades que comparteix amb B. El tractament de la informació per part de B pot ser directe ja que l'ha rebut classificada.

FIPA estableix que la formà estàndard de classificar la informació continguda en un missatge ACL és fent ús de ontologies. A més, es defineixen unes primitives bàsiques per representar la informació mitjançant cadenes de caràcters (*strings*), nombres sencers (*integers*), nombres en punt flotant (*floats*), dates, booleans i seqüències de bytes.

Classificant informació representada amb les primitives anteriors i fent ús d'ontologies es poden definir entitats complexes. A més, habilita la comunicació entre agents desenvolupats per diferents plataformes de desenvolupament, per exemple Java i C++. Tota comunicació que fa servir primitives no mencionades anteriorment queda fora dels estàndards de FIPA.

En aquest capítol s'han presentat tots els elements que conformen l'entorn on s'emmarca el projecte. Al següent capítol es presenta l'etapa d'anàlisi on s'identificaran els mòduls en que es divideix el treball i es presentaran els seus requeriments a través d'un petit anàlisi.

Capítol 3

Anàlisi.

Al llarg d'aquest capítol es mostraran els diferents mòduls que formen el projecte al mateix temps que es desenvolupa un estudi dels seus requisits. L'estudi començarà exposant els requisits inherents a l'agent mòbil argumentant la seva raó de ser. Durant aquesta secció apareixerà la necessitat de l'agent de suport. Acte seguit s'analitzarà l'agent de suport per a continuació, estudiar els requisits en les comunicacions entre agents i/o components. Per últim, amb l'estudi dels components de mostra, que es desenvoluparan en el projecte, es tancarà aquest capítol.

Cal tenir present que aquest projecte s'emmarca dins de l'esquema de desenvolupament d'agents del grup SeNDA. Un dels objectius del projecte és realitzar la integració amb la interfície de creació de l'itinerari (IDT) i el constructor d'agents (*Agent Builder*). Aquest fet es pot considerar un restricció implícita tant d'anàlisi com de disseny.

3.1 L'agent mòbil

L'agent mòbil és l'actor principal del paradigma de desenvolupament que ens concerneix. Com s'ha comentat prèviament l'enfocament monolític del problema de disseny d'un agent presenta greus problemes si es valora l'esforç dedicat a implementar les tasques secundàries de l'agent. Al capítol anterior hem vist, també,

que Jade proposa l'estructura d'un agent com una classe Agent principal i una serie de *behaviours* que defineixen la funcionalitat de l'agent. La idea bàsica de construcció d'agents per components és que els components siguin independents de l'agent en quant a la seva funcionalitat, es a dir; no podem inserir mètodes al cos de l'agent per cada component nou que dissenyem. No obstant això, si que podem considerar que l'agent ha de proporcionar serveis als seus *behaviours* com a mecanisme per evitar replicar codi a cada component. Un clar exemple son les mecanismes de comunicació amb l'usuari.

Funcions:

S'ha comentat que els components son funcionalitats afegides a un agent mòbil. No obstant, poden haver components que requereixen establir una acte comunicatiu amb l'usuari per dur a terme els seus objectius. Un clar exemple podria ser un component que indica la posició de l'agent mòbil a l'usuari després de cada migració. A [MR] es proposa el concepte d'agent de suport. L'agent de suport és un agent estàtic que s'executa en un entorn controlat per l'usuari i que fa d'interfície entre aquest i l'agent mòbil. Cal aclarir que tot i que la comunicació entre components es realitza d'agent a agent, hi ha mecanismes per derivar l'entrega dels missatges al seu corresponent component. Un cop introduït el marc de treball passem a analitzar tots els mòduls de l'esquema proposat.

En temps de construcció, l'agent mòbil ha de rebre les dades necessàries per comunicar-se amb l'usuari. S'ha considerat que ha de ser el propi agent qui ha de posseir aquesta informació per evitar tenir-la replicada per cada component que la necessiti. La comunicació amb l'usuari es pot establir mitjançant l'agent de suport o via *e-mail*. Depenent dels components que executi l'agent pot ser necessària la localització de l'agent de suport o bé l'adreça de correu electrònic del propietari, o ambdues informacions alhora.

Una altra dada important que ha de posseir l'agent és la clau publica de l'usuari en cas de que es vulgui fer servir per cobrir certs aspectes de confidencialitat. Com a últim apunt, s'ha de tenir present que un futur aquestes dades podrien ser accedides a través d'un servei de directoris.

Ja que l'agent mòbil conté les dades necessàries per establir comunicacions amb l'usuari, un altre requisit és que hauria de fer d'interfície entre els components i el sistema de comunicacions. Aquest requisit té com objectiu evitar replicar codi a cada component. La informació del destinatari de la comunicació roman a la classe agent. L'agent mòbil hauria de ser capaç de rebre el missatge destinar a ser enviat i completar-lo amb la informació restant per l'entrega de forma transparent als components.

El servei de criptografia, presentat al capítol 2, ofereix a l'agent mecanismes per l'autenticació i la confidencialitat de les comunicacions mitjançant signatura digital i xifratge. Tot i que l'accés al servei de criptografia es pot fer des de cada component sense cap problema, la part de signatura guarda una relació molt especial amb la classe agent. A cada agent, sota demanda, se li proporciona una clau diferent a cada plataforma que li servirà per fer ús dels mecanismes de signatura digital que proporciona el servei de criptografia. Per tal de mantenir la relació agent-clau, l'agent mòbil hauria de fer d'interfície entre els components i el servei de criptografia oferint mecanismes de signatura i xifratge.

L'ordre d'execució dels components és un factor important pel correcte funcionament de l'agent mòbil. Si, per exemple, s'executés abans un component que fa migrar l'agent que el *behaviour* d'usuari pensat per ser executat en una certa agència el resultat podria ser catastròfic. Per tant, l'agent mòbil hauria de proporcionar els mecanismes per poder garantir l'ordre correcte d'execució de tots els *behaviours*, entenent com a *behaviours* els components o les tasques creades per dur a terme l'objectiu de l'agent mòbil. L'establiment de l'ordre d'execució dels *behaviours* ve donat, en ultima instància, per la informació proporcionada per l'usuari en temps de construcció de l'agent. L'agent mòbil no ha de planificar l'execució dels *behaviours*, sinó que els ha d'executar estrictament en l'ordre que li proporciona el *Agent Builder*.

Els *behaviours* que s'executen a cada agència no son sempre els mateixos. Una característica desitjable a l'agent mòbil és que fes de portador de resultats per tal de que aquests es puguin utilitzar en agències posteriors. Per tant, l'agent hauria de presentar mecanismes per l'emmagatzemament i recuperació de dades.

Restriccions:

L'única restricció de l'agent mòbil ve donada en termes d'escalabilitat de la classe. La classe agent hauria de proporcionar únicament els serveis fonamentals que requereixen els components. L'objectiu de proporcionar aquests serveis és evitar replicar codi a cada component i garantir l'accés al servei de criptografia. En cap cas s'afegiran funcionalitats a la classe agent que realitzin total o parcialment la tasca destinada a un component és a dir; la classe agent no pot proporcionar mecanismes, per exemple, de seguiment de l'agent ja que aquesta tasca estarà destinada a un component. En cas contrari es podria caure en l'error de haver de modificar la classe de l'agent per cada component nou que es generi.

Atributs de seguretat

L'agent mòbil hauria de ser capaç d'enviar missatges de forma confidencial cap a l'agent de suport. Per la seva condició de mòbil, sofreix la impossibilitat de portar una clau privada. D'altra manera, el secret d'aquesta podria quedar exposat en arribar l'agent a una plataforma maliciosa. En conseqüència, la confidencialitat en la transmissió només es pot garantir en els missatges que van des de l'agent mòbil cap a l'agent de suport. Per tant, l'agent mòbil hauria de proporcionar mecanismes transparents als components per l'enviament de missatges de forma confidencial cap a l'agent de suport.

L'accés a la part d'autenticació del servei criptogràfic requereix que l'agent mòbil faci una petició al servei que li retornarà una clau. Per tant, l'agent mòbil hauria de vetllar que a cada plataforma es gaudís d'accés total al servei criptogràfic per part dels seus components. D'aquesta manera els components podrien establir comunicacions autèntiques amb l'agent de suport.

3.2 L'agent de suport

L'agent de suport s'executa en un entorn controlat per l'usuari. La seva feina principal és la de fer d'interfície entre l'usuari i l'agent mòbil. Per la seva condició

d'agent estàtic en entorn controlat rep el benefici de poder posseir una parella de claus que l'habiliten per rebre comunicacions confidencials provinents de l'agent mòbil.

En ocasions, aquest tipus d'agent ha esta catalogat com agent de control. De fet, la tendència actual és abandonar el qualificatiu "control" per adoptar el qualificatiu "suport". La raó de base és que es pretén evitar caure el l'equivoc de pensar que l'agent de suport realitza tasques inherents a l'agent mòbil. L'agent de suport realitza tasques auxiliars a l'agent mòbil però no condiciona la seva execució.

Funcions:

L'agent de suport inclourà la part complementaria dels components que contingui l'agent mòbil. Els *behaviours* que s'executen a l'agent de suport son de naturalesa passiva, és a dir; no inicien comunicacions cap a l'agent mòbil sinó que esperen a rebre-les, tot i que poden haver excepcions. Aquesta característica respon al fet de que l'agent de suport, per norma general, no coneix la localització de l'agent mòbil en cada instant de temps. Per això, sovint l'única manera de comunicar-se amb l'agent mòbil és esperar a rebre un missatge d'ell i limitar-se a respondre a l'origen de la comunicació. Aquest fet té una implicació immediata: mentre que l'agent mòbil executa els components seqüencialment, l'agent de suport no coneix l'ordre en que les comunicacions li arribaran a ell. Per tant l'agent de suport hauria d'executar els components en paral·lel planificant, d'aquesta manera, les comunicacions amb l'agent mòbil. Un altre esquema de comunicació possible és que l'agent de suport escanegi un conjunt d'agències enviant missatges ACL amb l'esperança de rebre resposta per part de l'agent mòbil. Aquesta situació, tot i que possible, és del tot inusual i es donarà en contades o nul·les ocasions.

Una característica desitjable a l'agent de suport és la de rebre comunicacions autentiques i secretes per part de l'agent mòbil. Si abans hem vist que l'agent mòbil pot establir comunicacions xifrades i autentiques cap a l'agent de suport, ara hem de veure com l'agent de suport ha de tractar aquestes comunicacions. Pel que fa a la confidencialitat, l'agent de suport ha de posseir una parella de claus asimètriques per poder signar i desxifrar missatges. Des del punt de vista de

l'autenticitat, l'agent de suport hauria de poder verificar, si s'escau, l'origen del missatge tenint sempre present que l'agent mòbil no pot portar cap secret a sobre, per tant, no pot posseir cap clau privada.

Un dels objectius de l'agent de suport és fer d'interfície entre l'agent mòbil i l'usuari. Poden haver components que reben informació de l'agent mòbil dirigida cap a l'usuari o simplement han d'informar sobre l'estat propi de la seva execució. L'agent de suport ha d'habilitar la recepció dels missatges per part del propietari de l'agent.

Restriccions:

Derivat del problema que suposa que la plataforma té control total sobre l'entorn d'execució dels agents que alberga, l'entorn d'execució de l'agent de suport ha d'estar controlat per l'usuari. D'aquesta manera l'agent de suport pot posseir un parell de claus asimètriques sense que quedi exposada la privadesa de la clau secreta. La implicació més immediata és que l'agent de suport ha de ser estàtic.

El temps de vida de l'agent de suport ve determinat pel temps de vida de l'agent mòbil. En general, l'agent de suport hauria de tenir un mecanisme per detectar la finalització de l'execució de l'agent mòbil i finalitzar, així, amb la seva execució.

Atributs de seguretat

L'agent de suport pot realitzar tasques molt importants per l'execució global d'una aplicació basada en agents. Per tant, comprovar l'autenticitat de l'agent mòbil per evitar atacs d'impersonació és vital. No obstant, aquest no és un problema trivial de resoldre si l'agent mòbil no pot posseir cap secret. Per solucionar el problema, el servei de criptografia ha de ser capaç de generar informació sobre els agents que s'executen a cada plataforma. Més endavant veurem com funciona aquest mecanisme. Ara només ens hem de quedar amb la idea de que si la informació que genera el servei de criptografia sobre l'agent mòbil és coneguda per endavant pel nostre agent de suport, podrem constatar l'origen de la comunicació amb un senzill protocol.

