AGENTS MÒBILS EN SITUACIONS D'EMERGÈNCIA. GESTIÓ DINÀMICA DE SERVEIS.

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

Bellaterra, Juny de 2009

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 Morcillo Muñoz

Bellaterra, Juny de 2009

Firmat: Sergi Robles Martínez

A la meva família.

Agraïments

En primer lloc m'agradaria agrair a tot l'equip del SeNDA, en especial al meu director Sergi Robles, el seu suport i guia durant el desenvolupament d'aquest projecte.

El recolzament de la família i els amics ha estat vital per a superar diverses proves al llarg de la carrera i també m'han acompanyat en aquest últim pas. Moltes gràcies a tots.

Índex

Índex de figures

Capítol 1

Introducció

En una situació d'emergència es mobilitzen moltes persones per tal de poder socórrer les víctimes de la forma més ràpida i ordenada. De vegades és difícil poder accedir a les víctimes a causa dels desperfectes generats per un gran desastre o accident i és necessària la intervenció d'equips especialitzats. Per al benestar de les víctimes, cal definir una sèrie de pautes i protocols per a poder realitzar tot un seguit de tasques: avaluar l'estat de les víctimes, transportar-les a l'hospital més pròxim i tractar-les, de la forma més eficient possible.

En el projecte d'investigació "Disseny i implementació d'una solució global per al suport de serveis crítics en situacions d'emergència" [TSI2006-03481] s'ha dissenyat una solució per a la utilització d'agents mòbils en aquest tipus de situacions. En concret, el projecte tracta els problemes de comunicació, distribució de serveis i logístics, per a la intervenció mèdica durant una emergència.

En aquest entorn apareixeran una sèrie de serveis que cal gestionar i organitzar per fer-los accessibles i usables. Aquest projecte consisteix en el disseny i desenvolupament d'una arquitectura que proporcionarà mecanismes de gestió i utilització de serveis.

1.1 Situació inicial

Actualment quan hi ha un desastre o un gran accident, l'equip mèdic assignat a l'assistència de les víctimes realitza un primer procés anomenat triatge. Aquest procés consisteix a avaluar l'estat de la víctima a partir d'una sèrie de paràmetres vitals bàsics. El resultat d'aquesta avaluació serveix per a assignar una etiqueta a la víctima. De menor a major urgència, el color corresponent a l'etiqueta és el següent:

- **Negre:** La víctima és morta.
- **Verd:** La víctima pot caminar per si sola i no presenta ferides greus.
- **Groc:** La víctima presenta ferides moderades i ha de ser evacuada ràpidament.
- **Vermell:** La víctima presenta ferides greus i ha de ser evacuada el més ràpid possible.

Fins ara el procés era realitzat de forma manual per l'equip mèdic. El grup de recerca SeNDA (*Security of Networks and Distributed Applications*) del Departament d'Enginyeria de la Informació i de les Comunicacions està desenvolupant un sistema de suport de serveis crítics en sistemes d'emergència. La solució proposada per el grup SeNDA per automatitzar el triatge de les víctimes dins el projecte [TSI2006-03481] s'anomena MAETT (*Mobile Agent Electronic Triage Tag*) i es basa en el paradigma dels agents mòbils per a la realització d'aquest procés.

Tal i com podem veure a la Figura 1.1, existeixen 4 zones ben diferenciades. A la zona 0 és on els *First Responders* estan efectuant el procés de triatge. Cada membre d'aquesta unitat porta una PDA i aquestes estan connectades entre elles formant una xarxa MANET (*Mobile Ad-hoc Network*).

Quan l'usuari realitza el triatge a través d'una interfície, es crea un agent mòbil que conté la informació de la víctima (etiqueta i localització). Aquest agent anirà

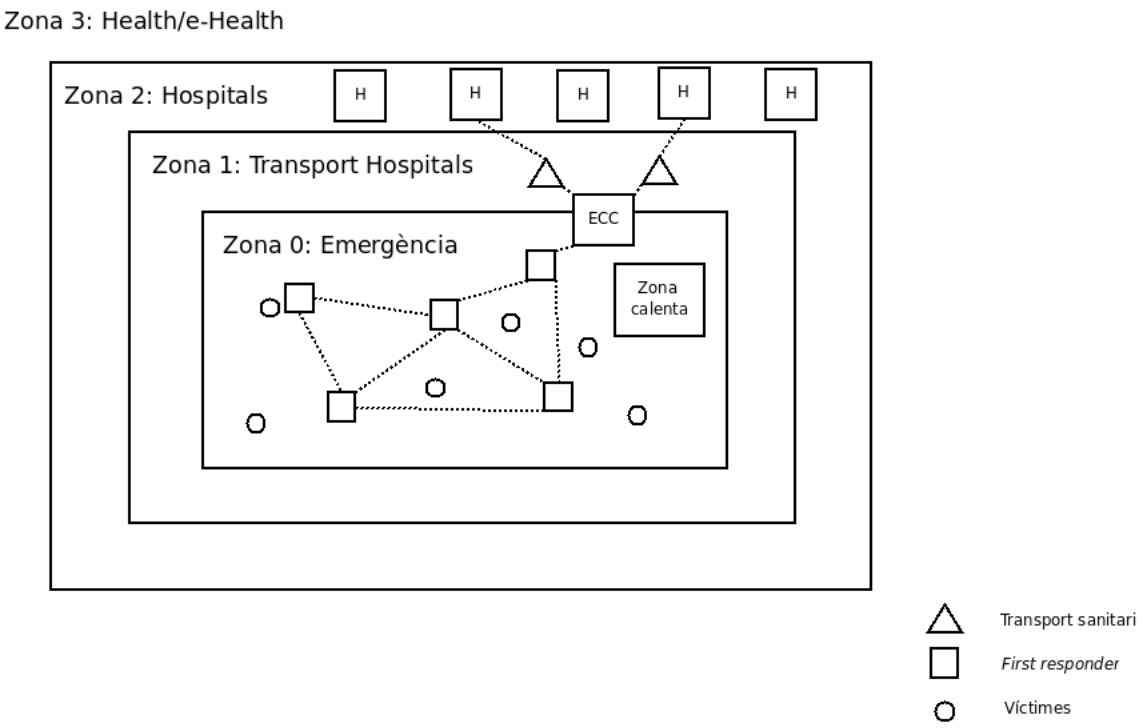

Figura 1.1: Escenari d'una emergència

saltant entre els diferents equips fins arribar al ECC (*Emergency Coordination Center*).

1.2 Motivació

En aquest entorn han de conviure una sèrie d'actors que en determinades ocasions oferiran una sèrie de serveis i quan sigui necessari n'utilitzaran d'altres. Aquesta gestió i utilització de serveis s'ha de controlar d'alguna manera, i no podem delegar aquestes funcions a l'aplicació actual ja que cada aplicació hauria de gestionar els seus propis serveis i això provocaria diversos problemes com ara la duplicació de codi, inconsistència, etc.

Presentat aquest escenari, podem realitzar diverses preguntes:

- Què passa si una víctima necessita un tractament molt específic d'urgència?
- Com sap l'agent mòbil a quina plataforma ha de saltar per a poder arribar al ECC el més ràpid possible?
- Què ha de fer l'agent mòbil quan arriba a l'ECC?

Aquestes preguntes ens serveixen per a situar una arquitectura de serveis en el sistema existent. L'agent mòbil que conté la informació del pacient cercarà un servei que li proporcioni un mecanisme per a entregar aquesta informació. Al centre de coordinació hi haurà un equip que acceptarà aquesta informació o, més ben dit, oferirà un servei a la xarxa que consistirà en el tractament d'aquestes dades.

Veiem doncs la necessitat de definir una arquitectura de serveis que ens serveixi per a gestionar i utilitzar els serveis que proporcionen per a aquests agents.

1.3 Objectius del projecte

L'objectiu principal del projecte és integrar una arquitectura basada en serveis en aquest sistema. Aquest objectiu es pot dividir en diverses etapes o sub-objectius:

- Estudiar el sistema i escenari actual [TSI2006-03481] per al suport de serveis crítics en situacions d'emergències.
- Dissenyar i implementar una arquitectura SOA (*Service-oriented architecture*) per agents intel·ligents.
- Definir un conjunt de serveis mèdics i no mèdics per a cada zona de l'escenari actual com ara la reserva de recursos.
- Implementar una sèrie de serveis mínims en aquesta nova arquitectura.
- Integrar el projecte en el sistema actual [TSI2006-03481].

1.4 Contingut de la memòria

La resta de la memòria està formada pels següents capítols:

Capítol 2: Sistemes multi-agent i ontologies Introduirem els aspectes tecnològics clau necessaris per a entendre la solució aportada en aquest projecte: els sistemes multi-agent i les ontologies.

Capítol 3: Anàlisi S'analitza el problema estudiant els requisits que ha de complir la nostra solució i es realitza un anàlisi de viabilitat sobre el projecte. A continuació es planifica el projecte i es divideix en tasques.

Capítol 4: Disseny de l'arquitectura En aquest capítol es realitza un disseny acurat que compleixi els requisits analitzats.

Capítol 5: Implementació Utilitzarem una tecnologia específica per a la implementació del projecte. En aquest capítol es presenten els punts més conflictius i de quina forma s'han resolt al llarg del projecte.

Capítol 6: Resultats Per a validar el *software* implementat cal realitzar una sèrie de proves per a garantir el seu correcte funcionament. En aquest capítol es presenten les proves i els seus resultats.

Capítol 7: Conclusions Per a concloure amb el projecte es fa una reflexió i s'analitza l'assoliment dels objectius.

Capítol 2

Sistemes multi-agent i ontologies

Després d'haver presentat el projecte cal fer una introducció als conceptes principals que es tractaran al llarg d'aquest escrit.

En primer lloc parlarem sobre la tecnologia que s'ha utilitzat principalment: els sistemes multi-agent. A continuació presentarem la IEEE FIPA (*Foundation for Intelligent Physical Agents*), una organització que s'encarrega de mantenir estàndards en aquesta tecnologia. Seguidament introduirem conceptes més específics i començarem a parlar d'agents: plataformes, serveis i comunicació.

Per acabar presentarem el concepte d'ontologia, de gran importància per al desenvolupament del projecte.

2.1 Sistemes multi-agent

Un sistema multi-agent és un conjunt format d'agents intel·ligents que poden interaccionar entre ells per a poder resoldre problemes complexos, mentre que de forma individualitzada hagués estat molt difícil donada la seva complexitat.

Les característiques més importants dels agents són:

- **Autonomia:** Poden prendre les seves pròpies decisions a partir del seu comportament.
- **Visió local:** No tenen una visió global del sistema.

- **Reactivitat i pro-activitat:** No només reaccionen envers esdeveniments externs sinó que també són capaços d'actuar en base als seus objectius.

- **Comportament social:** Es comuniquen entre ells per assolir un objectiu comú.

Els agents també poden ser mòbils, capaços de moure's per la xarxa transportant el seu propi codi i crear el seu propi àmbit d'execució.

Els sistemes multi-agent també ofereixen molta flexibilitat ja que és possible introduir nous agents en un sistema sense necessitat de tornar a reescriure l'aplicació.

2.1.1 IEEE FIPA

IEEE FIPA (*Foundation for Intelligent Physical Agents*) és una secció d'IEEE que promou l'estandardització de tecnologies relacionades amb agents intel·ligents.

Actualment es considera l'estàndard d'agents més àmpliament reconegut i un referent a l'hora d'utilitzar aquesta tecnologia. Les especificacions creades per la IEEE FIPA representen una col·leció d'estàndards que promouen la interoperabilitat d'agents desenvolupats en marcs de treball que suporten transports heterogenis i aplicacions basades en agents.

2.2 Agents i plataformes

Després d'introduir el concepte d'agent intel·ligent i presentar una mica la tecnologia que utilitzarem al llarg del projecte cal aprofundir una mica més en alguns aspectes. Podem veure un agent com una màquina d'estats des del moment que es crea l'agent fins al moment que es destrueix. A la figura 2.1 podem veure el cicle de vida d'un agent i els diferents estats d'aquest.

En una plataforma poden conviure diversos agents que en algun moment, depenent del seu comportament, poden decidir migrar a una altra plataforma. A la figura 2.2 podem veure l'esquema bàsic d'una plataforma IEEE FIPA d'agents.

Els elements més importants de la plataforma IEEE FIPA són:

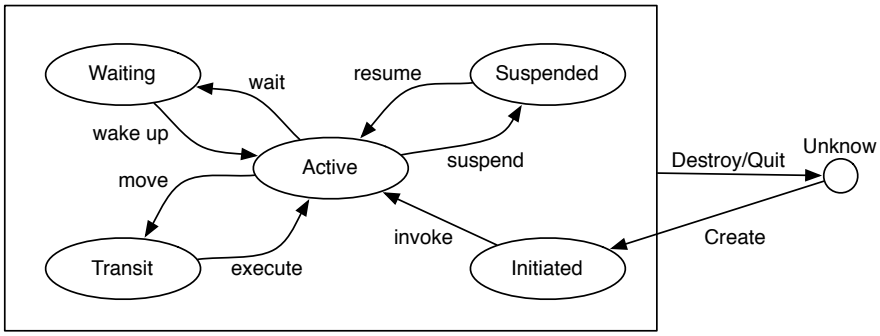

Figura 2.1: Cicle de vida de l'agent

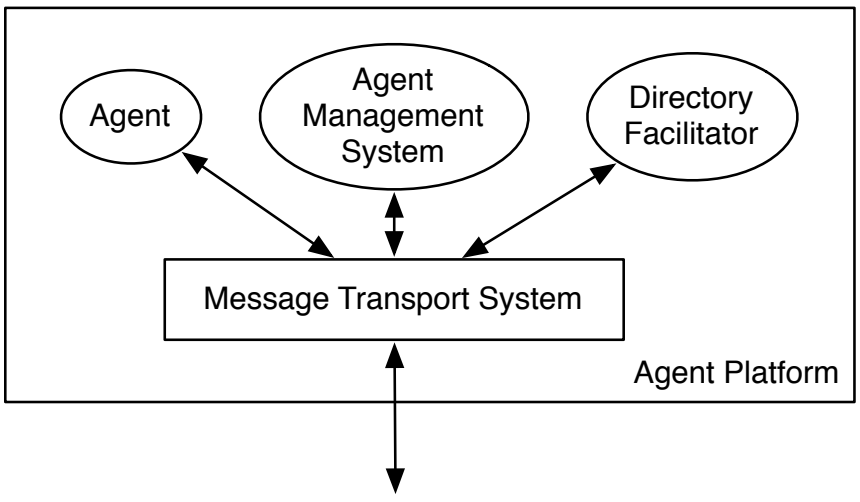

Figura 2.2: Esquema bàsic d'una plataforma IEEE FIPA d'agents

- **AMS:** L'agent *Agent Management System* és l'autoritat de la plataforma. És l'únic agent que pot crear o destruir altres agents i pot parar la plataforma en qualsevol moment.

