CONTROL D'ACCÉS DEPENENT DEL CONTEXT. DISSENY I IMPLEMENTACIÓ DINS L'ENTORN PROSES.

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per David Quintana Brígido i dirigit per Sergi Robles Martínez.

Bellaterra, juny de 2010

El firmant, Sergi Robles Martínez , 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 David Quintana Brígido

Bellaterra, juny de 2010

Firmat: Sergi Robles Martínez

A tothom que m'ha acompanyat en l'aventura d'aquests darrers 5 anys

Agraïments

M'agradaria recordar a tothom que ha estat al meu costat aquests cinc anys, des dels companys que vaig tenir al principi de la carrera fins els que he tingut l'últim dia, especialment al Xavier Martínez que m'ha acompanyat en aquest viatge des del principi. I per suposat als meus amics i a tota la meva família, especialment al meu avi que ens va deixar l'estiu passat. Descansa en pau.

Gràcies a tots pel vostre suport.

Índex

Índex de figures

Capítol 1

Introducció

Fa poc més de quaranta anys, al 1969, es va produir la primera connexió entre dos ordinadors. Aquest va ser el primer pas cap al que avui anomenem Internet: una xarxa global d'ordinadors connectats entre ells, permetent als usuaris compartir informació. Han passat molts anys des de llavors, i en aquest temps el nombre de nodes connectats a la xarxa no ha parat de crèixer: actualment la xarxa consta de milions d'ordinadors interconnectats arreu del món. Però no només ha canviat el nombre de nodes connectats, sinó que també ha variat el mode de connectar-se a Internet: aquests nodes de la xarxa ja no són només estàtics (com els ordinadors personals), sino que també són dinàmics (ordinadors portàtils i també els telèfons mòbils). Aquest canvi de paradigma es reflexa en molts aspectes de la nostra societat, i fa que Internet evolucioni cap a un futur que farà que la xarxa ocupi tots els espais físics, tant els terrestres que coneixem avui en dia com els aeris i espacials que ocuparà en el futur.

Però aquesta evolució al llarg dels anys no només afecta al mode en com es connecten els elements a la xarxa, sinó que també afecta a altres elements que formen part de la comunicació entre elements, com podria ser la seguretat del sistema. El model tradicional de seguretat computacional va ser formulat a la dècada del 70, moment en els quals els ordinadors eren cars i només es podien trobar en determinades instal·lacions [Blakley]. Desde llavors, els avenços incessants en el món de la informàtica i de les comunicacions han provocat el creixement de

moltes aplicacions que s'allunyen del paradigma de sistemes aïllats per apropar-se a la integració d'aquests sistemes amb l'entorn.

Tot i que fa uns anys podria semblar inverosímil, la integració de vehicles aeris a la xarxa, ja siguin tripulats o no tripulats, està molt més a prop del que sembla. Ja ho podem trobar en sistemes militars que es comuniquen amb nodes de comunicació en xarxes distribuides. Però l'aviació civil també es prepara per aquest canvi, amb la introducció i posada en funcionament del programa SESAR (Single European Sky ATM Research). Aquest programa preveu l'ús massiu de transmissió de dades entre sistemes aeris, terrestres i orbitals. Per aconseguir aquest model de comunicació, els sistemes aeris actuaran com un node qualsevol de la xarxa. No obstant, a l'entorn on operen les aeronaus les connexions continues punt a punt com les que utilitzen els protocols que trobem a Internet són massa cares. En aquest punt sorgeix la necessitat de desenvolupar nous protocols que permetin l'intercanvi de dades en entorns on la connexió no és continua, però que a la vegada siguin integrables en Internet, i així poder aprofitar tota la experiència que suposa tots els anys de funcionament de la xarxa. Amb aquest objectiu es posa en funcionament PROSES: Protocolos de Red para el Cielo Único Europeo (Single European Sky) [proses].

Degut a característiques de l'entorn PROSES, com la mobilitat i l'absència d'un connexió continua, no es pot aplicar un mecanisme de seguretat com els que hi havia anteriorment, sino que necessitem un mecanisme que s'adapti a l'entorn a l'hora de prendre les decisions. PROSES fa servir agents mòbils per aconseguir traslladar safisfactoriament informació entre un node i un altre. Els agents mòbils són un tipus de software amb les qualitats d'autonomia, aprenentatge i la més important, la mobilitat. Això permet que els agents mòbils siguin capaços de moure's d'un node a un altre de forma autònoma, i després poder continuar la seva execució al node de destí. Els agents mòbils que contenen la informació de l'aplicació són llençats per aconseguir satisfer les necessitats d'un usuari que ha fet servir algunes de les aplicacions disponibles en aquest entorn. Com que és aquest agent el que va des de un node fins a un altre el control d'accés s'ha d'aplicar sobre aquest agent, i no pas directament sobre l'usuari. Els agents necessiten unes plataformes

per poder executar-se, i a l'entorn PROSES utilitzarem Jade [jade], que és una plataforma provada i sobre la qual s'han desenvolupat moltes aplicacions.

La integració amb l'entorn PROSES és el que provoca la necessitat de tenir en compte el context a l'hora de desevolupar els mecanismes de control d'accés que s'encarreguen de protegir els recursos. És a dir, el control d'accés al medi ha de ser depenent del context. Precisament per això els models tradicionals de control d'accés no són els adequats en un entorn com PROSES, ja que estan dissenyats per dur a terme el control d'accés en entorns estàtics, sense tenir en compte la informació de l'entorn al moment de pendre la decisió. Amb el creixement de les aplicacions que s'integren amb l'entorn sorgeix, doncs, la necessitat de desenvolupar nous mecanismes que tinguin en compte l'entorn a l'hora de pendre una decisió.

Un cop hem vist les motivacions que porten al desenvolupament del projecte, es parlarà dels objectius que indiquen el camí a seguir. L'objectiu principal del projecte és dur a terme la implementació d'un mecanisme de control d'accés capaç de gestionar els elements que participen a l'entorn PROSES, i poder fer-ho mitjançant la informació que podem obtenir del context.

Aquest és l'objectiu principal del projecte que ens ocupa. No obstant, podriem dividir aquesta tasca en altres objectius més concrets que ens ajudaran a assolir-lo:

- Estudiar el model ABAC (Atribute Based Control Acces) i la implementació Sun de XACML, utilitzats en entorns on és necessari el control d'accés al medi depenent del context. També cal estudiar els agents mòbils i la plataforma JADE per tenir tot el coneixement necessari previ al desenvolupament del projecte.

- Analitzar els requisits d'un mecanisme que s'encarregui del control d'accés al medi en l'entorn PROSES. També cal dur a terme un estudi de viabilitat del projecte per veure si realment és viable la seva realització.

- Dissenyar els mòduls que ens permeten dur a terme el mecanisme de control d'accés incloent la forma d'afegir informació contextual a la presa de decisió.

- Implementar el codi necessari pel funcionament d'un mecanisme a l'entorn PROSES amb les característiques del model ABAC.
- Validar el comportament del mecanisme.

La memòria està estructurada en capítols per aconseguir que sigui més fàcil de seguir. En cadascun d'aquests capítols podem trobar el seguent:

- **Capítol 2: Conceptes previs.** En aquest capítol coneixerem tot allò que serà necessari per dur a terme el nostre projecte. Veurem els mòduls utilitzats per desenvolupar el projecte, com poden ser la llibreria XACML de Sun o la plataforma Jade per la utilització d'agents. Gràcies als coneixements que tindrem després d'aquest capítol serà molt més fàcil i amè seguir el que explicarem més endavant.
- **Capítol 3: Anàlisi.** Un cop sabem tots els components necessaris per dur a terme el projecte, hem de fer un anàlisi dels mòduls que seran necessaris pel correcte desenvolupament del projecte, així com dels requisits que tindrà cadascun d'aquests mòduls i del projecte en general. També s'inclou un estudi de viabilitat del projecte per veure si la seva realització és factible.
- **Capítol 4: Disseny i implementació.** En aquest capítol tindrem dues parts: en la part de disseny es pot veure el procés de disseny del projecte, contemplant les possibles alternatives de cada mòdul del projecte i justificant la tria en cada cas; en la segona part es veurà com s'ha implementat la solució final, així com els problemes que han anat sorgint mentre es desenvolupava la solució i el mètode triat per superar aquests contratemps.
- **Capítol 5: Proves.** Quan el projecte s'ha acabat d'implementar, s'ha d'implementar un joc de proves robust i complet per comprovar que el sistema es comporta de la forma esperada. En aquest capítol descriurem les proves que hem dut a terme per aconseguir aquests objectius.
- **Capítol 6: Conclusions.** En aquest darrer capítol farem balanç del treball realitzat al llarg del projecte i també observarem les possibles línies d'ampliació que tindria aquest projecte. A més, un cop s'ha finalitzat el projecte

també veurem si hem aconseguit els objectius que ens haviem establert en un principi i, en cas que no sigui així, justificarem el perquè.

Capítol 2

Conceptes previs

Un cop hem vist les motivacions que han portat al desenvolupament del projecte, en aquest capítol veurem les bases de tot allò que s'ha necessitat per entendre millor totes les parts que conformen el projecte.

2.1 ABAC

ABAC (Atribute Based Acces Control) és un model de control d'accés que generalitza models clàssics com MAC i RBAC [ABAC] i descomposa la relació usuari - privilegi en dos passos. El primer pas es basa en que cada usuari té assignat uns atributs (assignacions usuari - atributs). Al segon pas, cada atribut té assignats uns privilegis que depenen de la tasca que ha de dur a terme (assignació atribut - privilegi). A part d'aquestes dos passos, també s'han de tenir en compte les regles de control d'accés, que són les condicions necessàries per poder aplicar un privilegi. El dinamisme de l'entorn requereix un mecanisme àgil per adaptar el sistema a les noves condicions. A més, ha de ser prou flexible per aconseguir que aquesta adaptació sigui fluida. Podem veure les relacions entre tots els grups a la figura 2.1.

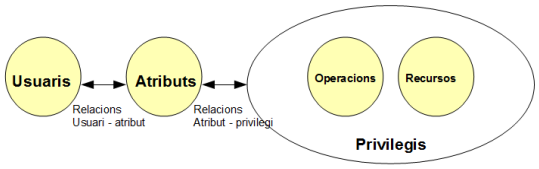

Figura 2.1: Relacions entre usuaris, atributs i privilegis

2.1.1 Relacions usuari - atribut

Les relacions usuari - atribut defineixen una relació entre el conjunt d'usuaris i els atributs. Cada usuari pot o no estar relacionat amb cada atribut, doncs un valor de 0 vol dir que no poseeix aquell atribut i per tant no hi haurà cap relació entre l'usuari i l'atribut, mentre que si té un valor de 1 aquesta relació sí que existirà. Les relacions depenen del context de l'usuari i de l'entorn on es produeixen. Les relacions usuari - atribut es defineixen com:

$$A_{UA} : U \times A \rightarrow \{0, 1\}$$

2.1.2 Relacions atribut - privilegi

L'entorn de computació i l'entorn físic poden condicionar les relacions atribut - permissos per aconseguir adaptar el sistema a l'actual estat del context. A ABAC, les relacions atribut - privilegi es defineixen com:

$$A_{AP} : A \times P \rightarrow \{0, 1\}$$

Com passava anteriorment, les relacions rol - privilegi també tenen un valor que pot ser 0 o 1, on 0 indica que un atribut no té cap relació amb aquest privilegi i 1 indica que sempre el tindrà disponible.

