
PROJECTE FINAL DE CARRERA



UNIVERSITAT
ROVIRA I VIRGILI



ESCOLA
TÈCNICA
SUPERIOR
D'ENGINYERIA

SMA d'ajuda a la gestió dels trasplants d'òrgans entre hospitals

Logística del transport dels òrgans

Alberto Ribes García, arg.is@alumne.etse.urv.es

Professors:

Antonio Moreno, amoreno@etse.urv.es

Aïda Valls, avalls@etse.urv.es

Escola Tècnica Superior d'Enginyeria
Universitat Rovira i Virgili (URV), <http://www.etse.urv.es>

Curs 1999-2000

1. DESCRIPCIÓ	3
1.1. Part A: Localització dels receptors més adients.....	3
1.2. Part B: Logística de les operacions de trasplant.	4
1.3. Part C: Logística del transport dels òrgans.	4
2. ESTRUCTURA DE LA DOCUMENTACIÓ.....	6
3. AGENTS I SISTEMES MULTIAGENT.....	7
3.1. Definicions.....	7
3.2. Atributs.....	8
3.3. Tipus d'agents.....	9
3.4. Administració d'agents FIPA.....	10
3.4.1. Directory Facilitator (DF).....	11
3.4.2. Agent Management System (AMS).....	11
3.4.3. Agent Communication Channel (ACC).....	12
3.4.4. Agent Platform (AP).....	12
3.5. El llenguatge de comunicació entre agents (ACL)	14
4. L'ENTORN JADE.....	18
4.1. La classe "Agent"	19
4.2. Els comportaments.....	20
4.3. Les eines de desenvolupament.....	21
4.3.1. RMA Agent	21
4.3.2. Dummy Agent	22
4.3.3. Sniffer	22
4.3.4. Socket Proxy Agent	23
5. PERQUÈ UN SMA.....	24
6. LA LOGÍSTICA DEL TRANSPORT DELS ÒRGANS	25
6.1. La classe Graf	26
6.2. L'agent coordinador de rutes.	28
6.3. Els agents "AgentTrens" i "AgentAvions"	31
6.4. Els agents que agentifiquen les BDs.....	32
6.5. Disseny de les BDs	34
6.6. Creació de les BDs en Access.....	35
6.7. Establiment de la connexió amb les BDs per mitjà de JDBC.	42
6.8. Registre de la base de dades al sistema.....	44
7. CODI FONT.....	45
7.1. La classe fill.....	45
7.2. La classe fitxer.....	45
7.3. La classe graf.	46
7.4. La classe moviment.	49
7.5. La classe AgentCoord.....	49
7.6. La classe AgentTrens.....	56
7.7. La classe AgentAvions.	61
7.8. La classe AgentRenfe.	65
7.9. La classe AgentRailRibes.	68
7.10. La classe AgentIberia.	71
7.11. La classe AgentHispanAir.	74
8. JOC DE PROVES	77
9. CONCLUSIONS I VALORACIÓ PERSONAL	90
10. BIBLIOGRAFIA	91

1. DESCRIPCIÓ

Aquest projecte forma part d'un projecte més gran que té com a objectiu principal ajudar a la gestió del trasplant d'òrgans entre hospitals, i que té tres parts:

- Part A: Localització dels receptors més adients.
- Part B: Logística de les operacions de trasplant.
- Part C: Logística del transport dels òrgans.

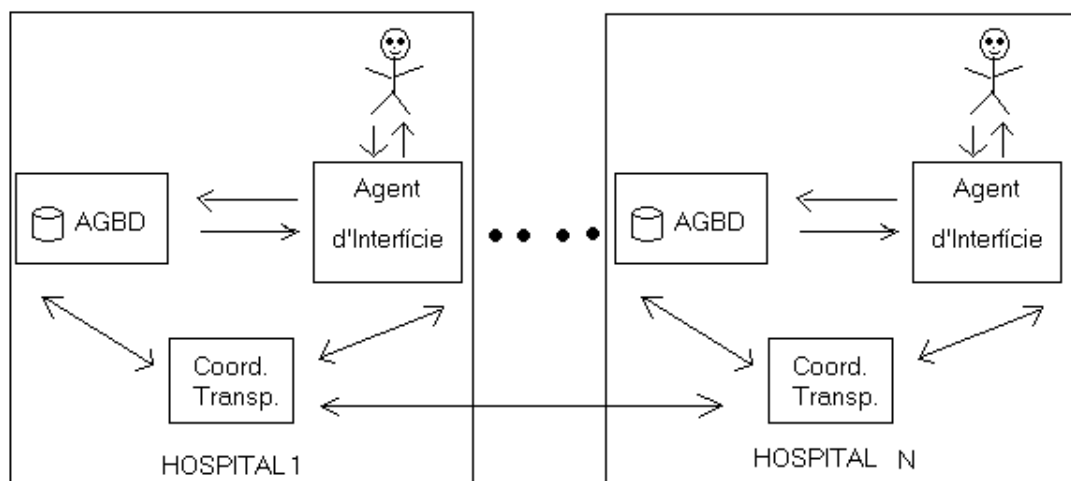
Molt sovint, quan es disposa d'un òrgan, és fonamental una ràpida organització per a que li pugui ser trasplantat l'òrgan al pacient més adient, com més aviat millor. Tasques com trobar el pacient més adient, organitzar l'equip mèdic per a un trasplantament o trobar la ruta i els medis de transport per traslladar l'òrgan a un altre hospital, actualment les fan un conjunt de persones utilitzant poca cosa més a part del telèfon. D'aquesta forma, un cop es disposa d'un òrgan, comença una cursa contra rellotge per realitzar les tasques anteriors, mantenint així ocupat a un conjunt de persones.

Amb la utilització de la informàtica i les noves tecnologies es poden automatitzar aquestes tasques per tal de realitzar-les molt més ràpida i més eficient del que es fa en la actualitat, a més de no suposar una ocupació afegida a cap persona dels hospitals.

A continuació es descriuen amb més detall cadascuna de les parts.

1.1. Part A: Localització dels receptors més adients.

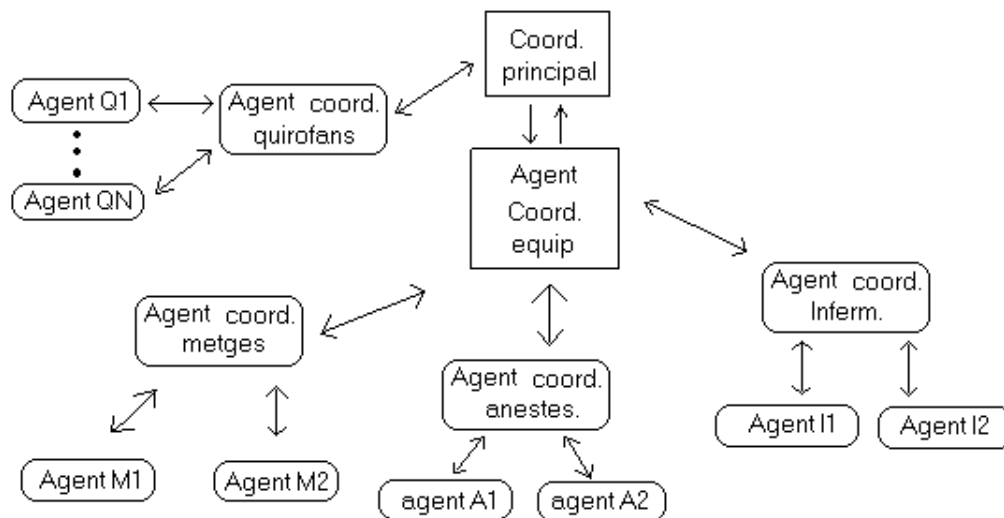
En el moment en que es disposa d'un òrgan en qualsevol hospital, se li ha de comunicar aquest fet a l'agent d'interfície. Llavors per mitjà de l'agent coordinador de trasplants i l'agent gestor de la base de dades, que disposarà de tota la informació sobre els pacients que esperen algun òrgan, s'esbrinarà si en el mateix hospital hi ha algun receptor que compleixi les condicions necessàries (grup sanguini, edat, estatura, etc.). Si no hi ha un receptor adient llavors es posarà en contacte amb els seus homòlegs als altres hospitals que duren a terme la mateixa tasca, com si l'òrgan es trobés al seu hospital, consultant les seves bases de dades.



Un cop esbrinats els possibles receptors enviaran aquesta informació a l'agent coordinador que l'ha demanada. Aquest estudiarà els receptors i prendrà una decisió per seleccionar-ne un grup reduït, tenint en compte el temps d'espera dels pacients, l'adequació de l'òrgan a les característiques físiques del pacient, etc.

1.2. Part B: Logística de les operacions de trasplant.

Un cop determinat el conjunt de possibles receptors, es comprovarà si és possible l'operació en un hospital determinat segons la disponibilitat dels quiròfans, infermeres, metges especialistes, etc. Aquesta part l'anomenem "Logística de l'operació de trasplant".



L'agent principal rep la petició d'una operació, determinada per una hora d'arribada de l'òrgan, i un màxim de temps per fer l'operació. Aquest agent es posarà en contacte amb l'agent coordinador de l'equip mèdic i amb l'agent coordinador de quiròfans.

L'agent dels quiròfans coneixerà la disponibilitat dels mateixos, i buscarà el forat més adient, tenint en compte la durada de l'operació.

L'agent coordinador de l'equip mèdic coneixerà quin equip es necessita per cada tipus d'operació, i es posarà en contacte amb els agents coordinadors dels diferents tipus de metges.

Aquesta part l'ha realitzat Jaime Bocio [1], presentat el 22/9/2000.

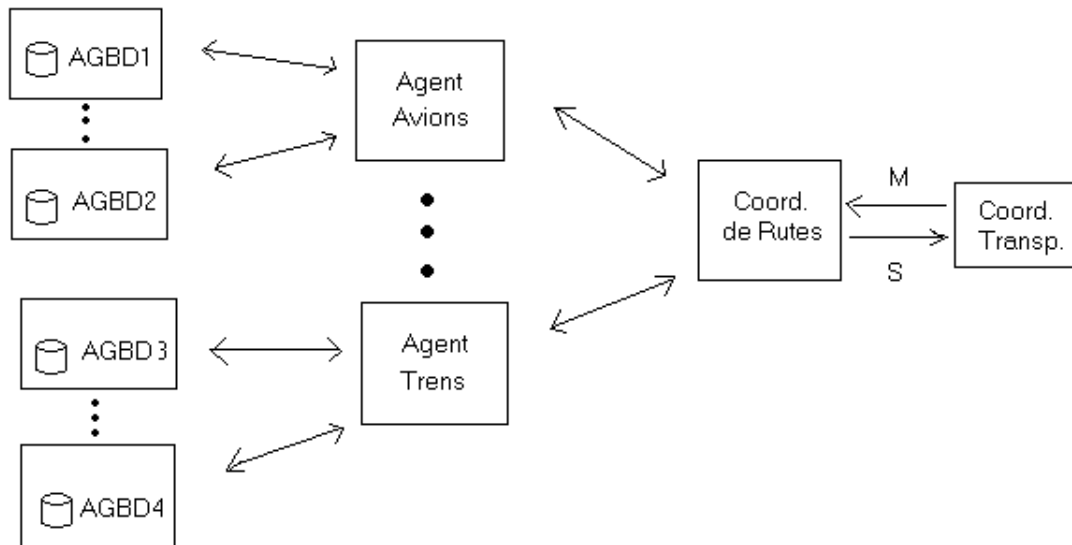
1.3. Part C: Logística del transport dels òrgans.

L'agent coordinador del trasplant dels òrgans quan necessiti saber si es pot transportar un òrgan a un altre hospital li ho demanarà a l'agent planificador de rutes. Aquest, per mitjà d'un graf que defineix les connexions entre les ciutats, trobarà un conjunt de rutes possibles.

Cada ruta pot tenir un o més trams, els trams són les connexions directes entre dues ciutats. Les rutes s'aniran comprovant tram a tram, gràcies a l'ajuda dels agents

encarregats de cada medi de transport. Aquestos demanaran la informació als agents que agentifiquen bases de dades de diferents companyies de transport.

Un cop realitzada la comprovació, les rutes acceptades s'enviaran com a resposta a l'agent de trasplants amb informació sobre els horaris dels medis de transport que pot agafar per dur a terme el transport dels òrgans.



Tindrem un agent per cada medi de transport diferent, aquestos ajudaran a l'agent coordinador de rutes demanant la informació als agents de les bases de dades de les empreses de cada medi de transport. Aquestos agents guardaran la informació obtinguda de les consultes a les bases de dades i en rebre una petició de l'agent coordinador de rutes comprovarà si la petició havia estat feta anteriorment, en aquest cas ja disposarà de la informació i la retornarà. Així ens estalviem la repetició de consultes. Per suposat aquestos agents no poden guardar tota la informació obtinguda dels agents de les bases de dades, per tant, arribarà un moment que hauran de desfer-se de dades antigues.

Els agent que agentifiquen les bases de dades s'encarregaran de realitzar la connexió amb les diferents bases de dades, realitzar la consulta, convertir els tipus de les dades a tipus de dades adients, i enviar com a resposta la informació obtinguda.

2. ESTRUCTURA DE LA DOCUMENTACIÓ

A continuació es proporciona una breu sinopsi de la resta dels apartats d'aquesta documentació.

Agents i sistemes multiagent

Aquest sistema d'ajuda al trasplant dels òrgans s'ha implementat usant Sistemes Multiagent (SMA¹). A l'apartat 3 s'introdueixen els conceptes bàsics d'aquesta metodologia.

L'entorn JADE

L'apartat 4 està dedicat a l'eina de programació i desenvolupament de sistemes multiagent utilitzada, anomenada JADE.

Perquè un SMA

A l'apartat 5 s'argumenta la necessitat d'usar un SMA, s'expliquen breument els avantatges dels sistemes multiagent.

Logística del transport dels òrgans

A l'apartat 6 s'entra en detall amb el treball realitzat en la logística del transport d'òrgans.

Codi font

Com el seu nom indica, a l'apartat 7 trobem tot el codi font comentat dels agents implementats.

Joc de proves

A l'apartat 8 es disposa de totes les dades fetes servir per realitzar el joc de proves, i les proves fetes per comprovar el correcte funcionament del treball.

Conclusions i valoració personal

Finalment, a l'apartat 9 hi ha les conclusions i valoració personal.

¹ Farem servir la sigla SMA fent referència a Sistema Multiagent

3. AGENTS I SISTEMES MULTIAGENT

Vista la diferència existent entre les diferents definicions del concepte agent trobades, la seva definició no sembla fàcil. Començem per la que dona la *Foundation for Intelligent Physical Agents* (FIPA²)[2]:

Un agent és l'actor fonamental en un domini. Combina un o més capacitats de servei en un model d'execució integrat, que pot incloure l'accés a *software* extern, usuaris humans i facilitats de comunicació.

3.1. Definicions

Altres definicions trobades al projecte de **David Isern Alarcón** [3] són:

Nicholas Negroponte i Alan Kay [4], diuen:

Un agent pot ser considerat un programa assistent personal (*personal software assistant*) amb autoritat delegada per l'usuari, per fer tasques que a l'usuari li són difícils o impossibles de realitzar. Aquests programes simulen una relació humana per fer alguna cosa que cap altra persona podria fer per tu.

Per Hyacinth S. Nwana de BT i, Michael Wooldridge de la Universista de Liverpool[5], un agent és un component de software i/o hardware que és capaç d'actuar per complir tasques en representació d'un usuari. Quan ha de realitzar alguna tasca, cal donar-li aquesta informació en expressions, meta-expressions, o classes. Aquesta informació cobreix diferents tipus d'agents i cal triar de quin tipus es tracta.

Segons Pattie Maes de l'MIT [6],

Un agent és un sistema computacional que habita en un entorn complex i dinàmic. L'agent pot sentir i actuar autònomament en el seu entorn, i té un conjunt d'objectius o motivacions que ell intenta d'aconseguir a través de les seves accions.

Per comprendre el que és un agent, les seves capacitats i com interacciona amb altres agents per forma un sistema multiagent (SMA), el millor és definir els seus atributs, els tipus d'agents que hi ha i fer un cop d'ull a les especificacions donades per la FIPA, com farem en les següents seccions.

² La FIPA és una associació sense ànim de lucre registrada a Ginebra, Suïza. El seu propòsit és promoure l'èxit de les aplicacions, serveis i equips emergents basats en agents. Aquests objectius són perseguits fent disponibles de forma ràpida les especificacions acceptades internacionalment que maximitzin la interoperabilitat entre aplicacions, serveis i equips basats en agents.

3.2. Atributs

Les propietats més importants que caracteritzen a un agent són les següents:

A) *Autonomia.*

Els agents han d'operar sense la intervenció directa dels humans o altres, i han de tenir algun tipus de control sobre les pròpies accions i l'estat intern. Això requereix aspectes com accions periòdiques, execucions espontànies i iniciativa, i que cada agent ha de poder prendre accions preventives o independents que beneficiaran l'usuari.

B) *Sociabilitat/cooperació.*

Els agents interactuen amb altres agents i/o humans a través d'algun tipus de llenguatge de comunicació. La interacció entre l'usuari i l'agent pot ser descrita com un contracte de col·laboració on l'usuari dóna unes especificacions que hauran de ser portades a terme per l'agent. Per exemple, una comanda d'un determinat producte a un agent, el qual controla la fabricació d'aquest producte. La interacció entre agents és un mecanisme pel qual els agents intercanvien el seu coneixement, les seves creences i els seus plans per treballar junts i solucionar grans problemes que es troben més enllà de les seves possibilitats.

C) *Reactivitat.*

Els agents perceben el seu entorn i responen en un temps raonable als canvis detectats. Les accions es realitzen quan es compleixen les precondicions de les regles, que donen lloc a l'execució d'aquestes, o per l'execució de plans ja predissenyats.

D) *Pro-activitat o capacitat de prendre iniciativa.*

Els agents no només han de respondre al seu entorn, sinó que han d'exhibir algun tipus de comportament encaminat a assolir algun tipus d'objectiu prenent la iniciativa.

E) *Mobilitat.*

Els agents poden moure's a altres entorns a través d'una xarxa electrònica. Poden traslladar dades amb instruccions que poden ser executades remotament.

F) *Continuïtat temporal*

Els agents estan contínuament executant processos, no són només un procés que s'acaba.

G) *Veracitat*

És l'assumpció que un agent no comunicarà informació falsa premeditadament.

H) *Benevolència.*

És l'assumpció que els agents no tenen objectius conflictius, i que cada agent sempre intentarà fer allò pel qual és requerit.

I) *Racionalitat.*

És l'assumpció que un agent actuarà per tal d'aconseguir el seu objectiu. No realitzarà una acció si les condicions per realitzar-la no es compleixen.

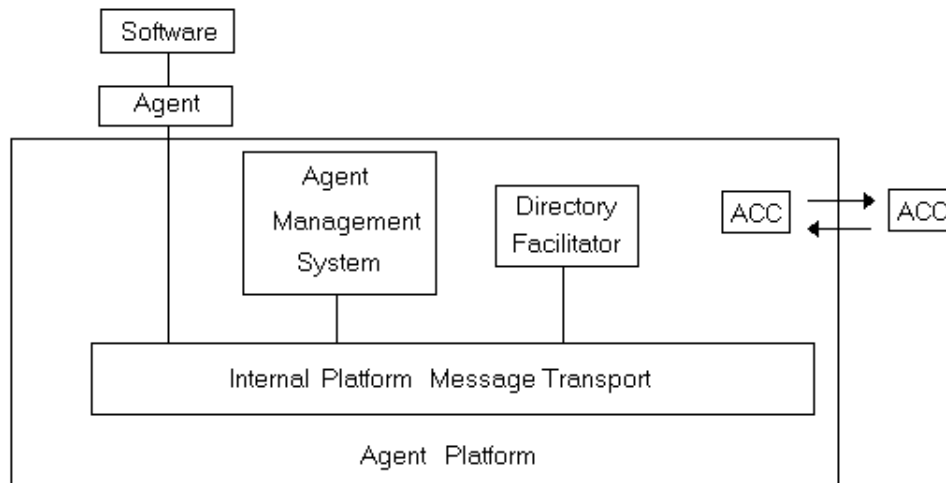
3.3. Tipus d'agents

- 1) **Agents d'Internet:** agents que tenen accés a les fonts d'informació (Internet) i poden posar en ordre i manipular informació obtinguda d'aquestes fonts, de manera que així poden respondre als requeriments imposats per l'usuari i/o altres agents. Molta gent es refereix als agents d'Internet com a agents que poden vagar per Internet per tal de recollir informació.
- 2) **Agents d'interfície:** sistema on els agents col·laboren amb l'usuari per a resoldre un problema. L'agent observa i monitoritza accions preses per l'usuari, aprenent d'aquestes, i suggerint millors maneres de dur a terme certes tasques.
- 3) **Agents de col·laboració:** aquests agents tenen com característiques principals la comunicació i la cooperació amb altres agents, per resoldre un problema en comú. Totes aquestes tasques que han de dur a terme els agents col·laboratius tenen com a base fonamental la comunicació. Aquest fet és molt important en el desenvolupament d'entorns multiagent perquè cal una metodologia que permeti que s'entenguin entre ells (llenguatge de comunicació –ACL (*Agent Communication Language*)- i un protocol de transmissió de dades (com per exemple TCP/IP).
- 4) **Agents mòbils:** aquests agents tenen com a principal característica la de poder moure's per una xarxa electrònica. Aquesta mobilitat es fa interactuant amb altres hosts, i permet als agents recollir informació, així com realitzar serveis requerits per l'usuari.
- 5) **Agents reactius:** els agents reactius tenen com a particularitat que no treballen amb models simbòlics de l'entorn. Enlloc d'això, actuen/responen a un estímul, segons un patró de l'estat actual en el qual es troben. El fet d'actuar segons un estímul seguint un patró els fa realment simples; també ho són les seves interaccions amb altres agents.
- 6) **Agents híbrids:** aquests tipus d'agents pretenen juntar diferents tipus d'agents per tal de reunir-los en un sol agent, de tal forma que es minimitzin els desavantatges de cadascun d'ells per separat. Hom pot pensar en un agent híbrid que estigui format per una part deliberativa i una part reactiva; així, els processos que hagin de ser ràpids i previstos a priori poden tractar-se amb la part reactiva, i les accions que comportin una planificació es realitzen a la part deliberativa.

3.4. Administració d'agents FIPA.

L'administració d'agents facilita una normativa de treball dins de la qual els agents existeixen i operen, aquesta normativa la podem trobar a la "Part 1: Agent Management" del document "FIPA 98 Specification"[\[2\]](#), que és d'on s'han extret les següents explicacions.

L'administració d'agents estableix el model lògic per a la creació, registre, comunicació, establiment, mobilitat i destrucció d'agents. La FIPA estableix el següent model per aquest entorn d'administració d'agents:



Components lògics que componen l'entorn de manipulació d'agents:

- **Agents:** com s'ha definit anteriorment, és l'element actor fonamental. Pot tenir un o més propietaris. Suporta moltes formes d'identificació; el més utilitzat és l'identificador únic global (GUID), també anomenat *agent-name*.
- **Agent Platform (AP):** proporciona la infraestructura física en la que els agents poden ser desplegats. La AP consisteix en una màquina(es), sistema operatiu, programes de suport a agents, els components d'administració d'agents FIPA (DF, AMS, ACC) i en la plataforma interna per al transport dels missatges (IPMT).
- **Directory Facilitator (DF):** proporciona un servei de "pàgines grogues" als agents. Els agents poden registrar els seus serveis al DF o demanar-li quins serveis són oferts per altres agents.
- **Agent Management System (AMS):** agent que exerceix un control de supervisió sobre l'accés i l'ús de la plataforma. Només un AMS existirà en una *Agent Platform (AP)*. L'AMS manté un directori de noms lògics d'agents, i la seva adreça de transport associada per a una AP. També ofereix un servei de "pàgines blanques" als altres agents.
- **Agent Communication Channel (ACC):** tots els agents tenen accés al menys a un ACC. L'ACC és el mètode de comunicació per defecte entre agents de diferents APs.

- **Internal Platform Message Transport (IPMT):** és el propietari del medi d'intercanvis de missatges dins d'un AP.

3.4.1. Directory Facilitator (DF)

El DF proporciona el servei de directori de “pàgines grogues” als agents. El DF és un agent obligatori a qui es confia la custòdia d'un directori d'agents. Ha de mantenir una acurada i completa llista d'agents, amb la informació actualitzada sobre tots els agents. Com a mínim hi ha d'haver un DF resident en cada AP, encara que pot haver més d'un DF al mateix AP.

El DF pot restringir l'accés a la informació del seu directori, i verificarà tots els permisos d'accés dels agents que intentin informar al DF sobre els canvis en el seu estat. El DF no controla el cicle de vida de cap agent.

Els agents poden registrar els seus serveis al DF o preguntar al DF quins serveis ofereixen altres agents. Els DFs poden registrar-se en altres DFs. De forma similar l'AMS i l'ACC poden registrar-se al DF.

El DF inclou un mecanisme de cerca que busca primer localment, i després, si és necessari, amplia la cerca a altres DFs. La cerca per defecte és una “depth first”. Per a propòsits específics es poden utilitzar les següents restriccions:

- En el nombre de respostes “:**df-search-resp-req**”.
- En el nombre de salts “:**df-search-depth**”.
- Amb un temps màxim “:**df-search-time-out**”.
- El protocol de cerca “:**df-search-algo**”.

Tots els DFs tenen un nom per defecte que és *df* afegit a la resta de l'agent-name. Les accions d'administració suportades pel DF són:

- register
- search
- deregister
- modify

3.4.2. Agent Management System (AMS)

L'AMS és un component obligatori de l'AP. Només existirà un en cada AP. L'AMS és responsable de la gestió de les operacions en un AP. Per tant, és responsable de la creació i destrucció d'agents, decidint si un agent pot registrar-se dinàmicament en l'AP i supervisant la migració d'agents des de i cap a altres APs.

L'AMS representa la autoritat administrativa d'un AP. Un AMS pot demanar a un agent que acabi la seva execució, i forçar-lo a terminar si la petició és ignorada.

L'AMS manté un índex de tots els agents que estan actualment residint en un AP. L'índex inclou un GUID i la seva adreça de transport associada per l'AP. Que un agent resideixi en una AP implica que l'agent ha estat registrat l'AMS de la plataforma.

Tots els AMS tenen un nom per defecte que és *ams* afegit a la resta de l'agent-name.

Les accions obligatòries d'administració suportades pel AMS són:

- register-agent
- deregister-agent
- modify-agent
- query-platform-profile
- authenticate
- search-agent

Altres operacions no obligatòries que pot suportar són:

- suspend agent
- terminate agent
- create agent
- resume agent execution
- invoke agent
- execute agent
- resource management

3.4.3. Agent Communication Channel (ACC)

L'ACC encamina els missatges entre agents dins una AP cap a agents residents en altres APs. Tots els agents FIPA tenen accés com a mínim a un ACC. Només els missatges adreçats a un agent poden ser enviats a un ACC.

L'ACC facilita l'enrutament de missatges entre agents de diferents APs. L'enrutament de missatges entre diferents plataformes requereix un acord en el protocol de transport, la codificació i l'esquema d'adreçament. En canvi, l'agent que es registra dinàmicament a una plataforma ja té un mètode disponible per l'intercanvi de missatges entre aquest agent i els agents que resideixen en la plataforma.