- **DF:** El *Directory Facilitator* és un directori de pàgines grogues emprat per a registrar els serveis proporcionats per els agents.

- **MTS:** El *Message Transport System* (o ACC) s'encarrega de gestionar l'intercanvi de missatges entre agents.

2.3 *Java Agent DEvelopment framework*

JADE (*Java Agent DEvelopment framework*) és una plataforma *software* per al desenvolupament d'agents, implementada en Java. La plataforma JADE suporta la coordinació de múltiples agents, seguint els estàndards d'IEEE FIPA i proporciona una implementació del llenguatge de comunicació FIPA-ACL, del qual en parlarem més endavant. JADE també ens proporciona un conjunt d'eines gràfiques per administrar i controlar l'execució dels agents.

Analitzem ara l'arquitectura de JADE tal i com podem veure a la Figura 2.3. JADE crea múltiples contenidors per als agents, cada un dels quals es pot executar en un o diversos sistemes. Un conjunt de contenidors constitueix una plataforma d'agents i cada plataforma ha de tenir un contenidor principal.

JADE presenta les següents característiques:

- **P2P:** Arquitectura *peer-to-peer* on cada agent pot prendre la iniciativa en una comunicació o bé respondre a peticions que facin altres agents.

- **Interoperabilitat:** Els agents desenvolupats en JADE es poden comunicar amb altres agents que hagin estat desenvolupats emprant una altra plataforma que segueixi els estàndards IEEE FIPA.

- **Portabilitat:** L'API de JADE és independent de la versió de Java utilitzada i la xarxa utilitzada per part dels agents.

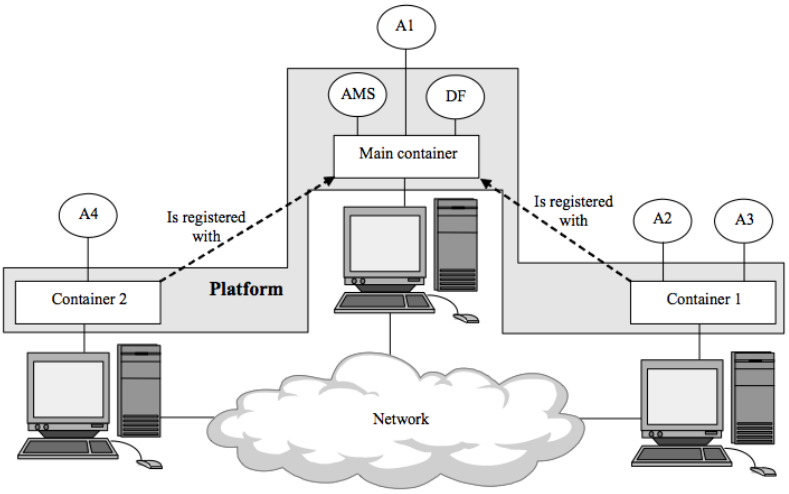

Figura 2.3: Arquitectura de JADE

- **Intuïtiva:** L'API de JADE és fàcil d'aprendre i permet al programador centrar-se en el seu ús.

2.4 Anunci i descobriment de serveis

A l'apartat anterior hem introduït l'agent DF relacionat amb la gestió de serveis. Com ja hem comentat, el DF es pot veure com un directori de pàgines grogues que els altres agents poden consultar i registrar-se. Existeix un únic DF per plataforma i s'utilitzen missatges ACL per a interaccionar amb ells. Podem veure un servei com una funció sense estat que accepta una petició i retorna una resposta. No depèn de l'estat d'altres funcions o processos. En l'entorn on ens trobem hi haurà agents productors de serveis, els quals proporcionaran una sèrie de serveis als agents consumidors.

Si un determinat agent proporciona un servei, el primer que ha de fer és registrar aquest servei en el DF. En el moment que deixi de proporcionar el servei haurà d'avisar el DF i indicar-li que es vol desregistrar. Altres operacions bàsiques del DF són la cerca i subscripció de serveis.

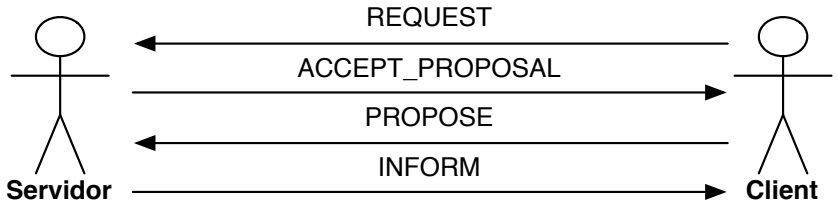

Figura 2.4: Conversa bàsica entre dos agents

2.5 Comunicació entre agents

En un sistema multi-agent és molt important la comunicació entre agents intel·ligents per tal d'assolir un objectiu concret. JADE ens proporciona una implementació del llenguatge de comunicació FIPA-ACL. Aquesta comunicació es realitza mitjançant la transmissió de missatges ACL amb el següent contingut:

- **Emissor:** Agent emissor.
- **Receptors:** Llista de receptors.
- ***Performative:*** Tipus.
- **Contingut:** Contingut del missatge.
- **Llenguatge:** Llenguatge utilitzat en el contingut.
- **Ontologia:** Ontologia utilitzada en el contingut.
- **Altres:** Conté altres camps relacionats amb les converses recurrents.

A la Figura 2.4 podem veure un exemple de conversa entre dos agents. La comunicació entre agents és molt necessària per a la gestió i utilització de serveis.

2.6 Ontologies

El contingut dels missatges ACL pot ser qualsevol, tot i que per tal que la conversa entre agents sigui possible les dues parts han d'entendre què hi diu en el missatge.

Una ontologia es pot veure com un diccionari que representa el significat d'una sèrie de paraules molt concretes.

Podem classificar aquestes paraules en tres grans grups:

- **Conceptes:** Defineixen els objectes amb els que tractem.
- **Predicats:** Defineixen una sintaxi lògica que permet demanar peticions en base a un conjunt de restriccions.
- **Accions:** Es refereixen directament als serveis que pot oferir un agent.

Relacionat amb les ontologies trobem el llenguatge, que no és res més que la manera com es codifiquen els objectes de la ontologia. Existeixen diferents llenguatges, tot i que JADE proporciona de manera nativa els llenguatges SL (*Semantic Language*) i XML.

Capítol 3

Anàlisi

Abans de començar qualsevol projecte cal realitzar un estudi per acotar l'abast del projecte i definir quins són els requisits que ha de complir per a solucionar el problema.

És necessari realitzar un estudi de viabilitat del projecte i planificar-lo adequadament per tal d'assolir els objectius inicials.

3.1 Anàlisi de requisits

Primer hem de definir el projecte en funció del que volem que pugui fer o més ben dit les funcions que ha de tenir. Ens centrarem en l'arquitectura de serveis, l'objectiu principal del projecte.

3.1.1 Requisits funcionals

Els requisits funcionals són característiques que expressen funcionalitats o capacitats per fer alguna cosa. Pensant en les necessitats del sistema, podem definir els següents requisits:

Registre de serveis L'arquitectura ha de mantenir el seu propi directori de pàgines grogues i registrar els serveis que s'ofereixin en un moment determinat.

Desregistre de serveis Cal mantenir una consistència amb els serveis disponibles i s'han de desregistrar aquells que ja no siguin accessibles.

Cerca de serveis L'aplicació no sempre coneix qui proporciona un determinat servei. La nova arquitectura ha de permetre la cerca de serveis tant a nivell local com a nivell remot.

Utilització de serveis L'arquitectura no només ha de mantenir un directori de pàgines grogues sinó que ha de proporcionar una interfície per a poder utilitzar els serveis registrats.

3.1.2 Requisits no funcionals

Els requisits no funcionals són característiques necessàries del sistema, del procés de desenvolupament o qualsevol altre aspecte del desenvolupament que el restringeix d'alguna manera.

Podem definir els següents requeriments no funcionals:

Entorn dinàmic L'aplicació s'executa principalment en un entorn mòbil i és necessari mantenir una consistència amb les dades.

Velocitat En una situació d'emergència el temps és molt important. La nova arquitectura ha de respondre en un temps mínim i adequat per a cada situació.

Dispositius mòbils L'arquitectura no pot ser gaire pesant i ha de poder ser portable als dispositius mòbils que s'utilitzin en aquest escenari.

Multi-plataforma Vàlid per a ser utilitzat sobre la plataforma JADE però també s'ha de poder utilitzar amb qualsevol plataforma IEEE FIPA.

Missatges La gestió i utilització de serveis s'ha de basar en el pas de missatges entre agents i no en crides a mètodes remots.

3.2 Estudi de viabilitat

En aquest apartat es durà a terme un estudi de viabilitat per analitzar la proposta del projecte de final de carrera.

3.2.1 Especificacions

Cal dissenyar una arquitectura orientada a serveis i integrar-la en el sistema actual [TSI2006-03481]. En primer lloc hem d'entendre el funcionament del sistema i dissenyar el conjunt de serveis que existiran.

A continuació cal definir els actors que interaccionen amb aquests serveis: els productors i consumidors. Ens centrarem en la zona principal de l'emergència i es definiran tots aquells serveis que siguin necessaris. Un cop coberts els serveis mínims de la primera zona passarem a avaluar la necessitat d'incloure serveis per a zones de més alt nivell o serveis entre zones.

Després de dissenyar els serveis els haurem d'implementar utilitzant una tecnologia concreta: els agents mòbils. Haurem de fer un estudi de l'arquitectura de serveis de la plataforma JADE (*Java Agent Development Platform*), plataforma utilitzada en el sistema actual, així com els diferents protocols d'anunci i descobriment de serveis.

Per acabar, existeix la possibilitat de fer una demostració junt amb els altres projectistes del SeNDA, simulant una situació d'emergència i fent un petit desplegament de dispositius mòbils.

3.2.2 Viabilitat tècnica

Per a la implementació de serveis en agents mòbils és necessari disposar d'una plataforma de desenvolupament. S'utilitzarà JADE per a poder realitzar la implementació satisfactòriament. JADE es pot obtenir de forma lliure a la pàgina oficial: `http://jade.tilab.com/`.

A més a més, el Departament d'Enginyeria de la Informació i de les Comunicacions disposa d'un laboratori per a projectistes amb una sèrie d'ordinadors amb el sistema operatiu GNU/Linux i connexió a Internet.

Per tant, es disposa de tots els mitjans físics i lògics necessaris per a la realització d'aquest projecte.

3.2.3 Viabilitat operativa

El projecte s'haurà de desenvolupar en diferents fases, sent necessària la paral·lelització d'algunes tasques per tal d'assolir els objectius plantejats. Així doncs és necessària una formació tècnica per part del grup SeNDA per a poder complementar els coneixements obtinguts durant la carrera.

3.2.4 Viabilitat econòmica

Els recursos que s'utilitzaran durant el desenvolupament del projecte seran eines de *software* lliure, com ara la plataforma JADE i el sistema operatiu GNU/Linux. Per tant no suposarà cap cost addicional al projecte.

Per a garantir la viabilitat del projecte és necessari disposar d'un Enginyer Superior que haurà de dedicar un mínim de 150 hores al projecte. A l'Apartat 3.3.3 podem veure un pressupost aproximat del cost que suposa la realització d'aquest projecte.

3.2.5 Viabilitat legal

La plataforma JADE és *Open Source* i està sota una llicència LGPL. S'utilitzaran dades mèdiques reals durant la implementació del projecte. Les dades mèdiques estan regulades per la Llei de Protecció de Dades de Caràcter Personal (LOPD), es consideren de nivell alt i requereixen les mesures de seguretat més estrictes.

3.2.6 Alternatives

Quan pensem en una arquitectura de serveis relacionem aquest concepte amb els *Web Services*. En un entorn on la topologia de la xarxa canvia contínuament a causa del moviment dels equips, i la poca cobertura que hi ha en una situació d'emergència, és impensable disposar d'una connexió a Internet estable.

Els equips que es trobin a la zona de l'emergència utilitzaran dispositius mòbils on hi haurà una plataforma d'agents mòbils en cada un d'ells. Els serveis que s'han de dissenyar, així com els protocols a estudiar han d'estar adaptats a aquest tipus de xarxa i al paradigma d'agents mòbils.

Actualment no hi ha cap tecnologia alternativa que satisfaci els requisits analitzats.

3.3 Planificació

3.3.1 Tasques

Les tasques a realitzar durant el desenvolupament del projecte són les següents:

#	Descripció	Data Inici	Data Final	Duració
	Projecte	**01/01/09**	**20/05/09**	**225h**
1	**Estudi del sistema existent**	**11/01/09**	**05/03/09**	**65h**
1.1	Anàlisi de l'escenari	11/01/09	30/01/09	20h
1.2	Anàlisi protocols d'anunci i descobriment	01/02/09	05/03/09	10h
1.3	Disseny arquitectura i serveis	01/02/09	05/03/09	35h
2	**Implementació**	**06/03/09**	**02/04/09**	**30h**
3	**Proves**	**03/04/09**	**01/05/09**	**30h**
4	**Integració**	**02/05/09**	**20/05/09**	**20h**
5	**Redacció documents**	**01/01/09**	**20/05/09**	**80h**
5.1	Informe previ	01/01/09	15/01/09	10h
5.2	Memòria	11/01/09	20/05/09	70h

3.3.2 Diagrama de gantt

A la Figura 3.1 podem veure el diagrama de Gantt de la planificació temporal del projecte.