2.1.3 Relacions usuari - privilegi

Amb la combinació d'aquestes dues relacions obtenim finalment la relació entre usuari i privilegi, on veurem quina tasca pot realitzar usuari del sistema i en quins

casos. Un cop sabem els privilegis de cada usuari, les regles de control d'accés determinaran si se li pot concedir el privilegi a aquest usuari. A més, les regles de control d'accés són les que tindran en compte les condicions de l'entorn que s'han de considerar en el moment de pendre la decisió. La relació entre usuaris i privilegis és la següent:

$$A_{UP} : A_{UA} \times A_{AP} : U \times P \rightarrow \{0, 1\}$$

Per tant, l'accés serà permès si l'usuari posseeix un atribut que tingui accés al priviliegi que l'usuari destija.

2.1.4 Context

En aquest projecte, però, en les relacions anteriors hem de tenir en compte un nou element: el context. La informació del context permet oferir noves possiblitats en els mecanismes de control d'accés que abans no es contemplaven, com podrien ser la càrrega del sistema en un moment determinat o l'hora del dia en la que un treballador intenta accedir al recurs de l'empresa, aspectes que fins fa un temps no es tenien en compte però que avui en dia semblen gairebé imprescindibles si volem dissenyar un bon mecanisme de control d'accés.

2.2 XACML

XACML (eXtensible Acces Control Markup Language) [xacml] és un estàndar aprovat en febrer de 2005 basat en XML que proveeix tant un llenguatge per les polítiques com per les peticions i respostes de control d'accés. Va ser disenyat amb el propòsit de proveïr un llenguatge universal en el camp del control d'accés i així permetre la interoperabilitat entre un gran rang d'eines d'administració i autorització.

El funcionament bàsic d'un mecanisme de control d'accés que utilitza XACML és el següent: es crea una petició amb els atributs de l'usuari que demana fer una acció sobre un recurs. Comprovem aquesta petició amb la política o conjunt de polítiques que tenim per controlar l'accés a aquell recurs i generem una resposta

que ens indica si podem o no accedir a aquell recurs amb un dels següents valors: *Permit* (es permet l'accés al recurs), *Deny* (es denega l'accés), *Indeterminate* (s'ha produït algún error o falta alguna informació necessària per pendre la decisió) o *Not applicable* (la request no es pot resoldre amb els mitjans dels que disposem).

Les característiques més significates de l'estàndar XACML són les següents:

- **Estàndar.** L'avantatge principal d'utilitzar un llenguatge que ha esdevingut un estàndar és que aquest ha estat revisat per una àmplia comunitat d'usuaris i experts, així et pots estalviar molts maldecaps corregint codi o tenint que crear el teu propi llenguatge. A més, gràcies a que és el llenguatge més utilitzat en aplicacions amb control d'accés serà molt més senzill integrar el projecte amb altres aplicacions en cas que sigui necessari.

- **Genèric.** XACML no està dissenyat per funcionar en un entorn particular o sobre un tipus de recurs específic, sino que està pensat per correr sobre qualsevol entorn, recurs o acció. Una mateixa poliítica es pot fer servir en aplicacions de molts tipus. Aquesta característica també facilita la gestió de polítiques.

- **Versàtil.** El llenguatge bàsic de XACML es pot extendre de diverses maneres, tot i que en la majoria de casos no farà falta. L'estàndard dona suport a una àmplia varietat de tipus de dades, funcions i regles per combinar algoritmes, regles, polítiques,... A més, hi ha grups de desenvolupadors que han desenvolupat altres estàndards que treballen en extensions que permetin combinar XACML com, per exemple, SAML (Security Assertion Markup Language) [saml] i LDAP (Lightweight Directory Acces Protocol) [ldap], aconseguint d'aquesta manera ampliar encara més el ventall d'ús de XACML.

- **Distribuït.** Aquest llenguatge permet escriure una política que faci referència a altres polítiques que poden estar a qualsevol altre lloc. Així, no cal que treballem amb una sola política que segons en quines aplicacions podria ser massa llarga i difícil d'entendre, sino que podem treballar amb conjunts de polítiques de la forma que nosaltres considerem més adient, i XACML ens

proveeix mètodes per poder combinar correctament les diverses polítiques que tenim.

2.2.1 Implementació de Sun XACML

Sun Microsystems [sun], com a membre actiu del grup d'estàndards OASIS [oasis], va decidir fer la seva pròpia implementació de XACML amb l'objectiu d'accelerar la seva adopció com a model en els mecanismes de control d'accés depenents del context. La implementació va ser programada en codi obert amb el llenguatge Java.

Aquesta implementació ens proporciona molts elements necessaris pel control d'accés. Ens ofereix un format per fer les peticions de control d'accés, les respostes a aquestes peticions i també tot el necessari per implementar una política. A més, ens dona ja programats mòduls d'una gran complexitat com podrien ser el Policy Enforcement Point (PEP), el Policy Decision Point (PDP) o el Policy Information Point (PIP) que són el centre de qualsevol mecanisme de control d'accés i que veurem amb més detall en seccions posteriors.

2.2.2 Polítiques de control d'accés

Les polítiques són imprescindibles a l'hora de prendre una decisió de control d'accés ja que és el lloc on es poden veure les condicions que hem establert per permetre l'accés. Cada política consta de regles que són les que ens indicaran si permetem l'accés o no a un determinat recurs.

Per cada regla podem tenir uns objectius, unes condicions i un efecte. Els objectius ens determinen qui vol realitzar l'acció, sobre quin recurs i quina acció vol dur a terme. Les condicions ens indiquen quins són els valors que han de prendre els objectius per poder prendre la decisió de control d'accés. Finalment, l'efecte ens indica si se'ns ha otorgat o no el permís per dur a terme l'acció que es volia realitzar.

Però, quina regla té preferencia en cas que poguem aplicar més d'una? Per solucionar aquest problema tenim els algoritmes per combinar regles, que ens per-

meten donar prioritat a unes regles o unes altres segons quin sigui el seu resultat. Els algoritmes més habituals que trobem són *permit-overrides* i *deny-overrides*, que dónen prioritat a aquelles regles que ens han permés / denegat l'accés. Segons com volem que sigui de restrictiva la política escollirem l'algorisme que trobem més adequat. En aquest cas, per defecte deneguem l'accés i si es compleixen les condicions d'una regla llavors permetem l'accés.

2.3 Agents mòbils

Tot i que no hi ha una definició única d'agent, ja que depén del domini de l'aplicació en la que es treballa, una definició bàsica podria ser que un agent és una entitat autònoma, reactiva i proactiva en funció d'uns objectius.

El gran potencial que proporcionen els agents és el poder que tenen d'interactuar entre ells. Podem trobar els agents a plataformes o entorns on poden collaborar per aconseguir els seus objectius. A aquests entorns on poden conviure els agents se'ls anomena sistemes multiagents. La col·laboració entre aquests agents dins d'un sistema es realitza mitjançant llenguatges propis del sistema multiagent, i això provoca que si no hi hagués un estàndar no fos possible la col·laboració amb agents d'altres sistemes. Amb aquesta idea es desenvolupen protocols que intenten convertir-se en norma per a la comunciació entre agents, com KQML (Knowledge Query Management Language) o el més recent ACL de FIPA (Foundation for Intelligent Phisycal Agents) [fipa].

Els agents mòbils afegeixen la possibilitat de traslladar-se entre màquines i són de gran utilitat per la informàtica distribuïda ja que ens dóna solució a problemes que sense ells no era possible solucionar.

2.3.1 Plataforma Jade

Hem comentat que els agents mòbils es mouen a través de plataformes. Les plataformes actuen com contenidors per controlar les accions que realitza un agent sobre la màquina i restringir els seus comportaments. Gràcies a això evitem que un agent realitzi accions no desitjades sobre la màquina on està la plataforma. A

més d'aquesta protecció, la plataforma també s'encarrega d'oferir a l'agent una interfície pel servei de missatges i el servei que els permet moure's entre diferents plataformes.

JADE (Java Agent Development Framework) és un entorn de desenvolupament d'agents que permet desenvolupar sistemes i aplicacions amb agents complint els estàndars aprobats per la FIPA. Les aplicacions basades en Jade estan formades per un conjunt d'agents, que són els encarregats de dur a terme la funcionalitat que volem que tingui la nostra aplicació. Aquest entorn de desenvolupament ens ofereix l'abstracció de les classes Agent i Behaviour (tasca executada per un agent), mobilitat dels agents a través d'una gran varietat de dispositius, comunicació punt a punt entre els agents i mecanismes de descobriment que permet als agents trobar-se.

L'entorn Jade està escrit en el llenguatge de programació Java i està format per Java packages que permeten als programadors treballar amb parts de codi que ja funcionen i interficies abstractes que es poden modificar per adaptar l'entorn a cada aplicació.

2.4 ACL

Els missatges ACL (Agent Communication Language) [acl] són l'estàndar de FIPA per dur a terme la comunicació entre agents. Aquests missatges contenen un conjunt d'un o més paràmetres. Per a que la comunicació entre els agents sigui efectiva aquests paràmetres poden variar segons la informació que volem transmetre. Només hi ha un paràmetre obligatori en tots els missatges ACL, *performative*, mentre que els altres poden apareixer o no segons el missatge. Són els següents:

- **Performative.** És l'únic paràmetre obligatori en tots els missatges ACL. Ens indica quin tipus de comunicació volem establir entre els agents que participen a la comunicació.

- **Sender.** Ens indica quin agent ha enviat el missatge ACL. Tot i que no és obligatori apareix en la majoria de missatges, ja que en la majoria de casos

aquest voldrà rebre la resposta de l'altre agent. Si, per contra, l'agent vol mantenir en secret la seva entitat aquest paràmetre no estarà present en el missatge ACL.

- **Receiver.** Aquest paràmetre ens mostra a quin o quins agents va dirigida la comunicació. Com passa amb l'emissor, tot i no ser obligatori aquest paràmetre apareix en la majoria dels casos.

- **Reply to.** En alguns casos ens pot interesar que la resposta d'un missatge ACL no vagi dirigida a l'agent que ha enviat el missatge, sino cap a un altre. En aquests casos farem servir el paràmetre *reply to.*

- **Content.** Aquí podem trobar el contingut del missatge.

- **Language.** Indica el llenguatge del contingut per a que el receptor sigui capaç d'interpretar-lo.

- **Encoding.** Indica la codificació del contingut.

- **Ontology.** En cas d'haver fet servir ontologies, mitjançant aquest paràmetre indiquem a l'agent quines hem fet servir per a que pugui extreure el contngut del missatge.

- **Protocol.** Defineix el protocol que farà servir l'emissor per enviar el missatge.

- **Conversation identifier.** Quan es produeix una conversa entre els agents, aquest paràmetre ens ajuda a mantenir l'ordre correcte.

- **Reply with.** Paràmetre que introdueix l'expressió que es farà servir per respondre el missatge.

- **In reply to.** Quan en una conversa es produeix un *reply with*, la resposta contindrà la expressió en el paràmtre *in reply to.*

- **Reply by.** Amb aquest paràmetre especifiquem el temps o la data en que s'acceptarà la resposta del missatge que s'ha enviat.

2.5 Ontologia

Com hem comentat anteriorment, una de les accions més comuna entre dos agents és l'intercanvi d'informació entre ells per aconseguir d'aquesta manera els seus objectius. Tot i que aquest acte pot semblar fàcil, hi ha aspectes a tenir en compte per a que la comunicació sigui adequada.

Per entendre millor que és una ontologia, recordarem quins són els elements principals d'un procés comunicatiu. En una comunicació tenim l'emisor, el receptor, el canal, el llenguatge i una forma per representar el missatge que volem transmetre. En aquest cas, els agents actuen com a emissors i receptors. La plataforma Jade és la que ens proporciona l'accés al canal sense que el programador s'hagi de preocupar d'aquest aspecte de la comunicació. El problema a l'hora de transemtre informació entre dos agents, doncs, es troba en el llenguatge i en la forma que representem el missatge.