L'ACC és un agent que porta el control de la comunicació. Té una interconnexió amb el mecanisme IPMT que FIPA no defineix.

Les accions obligatòries d'administració suportades pel AMS són:

- forward

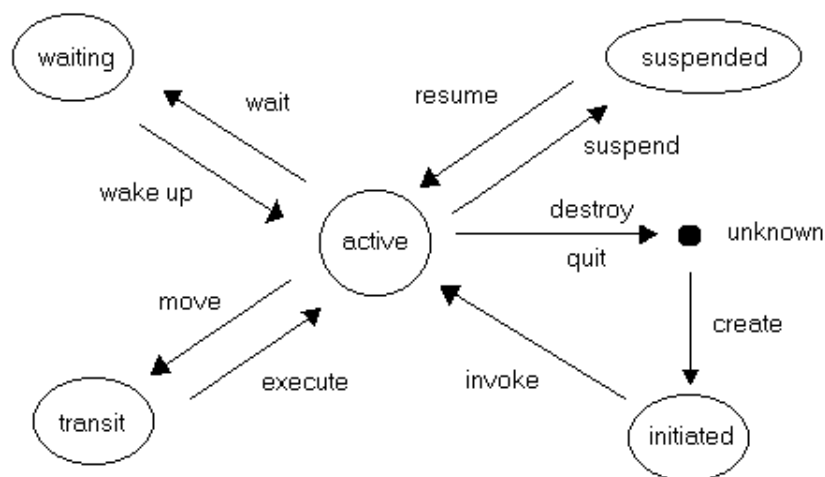
3.4.4. Agent Platform (AP)

Els agents FIPA existeixen físicament en una AP i utilitzen les facilitats que ofereix per realitzar les seves funcions. En aquest context, un agent, com un process de software físic, té un cicle de vida que ha de ser gestionat per l'AP.

El cicle de vida per a un agent FIPA té les següent característiques:

- 1) Circumscriu a una AP: un agent és gestionat dins d'una AP. El cicle de vida d'un agent estàtic és per tant circumscriu a una AP específica.
- 2) Independent de l'aplicació: el model de cicle de vida és independent de qualsevol aplicació de sistema. Aquesta defineix només els estats i les transicions del servei de l'agent en el seu cicle de vida.
- 3) Orientat a la instància: l'agent descrit al model del cicle de vida és assumit una instància (un agent amb un nom únic i executat independentment).
- 4) Singularitat: cada agent es troba només en un estat del cicle de vida en un instant de temps i només en una AP.

El cicle de vida d'un agent és representat per un conjunt d'estats i transicions:



Accions sobre els agents:

- Create -> creació d'un nou agent.
- Invoke -> invocació d'un nou agent.
- Destroy -> terminació forçada d'un agent. Aquesta només pot ser iniciada per l'AMS i no pot ser ignorada per l'agent.
- Quit -> la terminació normal d'un agent. Pot ser ignorada per l'agent.
- Suspend -> posa un agent en l'estat de suspens. Pot ser iniciada per un agent qualsevol o per l'AMS.
- Resume -> porta a l'agent des de l'estat de "suspended" al d'"active". Només pot ser iniciada per l'AMS.
- Wait -> posa l'agent en l'estat de "waiting". Aquesta només pot ser iniciada per l'agent.
- Wake Up -> porta a l'agent de l'estat de "Waiting" al d'"Active". Pot ser iniciada només per l'AMS.

Les accions següents només són usades per agents mòbils:

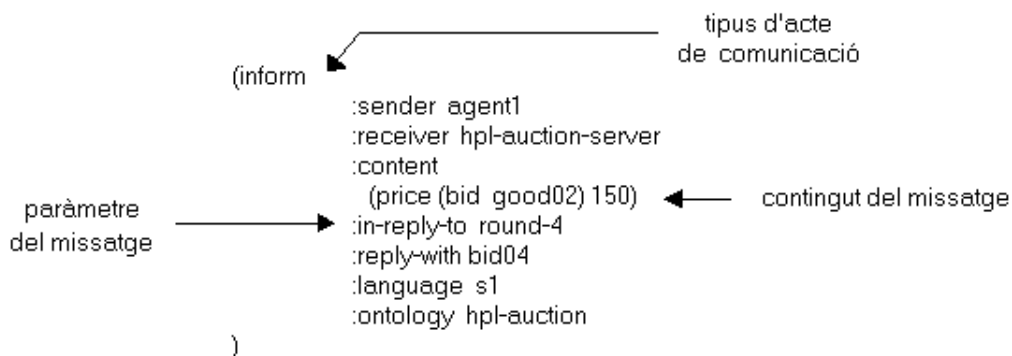
- Move -> posa l'agent en un estat transitori. Només pot ser iniciada per l'agent.
- Execute -> Porta l'agent des de l'estat transitori cap a l'estat "active". Només pot ser iniciada per l'AMS.

3.5. El llenguatge de comunicació entre agents (ACL)

Els agents individualment proporcionen una funcionalitat molt interessant, però el que els dona un valor afegit és la capacitat de cooperar per a resoldre problemes que sols potser no podrien solucionar o els hi seria molt difícil. A més, normalment el coneixement que té cada agent és limitat, pot ser incorrecte o incomplet, i per això cal que formi part d'un col·lectiu en el qual hi participa i del qual se'n beneficia.

Per tal de cooperar efectivament, els agents s'han de comunicar els uns amb els altres i han de treballar amb algun llenguatge de comunicació per agents (*Agent Communication Language (ACL)*).

Els ACL estan basats en la teoria de la parla, en la qual les comunicacions individuals entre agents poden reduir-se a un petit nombre d'idees o més generalment actes comunicatius. Aquests actes donen forma al significat bàsic de la comunicació.



Els missatges estan representats com expressions. El primer element del missatge és una paraula que identifica l'acte de comunicació, que defineix el principal significat del missatge. Després li segueix una sèrie de paràmetres, definits pel caràcter ':' seguit d'una paraula clau. Un dels paràmetres conté el contingut del missatge, codificat com una expressió en algun llenguatge. Altres paràmetres ajuden a entregar correctament el missatge ("sender" i "receiver"), ajuden a interpretar correctament el missatge ("language" i "ontology"), o ajuden al receptor a respondre cooperativament ("reply-with" i "reply-by").

Llista d'accions de comunicació:

La següent és una llista d'algunes de les accions més usuals:

- 1) **accept-proposal** -> acció d'acceptar una proposta feta anteriorment per realitzar una acció.
- 2) **agree** -> acció d'acceptació per realitzar alguna acció, possiblement en un futur.
- 3) **cancel** -> acció de cancel·lar alguna acció demanada anteriorment.
- 4) **cfp** -> crida per proposar la realització d'una acció.
- 5) **confirm** -> l'emissor informa al receptor que una proposició és certa, quan el receptor dubtava sobre la proposició.
- 6) **disconfirm** -> l'emissor informa al receptor d'una proposició és falsa, quan el receptor creia que era certa.
- 7) **inform** -> l'emissor informa al receptor que una proposició.
- 8) **not-understood** -> l'emissor informa al receptor que no ha entès una petició realitzada anteriorment pel receptor.
- 9) **propose** -> acció de proposar la realització d'una acció, donades certes precondicions.
- 10) **query-if** -> acció de preguntar a un altre agent si una acció és certa o no.
- 11) **refuse** -> acció de refusar la petició de realitzar una acció donada, explicant la raó de la negativa.
- 12) **reject-proposal** -> acció de rebutjar una proposta per a la realització d'una acció durant una negociació.
- 13) **request** -> l'emissor demana al receptor realitzar una acció.
- 14) **request-when** -> l'emissor vol que el receptor realitzi alguna acció quan alguna proposició esdevingui certa.

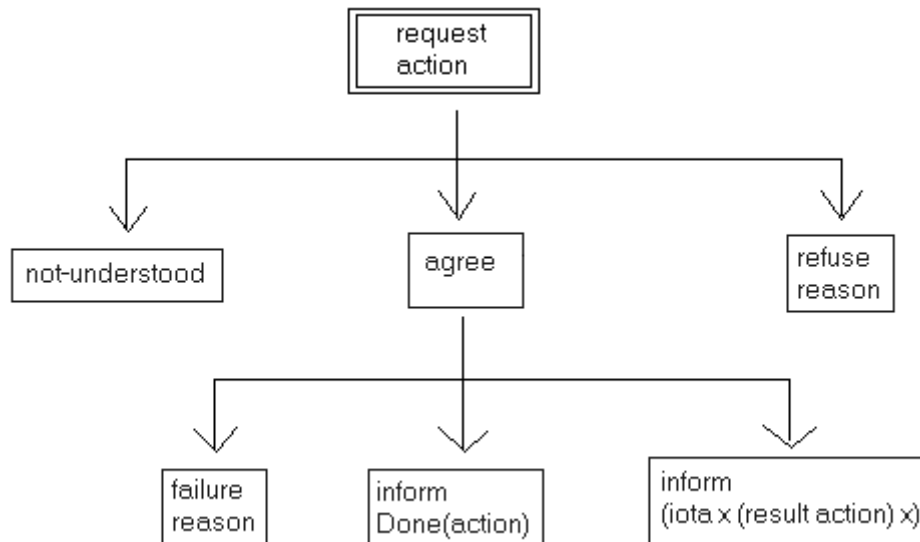
Llista de protocols d'interacció:

Les conversacions entre agents sovint segueixen models típics. En aquests casos, s'espera certa seqüència de missatges. Aquests models típics d'intercanvi de missatges són anomenats protocols. Aquests protocols ajuden a seguir correctament la conversació entre els agents.

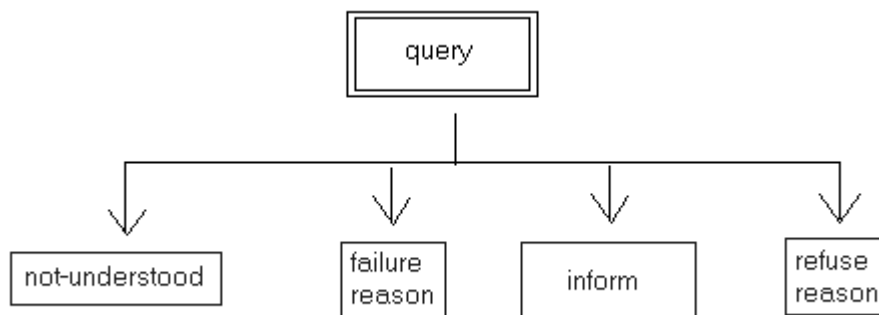
Notació utilitzada per descriure els protocols d'interacció:

- Les capsles amb doble requadre representen les accions realitzades per l'iniciador.
- Les altres capsles són accions realitzades pels altres participants al protocol.

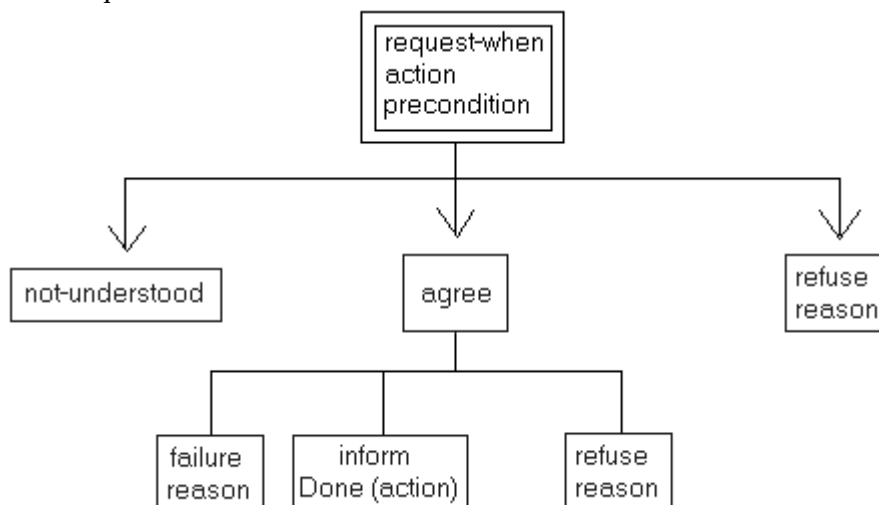
- 1) **fipa-request**: aquest protocol simplement permet a un agent demanar a un altre que realitzi alguna acció, i al receptor realitzar l'acció o respondre que no ha pogut.



- 2) **fipa-query**: al receptor li és demanat de fer alguna acció d'informació.

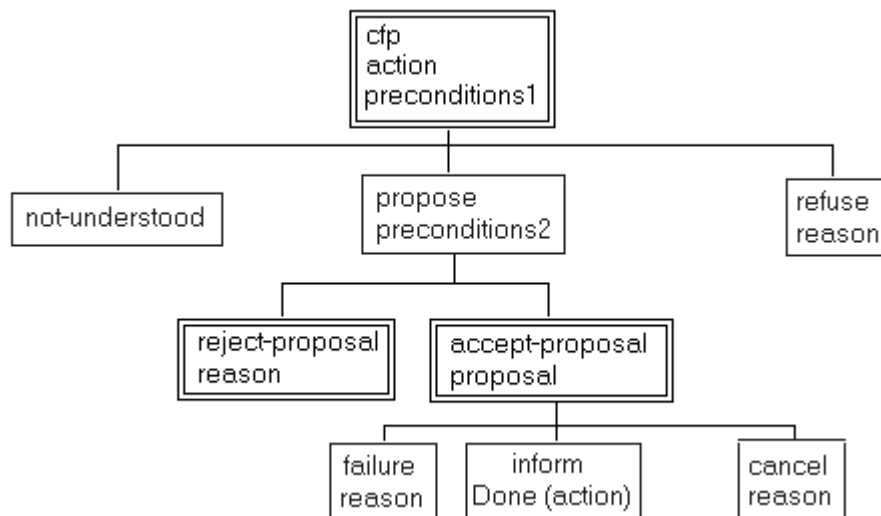


- 3) **fipa-request-when**: protocol que es segueix quan un agent utilitza l'acció request-when.

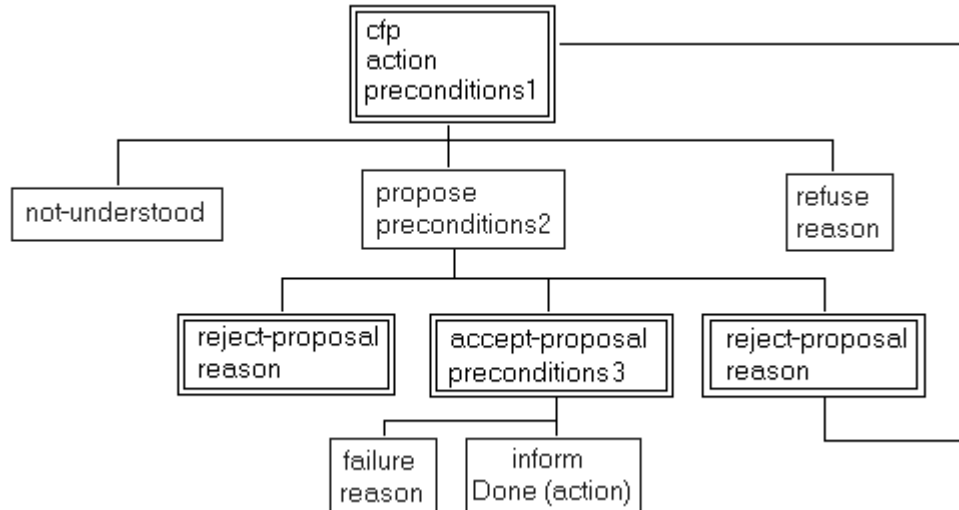


- 4) **fipa-contract-net**: en aquest protocol un agent és el director, que vol que alguna tasca sigui realitzada per un o més agents, i desitja optimitzar una

funció que caracteritza la tasca. Aquesta característica és normalment anomenada “preu”. El director sol·licita les propostes d’altres agents per realitzar la tasca. Cada agent envia la proposta amb el seu preu. Igual que en una subhasta, el director accepta la proposta que li sembli millor.



- 5) **fipa-iterated-contract-net**: extensió del protocol “contract net” anterior. A diferència de la versió anterior, en que aquesta es permet un cicle de propostes.



4. L'ENTORN JADE

JADE (Java Agent Development Environment) és un software que permet el desenvolupament d'aplicacions i sistemes multi-agent tenint en compte les especificacions donades per la FIPA. Inclou una plataforma que compleix l'estàndard FIPA, i un paquet per desenvolupar agents Java.

JADE es compon de quatre paquets principals:

- **jade.core** -> conté el nucli del sistema, que proporciona la classe "Agent". També disposa del subpaquet *jade.core.behaviour*, conté la jerarquia de la classe *Behaviour*. Els "Behaviours" són les unitats d'activitat lògica per als agents i poden compondre-se per aconseguir models complexos d'execució.
- **jade.lang** -> té un subpaquet per cada llenguatge utilitzat en JADE. En particular el subpaquet *jade.lang.acl* proporciona el llenguatge de comunicació entre agents (ACL) seguint les especificacions estàndards de la FIPA.
- **jade.domain** -> conté totes les classes per representar la plataforma d'agents FIPA i els models de domini, tals com entitats per a la administració d'agents, llenguatges i ontologies (AMS, DF, ACC, etc).
- **jade.proto** -> paquet que conté els protocols d'interacció del model FIPA.

Al paquet *jade.tools* trobem algunes eines que faciliten el desenvolupament d'aplicacions i l'administració de la plataforma, com per exemple:

- Un *Remote Management Agent* (RMA), que actua com una consola gràfica per a l'administració i control de la plataforma.
- Un *Dummy Agent* és una eina de monitorització i depuració, composta per una interfície gràfica i un agent JADE. Utilitzant la interfície és possible crear missatges ACL i enviar-los a altres agents, a més de poder visualitzar tots els missatges rebuts o enviats per aquest agent.
- Un Sniffer és un agent que pot interceptar missatges ACL i visualitzar-los gràficament utilitzant una notació similar als diagrames UML.
- Un *SocketProxyAgent* és un simple agent, que actua com un *gateway* bidireccional entre una plataforma JADE i una connexió TCP/IP. Els missatges ACL, viatjant per mitjà del servei de transport JADE, són convertits a strings ASCII i enviats al dispositiu de connexió. Aquest agent és d'utilitat per manipular firewalls o per proporcionar interaccions amb applets Java dins un navegador.

4.1. La classe “Agent”

Des del punt de vista del programador, un agent JADE és una instància d'una classe implementada per l'usuari, que estén la classe “Agent”. La nova classe heretarà de la classe “Agent” els mètodes necessaris per disposar dels avantatges dels agents JADE. Un agent té diferents comportaments (behaviours) que s'executaran concurrentment. Cada funció o servei que hagi de donar un agent serà implementat com un o més “behaviours”. Un *scheduler* intern controlarà automàticament l'execució de tots els comportaments.

Seguint les especificacions sobre el cicle de vida vist a la secció 2.4.4, els agents podran estar en un dels següents estats:

- **AP_INITIATED**: no s'ha registrat a l'AMS, no té encara nom ni adreça.
- **AP_ACTIVE**: s'ha registrat, té nom i adreça.
- **AP_SUSPENDED**: està parat, cap dels seus comportaments és executat.
- **AP_WAITING**: està bloquejat, esperant algun event, com per exemple l'arribada d'un missatge.
- **AP_DELETED**: està mort, s'ha esborrat el seu registre a l'AMS.

La classe agent disposa de mètodes per canviar l'estat d'un agent, com per exemple:

doWait() -> Bloqueja tots els behaviours d'un agent, i l'agent passa de l'estat “Active” a l'estat “wait”.

block() -> suspen un comportament. Aquest passa d'“active” a “wait”. Opcionalment se li pot passar com a paràmetre les mil·lèsimes de segon que volem que estigui bloquejat.

restart() -> passa l'agent de l'estat “wait” a l'estat “active”.

Per a la inicialització i la finalització dels agents s'han de sobrescriure els mètodes “setup()” i “takedown()” heretats de la classe “Agent”. Al setup s'ha d'afegir obligatòriament al menys un comportament amb el mètode “addBehaviour()”; opcionalment, també es pot modificar el registre de dades o definir la descripció de l'agent, els seus serveis i registrar l'agent.

Així, el primer que s'executa és el mètode setup, després el *scheduler* passa a executar els diferents comportaments segons una política de *round robin*.

La classe “AgentGroup” permet enviar un missatge a més d'un agent.

La classe “MessageTemplate” permet formar un model que s'utilitzarà per agafar de la cua de missatges un que compleixi amb les condicions que marca el model. En les condicions es poden utilitzar expressions booleanes. Aquesta classe permet així que un agent pugui estar en vàries conversacions i respondre ràpidament encara que hi hagin molts missatges a la cua.

4.2. Els comportaments

Un agent és un únic fil (thread) d'execució, i tots els seus serveis han de ser implementats com a objectes "Behaviour". Per tal d'executar concurrentment els diferents comportaments, cada agent disposa d'una cua de comportaments que és controlada per el scheduler intern seguint una política de round robin. Aquesta cua pot ser modificada en qualsevol moment per mitjà dels mètodes "addBehaviour(...)" i "removeBehaviour(...)".

Cada comportament té un mètode "action()" i un mètode "done()". Primer el scheduler executa el mètode "action" que realitzarà el servei del comportament, i després s'executa el mètode "done" que retorna un boolea indicant si s'ha acabat o no el comportament. Quan el mètode "done" retorna cert el comportament és esborrat de la cua, altrament és encuat de nou.

Al canviar l'execució entre els comportaments, com no hi ha cap pila per salvar l'estat, aquest s'ha de mantenir per mitjà de variables a cada comportament i a l'agent.

Per evitar esperes actives cada comportament pot bloquejar-se cridant al mètode "block()". Aquest mètode posa el comportament en una cua de comportaments bloquejats. Tan aviat com rebí un missatge el comportament serà encuat a la cua de comportaments preparats per executar-se.

S'han d'evitar els bucles infinits i les operacions llargues al mètode "action()", ja que mentre aquest s'executa no es pot executar cap altre comportament dins del mateix agent. Si una operació és massa llarga per ser executada en un pas, s'ha de dividir en vàries suboperacions, i en cada execució del comportament realitzar una de les suboperacions. Si el comportament és massa complex com per dividir-ho, JADE disposa de diferents classes de comportaments que s'executen segons una política determinada. Exemples:

SimpleBehaviour -> comportament atòmic, no pot ser interromput. S'ha de sobrescriure el seu mètode "done".

OneShotBehaviour -> és una subclasse de "SimpleBehaviour" però el seu mètode "done()" sempre retorna true, i així només s'executa una vegada.

CyclicBehaviour -> és una subclasse de "SimpleBehaviour" però el seu mètode "done()" sempre retorna false, i així s'executa infinits cops.

ComplexBehaviour -> permet compondre comportaments en forma d'arbre gràcies als mètodes "addSubBehaviour", "removeSubBehaviour", "preAction", "postAction" i "reset".

SequentialBehaviour -> és una subclasse de "ComplexBehaviour" que executa els seus fills seqüencialment i acaba quan tots els mètodes "done" retornen cert.

NonDeterministicBehaviour -> és una subclasse de "ComplexBehaviour", que executa els seus fills segons una seqüència no determinista.

ReceiverBehaviour / SenderBehaviour -> rep/envia un missatge.

4.3. Les eines de desenvolupament.

4.3.1. RMA Agent

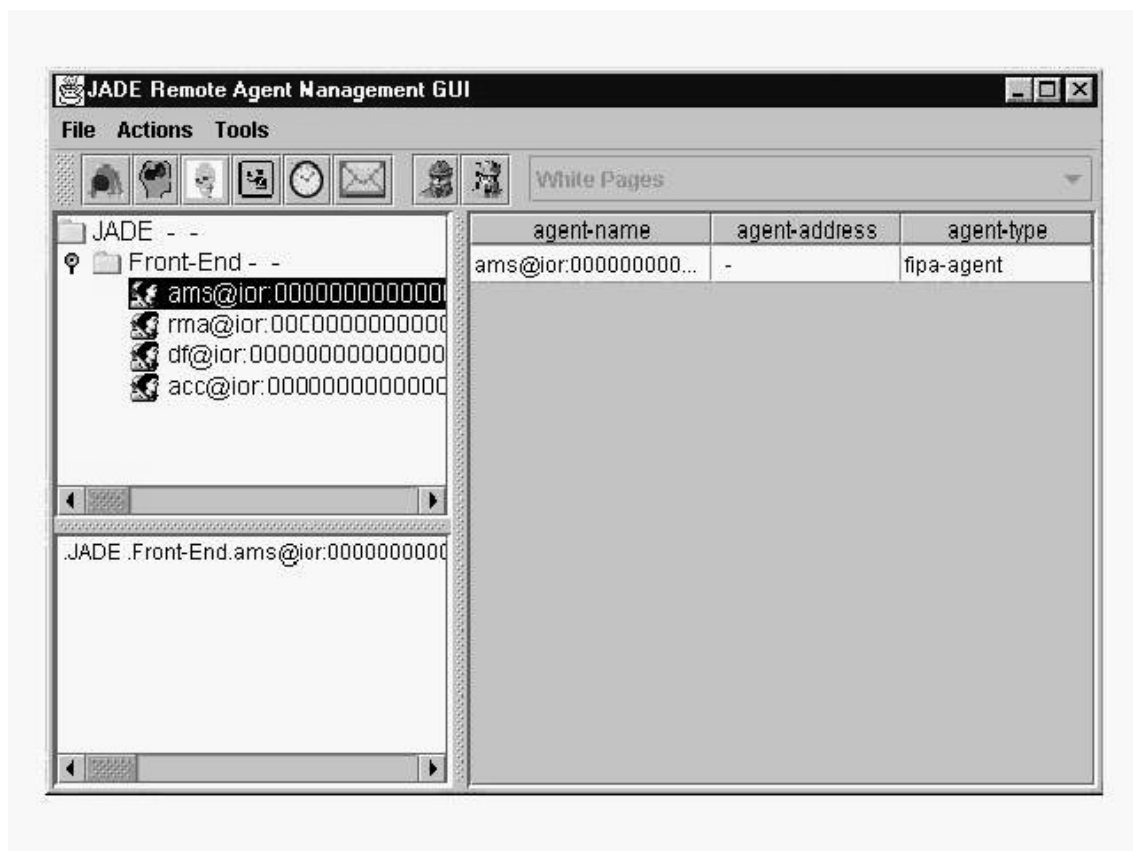
JADE ofereix una interfície gràfica per l'administració de la plataforma per mitjà de l'agent RMA, aquest agent mostra l'estat de la plataforma a la que pertany (agents i containers), i proporciona eines per demanar accions administratives des de l'agent AMS i per depurar i testejar aplicacions basades en JADE.

Un RMA és un objecte Java, instància de la classe *jade.tools.rma.rma* i pot iniciar-se de dues formes:

- 1) Des de la línia de comandes com un agent ordinari.
Exemple: `java jade.Boot myConsole:jade.tools.rma.rma`
- 2) Utilitzant el paràmetre `-gui` en la línia de comandes.
Exemple: `java jade.Boot -gui`

Més d'un agent RMA es pot iniciar a la mateixa plataforma, però només un per container.