3.3 ComponentOnto: L'ontologia dels components

A l'hora de dissenyar una ontologia per als components sorgeixen diferents possibilitats. Per una banda existeix la possibilitat de fer una ontologia general per a tots els components. Aquesta ontologia especificarà la classificació de la informació necessària en l'acte comunicatiu entre dos components. Un camp contenidor del nom del component dins de la ontologia seria perfecte per tal de poder planificar la comunicació simultània entre els diferents components. Però el problema és més complicat. Definir una ontologia prou general per tots els components i que sigui la òptima en cada cas és poc menys que impossible. Aquest problema rau del fet que avui dia hi ha planificats i estudiats dos, com a molt tres, components que han de fer servir aquesta ontologia, però en un futur potser n'hi hauran milers, cadascú amb unes característiques molt diferents. És obvi doncs que preveient aquest fet l'ontologia fruit d'aquest estudi ha de ser prou general. No obstant, aquest no és l'únic problema, ni el més greu. Pensem per un moment en un component que ha de transmetre informació sobre una empresa E que ven un cert producte al millor preu com a resultat de la seva execució. El cas ideal seria que la ontologia de que fan servir en la comunicació classifiqués la informació de la empresa P de forma fàcilment interpretable pels dos agents. Una ontologia general no ho permetria.

Podem pensar en el problema en un nivell més alt d'abstracció. Qui ha de tractar la informació, l'agent de suport o en ultima instància l'usuari? La resposta un cop més és: depèn. Existeix una difusa frontera entre les funcionalitats que han de proporcionar els components i les funcionalitats que han d'implementar els programadors d'agents. En principi, un component ha d'oferir una funcionalitat que es pressuposa comuna en conjunt més o menys gran d'agents i tindria que ser, a més, independent de l'objectiu final de l'agent mòbil. No podem entrar en la creació de components en el domini de les funcionalitats que ofereix un agent. Però això no implica que cap agent de suport no hagi de tractar informació que li ofereix una funcionalitat de l'agent mòbil i actuar en conseqüència. Per exemple, es pot donar un cas d'un component que tracta certa informació que ofereix l'agent mòbil i calcula estadístiques sobre el temps d'execució sobre cada plataforma.

En aquest últim cas sembla justificat la creació d'una ontologia especifica per

cada component. És precisament aquest l'altre camí que defineix l'itinerari a seguir: la creació d'una ontologia específica o una ontologia feta a mida per cada component. No obstant, cap la possibilitat d'estendre l'ontologia general per adaptar-la al tipus de problema. De fet, l'objectiu d'una ontologia és establir la forma de representar la informació referent al món i cada agent (o component) veu només la part del món que necessita o simplement romanen en mons diferents.

L'alternativa de generar una ontologia especifica per cada component, però, presenta varis problemes importants. El primer lloc fer una ontologia per cada component ralentiria la fase de disseny i implementació de nous components quan en la majoria dels casos segurament hi hauria prou amb una ontologia general. De fet, aquest problema no és tan greu si el comparem amb el problema que suposa en termes de volum que un agent porti a sobre N components amb N ontologies diferents.

En aquesta secció s'exposaran els requisits d'anàlisi d'una ontologia general. Aquesta ontologia rep el nom de ComponentOnto i pretén ser una ontologia comuna a gran part dels components. És per això que ha de ser de caire generalista per cobrir un ampli ventall d'escenaris possibles en la comunicació dels components.

Funcions:

L'ontologia hauria de contenir informació sobre el component que l'està fent servir. D'aquesta manera es podria decidir cap a quin component va dirigit cada missatge. És bàsic que cada missatge contingui informació sobre el nom del component així com informació sobre la seva versió. D'aquesta manera no tan sols s'habilitaria l'entrega de cada missatge al seu component sinó que a més es podria fer un control de versions per assegurar la compatibilitat.

Un dels objectius principals dels components és proporcionar informació sobre l'execució a l'usuari. Per tant, l'ontologia hauria de possibilitar la transmissió d'un missatge en llenguatge natural. Típicament aquesta funcionalitat es farà servir quan l'agent de suport rep informació que l'únic tractament que ha de rebre és la de ser mostrada a l'usuari.

Poden ser habituals els escenaris on l'intercanvi d'informació entre els agents

faci referència a una entitat complexa. Per tant, l'ontologia hauria de possibilitar la transmissió d'un objecte java serialitzat. D'aquesta manera podríem fer servir l'ontologia per classificar informació de tota mena sempre i quan tant l'emissor com el receptor intercanvien objectes de classes comunes entre ells.

Restriccions:

És molt important diferenciar entre classificar la informació mitjançant ontologies o mitjançant objectes java. Tot i que en un principi ofereixen semblants possibilitats, l'ús d'ontologies permet la comunicació entre agents creats amb diferents llenguatges de programació mentre que l'intercanvi d'objectes impossibilita aquest fet. Per tant, una comunicació amb intercanvis d'objectes es surt fora del estàndards de FIPA. S'ha de ser conscient en tot moment d'aquest fet: mentre el missatge no contingui cap objecte la comunicació complirà els estàndards. No obstant, pot ser imprescindible fer servir aquesta funcionalitat.

Aspectes de seguretat:

L'ontologia dels components no hauria de cobrir cap aspecte de seguretat. En tot cas la seguretat s'hauria d'implementar a un altre nivell.

3.4 Comunicacions segures

Hi ha casos en els que pot interessar que les comunicacions entre agents es facin de forma confidencial. No obstant, Jade no proporciona aquest servei. Cal doncs trobar un esquema que ho possibiliti. A [GND03] es proposa embolicar un missatge ACL de manera que quedi visible la informació necessària per al transport a més de la informació sobre la seguretat del contingut. Un camp adjacent ha de presentar el contingut del missatge de manera xifrada.

Funcions:

L'esquema hauria de permetre l'enviament de missatges tant signats com xifrats. A més, hauria de ser prou genèrica per suportar un ventall ampli d'esquemes de seguretat. La idea d'aquest mòdul és que es pugui fer servir no només en les comunicacions entre components sinó en qualsevol tipus de comunicació entre agents.

Restriccions:

Aquest esquema de comunicacions segures pretén ser prou ampli com per poder ser utilitzat més enllà de l'àmbit d'aquest treball. Els estàndards de FIPA no tan sols es refereixen comunicacions entre agents pensats i implementats per ser executats sota la mateixa plataforma de desenvolupament. Cal, doncs, que el disseny respecti les restriccions que proposa FIPA, és a dir; no transmetre informació en un format depenent de la plataforma.

No és un dels objectius d'aquest projecte modificar els serveis d'enviament, transport i recepció de missatges de Jade. L'esquema de seguretat hauria cenyir-se a les funcionalitats que presenta la plataforma Jade en el moment d'anàlisi d'aquest projecte.

3.5 Els components

Fins ara hem estat parlant de components de manera genèrica. En aquesta secció és pretén fer un anàlisi més acurat sobre els components i les seves característiques.

Es poden diferenciar dos classes de components: els components simples i els components compostos. Els components simples son mòduls que s'afegeixen únicament a l'agent mòbil i que no requereixen cap mòdul complementari a l'agent de suport. Els components compostos, en canvi, requereixen d'un mòdul que s'afegeix a l'agent mòbil i un altre en l'agent de suport. A partir d'ara anomenarem component d'acció a tots aquells components o parts de components compostos

que s'executin a l'agent mòbil i components de suport a totes aquelles parts de components compostos que s'executen a l'agent de suport.

3.5.1 Requisits globals

S'ha comentat prèviament que a Jade la comunicació s'efectua d'agent a agent però hi ha mecanismes que ens permetran "redirigir" els missatges cap als components que es consideri oportú. És de vital importància que cada component vetlli per la recepció única i exclusivament dels seus missatges. Un component que rep missatges d'un altre component pot bloquejar l'execució tant de l'agent mòbil com de l'agent de suport. Així doncs, un requisit global a tots els components és que haurien de filtrar les seves comunicacions.

Imaginem per un moment que l'agent mòbil porta hores d'execució realitzant una feina qualsevol i a l'hora d'executar un component, aquest falla i es perd tota la feina realitzada. Aquesta situació és intolerable. En especial, és l'agent mòbil qui és més sensible al bloqueig d'un component. Mentre que a l'agent de suport els components s'executen de forma concurrent, a l'agent mòbil per norma general ho fan de forma seqüencial. Si un component entra en un estat bloquejant, l'execució de l'agent mòbil en general pot quedar aturada. Cal doncs que els components siguin dissenyats amb la màxima garantia de que finalitzaran la seva execució.

3.5.2 El component de Tracking

Fins ara, si un usuari volia saber la posició de l'agent mòbil que havia llançat tenia dos opcions. La primera opció era controlar totes les agències per les quals passarà l'agent. La segona opció era programar una funcionalitat que informés a l'usuari sobre la seva posició. Desconèixer la posició d'un agent mòbil complica greument la tasca de seguiment, control, i depuració d'errors de l'agent. Si aquest falla per qualsevol motiu no sabrem en quin punt de l'itinerari ho ha fet. El component de *tracking* és un component compost pensat per cobrir aquesta mancança, informant de forma automàtica sobre la posició de l'agent mòbil immediatament després de

cada migració. *Tracking* significa seguiment en anglès.

Requisits

El component de *tracking* és un component compost. La part d'acció, que s'executa a l'agent mòbil, ha d'informar, de manera autèntica, sobre el nom de l'agent, la plataforma on es troba i l'instant de temps actual. La part complementaria, que s'executa a l'agent de suport, ha de rebre la informació, mostrar-la a l'usuari i confirmar la recepció a l'agent mòbil. A més, ha de verificar en tot moment la veracitat de l'emissor de les dades.

Restriccions:

L'entorn d'execució dels agent no té, ni molt menys, per que ser totalment fiable. Cal que el component d'acció previngui aquesta situació acotant el temps d'espera de per la recepció dels missatges de confirmació de l'agent de suport. Els temps de marge han de ser raonables movent-se dins d'uns marges lògics.

De la mateixa manera que amb el temps de *timeout*, el nombre d'intents en la comunicació amb l'agent de suport ha de estar acotat. Si l'agent de suport no respon a un nombre raonable d'intents de comunicació, el component ha de finalitzar la seva execució per tal de l'execució de l'agent mòbil pugui continuar.

Atributs de seguretat

Tant el component d'acció com el component de control s'han de autenticar mútuament com a pas previ per acceptar els missatges que s'intercanvien. En cas contrari el component quedaria exposat a atacs d'impersonació que podrien falsejar la informació final mostrada a l'usuari.

3.5.3 El component Data retrieval

En l'actualitat, un agent mòbil pot executar-se i generar resultats. No obstant, no hi ha cap mecanisme automàtic que li permeti entregar les dades directament al

propietari sense fer migrar a l'agent a una agència controlada per ell. El component data retrieval pretén cobrir aquesta mancança oferint un mecanisme d'entrega automàtica de resultats a l'usuari presentant, a més, garanties d'autenticitat i confidencialitat.

Requisits

Un primer requisit és que el component hauria de fer de contenidor dels resultats generats a cada plataforma. D'aquesta menera permetria emmagatzemar de forma ordenada els resultats que es generen a cada plataforma.

L'usuari hauria de poder especificar quan vol que els resultats parcials siguin enviats al llarg de l'itinerari. D'aquesta manera es podrien fer copies de recolzament dels resultats parcials durant l'itinerari. Per tant, un requisit és que l'usuari, sota demanda, ha de ser capaç d'iniciar el procés de transmissió de dades.

No obstant, tot i que l'usuari té la potestat d'especificar els punts on les dades li seran enviades, cal un mecanisme automàtic per tal de que a l'ultima plataforma de l'itinerari s'iniciï el procés de lliurament. D'aquesta manera hi haurà garanties de que les dades son, com a mínim, enviades un cop.

Restriccions:

Poden haver escenaris on l'usuari requereix desconnexió de la xarxa. Sobretot si el temps d'execució de l'agent s'estima que sigui molt elevat. Això implica que no només les dades han de poder ser lliurades a través de l'agent de suport sinó que s'ha de contemplar la possibilitat de lliurar-les via correu electrònic.

El nombre d'intents de transmissió de les dades així com el temps d'espera per la recepció de la confirmació haurien d'estar acotats. D'aquesta manera s'evita bloquejar l'execució de l'agent si es produeixen errors de comunicació.

Atributs de seguretat

Permetre l'entrega de les dades de forma segura és un dels requisits de seguretat bàsics d'aquest component. Aquest hauria de poder entregar les dades de forma

segura per tal de garantir la confidencialitat en la transmissió.

Un problema greu, que presenta aquest component, és que una agència maliciosa pot alterar els resultats generats a punts de l'itinerari anteriors a ella. El component hauria de presentar un mecanisme que permetés la verificació del conjunt de dades en general per tal de certificar la seva autenticitat i integritat.

Arribats a aquest punt tanquem el capítol d'anàlisi on s'han mostrat totes les parts que conformen el projecte i quins son els seus requisits. Al capítol següent es mostrarà com s'han materialitzat aquests requisits a través dels detalls de disseny i implementació.