En qualsevol acte comunicatiu si el llenguatge que parlen les dues parts no és el mateix la comunicació serà infructífera. Però no només és necessari que les dues parts comparteixin el mateix llenguatge, també cal que les dues parts tinguin un coneixement de com és l'estructura del missatge per a que la comunicació sigui útil.

Aquest és l'objectiu principal d'una ontologia, fer que les dues parts de la comunicació siguin capaces d'entendre la informació que estan intercanviant. Quan un agent A es vol comunicar amb un agent B, aquest envia un missatge ACL amb la informació I que vol transmetre. Aquesta informació està representada amb el llenguatge que nosaltres haguem escollit (en el cas de Jade, podem codificar aquest missatge en SL o en LEAP) i l'enviem com una cadena de caràcters. Si aquesta informació no és massa gran, l'agent B podrà tractar amb ella sense massa problema. Però en el moment que augmentem la informació que s'envia des de l'emissor, no és viable tractar amb una cadena de milers de caràcters per extreure la informació que es necessita. Amb una ontologia aconseguirem tractar aquesta informació de forma rapida i eficient, com podem veure a la figura 2.2.

Per aconseguir que Jade pugui comprovar les característiques semàntiques d'una expressió és necessari que classifiquem tots els elements que la conformen

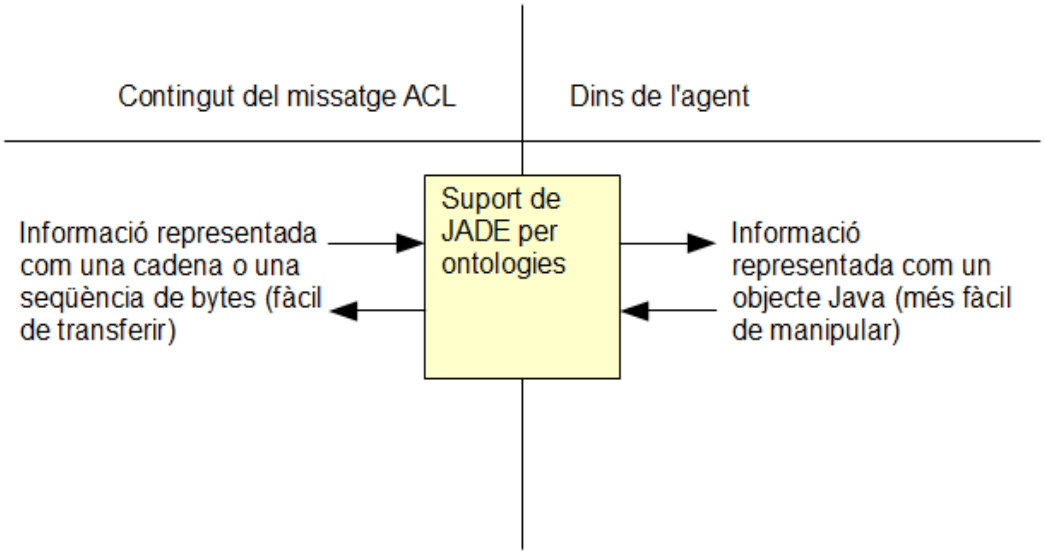

Figura 2.2: Canvi d'informació

en funció de la seva semàntica. Per això diferenciem entre termes i predicats.

Predicats (o fets). Són expressions que ens donen informació sobre l'estat del món i es poden respondre amb veritat o fals. Per exemple, si tenim el predicat treballa per amb els termes persona i companyia: Joan treballa per Microsoft.

Termes (o entitats). Són expressions que ens identifiquen entitats que existeixen en el món i dels quals els agents ens donen informació. Es poden classificar en:

- **Conceptes.** Expressions que representen entitats amb una estructura complexa. Els conceptes per si sols no tenen cap sentit en un missatge ACL. Normalment els trobem junt amb predicats o altres termes que els hi donen significat.

- **Accions de l'agent.** Conceptes especials que indiquen accions que poden dur a terme alguns agents.

- **Primitives.** Expressions que indiquen entitats atòmiques com cadenes o

enters.

- **Agregats.** Expressions que serveixen per expressar entitats formades per grups d'altres entitats.
- **Expressions identificatives referencials.** Expressions que identifiquen entitats per les quals es compleix un predicat.
- **Variables.** Indiquen un element genèric que no coneixem a priori.

Un cop vistos i entesos els conceptes necessaris pel correcte seguiment del projecte, en el següent capítol analitzarem les necessitats que han de complir les diferents parts que conformen el projecte.

Capítol 3

Anàlisi

En aquesta part de la memòria es tracten els grans blocs dels que consta el projecte, la seva funcionalitat i els requisits que ha de complir. Començarem parlant fent un anàlisi exhaustiu de l'entorn PROSES amb totes les caracterísitiques que el defineixen i un estudi de les aplicaciones que es podrien dur a terme i els seus aspectes de seguretat i control d'accés. Després farem un estudi de viabilitat de tots els aspectes del projecte, incloent un pressupost amb el seu cost aproximat. Finalment realitzarem una planificació temporal de tot el projecte per marcar-nos uns plaços de temps raonables per enllestir tota la feina.

3.1 Anàlisi de l'entorn PROSES

Com hem pogut veure a la introducció, l'entorn PROSES és un entorn bastant particular on les característiques d'un mecanisme de control d'accés poden ser bastant diferents respecte a un altre entorn. Per això és necessari un estudi a fons de les característiques d'aquest entorn que poden afectar al control d'accés, i més concretament a l'aplicació de correu electrònic

3.1.1 Aplicació de correu electrònic

A l'actualitat, si volem enviar un e-mail desde el nostre portàtil quan som a un avió en vol, hem de fer servir un enllaç per satè l·lit que té un preu desorbitat.

No obstant, PROSES ens oferirà la possibilitat d'enviar e-mails sense cap tipus de cost addicional, aprofitant la comunicació que es pot establir entre els avions i la torre de control. La idea bàsica consistiria en que si algú vol enviar un e-mail, en el moment en que el nostre avió es creua amb un altre li passem el mail si el temps de vol per arribar al seu destí és inferior al nostre. Fariem aquest pas successivament fins arribar a la torre de control, connectada a Internet que seria l'encarregada de fer arribar el mai a la xarxa.

Cal tenir en compte, però, aspectes de seguretat a l'hora de fer la transmissió de les dades entre els diferents elements. Pot semblar que no és un entorn on hi hagi massa atacs, però també cal tenir en compte l'increment del nombre de vehicles aeris no tripulats (UAV) que incrementarien les possibilitats d'aquests tipus d'atacs. Primer de tot, ens hem de plantejar quins són els rols dels diferents elements que intervenen en aquesta comunicació. A simple vista, a la comunicació participen el passatger que envia el correu, els avions que tenim al cel i les torres de control. Però algun d'aquest grup es pot dividir per tenir elements més concrets.

Per una banda, podriem diferenciar la classe dels passatgers de l'avió (Quality of Service, QoS). Per posar un exemple, els passatgers que viatgessin en classe business tindrien preferència respecte als de classe turista. Així, els mails es guardarien en una cua prioritaria i s'enviarien a un altre avió respectant aquesta prioritat. En un principi no hi hauria d'haver problemes per poder enviar tots els correus de la cua al nou enllaç, doncs els mails no tenen un tamany massa gran i amb l'ample de banda disponible no hi hauria d'haver cap problema. Cal tenir en compte, però, que això no forma part d'aquest projecte i seria l'aplicació la que s'hauria de preocupar d'aquest aspecte.

Després també tenim el problema de que l'avió amb el que ens trobem fos d'una altra companyia. Si tinguessim un acord de co·laboració amb aquesta companyia podriem enviar les dades sense cap problema, però en el cas contari no podriem enviar les dades i hauriem d'esperar a trobar-nos un altre avió. En l'aspecte de la seguretat és bastant clar que la informació s'ha d'enviar xifrada entre els diferents nodes. Per poder encriptar la informació i garantir l'autenticitat de

les dades es podria fer servir una infraestructura de clau pública. Una opció seria que cada companyia aeria disposés d'un parell de claus (la pública i la privada). Cada companyia tindria al seu repositori la informació de les seves claus i la de les companyies amb les que co·labora. Així, al rebre les dades d'una companyia que coneix es comprovaria l'autenticitat del missatge i no hi hauria cap inconvenient. En cas contrari, no es podrà comprovar l'autenticitat del missatge i es descartarà.

No obstant, en una infraestructura de clau pública el procés d'intercanvi de claus entre les dues parts de la comunicació pot resultar massa costós en termes de temps en aquest entorn. L'alta velocitat dels avions pot provocar que la comunicació només sigui possible en un interval de temps molt petit i fer impossible l'intercanvi de claus i la transimissió de dades en aquesta fracció de temps. Per aquest motiu, potser seria interessant utilitzar algun altre mecanisme de seguretat. Aquest seria utilitzar una única clau de seguretat per totes les companyies per xifrar i desxifrar els missatges. Aquest esquema, però, no seria tan segur com l'altre ja que la seguretat de tot el sistema es basaria únicament en un sol element. S'hauria d'estudiar més a fons quin és el temps que dura aproximadament l'enllaç entre dos avions per poder establir quin dels dos sistemes escauria millor en aquesta aplicació.

A la figura 3.1 es pot observar millor com serien les comunicacions necessaries per enviar un mail des d'un avió:

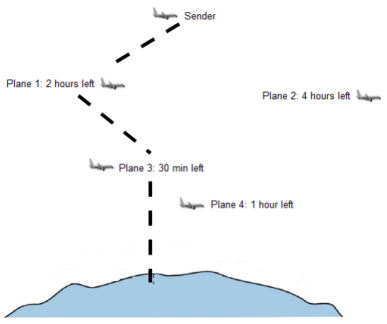

Figura 3.1: Difusió de correu entre nodes

3.1.2 Context

En aquest projecte un dels objectius és aconseguir implementar un sistema de control d'accés i, per tant, cal fer un anàlisi exhaustiu sobre les característiques de l'entorn PROSES. Cal destacar que aquest anàlisi engloba tota la informació d'entorn que es podria fer servir en un mecanisme de control d'accés de qualsevol aplicació, però en l'aplicació del e-mail no són necessàries totes aquestes característiques del context.

Context d'usuari

El context d'usuari és la informació rellevant i mesurable per l'aplicació relacionada amb l'usuari. A l'entorn PROSES, les següents característiques defineixen el context d'usuari:

- Tipus de passatger. Un passatger és usuari business si ha pagat la diferència i el personal de vol i el pilot tenen els mateixos privilegis que un usuari business.

- Cada usuari té un límit a l'hora d'utilitzar les aplicacions. Si sobrepassa aquest límit, es denega l'accés a les aplicacions.

Context de sistema

L'entorn de sistema es refereix a quelcom rellevant i mesurable relacionada amb el sistema. En aquest entorn podem trobar la següent informació:

- Utilització del buffer. Es poden fer servir les aplicaciones si el buffer no està ple.

- Neteja del buffer. Es pot netejar el buffer un cop hem enviat el correu i en el moment en que l'avió aterra.

Context físic

El context físic es pot definir com qualsevol informació mesurable i rellevant relacionada amb l'entorn físic o virtual on són el sistema i els usuaris. En aquest cas tenim les següents característiques:

- Temps restant per aterrar.
- Temps que fa que l'avió va enlairar-se.
- País que sobrevola l'avió en aquell moment.
- País d'origen de l'avió.
- Meteorologia.

3.2 Estudi de viabilitat amb pressupost

L'objectiu d'aquest estudi de viabilitat és fer un estudi de les característiques del projecte des de diferents punts de vista, i d'aquesta manera avaluar si és possible la seva realització.