A la següent figura podem veure la interfície gràfica de l'agent RMA.



Utilitzant el menú "Tools" podem accedir a les eines de desenvolupament "Dummy Agent" i "Sniffer Agent". Es pot enviar missatges a qualsevol agent de la plataforma, etc.

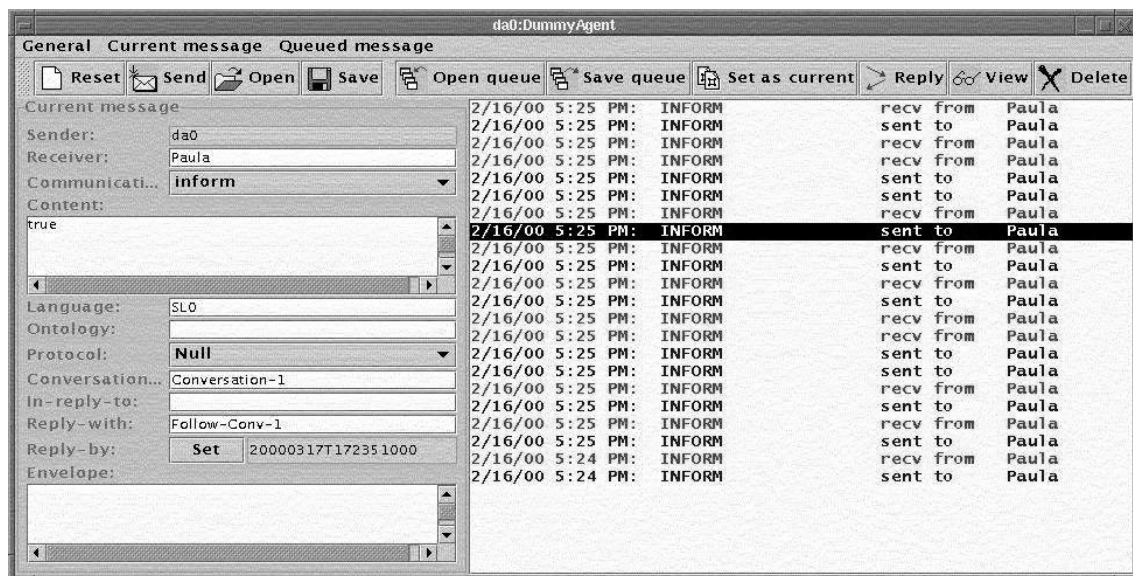
4.3.2. Dummy Agent

Aquest és un agent per inspeccionar l'intercanvi de missatges entre agents JADE.

Hi ha dues formes d'iniciar aquest agent:

- 1) Des de el menú "Tools" de l'interfície gràfica del RMA.
- 2) Amb la següent comanda:
java jade.Boot da:jade.tools.DummyAgent.DummyAgent

Aquest agent permet enviar missatges ACL als agents, rebre i inspeccionar missatges d'altres agents, llegir i salvar missatges des de/a un fitxer.



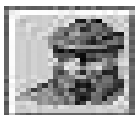
4.3.3. Sniffer

El Sniffer, com el seu nom indica, és una aplicació Java creada per restrejar l'intercanvi de missatges en un entorn basat en Jade. El Sniffer està completament integrat a l'entorn Jade i és particularment útil per la depuració dels comportaments.

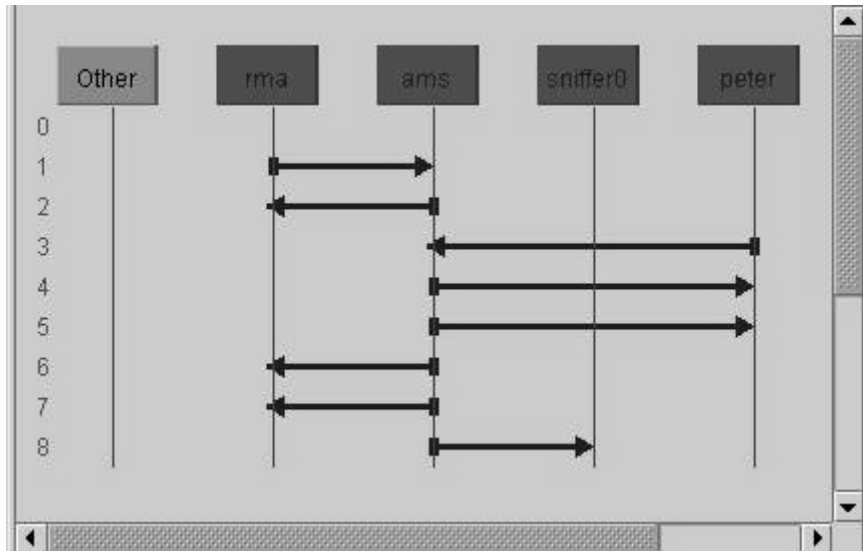
Quan l'usuari decideix rastrejar un agent o un grup d'agents, cada missatge dirigit a l'agent o enviat per ell és rastrejat i visualitzat en la interfície gràfica del Sniffer. L'usuari pot veure cada missatge, salvar-ho a disc com un fitxer de text o com un fitxer binari.

Pot inicialitzar-se de tres formes:

- 1) Primer es pot inicia amb la següent icona al RMA.



- 2) Des de el menú d'eines del RMA.
- 3) Creant una nova instància del Sniffer. Això és útil si es vol que el Sniffer tingui un nom diferent del estàndard. Per fer això s'ha d'utilitzar l'opció "Start New Agent" a la barra d'eines del RMA. Això fa que es crei una capça de diàleg on hem de posar el nom que volem donar-li a l'agent i el nom de la classe que serà *jade.tools.sniffer.sniffer*.



4.3.4. Socket Proxy Agent

Aquest és un agent JADE que pot ser executat per comunicar amb clients remots per mitjà d'un *socket*.

S'inicia amb la següent comanda:

```
java jade.Boot spa:jade.tools.SocketProxyAgent.SocketProxyAgent
```

Al inicialitzar-se, l'agent busca un fitxer que conté els paràmetres d'inicialització, aquest fitxer s'anomena **<agentname>.inf** i s'ha de ser posat al directori de treball actual. La primera línia del fitxer és el n° de port que l'agent escoltarà per connexions (normalment es posa 0 o 6789). La segona línia al fitxer conté la llista d'agents que desitgen que aquest agent proxy lis enviï missatges. L'agent accepta més de 50 connexions paral·leles. Cada missatge que arribi pel *socket*, és analitzat, el seu agent destinatari és comprovat amb la llista anomenada anteriorment, i finalment enviat al seu destinatari. Cada resposta que enviï l'agent destinatari és enviada a través del mateix *socket*.

5. PERQUÈ UN SMA

Els motius d'escollir la metodologia de Sistemes multiagent per a solucionar aquest problema es pot resumir en 4 punts:

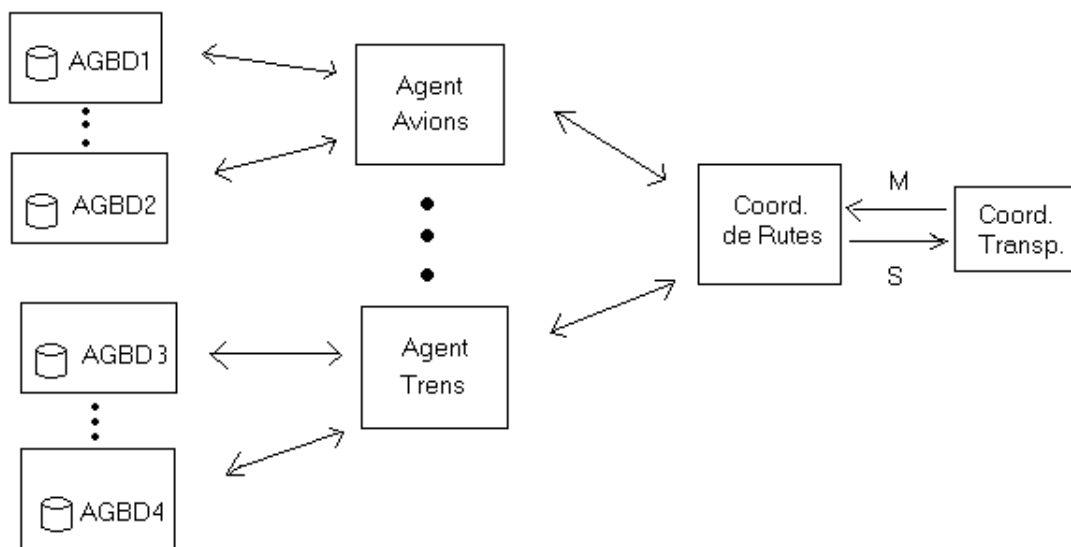
- 1) **Modularitat:** la divisió dels diferents serveis o funcions que es volen aconseguir en diferents agents, i dels agents en comportaments permet una programació més estructurada.
- 2) **Eficiència:** una dels principals avantatges dels sistemes multiagent és la programació distribuïda, aquesta permet distribuir les feines a fer entre diferents ordinadors i augmentar l'eficiència per mitjà del treball en paral·lel.
- 3) **Fiabilitat:** al ser un sistema distribuït, el fet de que algun dels ordinadors que componen el sistema deixi de funcionar no suposa que els serveis que proporciona un sistema multiagent deixin d'estar disponibles, ja que es pot disposar de diferents ordinadors que donin el mateix servei (redundància d'informació), formant així un sistema amb més fiabilitat.
- 4) **Flexibilitat:** la fàcil incorporació i eliminació dels agents al sistema, a més de la col·laboració entre ells, proporciona una gran flexibilitat.

6. LA LOGÍSTICA DEL TRANSPORT DELS ÒRGANS

Com ja s'ha explicat a la secció 1.2 l'objectiu d'aquest projecte és la implementació d'una aplicació basada en agents, que tracti de trobar les possibles rutes entre dues ciutats i comprovar per quins medis de transport es pot arribar al destí abans d'un temps màxim.

Per simplificar el problema farem les següents suposicions:

- Només treballarem amb els següents medis de transport: trens i avions.
- Només considerarem trasllats entre capitals de província d'Espanya.
- L'agent coordinador de rutes tindrà una estructura de dades del següent estil:
 - Multi-graf etiquetat no dirigit, on cada node representa una capital i els arcs representen connexions en tren o en avió entre les ciutats.
 - Cada arc estarà etiquetat amb el temps estimat de duració del viatge entre les dues ciutats.
 - Cada arc representa un viatge directe, sense passar ni parar en ciutats intermitjes.
 - És un multi-graf perquè pot haver dos arcs entre dos nodes, un per indicar el viatge en tren i l'altre per a l'avió (cadascun amb la seva duració).



Tipus d'agents que componen el SMA:

- Hi ha un agent coordinador de rutes únic.
- Cada medi de transport té un agent associat per coordinar les diferents alternatives (avió, tren, etc.).

- Hi ha un agent associat a cada base de dades on hi ha la informació de les companyies de transport que es poden utilitzar per realitzar el transport dels òrgans.

6.1. La classe Graf

Per tal de trobar les possibles rutes entre dues ciutats s'ha implementat la classe "Graf", aquesta contindrà la informació de les connexions entre les ciutats i disposa d'un mètode que realitza l'algorisme de cerca de les rutes.

La informació del graf es troba en el fitxer de text "ciutats.txt" guarda la informació de la següent forma:

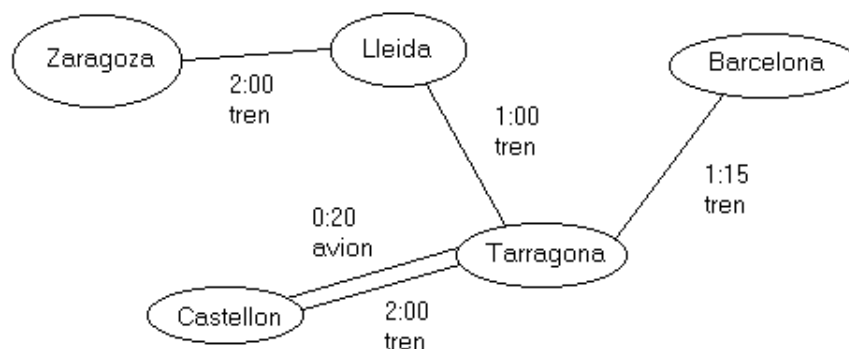
- Cada entrada del fitxer té el següent format:

Ciutat_origen - ciutat_destí hora:minut transport - - ciutat_desti hora:minut transport

- "Ciutat_origen" és el nom del node origen, la resta de l'entrada són els possibles destins als que es pot arribar des de aquesta ciutat.
- "ciutat_destí" és el nom del node destí.
- "hora:min" és el temps mig que es triga en fer el recorregut amb el medi de transport "transport".
- Cada ciutat pot tenir tants destins com sigui necessari, encara que només s'agafaran els destins més propers pels que hi ha alguna companyia de transport que faci el recorregut.

Exemple: les dues entrades següents representen el graf de sota.

TARRAGONA- BARCELONA 1:15 tren- LLEIDA 1:00 tren-CASTELLON 2:00 tren-
CASTELLON 0:20 avion
LLEIDA- TARRAGONA 1:00 tren- ZARAGOZA 2:00 tren



Com s'havia dit anteriorment és un graf no dirigit ja que per anar d'un node origen a un node destí s'agafa la mateixa aresta que per fer el camí invers. Com podem veure en les entrades del fitxer "ciutats.txt" anteriors es repeteix informació ja que aquesta es guarda com si es tractes d'un graf dirigit, per anar de Lleida a Tarragona hi ha informació en l'entrada de Lleida i per anar de Tarragona a Lleida tenim la mateixa

informació, així doncs s'ha d'anar amb compte en que hi hagi la mateixa informació en aquestos casos ja que sinó tindríem un graf dirigit. S'ha decidit aquesta representació del graf més que res per simplicitat, aquesta representació suposa repetició d'informació, però aquest fet fa que sigui més fàcil d'implementar l'algorisme de cerca.

Aquesta classe disposa del mètode “Inicialitza” que té per paràmetre el nom del fitxer amb el que s'ha d'inicialitzar el graf, llavors per mitjà de la classe “fitxer”, que disposa de mètodes per llegir strings i enters de fitxers de text, es van llegint les entrades del fitxer i es guarden en una taula hash que té com a clau el nom de la ciutat origen i com a valor la resta de l'entrada guardada en un vector d'objectes “fill”.

La classe “fill” guarda la informació d'un node destí i la seva aresta, al constructor se li passa el nom de la ciutat destí, el medi de transport, les hores i els minuts. Els atributs que té són els mateixos excepte les hores i minuts que es guarden en l'atribut “temps” que contindrà el nombre de mil·lèsimes de segon que es triga en fer el recorregut, aquest valor ens servirà en l'algorisme de cerca per tallar aquelles rutes que es passin del temps màxim.

L'algorisme de cerca:

Després d'inicialitzar la taula hash amb el contingut del fitxer “ciutats.txt” podem trobar les rutes més adients per mitjà del mètode “CercaCamins” que implementa l'algorisme de cerca, i té com a paràmetres: la ciutat origen, la ciutat destí i el temps màxim del que disposem en mil·lèsimes de segon.

Per realitzar la cerca de les millors rutes s'ha escollit l'algorisme de cerca en amplada, amb control de cicles. Intentarem trobar només unes poques solucions factibles, degut a que els primers camins solució seran els que tenen menys connexions aquestos seran els més ràpids i per tant no ens interessen gaire la resta de camins

L'algorisme consisteix en, donat el node origen, s'inicialitza una llista de camins oberts. A cada passada pel bucle s'agafa el primer camí de la llista i d'aquest l'últim node, si és el node destí tot el camí s'afegeix a la llista de camins solució sinó s'expandeix el darrer node del camí generant tants camins nous com fills s'obté, aquestos s'afegiran a la llista de camins oberts si no superen el temps màxim i no hi ha cap node repetit dins del camí, així evitem que un camí passi dos cops per la mateixa ciutat. La cerca s'aturarà quan la llista de camins oberts estigui buida o haguem trobat suficients camins solució, per defecte es buscaran 10 camins.

```
Algorisme(node_inici,node_final, temps_max)
    Afegir al Cami Inicial el node_inici;
    Afegir a la Llista_Camins_Oberts el Cami Inicial;
    Mentre no(buida(Llista_Camins_Oberts)) i no(numCaminsSolucio<maxim) fer
        Cami:= primer(Llista_Camins_Oberts);
        node:= ultim node de Cami;
        Si (node == node_final) llavors
            Afegir Cami a Llista_Camins_Solució;
        Sino
            Fills=successors(node);
            Per a tots els fills
                Si (Cami no conte fill i costFill < temps_max) llavors
```

```

                                Afegir fill a Cami;
                                Afegir Cami a Llista_Camins_Oberts;
                            FiSi
                        FiPer
                    FiSi
                Fi_mentres
            Retorna Llista_Camins_Solució;
        Fi_algo

```

Com es pot veure no s'utilitza cap funció heurística que permeti arribar abans a les rutes solució, així doncs una possible millora per trobar abans els camins solució seria la utilització d'alguna funció heurística.

6.2. L'agent coordinador de rutes.

Com s'ha dit anteriorment l'agent coordinador de rutes és l'encarregat de rebre les peticions de l'agent coordinador de trasplants per trobar les possibles rutes entre dues ciutats, controlar tot el procés i enviar els resultats. Anem a veure com realitza aquesta tasca.

L' agent coordinador realitza les següents accions al seu setup:

- Registra els seus serveis al DF: servei "CoordinarRuta".
- Inicialitza l' objecte "Grafer mitjà del fitxer "ciutats.txt".
- Inicialitza les variables "AgentTrens" i "AgentAvions" amb les direccions dels agents que donen els serveis "CoordinarTrens" i "CoordinarAvions" respectivament.
- Afegeix el comportament "RebreP".

El fet de registrar-se al DF serveix per donar publicitat dels serveis que realitza l'agent per tal de que altres agents els puguin utilitzar. Això no implica un compromís o obligació de realitzar qualsevol petició de servei per part d'altres agents. Un agent pot refusar una petició d'un servei que ha estat registrat al DF. Per aquest agent l'únic servei que es registrarà al DF serà "CoordinarRuta".

Si qualsevol de les accions anteriors no es realitza correctament l' agent visualitzarà el corresponent missatge d' error i terminarà, ja que no podria dur a terme el servei que té per objectiu si no realitza abans aquestes accions, exceptuant la inicialització de les variables "AgentTrens" i "AgentAvions", degut a que el que fa és demanar al DF que realitzi una cerca del agents que donen els serveis esmentats abans i potser que quan s'intenti fer aquesta acció aquests agents encara no hagin estat registrats al DF, per tant si no es troben aquests agents es visualitzarà un missatge que ho indiqui però l'agent continuarà executant-se de forma normal.

El comportament "BehaviourRebreP": aquest comportament és de tipus "OneShotBehaviour", això significa que només s'executa un cop. En el seu constructor s'inicialitza la variable "patron" que defineix una plantilla de missatge amb protocol "fipa-request" i de tipus "request". Ja al mètode "action" el primer que fa és quedar-se bloquejat esperant un missatge (petició) per mitjà del mètode "blockingReceive", que bloqueja tot l'agent fins que no arribi un missatge que concordi

amb el patró passat com a paràmetre, que serà inicialitzat al constructor. Per tant fins que no arribi un missatge de tipus “request” amb protocol “fipa-request” l’agent no farà res.

El protocol “fipa-request” simplement permet a un agent demanar a un altre que realitzi una acció. L’agent receptor ha de respondre positivament o negativament, segons l’esquema vist a la secció 2.5.2.

Un cop arribat un missatge de l’estil esperat el següent que fa es comprovar el format i els valors de cada camp del contingut del missatge. Com ja s’ha dit el format del missatge és el següent:

```
ciutat_origen-ciutat_desti-data_max-hora_max
data_max -> dia/mes/any
hora_max -> hora:min
```

Regles per a als valors dels camps del missatge:

- Tant la ciutat origen com la ciutat destí han d’existir com a clau a l’objecte “Graf”, es ha de tenir una entrada al fitxer “ciutats.txt”.
- “dia”, “mes”, “hora” i “min” ha de ser nombres de dos dígit i “any” de quatre dígit.
- El nombre de mil·lèsimes de segon entre la data actual i la data màxima ha de ser superior a 0, es a dir, no s’accepten dates posteriors a la data actual.

Si algun dels valors no és correcte s’enviarà com a resposta un missatge de tipus “refuse” seguint el protocol “fipa-request”, amb el contingut:

“Error de format o de contingut al missatge. Format: ciutat_origen-ciutat_desti-dia/mes/any-hora:minut”

En cas contrari s’enviarà com a resposta un missatge de tipus “agree” amb el mateix contingut que el missatge rebut, es visualitzaran per pantalla els valors rebuts, es cridarà al mètode “CercaCamins” de l’objecte “Graf” i afegeix un nou comportament del tipus “BehaviourDemanaInfo”.

El comportament “BehaviourDemanaInfo”: a aquest comportament se li passa com a paràmetres el resultat de l’execució del mètode “CercaCamins” en un objecte de la classe “ArrayList”, el temps màxim de que és disposa en un objecte de la classe “Calendar” i la direcció de l’agent que ha fet la petició en un “String”. És de tipus “SimpleBehaviour” és a dir s’executa tants cops com sigui necessari fins que al seu mètode “done” retorni true. Al constructor d’aquest comportament el primer que es fa és comprovar si estan inicialitzades les variables “AgentTrens” i “AgentAvions” esmentades abans, en el cas de no estar inicialitzades es torna a intentar inicialitzar-les per mitjà del mètode “BuscaServei” que retorna la direcció dels agents que realitzen un servei determinat, en el cas de no haver pogut inicialitzar cap de les dues variables s’envia com a resposta un missatge de tipus “failure” amb el contingut:

“No s’ ha trobat agent encarregat del transport en trens ni avions.”

I es posa la variable “finalitza” a true, aquesta variable és la que es retorna al mètode “done”, per tant termina el comportament.

En cas contrari s'inicialitzen algunes variables com ara la “camins” que és un *ArrayList* que conté les rutes que volem confirmar. Després passa a executar-se el mètode “action”. En aquest hi ha una sentència “switch” que segons el valor de la variable “state” realitza una acció o una altra, així els diferents passos són:

- 1) Construcció del contingut del missatge que s'enviarà a l'agent encarregat d'un transport determinat: el que es pretén és recórrer la llista de camins, per cada camí i cadascun dels seus trams confirmar si hi ha alguna companyia que realitza el tram, així el que s'enviarà a l'agent del transport és una ciutat origen i una ciutat destí. Després passem al següent estat.
- 2) S'acaba de crear el missatge de tipus “request”, protocol “fipa-request”, amb el destí de l'agent encarregat del medi de transport que es necessita, sempre que s'hagi trobat abans l'agent, sinó es passa a l'estat anterior. Un cop acabat de crear el missatge s'envia i s'espera la resposta amb el mètode “blockingReceive” passant com a paràmetre la variable “patronAgreeORefuse”, així només intentarà rebre un agree o un refuse. En cas de rebre un refuse es terminarà el procés posant la variable “finalitza” a cert, i s'enviarà el mateix refuse a l'agent origen que havia iniciat el procés, és a dir, a l'agent coordinador de trasplants. Així l'usuari o l'agent origen podran saber el motiu del refús. En cas de que el missatge rebut sigui un “agree” llavors passem al següent estat.
- 3) Seguint el protocol “fipa-request” ara per mitjà del mètode “blockingReceive” passant-li com a paràmetre la variable “patronInformOFailure” esperem rebre un missatge de tipus “inform” o “failure”. En el cas de que sigui un “failure” el reenviem a l'agent coordinador de trasplants i acabem el comportament posant la variable “finalitza” a cert. Si és un “inform” llavors es crida al mètode “TractaInfo” que descomposa el contingut del missatge agafant cada camp, el format del missatge que es rebrà és el següent:

```
$nom_companyia id_rec id_trans hora_sortida hora_arribada preu- .... -id_rec id_trans
hora_sortida hora_arribada preu-
.....
$nom_companyia id_rec id_trans hora_sortida hora_arribada preu- .... id_rec id_trans
hora_sortida hora_arribada preu-
```

Aquests seran tots els possibles transports que es podran agafar per fer un dels trams d'un dels camins. De totes les possibilitats agafem la que arribi abans al destí i no hagi superat el temps màxim que tenim per fer el recorregut, i guardem la informació en un objecte de la classe “moviment”, aquest objecte és el que retorna el mètode “TractaInfo”. Tots els objectes “moviment” del mateix camí es guarden en una llista anomenada “ruta”, quan terminem amb el camí l'objecte “ruta” és guarda en una altra llista anomenada “rutes”. Si per a algun dels trams no hi ha cap transport que el realitzi o que arribi abans del temps màxim llavors passem a tractar la següent ruta ja que aquesta no es podrà dur a terme. Un cop s'hagin comprovat totes les rutes es crida al mètode “EnviaResultats” que recorre la llista “rutes”. S'utilitzaran dues variables que guardaran el millor preu i la millor hora d'arribada, recorrent la llista si la ruta actual és menys cara o arriba abans comparant-la amb els valors de les variables anteriors, llavors s'actualitzen les variables i s'afegeix la ruta al resultat. Així la primera ruta sempre resultarà escollida, s'actualitzaran les variables amb el seu preu i temps d'arribada, i es compararan aquests valors amb la següent ruta. D'aquesta forma es seleccionaran les

rutes menys cares i les que arriben abans al destí, el resultat es envia en un missatge de tipus “inform” a l’agent que havia fet la petició.

El missatge enviat com a resposta tindrà el següent format:

```
-RUTA 1: ciutat_1 hora_sortida ciutat_2 hora_arribada dia preu nom_companyia id_rec  
id_trans-.....-ciutat_n hora_sortida ciutat_n+1 h._arribada dia preu nom_companyia id_rec  
id_trans-RUTA 2: ciutat_1 hora_sortida ciutat_2 h._arribada dia preu nom_companyia id_rec  
id_trans- .....
```

Si no hi ha cap ruta possible s’enviarà un missatge buit.

Val a dir que si en tot aquest procés es produeix qualsevol tipus d’excepció llavors s’envia un missatge “failure” i es para el procés.

Un cop acabat el comportament “DemanaInfo” es torna a afegir un comportament “RebreP”.

6.3. Els agents “AgentTrens” i “AgentAvions”

Aquests dos agents són els encarregats d’interaccionar amb els agents que agentifiquen les Bases de Dades d’un medi de transport determinat. El codi de tots dos és molt semblant, la única diferència és que tracten medis de transport diferents. Així una possible millora seria crear un agent anomenat “AgentTransport” que tingués com a paràmetre al constructor el nom del medi de transport del que s’ha d’encarregar, aquesta millora no s’ha implementat per dificultats al passar els paràmetres als agents en el moment de la seva creació.

Aquestos agents realitzen les següents accions al seu setup:

- Registren els seus serveis al DF: servei “CoordinarTrens” l’un i “CoordinarAvions” l’altre, tots dos es registren amb el tipus “AgentTransport”.
- Afegeixen el comportament “RebreP”.