Capítol 4

Disseny i implementació.

Un cop analitzades les parts que formen el projecte així com els seu requisits, ens endinsem en la fase de disseny i implementació. Al llarg d'aquest capítol veurem quines solucions s'han proposat per cobrir les necessitats extretes en la fase de disseny i els detalls de la seva implementació. Val a dir que aquest capítol és de naturalesa pràctica.

Si al capítol d'anàlisi s'ha començat veient els requisits de l'agent mòbil i de suport per després analitzar les comunicacions i els components, en aquesta secció es mostrarà primer la fase de disseny i implementació de tota la part que concerneix a la comunicació entre agents per continuar amb l'agent mòbil i de suport i es finalitzarà amb els components.

4.1 L'ontologia dels components: ComponentOnto

El disseny d'una ontologia a Jade es compon de predicats que agrupen conceptes i/o accions d'agent. Els predicats contenen informació veritable o falsa sobre l'estat del món. Els conceptes son expressions que representen una entitat amb una estructura complexa, mentre que les accions especifiquen actes que realitzen agents sobre una entitat del món. A l'ontologia dels components es faran servir només conceptes i predicats ja que pretén ser de caire generalista. Les accions d'agent queden reservades per ontologies utilitzades en tasques més especifiques

com ara una ontologia pel comerç electrònic on es donen situacions, per exemple, on un agent compra un llibre.

L'ontologia ComponentOnto és una ontologia general que faran servir gran part dels components. A l'anàlisi de requisits s'ha exposat la necessitat de que l'ontologia dels components contingués informació respecte al component que l'està fent servir. Per tal de cobrir aquesta necessitat l'ontologia ha presentar el següent concepte:

ComponentDescription		Obligatori	
Nom	**Tipus**	**Presencia**	**Descripció**
Name	String	Obligatori	Camp que especifica el nom del component.
Version	Integer	Obligatori	Nombre sencer que especifica la versió del component.

Un altre requisit extret de la fase d'anàlisi és que l'ontologia dels components ha de donar suport a la transmissió d'un missatge de text. D'aquesta manera es podrà comunicar informació directament a l'usuari. A part del propi missatge, s'ha considerat que el concepte ha de contenir camps auxiliars que especifiquin el nombre de seqüència del missatge i la data en la qual va ser emès. Per cobrir aquesta necessitat es presenta el següent concepte:

MessageContainer		Opcional	
Nom	**Tipus**	**Presencia**	**Descripció**
Message	String	Obligatori	String que conté el missatge.
Seq. num.	Integer	Opcional	Nombre sencer que especifica el numero de seqüència del missatge.
Timestamp	String	Opcional	Camp que especifica la data de creació del missatge.

Per últim, a l'anàlisi de requisits, s'ha citat la necessitat de que l'ontologia suportés la transmissió d'un objecte Java entre dos agents. Per cobrir aquesta ne-

cessitat l'ontologia ha presentar el següent concepte:

ObjectContainer		Opcional	
Nom	**Tipus**	**Presencia**	**Descripció**
Name	String	Obligatori	Camp que conté el nom de l'objecte enviat.
Object	String	Obligatori	String codificat en Base64 que conté l'objecte a ser enviat.
Class	String	Opcional	Camp que conté el nom de la classe de l'objecte
Hash	Integer	Opcional	Sencer que conté el hash de l'objecte a ser enviat.

Val a dir que el fet de fer ús d'aquest últim concepte vulnera els estàndards de FIPA per la comunicació entre agents. Imaginem que un agent construït en Java envia un missatge d'aquesta mena a un agent desenvolupat sota l'entorn C++. Aquest últim no serà capaç d'entendre el contingut. Per contra, és aquest l'únic mecanisme actual que permet l'autenticació dels agent mòbils com veurem més endavant.

Per possibilitar l'aparició dels anteriors conceptes dins d'un missatge ACL és obligatòria la construcció d'un predicat que els agrupi. El predicat "ComponentPred" conté un atribut de cadascun dels conceptes comentats anteriorment.

L'últim pas per la construcció de l'ontologia és definir la classe ComponentOnto. La classe defineix el vocabulari de l'ontologia i la seva estructura. Aquesta informació és necessària per que un *parser* ompli el contingut del missatge ACL amb els camps de l'ontologia així com, a posteriori, sigui capaç d'extreure'ls. Un altre aspecte de la classe ComponentOnto és que ha estat implementada sota el patró de disseny *Singleton* que garanteix una única instància de la classe ComponentOnto carregada a memòria. D'aquesta manera es redueixen els recursos de memòria necessàries per l'execució de l'agent. La figura 4.1 mostra el diagrama de classes que conforma l'ontologia.

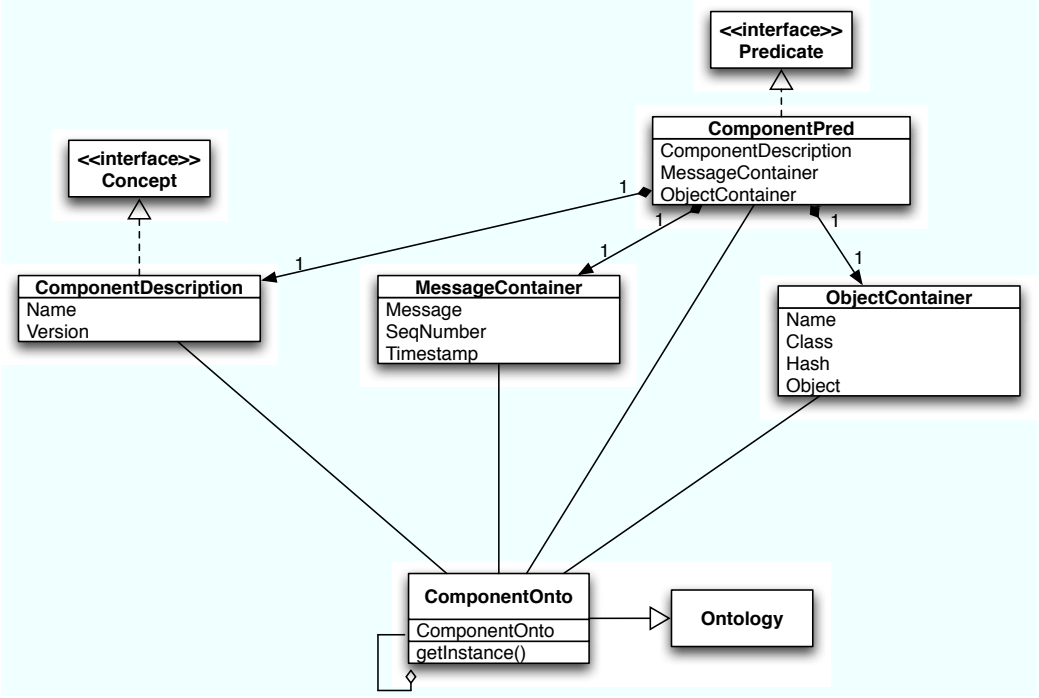

Figura 4.1: Diagrama de classes de l'ontologia ComponentOnto

4.2 Comunicacions segures

Quan parlem de seguretat en les comunicacions entre agents mòbils ens referim a aspectes de confidencialitat i autenticació. La confidencialitat tracta de garantir que les dades que s'intercanvien dos o més entitats es transfereixin de manera que qualsevol entitat aliena a la comunicació no sigui capaç d'interpretar-les. Per aconseguir aquest objectiu cal xifrar les dades com a pas previ a la transmissió. Els aspectes d'autenticació tracten de garantir que les entitats involucrades en un acte comunicatiu quedin identificades. D'aquesta manera s'eviten els atacs d'impersonació on una entitat suplanta la identitat d'una altra. Si l'esquema d'autenticació és prou robust, el sistema gaudeix del beneficis del no repudi: una entitat identificada que realitza una acció es podrà desdir al·legant que no ha estat ella qui realment va realitzar l'acció. En aquesta secció veurem com es poden cobrir els aspectes de confidencialitat i autenticació en les comunicacions entre agents mòbils.

4.2.1 Confidencialitat

L'esquema proposat en aquesta secció per tal de dur a terme comunicacions segures ha estat batejat sota el nom Acl in Acl. La idea bàsica és encapsular un missatge ACL de manera xifrada dins d'un altre missatge ACL de manera que el missatge interior contindrà la informació que vol comunicar l'emissor mentre que el missatge exterior contindrà la informació bàsica per al lliurament i extracció de la informació xifrada. L'esquema es pot observar a la figura 4.2. La princi-

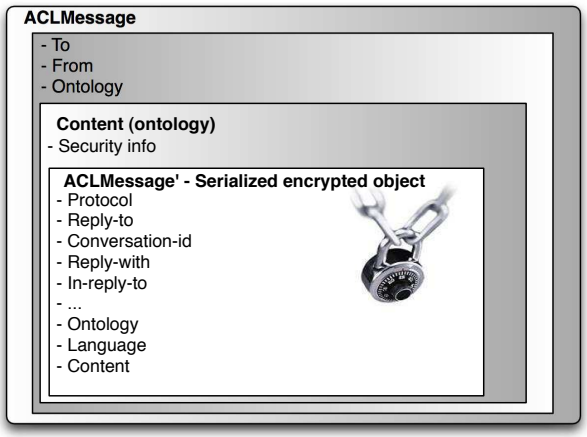

Figura 4.2: Esquema ACL in ACL

pal funcionalitat d'aquest esquema és oferir un mecanisme transparent a l'hora de protegir les dades; els components construiran un missatge ACL utilitzant els procediments habituals i en última instància es decidirà si s'envia de forma normal o de forma xifrada. El missatge intern pot ser qualsevol missatge ACL, amb ontologies o sense. El missatge embolcall, però, ha de fer ús de l'ontologia SecureAciInAclOnto.

L'ontologia SecureAclInAclOnto presenta el següent concepte:

SecurityInfo	Obligatori		
Nom	**Tipus**	**Presencia**	**Descripció**
Algorithm	String	Obligatori	Paràmetre que especifica l'algorisme que s'ha fet servir per xifrar el missatge intern.
Key	String	Opcional	Conté informació corresponent a una clau, per exemple, una clau pública.
Certificate	String	Opcional	Camp que conté un certificat.
KeyRef	String	Opcional	String que conté referències a una clau si aquesta no ha estat inclosa per qüestions d'eficiència.
Data	String	Opcional	Paràmetre que pot contenir informació genèrica de qualsevol tipus.
Parameters	String	Opcional	Camps que conté informació sobre paràmetres que poden ser requerits en el sistema de xifratge.
Acl	String	Opcional	String que conté un missatge ACL xifrat. La representació a String del missatge ACL original s'ha obtingut a partir de la classe StringAclCodec que permet la conversió d'un missatge ACL a String i viceversa.

AclInAclPred és un predicat que conté el concepte SecurityInfo.

La classe SecureACLinACLOnto és la classe que defineix l'ontologia i el seu vocabulari. De la mateixa manera que l'ontologia dels components, la classe SecureACLinACLOnto implementa el patró Singleton. La Figura 4.3 mostra el diagrama de classes que conforma l'ontologia.

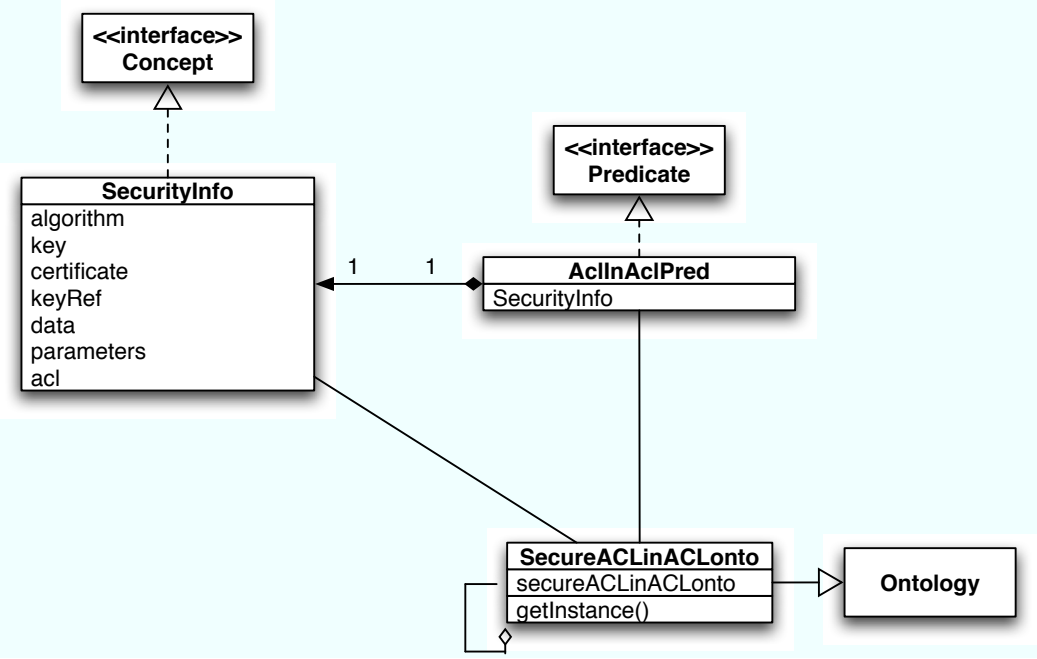

Figura 4.3: Diagrama de classes de l'ontologia SecureAclInAclOnto

4.2.2 Autenticació

Els aspectes d'autenticació entre l'agent mòbil i l'agent de suport es basen en signatures, però podem trobar dos casos ben diferenciats: l'autenticació de l'agent de suport vers l'agent mòbil i l'autenticació de l'agent mòbil vers l'agent de suport.