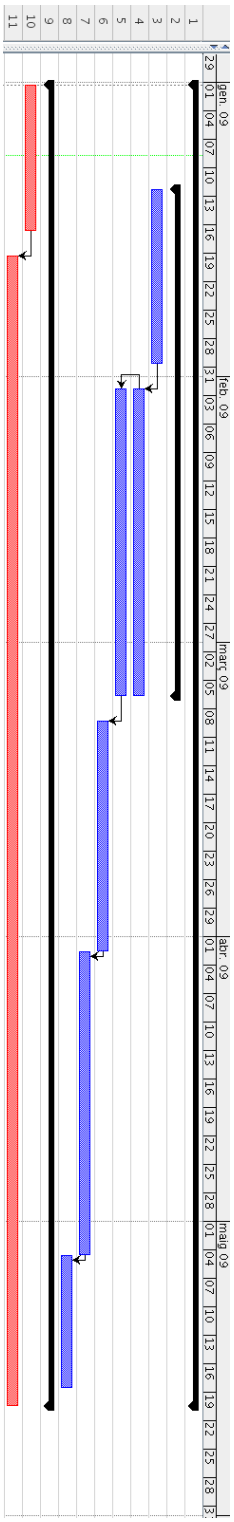

Figura 3.1: Diagrama de gantt del projecte

3.3.3 Pressupost

A continuació es presenta el pressupost desglossat tenint en compte que l'enginyer superior que es dedicarà a fer el projecte cobra 20 € per hora de treball.

Cal comentar que en aquest projecte només es contempla el cost del treball de l'enginyer ja que no hem hagut de comprar material informàtic i el departament ens l'ha proporcionat.

Concepte	**Duració**	**Cost**
Estudi del sistema existent	65h	1300 €
Implementació	30h	600 €
Proves	30h	600 €
Integració	20h	400 €
Redacció documents	80h	1600 €
Projecte	**225h**	**4500 €**

Capítol 4

Disseny de l'arquitectura de serveis

Crearem una arquitectura que actuarà com a intermediari entre l'aplicació i els agents que proporcionen serveis. A continuació es realitza el disseny d'ADASMI (*Architecture for Dynamic Agent Services Management and Interaction*), començant per un anàlisi de la part principal de l'arquitectura (veure 4.1) i a continuació fent un disseny orientat a objectes per a una posterior implementació (veure 4.1.2).

Necessitem definir una arquitectura que suposi el mínim impacte en l'aplicació i proporcioni un mecanisme d'interacció entre aquesta i els nous serveis.

4.1 L'agent *Service Manager Agent*

La nova arquitectura de serveis ha de permetre a l'aplicació accedir als serveis sense preocupar-se de com estiguin implementats. Així doncs apareix la figura de l'SMA (*Service Manager Agent*). Aquest nou agent servirà per a definir l'arquitectura de serveis del nostre sistema.

Com podem veure a la Figura 4.1, l'agent SMA permet a l'aplicació la utilització de serveis sense necessitat d'accedir al DF. D'aquesta manera és possible utilitzar un altre mecanisme d'anunci i descobriment de serveis o fins i tot emprar una altra tecnologia per a la plataforma d'agents.

Figura 4.1: Situació de l'agent SMA en una arquitectura de capes

4.1.1 Aspectes clau en el disseny de l'SMA

A partir d'aquest disseny inicial sorgeixen dues possibilitats que afectaran a la implementació de l'agent SMA:

- Utilitzar un agent diferent per a cada servei. Aquest agent es comunicarà amb l'agent SMA i l'aplicació.

- Delegar la responsabilitat dels serveis a l'SMA fent que sigui ell qui proporcioni els serveis directament.

A continuació s'analitzaran els avantatges i inconvenients de les dues opcions.

L'agent SMA proveeix el serveis directament

En aquest cas l'agent SMA és el productor de serveis i es converteix en un element indispensable per a utilitzar serveis en la nostra aplicació (veure figura 4.2).

El temps de resposta també es pot veure reduït ja que l'execució del servei també serà realitzada per l'agent i perjudica la paral·lelització de tasques. És cert que en cas d'utilitzar JADE com a plataforma pels nostres agents podem decantar-nos per una solució que permeti la concurrència, com per exemple els *behaviours* paral·lels.

Tot i així, es desitja que l'agent SMA pugui ser utilitzat en altres plataformes que poden emprar una tecnologia diferent a JADE. Per tant, no podem basar el nostre disseny en una tecnologia concreta.

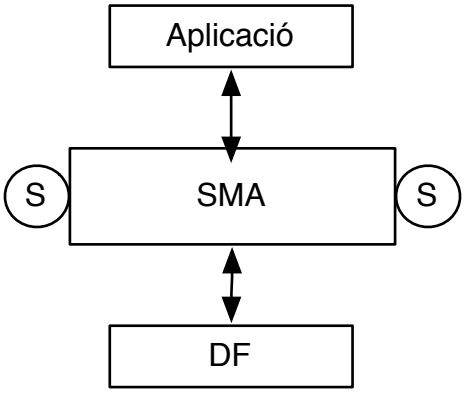

Figura 4.2: Primera alternativa per a dissenyar l'agent SMA

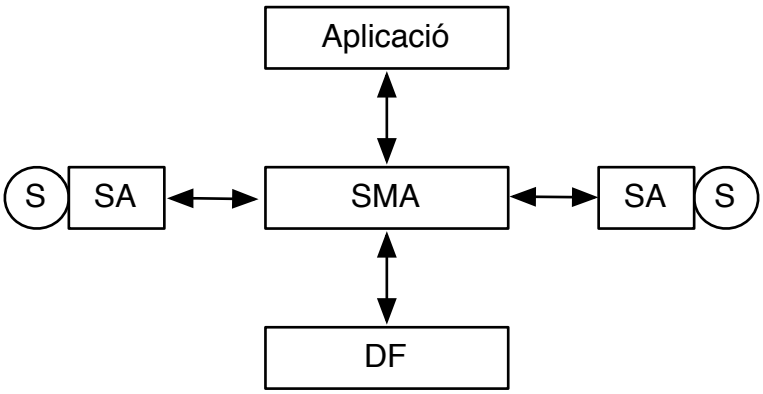

Figura 4.3: Segona alternativa per a dissenyar l'agent SMA

L'agent SMA coordina diferents agents que proporcionen serveis

Amb aquesta solució es pretén reduir la càrrega de l'SMA i delegar l'execució de serveis per agents autònoms (veure Figura 4.3).

Així doncs en aquest escenari podem veure l'agent SMA com un coordinador d'agents facilitant la interacció entre aplicació, serveis i el DF.

No obstant, l'agent SMA esdevé més abstracte i això pot afectar a la usabilitat de l'arquitectura de serveis.

Decisió final

Hem analitzat els punts forts i els punts febles de les dues alternatives. Utilitzarem l'agent SMA com a coordinador d'altres agents que proporcionin serveis a la nostra aplicació.

Restringirem la funcionalitat de l'agent SMA a coordinar aquests agents i comunicar-se amb l'aplicació i el DF.

Aquesta decisió s'ha pres valorant les característiques més importants que s'han comentat durant el disseny inicial de l'agent. Volem que l'agent SMA sigui un element opcional, independent de la tecnologia d'agents emprada i fàcil d'utilitzar per a poder explotar el màxim les seves possibilitats.

4.1.2 Disseny de classes

Després d'analitzar les característiques del nou agent i avaluar les diferents possibilitats d'integració amb el sistema actual es dissenyaran les classes Java que s'implementaran durant el projecte.

Es realitzarà el disseny sense tenir en compte la implementació per tal de concentrar-se en les funcionalitats que volem que tingui l'agent SMA. En capítols posteriors es veurà com s'ha realitzat la implementació durant el projecte adaptant el disseny en funció de les necessitats del sistema actual.

Veiem a la Figura 4.4 el diagrama de classes generals que formen l'arquitectura de serveis.

L'agent SMA i els agents SA

Per a la implementació de l'agent SMA (veure Figura 4.5) es crearà una classe encarregada de tota la gestió de serveis. Els atributs principals de la classe són els serveis registrats, els modes d'operació permesos, les extensions instal·lades i finalment una interfície amb la gestió de serveis a baix nivell.

Per a implementar els diferents serveis s'utilitzarà la classe base `SA` (veure figura 4.6) que encapsula la part comuna de tots els serveis, bàsicament el registre i desregistre del servei.

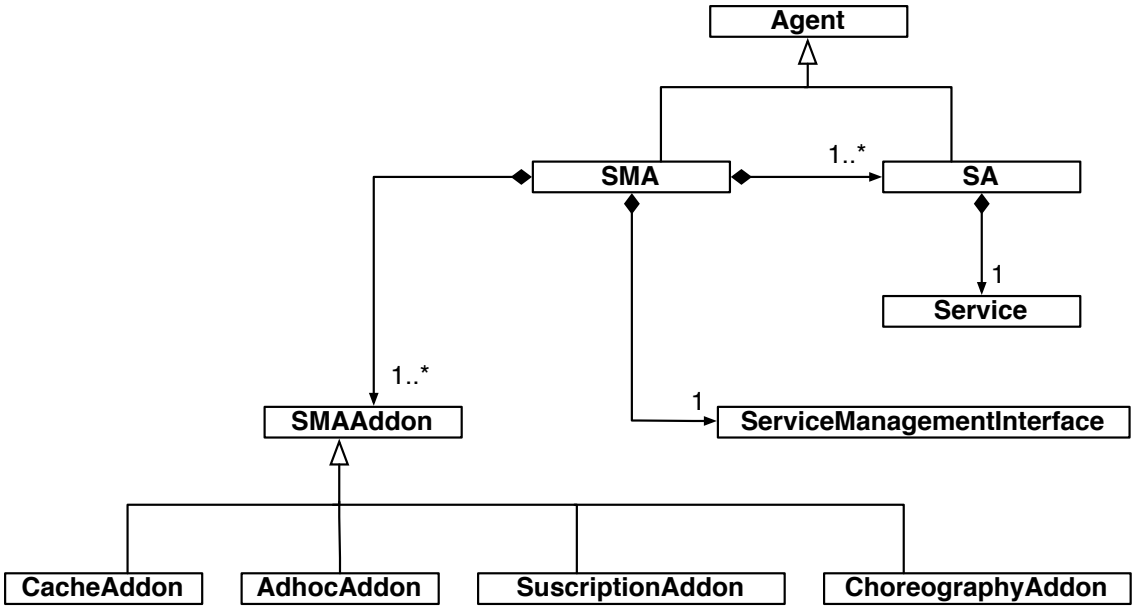

Figura 4.4: Diagrama de classes de l'arquitectura de serveis

SMA
ArrayList<String> supportedModes ArrayList<Map<AgentID,String>> registeredServices ServiceManagementInterface serviceManagementInterface ArrayList<SMAAddon> installedAddons
void addServiceAgent(AgentID agent, String serviceName) void removeServiceAgent(AgentID agent) void useService(String serviceName, String mode, ArrayList args) ArrayList<String> getRegisteredServices() void addAddon(SMAAddon addon) void removeAddon(String name) ArrayList<String> getInstalledAddons() ArrayList<String> getSupportedModes()

Figura 4.5: Diagrama de la classe `SMA` detallat

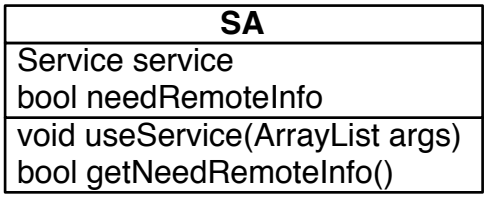

Figura 4.6: Diagrama de la classe `SA` detallat

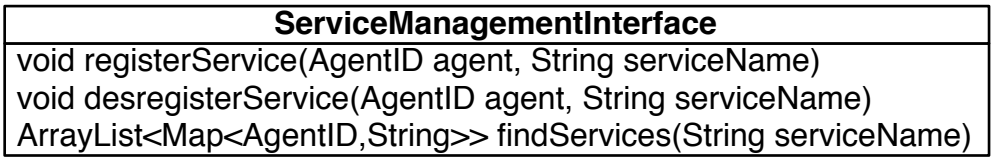

Figura 4.7: Diagrama de classe `ServiceManagementInterface` detallat

Interfície per a gestionar l'anunci i descobriment de serveis

L'agent SMA s'encarregarà de gestionar el serveis a alt nivell. Tot i així, si alguns agents volen utilitzar altres mecanismes, l'agent SMA propagarà totes les accions que es realitzin amb serveis a la capa inferior, com per exemple el `Directory Facilitator`.

Per aquesta raó l'agent SMA disposa d'una interfície amb la capa superior i utilitzarà la classe `ServiceManagementInterface` (veure Figura 4.7).

Estenent la funcionalitat de l'agent SMA

Algunes funcionalitats de l'arquitectura no són necessàries en tots els casos, per exemple si la fiabilitat de les dades és més important que el temps de resposta, aleshores no utilitzarem mecanismes de *cache*. En aquests casos es prefereix que l'arquitectura no ocupi gaire espai i el disseny s'ha realitzat de tal manera que les funcionalitats de l'agent SMA es puguin ampliar mitjançant extensions (veure Figura 4.8).