El comportament “BehaviourRebreP”: aquest comportament és semblant al que té el mateix nom en l’agent coordinador de rutes, però aquest és de tipus “CyclicBehaviour”, això significa que quan acaba es torna a executar de forma cíclica, sense fi. En el seu constructor s’inicialitza també la variable “patron” que defineix una plantilla de missatge amb protocol “fipa-request” i de tipus “request”. Ja al mètode “action” el primer que fa és mirar si en la cua de missatges hi ha algun del tipus esmentat per mitjà del mètode “receive”, en el cas de que no hi hagi cap retornarà “null” llavors es crida al mètode “block” que bloqueja el comportament esperant que es produeixi qualsevol event com ara rebre un nou missatge. Per tant fins que no arribi un missatge de tipus “request” amb protocol “fipa-request” l’agent no farà res. En arribar el missatge, que s’espera tingui el format “ciutat_origen-ciutat_destí”, s’envia com a resposta un missatge “agree” amb el mateix contingut, després es comprova si s’ha realitzat anteriorment la mateixa consulta mirant si a la taula hash “repe” es troba cap entrada amb la mateixa clau que el contingut, si és així s’envia un missatge “inform” amb el valor que es troba a la taula, sinó es comprova si la variable “grupAgents” ha estat inicialitzada anteriorment, aquesta variable conté el nom de tots els agents que agentifiquen una base de dades d’un medi de transport determinat, el primer cop no estarà inicialitzada, així que s’afegirà el comportament “BuscarServei”, un cop inicialitzada s’afegirà directament el comportament “RebreInfo”. En tot el procés si es

produeix qualsevol excepció s'envia com a resposta un missatge de tipus "refuse" amb el següent contingut:

"Possible error de format. Format: ciutat_origen-ciutat_desti. Excepcio:"

El comportament "BehaviourBuscarServei": és de tipus "OneShotBehaviour", i cerca tots els agents que donen el servei de "consultaBDTrens", si és l'agent de trens i "consultaBDAvions" per a l'agent d'avions. El resultat de la cerca el posa a la variable "grupAgents", si s'ha trobat algun agent llavors s'envia a tots ells un missatge de tipus "request" i protocol "fipa-request", amb el mateix contingut que el missatge rebut (ciutat_origen-ciutat_desti) i s'afegeix el comportament "RebreInfo" sinó s'envia com a resposta un missatge de tipus "refuse" amb el contingut:

"Error en la cerca, o no trobat cap agent registrat a un DF que realitzi el servei que necessitem. Excepcio:"

El comportament "BehaviourRebreInfo": és de tipus "SimpleBehaviour". El primer que fa és esperar un missatge de resposta que sigui "agree", "failure", "inform" o "refuse". Si és un "agree", un "failure" o un "refuse" només incrementa un comptador per cada tipus de missatge, en cas de ser "inform" també s'incrementa un comptador però a més es comprova que el contingut del missatge no estigui buit i llavors es guarda la informació rebuda en un string anomenat "resultats" afegint el caràcter '\$' al principi per distingir els diferents resultats, a més també es guarda en la taula hash anomenada "repes" si aquesta taula arriba a tenir tants elements com el valor "MaxRepes" llavors es buida la taula per poder emmagatzemar altres resultats. Llavors si encara queden missatges per rebre no es fa res més, així es torna a executar el comportament esperant el següent missatge, en cas contrari s'envia la informació obtinguda en un missatge de tipus "inform" i es posa la variable "finalitza" a true fent que s'acabi el comportament. Si és produeix qualsevol excepció s'envia com a resposta un missatge de tipus "refuse" amb el tipus d'excepció i es finalitza el comportament posant la variable "finalitza" a true.

Com ja s'ha vist, el procés de demanar informació als agents que agentifiquen les bases de dades és fa en paral·lel, s'envia la consulta a totes les bases d'un mateix medi de transport alhora, així obtenim una major eficiència.

6.4. Els agents que agentifiquen les BDs

Tenim quatre bases de dades, dues de companyies ferroviàries i dues de companyies aèries, agentificades per els agents: "AgentRenfe", "AgentRailRibes", "AgentIberia" i "AgentHispanAir". Degut a que s'ha utilitzat el mateix disseny a les quatre BDs els agents són molt semblants. La connexió amb la base de dades s'explicarà més detingudament a la secció 5.7, en aquest apartat només s'explicaran per sobre.

Aquestos agents realitzen les següents accions al seu setup:

- Intenten carregar el driver JDBC-ODBC per mitjà del mètode "forName" de la classe "Class". Si no es troba el driver es visualitza un missatge indicant el problema i es termina el comportament.

- Registren els seus serveis al DF: servei "ConsultarBDTrens" els agents "AgentRenfe" i "AgentRailRibes", mentre que els agents "AgentIberia" i "AgentHispanAir" es registren amb el servei "ConsultarBDAvions".
- Afegeixen el comportament "Rebre".

El comportament "BehaviourRebre": és de tipus "OneShotBehaviour". Es pot dir que és idèntic al comportament "RebreP" dels agents de transport. Espera rebre un missatge tipus "request" amb el contingut "ciutat_origen-ciutat_desti", un cop rebut envia un missatge "agree" amb el mateix contingut i afegeix el següent comportament, però si hi ha qualsevol excepció envia un missatge "failure" i torna a esperar un altre missatge.

El comportament "BehaviourConsulta": és de tipus "OneShotBehaviour" i s'encarrega de realitzar la consulta a la base de dades. Primer per mitjà del mètode "getConnection" de la classe "DriverManager" s'intenta establir una connexió amb la base de dades determinada per la variable "url" que segons l'agent serà "jdbc:odbc:railRibes", "jdbc:odbc:renfe", etc. aquest mètode ens retorna un objecte de tipus "Connection" que és el que ens permet crear sentències SQL dirigides a la base de dades per mitjà del seu mètode "createStatement" que ens retorna un objecte de la classe "Statement". Després s'executa la consulta per mitjà del mètode "executeQuery" de l'objecte "Statement", a aquest mètode li passem un string que conté la consulta SQL:

```
SELECT A.*, P1.temps, P2.temps
FROM Assigna A, Paradas P1, Paradas P2
WHERE P1.ciutat = ciutat_origen and
      P2.ciutat = ciutat_desti and
      P1.id_rec = P2.id_rec and
      A.id_rec = P1.id_rec and
      P1.temps < P2.temps
```

Aquesta sentència ve determinada pel disseny de la base de dades, així que s'explicarà en la següent secció.

Un cop realitzada la consulta obtenim un objecte de la classe "ResultSet" que ens permet obtenir cadascun dels camps escollits amb el "select". Tota la informació s'emmagatzema en una variable de tipus string anomenada "resultats" que tindrà el següent format:

```
Nom_companyia id_tren id_recorregut hora_sortida hora_arribada preu-
id_tren id_recorregut hora_sortida hora_arribada preu-
.....
id_tren id_recorregut hora_sortida hora_arribada preu-
```

Aquest string s'enviarà per mitjà d'un missatge "inform" a l'agent que va demanar el servei, després es tanca la connexió amb la base de dades i s'afegeix un nou comportament "Rebre". Si succeeix qualsevol excepció durant el procés llavors s'envia un missatge de tipus "failure".

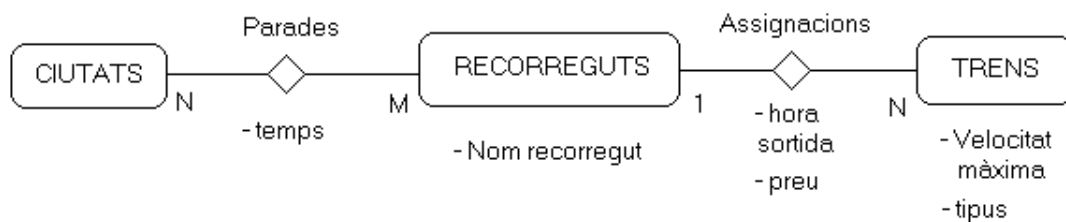
6.5. Disseny de les BDs

Totes quatre bases de dades parteixen del mateix disseny, l'única diferència és el nom d'alguna de les taules, així només s'explicarà el disseny d'una de les bases de dades.

Tindrem tres entitats: ciutats, recorreguts i trens.

Cada recorregut estarà assignat a un o més trens. Els recorreguts tindran establertes les parades que hauran de fer per les ciutats que passen, per cada parada tindrem el temps que triga en arribar des de la ciutat inicial del recorregut, és a dir, si volem l' hora en que arriba un tren a una ciutat només hem de sumar l' hora inicial del recorregut, que es trobarà a la relació "Assignacions", i el temps que triga en arribar a aquella ciutat, que es troba a la relació "Parades", d' aquesta forma els trens poden fer un mateix recorregut iniciat a una hora i data diferent, i saber fàcilment l' hora d' arribada a qualsevol ciutat, sense tenir que repetir tot el recorregut amb diferents hores per a cada recorregut que s' iniciés a una hora diferent.

Un tren només tindrà assignat un recorregut, però un mateix recorregut podrà ser assignat a diferents trens. Per cada assignació tindrem el preu que costa anar entre dues ciutats consecutives del viatge segons el tren que el realitza, i l' hora de sortida de la primera ciutat del recorregut, no s'afegeix la data de sortida ja que per simplificar les entrades a la base de dades es suposarà que cada dia els trens realitzen els mateixos viatges.



Les taules al model relacional són:

Parades (**id_rec**, ciutat, temps)

Recorreguts (**id_rec**, nom)

Assigna (**id_tren**, **id_rec**, hora, preu)

Trens (**id**, vel_max, tipus)

No tenim cap taula per a les ciutats degut a que només necessitem saber el nom de les ciutats on paren els trens i aquesta informació ja es troba a la taula "Parades".

La taula "Parades" conté l'identificador de recorregut, el nom de la ciutat on es produeix la parada i el temps que s'ha trigat en arribar.

6.6. Creació de les BDs en Access

La informació que es troba a continuació és la que dona el “Documentador” d’Access, determina les propietats de tots de taules, camps i relacions entre taules.

BD -> renfe.mdb

Tabla: **Assigna**

Propiedades

ActivarOrden:	Falso	Def. actualizable:	Verdadero
Fecha de creación:	11/07/00 14:33:42	RecordCount:	27
Última actualización:	13/08/00 17:50:01		

Columnas

Nombre		Tipo	Tamaño
id_tren	AllowZeroLength: Falso Atributos: Tamaño fijo Campo de origen: id_tren ColumnHidden: Falso ColumnOrder: Predeterminado ColumnWidth: Predeterminado DisplayControl: Cuadro de texto Lugares decimales: Automático Orden de combinación: General Posición ordinal: 1 Required: Falso Tabla de origen: Assigna Valor predeterminado: 0	Número (largo)	4
id_rec	AllowZeroLength: Falso Atributos: Tamaño fijo Campo de origen: id_rec ColumnHidden: Falso ColumnOrder: Predeterminado ColumnWidth: Predeterminado DisplayControl: Cuadro de texto Lugares decimales: Automático Orden de combinación: General Posición ordinal: 2 Required: Falso Tabla de origen: Assigna Valor predeterminado: 0	Número (largo)	4
hora	AllowZeroLength: Falso Atributos: Tamaño fijo Campo de origen: hora ColumnHidden: Falso ColumnOrder: Predeterminado ColumnWidth: Predeterminado Formato: Hora corta Orden de combinación: General Posición ordinal: 3 Required: Falso Tabla de origen: Assigna	Fecha/Hora	8
preu	AllowZeroLength: Falso Atributos: Tamaño fijo Campo de origen: preu	Moneda	8

ColumnHidden:	Falso
ColumnOrder:	Predeterminado
ColumnWidth:	Predeterminado
Formato:	Moneda
Lugares decimales:	Automático
Orden de combinación:	General
Posición ordinal:	4
Required:	Falso
Tabla de origen:	Assigna
Valor predeterminado:	0

Relaciones

RecoregutsAssigna

	Recoreguts		Assigna
id_rec	1	∞	id_rec
Atributos:			Exigir
Atributos:			Uno a varios

TrensAssigna

	Trens		Assigna
Id	1	∞	id_tren
Atributos:			Exigir
Atributos:			Uno a varios

Índices de tabla

Nombre	Número de campos
id_rec	1
Agrupado:	Falso
Cuenta distinta:	26
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	id_rec
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	id_rec, Ascendente
id_tren	1
Agrupado:	Falso
Cuenta distinta:	27
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	id_tren
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	id_tren, Ascendente
PrimaryKey	2
Agrupado:	Falso
Cuenta distinta:	24
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	PrimaryKey
Principal:	Verdadero
Required:	Verdadero
Única:	Verdadero
Campos:	id_tren, Ascendente
RecoregutsAssigna	1
Agrupado:	Falso
Cuenta distinta:	26
Extranjero:	Verdadero

Ignorar Nulos:	Falso
Nombre:	RecorregutsAssigna
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	id_rec, Ascendente
TrensAssigna	1
Agrupado:	Falso
Cuenta distinta:	27
Extranjero:	Verdadero
Ignorar Nulos:	Falso
Nombre:	TrensAssigna
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	id_tren, Ascendente

Permisos de usuario

admin

Permisos de grupo

Admins
Users

Tabla: Paradas

Propiedades

ActivarOrden:	Verdadero	Def. actualizable:	Verdadero
Fecha de creación:	11/07/00 14:29:35	OrdenarPor:	Paradas.id_rec
RecordCount:	134	Última actualización:	20/07/00 21:34:02

Columnas

Nombre	Tipo	Tamaño
id_rec	Número (largo)	4
AllowZeroLength:	Falso	
Atributos:	Tamaño fijo	
Campo de origen:	id_rec	
ColumnHidden:	Falso	
ColumnOrder:	Predeterminado	
ColumnWidth:	Predeterminado	
DisplayControl:	Cuadro de texto	
Lugares decimales:	Automático	
Orden de combinación:	General	
Posición ordinal:	1	
Required:	Falso	
Tabla de origen:	Paradas	
Valor predeterminado:	0	
ciutat	Texto	50
AllowZeroLength:	Falso	
Atributos:	Longitud variable	
Campo de origen:	ciutat	
ColumnHidden:	Falso	
ColumnOrder:	Predeterminado	
ColumnWidth:	Predeterminado	
DisplayControl:	Cuadro de texto	
Orden de combinación:	General	
Posición ordinal:	2	
Required:	Falso	
Tabla de origen:	Paradas	

temps	AllowZeroLength:	Falso	Fecha/Hora
	Atributos:	Tamaño fijo	
	Campo de origen:	temps	
	ColumnHidden:	Falso	
	ColumnOrder:	Predeterminado	
	ColumnWidth:	Predeterminado	
	Formato:	Hora corta	
	Orden de combinación:	General	
	Posición ordinal:	4	
	Required:	Falso	
	Tabla de origen:	Paradas	

8

Relaciones

RecorregutsParadas

Recorreguts	Paradas
id_rec	1 ∞ id_rec
Atributos:	Exigir
Atributos:	Uno a varios

Índices de tabla

Nombre	Número de campos
id_ciutat	1
Agrupado:	Falso
Cuenta distinta:	49
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	id_ciutat
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	ciutat, Ascendente
id_rec	1
Agrupado:	Falso
Cuenta distinta:	31
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	id_rec
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	id_rec, Ascendente
PrimaryKey	3
Agrupado:	Falso
Cuenta distinta:	117
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	PrimaryKey
Principal:	Verdadero
Required:	Verdadero
Única:	Verdadero
Campos:	id_rec, Ascendente ciutat, Ascendente temps, Ascendente
RecorregutsParadas	1
Agrupado:	Falso
Cuenta distinta:	31
Extranjero:	Verdadero
Ignorar Nulos:	Falso
Nombre:	RecorregutsParadas
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	id_rec, Ascendente
temps	1

Agrupado:	Falso
Cuenta distinta:	7
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	temps
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	temps, Ascendente

Permisos de usuario

admin

Permisos de grupo

Admins
Users

Tabla: Recorreguts

Propiedades

ActivarOrden:	Falso	AltoDeLaFila:	300
Def. actualizable:	Verdadero	Fecha de creación:	11/07/00 14:30:20
RecordCount:	27	Última actualización:	14/08/00 11:49:46

Columnas

Nombre		Tipo	Tamaño
id_rec	AllowZeroLength: Falso Atributos: Tamaño fijo Campo de origen: id_rec ColumnHidden: Falso ColumnOrder: 1 ColumnWidth: Predeterminado DisplayControl: Cuadro de texto Lugares decimales: Automático Orden de combinación: General Posición ordinal: 1 Required: Falso Tabla de origen: Recorreguts Valor predeterminado: 0	Número (largo)	4
nom	AllowZeroLength: Falso Atributos: Longitud variable Campo de origen: nom ColumnHidden: Falso ColumnOrder: Predeterminado ColumnWidth: 2775 DisplayControl: Cuadro de texto Orden de combinación: General Posición ordinal: 2 Required: Falso Tabla de origen: Recorreguts	Texto	50

Relaciones

RecorregutsAssigna

Recoreguts	Assigna
id_rec	1 ∞ id_rec
Atributos:	Exigir
Atributos:	Uno a varios

RecoregutsParadas

Recoreguts	Paradas
id_rec	1 ∞ id_rec
Atributos:	Exigir
Atributos:	Uno a varios

Índices de tabla

Nombre	Número de campos
id_rec	1
Agrupado:	Falso
Cuenta distinta:	27
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	id_rec
Principal:	Falso
Required:	Falso
Única:	Falso
Campos:	id_rec, Ascendente
PrimaryKey	1
Agrupado:	Falso
Cuenta distinta:	27
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	PrimaryKey
Principal:	Verdadero
Required:	Verdadero
Única:	Verdadero
Campos:	id_rec, Ascendente

Permisos de usuario

admin

Permisos de grupo

Admins
Users

Tabla: **Trens**

Propiedades

ActivarOrden:	Falso	Def. actualizable:	Verdadero
Fecha de creación:	01/07/00 11:41:37	RecordCount:	27
Última actualización:	13/08/00 17:50:01		

Columnas

Nombre	Tipo	Tamaño
Id	Número (largo)	4
AllowZeroLength: Falso Atributos: Tamaño fijo Campo de origen: Id ColumnHidden: Falso ColumnOrder: Predeterminado ColumnWidth: Predeterminado DisplayControl: Cuadro de texto Lugares decimales: Automático Orden de combinación: General Posición ordinal: 1 Required: Falso Tabla de origen: Trens Valor predeterminado: 0		
Vel_max	Número (largo)	4
AllowZeroLength: Falso Atributos: Tamaño fijo Campo de origen: Vel_max ColumnHidden: Falso ColumnOrder: Predeterminado ColumnWidth: Predeterminado DisplayControl: Cuadro de texto Formato: Número general Lugares decimales: Automático Orden de combinación: General Posición ordinal: 2 Required: Falso Tabla de origen: Trens		
tipus	Texto	50
AllowZeroLength: Falso Atributos: Longitud variable Campo de origen: tipus ColumnHidden: Falso ColumnOrder: Predeterminado ColumnWidth: Predeterminado DisplayControl: Cuadro de texto Orden de combinación: General Posición ordinal: 3 Required: Falso Tabla de origen: Trens		

Relaciones

TrensAssigna

Trens	Assigna
Id	1 ∞ id_tren
Atributos:	Exigir
Atributos:	Uno a varios

Índices de tabla

Nombre	Número de campos
PrimaryKey	1
Agrupado:	Falso
Cuenta distinta:	27
Extranjero:	Falso
Ignorar Nulos:	Falso
Nombre:	PrimaryKey
Principal:	Verdadero
Required:	Verdadero
Única:	Verdadero
Campos:	Id, Ascendente

Permisos de usuario

admin

Permisos de grupo

Admins
Users

6.7. Establiment de la connexió amb les BDs per mitjà de JDBC.

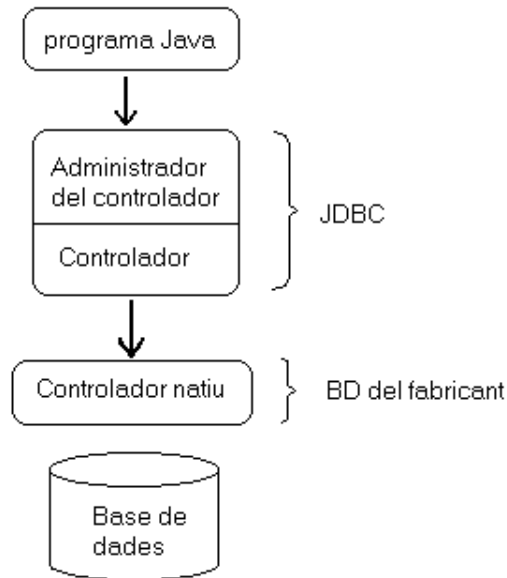
Pasos a seguir per comunicar amb una base de dades relacional:

- 1) Establiment d'una connexió entre el programa Java i l'administrador de la base de dades.
- 2) Enviar una sentència SQL a la base de dades mitjançant l'objecte *Statement*.
- 3) Recuperar els resultats de la base de dades i utilitzar-los al programa.

Utilització de DriverManager: JDBC està dissenyat per treballar amb diferents sistemes d'administració de bases de dades de diferents fabricants. Per poder establir una connexió amb la base de dades, l'entorn Java ha de carregar el controlador de la base de dades. La classe "DriverManager" és l'encarregada de carregar i descarregar aquests controladors. Aquesta classe manté una estructura de dades que conté els propis objectes del controlador, així com informació sobre els "Drivers".

Càrrega dels drivers: els drivers de JDBC normalment estan fets pel fabricant de la base de dades; accepten connexions i sentències JDBC per un costat i realitzen crides natives a la BD per l'altre.

Alguns fabricants de bases de dades encara no tenen controladors JDBC, però sí controladors compatibles amb l'estàndar de *Microsoft Open Database Connectivity* (ODBC), connectivitat oberta de base de dades. Si es treballa en un entorn Windows, es probable que el fabricant de la BD tingui un controlador ODBC.



Per a utilitzar un driver JDBC (inclós el controlador pont entre JDBC i ODBC) s’ha de demanar al sistema que el carregui. La forma més fàcil d’especificar els drivers és establir la propietat del sistema “jdbc.drivers”. El controlador pont JDBC-ODBC pot carregar-se des de la línia de comandes mitjançant la ordre:

```
Java -Djdbc.drivers=sun.jdbc.odbc.JdbcOdbcDriver MyAplication
```

Donat que és l’usuari final i no el programador el qui controla normalment l’accés a la línia de comandes, es aconsellable incloure el nom del Driver directament al programa. Aquest mètode és menys flexible, l’usuari no pot canviar amb facilitat a una altra versió de la base de dades, però resulta més robust (aquest mètode és l’utilitzat al projecte). Codi que carrega el controlador pont JDBC-ODBC:

```
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (ClassNotFoundException e) {
    System.err.println("No s' ha pogut trobar la classe del driver JDBC/ODBC.");
}
```

Utilització de la classe “Connection”: un cop que registrat el driver amb el “DriverManager”, pot utilitzar-se per connectar amb la base de dades. Per fer-ho s’ha de demanar a “DriverManager” que realitzi una nova “Connection”. Aquest invoca al “Driver” i retorna una referència a la “Connection”. És necessari especificar la ubicació de la base de dades i en molts casos un nom i una contrasenya. Per exemple, es pot utilitzar el següent codi:

```
Connection con=DriverManager.getConnection(url,login,password);
```

Quan “DriverManager” rep una petició “getConnection()”, envia la URL de JDBC a cada driver registrat. El primer driver que reconeix la URL i declara que pot connectar-se és el que estableix la connexió. Si cap dels drivers reconeix la URL,

“DriverManager” llença una excepció “SQLException” indicant que no existeix un controlador adient.

Utilització de l'objecte “Statement”: aquests objectes s'utilitzen per allotjar les sentències SQL. Quan s'envia un objecte “Statement” a la base de dades a través de la “Connection”, la base de dades executa la consulta i retorna un “ResultSet”.

Exemple:

```
ResultSet theSet= theStatement.executeQuery(“SELECT * FROM books”);
```

Manipulació del conjunt de resultats: la classe “ResultSet” funciona com un cursor de SQL, ja que permet examinar les dades de la sentència SELECT fila a fila. Per passar d'una fila a la següent s'utilitza el mètode “next()”, aquest retorna un valor booleà indicant si existeixen més files.

Dins d'una fila es pot llegir el contingut de les columnes d'esquerra a dreta. Es disposa de mètodes “getType()” per a tots els tipus primitius que SQL pot emmagatzemar.

Exemple:

```
ResultSet resultats= theStatement.executeQuery(“SELECT titol, autor, preu FROM books”);
While (resultats.next())
{
    String titol= resultats.getString(“titol”);
    String autor= resultats.getString(“autor”);
    Float preu= resultats.getFloat(“preu”);
    System.out.println(titol+” “+autor+” “+ preu.toString());
}
```

Un cop acabats “Statements” i “Connection”, es aconsellable cridar a “close()” en tots dos casos. Això alliberarà recursos abans que ho fassi el recolector de brosa.

6.8. Registre de la base de dades al sistema.

Tot el projecte s'ha desenvolupat sota “Windows 98”, així que només s'explicarà el registre en aquest sistema operatiu.

Passos:

- 1) Panel de control
- 2) ODBC de 32 bits
- 3) Etiqueta: “DSN de usuario”.
- 4) Agregar
- 5) Microsoft Access Driver-> “Finalitzar”
- 6) “Seleccionar” -> Indicar la base de dades que es vulgui registrar.
- 7) “Nombre del origen de datos” -> posar el nom de la base de dades.
Exemple: “railRibes”.
- 8) “Aceptar”.

