EPS
Escola Politècnica Superior
UdG

Projecte/Treball Fi de Carrera

Estudi: Enginyeria Informàtica. Pla 1997

Títol: AXARM: Una Aplicació eXtensible per Assistència Remota i Monitorització

Document: 1. Memòria

Alumne: Xavier Vallejo López

Director/Tutor: Antonio Bueno Delgado
Departament: Arquitectura i Tecnologia de Computadors
Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): 09/2008

Agraïments:

Al meu tutor Antonio Bueno Delgado per la seva inestimable ajuda a l'hora de treballar en aquest projecte.

Als membres del grup de recerca BCDS (Broadband Communications and Distributed Systems) de la Universitat de Girona per resoldre'm tots els meus dubtes.

Als meus companys de laboratori per aguantar-me tant en els bons moments com en els dolents.

A la meva família, ja que sempre ha estat quan els he necessitat oferint-me el seu suport.

Índex

Capítol 1

Introducció

1.1 L'Esclerosi Múltiple

L'Esclerosi Múltiple (EM, també coneguda amb el nom encephalomyelitis disseminata) és una malaltia neurodegenerativa, crònica i no contagiosa que afecta al sistema central nerviós. Actua disminuint la mielina, una capa amb funcions aïllants que envolta a la fibra nerviosa de la neurona. Actualment no hi ha cap tractament per curar-la, però sí que hi ha medicaments per intentar alleugerir els seus efectes. Tampoc es saben les causes exactes que provoquen aquesta malaltia. A més a més, es presenten vàries subformes d'esclerosi múltiple, la qual cosa fa que els pacients reaccionin de manera diferent al llarg dels anys.

Un dels símptomes més freqüents de la malaltia és la reducció de la mobilitat del pacient que pot arribar fins a invalidesa. Una dada esperançadora és que si és tractada, el 50% dels pacients recuperen bona part de la mobilitat, i només el 10% moren per aquesta causa. Després de l'epilèpsia, és la malaltia neurològica més freqüent. Afecta a 1 de cada 1000, en persones que solen estar entre 20-40 anys, i sol afectar més a les dones.

Els símptomes típics dels pacients són: l'adormiment de les extremitats, espasmes,

fatiga, dolor, cansament, alteracions en la vista... No necessàriament s'han de complir tots, ja que dependrà de cada pacient. Un dels principals problemes dels doctors és la dificultat d'assegurar el diagnòstic de la malaltia, ja que es sol confondre amb altres símptomes de menor importància, encara que en realitat l'EM ja es troba en el pacient. És clau, poder detectar l'EM a temps, ja que el seu tractament immediat pot millorar molt la salut del pacient.

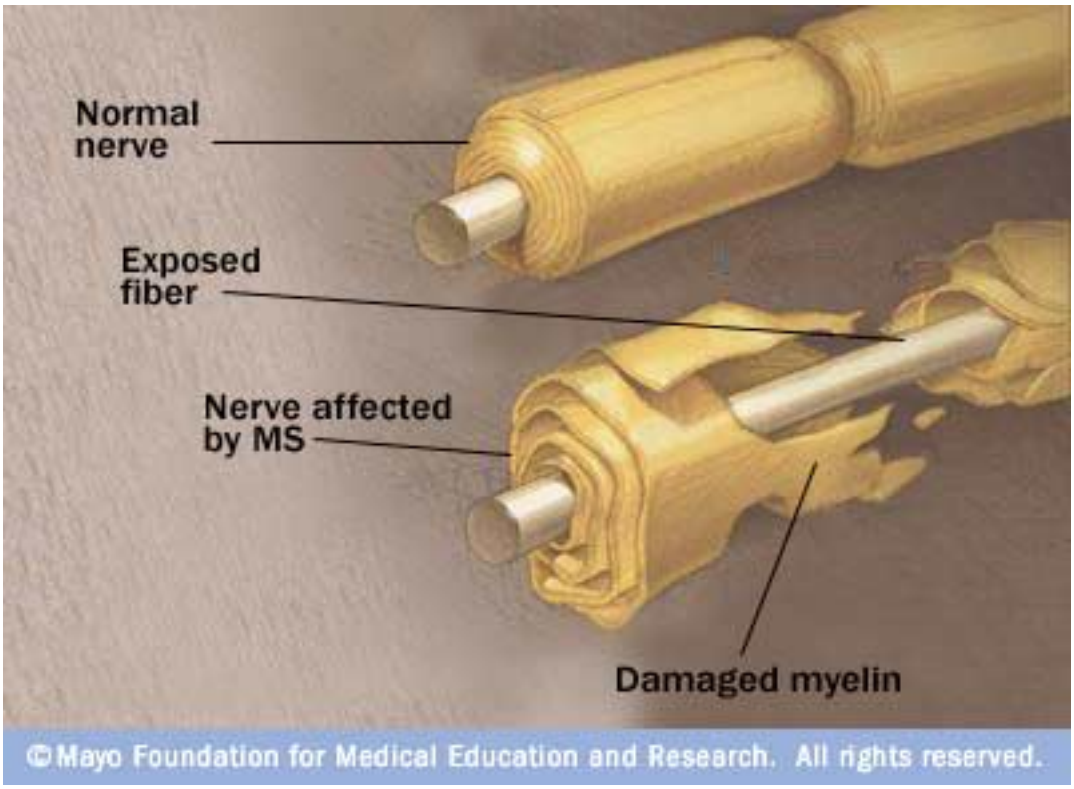

Figura 1.1: L'esclerosi múltiple afecta i danya al sistema nerviós del cos humà

1.2 El projecte TRiEM

Des de l'any 2005, el grup de recerca BCDS[1] de la Universitat de Girona ha estat treballant amb l'Hospital de Dia de Girona de la FEM (Fundació Esclerosi Múltiple)[2] en el projecte TeleRehabilitació i Esclerosi Múltiple (TRiEM). desenvolupant el prototipus d'una aplicació multiplataforma de videoconferència amb gravació. Això permet ajudar als especialistes del centre a dur a terme activitats de rehabilitació a distància (el pacient a casa seva i l'especialista al centre). Les característiques de les malalties neurodegeneratives tractades i les circumstàncies en les que treballa el centre (econòmiques, geogràfiques, etc) fan molt desitjable poder realitzar remotament activitats de consulta mèdica, fisioteràpia, psicologia i neuropsicologia, logopèdia o teràpia ocupacional i recreativa.

Figura 1.2: Logotip del Projecte TRiEM

Objectius del projecte TRiEM

Desenvolupar una aplicació que permeti als doctors visitar d'una manera virtual als pacients a casa i minimitzar els desplaçaments a les consultes mèdiques, amb el doble objectiu d'oferir un seguiment més llarg i continu i alhora millorar la qualitat de vida dels pacients.

- Aplicació de videoconferència robusta.
- Requeriments típics d'una llar, pensats des d'un punt de vista de l'usuari.
- Interfície d'usuari fàcil de fer funcionar.