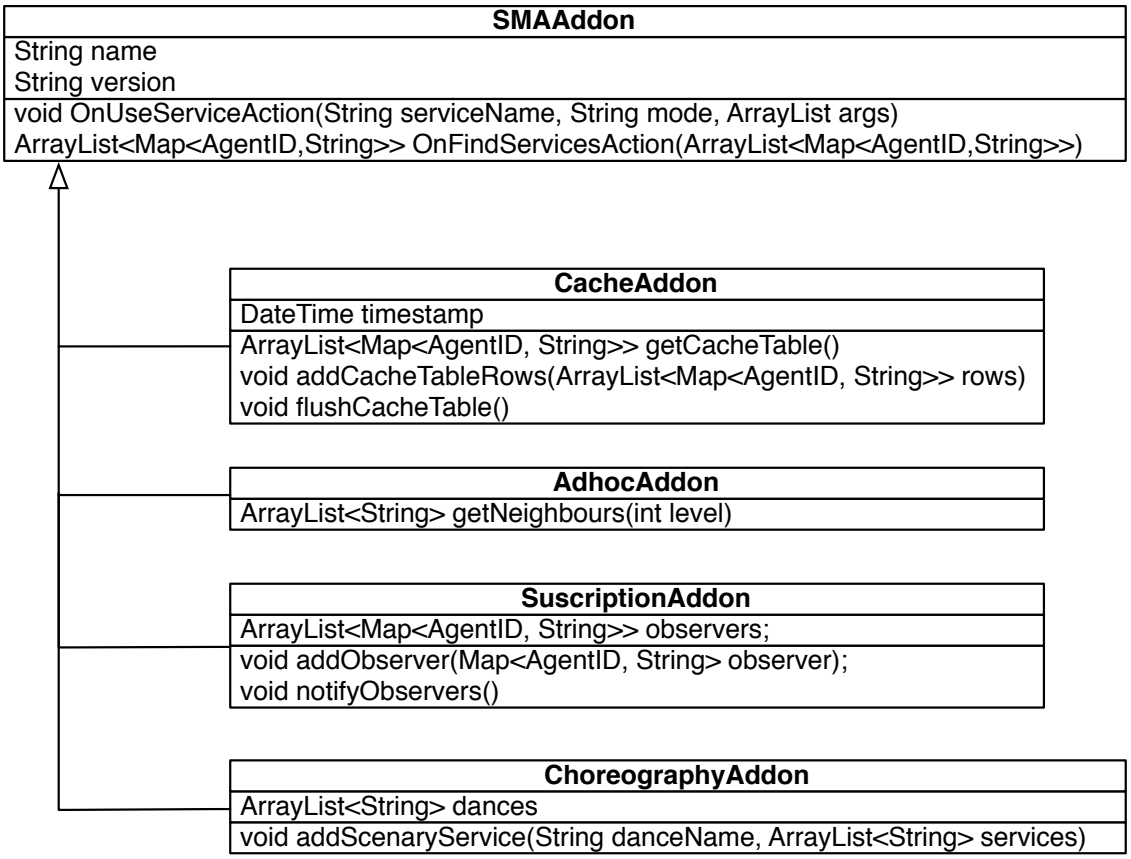

Figura 4.8: Diagrama de classes dels *addons* de l'agent SMA

4.2 Avantatges de la nova arquitectura

L'arquitectura que hem dissenyat ens aporta els següents avantatges:

- **Independent de l'aplicació:** qualsevol canvi en la implementació dels serveis és transparent de la utilització d'aquest per part de l'aplicació.
- **Integració:** ens facilita la integració dels serveis en el sistema encapsulant tasques com el registre, desregistre i cerca de serveis.
- ***Cache*:** proporciona mecanismes de *cache* per accedir als serveis ràpidament i amb un temps de resposta baix, molt important en l'entorn on ens trobem.
- **Fiabilitat:** permet centralitzar el tractament d'errors en cas de no poder accedir a un determinat servei o les excepcions que puguin aparèixer durant l'ús d'aquests.
- **Sincronització en xarxes *ad-hoc*:** en un entorn dinàmic com les xarxes MANET és necessari que la transmissió de dades entre els diferents nodes estigui ben sincronitzada. Permet consultar quins nodes són visibles i evitar la pèrdua de paquets durant la utilització de serveis.
- ***Choreography*:** un determinat servei pot requerir la participació d'altres serveis per a ser utilitzats. La nova arquitectura permet coordinar els diferents serveis amb aquesta finalitat.

4.3 Disseny de serveis

4.3.1 Classificació

Abans de començar a dissenyar cada un dels serveis que es volen implantar en el sistema cal fer una distinció entre ells. En primer lloc caldrà distingir els serveis segons la seva zona d'aplicació:

- **Zona 0:** zona de l'emergència. El ECC farà de nexe entre aquesta zona i la zona 1.

- **Zona 1:** zona de transport. Els vehicles faran de nexe entre aquesta zona i la zona 2.
- **Zona 2:** zona d'hospitals.

A cada zona trobem una sèrie d'agents productors i consumidors de serveis propis de la mateixa zona. Tot i així hi ha alguns agents que interactuen amb la zona immediatament superior i poden consumir o oferir serveis en aquella zona.

Tindrem una segona classificació en quant a la seva importància o utilitat:

- **Serveis primaris:** són aquells serveis essencials durant el rescat d'una víctima.
- **Serveis secundaris:** són aquells serveis que poden ser útils en determinades situacions però no són essencials per al benestar de les víctimes.

A continuació detallarem alguns dels serveis que s'han previst que s'utilitzaran a cada zona fent la distinció entre primaris i secundaris. Primer veurem un diagrama de casos d'ús per a tenir una visió general dels serveis d'una determinada zona i seguidament trobarem una explicació detallada de cada un dels serveis acompanyats per diagrames de seqüència.

4.3.2 Zona 0: emergència

Quan hi ha una gran emergència el primer que s'ha de fer és realitzar un triatge de les víctimes, és a dir, decidir l'estat i la urgència de les ferides de les víctimes. En aquesta zona l'objectiu principal és obtenir la informació del triatge des de les PDAs de l'equip mèdic i fer arribar aquesta informació a l'ECC.

En el moment d'analitzar l'estat de la víctima es crea un agent mòbil anomenat ETTMA (*Electronic Triage Tag Mobile Agent*). Aquest agent conté la informació del triatge que haurà de ser entregada al centre de coordinació. Així doncs aquest agent mòbil cercarà el servei `CheckInVictim`, el qual li proporciona un espai on desar aquestes dades. Per decidir quin és el millor camí per arribar a l'ECC sol·licitarà el TTR als equips disponibles a la xarxa. Aquesta sol·licitud es durà a terme gràcies al servei `GetRoutingInfo`.

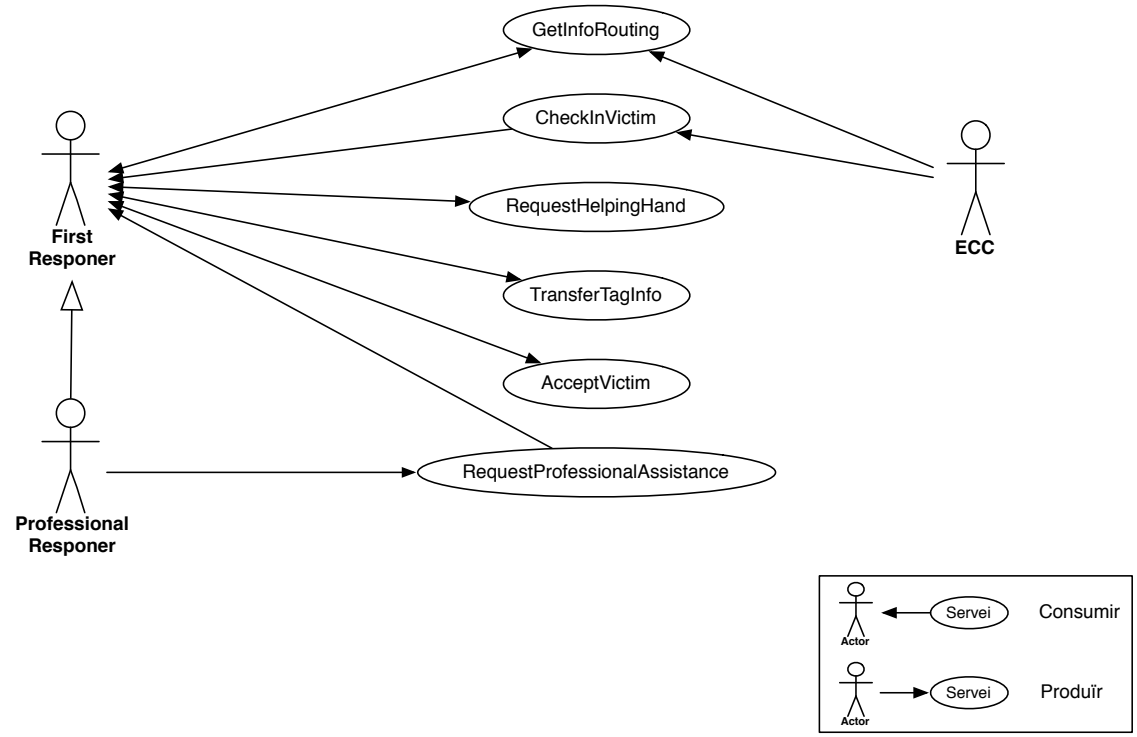

Figura 4.9: Serveis disponibles a la zona 0

Els dos serveis esmentats anteriorment són els serveis primaris de la zona 0. Veiem ara un diagrama de casos d'ús on es mostren els actors (productors i consumidors) i els serveis disponibles:

`GetRoutingInfo`

L'agent mòbil ETTMA s'encarrega de portar les dades de la víctima cap al centre de coordinació. Per a determinar la millor ruta d'encaminament es basa en trobar quin dels seus veïns té un TTR (*Time To Return*) més petit.

El servei `GetRoutingInfo` serveix per a sol·licitar informació referent a l'encaminament, com per a exemple el valor del TTR. El servei està preparat per proporcionar qualsevol altre tipus d'informació.

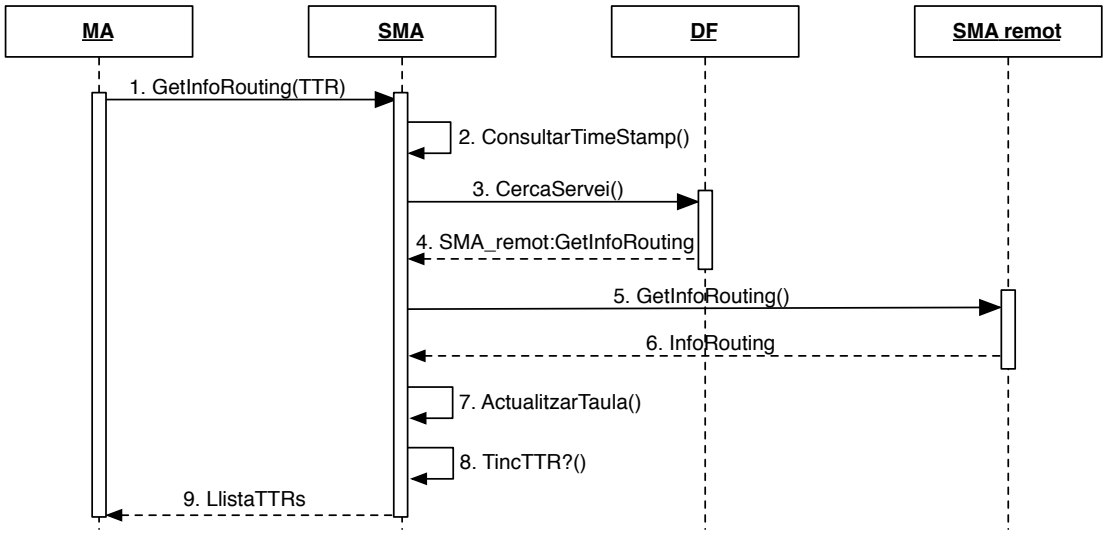

Figura 4.10: Diagrama de seqüència del servei `GetRoutingInfo` (primer cas)

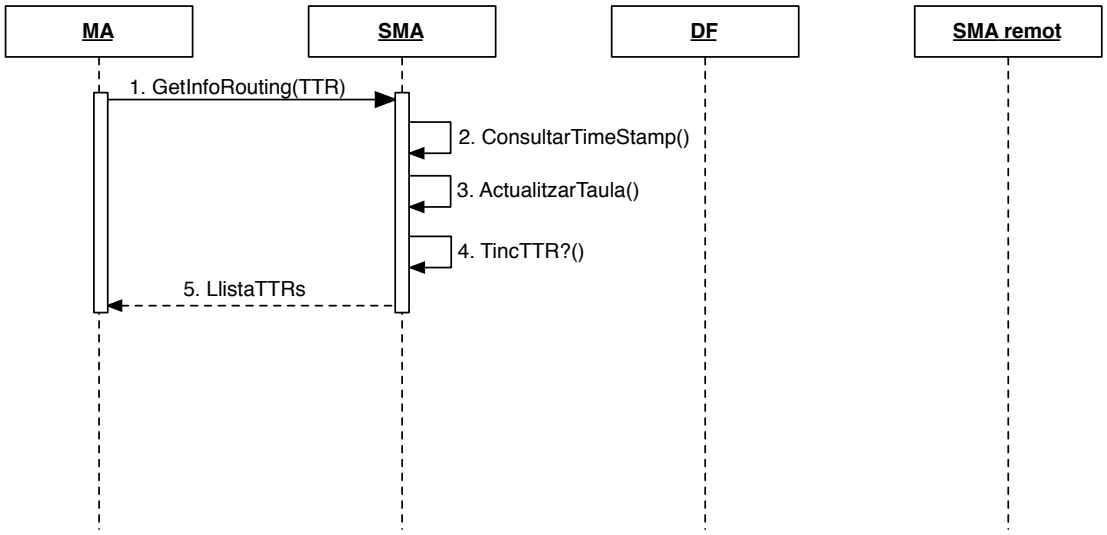

Figura 4.11: Diagrama de seqüència del servei `GetRoutingInfo` (segon cas)

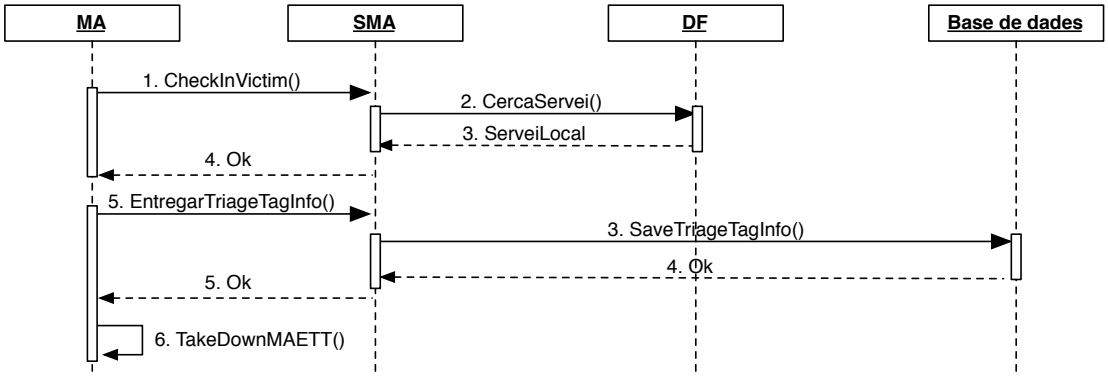

Figura 4.12: Diagrama de seqüència del servei `CheckInVictim`

`CheckInVictim`

La informació del triatge s'ha d'entregar al centre de coordinació. A la plataforma de l'ECC hi haurà un agent que produirà el servei `CheckInVictim` encarregat de registrar la informació que vagi arribant.

L'objectiu principal de l'agent ETTMA serà consumir aquest servei per a poder desar les dades que transporta.

`RequestHelpingHand`

Quan l'equip mèdic arriba a la zona de l'emergència a priori no sap en quines condicions es trobaran les víctimes o l'entorn. Es pot donar la situació en què un assistent mèdic necessiti l'ajuda d'una o més persones per a realitzar una tasca (p.e. moure mobiliari per a alliberar una persona).