7. CODI FONT

7.1. La classe fill.

```
/** Classe que defineix un fill d'un node del graf
    de connexions entre les ciutats */

class fill {
    private long temps;          // temps en arribar a la ciutat en milisegons
    private String ciutat;       // nom de la ciutat destí
    private String transport;    // medi de transport utilitzat

    public boolean equals(Object o)
    { fill f= (fill) o;
      return ciutat.equals(f.ciutat);
    }

    //constructor de la classe
    public fill (String c,int h, int m,String medi)
    {
        ciutat=c;
        transport=medi;
        temps= 360000*(long)h+6000*(long)m;
    }

    public String getCiutat() { return ciutat; }
    public String getMedi() { return transport; }
    public long getTemps() { return temps; }
    public void Visu()
    {
        System.out.println(ciutat+" "+transport+" "+temps);
    }
}
```

7.2. La classe fitxer.

```
import java.util.*;
import java.io.*;

/** Classe que permet manipular un fitxer de text, per
    mitjà dels objectes "FileInputStream" i "File" */

class fitxer
{
    private File fitxerF;
    private FileInputStream dades;
    private int ultimaDada;
    private boolean fiFitxer=false;

    /** Constructor de la classe, se li passa el nom del
        fitxer, llença una excepció si no troba el fitxer */
    public fitxer(String nom) throws FileNotFoundException
    {
        fitxerF= new File(nom);
        dades = new FileInputStream(fitxerF);
    }

    /** Com es pot deduir, llegeix caràcters fins que troba
        un que sigui diferent de blanc, també passa els "returns" (13,10),
        i retorna el primer codi ASCII diferent */
    public int PassaBlancs() throws IOException
    {
        int dada=dades.read();
        while((char) dada==' ' || dada==13 || dada==10) dada=dades.read();
        return dada;
    }
}
```



```

import java.util.*;
import java.io.*;

/** Classe que defineix un graf */

class Graf {
    /** Taula hash que representarà el graf de connexions entre
        diferents ciutats */
    private Hashtable graf=new Hashtable();
    private int MaxNumRutes=10;

    //constructor de la classe
    public Graf ()
    {
    }

    public Graf(int MaxNumRutes)
    {
        this.MaxNumRutes=MaxNumRutes;
    }

    /** Inicialitza el graf amb el fitxer que té per nom el
        String passat com a paràmetre amb el format:
        Origen-Destí hh:mm medi-Destí2 hh:mm medi2- ..... */
    public void Inicialitza(String nom) throws IOException
    {
        fitxer f=new fitxer(nom);
        String nomCiutat= new String();
        while (!f.FinalFitxer())
        {
            nomCiutat= f.llegirString('-');
            Vector fills=new Vector();
            while(f.UltimaDada()!=10 && !f.FinalFitxer())
            {
                fill branca = new
                    fill(f.llegirString(' '), // Ciutat destí
                        f.llegirInt(':'), // hores que es triga
                        f.llegirInt(' '), // minuts " " "
                        f.llegirString('-')); // medi transport
                fills.add(branca);
            }

            if (!fills.isEmpty())
                if (graf.put(nomCiutat,fills)!=null)
                    System.out.println("Hi han dos entrades amb la mateixa clau al fitxer
"+nom+".");
        }
        f.tancar();
    }

    /** Donades una ciutat origen i una destí, busca al graf alguns
        camins possibles i retorna un "ArrayList" que conté els camins.*/
    public ArrayList CercaCamins(String ciutatO, String ciutatD,long temps)
    {
        ArrayList Loberta=new ArrayList(); // llista de camins oberts
        ArrayList Lcosts =new ArrayList(); // llista d'objectes "Long" que indiquen el
temps
        ArrayList Lcamins=new ArrayList(); // en milisegons per a cada camí
        ArrayList cami =new ArrayList(); // llista de camins solució
        Vector fills =new Vector(); // objecte camí
        Long cost =null;
        String ultimaCiutat;
        int i;

        fill node = new fill(ciutatO,0,0," ");
        cami.add(node);
        Loberta.add(cami);
        Lcosts.add(new Long(0));

        while (!Loberta.isEmpty() && Lcamins.size()<MaxNumRutes)
        {
            cami=(ArrayList) Loberta.remove(0);
            cost=(Long) Lcosts.remove(0);
            node=(fill)camí.get(cami.size()-1);
            ultimaCiutat=node.getCiutat();

```

```

// Si hem arribat a la ciutat destí
if (ultimaCiutat.equals(ciutatD))
{
    // afegim el camí a la llista de camins solució
    Lcamins.add(cami);
} else {
    if (graf.containsKey(ultimaCiutat))
    {
        fills=(Vector) graf.get(ultimaCiutat);
        for (i=0;i<fills.size();i++)
        {
            node=(fill) fills.get(i);
            if (!cami.contains(node))
            {
                // sumem al cost del camí el cost del nou node
                long nouCost= cost.longValue()+ node.getTemps();
                // si el temps maxím no supera al nou cost
                // afegim el nou camí a la llista de camins oberts
                if (nouCost < temps)
                {
                    ArrayList c= new ArrayList();
                    c=(ArrayList) cami.clone();
                    c.add(node);
                    Lcosts.add(new Long(nouCost));
                    Loberta.add(c);
                }
            }
        }
    } else System.out.println(" la clau "+ultimaCiutat+" no existeix");
}
return Lcamins;
}

public void Visualitza()
{
    Collection c=graf.values();
    Iterator i=c.iterator();
    Enumeration e= graf.keys();

    while (e.hasMoreElements())
    {
        System.out.print((String)e.nextElement()+"=");
        Vector f=(Vector) i.next();
        Enumeration en=f.elements();
        while (en.hasMoreElements())
        {
            fill node= (fill) en.nextElement();
            System.out.print(node.getCiutat()+" "+node.getTemps()+" "+
node.getMedi()+"--");
        }
        System.out.println();
    }
    System.out.println();
}

//Donada una clau, retorna el valor que li correspon
public Vector Valors(String clau)
{
    Vector l=new Vector();
    l=(Vector) graf.get(clau);
    return l;
}

public boolean ExisteixClau(String clau)
{
    return graf.containsKey(clau);
}
}

```

7.4. La classe moviment.

```
/** Classe que defineix un moviment d'una ciutat a una altra */

class moviment implements Cloneable {
    private String ciutatO,ciutatD;    // nom de la ciutat  origen i destí
    private String horaS,minS,horaA,minA,preu,info;
    private int dia;

    //constructor de la classe
    public moviment (String co,String cd,int dia,String hs,String ms,String ha,String
ma,String preu,String info)
    {
        ciutatO=co;  ciutatD=cd;      this.dia=dia;
        horaS=hs;    minS=ms;         horaA=ha;
        minA=ma;     this.preu=preu;  this.info=info;
    }

    public boolean equals(Object o)
    {  moviment m= (moviment) o;
      return ciutatO.equals(m.ciutatO) && ciutatD.equals(m.ciutatD);
    }

    protected Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }

    public String getCiutat(){      return ciutatO;  }
    public String getHores() {      return horaS;  }
    public String getMinS()  {      return minS;  }
    public String getHoresA() {      return horaA;  }
    public String getMinA()  {      return minA;  }
    public int getDia()      {      return dia;  }
    public long getPreu()    {      return Long.parseLong(preu);  }
    public String toString()
    {
        return ciutatO+" "+horaS+" "+minS+" "+ciutatD+" "+horaA+" "+minA+" "+dia+" "+preu+"
"+info;
    }
}
```

7.5. La classe AgentCoord.

```
/*
*****
* @Nom Fitxer: AgentCoord.java
* @Data:      11-9-2000
* @Fet per:   Alberto Ribes García
* @Descripció: definició de la classe "AgentCoord", agent encarregat de coordinar
*              les rutes. Utilitza la classe "Graf" per torbar un conjunt de rutes
*              entre dues ciutats capitals de província (d'Espanya) utilitzant un
*              algorisme de cerca amb cicles.
*****
*/

import jade.core.*;          // Importem les classes de Jade:
import jade.core.behaviours.*; // - Totes les classes del nucli (agent,...)
import jade.domain.*;        // - Elements de gestió (AMS,DF,ACC,...)
import jade.lang.acl.*;      // - Llenguatge de comunicació ACL
import java.io.*;            // Altres:
import java.util.*;          // - Entrada / Sortida de dades

public class AgentCoord extends Agent
{
    private Graf graf= new Graf();
    private String AgentTrens=null;
    private String AgentAvions=null;
    private static String NomServeiTrens= "CoordinarTrens";
    private static String NomServeiAvions= "CoordinarAvions";
}
```

```

/** Definim el comportament "RebreP", que només rebrà missatges:
- Tipus          -> REQUEST
- Protocol       -> FIPA-REQUEST
- Format del contingut -> (ciutat_origen-ciutat_desti-data_max-hora_max)
    data_max -> dia/mes/any
    hora_max -> hora:min
*/

class BehaviourRebreP extends OneShotBehaviour
{
    MessageTemplate patron;
    String ciutat0=null,ciutatD=null,data_max=null,hora_max=null;
    private Calendar max=null;

    public BehaviourRebreP() {
        MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
        MessageTemplate mt2 = MessageTemplate.MatchType("request");
        patron = MessageTemplate.and(mt1,mt2);
    }

    long ComprovaArgs(String missatge) throws Exception
    {
        int index1 =0;
        int index2;
        char separador='-';

        // Agafem cada camp del contingut del missatge
        index2 = missatge.indexOf(separador,index1);
        if (index2==-1) throw new Exception();
        ciutat0 = missatge.substring(index1,index2);
        index1 = index2+1; index2 = missatge.indexOf(separador,index1);
        if (index2==-1) throw new Exception();
        ciutatD = missatge.substring(index1,index2);
        index1 = index2+1; index2 = missatge.indexOf('/',index1);
        if (index2==-1) throw new Exception();
        String sdia= missatge.substring(index1,index2);
        index1 = index2+1; index2 = missatge.indexOf('/',index1);
        if (index2==-1) throw new Exception();
        String smes= missatge.substring(index1,index2);
        index1 = index2+1; index2 = missatge.indexOf(separador,index1);
        if (index2==-1) throw new Exception();
        String sany= missatge.substring(index1,index2);
        index1 = index2+1; index2 = missatge.indexOf(':',index1);
        if (index2==-1) throw new Exception();
        String shora= missatge.substring(index1,index2);
        String smin = missatge.substring(index2+1);

        if (ciutat0.length()==0 || ciutatD.length()==0 || sdia.length()==0 ||
            smes.length()==0 || sany.length()==0 || shora.length()==0 ||
            smin.length()==0) throw new Exception();

        System.out.println(" Ciutat origen: -"+ciutat0+"-");
        System.out.println(" Ciutat desti: -"+ciutatD+"-");
        System.out.println(" Data maxima: -"+sdia+"/"+smes+"/"+sany+"-");
        System.out.println(" Hora maxima: -"+shora+": "+smin+"-");

        int dia= Integer.parseInt(sdia); int mes= Integer.parseInt(smes);
        int any= Integer.parseInt(sany); int hora=Integer.parseInt(shora);
        int min= Integer.parseInt(smin);

        max= Calendar.getInstance();
        Calendar avui = Calendar.getInstance();
        System.out.println(" Avui: "+(avui.getTime()).toString());
        max.clear();
        max.set(any,mes-1,dia,hora,min);
        Date maxim = max.getTime();

        System.out.println(" Maxim: "+maxim.toString());
        long temps= maxim.getTime() - (avui.getTime()).getTime();
        if (!graf.ExisteixClau(ciutat0.toUpperCase()) || temps<=0 ||
            !graf.ExisteixClau(ciutatD.toUpperCase()) ||
            (ciutat0.toUpperCase()).equals(ciutatD.toUpperCase())) throw new
Exception();
        data_max= sdia+"/"+smes+"/"+sany;
        hora_max= shora+": "+smin;
        return temps;
    }
}

```

```

public void action()
{
    ACLMessage msg = blockingReceive(patron);    // Esperem a rebre el missatge
    try {
        System.out.println(getLocalName()+ ": comprovant el format del missatge.");
        long temps=ComprovaArgs(msg.getContent());
        System.out.println(getLocalName()+ ": "+temps+" milisegons disponibles.");

        // Enviant resposta afirmativa al missatge rebut,
        // seguint el protocol fipa-request
        ACLMessage resposta = new ACLMessage(ACLMessage.AGREE);
        resposta.setProtocol("fipa-request");
        resposta.setSource(getLocalName());
        resposta.setLanguage("\Plain Text\");
        resposta.removeAllDests();
        resposta.addDest(msg.getSource());
        resposta.setContent(msg.getContent());
        send(resposta);
        ArrayList l=new ArrayList();
        l=graf.CercaCamins(ciutatO.toUpperCase(),ciutatD.toUpperCase(),temps);
        addBehaviour(new BehaviourDemanaInfo(l,max,msg.getSource()));
    } catch (Exception e ) {
        System.out.println(getLocalName()+ ": Error al contingut enviant REFUSE.");
        ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
        resposta.setSource(getLocalName());
        resposta.setLanguage("\Plain Text\");
        resposta.removeAllDests();
        resposta.addDest(msg.getSource());
        resposta.setContent("Error de format o de contingut al missatge."
            +"Format: ciutat_origen-ciutat_desti-dia/mes/any-hora:minut ");
        send(resposta);
        addBehaviour(new BehaviourRebreP());
    }
}
}

/** El comportament "DemanaInfo" és l'encarregat de demanar els
    horaris dels camins trobats, i trobar les rutes possibles */
class BehaviourDemanaInfo extends SimpleBehaviour
{
    private ArrayList camins,cami;
    private String origen,ciutatO,missatge;
    private int state=1,numCami=0,numNode=0;
    private boolean finalitza=false;
    private fill nodeD=null;
    private ACLMessage resposta=null,resposta2=null;
    private ArrayList ruta= new ArrayList(),rutes= new ArrayList();
    MessageTemplate patronAgreeORefuse,patronInformOFailure;
    Calendar millorTemps=null,tempsAct=null,tempsMig=null,max=null;

    public BehaviourDemanaInfo(ArrayList l,Calendar maxim,String origen)
    {
        if (AgentTrens==null) AgentTrens =BuscaServei(NomServeiTrens);
        if (AgentAvions==null) AgentAvions=BuscaServei(NomServeiAvions);
        if (AgentTrens==null && AgentAvions==null)
        {
            ACLMessage resposta = new ACLMessage(ACLMessage.FAILURE);
            resposta.setSource(getLocalName());
            resposta.setLanguage("\Plain Text\");
            resposta.removeAllDests();
            resposta.addDest(origen);
            resposta.setContent("No s'ha trobat agent encarregat del transport en trens
ni avions.");
            send(resposta);
            System.out.println(getLocalName()+ ": no s'ha trobat cap agent de
transport.");
            state=0;
            finalitza=true;
        } else {
            max=maxim;
            camins=l;
            this.origen=origen;
            MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
            MessageTemplate mt2 = MessageTemplate.MatchType("inform");
            MessageTemplate mt3 = MessageTemplate.MatchType("failure");

```

```

        MessageTemplate mt4 = MessageTemplate.MatchType("refuse");
        MessageTemplate mt5 = MessageTemplate.MatchType("agree");
        MessageTemplate mt6 = MessageTemplate.or(mt5,mt4);
        MessageTemplate mt7 = MessageTemplate.or(mt3,mt2);
        patronAgreeORefuse = MessageTemplate.and(mt1,mt6);
        patronInformOfFailure = MessageTemplate.and(mt1,mt7);
        tempsMig= Calendar.getInstance();
        tempsAct= Calendar.getInstance();
        millorTemps= (Calendar)max.clone();
        // Agafem el primer camí
        cami =(ArrayList) camins.get(0);
        nodeD =(fill) cami.get(0);
    }
}

public void SeguentCami()
{
    numCami++;
    if(numCami < camins.size())
    {
        numNode=1;
        tempsMig= Calendar.getInstance();
        tempsAct= Calendar.getInstance();
        millorTemps= (Calendar)max.clone(); // Inicialitzem el millor temps al temps
màxim
        cami=(ArrayList) camins.get(numCami);
        ciutat0=((fill) cami.get(0)).getCiutat();
        System.out.print("(" + (100*numCami)/camins.size()+"%");
    } else finalitza=true;
}

public moviment TractaInfo()
{
    int i1=0,i2=2,i3;
    String hs,ms,ha,ma,preu,info,p,nomc;
    moviment m=null;

    p = resposta2.getContent();
    if (p.indexOf('$')!=-1) { // Si hi ha algun camí tractem la info.
        do {
            i1 = p.indexOf(' ',i2);
            nomc= p.substring(i2,i1); // nom companyia
            do {
                i1++;
                i2 = p.indexOf(' ',i1); i3 = p.indexOf(' ',i2+1);
                // id. tren, id. recorregut
                info=nomc.concat(" "+p.substring(i1,i3)); i1 = p.indexOf(':',i3+1);
                hs = p.substring(i3+1,i1); // hora sortida
                i2 = p.indexOf(':',i1+1);
                ms = p.substring(i1+1,i2); // minut "
                i2 = p.indexOf(' ',i2); // no considerem els segons
                i1 = p.indexOf(':',i2+1);
                ha = p.substring(i2+1,i1); // hora arribada
                i2 = p.indexOf(':',i1+1);
                ma = p.substring(i1+1,i2); // minut "
                i2 = p.indexOf(' ',i2); // no considerem els segons
                i1 = p.indexOf('-',i2);
                preu = p.substring(i2+1,i1); // preu
                i2=i1;
                // Mirem si podem agafar avui aquest transport segons el temps
                // actual del camí sino augmentem el dia del mes
                tempsMig=(Calendar) tempsAct.clone();
                tempsMig.set(Calendar.HOUR_OF_DAY,Integer.parseInt(hs));
                tempsMig.set(Calendar.MINUTE,Integer.parseInt(ms));
                tempsMig.set(Calendar.SECOND, 0);
                if (tempsMig.before(tempsAct) || Integer.parseInt(ha) <
Integer.parseInt(hs))
                    tempsMig.add(Calendar.DAY_OF_MONTH,1);
                tempsMig.set(Calendar.HOUR_OF_DAY,Integer.parseInt(ha));
                tempsMig.set(Calendar.MINUTE,Integer.parseInt(ma));
                if (tempsMig.before(millorTemps))
                {
                    millorTemps=(Calendar) tempsMig.clone();
                    m=new
                    moviment(ciutat0,nodeD.getCiutat(), tempsMig.get(Calendar.DAY_OF_MONTH),
                        hs,ms,ha,ma,preu,info);
                }
            }
        }
    }
}

```

```

        } while ((i1+1)!=p.length() && (p.charAt(i1+1)!='$'));
        i2=i2+2;
    } while ((i1+1)!=p.length());
}
return m;
}

public void EnviaResultats ()
{
    ACLMessage informa = new ACLMessage(ACLMessage.INFORM);
    String resultats=new String(" ");
    long millorPreu= Long.MAX_VALUE,preuRuta;
    Calendar tempsArribada= Calendar.getInstance(),millorTemps= (Calendar)
max.clone();
    moviment m=null;
    int j,k,i=1;

    // recorrem la llista de rutes
    for (k=0;k<rutes.size();k++)
    {
        String mis= new String("-RUTA "+i+": ");
        ruta=(ArrayList) rutes.get(k);
        preuRuta=0;
        // posem la informació de la ruta en un string
        for (j=0;j<ruta.size();j++)
        {
            m=(moviment) ruta.get(j);
            mis=mis.concat("-"+m.toString());
            preuRuta=preuRuta + m.getPreu();
        }
        tempsArribada.add(Calendar.DAY_OF_MONTH,m.getDia());
        tempsArribada.set(Calendar.HOUR_OF_DAY, Integer.parseInt(m.getHoraA()));
        tempsArribada.set(Calendar.MINUTE, Integer.parseInt(m.getMinA()));
        tempsArribada.set(Calendar.SECOND,0);
        if (preuRuta <= millorPreu || !millorTemps.before(tempsArribada))
        {
            if (preuRuta< millorPreu) millorPreu= preuRuta;
            if (!millorTemps.before(tempsArribada)) millorTemps= (Calendar)
tempsArribada.clone();
            resultats=resultats.concat(mis);
            i++;
        }
    }
    System.out.println(resultats);
    informa.removeAllDests();
    informa.addDest(origen);
    informa.setSource(getLocalName());
    informa.setLanguage("\Plain Text\");
    informa.setContent(resultats);
    informa.setProtocol("fipa-request");
    send(informa);
}

public void action()
{
    try {
        switch (state) {
            case 1:
                // Construeix el següent missatge
                numNode++;
                // Si arribem a l'últim tram del camí
                // actual seguim amb el següent camí
                if (numNode >= cami.size())
                {
                    SeguentCami();
                } else ciutat0= nodeD.getCiutat();
                if (!finalitza)
                {
                    nodeD= (fill) cami.get(numNode);
                    missatge= new String(ciutat0+"-"+nodeD.getCiutat());
                    state++;
                }
                break;
            case 2:
                // Envia request i espera Agree
                ACLMessage peticio = new ACLMessage(ACLMessage.REQUEST);
                peticio.removeAllDests();

```

```

        peticio.setSource(getLocalName());
        peticio.setLanguage("\Plain Text\");
        peticio.setContent(missatge);
        peticio.setProtocol("fipa-request");
        if ((nodeD.getMedi()).equals("tren") && AgentTrens!=null)
peticio.addDest(AgentTrens);
        else if ((nodeD.getMedi()).equals("avion") && AgentAvions!=null)
peticio.addDest(AgentAvions);
        else {
            state=1;
            numNode=cami.size();
            ruta.clear();
            System.out.println("Error medi: *"+nodeD.getMedi()+"");
        }

        if (state!=1)
        {
            send(peticio);
            resposta= blockingReceive(patronAgreeORefuse);
            if ((resposta.getType()).equals("REFUSE")) {
                resposta.removeAllDests();
                resposta.addDest(origen);
                send(resposta);
                finalitza=true;
            } else state++;
        }
        break;
    case 3:
        resposta2=blockingReceive(patronInformOFailure);
        if ((resposta.getType()).equals("FAILURE"))
        {
            resposta.removeAllDests();
            resposta.addDest(origen);
            send(resposta2);
            finalitza=true;
        } else {
            // Tractem la informació rebuda
            moviment m=TractaInfo();

            if (m!=null) {
                ruta.add(m);

tempsAct.set(Calendar.HOUR_OF_DAY,Integer.parseInt(m.getHoraA()));
                tempsAct.set(Calendar.MINUTE,Integer.parseInt(m.getMinA()));

tempsAct.set(Calendar.DAY_OF_MONTH,millorTemps.get(Calendar.DAY_OF_MONTH));
                tempsAct.set(Calendar.SECOND,0);
                millorTemps= (Calendar) max.clone();
                if (numNode==cami.size()-1) // Si final ruta -> guarda ruta
en routes
                    {
                        routes.add(ruta.clone());
                        ruta.clear();
                    }
                } else { // No s'ha trobat cap transport per realitzar aquest
tram
                    // per tant no es pot acabar el camí actual, canviem de camí
                    state=1;
                    numNode=cami.size();
                    ruta.clear();
                }

                if ((numCami >= camins.size()-1) && (numNode >= cami.size()-1))
                {
                    System.out.println("(100%)");
                    EnviaResultats();
                    finalitza=true;
                } else state=1;
            }
        }
        break;
    default:
    }
} catch (Exception e ) {
    ACLMessage resposta = new ACLMessage(ACLMessage.FAILURE);
    resposta.setSource(getLocalName());
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(origen);

```

```

        resposta.setContent(getLocalName()+" : Estat -> "+state+". Exception ->
"+e.getMessage());
        send(resposta);
        System.out.println(getLocalName()+" : Estat -> "+state+". Exception ->
"+e.getMessage());
        finalitza=true;
    }
}

public boolean done() {
    if (finalitza) addBehaviour(new BehaviourRebreP());
    return finalitza;
}
}

// Mètode que registra l'agent al DF
public void RegistraAgent()
{
    AgentManagementOntology.DFAgentDescriptor dfd = new
AgentManagementOntology.DFAgentDescriptor();
    AgentManagementOntology.ServiceDescriptor sd = new
AgentManagementOntology.ServiceDescriptor();

    dfd.setName(getName());
    dfd.addAddress(getAddress());
    sd.setName("CoordinarRuta"); // descripció del servei
    dfd.addAgentService(sd);
    dfd.setType("AgentCoordinador");
    dfd.setOntology("fipa-agent-management");
    dfd.setOwnership("AlbertoRibes");
    dfd.setDFState("active");
    try {
        registerWithDF("DF", dfd);
    } catch (FIPAException e) {
        System.err.println(getLocalName()+" : error al registrar l'agent al DF.");
        System.err.println("Rao: "+e.getMessage());
        doDelete();
    }
    System.out.println(getLocalName()+" : registre al DF realitzat amb exit.");
}

// Metode que busca un servei i retorna la direccio resultant
public String BuscaServei(String servei)
{
    AgentManagementOntology.DFAgentDescriptor dfd = new
AgentManagementOntology.DFAgentDescriptor();
    AgentManagementOntology.ServiceDescriptor sd = new
AgentManagementOntology.ServiceDescriptor();
    AgentManagementOntology.DFSearchResult agents = null;
    String nom= null;

    try {
        sd.setName(servei);
        dfd.addAgentService(sd); // afegim el tipus de servei
        // realitzem la petició de cerca al DF per defecte (nom="df")
        agents = searchDF("df", dfd, null);
        int i=0;
        Enumeration llista= agents.elements();
        while (llista.hasMoreElements())
        { i++;
            dfd=(AgentManagementOntology.DFAgentDescriptor) llista.nextElement();
            nom=dfd.getName();
            System.out.println(i+"->"+nom.substring(0,nom.indexOf('@',0)));
        }

        switch (i) {
            case 0: System.out.println(getLocalName()+" : No s'ha trobat cap agent.");
break;
            case 1: System.out.println(getLocalName()+" : Busqueda realitzada amb
exit."); break;
            default: System.out.println(getLocalName()+" : error->trobats més d'un
agent.");
                doDelete();
                break;
        }
    }
}

```

```

    } catch (Exception e) {
        System.err.println(getLocalName()+" : error en la busqueda d'agents al DF.");
        System.err.println("Rao: "+e.getMessage());
        doDelete();
    }
    return nom;
}

/** L'agent Coordinador realitza les següents accions al setup:
    - Registra els seus serveis al DF
    - Inicialitza l'objecte "Graf" per mitjà del fitxer "ciutats.txt"
    - Inicialitza les variables "AgentTrens" i ... amb les direccions dels
      agents que donen uns serveis determinats.
    - Afegeix el comportament "RebreP".

    Si qualsevol de les accions anteriors no es realitza correctament
    l'agent visualitzarà el corresponent missatge d'error i terminarà.
*/
protected void setup ()
{
    RegistraAgent();
    System.out.println(getLocalName()+ " : inicialitzant graf.");
    try {
        graf.Inicialitza("ciutats.txt");
    } catch (Exception e) {
        System.err.println("Rao: "+e.getMessage());
        doDelete();
    }

    System.out.println(getLocalName()+" : Buscant agents dels medis de transport.");
    AgentTrens =BuscaServei(NomServeiTrens);
    AgentAvions=BuscaServei(NomServeiAvions);

    // Afegim els seus comportaments
    System.out.println(getLocalName()+ " : afegint comportaments.");
    addBehaviour(new BehaviourRebreP());
}
}

```

7.6. La classe AgentTrens.

```

/*****
* @Nom Fitxer: AgentTrens.java
* @Data: 13-7-2000
* @Fet per: Alberto Ribes García
* @Descripció: definició de la classe "AgentTrens" que consisteix
* en un agent encarregat d'interaccionar amb els agents
* que agentifiquen les Bases de Dades de trens.
*****/

// Importem les classes de Jade:
import jade.core.*; // - Totes les classes del nucli (agent,...)
import jade.core.behaviours.*; // - Tots els comportaments (Behaviours,...)
import jade.domain.*; // - Elements de gestió (AMS,DF,ACC,...)
import jade.lang.acl.*; // - Llenguatge de comunicació ACL
// Altres:
import java.io.*; // - Entrada / Sortida de dades
import java.util.*;
import java.util.Hashtable;

public class AgentTrens extends Agent
{
    /** Guarda informació de les consultes fetes, abans de demanar el servei
    d'informació d'un agent de base de dades mirarà si es troba la info.
    en aquesta taula.
    */
    private int maxRepes=200;
    private Hashtable repes= new Hashtable();
    private AgentGroup grupAgents= new AgentGroup();
    private String nomServei= "consultarBDTrens";
}