3.2.1 Estudi tècnic i operatiu

Des del punt de vista tècnic i operatiu el projecte s'adapta perfectament a l'entorn PROSES, que pel seu funcionament intrínsec està totalment adaptat al treball amb agents mòbils. A l'actualitat existeixen implementacions del model ABAC, de la mateixa manera que existeixen aplicacions que treballen en xarxes DTN [rfc 4838]. Per acabar, també és veritat que les aplicacions amb agents mòbils a la plataforma Jade estan bastant esteses. Per tant, aquest projecte és viable des del punt de vista tècnic i operatiu.

3.2.2 Estudi legal

La LOPD (Llei Orgànica de Protecció de Dades) [lopd] afecta al nostre projecte en l'entorn PROSES, ja que treballem amb dades personals. Per tant, hem de tenir en

compte aquesta llei a l'hora de tractar tota aquesta informació. Totes les llicències del programari que utilitzarem al projecte tenen vigència. Així, el projecte també es viable legalment.

3.2.3 Pressupost

Segons el IV Conveni per al personal laboral de les universitats públiques catalanes, i en condició de treballador de Grup II (Personal amb Titulació Universitària de grau mitjà, o amb capacitat suficientment provada en el concurs de cobertura de plaça vacant), el salari anual del projectista seria aproximadament d'uns 18.000 euros. Tenint en compte que la duració del projecte és d'un semestre i que la jornada laboral seria aproximadament mitja jornada (això suposaria un treball d'aproximadament 270 hores) el cost del projecte seria d'uns 4.500 euros. En un projecte amb la magnitud de PROSES, aquest projecte és viable des del punt de vista econòmic. Pel que fa al tema de software i les seves llicències corresponents, tot el codi amb el que es treballa és obert, motiu pel qual no suposa cap cost.

Per tant, el cost aproximat del projecte rondaria els 4500 euros.

La conclusió després de tots els aspectes que hem tractat en l'estudi de viabilitat és que el projecte és viable.

3.3 Planificació temporal del projecte

Abans de començar la planificació del projecte és important recordar que aquesta planificació va ser un dels primers passos en aquest projecte i l'ordre dels continguts a la memòria no és el mateix que el que s'ha seguit durant el desenvolupament com podrem veure a continuació.

En primer lloc, hem de realitzar un estudi del context en el que es realitzarà el projecte. Per tenir del tot clar els requeriments i les especificacions del mecanisme, primer hem de saber en quin entorn es mourà. En el nostre cas l'entorn

és PROSES, del qual cal fer un estudi exhaustiu de les característiques de l'entorn i dels elements que el conformen. Per afrontar aquesta tasca, hem de fer un estudi del model de control d'accés que utilitzarem, ABAC, per després poder implementar-lo de forma coherent. Però encara no hem acabat amb les tasques previes al disseny i desenvolupament del nostre mecanisme, doncs necessitem també conèixer a fons el punt d'unió entre PROSES i ABAC, que en el nostre cas seran els agents i la plataforma JADE.

Un cop hem estudiat tots els elements necessaris pel desenvolupament del nostre mecanisme, ja podem passar al seu disseny. Aquesta tasca és crucial en el desenvolupament del nostre projecte, doncs segons la qualitat del disseny després trobarem més o menys problemes al moment de la implementació. La part final del disseny i la implementació coincideixen al temps, doncs sempre sorgeixen problemes que al disseny no havíem previst i que no veiem fins al moment de realitzar la programació del nostre mecanisme. Això inclou la realització de proves de caixa negra i caixa blanca per comprovar que el comportament del nostre mecanisme és el que nosaltres esperàvem.

Després de totes aquestes tasques, l'últim pas de tots, però no menys important, és la comprovació que tot funciona correctament i no es produeixen errors inesperats.

Un cop explicades les fases del nostre projecte, mitjançant el diagrama de Gantt de la figura 3.3 podrem veure la cronologia de cadascuna de les tasques del nostre projecte i les relacions de precedència entre elles.

Nombre	Fecha de inicio	Fecha de fin
1. Realització Informe Previ	8/01/10	15/01/10
2. Estudi del model ABAC i la implementació Sun de XACML	15/02/10	23/02/10
2.1 Anàlisi model RBAC	15/02/10	16/02/10
2.2 Model de Control d'Accés ABAC	16/02/10	19/02/10
2.3 Estudi de la implementació Sun de XACML	19/02/10	23/02/10
3. Anàlisi Entorn PROSES	23/02/10	28/02/10
3.1 Anàlisi Entorn SESAR	23/02/10	24/02/10
3.2 Estudi Xarxes DTN	24/02/10	25/02/10
3.3 Relació PROSES - ABAC	25/02/10	28/02/10
4. Estudi AM / JADE	28/02/10	5/03/10
4.1 Agents mòbils	28/02/10	1/03/10
4.2 Plataforma JADE	1/03/10	5/03/10
5. Disseny i Implementació Mecanisme	5/03/10	21/04/10
5.1 Disseny	5/03/10	7/04/10
5.2 Implementació	22/03/10	21/04/10
6. Proves de validació	25/04/10	2/05/10
7. Realització de la Memòria	1/05/10	31/05/10
Fita 1: Lliurament Informe Previ	13/01/10	15/01/10
Fita 2: Sol.licitud Lectura	16/05/10	31/05/10
Fita 3: Entrega Memòria	17/06/10	22/06/10
Fita 4: Lectura	28/06/10	14/07/10

Figura 3.2: Planificació temporal del projecte

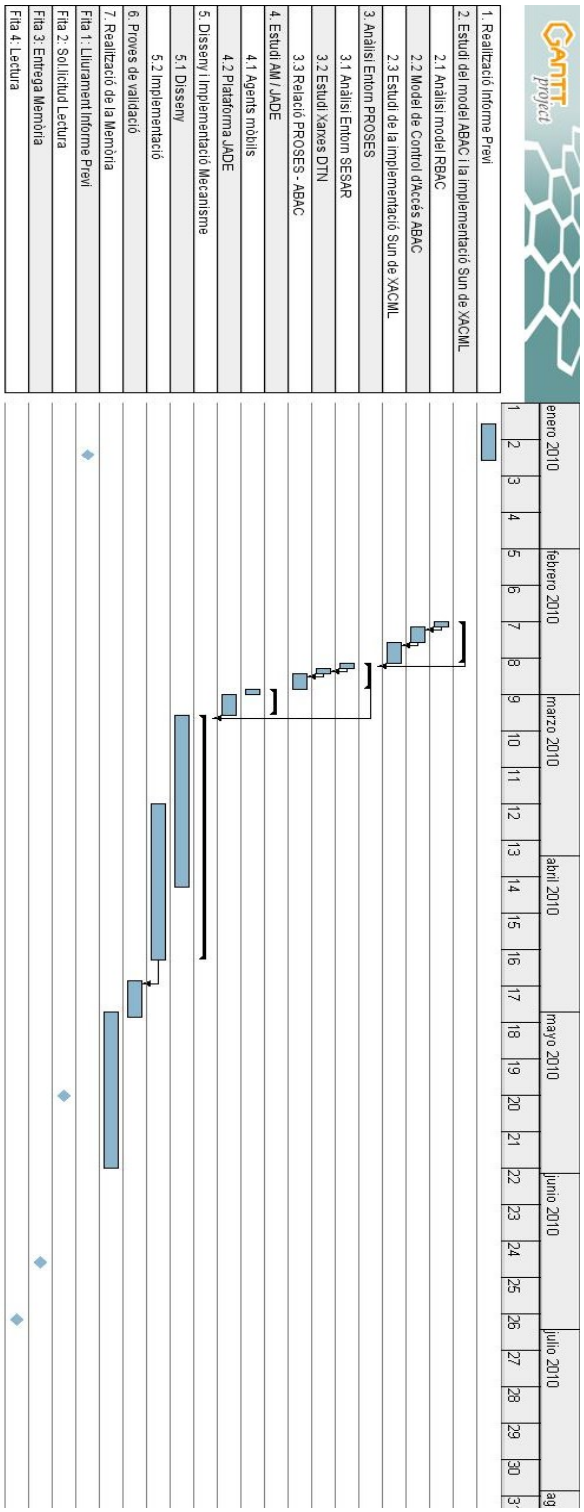

Figura 3.3: Diagrama de Gantt

Capítol 4

Disseny i implementació

Un cop finalitzat l'anàlisi de l'entorn on es desenvoluparà el projecte, el següent pas és fer el disseny dels mòduls que el conformen i posteriorment implementar-lo correctament.

Per començar parlarem de com està estructurat el projecte en general per veure com interactuen els diferents mòduls que el conformen entre ells i així tenir una visió més global del funcionament del programa, per després explicar aquests mòduls independentment i de forma més detallada. Per finalitzar tractarem com hem fet la política de control d'accés en aquest entorn.

4.1 El mecanisme de control d'accés

En aquest apartat dissenyarem i implementarem el mòdul principal del projecte, el mecanisme de control d'accés. Aquest mòdul és el que s'encarrega de prendre la decisió de permetre l'accés o denegar-lo a partir de la informació que rep de l'aplicació de PROSES. Com ja hem apreciat en la secció de conceptes previs, per implementar aquest mecanisme es fa servir XACML. A la figura 4.1 podem veure el diagrama de classes del projecte.

Abans de passar a implementar cada mòdul, però, ens hem d'aturar un moment a pensar com estaran relacionats entre ells. Recordem els mòduls que hem establert a la fase d'anàlisi: el primer de tots d'encarrega de generar la petició

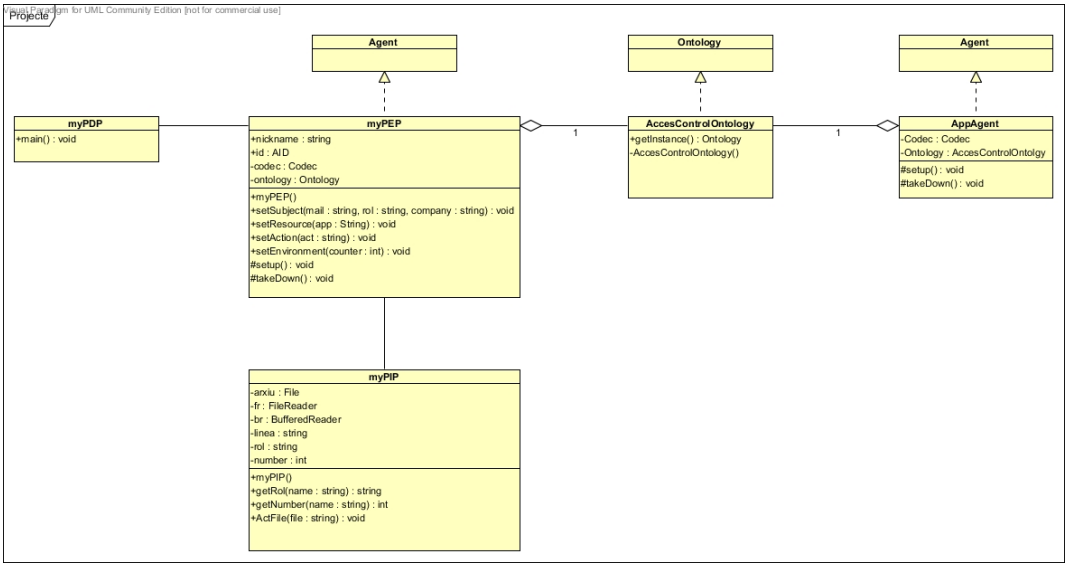

Figura 4.1: Diagrama de classes del projecte

o com anomenarem a partir d'ara per utilitzar una nomenclatura més adequada per un mecanisme de control d'accés, Policy Enforcement Point(PEP); el mòdul que s'encarrega de gestionar i recollir la informació necessària s'anomena Policy Information Point(PIP); i per acabar, el mòdul encarregat de pendre una decisió i informar de quina ha estat rep el nom de Policy Decision Point(PDP). A la figura 4.2 podem veure un diagrama de flux de dades amb el qual veiem com interactuen els diferents mòduls entre ells.