- Configuració el menys complicada possible.
- Extensible a noves funcionalitats.

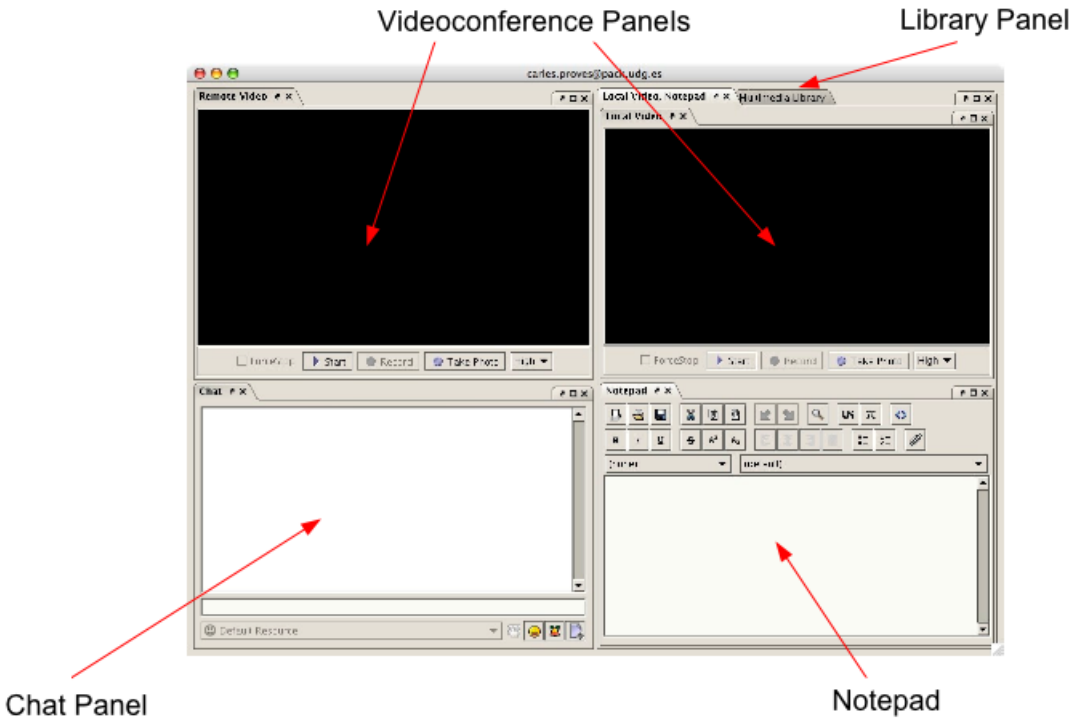

Figura 1.3: Finestra que es veu quan es parla amb un altre usuari

El sistema va ser dissenyat per fer servir, el més possible, una infraestructura estàndard de baix cost; tant en termes d'equipament informàtic (CPU, tarja gràfica, webcams) com en comunicacions. Per exemple: el prototipus pot oferir una comunicació d'àudio i vídeo bona a dues bandes amb una connexió bàsica que sol ser de 3 Mbps de baixada i 300 kbps de pujada.

Arquitectura

L'aplicació TRiEM és una modificació de l'aplicació JBother[3]. Aquesta aplicació és un client de missatgeria instantània lliure que està escrit en el llenguatge de programació Java. Una de les avantatges del Java és que és un llenguatge multiplataforma (funciona en varis sistemes operatius que disposin d'una màquina virtual de Java).

El prototipus creat en el projecte TRiEM disposa de les següents funcionalitats:

• Videoconferència

• Enregistrament Àudio / Vídeo

• Bloc notes

• Llibreria Multimèdia

• Xat

Per poder fer aquestes tasques, es fa servir una arquitectura peer-to-peer híbrida.

• El P2P pur (connexió directa) es fa servir per transmetre les dades de la videoconferència entre usuaris. També es pot estendre a altres funcionalitats que requereixin una amplada de banda més gran.

• L'arquitectura Client-Servidor es fa servir per enviar missatges de xat i missatges de control amb el servidor (login, llista de contactes...).

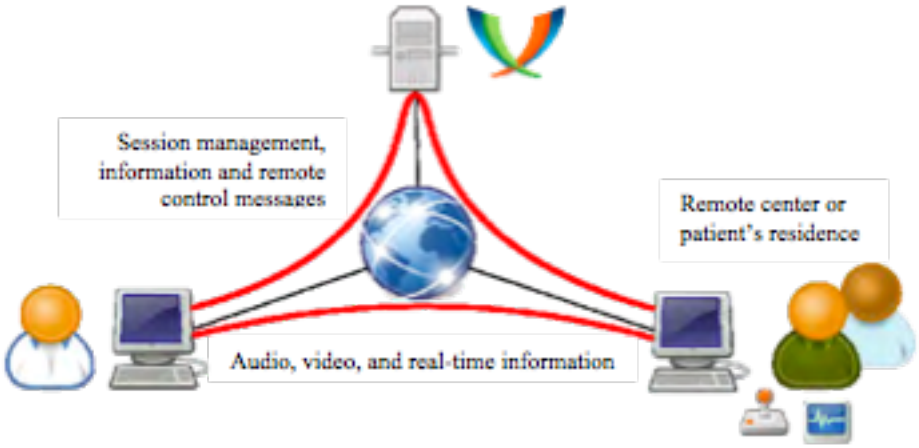

Figura 1.4: Arquitectura híbrida implementada en el TRiEM

Tecnologia de comunicació

El programa TRiEM està basat en la tecnologia de comunicació XMPP[4] (*eXtensible Messaging and Presence Protocol*). És un protocol de missatges obert, documentat i ampliable fent servir XML, i especialment pensat per la missatgeria instantània. A més a més, és un protocol estandarditzat i té els seus orígens en la tecnologia Jabber. Un exemple famós que el fa servir, és el Google Talk.

• Descentralització: Qualsevol pot fer anar el seu propi servidor de XMPP i no hi ha un únic servidor central.

• Estàndards oberts: La IETF[5] ha formalitzat el XMPP com una tecnologia de missatgeria instantània (RFC 3920, RFC 3921)

• Seguretat: Els servidors de XMPP poden estar aïllats del servidor públic de XMPP i són robusts (per SASL i TLS).

• Flexibilitat: Funcionalitats pròpies es poden construir a sobre del protocol.

Tecnologia multimèdia

Per transmetre la informació multimèdia (videoconferència), es fan servir les llibreries JMF de Sun Microsystems[6]. Actualment s'està mirant de fer servir una nova llibreria multimèdia (FMJ [7]) ja que JMF ha quedat desfasada (l'última actualització és del 2004).

• Protocol RTP i còdecs: Totes les dades multimèdia són enviades fent servir el protocol de streaming RTP a través de UDP, ja que resulta més important la fluïdesa de la videoconferència que no pas la seva qualitat. Amb UDP no hi ha retransmissions, i és més indicat per aquest tipus de tasca.