L'autenticació de l'agent de suport segueix un esquema d'autenticació clàssica. L'agent de suport posseïx una parella de claus asimètriques; una publica i una privada. Si l'agent de suport xifra unes dades amb la seva clau privada, qualsevol coneixedor de la clau pública de l'agent de suport serà capaç de verificar que ha estat l'agent de suport i només l'agent de suport qui les ha generat. Aquest esquema és robust mentre el secret de la clau privada no sigui compromès. Val a dir que l'esquema presenta robustesa contra els atacs de repudi, es a dir; un agent no es pot desdir de les signatures generades amb la seva clau privada a no ser que la privadesa de la clau hagi estat compromesa.

L'autenticació de l'agent mòbil vers l'agent de suport és un cas especial. L'agent mòbil no pot posseir cap clau privada ja que aquesta pot quedar compromesa

si durant l'itinerari, l'agent es troba amb una plataforma maliciosa. Recordem que les plataformes son l'entorn d'execució dels agents, per tant, poden accedir a tota la informació que l'agent porti a sobre per ser executada en aquella plataforma. Si una plataforma maliciosa capta la clau privada de l'agent mòbil i la difon, els atacs d'impersonació és podran realitzar des de qualsevol punt del sistema. La solució a aquest problema ve donada per l'us del servei de criptografia. Quan un agent vol demanar una signatura, és la plataforma qui fent servir la seva clau privada certifica que aquelles dades han estat creades per un cert agent. La manera de fer-ho és senzilla: quan l'agent demana una signatura, la plataforma crea un objecte CryptoAgentData que conté les dades originals i un *hash* de la classe principal de l'agent mòbil. Acte seguit ho signa fent servir al seva clau pública. D'aquesta última signatura s'extreu un objecte de la classe SignedObject. Val a dir, que aquesta solució no elimina els problemes d'impersonació. Una plataforma maliciosa pot impersonar a l'agent mòbil generant un *hash* fals en les signatures però ara fora d'aquesta plataforma la seguretat no ha estat compromesa. De la mateixa manera l'esquema es també susceptible als atacs de repudi.

En temps de construcció, el *builder* ha de proporcionar a l'agent de suport quin és el *hash* que el servei de criptografia calcularà cada cop que l'agent mòbil demani una signatura. D'aquesta manera, en rebre un objecte signat per l'agent mòbil es podrà comprovar que la signatura d'aquell objecte ha estat demanat per l'agent mòbil en una certa plataforma.

4.3 L'agent mòbil: MyMobileAgent

La classe MyMobileAgent ha de ser executable sota la plataforma Jade, fet que implica que sigui una extensió de la classe Agent. L'accés a la part de signatura del servei de criptografia implica que la classe MyMobileAgent implementi la interfície CryptoAgent.

Un requisit presentat a la fase d'anàlisi és que l'agent mòbil ha de posseir les dades necessàries per tal d'establir comunicacions amb l'usuari. La classe ha de presentar els següents atributs:

1. ownerAddress: És un atribut de tipus *string* que conté la informació de localització de l'agent de suport necessària per establir comunicacions amb ell. Opcionalment aquest atribut pot no contenir informació si cap component dels que posseeix l'agent ho requereix.

2. ownerMail: És un tribut de tipus *string* que conté la informació necessària per comunicar un component amb l'usuari via el seu compte de *e-mail*. Opcionalment aquest atribut pot no contenir informació si cap component dels que posseeix l'agent ho requereix.

3. ownerPublicKey: És de tipus java.security.PublicKey i conté informació sobre la clau pública de l'usuari. Aquest atribut habilitarà el poder establir comunicacions secretes amb l'usuari ja sigui mitjançant l'agent de suport o via *e-mail*.

4. behs: És una llista de *behaviours* que ha d'executar l'agent. En ella s'inclouen tots els components i *behaviours* que fan la feina desitjada per l'usuari.

Un altre requisit presentat a la fase d'anàlisi és que l'agent mòbil ha de ser l'encarregat de enviar missatges cap a l'agent de suport. Per possibilitar-ho ha de presentar el següents mètodes:

1. sendMessage: sendMessage és un mètode que no retorna cap paràmetre. Rep com a argument un missatge ACL i el complerta amb la informació del receptor del missatge. Acte seguit ho envia.

2. sendSecureMessage: sendSecureMessage és un mètode que no retorna cap paràmetre. El seu objectiu és enviar un missatge de forma segura fent servir l'ontologia AclInAclOnto. Rep com a paràmetre d'entrada un missatge ACL creat sense informació de destinació del missatge. Aquest missatge serà el contingut del missatge embolcall. El primer pas que realitza el mètode és passar el contingut del missatge original a una representació en *string* . D'aquesta manera garantim la compatibilitat amb els estàndards de FIPA. Acte seguit, la representació en *string* del missatge original és xifrada fent

servir l'algorisme RSA. Un cop xifrat el contingut es crea l'embolcall amb la informació de xifrat i entrega. Per últim s'envia el missatge.

Un tercer requisit presentat a la fase d'anàlisi és que l'agent mòbil ha d'actuar d'interfície entre els components i el servei de criptografia. Per fer-ho possible ha de presentar el següents mètodes:

1. setCCHkey: Aquest mètode implementa l'interfície CryptoAgent i s'executarà automàticament al demanar al servei la generació de la clau que ens permetrà realitzar signatures des de l'agent. Al executar el mètode generateCCHkey passant com a paràmetre la ID de l'agent mòbil, es crida al mètode setCCHkey que rep de la plataforma la clau de forma signada. El mètode setCCHkey s'encarrega de verificar que la clau ha estat generada per la plataforma on es troba actualment i en cas afirmatiu, la emmagatzema.

2. sign: L'execució d'aquest mètode ha de ser posterior a la generació de la CCHkey. El mètode rep una seqüència de bytes a signar i retorna un objecte de la classe SignedObject. El mètode, en rebre les dades a signar, demana al servei de criptografia que generi la signatura de les dades a partir de la clau CCHkey de l'agent.

L'agent mòbil ha de ser l'encarregat de garantir el correcte ordre d'execució dels components. Per que sigui possible ha d'incloure un atribut de la classe FSMBehaviour. Les sigles provenen de l'expressió Finite State Machine Behaviour. Aquesta classe permet definir l'ordre exacte d'execució de N *behaviours* diferents. La configuració inicial d'aquesta màquina d'estats és farà de forma automàtica processant la llista de *behaviours* que ha proporcionat el *builder*. A més, després de cada migració s'haurà de tornar a reprogramar incloent els nous *behaviours* a ser executats en la nova agència.

L'últim requisit de la fase d'anàlisi estableix que l'agent mòbil ha de fer de portador dels resultats que es generen a cada agència. Aquest requisit queda cobert amb la inclusió dels següents atributs:

1. dades: Atribut de tipus java.Object. Mitjançant aquest atribut es pot realitzar l'intercanvi de qualsevol tipus de dades entre *behaviours* ja siguin

components o *behaviours* de tasca de l'agent.

2. sendFlag: És de tipus booleà. Aquest atribut serveix per indicar que les dades emmagatzemades en l'atribut "dades" han de ser enviades cap a l'usuari.

4.4 L'agent de suport: MySupportingAgent

De la mateixa manera que la classe MyMobileAgent, la classe MySupportingAgent ha de ser executable sota la plataforma Jade. En conseqüència ha de ser una extensió de la classe Agent. L'agent de suport ha de tenir accés al servei de criptografia a través de la implementació de la interfície CryptoAgent.

El primer requisit esmentat de l'agent de suport és que ha d'executar els components de forma concurrent. En conseqüència l'agent de suport ha de posseir l'atribut "behs" que conté una llista de *behaviours* que s'han d'executar de forma concurrent. Per defecte els *behaviours* no s'executen de forma concurrent. Cal passar un a un cada *behaviour* al mètode *wrap* de la classe ThreadedBehaviourFactory per tal de transformar la llista de *behaviours* en *behaviours* que s'executen com un *thread* i de forma paral·lela entre ells.

Un altre requisit presentat a l'anàlisi de requisits és que l'agent de suport ha de ser capaç de rebre comunicacions autèntiques i secretes. Per fer-ho possible ha de posseir els següents atributs:

1. mobileAgentHash. L'atribut mobileAgentHash conté el *hash* de la classe principal de l'agent mòbil. Sempre que l'agent mòbil demani una signatura automàticament es generarà un objecte de la classe CryptoAgentData que contindrà les dades originals i el *hash* de la classe principal de l'agent mòbil com a element d'autenticació. A més, l'objecte generat anirà signat per la plataforma. L'agent de suport ha de conèixer el *hash* que es calcularà de l'agent mòbil per tal de que els seus components puguin verificar l'autenticitat de les comunicacions quan ho requereixin.

2. keys. L'atribut keys és de tipus KeyPair i conté una parella de claus per tal de

poder signar i desxifrar dades. A partir de la clau pública l'agent de suport pot signar dades i autenticar-se en les comunicacions amb l'agent mòbil. De la mateixa manera, la clau privada habilitarà el desxifrat de les dades rebudes a través de comunicacions confidencials mitjançant l'esquema ACL in ACL.

4.5 Components

Arribats a aquest punt, ja tenim tots els elements necessaris per poder endinsar-nos en el disseny i la implementació els components de mostra. Començarem materialitzant els requisits globals de tots el components per després centrar-nos en cada component per separat.

4.5.1 Assolint els requisits globals.

A l'anàlisi de requisits s'han mostrat dos característiques que haurien de complir tots els components. La primera és que els components han de vetllar per tractar només els missatges que son dirigits cap a ell. La segona característica és que els components han d'estar dissenyats de forma que no atemptin contra la fiabilitat dels agent mòbils. Ja que la tolerància a falles és un requisit que s'ha de plasmar de forma diferent component a component, en aquesta secció mostrarem els elements necessaris per que cada component capturi exclusivament els seus missatges.

Filtrar les comunicacions: MessageTemplate

Els emissors i receptors de missatges son els agents, no pas els *behaviours* que hi contenen. De fet, els mètodes d'emissió i recepció de missatges ACL es troben a la classe Agent i son públics per tots els *behaviours*. En aquesta secció ens hem de centrar únicament en la recepció dels missatges. Existeixen dos mètodes que la implementen: *receive()* i *blockingReceive()*. El primer d'ells no atura l'execució de l'agent esperant la recepció d'un missatge mentre que el segon si que ho fa. Hi ha diverses formules que permeten la utilització d'ambdós mètodes de manera

indiferent.

Si el mètode de recepció de missatges és bloquejant, el codi s'estructura potser de forma molt intuïtiva: arribats a un cert punt del flux d'execució al qual s'espera rebre un missatge, no es continua fins que aquest missatge arribi o s'exhaureixi un temps determinat d'espera. En rebre el missatge l'execució continua. Però, si el mètode de recepció no és bloquejant, no podem assegurar que en l'instant just d'execució d'aquell mètode es rebrà un missatge entrant. Si es vol esperar al missatge s'haurà de bloquejar el *behaviour* sencer. No obstant, per bloquejar el *behaviour* cal finalitzar la seva execució i tornar a començar des del principi. Si el *behaviour* s'implementa com una màquina d'estats es podrà reemprendre el flux d'execució just al punt posterior a la recepció del missatge.

Quan un *behaviour* crida a algun dels mètodes de recepció de missatges, rep el primer missatge entrant. No obstant, existeix la possibilitat d'establir uns filtres de recepció per tal de capturar únicament els missatges que acompleixin una certa característica. Aquests filtres reben el nom de MessageTemplate.

Quan s'ha dissenyat l'ontologia dels components s'ha establert un camp obligatori que contindrà el nom i la versió del component que l'esta fent servir. A través de la implementació de la interfície MessageTemplate.MatchExpression es pot definir un filtre que, accedint al contingut del missatge decideixi si s'ha de tractar o no. D'aquesta manera, cada component haurà d'establir el seu filtre de recepció de missatges i passar-ho com a argument a les funcions *receive()* i *blockingReceive()*.