No obstant, a l'hora de fer aquest disseny va haver varies alternatives que es van descartar abans d'escollir aquesta. La primera d'elles contemplava la possiblitat que el PIP interactués amb el PDP enllloc del PEP. Aquesta possibilitat es va descartar perquè la implementació Sun de XACML dona moltes més facilitats per fer interactuar el PEP i el PIP, aconseguint així que tota la informació necessària ja estigui inclosa en la request i evitant tenir que buscar-la posteriorment. Un altre dubte va ser saber com capturar el context per fer-lo servir a l'hora de pendre una decisió de control d'accés. La primera possiblitat que es va contemplar va ser capturar el context desde les politíques, però aquesta opció es va descartar ja que no estava molt clar com es podia implementar. La possibilitat que es va triar final-

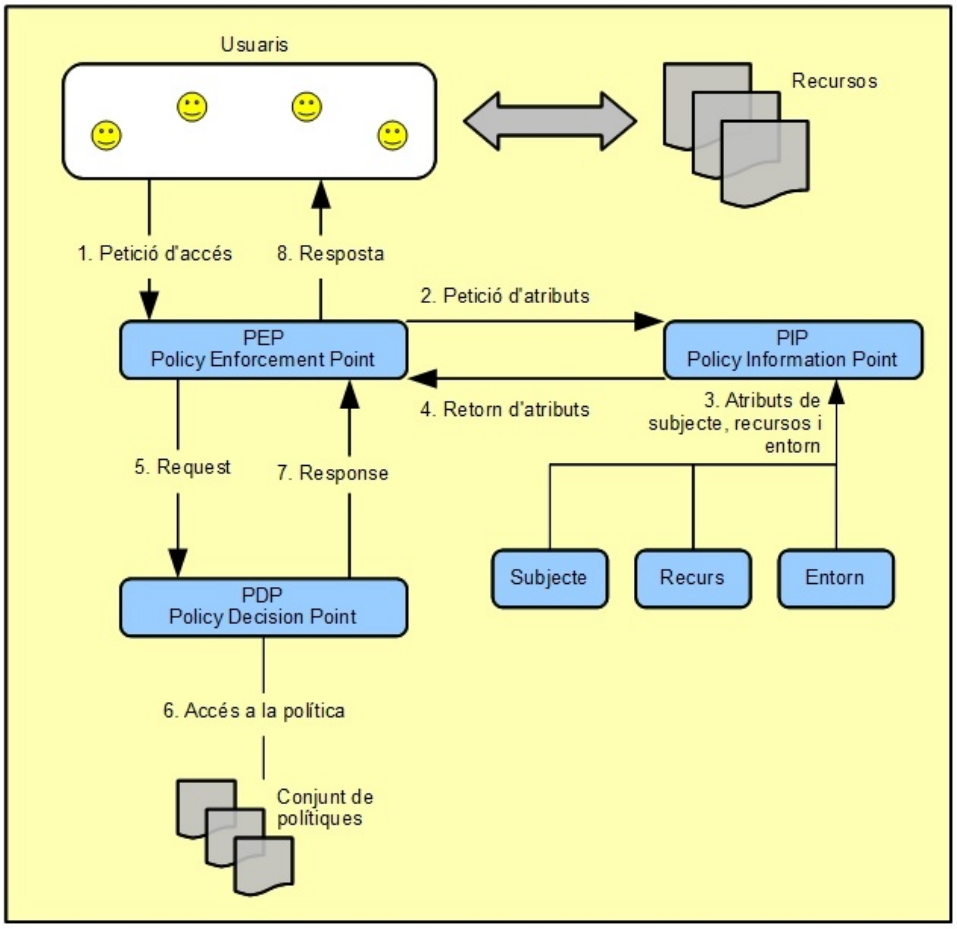

Figura 4.2: Diagrama de flux de dades del programa

ment va ser obtenir la informació de l'entorn desde el PIP, ja que la implementació Sun ens ofereix aquest avantatge i així també podem incloure la informació del context a la petició.

4.1.1 Interfície entre l'aplicació PROSES i el mecanisme de control d'accés: l'agent

En aquest projecte l'agent és necessari per poder dur a terme la comunicació entre l'aplicació i el mecanisme de control d'accés. Tot i que l'agent forma part del Policy Enforcement Point(PEP), el tractarem a part per veure millor les seves característiques i entendre el seu funcionament.

L'agent és un component imprescindible per poder comunicar el mecanisme de control d'accés amb les aplicacions de l'entorn PROSES. Totes les aplicaciones de l'entorn funcionen amb agents mòbils i, per tant, per poder facilitar la comunicació amb aquests elements necessitem que el mòdul del projecte que interactui amb els agents mòbils també sigui un agent.

En aquest cas, però, no cal que l'agent del mecanisme del control d'accés sigui un agent mòbil. Les dades que necessita l'agent per dur a terme la seva funció les rep de l'aplicació de l'entorn PROSES i d'aquesta manera queda fora de lloc que l'agent inclogui la característica de la mobilitat.

La funció bàsica de l'agent en aquest projecte és actuar com a interfície de comunicació entre l'aplicació de PROSES i el mecanisme de control d'accés. Com hem comentat anteriorment, les aplicacions en aquest entorn treballen amb agents mòbils, de manera que si volem establir una comunicació amb ells necessitem un agent per a poder realitzar el intercanvi de informació amb missatges ACL.

A la plataforma Jade els agents contenen una sèrie de comportaments anomenats behaviours. Els behaviours contenen les accions que realitza l'agent dins d'una plataforma. En aquest projecte, però, no és necessari la utilització d'aquests components de Jade ja que el seu comportament és garirebé trivial, només s'encarrega de rebre un missatge ACL de l'aplicació i transmetre la informació al mecanisme de control d'accés.

La restricció principal d'aquest agent és que no té implementada la característica de la mobilitat. Un dels motius de la utilització d'agents mòbils és la impossibilitat de traslladar un volum d'informació massa gran entre dos màquines, motiu pel qual la informació es queda a la màquina i el que passa entre els diferents nodes és *l'algorisme*. Com en el nostre cas la informació ens arriba de l'aplicació, no cal que implementem la funcionalitat de la mobilitat en el nostre agent.

4.1.2 Policy Enforcement Point

Aquest mòdul fa d'interfície entre els subjectes i el sistema de control d'accés. Les peticions dels usuaris arribaran al PEP, que transmetrà aquesta petició al sistema. Quan el sistema hagi resolt la petició aquest mòdul serà l'encarregat de comunicar a l'usuari si pot accedir al recurs que ha demanat. Al ser una interfície entre una aplicació que treballa amb agents mòbils i un mecanisme que fa servir una implementació Sun per a realitzar el control d'accés, aquest mòdul ha d'extendre la classe Agent de la plataforma Jade, utilitzar les llibreries necessàries per poder utilitzar les ontologies i a més també ha d'incloure les llibreries XACML per poder generar la petició(request).

Les peticions tenen quatre camps principals que hem d'omplir: en el primer d'ells hem de posar la informació relacionada amb el subjecte que fa la petició d'accés; el segon camp indica l'acció que volem realitzar; el tercer ens fa saber sobre quin recurs es vol realitzar l'acció esmentada anteriorment; i finalment introduïm la informació relacionada amb l'entorn. Les peticions tenen aquest format establert per la OASIS i les podem generar gràcies a la implementació Sun de XACML.

Per a poder obtenir la informació necessària que ens arriba des de l'agent mòbil de l'aplicació de correu electrònic aquest mòdul ha de quedar a l'espera. És a dir, es llançarà aquest mòdul a la plataforma Jade i romandrà inactiu fins que no arribi un agent que iniciï una comunicació.

Un cop arriba l'agent mòbil, per poder crear la petició es necessita informació dels atributs del subjecte, el recurs al qual es vol accedir, l'acció i altres dades relacionades amb l'entorn. Tots aquests camps de la request han de seguir un for-

mat estàndar, sinó el programa fallarà en temps d'execució ja que no serà capaç de reconèixer el seu tipus. Per aconseguir-ho, la implementació Sun de XACML ens proporciona l'element *Attribute*, en el qual podem trobar el valor de l'atribut i la informació relacionada amb aquest (com, per exemple, l'identificador per identificar aquest atribut i el seu tipus).

No obstant, ens sorgeix un problema relacionat amb la informació necessària per generar aquesta petició: de l'aplicació no rebem tota la informació requerida per generar-la. En conseqüència, hem d'obtenir aquestes dades d'un altre lloc. Aquest fer provoca la necessitat de la creació d'un mòdul que s'encarregui de buscar la informació que necessitem a partir de les dades que rebem de l'aplicació, el mòdul PIP, que ens retornarà la informació restant.

Un cop tenim tota la informació ja podem crear la request gràcies a la classe *RequestCtx*. Aquesta request va directament fins al mòdul PDP que serà l'encarregat de prendre la decisió.

4.1.3 Policy Information Point

Aquest mòdul s'encarrega de subministrar al PEP tota la informació que aquest necessita per generar una petició. La informació normalment es presenta en la forma d'atributs del subjecte, els recursos i l'entorn.

En un fitxer tindrem emmagatzemada una llista amb la informació relacionada de cada usuari. Aquest mòdul rebrà el nom d'usuari des del PEP i retornarà tots els atributs que aquest necessita per resoldre una peticio. En l'aplicació del correu electrònic, del fitxer només necessitem el grup al que pertany l'usuari (turista o business) i també el número de cops que l'usuari ha fet servir aquesta aplicació. Cada cop que es consulti la informació de l'usuari d'incrementarà aquest contador. Aquest comptador és imprescindible per evitar un abús de qualsevol usuari de l'aplicació de correu electrònic.

4.1.4 Policy Decision Point

El PDP és el motor que resol les peticions d'accés dels usuaris. Qualsevol mecanisme de control d'accés té el seu component principal a aquell que s'encarrega de decidir si permetre o no l'accés d'un subjecte a un recurs. Per poder dur a terme aquesta funcionalitat, necessita dos components imprescindibles: la petició d'accés i les polítiques que regulen aquest accés. Aquest mòdul s'ha d'encarregar de rebre la petició generada anteriorment i fer servir la política adequada per a la resolució de la petició. Un cop ha pres la decisió, envia la resposta al PEP per a que aquest pugui transmetre el resultat de la petició a l'usuari.

Per poder avaluar la petició, necessita comprovar la informació que li proporciona la petició amb la que tenim a les polítiques. El primer pas és utilitzar la classe *FilePolicyModule* que serveix per representar una col·lecció d'arxius que contenen polítiques. Després, per poder trobar totes les polítiques necessàries per a l'avaluació d'entre totes fem servir la classe *PolicyFinder*. Normalment aquesta classe la fa servir el PDP, però el seu disseny suporta el seu ús individual per permetre treballar amb serveis distribuïts o en aplicacions on només vulguis trobar la política però no aplicar-la.

Abans de poder fer servir el PDP hem de realitzar un darrer pas, que és configurar-lo correctament pel seu ús mitjançant la classe *PDPConfig*. Ara només queda evaluar la petició que hem rebut per generar la resposta a la petició d'accés.

Un cop hem pres la decisió de control d'accés, hem de fer saber al subjecte que ha generat la petició la resolució que ha pres el mecanisme de control d'accés. Aquesta resposta tindrà el format establert per OASIS que ja tenim implementat a XACML.