```

```

/** Definim el comportament "Rebre", que només rebrà missatges:
- Tipus          -> REQUEST
- Protocol       -> FIPA-REQUEST
- Format del contingut -> ciutat_origen-ciutat_desti
*/
class BehaviourRebreP extends CyclicBehaviour
{
    MessageTemplate patron;

    public BehaviourRebreP() {
        System.out.println(getLocalName()+" : esperant missatge.");
        MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
        MessageTemplate mt2 = MessageTemplate.MatchType("request");
        patron = MessageTemplate.and(mt1,mt2);
    }

    public void action()
    {
        int index1 =0;
        int index2;
        char separador='-';

        ACLMessage msg = receive(patron);    // Esperem a rebre el missatge
        if (msg==null) block();
        else {
            try {
                String Content=msg.getContent();

                // Agafem cada camp del contingut del missatge
                index2 = Content.indexOf(separador,index1);
                if (index2==-1) throw new Exception();
                String ciutat0 = Content.substring(index1,index2);
                String ciutatD = Content.substring(index2+1);

                if (ciutat0.length()==0 || ciutatD.length()==0) throw new Exception();
                // Enviant resposta afirmativa al missatge rebut,
                // seguint el protocol fipa-request
                ACLMessage resposta = new ACLMessage(ACLMessage.AGREE);
                resposta.setProtocol("fipa-request");
                resposta.setSource(getLocalName());
                resposta.setLanguage("\Plain Text\");
                resposta.removeAllDests();
                resposta.addDest(msg.getSource());
                resposta.setContent(msg.getContent());
                send(resposta);

                // Si no és una consulta repetida
                if (!repes.containsKey(msg.getContent()))
                {
                    // Si no està inicialitzada la variable "grupAgents"
                    if (grupAgents.size()==0) {
                        // la inicialitzem amb el comportament ...
                        addBehaviour(new
BehaviourBuscarServei(msg.getSource(),msg.getContent()));
                    } else {
                        // si ja està inicialitzada enviem la consulta a tots els agents
                        ACLMessage peticio = new ACLMessage(ACLMessage.REQUEST);
                        peticio.removeAllDests();
                        peticio.setSource(getLocalName());
                        peticio.setLanguage("\Plain Text\");
                        peticio.setContent(msg.getContent());
                        peticio.setProtocol("fipa-request");
                        send(peticio,grupAgents);
                        // Afegim el comportament que rebrà els missatges amb els resultats
                        // dels diferents agents que agentifiquen les bases de dades
                        addBehaviour(new
BehaviourRebreInfo(msg.getSource(),msg.getContent()));
                    }
                } else {
                    /** Enviam els resultats*/
                    resposta = new ACLMessage(ACLMessage.INFORM);
                    resposta.setProtocol("fipa-request");
                    resposta.setSource(getLocalName());
                    resposta.setLanguage("\Plain Text\");
                    resposta.removeAllDests();
                    resposta.addDest(msg.getSource());
                    resposta.setContent((String)repes.get(msg.getContent()));
                }
            }
        }
    }
}

```

```

        send(resposta);
    }
} catch (Exception e ) {
    System.out.println("Error al contingut, enviant resposta ...");
    ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
    resposta.setSource(getLocalName());
    resposta.setProtocol("fipa-request");
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(msg.getSource());
    resposta.setContent("Error del format al contingut del missatge."
        +"Format: ciutat_origen-ciutat_desti. Excepcio: "+e.getMessage());
    send(resposta);
}
}
}

/** Definim el comportament "BuscarServei" que s'encarregarà de realitzar
la busqueda de tots els agents que donin un determinat servei de
consulta sobre una base de dades d'un medi de transport determinat,
aquest servei se li passarà al constructor de la classe, un cop trobats
els agents s'enviarà el mateix missatge a tots. */

class BehaviourBuscarServei extends OneShotBehaviour
{
    private String direccio_font;
    private String consult;
    // Obj. que ens permetrà agafar els resultats de la cerca al DF
    private AgentManagementOntology.DFSearchResult agents = null;
    // Obj. per determinar el tipus d'agent que volem cercar al DF
    private AgentManagementOntology.DFAgentDescriptor dfd = new
        AgentManagementOntology.DFAgentDescriptor();
    private AgentManagementOntology.ServiceDescriptor sd = new
AgentManagementOntology.ServiceDescriptor();
    String nom;

    public BehaviourBuscarServei(String font, String consulta) {
        direccio_font=font;
        consult= consulta;
    }

    public void action()
    {
        try {
            sd.setName(nomServei);
            dfd.addAgentService(sd);      // afegim el tipus de servei

            // realitzem la petició de cerca al DF per defecte (nom="df")
            agents = searchDF("df", dfd, null);

            // Posem tots els agents en un grup
            Enumeration llista= agents.elements();
            while (llista.hasMoreElements())
            {
                dfd=(AgentManagementOntology.DFAgentDescriptor)
                    llista.nextElement();
                nom=dfd.getName();
                grupAgents.addMember(nom);
            }

            /** Si hi ha algun agent al grup anterior llavors enviem a tots
la petició de realització del servei, sino enviem missatge
per refusar la coordinació de la ruta
*/
            if (grupAgents.size(>0)
            {
                ACLMessage peticio = new ACLMessage(ACLMessage.REQUEST);
                peticio.removeAllDests();
                peticio.setSource(getLocalName());
                peticio.setLanguage("\Plain Text\");
                peticio.setContent(consult);
                peticio.setProtocol("fipa-request");
                send(peticio,grupAgents);
                // Afegim el comportament que rebrà els missatges amb els resultats
                // dels diferents agents que agentifiquen les bases de dades

```

```

        addBehaviour(new BehaviourRebreInfo(direccio_font,consult));
    } else throw new Exception();
} catch (Exception e ) {
    System.out.println("Error en fer la cerca d'agents AGBD ...");
    System.err.println("Rao: "+e.getMessage());
    ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
    resposta.setProtocol("fipa-request");
    resposta.setSource(getLocalName());
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(direccio_font);
    resposta.setContent("Error en la cerca, o no trobat cap agent "
        +"registrat a un DF que realitzi el servei "
        +"que necessitem. Excepcio: "+e.getMessage());
    send(resposta);
}
}
}

/** Comportament que intenta rebre els missatges de tipus "inform" dels
agents de les bases de dades, s'acaba quan s'hagin rebut tants missatges
com agents s'hagin trobat o hagi passat un "timeout" i no haguem rebut
tots els missatges esperats.*/

class BehaviourRebreInfo extends SimpleBehaviour
{
    private int informs=0,agrees=0,refuses=0,failures=0;
    private String resultats= new String(" ");
    private String d_font,clau;
    private boolean finalitza=false;

    MessageTemplate patron;

    public BehaviourRebreInfo(String font,String clau) {
        MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
        MessageTemplate mt2 = MessageTemplate.MatchType("inform");
        MessageTemplate mt3 = MessageTemplate.MatchType("agree");
        MessageTemplate mt4 = MessageTemplate.MatchType("refuse");
        MessageTemplate mt5 = MessageTemplate.MatchType("failure");
        MessageTemplate mt6 = MessageTemplate.or(mt2,mt3);
        MessageTemplate mt7 = MessageTemplate.or(mt4,mt5);
        MessageTemplate mt8 = MessageTemplate.or(mt6,mt7);
        patron = MessageTemplate.and(mt8,mt1);
        d_font=font;
        this.clau=clau;
    }

    public void action ()
    {
        /** Esperem a rebre el missatge */
        ACLMessage msg = receive(patron);
        if (msg == null) block();
        else {
            try {
                String tipus= msg.getType();
                if (tipus.equals("AGREE")) agrees++;
                else if (tipus.equals("REFUSE")) refuses++;
                else if (tipus.equals("INFORM"))
                {
                    informs++;
                    String info= msg.getContent();
                    if ((1+info.indexOf(' '))!=info.length())
                        resultats=resultats.concat("$"+info);
                    if (repes.size() > maxRepes) repes.clear();
                    repes.put(clau,resultats);
                } else if (tipus.equals("FAILURE")) failures++;

                if ((agrees+refuses)==grupAgents.size() && (informs+failures) ==
                    grupAgents.size())
                {
                    /** Enviem els resultats de la consulta */
                    ACLMessage resposta = new ACLMessage(ACLMessage.INFORM);
                    resposta.setProtocol("fipa-request");
                    resposta.setSource(getLocalName());
                    resposta.setLanguage("\Plain Text\");
                    resposta.removeAllDests();
                    resposta.addDest(d_font);
                }
            }
        }
    }
}

```

```

        resposta.setContent(resultats);
        send(resposta);
        finalitza=true;
    }

    } catch (Exception e ) {
        System.out.println(getLocalName()+" Excepció -> "+e.getMessage());
        ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
        resposta.setProtocol("fipa-request");
        resposta.setSource(getLocalName());
        resposta.setLanguage("\Plain Text\");
        resposta.removeAllDests();
        resposta.addDest(d_font);
        resposta.setContent("Exception:"+e.getMessage());
        send(resposta);
        finalitza=true;
    }
}

}

}

public boolean done() {
    return finalitza;
}

}

protected void setup ()
{
    AgentManagementOntology.DFAgentDescriptor dfd = new
        AgentManagementOntology.DFAgentDescriptor();
    AgentManagementOntology.ServiceDescriptor sd = new
        AgentManagementOntology.ServiceDescriptor();

    // Registrem l'agent al DF
    dfd.setName(getName());
    dfd.addAddress(getAddress());
    // descripció del servei
    sd.setName("CoordinarTrens");
    //sd.setType("");
    //sd.setOntology("");
    //sd.setFixedProps("");
    //sd.setNegotiableProps("");
    //sd.setCommunicationProps("");
    dfd.addAgentService(sd);
    dfd.setType("AgentTransport");
    dfd.setOntology("fipa-agent-management");
    dfd.setOwnership("AlbertoRibes");
    dfd.setDFState("active");
    try {
        registerWithDF("DF", dfd);
    } catch (FIPAException e) {
        System.err.println(getLocalName()+" error al registrar l'agent al DF.");
        System.err.println("Rao: "+e.getMessage());
        doDelete();
    }
    System.out.println(getLocalName()+" : registre al DF realitzat amb exit.");

    // Afegim els seus comportaments
    System.out.println(getLocalName()+" : afegint comportaments.");
    addBehaviour(new BehaviourRebreP());
}
}
}

```

7.7. La classe AgentAvions.

```

/*****
 * @Nom Fitxer: AgentAvions.java
 * @Data:      31-8-2000
 * @Fet per:    Alberto Ribes García
 * @Descripció: definició de la classe "AgentAvions" que consisteix
 *              en un agent encarregat d'interaccionar amb els agents
 *              que agentifiquen les Bases de Dades de avions.
 *****/

// Importem les classes de Jade:
import jade.core.*; // - Totes les classes del nucli (agent,...)
import jade.core.behaviours.*; // - Tots els comportaments (Behaviours,...)
import jade.domain.*; // - Elements de gestió (AMS,DF,ACC,...)
import jade.lang.acl.*; // - Llenguatge de comunicació ACL
// Altres:
import java.io.*; // - Entrada / Sortida de dades
import java.util.*;
import java.util.Hashtable;

public class AgentAvions extends Agent
{
    /** Guarda informació de les consultes fetes, abans de demanar el servei
    d'informació d'un agent de base de dades mirarà si es troba la info.
    en aquesta taula.
    */
    private Hashtable repes= new Hashtable();
    private int maxRepes=200;
    private AgentGroup grupAgents= new AgentGroup();
    private String nomServei= "ConsultarBDAvions";

    /** Definim el comportament "Rebre", que només rebrà missatges:
    - Tipus -> REQUEST
    - Protocol -> FIPA-REQUEST
    - Format del contingut -> ciutat_origen-ciutat_desti
    */
    class BehaviourRebreP extends CyclicBehaviour
    {
        MessageTemplate patron;

        public BehaviourRebreP() {
            System.out.println(getLocalName()+" : esperant missatge.");
            MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
            MessageTemplate mt2 = MessageTemplate.MatchType("request");
            patron = MessageTemplate.and(mt1,mt2);
        }

        public void action()
        {
            int index1 =0;
            int index2;
            char separador='-';

            ACLMessage msg = receive(patron); // Esperem a rebre el missatge
            if (msg==null) block();
            else {
                try {
                    String Content=msg.getContent();

                    // Agafem cada camp del contingut del missatge
                    index2 = Content.indexOf(separador,index1);
                    if (index2==-1) throw new Exception();
                    String ciutat0 = Content.substring(index1,index2);
                    String ciutatD = Content.substring(index2+1);

                    if (ciutat0.length()==0 || ciutatD.length()==0) throw new Exception();
                    // Enviant resposta afirmativa al missatge rebut,
                    // seguint el protocol fipa-request
                    ACLMessage resposta = new ACLMessage(ACLMessage.AGREE);
                    resposta.setProtocol("fipa-request");
                    resposta.setSource(getLocalName());
                    resposta.setLanguage("\Plain Text\");
                    resposta.removeAllDests();
                }
            }
        }
    }
}
```

```

        resposta.addDest(msg.getSource());
        resposta.setContent(msg.getContent());
        send(resposta);

        // Si no és una consulta repetida
        if (!repes.containsKey(msg.getContent()))
        {
            // Si no està inicialitzada la variable "grupAgents"
            if (grupAgents.size()==0) {
                // la inicialitzem amb el comportament ...
                addBehaviour(new
BehaviourBuscarServei(msg.getSource(),msg.getContent()));
            } else {
                // si ja està inicialitzada enviem la consulta a tots els agents
                ACLMessage peticio = new ACLMessage(ACLMessage.REQUEST);
                peticio.removeAllDests();
                peticio.setSource(getLocalName());
                peticio.setLanguage("\Plain Text\");
                peticio.setContent(msg.getContent());
                peticio.setProtocol("fipa-request");
                send(peticio,grupAgents);
                // Afegim el comportament que rebrà els missatges amb els resultats
                // dels diferents agents que agentifiquen les bases de dades
                addBehaviour(new
BehaviourRebreInfo(msg.getSource(),msg.getContent()));
            }
        }
        } else {
            /** Enviem els resultats*/
            resposta = new ACLMessage(ACLMessage.INFORM);
            resposta.setProtocol("fipa-request");
            resposta.setSource(getLocalName());
            resposta.setLanguage("\Plain Text\");
            resposta.removeAllDests();
            resposta.addDest(msg.getSource());
            resposta.setContent((String)repes.get(msg.getContent()));
            send(resposta);
        }
    } catch (Exception e ) {
        System.out.println("Error al contingut, enviant resposta ...");
        ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
        resposta.setSource(getLocalName());
        resposta.setProtocol("fipa-request");
        resposta.setLanguage("\Plain Text\");
        resposta.removeAllDests();
        resposta.addDest(msg.getSource());
        resposta.setContent("Error del format al contingut del missatge."
            +"Format: ciutat_origen-ciutat_desti. Excepcio: "+e.getMessage());
        send(resposta);
    }
}
}
}
}

```

/** Definim el comportament "BuscarServei" que s'encarregarà de realitzar la busqueda de tots els agents que donin un determinat servei de consulta sobre una base de dades d'un medi de transport determinat, aquest servei se li passarà al constructor de la classe, un cop trobats els agents s'enviarà el mateix missatge a tots. */

```

class BehaviourBuscarServei extends OneShotBehaviour
{
    private String direccio_font;
    private String consult;
    // Obj. que ens permetrà agafar els resultats de la cerca al DF
    private AgentManagementOntology.DFSearchResult agents = null;
    // Obj. per determinar el tipus d'agent que volem cercar al DF
    private AgentManagementOntology.DFAgentDescriptor dfd = new
        AgentManagementOntology.DFAgentDescriptor();
    private AgentManagementOntology.ServiceDescriptor sd = new
        AgentManagementOntology.ServiceDescriptor();
    String nom;

    public BehaviourBuscarServei(String font, String consulta) {
        direccio_font=font;
        consult= consulta;
    }
}

```

```

public void action()
{
    try {
        sd.setName(nomServei);
        dfd.addAgentService(sd);        // afegim el tipus de servei

        // realitzem la petició de cerca al DF per defecte (nom="df")
        agents = searchDF("df", dfd, null);

        // Posem tots els agents en un grup
        Enumeration llista= agents.elements();
        while (llista.hasMoreElements())
        {
            dfd=(AgentManagementOntology.DFAgentDescriptor)
            llista.nextElement();
            nom=dfd.getName();
            grupAgents.addMember(nom);
        }

        /** Si hi ha algun agent al grup anterior llavors enviem a tots
            la petició de realització del servei, sino enviem missatge
            per refusar la coordinació de la ruta
        */
        if (grupAgents.size()>0)
        {
            ACLMessage peticio = new ACLMessage(ACLMessage.REQUEST);
            peticio.removeAllDests();
            peticio.setSource(getLocalName());
            peticio.setLanguage("\Plain Text\");
            peticio.setContent(consult);
            peticio.setProtocol("fipa-request");
            send(peticio,grupAgents);
            // Afegim el comportament que rebrà els missatges amb els resultats
            // dels diferents agents que agentifiquen les bases de dades
            addBehaviour(new BehaviourRebreInfo(direccio_font,consult));
        } else throw new Exception();
    } catch (Exception e ) {
        System.out.println("Error en fer la cerca d'agents AGBD ...");
        System.err.println("Rao: "+e.getMessage());
        ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
        resposta.setProtocol("fipa-request");
        resposta.setSource(getLocalName());
        resposta.setLanguage("\Plain Text\");
        resposta.removeAllDests();
        resposta.addDest(direccio_font);
        resposta.setContent("Error en la cerca, o no trobat cap agent "
            +"registrat que realitzi el servei que necessitem. Excepcio: "
            +"e.getMessage());
        send(resposta);
    }
}

/** Comportament que intenta rebre els missatges de tipus "inform" dels
    agents de les bases de dades, s'acaba quan s'hagin rebut tants missatges
    com agents s'hagin trobat o hagi passat un "timeout" i no haguem rebut
    tots els missatges esperats.*/

class BehaviourRebreInfo extends SimpleBehaviour
{
    private int informs=0,agrees=0,refuses=0,failures=0;
    private String resultats= new String(" ");
    private String d_font,clau;
    private boolean finalitza=false;

    MessageTemplate patron;

    public BehaviourRebreInfo(String font,String clau) {
        MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
        MessageTemplate mt2 = MessageTemplate.MatchType("inform");
        MessageTemplate mt3 = MessageTemplate.MatchType("agree");
        MessageTemplate mt4 = MessageTemplate.MatchType("refuse");
        MessageTemplate mt5 = MessageTemplate.MatchType("failure");
        MessageTemplate mt6 = MessageTemplate.or(mt2,mt3);
        MessageTemplate mt7 = MessageTemplate.or(mt4,mt5);
        MessageTemplate mt8 = MessageTemplate.or(mt6,mt7);
    }
}

```

```

        patron = MessageTemplate.and(mt8,mt1);
        d_font=font;
        this.clau=clau;
    }

    public void action ()
    {
        /** Esperem a rebre el missatge */
        ACLMessage msg = receive(patron);
        if (msg == null) block();
        else {
            try {
                String tipus= msg.getType();
                if (tipus.equals("AGREE")) agrees++;
                else if (tipus.equals("REFUSE")) refuses++;
                else if (tipus.equals("INFORM"))
                {
                    informs++;
                    String info= msg.getContent();
                    if ((1+info.indexOf(' '))!=info.length())
                    resultats=resultats.concat("$"+info);
                } else if (tipus.equals("FAILURE")) failures++;

                if ((agrees+refuses)==grupAgents.size() && (informs+failures) ==
                    grupAgents.size())
                {
                    if (repes.size() > maxRepes) repes.clear();
                    repes.put(clau,resultats);
                    /** Enviem els resultats de la consulta */
                    ACLMessage resposta = new ACLMessage(ACLMessage.INFORM);
                    resposta.setProtocol("fipa-request");
                    resposta.setSource(getLocalName());
                    resposta.setLanguage("\Plain Text\");
                    resposta.removeAllDests();
                    resposta.addDest(d_font);
                    resposta.setContent(resultats);
                    send(resposta);
                    finalitza=true;
                }
            } catch (Exception e ) {
                System.out.println(getLocalName()+" Exception: "+e.getMessage());
                ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
                resposta.setProtocol("fipa-request");
                resposta.setSource(getLocalName());
                resposta.setLanguage("\Plain Text\");
                resposta.removeAllDests();
                resposta.addDest(d_font);
                resposta.setContent("Exception:"+e.getMessage());
                send(resposta);
                finalitza=true;
            }
        }
    }

    public boolean done() {
        return finalitza;
    }
}

protected void setup ()
{
    AgentManagementOntology.DFAgentDescriptor dfd = new
        AgentManagementOntology.DFAgentDescriptor();
    AgentManagementOntology.ServiceDescriptor sd = new
        AgentManagementOntology.ServiceDescriptor();

    // Registrem l'agent al DF
    dfd.setName(getName());
    dfd.addAddress(getAddress());
    // descripció del servei
    sd.setName("CoordinarAvions");
    //sd.setType("");
    //sd.setOntology("");
    //sd.setFixedProps("");
    //sd.setNegotiableProps("");
    //sd.setCommunicationProps("");
    dfd.addAgentService(sd);
}

```

```

dfd.setType("AgentTransport");
dfd.setOntology("fipa-agent-management");
dfd.setOwnership("AlbertoRibes");
dfd.setDFState("active");
try {
    registerWithDF("DF",dfd);
} catch (FIPAException e) {
    System.err.println(getLocalName()+": error al registrar l'agent al DF.");
    System.err.println("Rao: "+e.getMessage());
    doDelete();
}
System.out.println(getLocalName()+ ": registre al DF realitzat amb exit.");

// Afegim els seus comportaments
System.out.println(getLocalName()+ ": afegint comportaments.");
addBehaviour(new BehaviourRebreP());
}
}

```

7.8. La classe AgentRenfe.

```

/*****
 * @Nom Fitxer: AgentRenfe.java
 * @Data:      8-7-2000
 * @Fet per:   Alberto Ribes García
 * @Descripció: definició de la classe "AgentRenfe" que consisteix en
 *              un agent encarregat d'agentificar la Base de Dades de
 *              RENFE.
 *****/

import jade.core.*;           // Importem les classes de Jade:
import jade.core.behaviours.*; // - Totes les classes del nucli (agent,...)
import jade.domain.*;         // - Elements de gestió (AMS,DF,ACC,...)
import jade.lang.acl.*;       // - Llenguatge de comunicació ACL
import java.sql.*;             // - Llenguatge SQL
import java.io.*;              // - Entrada / Sortida de dades
import java.util.*;

/**
 * Definició de la classe "AgentRenfe" que consisteix en
 * un agent encarregat d'agentificar la base de dades de Renfe.
 */

public class AgentRenfe extends Agent
{
    static String company="RENFE";
    static String login="login";
    static String password="password";
    static String url="jdbc:odbc:renfe";
    private Connection con;
    // Aquest objecte de la classe DfAgentDescriptor ens permet realitzar
    // qualsevol acció suportada per el DF de forma fàcil
    private AgentManagementOntology.DfAgentDescriptor dfd = new
        AgentManagementOntology.DfAgentDescriptor();

    /** Definim el comportament "Rebre", que només rebrà missatges:
     * - Tipus          -> REQUEST
     * - Protocol       -> FIPA-REQUEST
     * - Format del contingut -> ciutat_origen-ciutat_desti */

    class BehaviourRebre extends OneShotBehaviour
    {
        MessageTemplate patron;

        public BehaviourRebre() {
            MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
            MessageTemplate mt2 = MessageTemplate.MatchType("request");
            patron = MessageTemplate.and(mt1,mt2);
        }

        public void action()
        {

```

```

int index1 =0;
int index2;
char separador='-';

/** Esperem a rebre el missatge */
ACLMMessage msg = blockingReceive(patron);
try {
    String Content=msg.getContent();

    /** Agafem cada camp del contingut del missatge */
    index2 = Content.indexOf(separador,index1);
    if (index2==-1) throw new Exception();
    String ciutat0 = Content.substring(index1,index2);
    String ciutatD = Content.substring(1+index2);

    /** Comprovem que el format sigui l'indicat */
    if (ciutat0.length()==0 || ciutatD.length()==0) throw new Exception();

    /** Enviem el missatge d'acceptació de la consulta a l'emisor */
    ACLMessage resposta = new ACLMessage(ACLMMessage.AGREE);
    resposta.setProtocol("fipa-request");
    resposta.setSource(getLocalName());
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(msg.getSource());
    resposta.setContent(msg.getContent());
    send(resposta);
    /** Afegim el comportament "Consulta" */
    addBehaviour(new BehaviourConsulta(ciutat0,ciutatD,msg.getSource()));
} catch (Exception e ) {
    System.out.println("Error al contingut, enviant resposta ...");
    ACLMessage resposta = new ACLMessage(ACLMMessage.REFUSE);
    resposta.setSource(getLocalName());
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(msg.getSource());
    resposta.setContent("Error del format al contingut del missatge."
+"Format esperat: ciutat_origen-ciutat_desti");
    send(resposta);
    reset();
}
}
}
}

```

```

// Definim el comportament "Consulta"
class BehaviourConsulta extends OneShotBehaviour
{
    private String ciutat_origen;
    private String ciutat_desti;
    private String direccio_font;
    private ACLMessage resposta;

    // Funcio constructora d'elements de la classe
    public BehaviourConsulta (String cOrigen,String cDesti,String s)
    {
        ciutat_origen=cOrigen;
        ciutat_desti =cDesti;
        direccio_font=s;
    }

    // Accio associada al comportament, realitza la consulta a la DB
    public void action ()
    {
        Statement stmt;

        try {
            // INTENTEM: CONEXIÀ A LA BASE DE DADES
            con=DriverManager.getConnection(url,login,password);
            stmt=con.createStatement();

            // INTENTEM: CONSULTA
            // Primer trobem els recorreguts possibles
            ResultSet results=stmt.executeQuery(
                "SELECT A.*, P1.temps, P2.temps "
                +"FROM Assigna A, Paradas P1, Paradas P2 "
                +"WHERE P1.ciutat = '"+ciutat_origen+"' and "

```

```

        +"      P2.ciutat = '"+ciutat_desti+"' and "
        +"      P1.id_rec = P2.id_rec and "
        +"      A.id_rec = P1.id_rec and "
        +"      P1.temps < P2.temps");

// MOSTREM: RESULTATS
String resultats= new String(company+" ");
while (results.next())
{
    int    id_tren = results.getInt(1);
    int    id_rec  = results.getInt(2);
    Time   hora    = results.getTime(3);
    int    preu    = results.getInt(4);
    Time   temps1  = results.getTime(5);
    Time   temps2  = results.getTime(6);

    Time h_sortida = new Time(hora.getTime() + temps1.getTime());
    Time h_arribada = new Time(hora.getTime() + temps2.getTime());
    resultats=resultats.concat(id_tren+" "+id_rec+"
"+h_sortida.toString()+" "
                                +h_arribada.toString()+" "+preu+"-");
}
// TANQUEM LA CONEXIÓ i Statement
stmt.close();
con.close();
//CONSTRUIM RESPOSTA
resposta = new ACLMessage(ACLMessage.INFORM);
resposta.setContent(resultats);
} catch (SQLException e) {
    resposta = new ACLMessage(ACLMessage.FAILURE);
    System.out.println(getLocalName()+" Codi d'error "+e.getErrorCode()+"
SQLException -> "+e.getMessage());
    resposta.setContent(getLocalName()+" SQLException -> "+e.getMessage());
} catch (Exception e) {
    resposta = new ACLMessage(ACLMessage.FAILURE);
    System.out.println(getLocalName()+" excepcio -> "+e.getMessage());
    resposta.setContent(getLocalName()+" excepcio -> "+e.getMessage());
} finally {
    resposta.setSource(getLocalName());
    resposta.setProtocol("fipa-request");
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(direccio_font);
//ENVIEM RESPOSTA
send(resposta);
}

// Intentem tancar la connexió per si de cas s'ha produït
// un error abans de poder tancar-la.
try { con.close();} catch (SQLException e) {}
addBehaviour(new BehaviourRebre());
}

protected void setup ()
{
    AgentManagementOntology.ServiceDescriptor sd = new
    AgentManagementOntology.ServiceDescriptor();

    try {
        // INTENTEM: carregar el driver JDBC-ODBC
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (ClassNotFoundException e) {
        System.err.println(getLocalName()+" no s'ha pogut trobar la classe del
driver JDBC/ODBC.");
        System.out.println(getLocalName()+" esborrant agent.");
        doDelete();
    }

    // Registrem l'agent al DF
    dfd.setName(getName());
    dfd.addAddress(getAddress());
    // descripció del servei
    sd.setName("ConsultarBDTrens");
    dfd.addAgentService(sd);
    dfd.setType("AgentGestorBD");
    dfd.setOntology("fipa-agent-management");
}