• El flux de vídeo és codificat fent servir el H.263 i l'àudio es codifica amb els formats GSM o ULAW.

• H.263/RTP només transmet en: SQCIF (128x96), QCIF (176x144) i CIF (352x288).

Una de les dificultats a l'hora de fer funcionar el programa a casa dels pacients, és que cal obrir un nombre de ports en el router. Per la videoconferència, calen 4 ports UDP oberts (del 4002 al 4005). El 4002 i 4003 són pel vídeo (4002 per dades i 4003 per control), el 4004 i 4005 per l'àudio (4004 per dades i 4005 per control).

1.3 El projecte AXARM

L'aplicació AXARM és una iniciativa del grup de Comunicacions i Sistemes Distribuïts de la Universitat de Girona per impulsar una eina d'assistència telemàtica entre doctors i pacients. Facilita una eina útil als especialistes d'un centre per realitzar tasques de rehabilitació, assistència remota o monitorització (sanitària, assistencial o d'una altra mena) amb pacients que es trobin en un altre punt físic a través d'Internet.

AXARM parteix de la base inicial del projecte TRiEM (TeleRehabilitació i Esclerosi Múltiple, FEM/UdG, 2005-06). Una de les principals tasques que s'ha perseguit és fer una refactorització complerta de tota l'aplicació. La idea general és modificar el programari per fer més manejables els canvis sense perdre o guanyar noves funcionalitats. És possible que l'usuari, a simple vista, no s'adoni de cap canvi entre l'aplicació antiga i la nova. Per això, AXARM comparteix amb TRiEM la seva estructura híbrida. La modularització permet afegir noves funcionalitats en forma d'extensions (plugins) que poden ser a nivell de programari (enviar vídeos pregravats a un pacient) o poden incorporar algun element extra de maquinari per ajudar a la monitorització d'un pacient.

El procés de la refactorització és progressiu: un cop familiaritzat amb el codi és necessari identificar quines parts són del nucli i quines acabaran sent plugins. D'aquesta manera, s'aconsegueix desfer la seva estructura monolítica, ampliar el seu abast i fer-lo més extensible amb una arquitectura modular. Les parts del nucli són disponibles per fer-les servir en els plugins actuals i futurs, així facilitant la feina de creació. Al ser modular, un programador pot desenvolupar noves funcionalitats sense haver de conèixer tot el programa. La documentació de com desenvolupar una extensió (la seva API) i alguns exemples, són suficients per poder estendre l'aplicació. Com a demostració d'aquesta API i exemple d'extensió s'ha desenvolupat un plugin que permet controlar un dispositiu d'entrada (en aquest cas un joystick) per part del pacient, i que l'especialista pugui assignar-li activitats

i supervisar-les.

L'aplicació s'ha realitzat amb eines de codi lliure i el llenguatge de programació Java, partint d'una versió modificada del programa de missatgeria JBother. Aquest programa es basa en el protocol estandarditzat XMPP explicat anteriorment. Per a les funcionalitats multimèdia, al TRiEM es va fer servir tecnologia JMF, però aquest projecte ha considerat variants més actualitzades, com les llibreries FMJ.

Per realitzar tots aquests canvis, s'ha seguit una planificació resumida en les següents fases:

• Primera fase: Estudi de l'aplicació actual, i les seves funcionalitats. Veure totes les interdependències entre les seves funcionalitats i com actuar-hi.

• Segona fase: Refactorització del codi, establir les diferències entre el que serà aplicació, i la part que serà d'extensions. Com interactuaran entre elles.

• Tercera fase: Proves, i creació de noves extensions per generar noves funcionalitats.

• Fase final: Proves en viu amb pacients i especialistes.

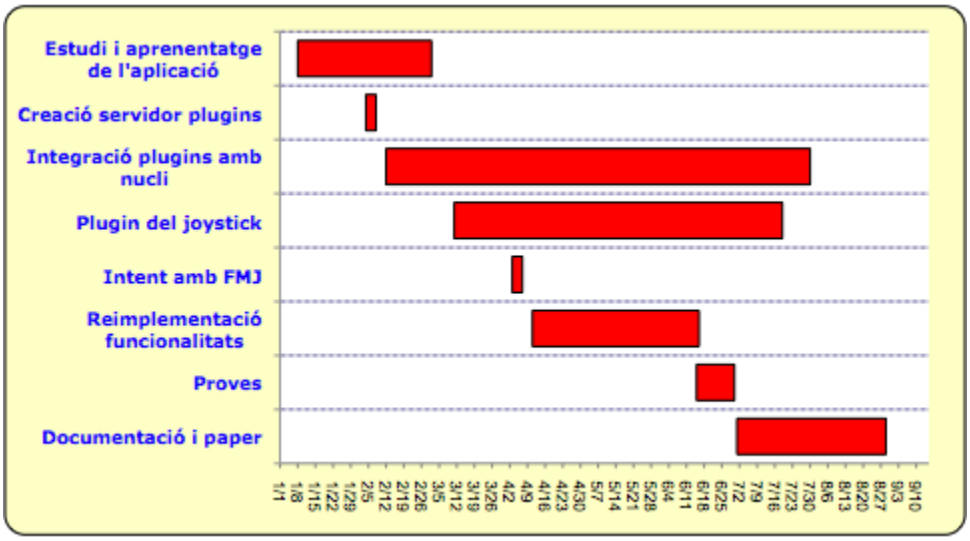

Figura 1.5: Diagrama de Gantt del projecte AXARM

1.4 Per què continuar amb JBother?

Com s'ha explicat, AXARM continua a partir d'on va acabar l'aplicació TRiEM. Per tant, és correcte afirmar que AXARM també és una modificació de l'aplicació original JBother. Hi ha varis motius pels quals s'ha decidit continuar amb aquesta aplicació i no fer-ne servir una més actual (ja que l'última actualització de JBother és del 2006):

- El motiu principal és que es disposa de tot el codi font (qualsevol part es pot modificar i no hi ha cap limitació de codi tancat).

- La feina anterior de l'aplicació TRiEM ha resultat ser molt útil i, a més, s'ha pogut aprofitar la part realitzada per aquesta nova aplicació. Com que estava ben estructurada, ha facilitat la seva refactorització i com a prova d'aquest fet, totes les funcionalitats del TRiEM s'han convertit en plugins de l'aplicació AXARM. Gràcies a aquest fet, s'ha evitat molta reescriptura de codi, temps i diners.

- El programa original (JBother), a pesar que no rep noves actualitzacions és una bona base fàcilment manipulable, i el cost de trobar un altre programa de missatgeria que pugui complir els nostres requeriments, el posterior anàlisi de tota l'aplicació i després, la implementació de les parts ja implementades en l'anterior, és molt més elevat que no pas seguir amb el JBother i anant fent les modificacions adients.

A pesar que per diferenciar el nou treball respecte a l'anterior, es fa servir el nou terme de l'aplicació, en alguns cops es parlarà del projecte TRiEM. Cal diferenciar entre el projecte i l'aplicació, ja que el primer es refereix a tota la feina que s'ha anat desenvolupant i que es continua fent, i el segon del programa pròpiament dit.