Tots els documents generats pel mecanisme de control han de seguir els estàndars especificats per OASIS. Aquesta és la limitació principal pel que fa al control d'accés, ja que en cas que no es seguiexin les especificacions qualsevol de les parts d'aquest mecanisme no funcionarà i per tant no es podria resoldre la petició de control d'accés.

4.2 Comunicació entre l'aplicació i el mecanisme de control d'accés

A l'hora de definir com es pot dur a terme la comunicació entre els dos agents sorgeix un problema. Podem fer que aquests agents es comuniquin fent servir simples cadenes o, per una altra banda, podem fer servir una ontologia. Quina de les dues opciones és, doncs, la millor?

Per entendre millor quina solució és la més adequada en aquest context el millor és veure un exemple. Suposem que volem enviar un e-mail des d'un avió. En aquest cas, necessitem una sèrie d'informació: el remitent del correu, el destinatari i el missatge. Si només hi haguessin aquests camps seria possible agafar la informació directament des d'una cadena, tot i que no seria el més eficient. No obstant, hi ha diversos problemes que fan que la recollida d'informació mitjançant una cadena no sigui la més adequada. Un d'ells podria ser que el remitent del missatge no és un camp simple, és a dir, que no només tenim emmagatzemat el seu nom sino que també tenim altres dades com el seu document d'identitat o la data de naixement. Quina informació del remitent necessitem per realitzar el control d'accés? Depèn de l'aplicació. En el cas d'aquest projecte sempre necessitarem la mateixa informació ja que només es realitza el control d'accés sobre l'aplicació d'enviar e-mails. Així, la data de naixement no ens cal per pendre una decisió. Però amb l'aparició de noves aplicacions si que podria fer falta aquesta dada. És evident que això complicaria molt el fet d'obtenir la informació adequada desde una cadena, ja que per una aplicació necessitariem unes dades i per l'altra les dades a tenir en compte serien unes altres. Això ho podem solucionar fent servir les ontologies, un mètode que ens permet afegir nous camps a cada classe sense que la manera d'obtenir la resta d'informació canviï gens. A més, gràcies a les ontologies trobem la informació que ens cal de forma més ordenada.

4.2.1 L'ontologia: Acces Control Ontology

Una ontologia es basa en un predicat que agrupa una sèrie de termes. Per tant, hem de decidir el predicat que serà la base de l'ontologia i crear els termes o entitats al

seu voltant. Per dur a terme el control d'accés de l'aplicació d'enviar correu ens calen 4 components: el predicat *Send* i els conceptes *User*, *E-mai*l i *Company*.

A l'hora de definir una ontologia, el primer pas consisteix en crear una instància de la classe Ontology de Jade, en la que podem trobar l'esquema que defineix l'estructura i el tipus dels predicats i els termes. Com que l'ontologia no canvia durant els cicles de l'agent és adequat que sigui declarada com una classe *singleton*. En aquesta classe trobem els esquemes de tots els elements de l'ontologia i, per tant, si modifiquem qualsevol dels predicats o termes que conformen aquesta ontologia també hem de modificar aquesta classe per a que tot sigui coherent. En cas que l'esquema i el predicat/terme no coincideixin l'ontologia no funcionarà correctament.

Però abans de poder definir aquesta classe amb els esquemes corresponents hem de decidir com són els predicats i els termes que formaran la nostra expressió. Primer trobarem els termes necessaris per expressar el que volem i després els unirem mitjançant un predicat.

En primer lloc, sembla evident que necessitem un concepte que ens indiqui qui vol fer servir l'aplicació de correu electrònic. A aquest concepte l'anomenarem *User*. Com hem comentat a la fase d'anàlisi hi haurà dades de l'usuari que podem trobar a l'ontologia mentre que la resta les recuperarem d'un fitxer. La informació que podem trobar en el fitxer és aquella que depén del context. En l'aplicació del correu electrònic, que és la que ens preocupa, aquesta context tindrà la forma de tipus de passatger. Per exemple, un passatger que ha comprat un bitllet de classe turista no tindrà els mateixos privilegis que un altre que hagi comprat un passatge de primera classe. Això comporta que no tots els passatgers tenen el privilegi de poder enviar un correu electrònic, sinó que només ho poden fer els de primera classe.

Cal tenir en compte que tot i que només s'ha implementat el control d'accés per a l'aplicació del correu electrònic, a l'hora del disseny de l'ontologia s'ha tingut en compte que aquesta hauria de ser el més general possible i, per tant, hi ha camps que en aquest cas es podrien considerar innecessaris però que serien de gran utilitat a l'hora de poder reutilitzar aquesta ontologia amb altres aplicacions.

USER

Nom	Tipus	Aparició	Comentari
name	String	Obligatori	Nom i cognoms de l'usuari que vol utilitzar l'aplicació
id	String	Obligatori	Número de document nacional d'identitat o de passaport
date	String	Obligatori	Data de naixement de l'usuari
email	String	Opcional	Email de l'usuari que fa servir l'aplicació
resource	String	Obligatori	Recurs al qual vol accedir l'usuari
action	String	Obligatori	Acció que volem realitzar sobre el recurs

Taula 4.1: Concepte User

EMAIL

Nom	Tipus	Aparició	Comentari
sender	User	Obligatori	Aquí podem trobar quin passatger és el que vol enviar el correu
destination	String	Obligatori	Direcció de correu electrònic on anirà el correu
content	String	Optional	Contingut del correu electrònic

Taula 4.2: Concepte Email

Un cop hem explicat tot el relacionat amb els camps d'aquest concepte, per aconseguir que sigui com nosaltres volem necessitem que tingui la forma que veiem a la taula 4.1.

Un cop hem definit qui vol realitzar l'acció d'enviar, ara toca definir el concepte d'allò que volem enviar. L'aplicació en la qual implantem el mecanisme de control d'accés serveix per enviar correu electrònic, així que el concepte que hem de definir és el del correu. El concepte *Email* ha de contenir l'emissor, el receptor i el missatge que volem enviar. Per tant, aquest concepte tindrà la forma que apreciem a la taula 4.2.

Un cop hem definit els conceptes d'usuari i de correu electrònic, podria semblar que ja hauriem acabat de definir els conceptes necessaris per pendre una de-

COMPANY

Nom	Tipus	Aparició	Comentari
name	String	Obligatori	Nom de la companyia aèria
country	String	Obligatori	País on la companyia té la seu central

Taula 4.3: Concepte Company

cisió en el mecanisme de control d'accés, però no és així. Suposem que entre un avió d'una companyia A i la torre de control només hi ha un avió de la companyia B. Si aquestes companyies no tenen un conveni de col·laboració, quan el correu arribi a l'avió B, aquest comprovarà que no es de la seva mateixa companyia i no permetrà el reenviament sinó que el descartarà. Per tant, necessitem algun concepte que ens definexi quina companyia és l'origen del correu electrònic. El concepte que necessitem per a que es compleixin aquestes condicions és el de la taula 4.3.

En darrer lloc, hem de trobar com unir tots els conceptes que hem explicat. Per aconseguir-ho necessitem el predicat *Send* que, com indica el seu nom, ens serveix per indicar que un usuari vol enviar un missatge de correu electrònic. En aquest cas, però, l'expressió que volem no serà: *L'usuari X envia un e-mail*, ja que la informació de l'usuari la podem trobar dins del correu, sinó que el que necessitem saber pel control d'accés en aquest cas és la companyia que envia el e-mail, ja que si les companyies no cooperen entre ells enviar el correu electrònic no serà permés. Per aconseguir aquesta funcionalitat el predicat ha de tenir la forma que veiem a 4.4.

Un cop hem definit tots els termes que conformen l'ontologia, ja tenim tot el necessari per poder crear la classe AccesControlOntology. El diagrama de classes resultant és el que es mostra a la figura 4.3.

SEND

Nom	Tipus	Aparició	Comentari
email	Email	Obligatori	En aquest camp trobem el correu que es vol enviar
company	Company	Obligatori	Camp necessari per saber de quina companyia aèria era l'avió des d'on s'ha enviat el correu

Taula 4.4: Predicat Send

4.3 Política de control d'accés

Un cop hem dissenyat i implementat el mecanisme de control d'accés ja només ens falta un dels components necessaris per pendre una decisió de control d'accés: la política. A la política trobem les restriccions i condicions que han de complir les diferents parts implicades en el control d'accés. Amb el diagrama de classes de la figura 4.4 podem entendre millor com estan estrucurades les polítiques i quins són els elements que les formen.

En el nostre mecanisme de control d'accés hem hagut d'implementar varies regles per aconseguir el comportament desitjat. Observant el diagrama de classes podem veure el perquè: necessitem controlar dues condicions abans de permentre l'accés al recurs (número de e-mails enviats i nom de la companyia aèria) però cada regla només permet una sola condició. En cas que la companyia no sigui la que hem declarat a la política denegarem la petició d'accés, que només permetrem en cas que l'usuari pertanyi a un grup amb aquest privilegi i no hagi sobrepassat el límit d'ús d'aquesta aplicació. Per aconseguir-ho, hem de fer servir aquestes regles:

- **Regles 1 i 2: Controlar el nom de la companyia.** Ho hem de fer en dues regles perquè la implementació Sun no té cap funció que detecti si dues cadenes són diferents. En ambdós casos denegarem l'accés si es compleixen aquestes condicions.

- **Regla 3: Controlar el grup i detectar un us abusiu de l'aplicació.** En

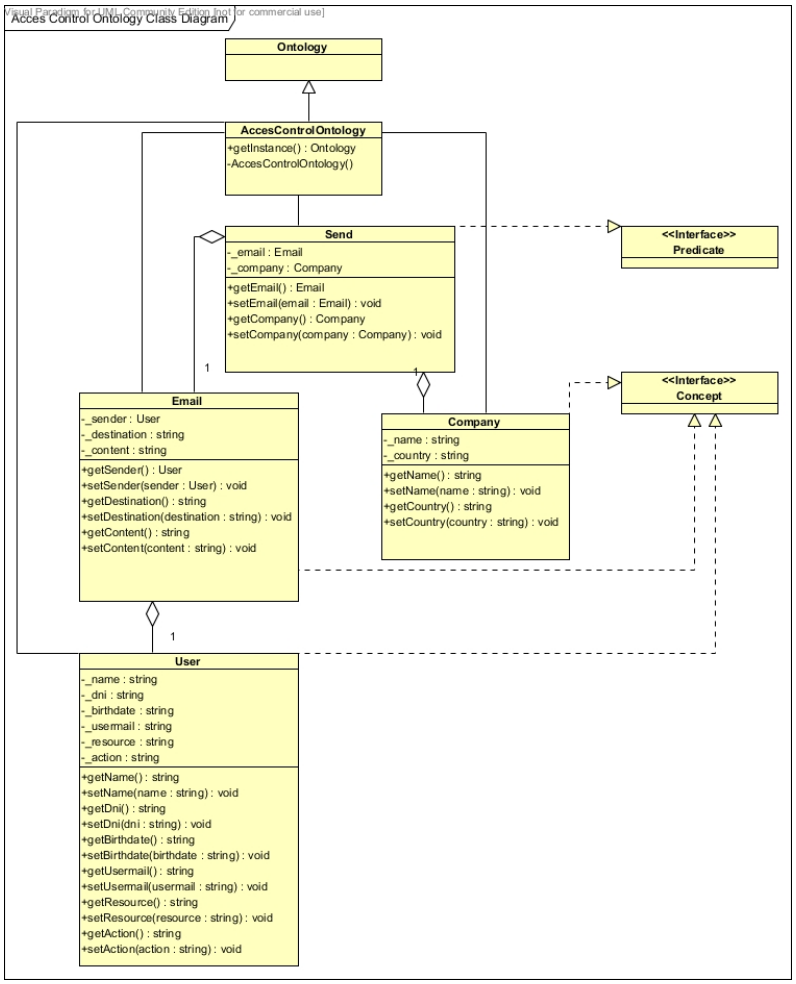

Figura 4.3: Diagrama de classes de l'ontologia

aquest cas permetem l'accés si l'usuari és un usuari business, personal de vol o pilot i a més compleix la condició que no ha sobrepassat el límit d'utilització de l'aplicació.