Les persones que no estiguin atenent cap víctima estaran disponibles per a ajudar altres membres de l'equip. Això quedarà representat a partir de l'anunci del servei `RequestHelpingHand`. En el moment de començar a realitzar el triatge es deixaria d'anunciar aquest servei indicant que aquella persona està ocupada.

En el moment de sol·licitar aquest servei apareixerà un missatge a la PDA indicant qui sol·licita ajuda i la seva localització.

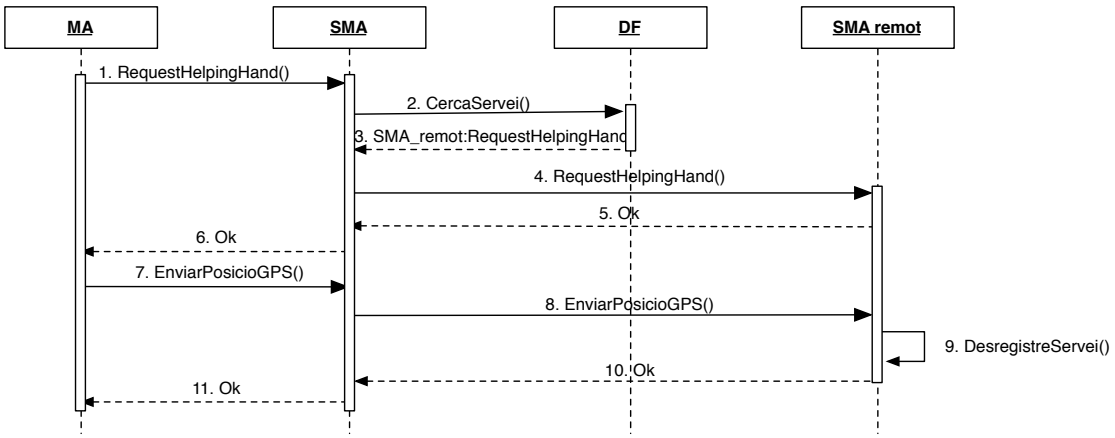

Figura 4.13: Diagrama de seqüència del servei `RequestHelpingHand`

`RequestProfAssistance`

Aquest servei es pot descriure d'una forma molt semblant al servei anterior amb la diferència que en aquest cas sol·licitem ajuda relacionada amb una tasca més específica.

Quan l'equip mèdic es troba una víctima es sol estabilitzar, depenent del seu estat, fins que arribi el transport adequat a recollir la víctima. En alguns casos el personal que estigui atenent la víctima necessitarà realitzar una tasca molt específica i pot requerir un personal especialitzat en un determinant camp (p.e. un traumatòleg).

El servei `RequestProfAssistance` serà anunciat mentre l'equip especialitzat estigui disponible.

`TransferTagInfo`

Durant el triatge s'utilitza un dispositiu mòbil per a fer un primer reconeixement de la víctima. De vegades sorgeixen complicacions que requereixen una cura de primers auxilis de la víctima. El servei `TransferTagInfo` ens pot servir per a delegar la responsabilitat d'una determinada víctima a un altre metge.

En aquest cas es passa la informació de triatge a un altre metge i aquest serà

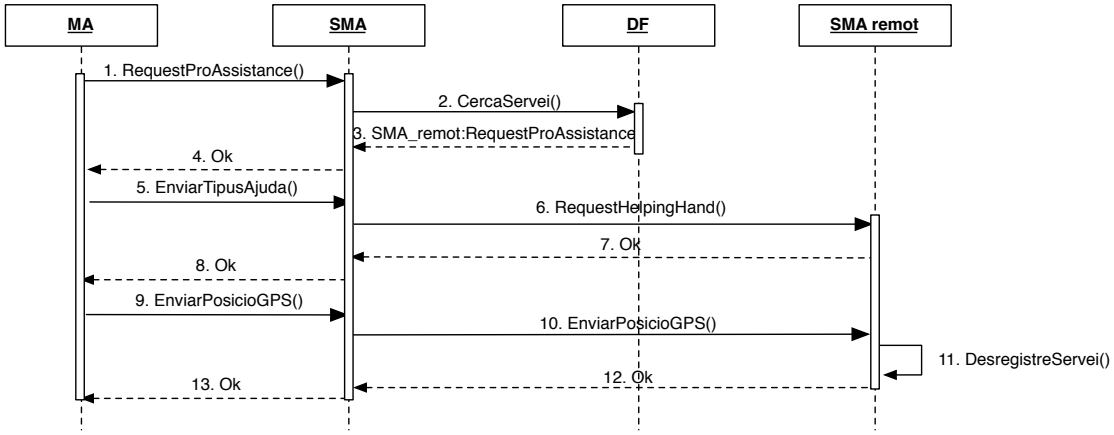

Figura 4.14: Diagrama de seqüència del servei `RequestProfAssistance`

l'encarregat d'enviar l'ETTMA quan sigui necessari.

`AcceptVictim`

Aquest servei és molt semblant al servei `CheckInVictim` amb la diferència que és proporcionat per un altre metge i no per l'ECC. Permet rebre la informació de triatge d'una víctima abans d'enviar-la al centre de control.

4.3.3 Zona 1: transport

A mesura que el centre de control va rebent la informació de triatge de les víctimes, aquest s'encarrega de coordinar tots els mitjans de transport disponibles per a traslladar les víctimes a una zona segura (normalment un hospital).

En aquesta zona trobarem una sèrie de serveis relacionats amb el transport. De la mateixa manera que a la zona 0 distingirem entre serveis primaris i serveis secundaris. El primer que caldrà fer en obtenir la informació de triatge d'una víctima serà consultar la informació del transport disponible mitjançant el servei `TransportRequest`. Un cop sabem de quins recursos disposem utilitzarem el servei `VictimRescue` proporcionat per cada transport disponible per donar l'ordre de rescat.

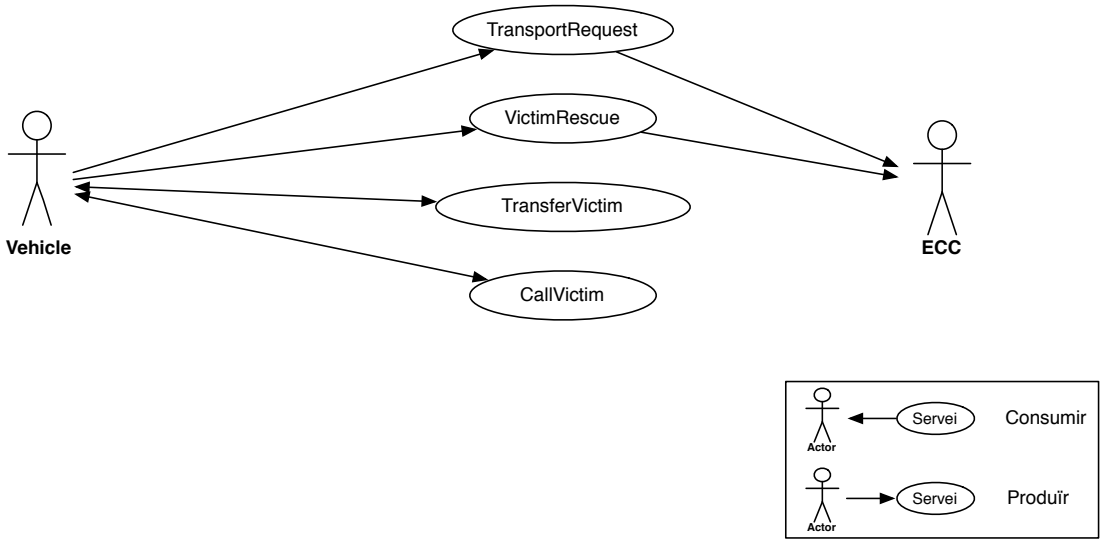

Figura 4.15: Serveis disponibles a la zona 1

`TransportRequest`

Aquest primer servei proporcionat per cada unitat de transport s'encarrega de proporcionar la informació necessària al centre de control. Podem sol·licitar informació referent al vehicle (mitjà de transport, nombre de víctimes que pot transportar, ...) així com informació referent a l'últim servei realitzat.

Aquest servei primari és un primer pas per a coordinar el rescat d'una víctima. Un cop el centre de control té la informació necessària es disposarà a donar l'ordre de rescat.

`RescueVictim`

Si un vehicle ofereix aquest servei significa que està disponible per a realitzar un servei de rescat. Quan el centre de coordinació sol·licita aquest servei haurà de proporcionar la informació de triatge al mitjà de transport.

Si el vehicle disposa de prou capacitat es pot sol·licitar el rescat de múltiples víctimes.

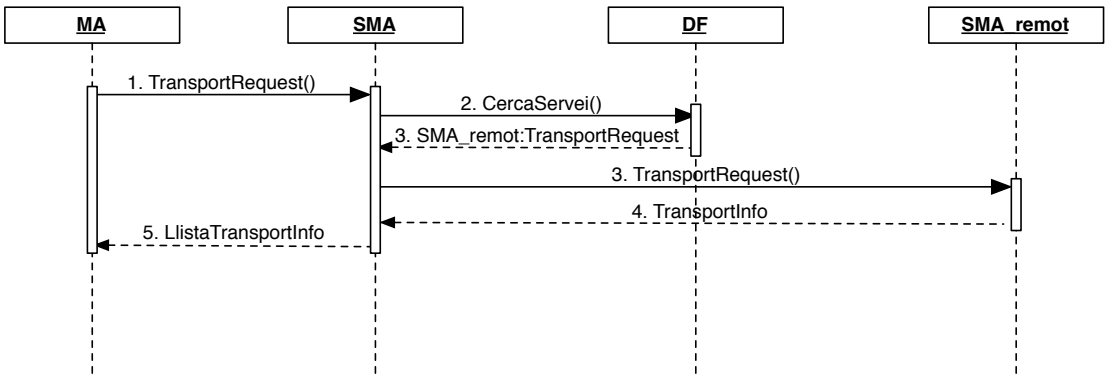

Figura 4.16: Diagrama de seqüència del servei `TransportRequest`

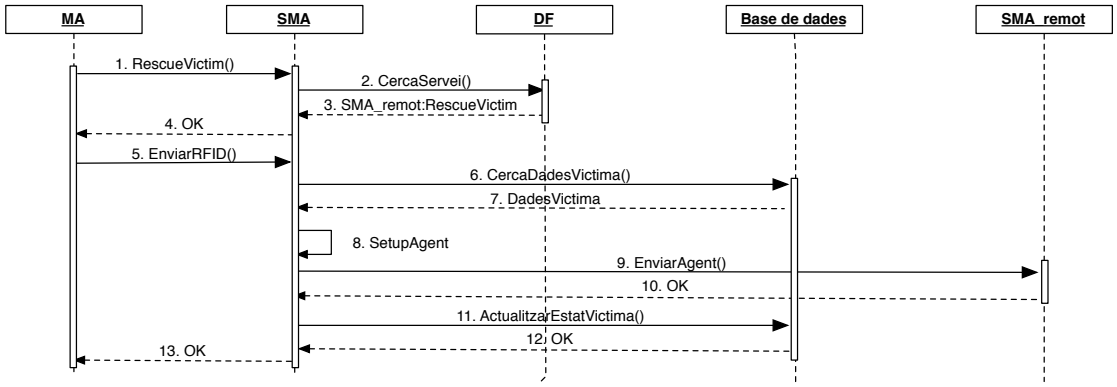

Figura 4.17: Diagrama de seqüència del servei `RescueVictim`

`TransferVictim`

En determinades ocasions el vehicle que realitza el rescat de la víctima no és el mateix que s'encarrega de transportar la víctima a l'hospital més pròxim.

En aquest cas, un cop la víctima ha canviat de transport, el transport que ha realitzat el rescat d'aquesta sol·licita el servei `CallVictim` a l'altre mitjà de transport. Un cop acceptat el canvi, el transport ofereix aquest servei per a transferir la informació de triatge a l'altre transport.

`CallVictim`

Tal i com hem comentat en la descripció del servei `TransferVictim`, el servei `CallVictim` és proporcionat pel mitjà de transport que s'encarregarà de portar la víctima al hospital.

Aquest servei serà ofert per aquells mitjans de transport que estiguin disponibles únicament per a realitzar el transport de la víctima cap a l'hospital.

4.3.4 Zona 2: hospitals

Les víctimes que pateixin lesions greus i necessitin hospitalització seran transportades cap a diferents hospitals per a rebre atenció mèdica.

Quan s'està realitzant el transport de la víctima seria interessant informar a l'hospital destinatari que tingués preparat el quiròfan en el cas d'intervenció, reservat el material per a fer una determinada cura o bosses de sang per a realitzar una transfusió. En aquesta zona trobarem el servei primari `AdmitVictim` que ens permetrà completar el cicle de rescat d'una víctima i una sèrie de serveis secundaris molt útils, entre ells la possibilitat d'obtenir un resum de l'historial mèdic del pacient.