És freqüent que els components estableixin comunicacions secretes a través de l'esquema ACL in ACL. Recordem que el missatge intern és qui porta la informació del component que ha emès el missatge. La interfície MessageTemplate.MatchExpression permet fer totes les comprovacions necessàries al missatge per tal d'establir si passa el filtre, o no. Així doncs, s'hauria d'implementar l'interfície de manera que, per cada component, desxifri el missatge intern i comprovi cap a quin component va dirigit. En termes d'eficiència el mètode presenta un problema greu. Cada component que esperi rebre un missatge xifrat haurà de desxifrar tots els missatges ACL in ACL entrants per comprovar si van dirigits a

ell.

Partim de la base que l'acte de desxifrat és una operació costosa que creix linealment depenent de la mida de les dades a desxifrar i exponencialment depenent de la mida de la clau. Si l'agent inclou N components en el millor dels casos per cada missatge entrant s'haurà de fer un únic desxifrat. En el pitjor dels casos N, un per cada component. Com a mitjana s'efectuaran N/2 desxifrats per cada missatge entrant. La complexitat total de l'operació és $((A * M)^K) * \frac{N}{2}$ on A és la complexitat de l'algorisme, M la mida de les dades i K la longitud de la clau.

Trobar solució al problema anterior no és trivial. Una manera de no tenir la necessitat d'haver de desxifrar missatge a missatge per cada component és collocar la informació del component fora de la part xifrada del missatge. Aquesta situació atemptaria contra la transparència de l'esquema ACL in ACL ja que cada component s'hauria d'implicar en la construcció de l'embolcall del missatge que demana enviar. A més, suposaria modificar l'esquema general ACL in ACL per tal de adaptar-ho al cas de comunicacions entre components. Per contra és redueix la complexitat a $((A*M)^K)$. En aquesta primera versió es fa servir el mètode citat al paràgraf anterior; desxifrar missatge a missatge, component a component tot i que s'és conscient de la ineficiència del mètode.

4.5.2 Tracking

Com s'ha exposat a la fase d'anàlisi, el component de *tracking* és de tipus compost. Això implica establir una comunicació entre l'agent mòbil i l'agent de suport sotmesa a un protocol especial que satisfaci tots els requeriments mostrats a la fase d'anàlisi. El protocol dissenyat és el següent:

Format dels missatges:

1. Missatge informatiu:
   El missatge informatiu és l'iniciador del protocol. El component emissor és el component d'acció i va dirigit cap al component de suport. El format del missatge és ACL amb l'ontologia ComponentOnto.

El predicat conté un concepte ComponentDescription, un MessageContainer i un ObjectContainer. El ComponentDescription té el contingut “Tracking” al camp Name i “1” al camp Version. El MessageContainer presenta el missatge “Agent _ located in platform _ at time _”. El concepte ObjectContainer conté un SignedObject signat per la plataforma incloent la sigantura de l’agent, d’aquesta manera es podrà verificar l’origen del missatge. Les restriccions per l’espera de la resposta son tres reexpedicions del missatge i un temps d’espera de 5 segons. Les verificacions en destí que requereix el missatge son: comprovar l’autenticitat de l’emissor i comprovar la signatura de la plataforma des d’on s’ha emès el missatge. Si les comprovacions han estat satisfactòries la resposta al missatge serà un missatge de tipus ACK, en cas contrari es respondrà amb un missatge Cancel.

2. Missatge ACK:
   El missatge ACK és la resposta al missatge informatiu si totes les verificacions sobre aquest han estat satisfactòries. Per tant, el remitent del missatge és el component de suport i el receptor és el component d’acció. El format del missatge novament és l’ontologia ComponentOnto sobre un missatge ACL.

   El predicat conté un concepte ComponentDescription i un ObjectContainer. El ComponentDescription té el contingut “Tracking” al camp Name i “1” al camp Version. El concepte ObjectContainer conté un SignedObject amb el contingut del missatge informatiu prèviament rebut. Quan el missatge ACK arriba al seu destí, el component d’acció ha de verificar que la informació que s’està confirmant és realment la que va comunicar al component de suport. A més, ha de verificar l’autenticitat de l’agent de suport mitjançant la seva signatura. Si les verificacions han estat satisfactòries, el component finalitza la seva execució. En cas contrari, el component espera la recepció d’un missatge correcte seguint les restriccions d’espera de resposta pròpies del missatge informatiu.

3. Missatge Cancel:
   El missatge cancel es produeix com a resposta a un missatge informatiu

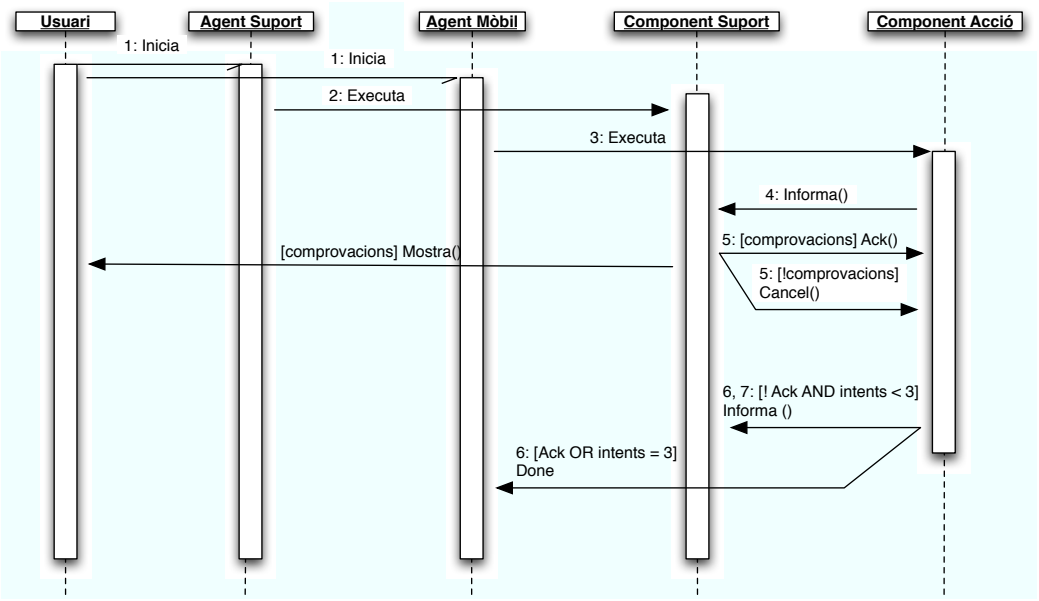

Figura 4.4: Diagrama de seqüència del component Tracking

quan no s'hagi pogut completar el procés de verificacions amb èxit. Novament, el remitent del missatge és el component de suport i el destinatari el component d'acció. El format és l'ontologia ComponentOnto sobre un missatge ACL.

El missatge conté el camp ComponentDescription de rigor juntament amb un ObjectContainer. El concepte ObjectContainer inclou un SignedObject amb el missatge: "Impossible to verify origin autenticity". L'única verificació verificació que s'estableix al destí és verificar l'autenticitat de l'emissor. Si les verificacions han estat satisfactòries el protocol s'ha de finalitzar de forma immediata. En cas contrari, el component ha d'esperar un missatge correcte seguint les restriccions d'espera i reexpedició del missatge informatiu.

A la Figura 4.4 es mostra el diagrama de seqüència del protocol.

4.5.3 Data retrieval

L'objectiu d'aquest component és emmagatzemar de forma segura les dades que genera l'agent mòbil durant el seu recorregut per posteriorment enviar-les a l'usuari. Tot seguit veurem quines implicacions te el fet de protegir les dades contra possibles alteracions.

La primera qüestió a resoldre és el significat de emmagatzemar les dades de forma segura. Els aspectes de seguretat desitjables a l'hora d'emmagatzemar dades son integritat en el conjunt de les dades, autenticitat dada a dada i confidencialitat opcional dada a dada. Anem a veure quines implicacions tenen cadascun d'aquests aspectes de seguretat. Pel que fa a la integritat en el conjunt de dades, és molt important visualitzar les dades o resultats que genera un agent mòbil no com una única entitat, sinó com un conjunt de dades que s'han generat a cada plataforma i que han de mantenir una coherència entre elles. Amb l'autenticitat dada a dada es pretén garantir que dins del conjunt de les dades generades totes i cadascuna de les dades han estat generades pel nostre agent mòbil i no han estat alterades per cap entitat externa. Per últim, la confidencialitat opcional de cada dada respon al fet que seria desitjable que en ultima instància fos l'usuari qui decidís si emmagatzemar les dades de forma confidencial o no. El mecanisme hauria de ser prou flexible com per poder emmagatzemar dades de qualsevol mena.

A [MS03] es proposa una estructura per poder complir amb les especificacions anteriorment presentades. La proposta és generar una seqüència de resultats on cada entrada conté un resum de la entrada anterior, la plataforma següent a la qual es traslladarà l'agent i les dades que ha generat en la plataforma actual. És obligatori afegir una entrada per cada plataforma visitada tot i que no es generin dades. Cada entrada anirà signada per la plataforma on s'ha generat deixant constància de quin agent ha demanat la signatura. L'esquema ha estat batejat sota el nom AppendOnlyStructure. A la figura 4.5 es mostra el diagrama de classes de la estructura.

El que aconseguim és impossibilitar l'alteració de les dades ja generades, impossibilitar la extracció i/o inserció de dades corruptes en l'estructura i garantir

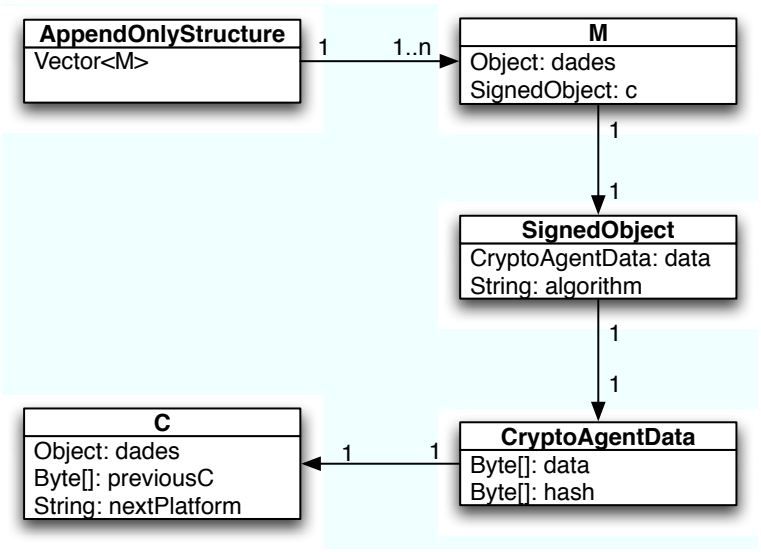

Figura 4.5: Diagrama de classes del mòdul AppendOnlyStructure

que les dades han estat generades per l'agent mòbil a una plataforma determinada.

Quan l'estructura és retornada a l'usuari, aquest, coneixent l'itinerari que ha recorregut l'agent mòbil, podrà verificar que les dades s'han generat en ordre correcte. Coneixent la signatura de l'agent mòbil podrà verificar que les dades han estat generades realment per l'agent mòbil. Per últim, tenint accés a les claus publiques de cada plataforma podrà verificar que les dades han estat generades a la plataforma especificada.

La forma d'interacció entre l'agent mòbil i el component d'acció és la següent: L'agent executa els *behaviours* definits per l'usuari a cada plataforma. Aquests *behaviours* han de emmagatzemar les dades que vulguin retornar a l'usuari en el camp dades de l'agent mòbil i a més hauran d'especificar si volen que les dades s'enviïn en aquell punt o no. Prèviament a la migració, l'agent mòbil executarà el component data retrieval. Aquest component agafarà les dades de la classe MyMobileAgent i les emmagatzemarà de forma segura en una estructura AppendOnlyStructure. És a dir, inclourà les dades en un objecte de tipus C amb la resta de la informació. Tot seguit demanarà al servei criptogràfic la signatura d'aquest

objecte. El resultat de la signatura serà un objecte de la classe SignedObject que contindrà un objecte de la classe CryptoAgentData. Aquest objecte serà inclòs dins d'un objecte M. De mode opcional, una copia de les dades originals serà adjuntada també. Finalment s'afegirà a dins de la estructura. Un cop emmagatzemades les dades, el component demanarà a l'agent si aquestes han de ser enviades o no en aquell punt de l'itinerari. A la figura 4.6 es mostra el diagrama de seqüència.