Per combinar aquestes regles fem servir l'algoritme *deny-overrides*, que en cas de poder aplicar alguna regla que denegui l'accés retornarà un *deny*. En cas contrari, si troba alguna regla que permeti l'accés ens retornarà un *permit* i si no pot aplicar cap regla ens retorna *Not applicable*. Aquí sorgeix un problema, ja que amb el comportament d'aquest algoritme no aconseguim el que esperem del

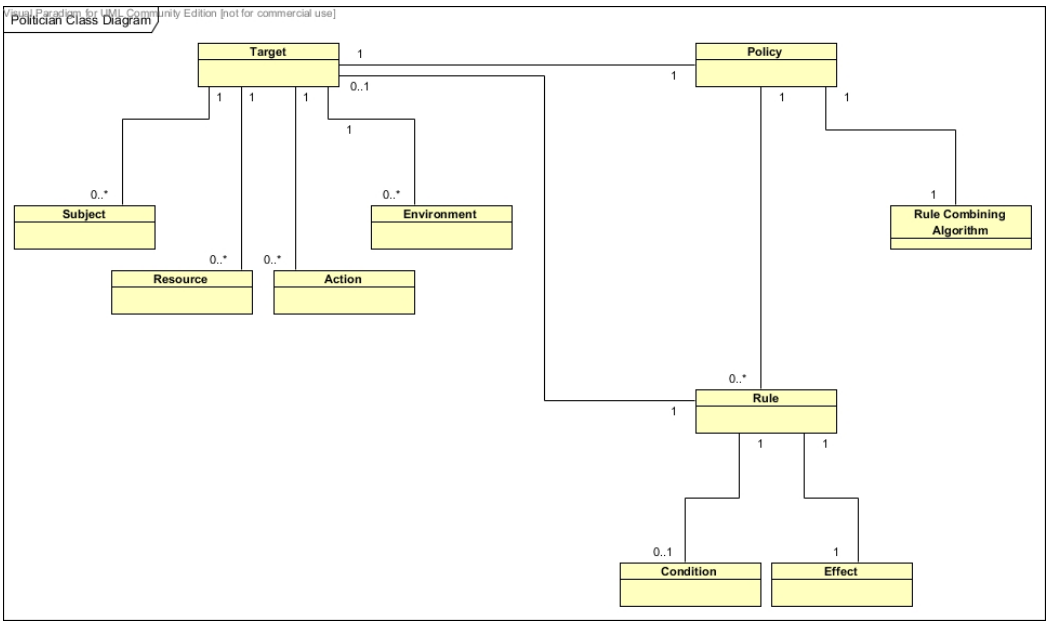

Figura 4.4: Diagrama classes d'una política

mecanisme de control d'accés. Suposem que en la petició d'accés la companyia coincideix, l'usuari pertany a un grup que té el privilegi de poder enviar correu però ha sobrepassat el límit. Les regles 1 i 2 no seran aplicables perquè la companyia és la que toca, i la regla 3 tampoc permetrà l'accés perquè no es compleix la condició. Així, el resultat que ens retornaria aquest algortime seria *Not applicable* quan el correcte seria denegar la petició.

Per solucionar aquest problema hem modificat l'algoritme *deny-overrides*, fent que en el cas que no pugui aplicar cap regla per defecte torni un *deny*. Així aconseguim un control d'accés més restrictiu i que compleix el que nosaltres esperàvem.

Un cop hem finalitzat la implementació del mecanisme de control d'accés, al següent capítol veurem el darrer pas però no per això menys important: dur a terme les proves que comproven que tot funciona de forma adequadament i sense errades.

Capítol 5

Proves

Un cop hem vist com hem desenvolupat tots els mòduls que conformen el projecte toca fer les proves adequades per veure si tot funciona com es preveu. Cal tenir en compte, però, que la majoria de mòduls amb els que treballem ja estaven implementats i per tant suposem que funcionen correctament.

Com hem pogut veure a la implementació, pel desenvolupament del mecanisme de control d'accés s'han fet servir la plataforma Jade i la implementació Sun de XACML, ambdós àmpliament utilitzats en les aplicacions dels seus respectius camps i, per tant, amb un comportament fiable. Així, no cobra sentit fer proves sobre aquest codi doncs suposem que tot funciona correctament.

D'aquesta manera el que ens queda probar és que el mecanisme de control d'accés en conjunt funciona com esperem. És a dir, que reacciona correctament davant de totes les possibles peticions d'accés i que les respostes generades per cadascuna d'aquestes peticions és la correcta. Provarem el mecanisme en situacions on hauria de permetre l'accés, altres on l'hauria de denegar i també en situacions imprevistes com dades incorrectes o mal formades.

5.1 Banc de proves 1. Accés permès

En primer lloc s'a implementat un joc de proves on totes les dades que rep el modul PEP són correctes, on l'usuari pertany a un dels grups que tenen permés

l'accés a l'aplicació de correu electrònic i no ha sobrepassat el límit d'usos de l'aplicació. Per tant, el mecanisme de control d'accés hauria de permetre l'accés de l'usuari, com podem veure a la figura 5.1.

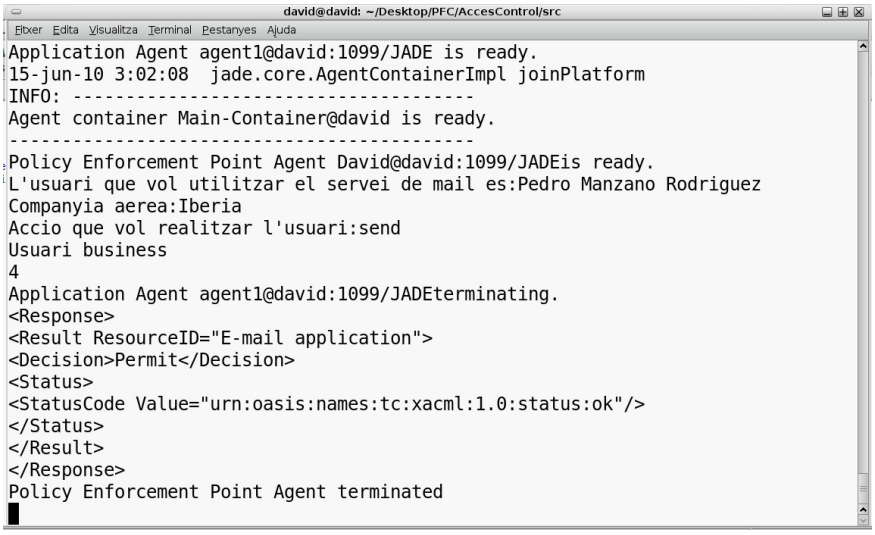

Figura 5.1: Privilegis OK, companyia OK i número d'usos OK

Aquest joc de proves constava d'un total de 20 casos diferents, i en el 100% dels casos els resultats han estat els esperats.

5.2 Banc de proves 2. Acces denegat

Aquest joc de proves permetia un nombre de combinacions més gran per poder comprovar el seu funcionament. L'objectiu principal en el nostre mecanisme de control d'accés era implementar la dependència del context a l'hora de prendre la decisió. Com hem explicat al llarg del projecte, aquesta dependència consistia en el grup al que pertanyia l'usuari (que podia ser usuari turista, usuari business, personal de vol i pilot, dels quals els tres últims tenien accés a l'aplicació de correu) i també en el número de vegades que un usuari havia fet servir l'aplicació. Per tant, en cas que una d'aquestes dues condicions no es complís s'hauria de denegar l'accés, tal com podem veure a les figures 5.2 i 5.3.

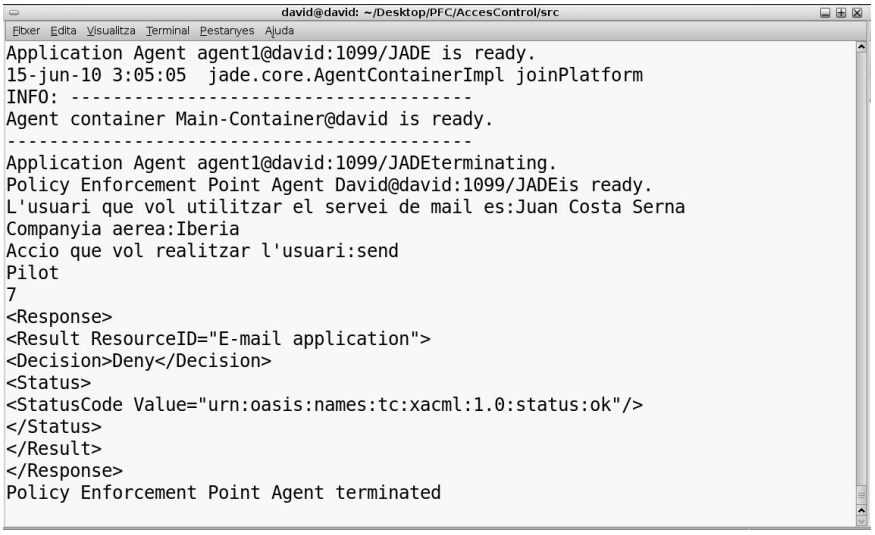

Figura 5.2: Privilegis OK, companyia OK i número d'usos no OK

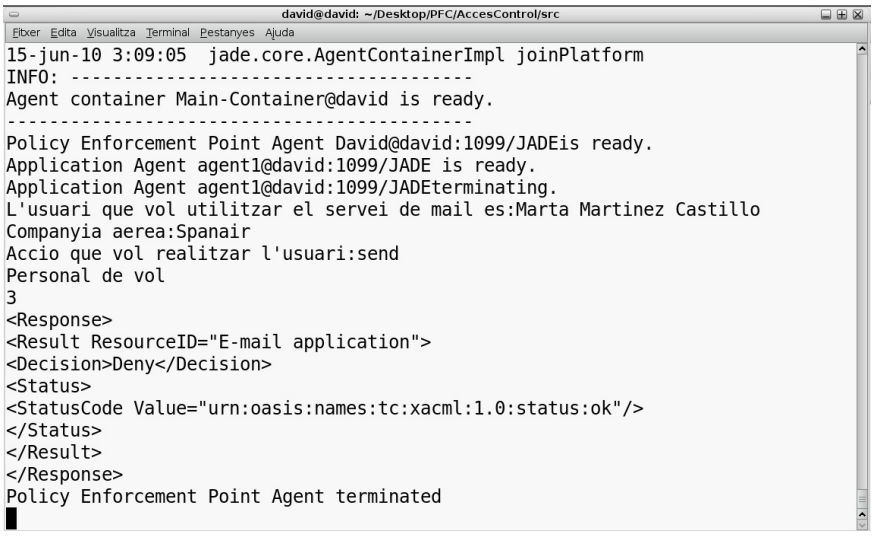

Figura 5.3: Privilegis OK, companyia no OK i número d'usos OK

Un altre aspecte que podria denegar la petició d'accés seria que la companyia aèria a la qual ha arribat el correu no col·labori amb la que ha enviat el missatge originalment. En aquest cas també s'hauria de denegar l'accés a la petició i descartar el missatge.

En aquest cas el banc de proves constava de prop de 40 combinacions i en totes elles el resultat va ser el que esperàvem.

5.3 Banc de proves 3. Peticions incorrectes o mal formades

Aquest banc de proves contempla situacions inesperades que el sistema ha de suportar sense que es produeixin comportaments inesperats. S'han de preveure situacions excepcionals com que la informació que ens arriba des de l'aplicació de PROSES no és de la classe de l'ontologia que esperàvem, que l'usuari que fa la petició no estigui al fitxer o que la informació que ens arriba no és correcta.