Capítol 2

Disseny del model

2.1 Estat de l'art

Està clar que amb la recent incorporació de les noves tecnologies aplicades a la societat, s'ha fet un gran avenç en molts camps (Internet, la telefonia mòbil...). Amb tot i això, encara no està tot inventat i de ben segur que en un futur proper sorgiran noves innovacions. La telemedicina es troba a mig camí entre la medicina convencional i la tecnologia, el que fa pensar que també alguns d'aquests avenços es succeiran en aquest camp. L'exemple més espectacular d'aquesta simbiosi es remunta al setembre de 2001. En aquell any, es va realitzar la primera intervenció quirúrgica transatlàntica. Un cirurgià a Nova York va controlar un braç robòtic per extreure la vesícula biliar d'un pacient que es trobava a Estrasburg (14.000 Km de distància). Aquell fet va fer eco en els mitjans de comunicació degut a l'espectacularitat de la fita aconseguida.

Dintre dels molts camps possibles on té presència la telemedicina, dues de les estrelles són la radiologia i la cardiologia. No només existeixen aquests dos, i mostra d'això és el nostre projecte, centrat en l'assistència sanitària a malalties neurodegeneratives. Resulta, a primera vista, sorprenent que molts d'aquests camps no estiguin més explorats. No es fa

pràcticament cap consulta mèdica a través de la xarxa però, des d'un punt personal, estic completament segur que aquest camp esdevindrà més i més gran i gairebé es convertirà en una costum, o necessitat, de la societat.

En l'àmbit internacional, la referència en aquests temes es troba als Estats Units. La major font de les inversions venen del departament de Defensa, mentre que a la Unió Europea se n'encarrega més la Comissió Europea i les grans empreses de telecomunicacions. Els camps de negoci també són diferents entre els dos: Estats Units busca mercat de comerç i Europa la cooperació. Degut a l'estructura del sector sanitari a Europa, la iniciativa pública s'ha decantat més per aquest tema que la privada. Els països escandinaus aporten més interès en aquests temes, a l'igual que Grècia, degut a la seva pròpia geografia dispersa.

A l'estat espanyol, el projecte TRiEM es pot considerar un dels pioners. Recordem que els seus inicis es situen l'any 2005. A més a més, actualment és un programa real, no un prototip. S'han realitzat proves pilot però encara queda molt de camí a fer. Té un gran potencial perquè és molt escalable, i el seu cost d'instal·lació és molt baix. També hi ha altres projectes en universitats espanyoles relacionats amb la telemedicina, però no es detallaran en aquesta memòria.

Altres iniciatives de telemedicina també estan començant a sorgir. Una d'elles és la consulta de telemedicina entre centres especialitzats i centres d'atenció primària. D'aquesta manera, es vol reduir desplaçaments dels pacients, però aquest no és l'objectiu de la nostra aplicació. Un altre exemple és que es comencen a donar màsters d'especialització en telemedicina, assignatures dins la carrera de medicina, etc.

2.2 Requeriments d'usuari

Com s'ha comentat anteriorment, l'aplicació AXARM no requereix d'un equip de grans prestacions per tal de funcionar. Uns dels objectius de l'aplicació TRiEM que s'ha mantingut en la nova remodelació era fer servir uns requeriments els més menors possibles, i que el pacient en disposi fàcilment a casa seva.

Uns requeriments aproximats serien els següents:

- Ordinador amb un mínim de processador (a partir d'un Pentium 3 ja s'obtenen resultats satisfactoris). És necessari per tal de fer funcionar amb fluïdesa la videoconferència, però per gravacions de vídeo es necessita mínim Pentium 4.
- Màquina virtual de Java, a partir de la versió 1.5.
- Connexió a Internet, mínim ADSL (la part més exigent és la videoconferència, que importa tant la pujada com la baixada). Amb 1Mbps de baixada i 300 kbps de pujada es pot fer funcionar.
- Una webcam compatible amb el SO i amb el JMF (en Windows ha de suportar Vídeo For Windows (VFW), i en Linux el Vídeo for Linux (V4L)).
- Micròfon i cascs són recomanables.

A pesar de ser programat en Java, en la part de la videoconferència existeixen alguns problemes amb la llibreria multimèdia que dificulten la posta a punt en alguns sistemes operatius (Linux i Mac OS X concretament). D'aquest punt se'n parlarà més a fons en l'apartat del plugin de la videoconferència.

2.3 Requeriments del servidor

Anteriorment s'ha parlat dels objectius del projecte TRiEM, un dels quals era l'escabilitat. Aquest aspecte es reflexa clarament a l'hora de parlar de la part del servidor.

El servidor és un component necessari en l'arquitectura del programa. Actualment, el servidor ha de proporcionar els següents serveis:

- Servidor de missatgeria XMPP.
- Servidor web.
- Servidor SFTP.
- I obert a noves funcionalitats segons el que puguin necessitar noves extensions...

Servidor XMPP: Existeixen moltes opcions a l'hora de muntar un servidor Jabber/XMPP gràcies a que el protocol és de codi obert. Hi ha implementacions en diferents llenguatges, que guarden les dades en diversos tipus de Bases de Dades (Mysql, Sqlite...), amb múltiples opcions a escollir.

En el nostre camp de proves, el dimoni que hem fet servir és el ejabberd. Està escrit en el llenguatge Erlang (de l'empresa Ericsson), i disposa de varies opcions interessants (manteniment d'usuaris, logging...). La més destacada és que permet establir clústers de servidors XMPP. Un sol servidor pot mantenir al voltant de 5000 usuaris connectats simultàniament. Per mantenir una connexió amb un client solen fer falta entre 2 i 3 sockets, i el nombre màxim de sockets d'un ordinador és de 32768 (contant els reservats pel SO). Ejabberd realitza una càrrega de balanceig automàtica quan està establert en forma de clúster, i per tant l'escabilitat de possibles usuaris potencials està bastant resolta.

També ens cal un servidor web, per mantenir el servidor dels plugins. Un Apache actualitzat és suficient per mantenir els fitxers i el llistat dels plugins. Es pot afegir PHP i MySql, per si es vol facilitar alguna pàgina d'informació, de benvinguda...

Degut a una nova funcionalitat d'un plugin, és necessari que el servidor disposi del servei de SFTP. D'aquesta manera, els pacients i els especialistes es poden transmetre informació de manera asíncrona (és a dir, no cal que els dos estiguin connectats alhora). La informació pot anar des de imatges del pacient, fragments de vídeo enregistrat, resultats d'exercicis dels plugins, etc. A més a més, al ser Secure, aquesta informació s'envia de manera xifrada per evitar problemes amb la seguretat. Un bon servidor SFTP és el mysecureshell[8], ja que permet definir perfectament tots els permisos de cada usuari, enregistra tots els moviments del SFTP i dóna una protecció necessària a l'hora d'oferir aquest servei. Sobre quins permisos cal oferir als pacients i especialistes, se'n parlarà en el punt 3.5.2 (en la Llibreria Multimèdia).