`AdmitVictim`

Els hospitals que puguin subministrar atenció mèdica a les víctimes d'una determinada emergència oferiran el servei `AdmitVictim` per a acceptar la informació

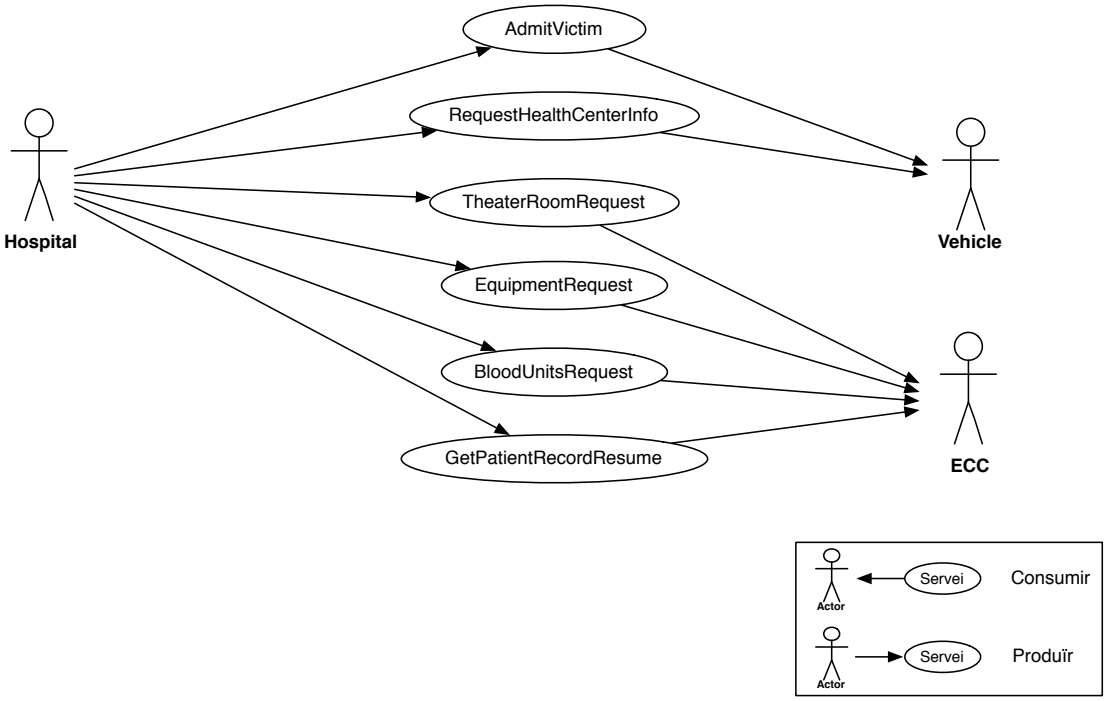

Figura 4.18: Serveis disponibles a la zona 2

de triatge d'una víctima. D'aquesta manera disposaran de les dades de la víctima abans de realitzar la intervenció corresponent.

RequestHealthCenterInfo

Quan el vehicle que transporta una o més víctimes ha de decidir a quin hospital les ha de portar necessitarà consultar quina és la disponibilitat dels hospitals del voltant. El servei `RequestHealthCenterInfo` proporciona informació relacionada amb el centre hospitalari, com per exemple el nombre de llits disponibles, informació sobre les assegurances mèdiques acceptades o si l'hospital pot tractar o no una determinada malaltia.

TheaterRoomRequest

Per a gestionar els diferents quiròfans dels quals disposa un determinat hospital s'utilitzarà el servei `TheaterRoomRequest` per a reservar un quiròfan. Aquest servei pot resultar molt útil per a poder reservar el quiròfan abans que el pacient arribi a l'hospital de tal manera que quan aquest arribi al centre el quiròfan estigui correctament condicionat i preparat per a la intervenció.

EquipmentRequest

Algunes intervencions poden requerir un cert instrumental que necessiti una preparació prèvia o sigui difícil de aconseguir. En aquests casos es pot reservar aquest material utilitzant el servei `EquipmentRequest`. Aquest servei serà utilitzat prèviament al trasllat de la víctima amb el servei `TheaterRoomRequest` per a poder atendre a la víctima en el mateix moment que arribi a l'hospital.

BloodUnitsRequest

De la mateixa manera que el servei anterior, el servei `BloodUnitsRequest` ens permetrà reservar un determinat nombre de bosses de sang d'un determinat tipus. Tot i ser considerat un servei secundari en determinades circumstàncies aquest servei pot significar poder salvar una víctima a temps. En el moment de

traslladar la víctima cap a l'hospital podem sol·licitar unitats de sang del tipus de la víctima per a poder realitzar la transfusió en el moment del seu ingrés al centre.

`GetPatientRecordResume`

Durant la vida d'una persona aquesta assistirà a diversos hospitals i per tant disposarà d'un historial mèdic distribuït. Si és possible identificar a una determinada víctima pot resultar molt interessant recopilar un resum de l'historial mèdic del pacient abans de subministrar-li un determinat medicament. Aquest servei ens proporciona un resum de l'historial del pacient determinat per tots els historials mèdics dels quals disposi en diferents hospitals.

Capítol 5

Implementació

Després d'analitzar el problema i dissenyar una possible solució, és el moment d'emprar una tecnologia concreta per a implementar-la. En el nostre cas utilitzarem el llenguatge Java i la plataforma *software* JADE per a la creació i gestió dels agents.

5.1 Paquets desenvolupats

El *software* implementat està organitzat en diferents classes Java i a la vegada aquestes classes s'organitzen en diferents paquets segons la seva utilitat. A continuació es llisten els paquets que s'han desenvolupat al llarg del projecte:

`adasmi.common` En aquest paquet trobem les classes que formen la part principal de l'arquitectura, l'agent SMA.

`adasmi.ontologies` Per a la comunicació entre els diferents agents s'utilitzen diferents ontologies. En aquest paquet trobem la definició de cada ontologia.

`adasmi.services` Durant el projecte s'han desenvolupat un conjunt de serveis primaris. Per cada servei trobem un agent que el representa així com la classe base per a tots els serveis.

`adasmi.utilities` Aquest paquet està format per algunes classes que faciliten la implementació del projecte.

`adasmi.test` L'últim paquet desenvolupat està format per totes aquelles classes utilitzades per a comprovar el correcte funcionament de l'arquitectura.

Començarem presentant la solució implementada per a la part principal de l'arquitectura. A continuació anirem detallant el contingut dels altres paquets i presentarem els agents i ontologies creades per a la utilització de serveis.

5.2 La classe `SMA`

5.2.1 Descripció general

L'agent SMA s'ha implementat tenint en compte els requisits analitzats durant el disseny i mantenint un compromís entre l'abstracció i la usabilitat d'aquest. Hem utilitzat la classe `jade.core.Agent` de `JADE` que ens proporciona una interfície per al desenvolupament d'agents.

Quan creem l'agent SMA inicialitzem la llista de serveis registrats i carreguem les extensions que es volen instal·lar. Aquestes extensions s'encarreguen d'ampliar la funcionalitat de l'agent SMA i són totalment opcionals. Per tal de saber quines extensions s'han de carregar hem de llegir el fitxer de configuració de l'agent.

Les funcionalitats bàsiques de l'agent SMA són el registre, desregistre, cerca i utilització de serveis. Veiem com les tres primeres funcionalitats són comunes a tots els serveis però en el darrer cas la utilització de serveis resulta específica per cada servei. Per tal d'interactuar amb l'agent SMA s'ha definit una ontologia (`adasmi.ontologies.serviceOntology`). Aquesta ontologia serà coneguda per tots els agents que vulguin utilitzar l'agent SMA per a la gestió i utilització de serveis.

A partir d'aquesta introducció de l'agent SMA podem extreure la necessitat de prendre una decisió quant a la utilització de serveis. Aquesta part resulta complicada ja que *a priori* sembla que depèn massa del servei en concret i això no ens interessa en el nivell d'abstracció on estem treballant. Així doncs ens plantegem dues opcions:

- Fer servir la ontologia de serveis per a tots els serveis que vulguem afegir.

- Crear una ontologia per a cada servei en concret.

Analitzarem les dues opcions, els avantatges i inconvenients de cadascuna d'elles i la solució que hem implementat.

Ontologia de serveis general

La primera opció consisteix a afegir el vocabulari corresponent a cada servei dins la ontologia de serveis general. Això significa que tots els agents que utilitzin l'arquitectura ADASMI només han de conèixer aquesta ontologia i l'agent SMA coneix informació sobre els serveis que té registrats.

L'avantatge principal d'aquesta opció és que l'agent pot participar en la utilització de un servei en concret ja que coneix l'anatomia d'aquest. Per contra, la opció no resulta escalable i perdem el nivell d'abstracció de l'agent SMA, condicionant-lo als serveis que coneix.

Ontologies específiques per a cada servei

La idea de crear una ontologia per a cada servei pot resultar poc atractiva en un principi. Suposem que a la plataforma on ens trobem hi ha tres agents que proporcionen tres serveis diferents. Això vol dir que estem treballant amb un total de quatre ontologies i *a priori* l'agent SMA ha de conèixer totes aquestes ontologies per tal que es puguin utilitzar els serveis a través d'ell.

L'avantatge principal d'aquesta solució és que resulta més escalable i modular que l'anterior. Cada nou servei que creem tindrà el seu propi vocabulari i els agents que vulguin utilitzar l'agent SMA com a interfície amb els serveis només hauran de conèixer la ontologia de serveis general i la seva pròpia. L'inconvenient que resulta més evident és que continuem perdent abstracció amb l'agent SMA i aquest ha de conèixer un conjunt d'ontologies que en alguns casos pot ser molt gran.

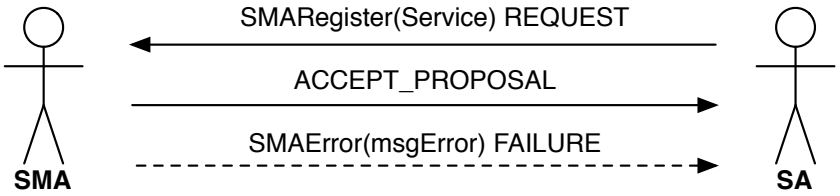

Figura 5.1: Registre d'un servei a l'agent SMA

5.2.2 Utilització de l'agent SMA

La prioritat és aconseguir que la interfície de serveis, l'agent SMA, sigui abstracte i usable pels altres agents. Tenint aquest objectiu en ment, sembla que cap de les dues opcions que hem analitzat anteriorment, quant a la utilització de serveis, resolgui el problema. Tot i així, es decideix emprar la segona opció amb alguna variació, adaptant-la a les nostres necessitats.

A continuació veurem de quina manera s'utilitza l'agent SMA com a interfície de serveis.

Registre i desregistre de serveis

Els agents en JADE executen tasques o comportaments (*behaviours*) durant el seu cicle de vida. L'agent SMA, un cop s'ha inicialitzat correctament comença a executar el *behaviour* `ServiceHandle` esperant diverses peticions.

A les Figures 5.1 i 5.2 podem veure l'intercanvi de missatges entre l'agent que proporciona el servei i l'agent SMA, en les operacions de registre i desregistre de serveis.

Quan l'agent SMA rep alguna d'aquestes dues peticions, a més a més d'actualitzar la seva pròpia llista de serveis també utilitza una interfície de baix nivell per registrar el servei a la plataforma, com per exemple al directori de pàgines grogues, el DF (*Directory Facilitator*).

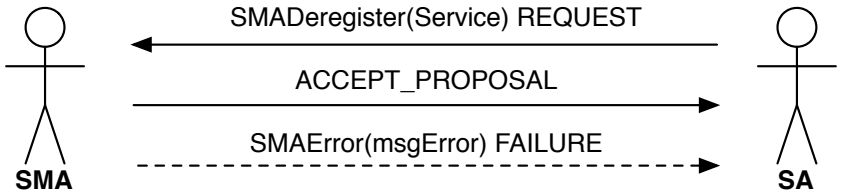

Figura 5.2: Desregistre d'un servei a l'agent SMA

Utilització de serveis

A l'Apartat 5.2.2 hem comentat les dues opcions que teníem per gestionar la utilització de serveis i per quina opció ens decantàvem després d'analitzar els avantatges i inconvenients de cadascuna.

Quan un agent vol utilitzar un determinat servei a través de l'agent SMA, aquest ha conèixer realment el contingut del missatge? Si volem aconseguir que l'agent SMA sigui una interfície, la resposta a aquesta pregunta és que no, el contingut del missatge hauria de ser transparent a l'agent SMA i aquest hauria de ser capaç d'entregar la informació a l'agent productor del servei.

Si suposem que l'agent SMA no ha de conèixer el contingut del missatge, aleshores no té sentit que hagi de conèixer la ontologia del servei corresponent i això afavoreix a la interfície de serveis. Per a aconseguir aquest objectiu, és necessari crear un mecanisme que encapsuli l'acció d'un determinat servei en un missatge genèric d'utilització de serveis.

A la Figura 5.3 observem quina conversa han de mantenir un agent consumidor d'un determinat servei, l'agent SMA i l'agent productor del servei. Com podem veure, l'agent SMA no coneix el contingut encapsulat que hi ha dins d'un missatge simple d'utilització de servei.

Encapsular el vocabulari específic de cada servei allibera l'agent SMA de conèixer totes les ontologies.

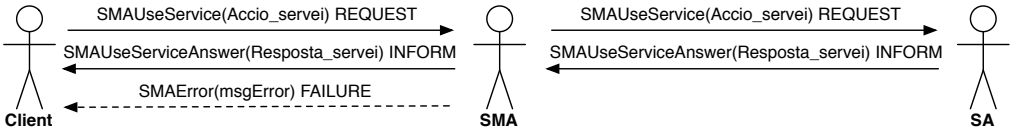

Figura 5.3: Desregistre d'un servei a l'agent SMA

Cerca de serveis

La cerca de serveis a l'SMA es realitza de forma molt semblant a una cerca en un directori de pàgines grogues, com per exemple el DF. Podem preguntar a l'agent SMA per tipus i nom del servei i ens retorna una llista amb els agents que hagi trobat. La primera cerca es realitza a nivell local, primer cercant el seu propi directori i després utilitzant un `ServiceManagementInterface` per a propagar la cerca.

En el missatge de cerca es pot especificar a quin nivell de profunditat es vol propagar la cerca. Després de realitzar la cerca local, l'agent SMA propaga la cerca als agents SMA remots.

Aquesta propagació de cerca requereix que l'agent SMA necessiti mantenir un estat amb les cerques que ha realitzat i les respostes que espera rebre. A la Figura 5.4 podem observar un esquema on un determinat agent realitza una cerca que es propaga a un SMA remot.

Aquesta primera aproximació de la cerca de serveis *a priori* presenta dos problemes:

- L'agent SMA ha de mantenir un estat i això pot donar lloc a errors si es perden alguns missatges.
- Un cop hem buscat els serveis, després els clients ens demanaran utilitzar aquell servei. Això significa enviar el doble de missatges i es podria fer aprofitant la pròpia cerca.