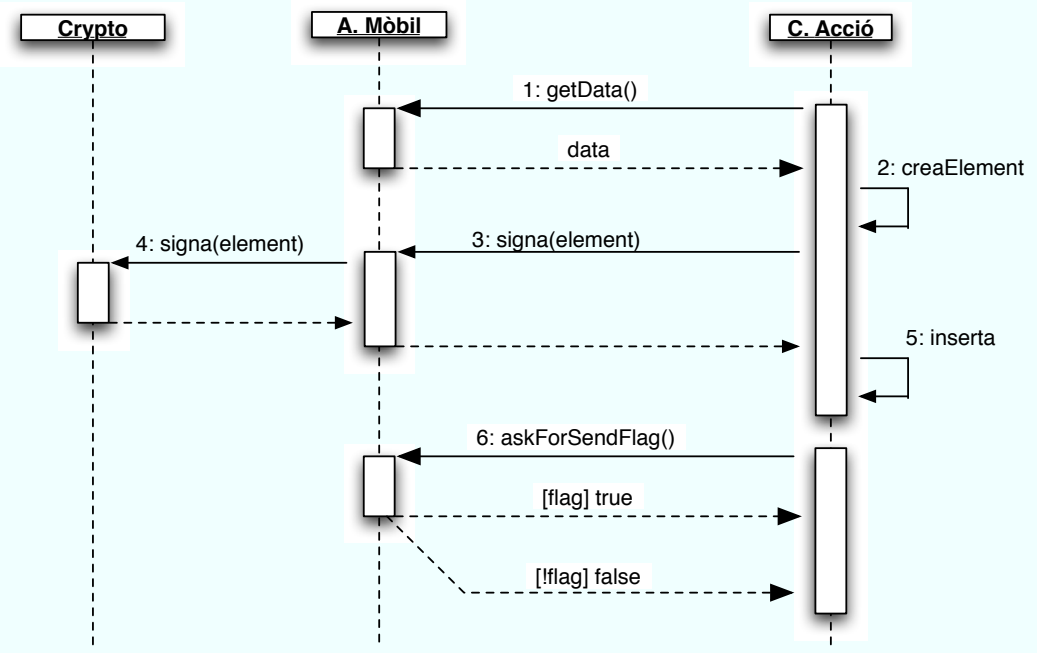

Figura 4.6: Interacció entre el component Data Retrieval, l'agent mòbil i el servei de criptografia.

S'han dissenyat dos versions d'aquest component. La primera s'anomena "data retrieval over AC" i la segona "data retrieval over SMTP". La manera d'emmagatzemar les dades és l'anteriorment citada en ambdós casos. La diferencia entre els dos components recau en la forma d'enviar les dades. Mentre que "data retrieval over ACL" fa arribar les dades mitjançant l'agent de suport, "data retrieval over SMT" permet la desconnexió de l'usuari fent-li arribar les dades via *e-mail*.

4.5.4 Data retrieval over ACL

Data retrieval over ACL és un component de tipus compost. L'especificació del protocol és la següent:

Format dels missatges:

1. Missatge "Inici":
   El component iniciador del protocol és el component d'acció que s'executa a l'agent mòbil. El destinatari del missatge és el component de suport. La comunicació s'estableix sobre l'esquema ACL in ACL. Per tant, l'ontologia de l'embolcall és SecureACLinACLonto mentre que l'ontologia del missatge intern és ComponentOnto.

   El missatge intern conté un predicat ComponentPred amb un concepte ComponentDescription, un altre de tipus MessageContainer i un últim concepte de tipus ObjectContainer. El ComponentDescription conté "Data retrieval over ACL" al camp Name i "1" al camp Version. El MessageContainer conté un *string* aleatori al camp Message i un "1" al camp Seq. num. El ObjectContainer conté el mateix *string* aleatori però de forma signada per tal de que es pugui verificar l'autenticitat de l'emissor.

   S'ha d'establir una única verificació en destí: comprovar l'autenticitat de l'agent mòbil. Si la verificació ha estat satisfactòria la resposta serà un missatge Accept, en cas contrari es respondrà amb un missatge Cancel.

   Les restriccions temporals i de reexpedició s'han establert a tres reexpedicions del missatge i 10 segons per rebre resposta.
2. Missatge "Accept":
   El missatge Accept és la resposta al missatge d'inici del protocol en cas de que es compleixin totes les verificacions. Degut a que la comunicació s'estableix de l'agent de suport cap a l'agent mòbil no es podrà fer servir l'esquema ACL in ACL ja que l'agent mòbil no pot posseir cap clau secreta. Per tant, el format del missatge serà ComponentOnto sobre ACL.

El missatge conté un predicat ComponentPred amb els conceptes ComponentDescription, MessageContainer i ObjectContainer. El ComponentDescription conté la mateixa informació sobre el component especificada al missatge d'inici. El concepte MessageContainer conté com a missatge el *string* aleatori rebut i el nombre de seqüència "2". El ObjectCOntainer conté el *string* aleatori signat per tal de que es pugui verificar l'autenticitat de l'emissor.

La comprovació a realitzar en destí d'aquest missatge és verificar l'autenticitat de l'emissor. Si la comprovació es realitza correctament es respondrà amb el missatge DataTransfer. En cas contrari s'haurà d'avortar el protocol.

Les restriccions temporals i de reexpedició s'han establert a tres reexpedicions del missatge i 10 segons per rebre resposta.

3. Missatge "Refuse":
   El missatge Refuse és la resposta al missatge d'inici en cas de que les verificacions no hagin estat satisfactòries. Degut a que el sentit és del component de suport cap al component d'acció, el format del missatge serà ComponentOnto sobre ACL.

   El concepte ComponentPred conté un ComponentDescription amb la informació del protocol, un concepte MessageContainer amb el missatge "Refuse" i "0" com a nombre de seqüència. El concepte ObjectContainer inclouria la signatura de l'agent de suport per tal de poder verificar la seva autenticitat.

   En rebre el missatge s'ha de verificar l'autenticitat de l'agent de suport. Si les comprovacions han estat satisfactòries s'ha d'avortar el protocol. En cas contrari el missatge serà ignorat i el component continuarà a l'espera de rebre un missatge correcte.

4. Missatge "DataTransfer":
   El missatge DataTransfer s'estableix com a resposta del missatge Accept. Les dues parts implicades en el protocol han estat autenticades i la predisposició és la de començar la transferència de dades. El missatge DataTranfer

te com a inici el component d'acció i com a destinatari el component de suport. El format del missatge és ComponentOnto sobre l'esquema ACL in ACL.

El predicat intern conté un concepte ComponentDescription amb la informació del component, un concepte MessageContainer amb el missatge "data" i nombre de seqüència "3" i un ObjectContainer amb una estructura AppendOnlyStructure inclosa dins d'un objecte SignedObject.

El component de suport en rebre el missatge ha de verificar l'autenticitat de l'emissor. Si la verificació es completa amb èxit, es respondrà amb un missatge ACK. En cas contrari la resposta serà un missatge Refuse i es continuarà amb l'execució a l'espera de rebre un missatge correcte.

Les restriccions temporals i de reexpedició per la recepció d'una resposta correcta s'han establert a tres reexpedicions del missatge i 10 segons per rebre resposta.

5. Missatge "ACK":
   El missatge ACK té com a emissor el component de suport i com a receptor el component d'acció. El format és ComponentOnto sobre ACL.

   El predicat ComponentPred conté un concepte ComponentDescription amb la informació del component. Un concepte MessageContainer conté el missatge "Done" i el nombre de seqüència "4". Un concepte ObjectContainer amb un SignedObject contenint el *string* "Done" es farà servir per poder autenticar a l'emissor.

   El component d'acció ha de verificar l'autenticitat de l'emissor del missatge. Si la verificació és correcta es finalitza el protocol. En cas contrari s'ignora el missatge i es continua amb el protocol tenint en compte les restriccions temporals i de reexpedició del missatge DataTransfer.

A la figura 4.7 es mostra el diagrama de seqüència del protocol.

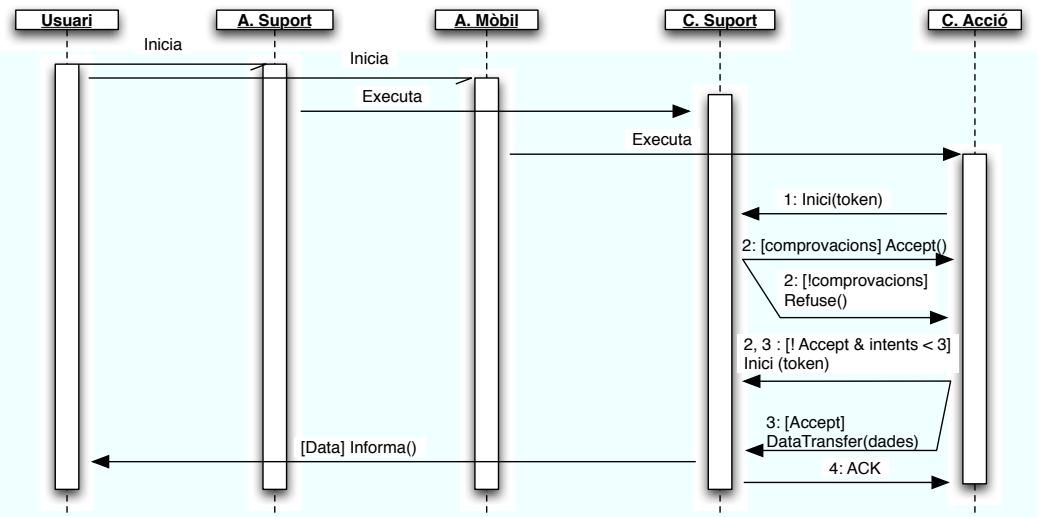

Figura 4.7: Diagrama de seqüència del component Data retrieval over ACL.

4.5.5 Data retrieval over SMTP

Data retrieval over SMPT és un component simple, és a dir; no requereix d'un agent de suport per tal de realitzar la seva feina. El component implementa un petit client del protocol SMTP [rfc2812] (Simple Mail Transfer Protocol) que mitjançant les extensions MIME [rfc2045] (Multipurpose Internet Mail Extensions) és capaç d'enviar un fitxer que conté l'estructura de dades de forma serialitzada. El protocol de transport sobre el que s'ha implementat el client ha estat SSL, d'aquesta manera les dades circulen de forma xifrada dificultant que cap entitat externa intercepti la comunicació. Un requisit indispensable per la correcta execució d'aquest component és que el firewall de la xarxa on es trobi executant-se el component ha de permetre les comunicacions exteriors cap al port 465. A la figura 4.8 es mostra el diagrama de seqüència del protocol.

Extractor

En última instància, els components "Data retrieval" generen fitxers que contenen les dades que ha generat l'agent. Aquests fitxers contenen un objecte serialit-

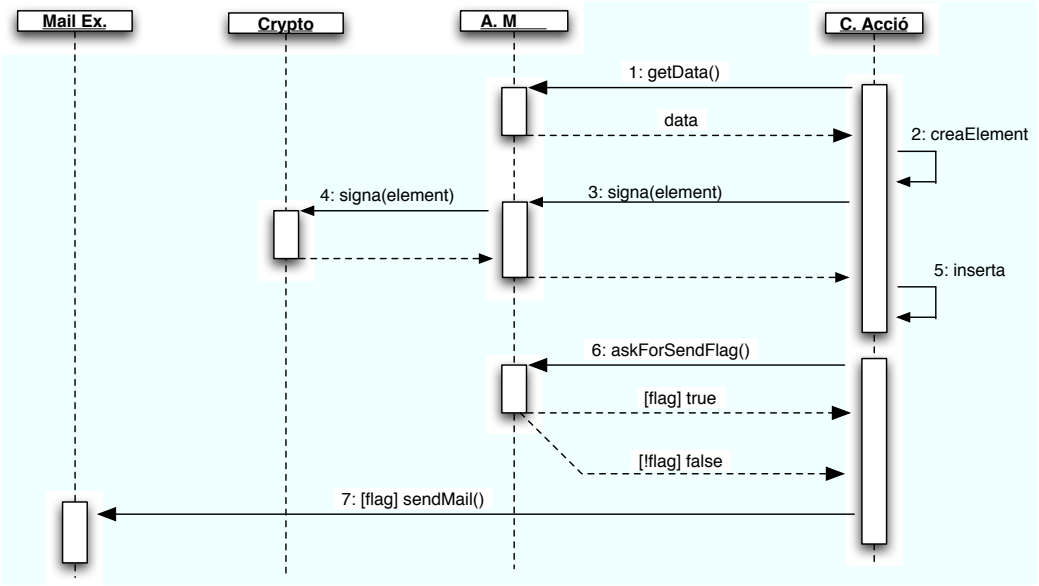

Figura 4.8: Diagrama de seqüència del component Data retrieval over SMTP.

zat de tipus AppendOnlyStructure. L'usuari necessita una entitat que sigui capaç d'extraure les dades que es troben dins dels fitxers alhora que verifica la seva coherència. A aquest mòdul l'anomenarem extractor.