Finalment, entre aquests serveis cal un mecanisme de coordinació en el servidor. Per exemple, si es dóna d'alta un usuari en el sistema de missatgeria, també cal donar d'alta en el servei de SFTP, i restringir els permisos segons si és un pacient, o un especialista. Com que es disposa de servidor web, una solució seria amb PHP, ja que permet realitzar la coordinació amb un script propi.

2.4 Noves propostes

Durant el desenvolupament d'aquest projecte, es va disposar de l'oportunitat de provar un nou dispositiu que podia ser útil. Es tractava d'una pantalla plana d'ordinador tàctil.

La pantalla tàctil és de la marca LG Flatron L1510BF i a diferència d'una pantalla normal, s'ha de connectar una sortida USB de la pantalla a l'ordinador per tal de capturar els moviments tàctils. Per fer-la funcionar, cal instal·lar uns controladors prèviament i calibrar-la.

Figura 2.1: Foto de la pantalla tàctil en funcionament en el laboratori

Es va estar provant el seu possible ús en l'entorn del nostre programa, però finalment es va descartar per les següents raons:

- El preu de la pantalla encara és elevat en el mercat, al voltant dels 500€. El nostre projecte es destaca pel baix cost dels seus components.
- Només hi ha controladors per Windows, la qual cosa restringeix el seu ús en altres SO.

- La precisió tàctil no era molt precisa, i per fer moltes operacions normals com tancar una finestra, o fer un doble clic era complicat i no es podia fer a la primera.
- L'estructura de la pròpia pantalla, fa que si un usuari pressiona massa fort la pantalla, aquesta es mou i resulta molt més difícil treballar amb ella.
- No és una pantalla multitàctil.
- Finalment, caldria adaptar el nostre programa al seu ús específic en el cas que es fes servir.

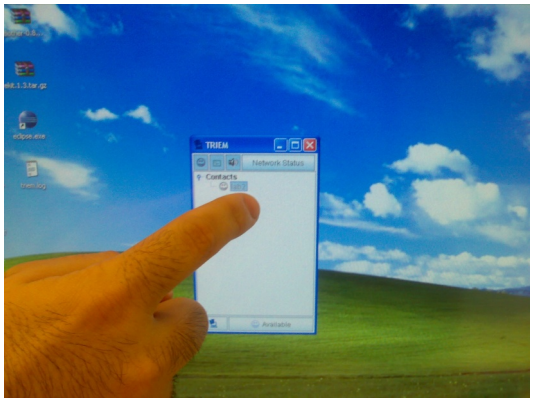

Figura 2.2: S'observa la dificultat per accedir al programa d'una forma tàctil

També durant el desenvolupament del plugin del joystick (s'explicarà més endavant), han sorgit noves idees molt interessants a l'hora d'aplicar nous dispositius físics, que es detallen a continuació:

- Fer servir una catifa de ball dels videojocs.
- Utilitzar el comandament de la videoconsola Wii de Nintendo.
- El dispositiu "Wii Balance Board" que es ven amb el videojoc WiiFit, també de l'empresa Nintendo.
- L'aparell TrackIR que permet controlar els moviments del cap.

2.4.1 Catifa de ball

La primera idea ve donada pel món dels jocs de ball, el qual el més famós d'ells és el DDR (*Dance Dance Revolution*). Per jugar en aquest joc, el jugador tria una cançó i l'ha d'anar seguint ballant, encertant els moviments en el moment precís.

El que ens interessa d'aquest joc és el dispositiu de la catifa de ball. És una quadrícula de 3x3, en la que el pacient es col·loca en el centre (posició de repòs) i pot anar amunt, avall, esquerra o dreta. Dins de la catifa hi ha una malla que, quan el pacient trepitja una direcció, es detecta una pulsació. Un exemple de possible plugin consistiria en que, al pacient se li mostra, amb unes senzilles instruccions, en quines direccions s'ha de moure. El mateix plugin en la part de l'especialista va enregistrant totes les accions del pacient per un posterior anàlisi. Es venen catifes de ball per videoconsoles i ordinadors, amb el qual el cost del perifèric és econòmic i es pot instal·lar fàcilment a casa del pacient.

Figura 2.3: Una catifa de ball DDR

Figura 2.4: Conversor on es senyala la sortida USB, i una de les dues entrades

A més a més, tal i com es mostra en la fotografia, encara que la catifa sigui dissenyada per videoconsoles, existeixen conversors que ofereixen una connexió tant estàndard com és el USB. D'aquesta manera, una catifa pensada per una videoconsola es pot fer servir en un PC. Addicionalment, el conversor permet connectar dues catifes de ball i fer-les funcionar alhora com si fos un sol dispositiu (amb un sol port USB). Encara que la idea original era per jugar amb dos jugadors als jocs de ball, en el nostre àmbit també pot ser molt interessant, ja que es pot expandir l'àrea d'acció al crear els exercicis pels pacients.

2.4.2 Wiimote

La segona idea, més interessant consisteix en fer servir el comandament inalàmbric de la Wii, anomenat Wiimote.

Aquest aparell fa servir la tecnologia Bluetooth que permet una comunicació inalàmbrica amb la consola. Incorpora varis acceleròmetres, que permet obtenir la posició 3D d'on es troba, la seva orientació i acceleració. En el comandament es disposa d'un sensor infraroig que permet situar-se amb dues llums infraroges (és la barra que es sol posar a sobre del televisor). La gran avantatge del comandament és el seu baix cost (aproximadament 45€). Un dispositiu industrial de posicionament amb encoders pot valer al voltant dels 2500€. Pel nostre objectiu, no ens cal una gran precisió i per tant ens resulta molt més atractiva la idea del comandament.

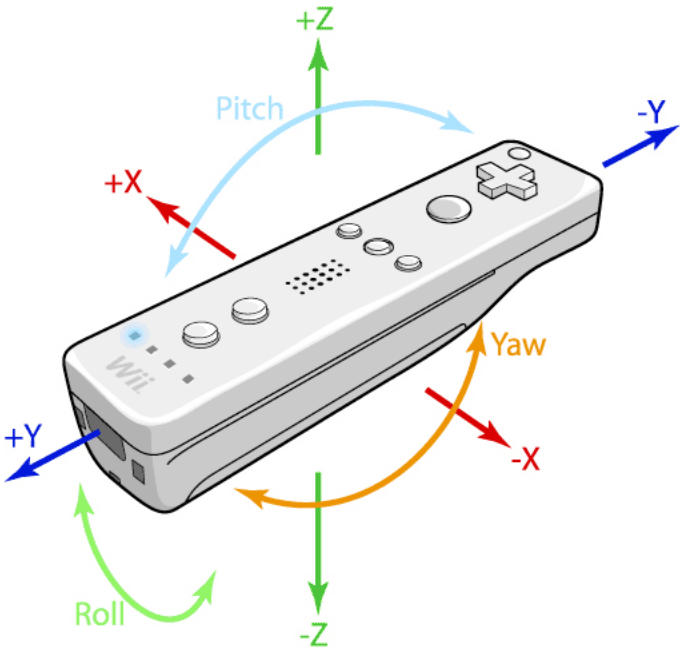

Figura 2.5: El controlador de la Wii: Wiimote

La comunicació amb el PC es fa a través del Bluetooth, i també ja existeixen vàries llibreries funcionals que permeten la lectura del comandament i el seu accés en varis llenguatges de programació. En Java es disposen de varies opcions, algunes d'aquestes les expliquem a continuació:

- WiiuseJ[9]: Java API pels Wiimotes, en el fons és una capa superior a les llibreries en C Wiiuse per fer-les servir en Java (hi accedeix a través del JNI). És una llibreria fàcil de fer servir, completa (llegeix acceleròmetres, vibració, la càmera infraroja...) i té uns requeriments tècnics inferiors que altres llibreries (per exemple, no necessita compatibilitat jsr-082 en la pila Bluetooth). És compatible amb Windows i Linux, però no en Mac OS X.

- WiiremoteJ[10]: Una altra llibreria amb requeriments jsr-082. Les últimes versions (a partir de la 1.4) permeten llegir dades de la Wii Balance Board, un fet molt interessant que s'explicarà amb més detall a continuació. És compatible amb Windows, Linux i Mac OS X sempre que es compleixin els requeriments del Bluetooth. Un

inconvenient és que la llibreria no és de codi lliure.

- motej[11]: Encara una altra llibreria en Java. Disposa d'una llicència Apache (el codi font és disponible), però no està tant desenvolupada com WiiremoteJ.

2.4.3 Wii Balance Board

La tercera opció és una de les més recents i prometedores. La Wii Balance Board és un nou accessori també de la videoconsola Wii que permet practicar exercici físic.

Figura 2.6: La taula d'exercicis Wii Balance Board amb una funda pels peus

Com es pot observar, té la forma d'una taula d'exercicis aeròbics que es col·loca en el terra. Disposa de quatre sensors de pressió situats a cada punta de la taula i que permeten mesurar tant la força amb què es pressiona com la posició on es fa la força. Amb només tres sensors la taula ja podria determinar els càlculs, però segurament s'ha afegit un quart per poder fer comprovacions d'errors. La posició on es fa la força ens permet determinar, per exemple, el centre de gravetat de la persona.

Un bon exemple del usos de la taula és amb el joc WiiFit, que es ven juntament amb la taula. A continuació es mostren vàries imatges del joc per il·lustrar les possibilitats de l'aparell, i com s'aplicarien en el nostre cas.

Figura 2.7: WiiFit mesura el centre de gravetat de la persona

Comparant les forces dels sensors de l'esquerra, amb els sensors de la dreta assoleix una molt bona precisió en aquesta tasca.

Alguns exercicis del WiiFit són d'equilibri, la qual cosa és realment útil de cara a un ús amb pacients. La imatge de l'esquerra mostra un exercici d'equilibri, mentre que la de dreta mostra un resultat d'un altre exercici (observi el seguiment que fa del centre de gravetat enregistrat).

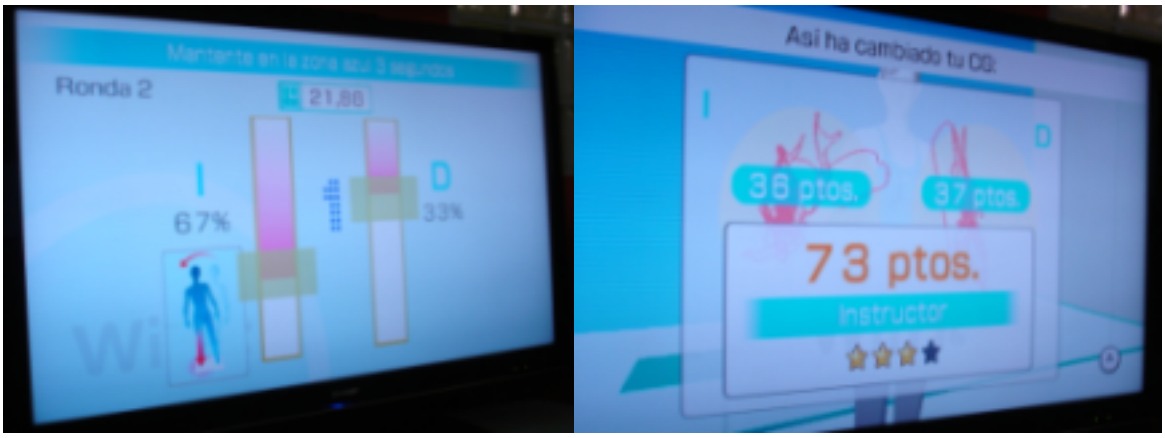

Figura 2.8: Exercicis d'equilibri amb el WiiFit

Una de les virtuts de l'exercici és que disposa d'una interfície molt simplificada de cara a l'usuari. A més a més, s'afegeix un plus que és que el joc és bonic i entretingut. També es poden fer exercicis molt més complexes com el següent: el jugador disposa d'un temps per fer entrar unes pilotes dins d'uns forats. Per fer-ho, amb el moviment detectat a la taula, fa que el terra també es mogui. És a dir, si el jugador fa força endavant, la plataforma s'inclinarà cap endavant i les boles seguiran el pendent. A més dificultat, més boles, menys temps i recorreguts més complexes.

Figura 2.9: El moviment del jugador fa que la plataforma també variï la seva orientació

Per tant, a mode de resum, aquest aparell pot resultat molt útil i ideal per fer exercicis amb extremitats inferiors amb èmfasi en aspectes d'equilibri. També és molt atractiu pel

seu baix cost (al voltant d'uns 90€). A l'igual que el comandament de la Wii, s'estan desenvolupant llibreries per poder capturar dades des d'un ordinador i no des de la consola amb el seu software.

2.4.4 TrackIR

L'últim element que s'explicarà és el TrackIR, encara que no s'ha considerat tant com els anteriors. La seva aplicació principal és en el món del videojocs i de la realitat virtual. Es considera un element immersiu (dins la Realitat Virtual) del tipus HMD (*Head Mounted Device*). És un visor que té forma d'un casc que es col·loca a sobre del cap de la persona. El dispositiu detecta els moviments que realitza l'individu, i els envia a l'ordinador, que aquest realitza una tasca en concret. El jugador veu a través d'uns displays LCD que incorpora les ulleres del casc. En simuladors de vol, o de cotxes és molt útil ja que simula com si realment el jugador estigués dins del vehicle o nau.