```

```

        dfd.setOwnership("AlbertoRibes");
        dfd.setDFState("active");
        try {
            registerWithDF("DF", dfd);
        } catch (FIPAException e) {
            System.err.println(getLocalName()+" : error al registrar l'agent al DF.");
            System.err.println("Rao: "+e.getMessage());
            doDelete();
        }
        System.out.println(getLocalName()+" : registre al DF realitzat amb exit.");

        // Afegim els seus comportaments
        System.out.println(getLocalName()+" : afegint comportaments.");
        addBehaviour(new BehaviourRebre());
    }
}

```

7.9. La classe AgentRailRibes.

```

/*****
 * @Nom Fitxer: AgentRailRibes.java
 * @Data:      8-7-2000
 * @Fet per:   Alberto Ribes García
 * @Descripció: definició de la classe "AgentRailRibes" que consisteix
 *              en un agent encarregat d'agentificar la Base de Dades de
 *              RailRibes.
 *****/

import jade.core.*;           // Importem les classes de Jade:
import jade.core.behaviours.*; // - Totes les classes del nucli (agent,...)
import jade.domain.*;         // - Tots els comportaments (Behaviours,...)
import jade.lang.acl.*;       // - Elements de gestió (AMS,DF,ACC,...)
import jade.lang.acl.*;       // - Lenguatge de comunicació ACL
import java.sql.*;             // - Altres:
import java.io.*;              // - Lenguatge SQL
import java.util.*;            // - Entrada / Sortida de dades

/**
 * Definició de la classe "AgentRailRibes" que consisteix en
 * un agent encarregat d'agentificar la base de dades de Renfe.
 */

public class AgentRailRibes extends Agent
{
    static String company="RAILRIBES";
    static String login="login";
    static String password="password";
    static String url="jdbc:odbc:railRibes";
    private Connection con;
    // Aquest objecte de la classe DfAgentDescriptor ens permet realitzar
    // qualsevol acció suportada per el DF de forma fàcil
    private AgentManagementOntology.DfAgentDescriptor dfd = new
        AgentManagementOntology.DfAgentDescriptor();

    /** Definim el comportament "Rebre", que només rebrà missatges:
     * - Tipus          -> REQUEST
     * - Protocol       -> FIPA-REQUEST
     * - Format del contingut -> ciutat_origen-ciutat_desti */

    class BehaviourRebre extends OneShotBehaviour
    {
        MessageTemplate patron;

        public BehaviourRebre() {
            MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
            MessageTemplate mt2 = MessageTemplate.MatchType("request");
            patron = MessageTemplate.and(mt1,mt2);
        }

        public void action()
        {
            int index1 =0;

```

```

int index2;
char separador='-';

/** Esperem a rebre el missatge */
ACLMMessage msg = blockingReceive(patron);
try {
    String Content=msg.getContent();

    /** Agafem cada camp del contingut del missatge */
    index2 = Content.indexOf(separador,index1);
    if (index2==-1) throw new Exception();
    String ciutat0 = Content.substring(index1,index2);
    String ciutatD = Content.substring(1+index2);

    /** Comprovem que el format sigui l'indicat */
    if (ciutat0.length()==0 || ciutatD.length()==0) throw new Exception();

    /** Enviem el missatge d'acceptació de la consulta a l'emisor */
    ACLMessage resposta = new ACLMessage(ACLMessage.AGREE);
    resposta.setProtocol("fipa-request");
    resposta.setSource(getLocalName());
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(msg.getSource());
    resposta.setContent(msg.getContent());
    send(resposta);
    /** Afegim el comportament "Consulta" */
    addBehaviour(new BehaviourConsulta(ciutat0,ciutatD,msg.getSource()));
} catch (Exception e ) {
    System.out.println("Error al contingut, enviant resposta ...");
    ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
    resposta.setSource(getLocalName());
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(msg.getSource());
    resposta.setContent("Error del format al contingut del missatge."
        +"Format esperat: ciutat_origen-ciutat_desti");
    send(resposta);
    reset();
}
}
}

```

```

// Definim el comportament "Consulta"
class BehaviourConsulta extends OneShotBehaviour
{
    private String ciutat_origen;
    private String ciutat_desti;
    private String direccio_font;
    private ACLMessage resposta;

    // Funcio constructora d'elements de la classe
    public BehaviourConsulta (String cOrigen,String cDesti,String s)
    {
        ciutat_origen=cOrigen;
        ciutat_desti =cDesti;
        direccio_font=s;
    }

    // Accio associada al comportament, realitza la consulta a la DB
    public void action ()
    {
        Statement stmt;

        try {
            // INTENTEM: CONEXIÀ A LA BASE DE DADES
            con=DriverManager.getConnection(url, login,password);
            stmt=con.createStatement();

            // INTENTEM: CONSULTA
            // Primer trobem els recorreguts possibles
            ResultSet results=stmt.executeQuery(
                "SELECT A.*, P1.temps, P2.temps "
                +"FROM Assigna A, Paradas P1, Paradas P2 "
                +"WHERE P1.ciutat = '"+ciutat_origen+"' and "
                +" P2.ciutat = '"+ciutat_desti+"' and "

```

```

        +"      P1.id_rec = P2.id_rec and "
        +"      A.id_rec = P1.id_rec and "
        +"      P1.temps < P2.temps");

// MOSTREM: RESULTATS
String resultats= new String(company+" ");
while (results.next())
{
    int    id_tren = results.getInt(1);
    int    id_rec  = results.getInt(2);
    Time   hora    = results.getTime(3);
    int    preu    = results.getInt(4);
    Time   temps1  = results.getTime(5);
    Time   temps2  = results.getTime(6);

    Time h_sortida = new Time(hora.getTime() + temps1.getTime());
    Time h_arribada = new Time(hora.getTime() + temps2.getTime());
    resultats=resultats.concat(id_tren+" "+id_rec+"
"+h_sortida.toString()+" "
                                +h_arribada.toString()+" "+preu+"-");
}
// TANQUEM LA CONEXIÓ i Statement
stmt.close();
con.close();
//CONSTRUIM RESPOSTA
resposta = new ACLMessage(ACLMessage.INFORM);
resposta.setContent(resultats);
} catch (SQLException e) {
    resposta = new ACLMessage(ACLMessage.FAILURE);
    System.out.println(getLocalName()+": Codi d'error "+e.getErrorCode()+".
SQLException -> "+e.getMessage());
    resposta.setContent(getLocalName()+": SQLException -> "+e.getMessage());
} catch (Exception e) {
    resposta = new ACLMessage(ACLMessage.FAILURE);
    System.out.println(getLocalName()+": excepcio -> "+e.getMessage());
    resposta.setContent(getLocalName()+": excepcio -> "+e.getMessage());
} finally {
    resposta.setSource(getLocalName());
    resposta.setProtocol("fipa-request");
    resposta.setLanguage("\\"Plain Text\\"");
    resposta.removeAllDests();
    resposta.addDest(direccio_font);
//ENVIEM RESPOSTA
send(resposta);
}
// Intentem tancar la connexió per si de cas s'ha produït
// un error abans de poder tancar-la.
try { con.close(); } catch (SQLException e) { }
addBehaviour(new BehaviourRebre());
}
}

protected void setup ()
{
    AgentManagementOntology.ServiceDescriptor sd = new
    AgentManagementOntology.ServiceDescriptor();

    try {
        // INTENTEM: carregar el driver JDBC-ODBC
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (ClassNotFoundException e) {
        System.err.println(getLocalName()+": no s'ha pogut trobar la classe del
driver JDBC/ODBC.");
        System.out.println(getLocalName()+": esborrant agent.");
        doDelete();
    }

    // Registrem l'agent al DF
    dfd.setName(getName());
    dfd.addAddress(getAddress());
    // descripció del servei
    sd.setName("ConsultarBDTrens");
    dfd.addAgentService(sd);
    dfd.setType("AgentGestorBD");
    dfd.setOntology("fipa-agent-management");
    dfd.setOwnership("AlbertoRibes");
    dfd.setDFState("active");
}

```

```

    try {
        registerWithDF("DF",dfd);
    } catch (FIPAException e) {
        System.err.println(getLocalName()+" : error al registrar l'agent al DF.");
        System.err.println("Rao: "+e.getMessage());
        doDelete();
    }
    System.out.println(getLocalName()+ " : registre al DF realitzat amb exit.");

    // Afegim els seus comportaments
    System.out.println(getLocalName()+ " : afegint comportaments.");
    addBehaviour(new BehaviourRebre());
}
}
}

```

7.10. La classe AgentIberia.

```

/*****
 * @Nom Fitxer: AgentIberia.java
 * @Data:      31-8-2000
 * @Fet per:   Alberto Ribes García
 * @Descripció: definició de la classe "AgentIberia" que consisteix en
 *              un agent encarregat d'agentificar la Base de Dades de
 *              IBERIA.
 *****/

import jade.core.*;           // Importem les classes de Jade:
import jade.core.behaviours.*; // - Totes les classes del nucli (agent,...)
import jade.domain.*;         // - Elements de gestió (AMS,DF,ACC,...)
import jade.lang.acl.*;       // - Llenguatge de comunicació ACL
                               // - Altres:
import java.sql.*;             // - Llenguatge SQL
import java.io.*;              // - Entrada / Sortida de dades
import java.util.*;

/**
 * Definició de la classe "AgentRenfe" que consisteix en
 * un agent encarregat d'agentificar la base de dades de Renfe.
 */

public class AgentIberia extends Agent
{
    static String company="IBERIA";
    static String login="login";
    static String password="password";
    static String url="jdbc:odbc:iberia";
    private Connection con;
    // Aquest objecte de la classe DfAgentDescriptor ens permet realitzar
    // qualsevol acció suportada per el DF de forma fàcil
    private AgentManagementOntology.DfAgentDescriptor dfd = new
        AgentManagementOntology.DfAgentDescriptor();

    /** Definim el comportament "Rebre", que només rebrà missatges:
     * - Tipus          -> REQUEST
     * - Protocol        -> FIPA-REQUEST
     * - Format del contingut -> ciutat_origen-ciutat_desti */

    class BehaviourRebre extends OneShotBehaviour
    {
        MessageTemplate patron;

        public BehaviourRebre() {
            MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
            MessageTemplate mt2 = MessageTemplate.MatchType("request");
            patron = MessageTemplate.and(mt1,mt2);
        }

        public void action()
        {
            int index1 =0;
            int index2;
            char separador='-';

```

```

/** Esperem a rebre el missatge */
ACLMMessage msg = blockingReceive(patron);
try {
    String Content=msg.getContent();

    /** Agafem cada camp del contingut del missatge */
    index2 = Content.indexOf(separador,index1);
    if (index2==-1) throw new Exception();
    String ciutat0 = Content.substring(index1,index2);
    String ciutatD = Content.substring(1+index2);

    /** Comprovem que el format sigui l'indicat */
    if (ciutat0.length()==0 || ciutatD.length()==0) throw new Exception();

    /** Enviem el missatge d'acceptació de la consulta a l'emisor */
    ACLMessage resposta = new ACLMessage(ACLMMessage.AGREE);
    resposta.setProtocol("fipa-request");
    resposta.setSource(getLocalName());
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(msg.getSource());
    resposta.setContent(msg.getContent());
    send(resposta);
    /** Afegim el comportament "Consulta" */
    addBehaviour(new BehaviourConsulta(ciutat0,ciutatD,msg.getSource()));
} catch (Exception e ) {
    System.out.println("Error al contingut, enviant resposta ...");
    ACLMessage resposta = new ACLMessage(ACLMMessage.REFUSE);
    resposta.setSource(getLocalName());
    resposta.setLanguage("\Plain Text\");
    resposta.removeAllDests();
    resposta.addDest(msg.getSource());
    resposta.setContent("Error del format al contingut del missatge."
        +"Format esperat: ciutat_origen-ciutat_desti");
    send(resposta);
    addBehaviour(new BehaviourRebre());
}
}
}

```

```

// Definim el comportament "Consulta"
class BehaviourConsulta extends OneShotBehaviour
{
    private String ciutat_origen,ciutat_desti,direccio_font;
    private ACLMessage resposta;

    // Funcio constructora d'elements de la classe
    public BehaviourConsulta (String cOrigen,String cDesti,String s)
    {
        ciutat_origen=cOrigen;
        ciutat_desti =cDesti;
        direccio_font=s;
    }

    // Accio associada al comportament, realitza la consulta a la DB
    public void action ()
    {
        Statement stmt;

        try {
            // INTENTEM: CONEXIÀ A LA BASE DE DADES
            con=DriverManager.getConnection(url, login,password);
            stmt=con.createStatement();

            // INTENTEM: CONSULTA
            // Primer trobem els recorreguts possibles
            ResultSet results=stmt.executeQuery(
                "SELECT A.*, P1.hora, P2.hora "
                +"FROM Assigna A, Parades P1, Parades P2 "
                +"WHERE P1.ciutat = '"+ciutat_origen+"' and "
                +" P2.ciutat = '"+ciutat_desti+"' and "
                +" P1.id_rec = P2.id_rec and "
                +" A.id_rec = P1.id_rec and "
                +" P1.hora < P2.hora");

```

```

// Guardem RESULTATS
String resultats= new String(company+" ");
while (results.next())
{
    int    id_avion= results.getInt(1);
    int    id_rec  = results.getInt(2);
    Time   hora    = results.getTime(3);
    long   preu     = results.getLong(4);
    Time   temps1  = results.getTime(5);
    Time   temps2  = results.getTime(6);

    Time h_sortida = new Time(hora.getTime() + temps1.getTime());
    Time h_arribada = new Time(hora.getTime() + temps2.getTime());
    resultats=resultats.concat(id_avion+" "+id_rec+"
+h_sortida.toString()+" "
                                +h_arribada.toString()+" "+preu+"-");
}
// TANQUEM LA CONEXIÓ i Statement
stmt.close();
con.close();
//CONSTRUIM RESPOSTA
resposta = new ACLMessage(ACLMessage.INFORM);
resposta.setContent(resultats);
} catch (SQLException e) {
    resposta = new ACLMessage(ACLMessage.FAILURE);
    System.out.println(getLocalName()+": Codi d'error "+e.getErrorCode()+".
SQLException -> "+e.getMessage());
    resposta.setContent(getLocalName()+": SQLException -> "+e.getMessage());
} catch (Exception e) {
    resposta = new ACLMessage(ACLMessage.FAILURE);
    System.out.println(getLocalName()+": excepcio -> "+e.getMessage());
    resposta.setContent(getLocalName()+": excepcio -> "+e.getMessage());
} finally {
    resposta.setSource(getLocalName());
    resposta.setProtocol("fipa-request");
    resposta.setLanguage("\\"Plain Text\\"");
    resposta.removeAllDests();
    resposta.addDest(direccio_font);
    //ENVIEM RESPOSTA
    send(resposta);
}
// TANQUEM LA CONEXIÓ
try { con.close(); } catch (SQLException e) { }
addBehaviour(new BehaviourRebre());
}
}

protected void setup ()
{
    AgentManagementOntology.ServiceDescriptor sd = new
    AgentManagementOntology.ServiceDescriptor();

    try {
        // INTENTEM: carregar el driver JDBC-ODBC
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (ClassNotFoundException e) {
        System.err.println(getLocalName()+": no s'ha pogut trobar la classe del
driver JDBC/ODBC.");
        System.out.println(getLocalName()+": esborrant agent.");
        doDelete();
    }

    // Registrem l'agent al DF
    dfd.setName(getName());
    dfd.addAddress(getAddress());
    // descripció del servei
    sd.setName("ConsultarBDAvions");
    dfd.addAgentService(sd);
    dfd.setType("AgentGestorBD");
    dfd.setOntology("fipa-agent-management");
    dfd.setOwnership("AlbertoRibes");
    dfd.setDFState("active");
    try {
        registerWithDF("DF", dfd);
    } catch (FIPAException e) {
        System.err.println(getLocalName()+": error al registrar l'agent al DF.");
        System.err.println("Rao: "+e.getMessage());
    }
}

```

```

        doDelete();
    }
    System.out.println(getLocalName()+ ": registre al DF realitzat amb exit.");

    // Afegim els seus comportaments
    System.out.println(getLocalName()+ ": afegint comportaments.");
    addBehaviour(new BehaviourRebre());
}
}

```

7.11. La classe AgentHispanAir.

```

/*****
 * @Nom Fitxer: AgentHispanAir.java
 * @Data: 31-8-2000
 * @Fet per: Alberto Ribes García
 * @Descripció: definició de la classe "AgentHispanAir" que consisteix en
 * un agent encarregat d'agentificar la Base de Dades de
 * HispanAir.
 *****/
// Importem les classes de Jade:
import jade.core.*; // - Totes les classes del nucli (agent,...)
import jade.core.behaviours.*; // - Tots els comportaments (Behaviours,...)
import jade.domain.*; // - Elements de gestió (AMS,DF,ACC,...)
import jade.lang.acl.*; // - Llenguatge de comunicació ACL
// Altres:
import java.sql.*; // - Llenguatge SQL
import java.io.*; // - Entrada / Sortida de dades
import java.util.*;
/**
 * Definició de la classe "AgentHispanAir" que consisteix en
 * un agent encarregat d'agentificar la base de dades de Renfe.
 */

public class AgentHispanAir extends Agent
{
    static String company="HISPANAIR";
    static String login="login";
    static String password="password";
    static String url="jdbc:odbc:HispanAir";
    private Connection con;
    // Aquest objecte de la classe DFAgentDescriptor ens permet realitzar
    // qualsevol acció suportada per el DF de forma fàcil
    private AgentManagementOntology.DFAgentDescriptor dfd = new
        AgentManagementOntology.DFAgentDescriptor();

    /** Definim el comportament "Rebre", que només rebrà missatges:
     * - Tipus -> REQUEST
     * - Protocol -> FIPA-REQUEST
     * - Format del contingut -> ciutat_origen-ciutat_desti */
    class BehaviourRebre extends OneShotBehaviour
    {
        MessageTemplate patron;

        public BehaviourRebre() {
            MessageTemplate mt1 = MessageTemplate.MatchProtocol("fipa-request");
            MessageTemplate mt2 = MessageTemplate.MatchType("request");
            patron = MessageTemplate.and(mt1,mt2);
        }

        public void action()
        {
            int index1=0;
            int index2;
            char separador='-';

            /** Esperem a rebre el missatge */
            ACLMessage msg = blockingReceive(patron);
            try {
                String Content=msg.getContent();

                /** Agafem cada camp del contingut del missatge */
                index2 = Content.indexOf(separador,index1);
                if (index2==-1) throw new Exception();
                String ciutat0 = Content.substring(index1,index2);
            }
        }
    }
}

```

```

String ciutatD = Content.substring(1+index2);

/** Comprovem que el format sigui l'indicat */
if (ciutatO.length()==0 || ciutatD.length()==0) throw new Exception();

/** Enviem el missatge d'acceptació de la consulta a l'emisor */
ACLMessage resposta = new ACLMessage(ACLMessage.AGREE);
resposta.setProtocol("fipa-request");
resposta.setSource(getLocalName());
resposta.setLanguage("\Plain Text\");
resposta.removeAllDests();
resposta.addDest(msg.getSource());
resposta.setContent(msg.getContent());
send(resposta);
/** Afegim el comportament "Consulta" */
addBehaviour(new BehaviourConsulta(ciutatO,ciutatD,msg.getSource()));
} catch (Exception e ) {
System.out.println("Error al contingut, enviant resposta ...");
ACLMessage resposta = new ACLMessage(ACLMessage.REFUSE);
resposta.setSource(getLocalName());
resposta.setLanguage("\Plain Text\");
resposta.removeAllDests();
resposta.addDest(msg.getSource());
resposta.setContent("Error del format al contingut del missatge."
+"Format esperat: ciutat_origen-ciutat_desti");
send(resposta);
addBehaviour(new BehaviourRebre());
}
}
}

```

```

// Definim el comportament "Consulta"
class BehaviourConsulta extends OneShotBehaviour
{
    private String ciutat_origen;
    private String ciutat_desti;
    private String direccio_font;
    private ACLMessage resposta;

    // Funcio constructora d'elements de la classe
    public BehaviourConsulta (String cOrigen,String cDesti,String s)
    {
        ciutat_origen=cOrigen;
        ciutat_desti =cDesti;
        direccio_font=s;
    }

    // Accio associada al comportament, realitza la consulta a la DB
    public void action ()
    {
        Statement stmt;

        try {
            // INTENTEM: CONEXIÀ A LA BASE DE DADES
            con=DriverManager.getConnection(url, login,password);
            stmt=con.createStatement();

            // INTENTEM: CONSULTA
            // Primer trobem els recorreguts possibles
            ResultSet results=stmt.executeQuery(
                "SELECT A.*, P1.hora, P2.hora "
                +"FROM Assigna A, Parades P1, Parades P2 "
                +"WHERE P1.ciutat = '"+ciutat_origen+"' and "
                +" P2.ciutat = '"+ciutat_desti+"' and "
                +" P1.id_rec = P2.id_rec and "
                +" A.id_rec = P1.id_rec and "
                +" P1.hora < P2.hora");

            // MOSTREM: RESULTATS
            String resultats= new String(company+" ");
            while (results.next())
            {
                int id_avion= results.getInt(1);
                int id_rec = results.getInt(2);
                Time hora = results.getTime(3);
                long preu = results.getLong(4);
            }
        }
    }
}

```

```

        Time temps1 = results.getTime(5);
        Time temps2 = results.getTime(6);

        Time h_sortida = new Time(hora.getTime() + temps1.getTime());
        Time h_arribada = new Time(hora.getTime() + temps2.getTime());
        resultats=resultats.concat(id_avion+" "+id_rec+"
"+h_sortida.toString()+" "
                                +h_arribada.toString()+" "+preu+"-");
    }
    // TANQUEM LA CONEXIÓ i Statement
    stmt.close();
    con.close();
    //CONSTRUIM RESPOSTA
    resposta = new ACLMessage(ACLMessage.INFORM);
    resposta.setContent(resultats);
} catch (SQLException e) {
    resposta = new ACLMessage(ACLMessage.FAILURE);
    System.out.println(getLocalName()+": Codi d'error "+e.getErrorCode()+".
SQLException -> "+e.getMessage());
    resposta.setContent(getLocalName()+": SQLException -> "+e.getMessage());
} catch (Exception e) {
    resposta = new ACLMessage(ACLMessage.FAILURE);
    System.out.println(getLocalName()+": excepcio -> "+e.getMessage());
    resposta.setContent(getLocalName()+": excepcio -> "+e.getMessage());
} finally {
    resposta.setSource(getLocalName());
    resposta.setProtocol("fipa-request");
    resposta.setLanguage("\\"Plain Text\\"");
    resposta.removeAllDests();
    resposta.addDest(direccio_font);
    //ENVIEM RESPOSTA
    send(resposta);
}
// TANQUEM LA CONEXIÓ
try { con.close(); } catch (SQLException e) {
    addBehaviour(new BehaviourRebre());
}
}

protected void setup ()
{
    AgentManagementOntology.ServiceDescriptor sd = new
    AgentManagementOntology.ServiceDescriptor();

    try {
        // INTENTEM: carregar el driver JDBC-ODBC
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (ClassNotFoundException e) {
        System.err.println(getLocalName()+": no s'ha pogut trobar la classe del
driver JDBC/ODBC.");
        System.out.println(getLocalName()+": esborrant agent.");
        doDelete();
    }
    // Registrem l'agent al DF
    dfd.setName(getName());
    dfd.addAddress(getAddress());
    // descripció del servei
    sd.setName("ConsultarBDAvions");
    dfd.addAgentService(sd);
    dfd.setType("AgentGestorBD");
    dfd.setOntology("fipa-agent-management");
    dfd.setOwnership("AlbertoRibes");
    dfd.setDFState("active");
    try {
        registerWithDF("DF", dfd);
    } catch (FIPAException e) {
        System.err.println(getLocalName()+": error al registrar l'agent al DF.");
        System.err.println("Rao: "+e.getMessage());
        doDelete();
    }
    System.out.println(getLocalName()+ ": registre al DF realitzat amb exit.");

    // Afegim els seus comportaments
    System.out.println(getLocalName()+ ": afegint comportaments.");
    addBehaviour(new BehaviourRebre());
}
}

```

8. JOC DE PROVES

Fitxer "ciutats.txt".