L'extractor no és un component, de fet pot no ser ni tan sols un agent. No obstant, en aquesta primera versió, ens hem vist forçats a implementar l'extractor dins d'un agent ja que l'única manera d'accedir a les claus públiques de les plataformes és a través del servei de criptografia. Es preveu que en un futur es podrà accedir a aquesta informació a través d'un servei de directori.

La informació necessària per verificar la integritat de les dades és l'itinerari que ha recorregut l'agent, les claus públiques de les plataformes incloses a l'itinerari i la signatura de l'agent mòbil que genera el servei de criptografia.

Mitjançant el coneixement de l'itinerari, l'extractor serà capaç de verificar les signatures de les plataformes. D'aquesta manera podem garantir que cada element de l'estructura ha estat generat a la seva plataforma corresponent. A part, coneixent la signatura de l'agent mòbil es podrà assegurar que les dades han estat generades exclusivament per ell.

L'algoritme d'extracció és el següent:

```
Des de l'ultima dada fins la primera
    Verificar la signatura de la plataforma.
    Verificar la signatura de l'agent.
    Verificar el hash de la dada anterior.
    Verificar la coherència de salt.
fiDesDe
```

Per verificar la signatura de la plataforma cal accedir a l'itinerari per tal de demanar la seva clau pública. La verificació de la signatura de l'agent implica conèixer per endavant quina és la signatura que generarà el servei de criptografia. La verificació del *hash* de la dada anterior impedeix la modificació dels elements continguts a l'estructura. Per últim, la coherència de salt tracta de verificar que el camp de següent plataforma de la dada anterior ha estat coherent amb l'itinerari seguit, d'aquesta manera podrem detectar si una agència no ha generat una entrada a l'estructura.

Si durant la validació de l'estructura es detecta alguna situació irregular, el mòdul extractor llença una excepció i termina la seva execució. Queda com a possible ampliació (Capítol 6), decidir que fer amb una estructura corrupta depenent de la sensibilitat de les dades que hi conté.

Fins aquí arriba la fase de disseny i implementació. Al llarg d'aquest capítol s'han mostrat les solucions adoptades per plasmar en la feina realitzada tots aquells requisits mostrats a la fase d'anàlisi. Al següent capítol es mostrarà el funcionament dels mòduls desenvolupats així com es farà una descripció de l'entorn necessari per l'execució. Per tancar el capítol es descriurà quin és el l'estat real d'integració que s'ha pogut assolir amb les eines del grup de recerca.

Capítol 5

Proves, execució i integració.

Fins ara hem vist els detalls d'implementació dels mòduls desenvolupats. En aquest capítol es mostrarà l'entorn necessari per l'execució d'agents mòbils que implementin el treball desenvolupat. A continuació es mostrarà proves testimonials de l'execució dels components desenvolupats i finalment es valorarà l'estat de l'art de la integració del projecte amb les eines del grup de recerca.

5.1 L'entorn d'execució

En aquesta secció és mostraran els elements necessaris per poder realitzar les proves amb els elements desenvolupats. Val a dir que la creació dels agents no ha estat realitzada mitjançant les eines del grup de recerca. A seccions posterior veurem quin és exactament l'estat de l'art del procés d'integració. La figura 5.1 mostra l'esquema de l'entorn necessari per l'execució.

Servidor de claus públiques: El servidor de claus públiques s'encarrega de proporcionar accés a les claus públiques de cada plataforma. Cal especificar al servei de criptografia la localització del servidor. Un cop configurat, a través de peticions al servidor criptogràfic, els agents podran consultar les claus públiques de les agències per tal de verificar les seves signatures.

Agent de suport: L'agent de suport ha d'executar-se en una agència controlada per l'usuari. Aquesta agència ha d'haver estat iniciada com a mínim amb el

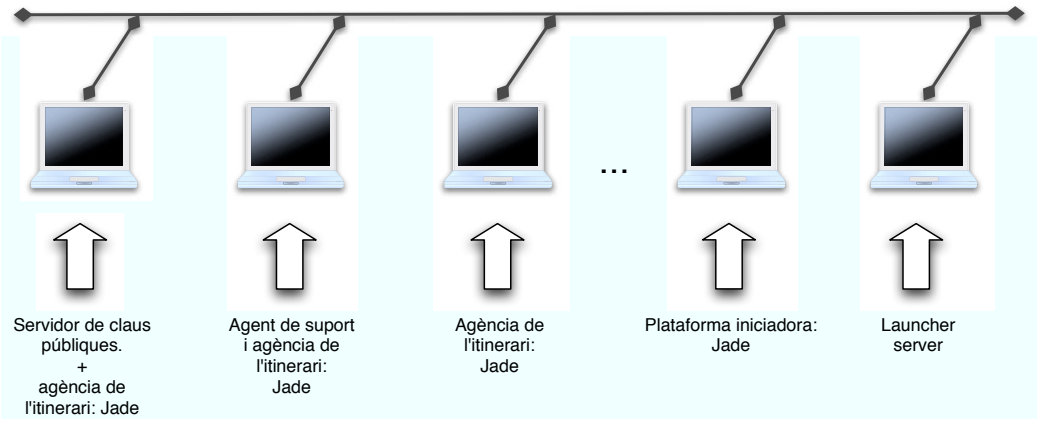

Figura 5.1: Esquema de l'entorn d'execució.

servei de criptografia. L'agent mòbil ha de conèixer tant el nom de l'agent com la seva localització. D'aquesta manera és podrà comunicar amb ell. L'agència on s'executa l'agent de suport port formar part de l'itinerari de l'agent mòbil si aquesta ha estat iniciada amb el servei de migració.

Agències de l'itinerari: Les agències que formen part de l'itinerari han d'haver estat iniciades amb el servei de migració i de criptografia. Val a dir que aquestes agències queden fora del control de l'usuari, per tant, no es consideren entorn d'execució controlat.

Launcher server: Aquest servidor rep peticions del *client launcher* i inicia l'execució d'un agent mòbil a la plataforma especificada.

Plataforma iniciadora: La plataforma iniciadora por ser de dos tipus. Per una banda pot contenir una agència que inicia l'execució de l'agent mòbil. Per l'altra es pot comunicar amb el *launcher* per tal d'iniciar l'execució d'un agent estalviant així executar una agència a la màquina.

5.2 Proves

Un cop vist l'entorn d'execució necessari mostrarem les proves realitzades així com una petita valoració dels resultats.

Totes les proves han estat realitzades a la sala de projectistes del departament. Aquesta sala consta de 12 màquines actives, des de la ccd-pr0 fins la ccd-pr11.

La configuració de l'entorn d'execució és la següent. El servidor de claus públiques es localitza a la plataforma ccd-pr3. L'agent de suport es localitza a la plataforma ccd-pr9. L'itinerari a seguir per l'agent és ccd-pr2, ccd-pr3, ccd-pr4, ccd-pr6, ccd-pr7, ccd-pr8 i ccd-pr2.

El primer component sotmès a test és el component de *tracking*. Aquest component implementa autenticació en les comunicacions. La informació mostrada per pantalla es mostra a la figura 5.2. El pes total de l'agent és de 32,2k bytes.

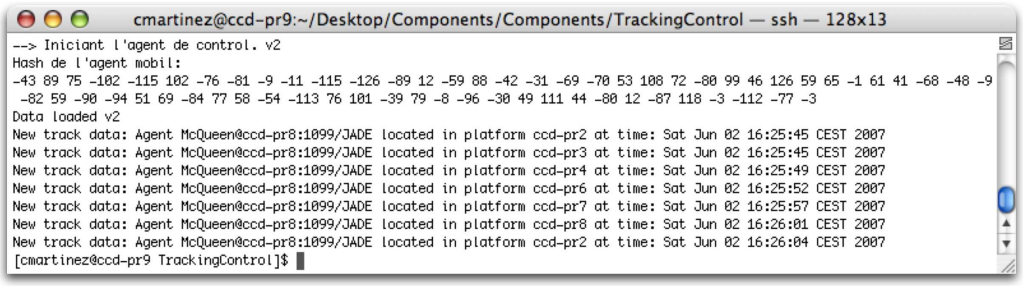

Figura 5.2: Informació de *tracking* arribada a l'agent de suport.

El segon component sotmès a proves és el component "Data retrieval over ACL". L'agent mòbil segueix l'itinerari citat i en arribar a l'ultima plataforma del recorregut envia les dades cap a l'agent de suport. A cada plataforma es generen com a objecte dades el string "data entry number X from platform Y" on X és el nombre de seqüència amb el que s'han generat les dades i Y és el nom de la plataforma. Aquest component implementa l'esquema ACL in ACL xifrant les dades mitjançant RSA amb una clau de longitud de 64 bits. Els resultats obtinguts amb l'extractor es mostren a la figura 5.3. El pes de l'agent és de 38,2k bytes.

El tercer component sotmès a test és el component "Data retrieval over SMTP". L'itinerari recorregut ha estat l'anteriorment citat. En arribar a l'última agència de en component ha enviat un mail amb les dades recol·lectades. La figura 5.4 mos-

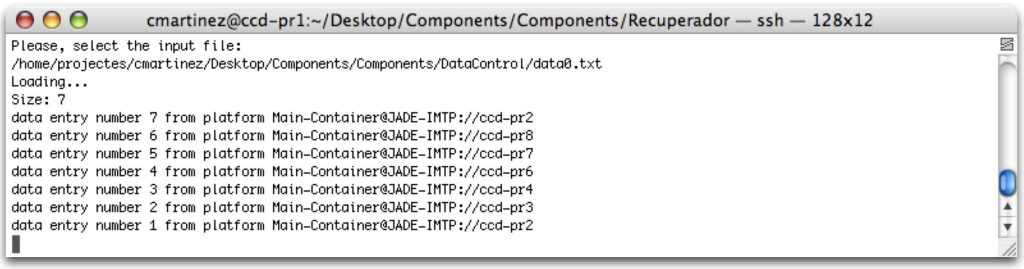

Figura 5.3: Extracció de les dades generades pels components “Data retrieval”.

tra el correu electrònic rebut. Els resultats de l’extracció de dades son idèntics als mostrats en la figura 5.3. El pes de l’agent és de 32,0k bytes.

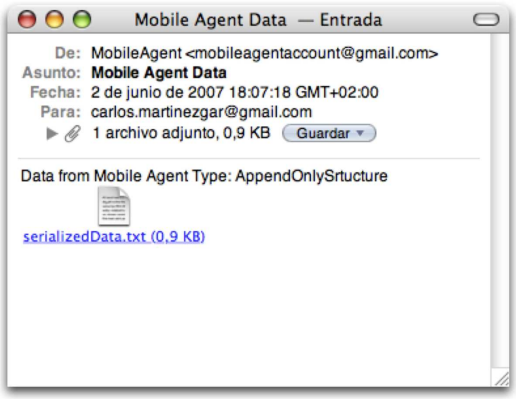

Figura 5.4: Correu electrònic enviat pel component “Data retrieval over SMTP”

Per últim, s’ha generat una prova que integra el component “Tracking” i el component “Data retrieval over ACL”. L’itinerari que segueix l’agent és l’anteriorment citat però en aquest cas els resultats seran enviats cap a l’agent de suport a des de la plataforma ccd-pr4 i des de l’ultima plataforma de l’itinerari. El component de tracking es comunica mitjançant missatges ACL mentre que el component de recuperació de dades ho fa mitjançant ACL in ACL amb una clau de 64 bits. A la figura 5.5 es mostra una captura de pantalla testimonial de l’execució de la prova. El pes de l’agent és de 42,1k bytes.

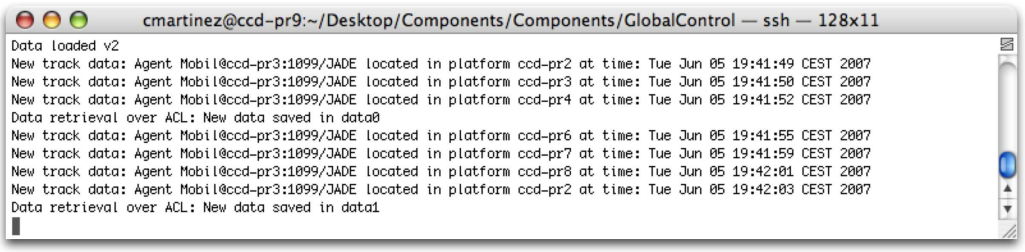

Figura 5.5: Agent de suport rebent informació simultània dels components "Tracking" i "Data retrieval over SMTP"

Aquestes proves testimonials demostren el funcionament dels components desenvolupats així com el funcionament de l'esquema ACL in ACL treballant sobre el servei de criptografia. A més, demostren la viabilitat de les comunicacions concurrents component a component. L'objectiu no ha estat mesurar en termes d'eficiència cap component.