Figura 2.10: Esquema de funcionament del TrackIR

El dispositiu disposa de 6 DOF (Graus de Llibertat), la qual cosa significa que l'aparell permet detectar la posició on es troba (X, Y, Z), l'orientació i la rotació. Encara que el seu ús és per jocs, es podria fer servir per altres aplicacions que necessitessin fer moviments (en aquest cas, els del cap). L'empresa que el comercialitza (NaturalPoint) el ven a un preu no molt elevat (al voltant de 150$), però com tot, a més precisió del dispositiu, un cost més elevat. Un dels inconvenients és que l'aplicació no és de codi obert, i potser accedir al dispositiu no sigui tant fàcil com en els casos anteriors.

Capítol 3

Plugins

3.1 Creació d'un plugin

Tots els plugins que es pugin crear han de seguir la següent estructura:

- Un plugin només pot ser un fitxer jar (en el fons, no és res més que un fitxer comprimit zip que conté el codi a executar i altres recursos com imatges o sons). En l'arrel del plugin ha d'existir un fitxer anomenat "plugin.properties" que conté el següent:

  mainClass: "com.valhalla.jbother.plugins.Nom_Classe_Plugin", és la classe principal que intentarà executar l'aplicació quan explori un plugin.

  description: Una breu descripció del plugin.

  name: Nom del plugin.

  APIVersion: Versió.

  author: Autor del plugin.

  releaseDate: Un data com la distribució del plugin.

  os: Sistema operatiu (si es vol per tots els SO es posa "all").

arch: Si el plugin està dirigit per una arquitectura concreta o no ("all").

Els dos últims paràmetres es comproven juntament amb les variables que es troben en execució en la màquina de Java.

Els següents dos paràmetres són opcionals i afecten a la mostra del plugin dins la finestra del xat:

leftSide: Si té el valor true, la pestanya del plugin es situarà a la part esquerra, en cas contrari a la dreta.

hide: Si té el valor true, el plugin no tindrà cap pestanya en la finestra (no es veurà), però seguirà funcionant. Pot ser útil per plugins que hagin de fer altres funcions invisibles a l'usuari.

- El codi font s'organitza dins d'un package del java: "com.valhalla.jbother.plugins". La classe principal (mainClass) ha d'implementar tota la interfície del plugin i tots els plugins han d'heretar els mètodes i implementar-los.

- Un plugin disposa dels següents mètodes:

  - **public boolean init()** Mètode de la classe principal del plugin que es crida quan l'aplicació carrega el plugin. El mètode s'encarrega d'inicialitzar-lo i registrar-lo per diferents events que pugui rebre. Un petit fragment de codi de com seria un mètode init:

```
1 import com.valhalla.jbother.JBother;
2 import com.valhalla.pluginmanager.*;
```

```
3
4  public boolean init()
5  {
6  PluginChain.addListener (this);
7  //Inicialitzar variables, mètodes...
8  com.vahalla.Logger.debug("Plugin initiated");
9  return true;
10 }
```

- **public void unload()** Representa l'acció contrària del mètode anterior, quan es descarrega el plugin del programa.

```
1  import com.valhalla.jbother.JBother;
2  import com.valhalla.pluginmanager.*;
3
4  public void unload()
5  {
6  PluginChain.removeListener( this );
7  }
```

El plugin s'ha de treure de la cadena de plugins en aquest mètode, per tal que es puguin carregar i descarregar dinàmicament en el programa principal.

- **public Object crear(Object obj)** Aquest mètode permet la comunicació entre el programa principal i el plugin. El programa crida aquest mètode, i l'objecte de sortida espera que sigui un JPanel per mostrar-lo per pantalla. Tota l'acció del plugin s'encarrega de fer-la el mateix plugin dins d'aquest mètode. Dit d'una altra forma, el programa principal no sap realment què és el que s'executa,

només sap que el que rebrà, ho mostrarà per pantalla.

- Tots els events que el programa pot crear, els reben tots els plugins. Si es vol actuar al rebre un determinat event cal implementar el següent codi en el plugin:

```
public void handleEvent (PluginEvent event)
{
if (event instanceof ConnectEvent)
{
   //fer el codi necessari per l'event
}
   //altres events...
}
```

Existeixen varis events ja programats, com quan es connecta al servidor, quan surt del programa... També se'n poden crear de nous si és necessari.

- Si es vol disposar d'un panell d'opcions propi en les opcions del programa, llavors cal afegir més aspectes en els codis de init() i unload().

```
import com.valhalla.jbother.JBother;
import com.valhalla.pluginmanager.*;
import com.valhalla.jbother.preferences.PreferencesDialog;

public boolean init()
{
PluginChain.addListener(this);
PreferencesDialog.registerPluginPanel
     ("Nom",new JPanel());
```

```
10  return true;
11  }
```

La nova línia de codi (PreferencesDialog) serveix per afegir un nou JPanel al menú de les opcions. Per afegir-lo, ens cal crear una nova classe en un altre package diferent dins el jar "com.valhalla.jbother.preferences".

- Aquesta classe nova derivada de PreferencePanels, ha d'implementar els següents mètodes:
  - getSettings()
  - getPreferencesPanelName() Ha de retornar el nom del Plugin, i és important que el nom contingui la paraula "Plugin".

- A més a més si es vol guardar dades en un fitxer de text, cal implementar aquests dos mètodes:
  - setSettings()
  - writeSettings()

3.2 Servidor de plugins i com s'integren

Un dels primers reptes d'aquest projecte, era saber com incorporar un sistema de plugins a l'aplicació. En primer lloc, calia veure si existia alguna estructura existent que es pogués fer servir. Si es trobava, llavors calia veure com aprofitar-la i fer-la servir conjuntament amb un servidor de plugins.

En la pròpia pàgina del JBother, es pot veure que hi ha una petita API de plugins. Lamentablement, el servidor online dels plugins està mort, i per tant, si no es modifica l'aplicació, tot el tema dels plugins no funciona, ja que per defecte es connecta a l'adreça del servidor.

Al no disposar de cap servidor per veure com estava muntat, la única opció era veure com estava estructurat per dins el codi i fer reingenieria inversa. A partir del parser que seguia per fer una lectura del servidor i una mica de prova i error, es va aconseguir muntar un servidor propi de plugins amb èxit.