A partir d'aquests dos problemes es decideix millorar la cerca de serveis sense mantenir un estat a l'agent SMA i encapsulant una acció d'un determinat servei

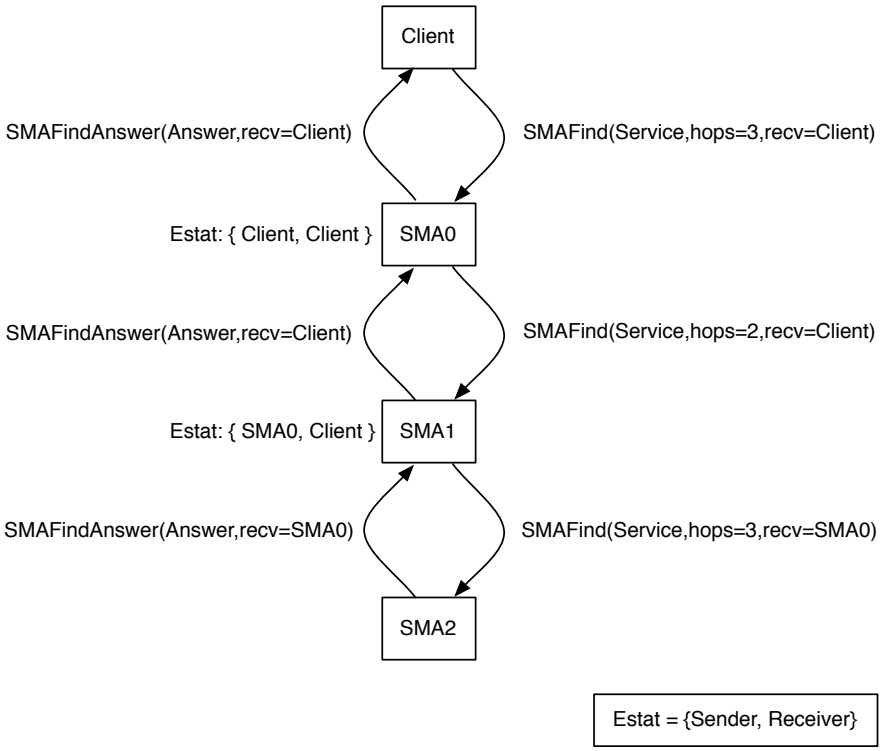

Figura 5.4: Cerca de serveis utilitzant l'agent SMA

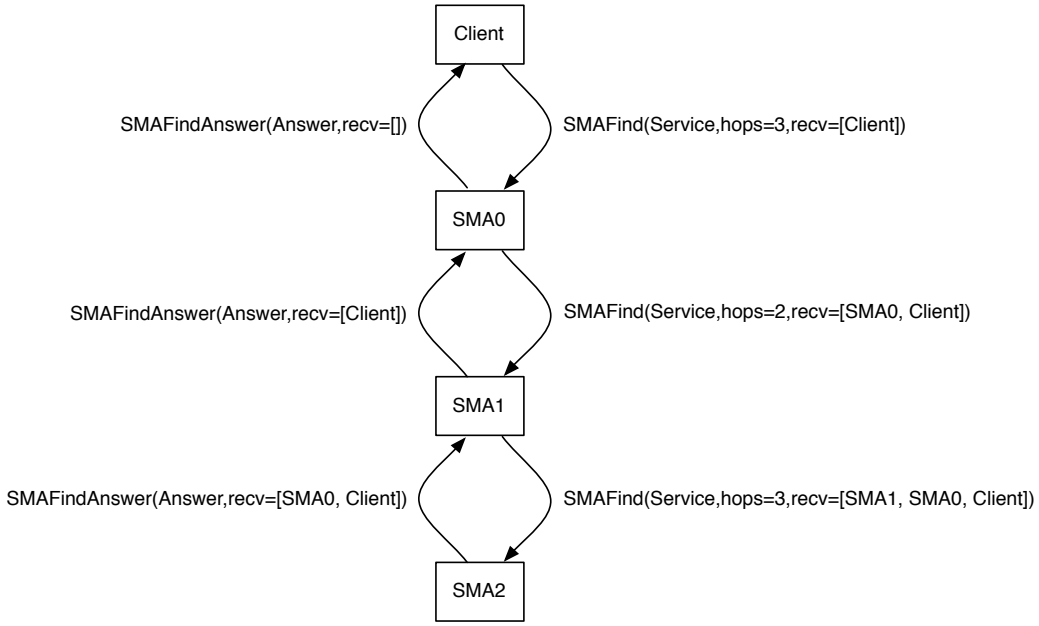

Figura 5.5: Millora en la cerca de serveis

en el mateix missatge de cerca. A la Figura 5.5 podem observar l'esquema de la cerca millorada.

5.3 Ontologies i serveis

Per a la implementació dels serveis s'ha hagut de definir un agent base que servirà per a ampliar l'arquitectura amb els nous serveis que vagin apareixent. L'agent SA (*Service Agent*) conté la funcionalitat bàsica de qualsevol agent que vulgui produir un servei.

Per a implementar un nou servei cal realitzar dos passos:

- Crear un nou agent com una classe derivada de la classe `SA`. Inicialitzar el servei especificant el seu nom i el seu tipus, abans de cridar el constructor de la classe base.

- Crear una ontologia per a aquest servei amb el vocabulari específic d'aquest.

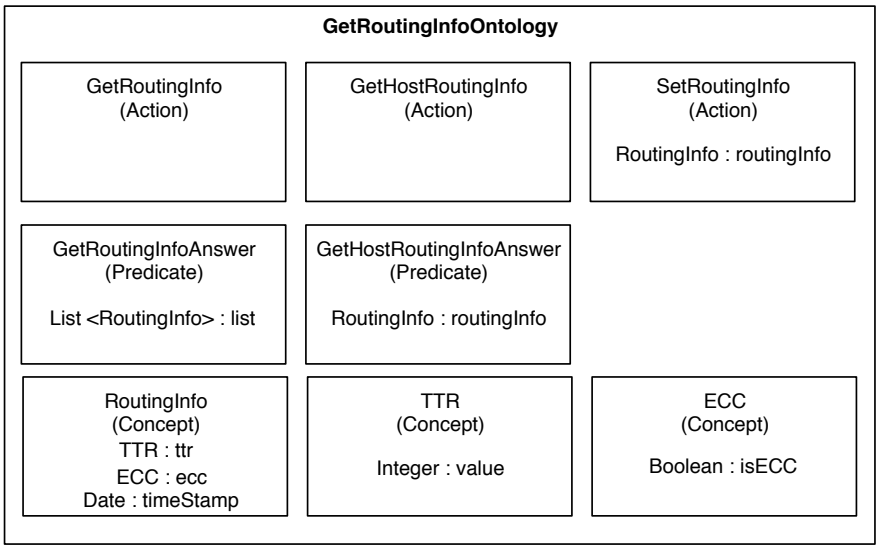

Figura 5.6: Ontologia del servei `GetRoutingInfo`

A la ontologia del nou servei cal definir els conceptes que s'utilitzaran, és a dir, els objectes amb els quals tractarem durant l'ús dels serveis. Per a poder utilitzar aquest servei caldrà definir una sèrie d'accions. A la Figura 5.6 podem veure l'exemple de la ontologia del servei `GetRoutingInfo`.

Capítol 6

Resultats

En aquest capítol es mostraran les proves realitzades per a validar el *software* realitzat. En primer lloc veurem les proves bàsiques de funcionament, registrant i desregistrant serveis a la nova arquitectura. A continuació s'analitzen algunes proves basades en escenaris, és a dir, es reconstrueixen algunes situacions reals per a comprovar que la participació entre diversos agents funciona correctament.

Per acabar, també s'han realitzat diferents proves d'integració amb el sistema actual i altres projectes relacionats.

6.1 Proves bàsiques de funcionament

Les següents proves s'han realitzat directament sobre l'agent SMA arrencant la plataforma JADE amb diversos agents que proporcionen serveis.

Registre de serveis

Primer arrenquem la plataforma amb l'agent SMA. Si tot va bé, ens apareix el missatge informant que l'agent s'ha creat correctament. A continuació creem un nou agent a la plataforma utilitzant la interfície gràfica de JADE.

Quan arranquem un nou agent que proporciona un servei aquest envia un missatge a l'agent SMA Per tal de registrar el servei. Aquesta és la sortida obtinguda:

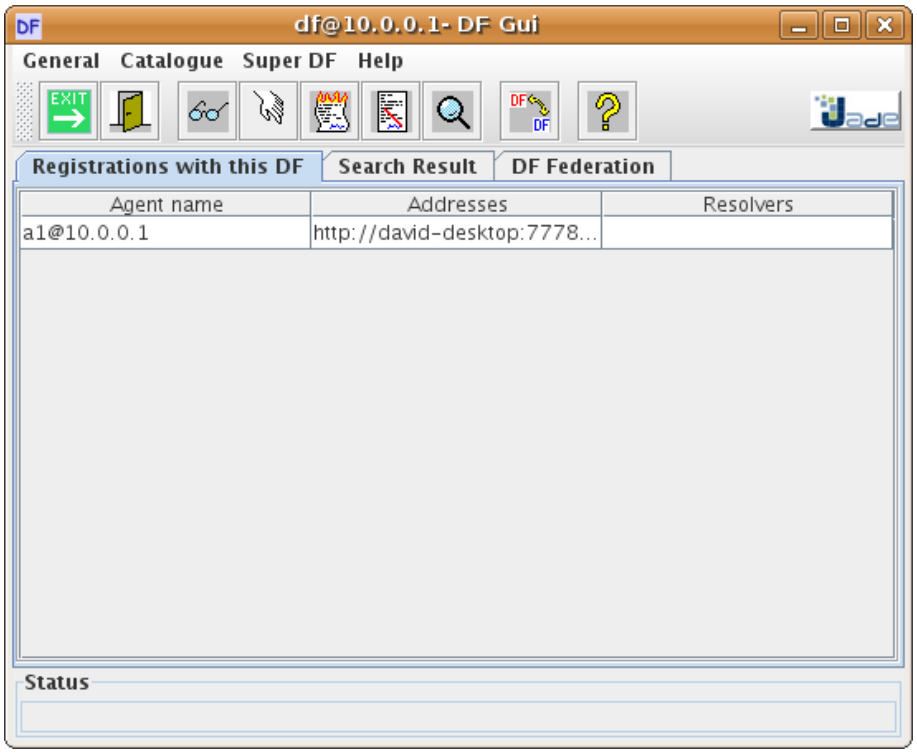

Figura 6.1: El servei queda registrat al DF correctament

```
SMA: L'agent SMA "SMA" s'ha creat correctament
a1: L'agent GetRoutingInfoSA "a1" s'ha creat correctament
SMA: Servei registrat: "getRoutingInfo" de l'agent "a1@10.0.0.1"
```

Utilitzant la interfície gràfica de JADE també podem veure si s'ha registrat correctament al DF. La Figura 6.1 mostra l'eina `DF GUI` per mostrar el contingut del *Directory Facilitator*.

Podem registrar múltiples agents que proporcionin el mateix servei i l'arquitectura segueix responent bé.

Desregistre de serveis

Un servei es pot desregistrar a causa de diversos factors, com ara que l'agent productor deixa d'oferir el servei en qüestió. Per a validar el correcte funcionament

hem utilitzat l'eina de JADE per a matar l'agent que oferia el servei. La sortida obtinguda en eliminar l'agent `a2` és la següent:

```
SMA: Servei deregistrat de l'agent "a2@10.0.0.1"
a2: L'agent GetRoutingInfoSA "a2" s'ha destruit correctament
```

També s'ha comprovat que el DF sigui consistent amb la nova arquitectura i s'observa com l'agent ja no està registrat al DF.

Cerca de serveis

L'agent SMA està preparat per a rebre la consulta de cerca de serveis. Aquesta cerca de serveis es pot realitzar de forma local o propagar la cerca als agents SMA remots. També podem utilitzar el propi missatge de cerca per a executar accions del servei que estem cercant.

En aquest primer cas, cercarem un servei que no està registrat ni a l'agent SMA ni al DF. El tipus de cerca que utilitzarem és local (`max-hops = 0`), deixant els altres casos per el següent apartat. Aquest és el resultat obtingut quan cerquem un servei que no ha estat registrat:

```
FindTestAgent: Find test ...
SMA: Searching service getRoutingInfo (max hops = 0) ...
FindTestAgent: Response received.
Error: No agents found providing service "getRoutingInfo"
```

Si registrem el servei a la nostra arquitectura de serveis, la resposta obtinguda és la següent:

```
FindTestAgent: Find test ...
SMA: Searching service getRoutingInfo (max hops = 0) ...
FindTestAgent: Response received.
1 agents found.
        ( agent-identifier :name a2@10.0.0.1
             :addresses (sequence http://david-desktop:7778/acc ))
```

En aquest cas hem trobat un agent que proporciona un servei local. Els casos més complexos (cerca remota i cerca amb acció) els deixem per a les proves basades en escenaris.

6.2 Proves basades en escenaris

Per a validar el correcte funcionament de l'arquitectura en casos semblants a la realitat, s'han creat dos escenaris en els quals han d'intervenir múltiples agents que produeixen i consumeixen serveis.

Hem limitat les proves a la zona 0 de l'escenari de les emergències. Aquestes proves utilitzen dos dels serveis implementats, el servei `GetRoutingInfo` i el servei `CheckInVictim`. Realitzarem dues simulacions ben diferents que acabaran de validar el correcte funcionament de l'arquitectura.

6.2.1 Primer escenari: `GetRoutingInfo`

L'esquema de la Figura 6.2 mostra la situació dels diferents agents que participen a la simulació. En aquest cas es mostra com participen diversos agents productors i consumidors de serveis. L'objectiu és que l'agent `MA` utilitzi un servei per a obtenir informació d'encaminament dels nodes veïns. Per a dur a terme aquesta tasca cal realitzar els següents passos:

- Registrar el servei `GetRoutingInfo` a totes les plataformes.
- Inicialitzar les dades que ha de proporcionar el servei a cada plataforma, mitjançant l'aplicació.
- Utilitzar el servei per obtenir la informació dels veïns.