5.3 Integració

En aquesta secció es pretén fer una valoració dels resultats d'integració del projecte amb la resta de mòduls que conformen l'entorn de desenvolupament i execució propi del grup de recerca.

En primer lloc, la integració amb el servei de criptografia s'ha completat amb èxit. Gràcies al servei s'han pogut assolir els requisits de confidencialitat i autenticació exigibles als components desenvolupats. L'esquema ACL in ACL gaudeix dels mecanismes de xifratge per tal de proporcionar secret a les comunicacions. A més, l'ontologia dels components suporta la transmissió de les signatures que genera el servei per tal d'afegir autenticació a les comunicacions. No obstant, en un futur, es podria realitzar una tasca de desenvolupament conjunta per tal de deslligar l'esquema d'autenticació de l'ontologia dels components i implantar-la directament sobre l'esquema ACL in ACL.

La integració amb el servei de migració ha estat total des del primer dia. Ens trobem amb una eina molt madura i perfectament integrada a l'entorn de Jade.

Realitzar una migració d'un agent cap a una altra agència és tan trivial com fer una crida a un senzill mètode.

L'altra eina d'integració més immediata és la la IDT. Degut a que es tracta d'un projecte a part [smard], encara en temps de desenvolupament, no s'ha pogut inserir la selecció dels components a l'interfície gràfica. Tampoc genera les especificacions necessàries com per que l'eina *agent builder* inclogui els components seleccionats a l'agent.

Per últim, la integració amb l'eina *agent builder* és potser la més costosa de fer. Aquesta eina també es troba en fase de desenvolupament dins d'un projecte a part [smard]. S'han fet proves d'inclusió manual de components simples i el resultat ha estat satisfactori. Per contra, el suport per components compostos, la generació de l'agent de suport, el pas dels paràmetres necessaris als agent i la generació d'agents capaços d'accedir al servei de criptografia són objectius pendents.

En aquest penúltim capítol s'han mostrat l'entorn necessari per l'execució així com les proves realitzades amb els components desenvolupats. Per últim, s'ha mostrat l'estat de l'art de la integració del projecte amb la resta d'eines desenvolupades al grup de recerca. Al proper capítol es presentaran les conclusions de la memòria i les possibles línies d'ampliació.

Capítol 6

Conclusions i línies d'ampliació.

Arribats a aquest punt de la memòria s'ha vist com la gran complexitat que suposa la construcció d'agents mòbils ha requerit dividir el problema en subproblemes més petits mitjançant la creació de components. Gràcies a les eines de desenvolupament del departament, el procés d'inclusió de components a un nou agent es redueix a seleccionar-los d'una llista. Una llibreria mantindrà ordenats per funcionalitats tots els components disponibles. D'aquesta manera, el temps de desenvolupament i la complexitat de disseny d'un agent es veurà reduïda considerablement. Mitjançant els fonaments presentats al capítol 2 s'ha tractat de submergir al lector en el marc del problema presentat i les eines existents per tractar de solucionar-lo. A través de la fase d'anàlisi i posteriorment disseny i implementació s'han identificat les parts que conformen la feina realitzada i s'ha mostrat quines son les solucions adquirides per tal de materialitzar-les. Al capítol anterior s'han mostrat els resultats de proves testimonials per tal de mostrar i demostrar el funcionament del treball realitzat. En aquest darrer capítol, però, ens manca valorar el grau de compliment dels objectius inicials i presentar línies d'ampliació del treball desenvolupat.

Més detalladament, la feina realitzada es pot concloure en els següents punts:

- S'han identificat els elements que conformen la infraestructura necessària per tal de realitzar un desenvolupament d'agents basat en components. Prenent com a punt de partida els agents mòbils, hem presentat on encaixen

els components i a partir de les funcionalitats hem establert la necessitat de la presència d'un agent de suport. A més s'han identificat els elements requerits en les comunicacions entre agents modulars.

- Un cop han estat diferenciades totes les parts que formen part del projecte, s'ha realitzat un estudi de requeriments de cada element. Hem vist els requeriments de l'agent mòbil i de l'agent de suport en quant als paràmetres que requereixen en temps de construcció així com la seva estructura interna. Hem estudiat, també, els requisits que presenta la comunicació entre agents modulars i com possibilitar que la comunicació es realitzi de component a component i de forma segura, si s'escau, mitjançant l'esquema "Acl in Acl".
- Com a part del treball realitzat han estat desenvolupats tant el component "Tracking" com els components "Data retrieval over ACL" i "Data retrieval over SMTP". El component "Tracking" afegeix la funcionalitat d'informar sobre la posició de l'agent mòbil després de cada migració. Els components "Data retrieval over ACL" i "Data retrieval over SMTP" habiliten l'entrega de dades cap a l'usuari via missatges ACL o via correu electrònic respectivament.
- El treball realitzat ha vetllat per la integració amb les eines existents del departament. Si bé és cert que la integració no ha estat total amb totes les eines per falta de temps, si que es pot afirmar que la feina ha estat pensada per realitzar aquesta integració amb els mínims canvis als components desenvolupats.
- Per últim ha estat possible la utilització del servei de criptografia desenvolupat de forma paral·lela al projecte "Desenvolupament d'un servei criptogràfic per a la plataforma JADE" gràcies a reunions per tal d'establir la forma d'interacció entre els dos projectes.

Arribats a aquest punt podem concloure que els objectius inicials s'han complert satisfactòriament. No obstant, a mesura que es va aprofundint en la temàtica del projecte apareixen línies d'ampliació que es desmarquen dels objectius inicials del projecte. A continuació es mostren possibles continuïtats de la feina realitzada.

1. La línia de continuïtat més òbvia, és potser el desenvolupament de nous components. Cal però, un estudi previ per tal d'establir quins son els requisits d'un conjunt raonablement gran d'agents que justifiqui la creació d'un nou component. Un exemple pot ser el component de terminació. El temps de vida de l'agent de suport ve determinat pel temps de vida de l'agent de mòbil. Un component compost de terminació podria alertar a l'agent de suport de la finalització de l'execució de l'agent mòbil.

2. Les comunicacions secretes en l'actualitat únicament es poden establir des de l'agent mòbil cap a l'agent de suport. Si l'agent mòbil portés una clau secreta, en arribar a una agència maliciosa, el secret de la clau quedaria exposat. No obstant, un protocol d'intercanvi de claus de sessió pot habilitar l'establiment de comunicacions secretes en ambdues direccions, des de l'agent mòbil cap a l'agent de suport i viceversa. La idea és generar una parella de claus per cada agència que es visita i desfer-se d'elles en abandonar la plataforma. L'agència maliciosa podria interceptar les comunicacions ja que tindria accés a les claus de sessió però dificultaria molt el fet de que una tercera entitat pogués interceptar la comunicació.

3. L'esquema de comunicacions ACL in ACL requereix haver d'extraure el missatge intern per decidir cap a quin component va dirigit. Una línia d'ampliació podria ser buscar un esquema que permetés accedir a l'informació d'entrega sense haver d'accedir al missatge intern tenint present que ACL in ACL es pot fer servir en tot tipus d'agents. També, mitjançant l'ajut del servei de criptografia es podria pensar en un esquema per incloure les signatures d'agent dins de l'embolcall ACL in ACL i no pas al missatge intern, sempre respectant els estàndards de FIPA.

4. Les estructures d'emmagatzemament segur de dades requereixen que l'itinerari que ha de seguir l'agent ha de ser conegut per habilitar la recuperació de les dades. Aquesta informació es pot extraure o bé mitjançant un *tracking* de l'agent o bé si l'itinerari és estàtic. De fet, cap de les dues possibilitats és bona. La primera per que trenca la independència entre el component de *tracking* i el component de recuperació de dades ja que implicaria incloure

un component sempre que s'hagi inclòs l'altre. La segona alternativa atempta contra unes de les funcionalitats base dels agents mòbils: possibilitat de generar itineraris dinàmics a mesura que exploren el món. Una modificació en l'algorisme d'extracció per que comencés extraient per la primera dada possibilitaria els itineraris dinàmics. A més caldria un mecanisme que indiqués als components "Data retrieval" quina serà la propera plataforma a visitar.

5. L'extractor de dades d'una estructura segura hauria de poder-se executar fora de l'entorn de Jade. A més, a l'hora de decidir si una estructura ha estat alterada o no, és possible trobar-se amb estructures corrompudes parcialment. Una possible ampliació és decidir que es fan amb les dades correctes d'una estructura semi-corrompuda depenent de com de crítiques son les dades.
6. Per qüestions de balanceig de càrrega i tolerància a falles l'agent de suport pot requerir moure's dins d'un entorn controlat. Es podria pensar en un protocol que alertes a l'agent mòbil del canvi de localització de l'agent de suport.

Bibliografia

[senda] Security of Networks and Distributed Applications. Universitat Autònoma de Barcelona. <http://www.senda.uab.es/>

[jade] Java Agent DEvelopment Framework. <http://jade.tilab.com/>

[FIPA] Foundation for Intelligent Physical Agents. <http://www.fipa.org/>

[ipmp] Inter-Platform Mobility Project. <https://tao.uab.cat/ipmp/>

[smard] Secure Mobile Agent Rapid Development. SourceForge project. <http://sourceforge.net/projects/smard/>

[VRCCNM525] P. Vieira-Marques, S. Robles, J. Cucurull, R. Cruz-Correia, G. Navarro and R. Martí. Secure Integration of Distributed Medical Data using Mobile Agents. IEEE Intelligent Systems. 21 (November-December 2006), no. 6.

[ARO347] J. Ametller, S. Robles and J.A. Ortega-Ruiz. Self-Protected Mobile Agents. In 3rd International Conference on Autonomous Agents and Multi Agents Systems. ACM Press, vol. 1, N.R. Jennings, C. Sierra, L. Sonenberg, M. Tambe, 362-367, Juliol 2004.

[M06] A. Moratalla. Entorn Integrat pel Desenvolupament i Seguiment d'Agents Mòbils. Master's thesis, Universitat Autònoma de Barcelona. Setembre 2006.

[MR] A. Moratalla and S. Robles. Component-Based Development of Secure Mobile Agents Applications. In International Central and Eastern European Conference on Multi-Agent Systems, September 2007.

[GND03] G. Vitaglion, N. Lhuilier and D. Greenwood. FIPA Agent Message Security Object Proposal, novembre 2003.

[MS03] P. Maggi R. Sisto. A configurable mobile agent data protection protocol. International Conference on Autonomous Agents juliol 2003.

[rfc2812] Simple Mail Transfer Protocol, abril 2001. <http://www.ietf.org/rfc/rfc2821.txt>

[rfc2045] Multipurpose Internet Mail Extensions (MIME) Part One, novembre 1996 <http://www.ietf.org/rfc/rfc2045.txt>

Firmat: Carles Martínez García.

Bellaterra, juny de 2007

Resum

La complexitat de disseny d'agents mòbils creix a mesura que s'incrementen les seves funcionalitats. Aquest projecte proposa enfocar el problema des d'un punt de vista modular. S'ha realitzat un estudi tant dels propis agents com de les parts que ho integren. De la mateixa forma s'han establert i implementat els mecanismes necessaris per habilitar les comunicacions segures entre agents. Finalment s'han desenvolupat dos components que ofereixen les funcionalitats de seguiment de l'agent mòbil i la recuperació dels resultats generats. El desenvolupament d'agents basats en components tracta d'aplicar la vella estratègia "divideix i venceràs" a la fase de disseny, reduint així, la seva gran complexitat.

Resumen

La complejidad de diseño de agentes móviles crece a medida que se incrementan sus funcionalidades. Este proyecto propone enfocar el problema desde un punto de vista modular. Se ha realizado un estudio de los propios agentes y de las partes que lo integran. De la misma manera se han establecido e implementado los mecanismos necesarios para las comunicaciones seguras entre agentes. Finalmente se han desarrollado dos componentes que ofrecen la funcionalidad de seguimiento del agente móvil y la recuperación de los resultados que pueda haber generado. El desarrollo de agentes por componentes trata de aplicar la vieja estrategia "divide y vencerás" en la fase de diseño, reduciendo así, su gran complejidad.

Abstract

The design complexity of mobile agents grows as their functionalities are increased. This project proposes to confront the problem from a modular point of view. A study has been made about the own agents and all the elements who integrate them. In the same way, necessary mechanisms for the secure communications between agents have been established and implemented. Finally two components have been developed that offer tracking and results retrieval functionalities. The component-based development of mobile agents tries to apply the old strategy "divide and conquer" in the design phase, reducing in this way, its great complexity.