ALBACETE-	MURCIA	1:20	tren-	CUENCA	1:00	tren-CIUDADREAL	2:10	tren
ALICANTE-	VALENCIA	1:20	tren-	MURCIA	1:10	tren		
ALMERIA-	MURCIA	2:00	tren-	GRANADA	1:00	tren-MALAGA	1:20	tren
AVILA-	MADRID	1:00	tren-	SALAMANCA	1:10	tren		
BADAJOS-	CACERES	0:30	tren-	HUELVA	1:20	tren		
BARCELONA-	GIRONA	2:00	tren-	TARRAGONA	1:15	tren-MADRID	1:30	avion- VALENCIA
0:50 avion-	BILBAO	0:55	avion-	MALLORCA	1:20	avion		
BILBAO-	VITORIA	0:30	tren-	SANSEBASTIAN	0:40	tren-SANTANDER	0:45	tren-MADRID
1:40 avion-	BARCELONA	0:55	avion-	COMPOSTELA	0:55	avion		
BURGOS-	PALENCIA	0:50	tren-	SANTANDER	0:40	tren		
CACERES-	BADAJOS	0:30	tren-	MERIDA	0:30	tren-SALAMANCA	2:00	tren
CADIZ-	MALAGA	1:10	tren-	SEVILLA	1:00	tren		
CANARIAS-	TENERIFE	0:20	avion-	SEVILLA	1:30	avion		
CASTELLON-	VALENCIA	1:10	tren-	TARRAGONA	2:00	tren		
CIUDADREAL-	TOLEDO	0:50	tren-	CORDOVA	1:50	tren-ALBACETE	2:10	tren
COMPOSTELA-	PONTEVEDRA	0:35	tren-	CORUÑA	0:35	tren-LUGO	1:10	tren-MADRID
1:30 avion-	BILBAO	1:40	avion					
CORDOVA-	SEVILLA	1:10	tren-	JAEN	1:00	tren-CIUDADREAL	1:50	tren
CORUÑA-	COMPOSTELA	0:35	tren					
CUENCA-	VALENCIA	1:30	tren-	ALBACETE	1:00	tren-GUADALAJARA	1:00	tren
GIRONA-	BARCELONA	2:00	tren					
GRANADA-	JAEN	0:40	tren-	ALMERIA	0:50	tren		
GUADALAJARA-	MADRID	1:20	tren-	SORIA	1:10	tren-ZARAGOZA	2:00	tren-CUENCA
1:00 tren								
HUELVA-	SEVILLA	0:40	tren-	BADAJOS	1:10	tren		
HUESCA-	ZARAGOZA	0:50	tren					
JAEN-	CORDOVA	1:00	tren-	MALAGA	1:30	tren-GRANADA	0:40	tren
LEON-	OVIEDO	1:00	tren-	ZAMORA	1:10	tren-VALLADOLID	0:50	tren
LOGROÑO-	SORIA	0:40	tren-	VITORIA	0:40	tren		
LUGO-	COMPOSTELA	1:10	tren-	ORENSE	1:00	tren-OVIEDO	1:55	tren
LLEIDA-	TARRAGONA	1:00	tren-	ZARAGOZA	2:00	tren		
MADRID-	AVILA	1:00	tren-	SEGOVIA	1:30	tren-GUADALAJARA	1:20	tren- TOLEDO
1:00 tren-	COMPOSTELA	1:30	avion-	BILBAO	1:00	avion-BARCELONA	1:30	avion-
VALENCIA	1:20	avion-	SEVILLA	1:00	avion			
MALAGA-	CADIZ	1:10	tren-	JAEN	1:30	tren-ALMERIA	1:20	tren
MALLORCA-	BARCELONA	1:20	avion-	VALENCIA	1:30	avion		
MERIDA-	BADAJOS	0:40	tren-	TOLEDO	2:10	tren		
MURCIA-	ALICANTE	1:10	tren-	ALBACETE	1:20	tren-ALMERIA	2:00	tren
ORENSE-	LUGO	1:00	tren-	ZAMORA	1:00	tren		
OVIEDO-	LUGO	1:55	tren-	LEON	1:00	tren-SANTANDER	2:20	tren
PALENCIA-	VALLADOLID	0:30	tren-	BURGOS	0:50	tren		
PONTEVEDRA-	COMPOSTELA	0:35	tren					
SALAMANCA-	AVILA	1:30	tren-	ZAMORA	0:40	tren-CACERES	2:00	tren
SANSEBASTIAN-	BILBAO	0:40	tren-	VITORIA	0:30	tren		
SANTANDER-	OVIEDO	2:20	tren-	BURGOS	0:40	tren-BILBAO	0:45	tren
SEGOVIA-	VALLADOLID	0:40	tren-	MADRID	1:30	tren-SORIA	1:10	tren
SEVILLA-	HUELVA	0:40	tren-	CORDOVA	1:10	tren-CADIZ	1:00	tren-MADRID
1:00 avion-	TENERIFE	2:00	avion-	CANARIAS	2:10	avion		
SORIA-	LOGROÑO	0:40	tren-	ZARAGOZA	2:00	tren-GUADALAJARA	1:10	tren-SEGOVIA
1:10 tren								
TARRAGONA-	BARCELONA	1:15	tren-	LLEIDA	1:00	tren-CASTELLON	2:00	tren
TENERIFE-	SEVILLA	1:30	avion-	CANARIAS	0:20	avion		
TERUEL-	CASTELLON	1:40	tren-	ZARAGOZA	1:00	tren		
TOLEDO-	MERIDA	2:10	tren-	CIUDADREAL	0:50	tren-MADRID	1:00	tren
VALENCIA-	CASTELLON	1:10	tren-	ALICANTE	1:20	tren-BARCELONA	1:40	avion-MADRID
1:20 avion-	CUENCA	1:30	tren-	MALLORCA	1:30	avion-SEVILLA	1:50	avion
VALLADOLID-	PALENCIA	0:30	tren-	LEON	0:50	tren-SEGOVIA	0:40	tren
VITORIA-	LOGROÑO	0:40	tren-	BILBAO	0:30	tren-SANSEBASTIAN	0:30	tren
ZAMORA-	ORENSE	1:00	tren-	LEON	1:10	tren-SALAMANCA	0:40	tren
ZARAGOZA-	LLEIDA	2:00	tren-	TERUEL	0:50	tren-HUESCA	1:10	tren- SORIA
1:20 tren-	GUADALAJARA	2:00	tren					

Dades de “renfe.mdb”.

Taula Trens

Id	Vel_max	tipus
1	300	talgo
2	450	ave
3	310	expreso
4	450	ave
5	450	ave
6	328	talgo
7	300	intercity
8	350	expreso
9	360	expreso
10	400	ave
11	300	talgo
12	300	talgo
13	325	talgo
14	350	expreso
15	350	intercity
16	310	talgo
17	320	talgo
18	350	expreso
19	300	talgo
20	450	ave
21	425	intercity
22	390	euromet
23	430	euromet
24	450	ave
25	425	intercity
26	450	expreso
27	390	talgo

Taula Paradas

id_rec	ciutat	temps
0	Barcelona	0:00
0	Tarragona	1:00
0	Lleida	1:50
1	Barcelona	0:00
1	Girona	1:30
1	Barcelona	3:00
2	Barcelona	0:00
2	Tarragona	1:00
2	Castellon	2:10
2	Valencia	3:10
3	Valencia	0:00
3	Castellon	2:00
3	Teruel	2:50
3	Zaragoza	3:50

4	Valencia	0:00
4	Castellon	1:00
4	Tarragona	3:00
4	Barcelona	3:50
5	Zaragoza	0:00
5	Teruel	1:00
5	Castellon	2:10
5	Valencia	3:00
6	Zaragoza	0:00
6	Huesca	1:10
6	Zaragoza	2:20
6	Teruel	3:10
7	Valencia	0:00
7	Alicante	1:00
7	Valencia	2:00
7	Castellon	3:00
7	Valencia	4:00
8	Murcia	0:00
8	Alicante	1:00
8	Valencia	2:00
8	Alicante	3:00
8	Murcia	4:00
9	Almeria	0:00
9	Murcia	1:30
9	Alicante	2:30
9	Valencia	3:30
10	Sevilla	0:00
10	Cadiz	1:00
10	Malaga	2:00
10	Jaen	3:20
10	Granada	4:10
10	Almeria	5:00
11	Sevilla	0:00
11	Huelva	0:40
11	Sevilla	1:50
11	Cordova	3:00
11	Jaen	4:10
12	Jaen	2:00
12	Malaga	3:20
12	Cadiz	4:30
12	Sevilla	5:20
12	Almeria	0:00
12	Granada	1:00
13	Madrid	0:00
13	Toledo	0:30
13	CiudadReal	0:50
13	Cordova	1:20
13	Sevilla	2:00
13	Cordova	2:40
13	CiudadReal	3:20
13	Toledo	4:00

13	Madrid	4:30
14	Albacete	0:00
14	Cuenca	1:00
14	Guadalajara	2:00
14	Madrid	3:00
14	Toledo	3:40
14	CiudadReal	4:30
15	CiudadReal	0:00
15	Toledo	1:00
15	Madrid	1:40
15	Guadalajara	2:30
15	Cuenca	3:20
15	Albacete	4:00
16	Caceres	0:00
16	Merida	0:30
16	Badajoz	1:00
16	Caceres	1:30
17	Huelva	0:00
17	Badajoz	1:10
17	Caceres	1:40
17	Salamanca	3:10
17	Caceres	4:30
17	Badajoz	5:00
17	Huelva	6:00
18	Leon	0:00
18	Valladolid	0:50
18	Palencia	1:20
18	Burgos	1:50
19	Leon	0:00
19	Valladolid	0:30
19	Segovia	1:10
19	Madrid	2:00
19	Toledo	2:40
20	Zaragoza	0:00
20	Soria	1:10
20	Segovia	2:00
20	Valladolid	2:30
20	Leon	3:10
21	Vitoria	0:00
21	SanSebastian	0:30
21	Bilbao	1:10
21	Vitoria	1:50
22	Bilbao	0:00
22	Santander	0:50
22	Oviedo	2:20
22	Lugo	3:30
22	Compostela	4:20
23	Coruña	0:00
23	Compostela	0:30
23	Pontevedra	1:00
23	Compostela	1:30

23	Coruña	2:00
24	Coruña	0:00
24	Compostela	0:30
24	Lugo	2:00
24	Oviedo	3:30
24	Santander	5:00
24	Bilbao	5:30
25	Bilbao	0:00
25	Vitoria	0:30
25	Logroño	1:00
25	Soria	1:30
25	Segovia	2:40
25	Madrid	3:30
26	Madrid	0:00
26	Avila	0:40
26	Salamanca	1:50
26	Zamora	2:20
26	Leon	3:10

Taula Recorreguts

id_rec	nom
0	Cataluña1
1	Barcelona<->Girona
2	Mediterrani
3	Valencia->Zaragoza
4	Mediterrani2
5	Zaragoza->Valencia
6	Aragon
7	PaisValencia
8	Murcia<->Valencia
9	Mediterrani3
10	Andalucia1
11	Andalucia2
12	Andalucia3
13	Madrid<->Sevilla
14	LaMancha
15	LaMancha2
16	Extremadura
17	Huelva<->Salamanca
18	CastillaLeon
19	Leon<->LaMancha
20	Aragon<->Leon
21	PaisVasco
22	Cantabrico
23	Galicia
24	Cantabrico2
25	Bilbao<->Madrid
26	CastillaLeon

Taula Assigna

id_tren	id_rec	hora	preu
1	0	13:00	2.000 pta
2	2	10:00	3.000 pta
3	1	17:10	1.500 pta
4	3	12:30	2.500 pta
5	4	13:00	3.000 pta
6	5	15:00	3.000 pta
7	6	10:00	2.500 pta
8	7	15:00	1.900 pta
9	8	17:00	2.500 pta
10	9	9:00	2.000 pta
11	10	11:00	3.000 pta
12	11	12:00	2.500 pta
13	12	13:00	3.000 pta
14	13	12:00	4.000 pta
15	14	15:00	2.500 pta
16	15	10:00	2.700 pta
17	16	16:00	1.900 pta
18	17	13:00	2.900 pta
19	18	15:00	3.500 pta
20	19	10:00	3.700 pta
21	20	11:00	4.000 pta
22	21	17:00	2.000 pta
23	22	13:00	3.000 pta
24	23	18:00	1.900 pta
25	24	14:00	2.000 pta
26	25	10:00	3.800 pta
27	26	20:00	3.000 pta

Dades de “RailRibes.mdb”.

Taula Trens

Id	Vel_max	tipus
1	300	ribesTrain
2	450	AlbTrain
3	310	SuperTrain
4	450	UltraTrain
5	450	AlbTrain
6	328	SilverRoad
7	300	UltraSpark
8	350	WinTrain
9	360	Linusin
10	400	Talguin
11	300	expresin
12	300	SuperTrain
13	325	RibesTrain
14	350	expreso

15	350	intercity
16	310	talgo
17	320	talgo
18	350	expreso
19	300	talgo
20	450	ave
21	390	JuliTrain
22	430	InfoTrain

Taula Parades

id_rec	ciutat	temps
0	Barcelona	0:00
0	Tarragona	1:00
0	Lleida	1:50
0	Tarragona	2:50
0	Barcelona	3:50
0	Girona	5:00
1	Huesca	0:00
1	Zaragoza	0:50
1	Teruel	1:30
1	Castellon	3:00
1	Teruel	4:30
1	Zaragoza	5:10
2	Alicante	0:00
2	Valencia	1:20
2	Castellon	2:30
2	Tarragona	4:10
2	Castellon	5:50
3	Sevilla	0:00
3	Cadiz	1:00
3	Malaga	2:00
3	Almeria	3:00
3	Granada	3:30
3	Jaen	4:00
3	Cordova	5:00
3	Sevilla	5:50
4	Guadalajara	0:00
4	Cuenca	1:00
4	Albacete	2:00
4	CiudadReal	4:00
4	Toledo	5:00
4	Madrid	5:40
5	Caceres	0:00
5	Merida	0:30
5	Badajoz	1:00
5	Huelva	2:00
5	Sevilla	2:30
6	Leon	0:00
6	Zamora	1:00
6	Salamanca	1:30
6	Avila	2:30

6	Madrid	3:00
6	Segovia	4:00
6	Valladolid	4:30
6	Palencia	5:00
6	Burgos	5:40
7	Coruña	0:00
7	Compostela	0:30
7	Pontevedra	1:00
7	Compostela	1:30
7	Lugo	2:30
7	Orense	3:30
7	Zamora	4:30
8	Vitoria	0:30
8	SanSebastian	1:00
8	Bilbao	1:40
8	Santander	2:20
8	Oviedo	4:20
8	Bilbao	0:00
9	Girona	0:00
9	Barcelona	1:00
9	Tarragona	2:00
9	Castellon	3:40
9	Valencia	4:20
9	Alicante	5:00
9	Murcia	6:00
9	Almeria	7:30
10	Compostela	0:00
10	Lugo	1:00
10	Oviedo	2:10
10	Santander	4:00
10	Bilbao	4:30
11	Tarragona	0:00
11	Lleida	1:00
11	Zaragoza	2:10
11	Huesca	3:30

Taula Recorreguts

id_rec	nom
0	Catalonia
1	Aragonia
2	Valencia
3	AltAndalus
4	LaMancha
5	Extremadura
6	CastiLeon
7	Galicia
8	PaisVasco
9	Mediterrani
10	Cantabric
11	Catal-Arag

Taula Assigna

id_tren	id_rec	hora	preu
1	0	10:00	2.000 pta
1	11	16:00	3.500 pta
2	1	17:10	1.500 pta
3	2	10:00	3.000 pta
4	3	12:30	1.000 pta
5	4	13:00	2.000 pta
6	3	15:00	3.000 pta
7	5	12:00	3.000 pta
8	6	20:00	2.500 pta
9	7	11:00	4.000 pta
10	8	17:00	2.500 pta
11	9	18:00	3.500 pta
12	10	10:00	4.000 pta
13	10	16:00	3.800 pta
14	9	11:00	3.600 pta
15	8	10:00	2.700 pta
16	7	21:00	3.900 pta
17	6	19:00	2.700 pta
18	5	19:00	3.200 pta
19	4	20:00	2.200 pta
20	3	10:00	3.200 pta
21	2	20:00	2.800 pta
22	1	11:00	1.700 pta

Dades de “Iberia.mdb”.

Taula Avions

id	tipus
0	jumbo
1	charter
2	Concord
3	Concord
4	superJumbo
5	Concord
6	charter
7	superChar
8	airA2
9	AirJet
10	Xwing

Taula Parades

id_rec	hora	ciutat
0	0:00	Compostela
0	1:30	Madrid
0	3:00	Compostela
1	0:00	Madrid

1	1:40	Bilbao
1	3:20	Madrid
2	0:00	Barcelona
2	1:30	Madrid
2	3:00	Barcelona
3	0:00	Madrid
3	1:20	Valencia
3	2:40	Madrid
4	0:00	Madrid
4	1:00	Sevilla
4	2:00	Madrid
5	0:00	Sevilla
5	2:10	Canarias
5	2:30	Tenerife
5	4:40	Sevilla
6	0:00	Mallorca
6	1:10	Barcelona
6	2:40	Valencia
6	3:50	Mallorca

Taula Vols

id_rec	nom
0	PontGallec
1	PontBilbair
2	PontCatala
3	PontValencia
4	PontAndalus
5	Canari
6	Mallorqui

Taula Assigna

id_avion	id_rec	hora	preu
0	0	11:00	15.000 pta
1	1	15:00	15.000 pta
2	2	17:00	18.000 pta
3	3	12:00	15.000 pta
4	4	16:00	17.000 pta
5	5	13:00	20.000 pta
6	6	12:00	17.000 pta
7	1	10:00	15.000 pta
8	2	11:00	18.000 pta
9	3	18:00	15.000 pta
10	4	9:00	17.000 pta

Dades de “HispanAir.mdb”.

Taula Avions

id	tipus
0	jumbo
1	charter
2	Concord
3	Concord
4	superJumbo
5	Concord
6	charter
7	superChar
8	airA2
9	AirJet
10	Xwing

Taula Parades

id_rec	hora	ciutat
0	0:00	Compostela
0	1:40	Bilbao
0	3:20	Compostela
1	0:00	Bilbao
1	1:00	Barcelona
1	2:00	Bilbao
2	0:00	Valencia
2	0:50	Barcelona
2	1:40	Valencia
3	0:00	Valencia
3	1:30	Mallorca
3	3:00	Valencia
4	0:00	Valencia
4	1:50	Sevilla
4	3:40	Valencia
5	0:00	Madrid
5	1:20	Valencia
5	2:40	Madrid
6	0:00	Tenerife
6	0:20	Canarias
6	0:40	Tenerife

Taula Vols

id_rec	nom
0	Compos-
1	Bilbao-
2	Valencia-Barc
3	Valencia-
4	Valencia-
5	Madrid-Val

Taula Assigna

id_avion	id_rec	hora	preu
0	0	11:00	15.000 pta
1	1	15:00	15.000 pta
2	2	17:00	18.000 pta
3	3	12:00	15.000 pta
4	4	16:00	17.000 pta
5	5	13:00	20.000 pta
6	6	12:00	17.000 pta
7	1	10:00	15.000 pta
8	2	11:00	18.000 pta
9	3	18:00	15.000 pta
10	4	9:00	17.000 pta

Per iniciar la plataforma utilitzem el fitxer “ini.bat” que inicia tots els agents al mateix ordinador i amb la següent comanda:

```
java jade.Boot -platform -gui agentA1:AgentIberia agentA:AgentAvions
agentA2:AgentHispanAir agentT1:AgentRenfe agentT2:AgentRailRibes
agentT:AgentTrens agentC:AgentCoord
```

Per realitzar el joc de proves s’ha utilitzat l’agent “Dummy”, enviant missatges ACL a l’agent “AgentC”, de tipus “request”, i amb protocol “fipa-request”. Només es mostrarà el contingut del missatge enviat i el contingut del missatge que s’obté com a resposta.

- 1) Missatge enviat a les 3:19 del 20/9/2000:

CONTINGUT: Barcelona-Tarragona-20/9/2000-17:00

RESPOSTA: -RUTA 1: -BARCELONA 10:00 TARRAGONA 11:00 20 2000 RAILRIBES 1 0

- 2) Missatge enviat a les 3:24 del 20/9/2000:

CONTINGUT: Salamanca-Barcelona-21/9/2000-17:00

RESPOSTA: -RUTA 1: -SALAMANCA 20:30 AVILA 21:30 20 2700 RAILRIBES 17 6-AVILA 21:30 MADRID 22:00 20 2700 RAILRIBES 17 6-MADRID 12:30 BARCELONA 14:00 21 18000 IBERIA 8 2

- 3) Missatge enviat a les 3:40 del 20/9/2000:

CONTINGUT: Oviedo-Almeria-22/9/2000-17:00

RESPOSTA: -RUTA 1: -OVIEDO 15:20 LUGO 16:30 20 3000 RENFE 23 22-LUGO 16:30 COMPOSTELA 17:20 20 3000 RENFE 23 22-COMPOSTELA 11:00 MADRID 12:30 21 15000 IBERIA 0 0-MADRID 13:00 VALENCIA 14:20 21 20000 HISPANAIR 5 5-VALENCIA 15:20 ALICANTE 16:00 21 3600 RAILRIBES 14 9-ALICANTE 16:00 MURCIA 17:00 21 3600 RAILRIBES 14 9-MURCIA 17:00 ALMERIA 18:30 21 3600 RAILRIBES 14 9-RUTA 2: - OVIEDO 15:20 LUGO 16:30 20 3000 RENFE 23 22-LUGO 16:30 COMPOSTELA 17:20 20 3000 RENFE 23 22-COMPOSTELA 11:00 MADRID 12:30 21 15000 IBERIA 0 0-MADRID 16:00 SEVILLA 17:00 21 17000 IBERIA 4 4-SEVILLA 10:00 CADIZ 11:00 22 3200 RAILRIBES 20 3-CADIZ 11:00 MALAGA 12:00 22 3200 RAILRIBES 20 3-MALAGA 12:00 ALMERIA 13:00 22 3200 RAILRIBES 20 3

- 4) Missatge enviat a les 3:48 del 20/9/2000:

CONTINGUT: Girona-Tenerife-22/9/2000-17:00
 RESPOSTA: -RUTA 1: -GIRONA 11:00 BARCELONA 12:00 20 3600 RAILRIBES 14 9-
 BARCELONA 17:00 MADRID 18:30 20 18000 IBERIA 2 2-MADRID 09:00 SEVILLA 10:00 21 17000
 IBERIA 10 4-SEVILLA 13:00 TENERIFE 15:30 21 20000 IBERIA 5 5-RUTA 2: -GIRONA 11:00
 BARCELONA 12:00 20 3600 RAILRIBES 14 9-BARCELONA 13:10 VALENCIA 14:40 20 17000
 IBERIA 6 6-VALENCIA 16:00 SEVILLA 17:50 20 17000 HISPANAIR 4 4-SEVILLA 13:00
 TENERIFE 15:30 21 20000 IBERIA 5 5-RUTA 3: -GIRONA 11:00 BARCELONA 12:00 20 3600
 RAILRIBES 14 9-BARCELONA 12:00 TARRAGONA 13:00 20 3600 RAILRIBES 14 9-
 TARRAGONA 13:00 CASTELLON 14:40 20 3600 RAILRIBES 14 9-CASTELLON 14:40 VALENCIA
 15:20 20 3600 RAILRIBES 14 9-VALENCIA 16:00 SEVILLA 17:50 20 17000 HISPANAIR 4 4-
 SEVILLA 13:00 TENERIFE 15:30 21 20000 IBERIA 5 5

5) Missatge enviat a les 3:51 del 20/9/2000:

CONTINGUT: Missatge amb format erroni
 RESPOSTA(refuse): Error de format o de contingut al missatge.Format: ciutat_origen-
 ciutat_desti-dia/mes/any-hora:minut

6) Missatge enviat a les 3:52 del 20/9/2000:

CONTINGUT: Tarragona-Tarragona-20/9/2000-17:00
 RESPOSTA(refuse): Error de format o de contingut al missatge.Format: ciutat_origen-
 ciutat_desti-dia/mes/any-hora:minut

7) Missatge enviat a les 3:54 del 20/9/2000:

CONTINGUT: Paris-Tarragona-21/9/2000-17:00
 RESPOSTA(refuse): Error de format o de contingut al missatge.Format: ciutat_origen-
 ciutat_desti-dia/mes/any-hora:minut

8) Missatge enviat a les 3:58 del 20/9/2000:

CONTINGUT: Paris-Tarragona-21/9/1999-17:00
 RESPOSTA(refuse): Error de format o de contingut al missatge.Format: ciutat_origen-ciutat_desti-
 dia/mes/any-hora:minut

9) Missatge enviat a les 19:05 del 21/9/2000:

CONTINGUT: Cordova-Madrid-22/9/2000-17:00
 RESPOSTA(inform): -RUTA 1: -CORDOVA 13:20 CIUDADREAL 15:20 22 4000 RENFE 14 13-
 CIUDADREAL 15:20 TOLEDO 16:00 22 4000 RENFE 14 13-TOLEDO 16:00 MADRID 16:30 22
 4000 RENFE 14 13

10) Missatge enviat a les 19:07 del 21/9/2000:

CONTINGUT: Cordova-Madrid-24/9/2000-17:00
 RESPOSTA(inform): -RUTA 1: -CORDOVA 13:20 SEVILLA 14:00 22 4000 RENFE 14 13-
 SEVILLA 17:00 MADRID 18:00 22 17000 IBERIA 4 4-RUTA 2: -CORDOVA 13:20 CIUDADREAL
 15:20 22 4000 RENFE 14 13-CIUDADREAL 15:20 TOLEDO 16:00 22 4000 RENFE 14 13-TOLEDO
 16:00 MADRID 16:30 22 4000 RENFE 14 13

11) Missatge enviat a les 19:10 del 21/9/2000

CONTINGUT: Cordova-Madrid-24/9/2000-17:00
 RESPOSTA(inform): -RUTA 1: -MADRID 09:00 SEVILLA 10:00 22 17000 IBERIA 10 4-
 SEVILLA 13:00 TENERIFE 15:30 22 20000 IBERIA 5 5-RUTA 2: -MADRID 12:00 TOLEDO
 12:30 22 4000 RENFE 14 13-TOLEDO 12:30 CIUDADREAL 12:50 22 4000 RENFE 14 13-
 CIUDADREAL 12:50 CORDOVA 13:20 22 4000 RENFE 14 13-CORDOVA 13:20 SEVILLA
 14:00 22 4000 RENFE 14 13-SEVILLA 13:00 TENERIFE 15:30 23 20000 IBERIA 5 5

9. CONCLUSIONS I VALORACIÓ PERSONAL

La realització d'aquest projecte ha suposat un esforç personal pel poc temps en que s'ha dut a terme, però es veu compensat pels nous coneixement adquirits en el desenvolupament de sistemes multiagent, una nova metodologia de la intel·ligència artificial que pot tenir un gran futur.

Per mitjà dels sistemes multiagent es poden desenvolupar aplicacions distribuïdes fàcilment, gràcies principalment a les especificacions publicades per la FIPA que han ajudat a que sorgeixin nous entorns de desenvolupament de sistemes multiagent, com ara JADE, que proporcionen eines molt útils per a la implementació d'aquests sistemes.

El fet de no tenir a priori cap coneixement sobre sistemes multiagent ha suposat un petit obstacle que s'ha vist superat gràcies al grup creat per Aïda Valls i Toni Moreno a l'abril de 2000, anomenat "GruSMA". Espero que aquest grup pugui seguir portant els sistemes multiagent a altres alumnes, i que aquests puguin desenvolupar noves aplicacions.

Així valoro molt positivament la realització d'aquest projecte, que espero pugui servir de punt de partida per altres aplicacions basades en sistemes multiagent.

10. BIBLIOGRAFIA

- [1] Jaime Bocio, Projecte Final de Carrera, URV, “*Logística de les operacions de trasplant*”, 1999-2000
- [2] Foundation for Intelligent Physical Agents, “*FIPA 98 Specification*”, FIPA, Ginebra, Suïza, 1998.
- [3] David Isern, Projecte Final de Carrera, URV, “*Avaluació d’entorns de desenvolupament de Sistemes Multiagent*”, 1998-1999
- [4] Galan, Alan, “*Jafmas complete thesis*”, Department of Electrical and Computer Engineering and Computer Science, 1998.
URL: <http://www.ececs.uc.edu/~abaker/JAFMAS>
- [5] Nwana, H (BT). i Wooldridge, M. (QMW), “*Software Agent Technologies*”, BT Labs.
URL: http://www.labs.bt.com/agents/publish/papers/sat_report.htm
- [6] Golobardes, E., Pérez, J., Porta, J.M., “*Tutorial sobre agents*”, Seminari d’Intel·ligència Artificial (UPC), 1996.
- [7] “JADE programmer’s guide”, 2000
- [8] “On agent-based software engineering”, Nicholas R. Jennings, Universitat de Southampton, 1999