La utilització del servei requereix realitzar una cerca amb acció que s'ha de propagar a la xarxa. Veiem primer els resultats que es mostren per pantalla:

```
SMA: Servei registrat: "getRoutingInfo" de l'agent "a1@david-pc:1099/JADE"
a1: L'agent GetRoutingInfoSA "a1" s'ha creat correctament
MA: Utilització del servei getRoutingInfo (SetRoutingInfo)
SMA: Utilització del servei getRoutingInfo reenviada a
   ( agent-identifier :name a1@david-pc:1099/JADE
   :addresses (sequence http://david-pc:7778/acc ))
a1: (SetRoutingInfo) Info Routing setup
Host: ( agent-identifier :name MA@david-pc:1099/JADE
   :addresses (sequence http://david-pc:7778/acc ))
        TTR: 15
```

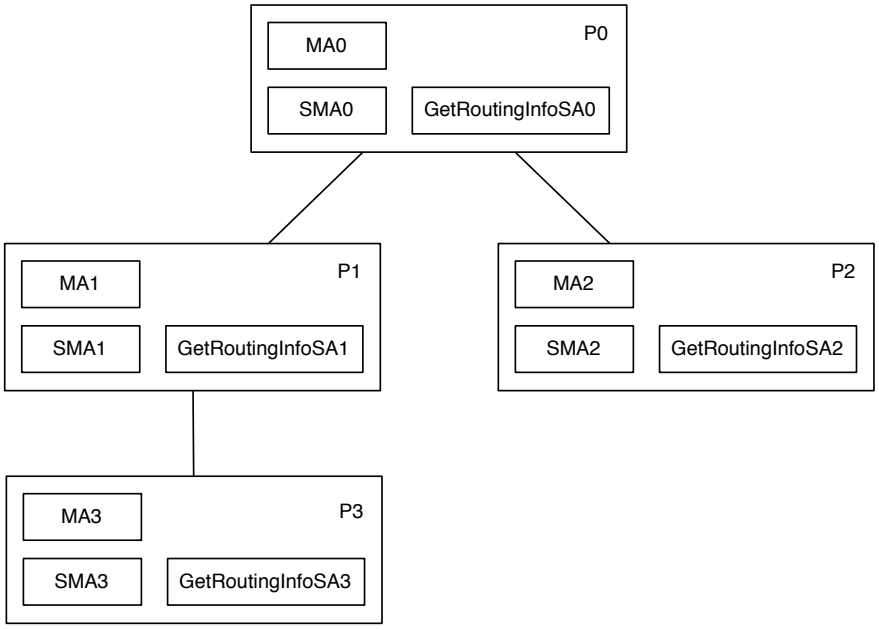

Figura 6.2: Esquema del primer escenari

```
        ECC: false
        Timestamp: Thu Jun 25 09:05:40 CEST 2009
MA: Utilització del servei getRoutingInfo (GetRoutingInfo)
SMA: Utilització del servei getRoutingInfo reenviada a
   ( agent-identifier :name a1@david-pc:1099/JADE
   :addresses (sequence http://david-pc:7778/acc ))
a1: (GetRoutingInfo) Searching services...
SMA: Searching service getRoutingInfo (max hops = 2) ...
SMA: Find answer received. Redirect to
   ( agent-identifier :name a1@david-pc:1099/JADE
   :addresses (sequence http://david-pc:7778/acc ))
SMA: Find answer received.
   Redirect to ( agent-identifier :name a1@david-pc:1099/JADE
   :addresses (sequence http://david-pc:7778/acc ))
SMA: Find answer received.
   Redirect to ( agent-identifier :name a1@david-pc:1099/JADE
   :addresses (sequence http://david-pc:7778/acc ))
a1: (GetRoutingInfo) Sending use service answer...
MA: resposta del servei getRoutingInfo (GetRoutingInfo)
Host: ( agent-identifier :name MA@david-laptop:1099/JADE
   :addresses (sequence http://david-laptop:7778/acc ))
        TTR: 50
        ECC: false
```

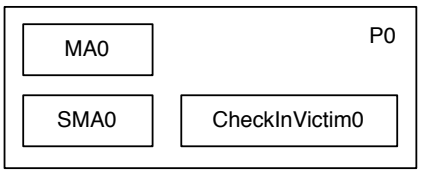

Figura 6.3: Esquema del segon escenari

```
        Timestamp: Thu Jun 25 09:04:09 CEST 2009
Host: ( agent-identifier :name MA@virtual1:1099/JADE
   :addresses (sequence http://virtual1:7778/acc ))
        TTR: 35
        ECC: true
        Timestamp: Thu Jun 25 08:59:43 CEST 2009
Host: ( agent-identifier :name MA@virtual2:1099/JADE
   :addresses (sequence http://virtual2:7778/acc ))
        TTR: 10
        ECC: false
        Timestamp: Wed May 20 23:47:24 CEST 2009
Host: ( agent-identifier :name MA@david-pc:1099/JADE
   :addresses (sequence http://david-pc:7778/acc ))
        TTR: 15
        ECC: false
        Timestamp: Thu Jun 25 09:05:40 CEST 2009
```

Com podem veure, la utilització del servei implica una sèrie de missatges de cerca i resposta. Observem com s'aprofiten aquests missatges de cerca per obtenir informació del servei. Finalment l'agent `MA` ha obtingut la informació d'encaminament dels seus veïns i pot utilitzar-la per a prendre decisions.

6.2.2 Segon escenari: `CheckInVictim`

El segon escenari es limita a la utilització del servei local `CheckInVictim` per a emmagatzemar la informació de triatge d'una determinada víctima. La Figura 6.3 mostra un esquema d'aquest escenari.

La sortida obtinguda durant l'execució de l'escenari és la següent:

```
SMA: L'agent SMA "SMA" s'ha creat correctament
SMA: Servei registrat: "checkInVictim" de l'agent "a1@david-pc:1099/JADE"
a1: L'agent CheckInVictimSA "a1" s'ha creat correctament
```

```
MAETT: Utilització del servei checkInVictim (CheckInVictim)
MAETT: Resposta del servei checkInVictim (CheckInVictim)
MAETT: Finalitzant execució
SMA: Utilització del servei checkInVictim reenviada a
   ( agent-identifier :name a1@david-pc:1099/JADE
   :addresses (sequence http://david-pc:7778/acc ))
a1: (CheckInVictim) Victim info received
        Triage Tag: Green
        RFID: 1
```

6.3 Test d'integració

Per a avaluar el grau d'integració de la nova arquitectura s'han realitzat un seguit de proves amb un altre projecte del mateix departament. El projecte [Jim09] ha estudiat les xarxes MANET i ha desenvolupat un *plugin* per a OLSR (*Optimized Link State Routing*) que genera el fitxer `/tmp/hosts` amb les direccions IP dels veïns i la qualitat del senyal.

Hem desenvolupat una extensió de l'agent SMA que s'anomena `MANETAddon` que s'encarrega de llegir aquest fitxer i generar una llista de veïns. En el moment de realitzar una cerca, l'agent SMA és capaç de recórrer aquesta llista i comunicar-se amb els SMA remots.

Les proves que s'han realitzat han implicat tres màquines connectades a la xarxa MANET i s'ha recreat un escenari molt semblant al de l'Apartat 6.2.1. El resultat ha estat satisfactori i l'arquitectura de serveis és compatible amb una xarxa MANET.

Capítol 7

Conclusions

En aquest darrer capítol farem una valoració de la feina feta al llarg del projecte començant per la revisió dels objectius inicials. A continuació es farà una discussió sobre les limitacions i l'ús de l'arquitectura i acabarem comentant la continuïtat del projecte.

7.1 Assoliment dels objectius

L'objectiu principal del projecte era dissenyar i implementar una arquitectura de serveis utilitzant la tecnologia d'agents intel·ligents. Aquesta arquitectura havia de complir una sèrie de requisits, entre ells, tenir una interfície abstracta i usable per l'aplicació. Aquest punt ha estat respectat durant el desenvolupament del projecte i s'ha separat molt bé cada funcionalitat de l'arquitectura fent-la independent dels serveis i aplicacions que la facin servir.

Abans de començar a analitzar el problema ens havíem de situar en el context d'aquest, analitzant el sistema existent i pensant en la forma d'integrar el projecte en aquest. Gràcies a la documentació facilitada referent al projecte [TSI2006-03481] l'estudi del sistema existent no va resultar complicat i l'objectiu es va poder assolir satisfactòriament.

L'arquitectura s'ha implementat seguint el disseny realitzat utilitzant Java com a llenguatge de programació i JADE com a plataforma *software*. Aquest objectiu

s'ha assolit amb la seva totalitat i l'arquitectura es pot fer servir en qualsevol plataforma.

El disseny de serveis mèdics i no mèdics a l'escenari de les emergències ha estat realitzat pensant en les necessitats de tots els actors presents. S'han dissenyat una sèrie de serveis primaris i secundaris, fent aquesta distinció en funció de la seva importància. Per facilitar la implementació dels serveis s'ha desenvolupat una interfície necessària per aquells serveis que vulguin utilitzar l'arquitectura. Per a validar el correcte funcionament de l'arquitectura també s'han implementat alguns dels serveis utilitzant la interfície esmentada anteriorment.

Finalment, calia integrar aquesta arquitectura amb el sistema existent. L'arquitectura s'ha desenvolupat seguint els requisits no funcionals imposats per l'aplicació existent. A més a més, els serveis i extensions implementades s'utilitzen en altres projectes, com per exemple el projecte [Jim09].

En resum, els objectius que ens havíem proposat al principi del projecte s'han pogut assolir satisfactòriament tenint en compte la planificació de les tasques amb les quals s'havia desglossat.

7.2 Discussió

Tot i els avantatges que ens proporciona la nova arquitectura de serveis, aquesta presenta algunes limitacions:

- La utilització de qualsevol servei requereix la transmissió d'una sèrie de missatges entre els diferents agents que participen. L'arquitectura ha d'estar connectada a una xarxa capaç de suportar el tràfic generat. Actualment totes les xarxes que existeixen poden suportar aquest volum de tràfic i podem utilitzar l'arquitectura fins i tot en un entorn *Bluetooth*.

- S'ha dissenyat una arquitectura de serveis en un entorn basat en agents intel·ligents. L'arquitectura no permet utilitzar serveis que no siguin proporcionats per agents. Aquesta limitació no és greu ja que podem incorporar un SA (*S*ervice Agent) que ens serveixi com a interfície entre un altra tecnologia de serveis i la nostra arquitectura.

L'arquitectura pot ser utilitzada per qualsevol aplicació que necessiti utilitzar serveis. Els agents que vulguin utilitzar-la hauran de conèixer la ontologia general de serveis que es proporciona amb l'arquitectura i hauran de crear la seva pròpia ontologia per a cada servei.

Els productors de serveis hauran de registrar-se a l'arquitectura enviant un missatge de registre a l'agent SMA i els consumidors hauran d'enviar missatges a l'agent SMA segons vulguin cercar o utilitzar un determinat servei.

Al principi només disposàvem d'un servei de pàgines grogues que ens permetia identificar quins agents proporcionàvem serveis. Hem dissenyat i implementat una arquitectura que ens permet estandarditzar com s'utilitzen els serveis i també ens permet la seva gestió.

En l'entorn de les emergències és molt important mantenir tots aquests serveis organitzats i permetre a les aplicacions crear nous serveis que es puguin afegir en aquesta arquitectura. Creiem que el projecte que hem desenvolupat és un bon punt de partida per crear nous sistemes basats en serveis.

7.3 Línies de futur

Tot i que l'arquitectura que hem desenvolupat funciona correctament, alguns aspectes dissenyats no s'han acabat d'implementar i alguns aspectes es podrien millorar i ampliar. Tot seguit es fa un recull d'aquestes línies de continuïtat:

Tolerància a fallades En les operacions que impliquin la gestió i utilització de serveis poden aparèixer errors a causa de diversos factors. La tolerància a fallades és molt bàsica i es podria ampliar per a prendre diferents decisions segons l'error. Es podria implementar una extensió de l'arquitectura amb aquesta finalitat.

Implementació de serveis S'ha dedicat una gran part del temps del projecte a l'etapa de disseny i ens hem centrat molt en crear una interfície per al desenvolupament de serveis. S'han implementat una sèrie de serveis mínims i es podria implementar la resta de serveis seguint els mateixos passos.

Jerarquia de serveis Alguns serveis poden requerir la participació de tercers. L'arquitectura podria tractar amb jerarquies de serveis per tal de poder oferir un conjunt de serveis a un nivell d'abstracció superior.

Desenvolupament d'extensions L'arquitectura està preparada per a ser ampliada utilitzant extensions que es poden carregar en arrencar l'agent SMA. Per exemple, es podria implementar una extensió per a utilitzar una memòria *cache* per accelerar la utilització de serveis.

Simulació conjunta L'arquitectura de serveis formarà part d'un projecte més gran que servirà per donar suport en situacions d'emergència. S'haurien de fer algunes proves en un entorn més real, utilitzant dispositius mòbils i agents reals.

Bibliografia

[TSI2006-03481] Projecte d'investigació: "Disseny i implementació d'una solució global per al suport de serveis crítics en situacions d'emergència", novembre 2008.

[uml05] UML Unified Modeling Language, gener 2007. <http://www.uml.org/>

[mas] Sistemes Multi-Agent. Wikipedia, juny 2009. <http://en.wikipedia.org/wiki/Multi-agent_system>

[jade] Pàgina principal de JADE, novembre 2008 . <http://jade.tilab.com/index.html>

[fipa] Pàgina principal de IEEE FIPA, gener 2009 . <http://www.fipa.org/>

[Jim09] Jiménez Díaz, Raúl. Seguretat en xarxes i aplicacions distribuides - MANET *Routing*: *Projecte Final de Carrera*. Bellaterra: Universitat Autònoma de Barcelona, 2009.

[Han08] Mark D. Hansen. SOA Using Java Web Services, maig 2008.

Firmat: David Morcillo Muñoz

Bellaterra, Juny de 2009

Resum

Aquest projecte consisteix en el disseny i desenvolupament d'una arquitectura de serveis sota el paradigma dels agents intel·ligents. El propòsit de ADASMI (*Architecture for Dynamic Agent Service Management and Interaction*) és permetre la gestió i utilització de serveis per altres agents. L'arquitectura s'ha implementat utilitzant la plataforma d'agents de JADE i es pot utilitzar amb qualsevol altra plataforma que compleixi els estàndards de IEEE FIPA. A més, és prou flexible com per adaptar-se en entorns dinàmics, com per exemple les xarxes ad-hoc en situacions d'emergència.

Resumen

Este proyecto consiste en el diseño i desarrollo de una arquitectura de servicios bajo el paradigma de los agentes inteligentes. El propósito d'ADASMI (*Architecture for Dynamic Agent Service Management and Interaction*) es permitir la gestión i utilización de servicios por otros agentes. La arquitectura se ha implementado utilizando la plataforma de agentes de JADE i se puede utilizar con cualquier otra plataforma que cumpla los estándares de IEEE FIPA. Además, es bastante flexible como para adaptarse en entornos dinámicos, como por ejemplo las redes ad-hoc en situaciones de emergencia.

Abstract

This project shows the design and development of an agent based services architecture. ADASMI (*Architecture for Dynamic Agent Service Management and Interaction*) aims at allowing agents the use and management of services. The architecture is implemented using JADE and it is IEEE FIPA-compliant. Moreover, it is very flexible and adapts well to dynamic environments, for example ad-hoc networks in emergency situations.