S'ha programat un banc de proves amb 10 situacions diferents i en cap d'elles el sistema ha mostrat un comportament inesperat.

En aquest capítol hem vist que el mecanisme que hem desenvolupat en aquest projecte funciona tal com esperàvem. En el següent capítol arriba l'hora de fer balanç de tot el treball que ha suposat aquest projecte traient les corresponents conclusions, veient com podriem ampliar el projecte, veient si hem complert els objectius que ens haviem marcat en un principi i si hem fet cas de la planifiicació que ens haviem proposat.

Capítol 6

Conclusions

Abans d'arribar a aquest capítol, hem pogut veure al llarg de la memòria que per aconseguir fitar els objectius que ens vam marcar al principi del projecte hem hagut de dividir el mecanisme de control d'accés en diferents mòduls per estructurar el codi millor. A més, dividint el codi també aconseguim poder fer modificacions més fàcilment en cas que sigui necessari. Cal tenir en compte, però, que per implementar el projecte hem utilitzat altre codi i, per tant, la distribució temporal de les diferents parts que conformen el desenvolupament d'un projecte potser no són les que haviem previst incialment. En un cas com aquest, cobra molta més importància el fet de conèixer a fons les eines i el codi amb el que treballes, ja que la resta del projecte depén d'aquest aspecte. A més, també és molt important fer un anàlisi correcte de l'entorn per poder extreure les característiques necessàries pel mecanisme de control d'accés. Aquests aspectes, però, no resten rellevancia a la part de disseny i desenvolupament del codi, ja que sobre la gran quantitat de codi de la que disposem hem de saber triar el que més escau i fer-lo servir de forma adequada per evitar que tinguin un comportament extranys. Totes aquestes característiques també afecten a les proves que fem per validar el comportament del projecte, ja que aquestes són bàsicament funcionals. Un cop hem dut a terme tot aquest treball ens queda fer un balanç per veure si hem aconseguit assolir els objectius que ens haviem proposat en un principi, pensar quines podrien ser les ampliacions d'aquest projecte i comprovar si la planificació proposada s'ha seguit

durant el desenvolupament del projecte.

Primer de tot, veurem la feina que hem realitzat i si aquesta aconseguiex assolir els objectius del projecte:

- Per aconseguir comprendre el funcionament dels mòduls de la implementació Sun es va haver de fer un estudi exhaustiu i moltes proves per entendre que feia exactament cada mòdul i que era el que necessitava. Per la plataforma Jade també va ser important comprendre el seu funcionament, especialment pel que fa als missatges ACL i les ontologies. Un cop sabiem sobre quina base treballar, per dissenyar les regles de control d'accés vam fer servir el model ABAC que era el més adequat per aquest entorn.

- Es va fer un anàlisi de l'entorn PROSES, del qual es van extreure les característiques que el defineixen i que són imprescindibles si es vol implementar un bon mecanisme de control d'accés. Característiques com el grup d'un usuari són prou habituals en qualsevol entorn, però altres com un comptador d'utillitzacions no són tant. A part de les característiques del context també s'han tingut en compte altres aspectes de l'entorn PROSES a l'hora de prendre les decisions de control d'accés (com la companyia que envia el correu).

- Un cop sabíem tot el necessari per fer el disseny va arribar el moment de dur-lo a terme. Arribat aquest punt es va concloure que era millor separar el mecanisme de control d'accés en diversos mòduls per tenir més clara l'estructura de tot el programa. A més, es va decidir que les característiques de l'entorn que fariem servir a l'hora de prendre una decisió les obtindriem d'un fitxer.

- Es va implementar el mecanisme de control d'accés del qual havíem fet el disseny. En aquesta part del projecte es van haver de desenvolupar els nostres mòduls PEP, PDP i PIP, que feien servir elements de la implementació Sun però que es van haver d'adequar per poder treballar amb agents en un entorn com aquest. En aquest projecte, però, només s'ha pogut implemen-

tar el control d'accés per l'aplicació del correu electrònic d'entre les que s'havien plantejat per PROSES.

- Per acabar, es va comprovar que tot funcionava correctament. Es van fer les proves necessàries per veure que el mecanisme de control d'accés es comportava com havia de fer-ho en totes les situacions possibles. En aquest projecte un altre tipus de proves no cobren massa sentit ja que es treballa sobre codi d'àmplia difusió i ja provat.

Després de veure tot això podem concloure que s'han acoseguit assolir tots els objectius que ens vam marcar al començar el projecte.

6.1 Línies d'ampliació

Evidentment, a mesura que es coneix més informació sobre el tema un s'adona de les noves possiblitats que fins aquell moment no s'havien plantejat i les possibles millores que es podrien aplicar al projecte. Tot això ho detallem a continuació:

1. Potser la línia d'ampliació més evident seria fer que el mecanisme de control d'accés pogués treballar amb més aplicacions a part del correu electrònic. Això podria provocar haver d'ampliar l'estudi sobre l'entorn PROSES, doncs hi ha aplicacions molt diverses que poden necessitar informació que en el nostre cas no era necessària.

2. Un altre aspecte que es podria millorar seria la seguretat del sistema, és a dir, poder tractar amb informació que ens arribés xifrada.

3. En cas d'haver de treballar amb més aplicacions sorgirien noves polítiques de control d'accés. En aquestes circumstàncies el més adient seria programar un nou mòdul que s'encarregués de la gestió d'aquestes polítiques, el Policy Authorisation Point (PAP).

4. Un major nombre d'aplicacions també provocaria un increment amb les dades que necessitem per dur a terme el control d'accés. Guardar informació en un fitxer de text no seria el més eficient, i possiblement la millor

solució seria tenir emmagatzemada tota aquesta informació en una base de dades.

6.2 Anàlisi de la planificació

Per acabar, veurem si hem seguit la planificació que ens haviem propsosat al principi del projecte. Amb la taula de tasques que podem veure a la figura 6.1 podem veure com ha estat finalment la planificació, posant especial èmfasi en la tasca que mes ha variat respecte a la planificació inicial.

Com hem comentat prèviament, l'estudi de la implementació Sun de XACML era de vital importància per aconseguir que el projecte funcionés com esperàvem. Per tant, el seu estudi va allargar-se més del previst inicialment. Això va provocar que la resta de tasques comencessin una mica més tard i que es retallés el temps de disseny i implementació del codi, ja que al tenir aquesta implementació disponible el temps dedicat a aquesta part del projecte va disminuir.

GANTT project

Nombre	Fecha de inicio	Fecha de fin
1. Realització Informe Previ	8/01/10	15/01/10
2. Estudi del model ABAC i la implementació Sun de XACML	15/02/10	10/03/10
2.1 Anàlisi model RBAC	15/02/10	16/02/10
2.2 Model de Control d'Accés ABAC	16/02/10	19/02/10
2.3 Estudi de la implementació Sun de XACML	19/02/10	10/03/10
3. Anàlisi Entorn PROSES	10/03/10	15/03/10
3.1 Anàlisi Entorn SESAR	10/03/10	11/03/10
3.2 Estudi Xarxes DTN	11/03/10	12/03/10
3.3 Relació PROSES - ABAC	12/03/10	15/03/10
4. Estudi AM / JADE	15/03/10	21/03/10
4.1 Agents mòbils	15/03/10	16/03/10
4.2 Plataforma JADE	17/03/10	21/03/10
5. Disseny i Implementació Mecanisme	15/03/10	30/04/10
5.1 Disseny	15/03/10	17/04/10
5.2 Implementació	10/04/10	30/04/10
6. Proves de validació	30/04/10	5/05/10
7. Realització de la Memòria	1/05/10	31/05/10
Fita 1: Lliurament Informe Previ	13/01/10	15/01/10
Fita 2: Sol.licitud Lectura	16/05/10	31/05/10
Fita 3: Entrega Memòria	17/06/10	22/06/10
Fita 4: Lectura	28/06/10	14/07/10

Figura 6.1: Planificació temporal final del projecte

Bibliografia

[Blakley] B.Blakley. The Emperor's Old Armor. Proceedings of the New Security Paradigms Workshop, 1996

[proses] Protocolos de Red Para el Cielo Único Europeo. Ministeri d'Indústria, Comerç i Turisme.
ref. TSI-020100-2009-115

[jade] Bellifemine, F.L.; Caire, G.; Greenwood, D. Developing Multi-Agent Systems with Jade. Febrer 2007. ISBN: 978-0-470-05747-6

[rbac] Ferraiolo, D.F. and Kuhn, D.R. Role-Based Acces Control. 15th National Computer Security Conference. Octubre 1992. pp. 554-563.

Sandhu, R., Coyne, E.J., Feinstein, H.L. and Youman, C.E. Role-Based Acces Control Models. IEEE Computer (IEEE Press) 29 (2): pp. 38-47. Agost 1996

[xacml] eXtensible Acces Control Markup Language
<http://www.oasis-open.org/committees/xacml/>

[ABAC] C. Schägler, T. Priebe, M. Liewald and G. Pernul, "Enabling Attribute-Based Access Control in Authentication and Authorisation Infrastructures", Proc. of the 20th Bled eConference - eMergence (Bled 2007).

[saml] Security Assertion Markup Language
<http://www.oasis-open.org/committees/security/>

[ldap] Lightweight Directory Acces Protocol
<http://www.openldap.org/>

[sun] Sun Mycrosistems, Inc
<http://www.oracle.com/us/sun/index.html>

[oasis] Advancing open standards for the information society
«http://www.oasis-open.org/l>

[fipa] The Foundation for Intelligent physical Agents
<http://www.fipa.org/>

[acl] Agent Comunication Language
<http://jade.tilab.com/doc/api/jade/lang/acl/ACLMessage.html>

[RFC 4838] Delay-Tolerant Networking Architecture. Informational. Abril 2007

[lopd] LEY ORGÁNICA 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.

Firmat: David Quintana Brígido

Bellaterra, juny de 2010

Resum

Amb l'aparició de nous entorns en el món de la informàtica sorgeixen noves necessitats. Un d'aquests entorns és SESAR, un entorn que preveu l'ús massiu de transmissió de dades entre sistemes aeris. L'objecitu principal d'aquest projecte és aconseguir implementar el control d'accés per una aplicació en aquest entorn. El primer pas és dur a terme un estudi exhaustiu dels diferents elements que necessitem per desenvolupar el projecte, així com un anàlisi de l'entorn per extreure les característiques pel control d'accés. Després es presenta el seu disseny i implementació, que acompleix el seu objectiu de gestionar el control d'accés de l'aplicació de correu electrònic en aquest entorn.

Resumen

Con la aparición de nuevos entornos en el mundo de la informática surgen nuevas necesidades. Uno de estos entornos es SESAR, que prevé el uso masivo de transmisiones de datos entre sistemas aéreos. Este proyecto parte con el objetivo de conseguir implementar el control de acceso para una aplicación en este entorno. El primer paso es un estudio exhaustivo de los elementos que necesitamos para desarrollar el proyecto, así como un análisis del entorno para extraer las características para el control de acceso. Después se presenta su diseño e implementación, que cumple su objetivo inicial al ser capaz de gestionar el control de acceso de la aplicación de correo electrónico en este entorno.

Abstract

The emergence of new environments in the world of computing causes new requirements. On of these environments is SESAR, which foresees a massive usage of data transmission between aerial systems. The main target of this project is to implement an acces control mechanism for an application in this environment. The first step is a comprehensive study of the elements we need to develop the project and an environmental analysis to extract the features that we need for the mechanism. After that we present its design and implementation, that fulfills its original purpose of being able of managing an access control for the email application in this environment.