3.2.1 Estructura del servidor

El servidor necessita d'una llista, en format text, que li enviï la informació dels plugins que disposa el servidor. Aquesta llista conté el següent format:

```
1 Plugin list will follow
2 description    name    APIVersion    version    author
     releaseDate    os    arch    size    fileName    user
3 Un exemple de descripció.    NomPlugin    91214    0.1.0
     Autor    Data    S.O    Arquitectura    tamany    nomfitxer.
     jar JBother
```

- La primera línia cal que sigui exactament igual per la lectura de la llista.
- Cada camp està separat per tabuladors. La segona línia indica els camps que té un plugin, i a partir d'aquesta cada línia nova és un plugin.
- El tamany del fitxer s'ha d'expressar en enter i format de bits (#ls -l), si no es posa correctament, el programa detectarà un error si s'intenta descarregar el plugin.
- La APIVersion ha de ser la 91214 per la nostra versió de AXARM.
- Segons el SO i l'arquitectura es pot ocultar plugins pel programa. Si es posa la cadena "all" es veuran per tots.
- El camp user implementa restriccions per usuaris i només hi ha tres opcions possibles: un d'administració, un altre per l'especialista i el tercer pels pacients.
  En el programa es fa servir el paràmetre recurs del JID (*Jabber ID*) que serveix per donar una prioritat a l'usuari[12]. Segons la seva prioritat tindrà accés a un nombre determinat de plugins per ell. Un exemple seria nom1@nomdomini.com/Pacient

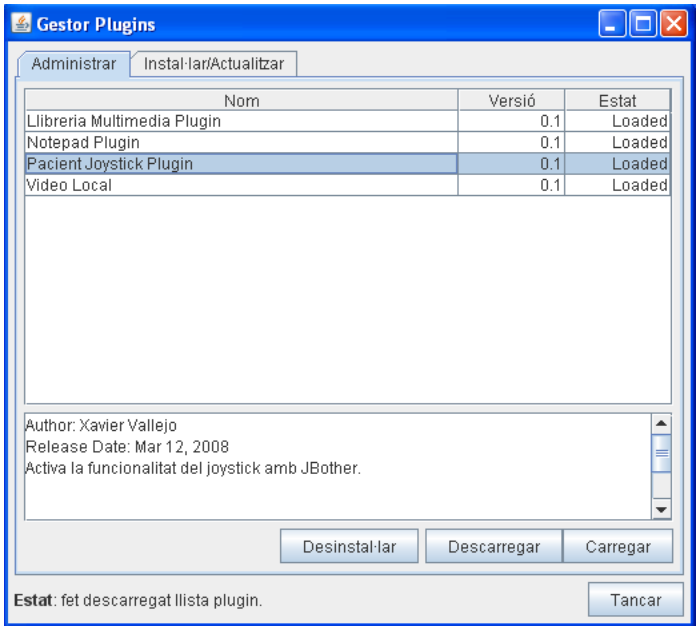

Figura 3.1: Pantalla principal de la configuració dels plugins

Quan un plugin es troba finalitzat pel seu funcionament, apareix en el repositori general dels plugins i pot ser instal·lat en l'aplicació.

3.2.2 Integració plugins amb el programa principal

3.2.2.1 Càrrega inicial d'un plugin

Per instal·lar un plugin com a part del sistema, simplement cal disposar del fitxer jar en la carpeta plugins de l'aplicació. Cada cop que s'inicií el programa, aquest intentarà carregar el plugin en el sistema. Els passos que segueix el programa es resumeixen en la següent numeració:

1. A través de la interfície gràfica, l'aplicació es connecta a un servidor on es mostren tots els plugins disponibles per descàrrega. L'usuari escull els plugins que vol i el programa se'ls descarrega.

2. Un cop descarregats, el programa principal explora tota la carpeta de plugins per classificar-los.

3. L'aplicació obre el fitxer jar, i consulta l'arxiu plugin.properties. A dins seu està definit la classe principal que ha d'executar.

4. Aquesta classe és una implementació de la interfície Plugin i que ha d'estar dins del package de Java: com.valhalla.jbother.plugins. El plugin ha d'implementar els mètodes vists a l'apartat 3.1.

5. El programa principal crea un objecte de la classe definida, i crida a la funció init. Si retorna verdader, l'aplicació reconeix el plugin i el té a punt per ser executat.

L'aplicació principal disposa una classe Plugin, que és una **interfície.** Tots els plugins hauran d'implementar els mètodes de la classe Plugin, que són init, unload i crear. També

es disposa de classes per poder descarregar un plugin del nucli i manejar el panell gràfic dels plugins, però potser interessa més, realment, com carrega un plugin en el sistema. Hi han dues classes principals encarregades d'aquesta feina: PluginJAR i PluginLoader. El nucli (aplicació) disposa de les següents classes per tractar amb els plugins:

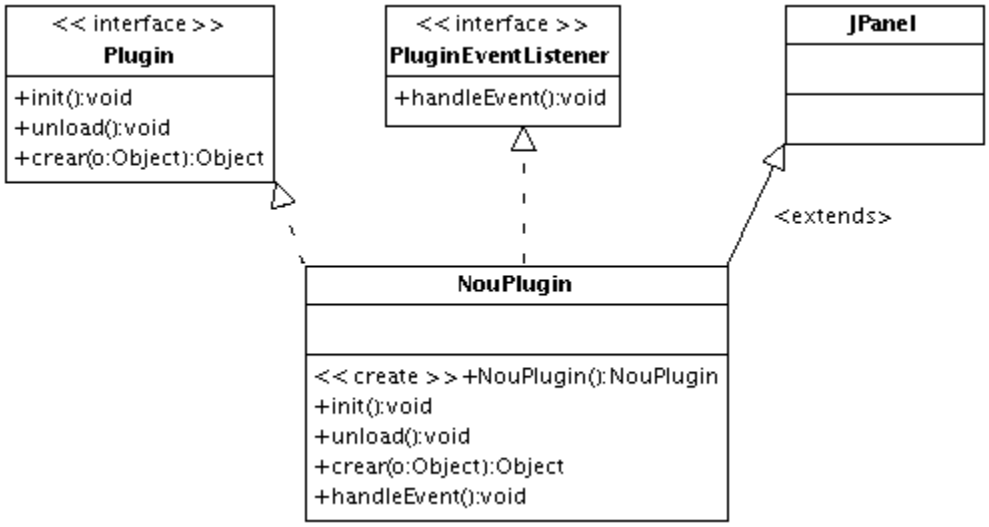

Figura 3.2: Diagrama de classes UML on es veu com actua un Plugin

- PluginChain és la cadena de plugins que es van afegint i traient de la llista de plugins carregats. Simplement consisteix d'un vector dinàmic de Java.

- PluginJAR: Aquesta classe representa un plugin des d'un fitxer JAR. Java disposa d'unes llibreries per tractar amb aquests fitxers (java.util.jar.*). La missió d'aquesta classe és obrir el JAR, examinar el seu contingut i propietats, i retornar un objecte PluginJAR que servirà pel Loader fer la seva feina.

  A més a més, aquesta classe s'encarrega de carregar els jar, i crear objectes de les classes del plugin. PluginLoader crida el mètode loadPlugin() de la classe PluginJAR, que intenta crear una nova instància de la classe (objecte Plugin).

```
public Plugin loadPlugin() {
PluginLoader loader = PluginLoader.getInstance();

try{
Class c = loader.loadClass
     (props.getProperty("mainClass"));
if (c == null)
return null;
plugin = (Plugin) c.newInstance();
this.loaded = plugin.init();
}
catch (Exception ex){
System.out.println(ex.getCause().getMessage());
com.valhalla.Logger.debug
   ("Could not load the main class from the jar file.");
}
return plugin;
}
```

L'objecte props conté les propietats del plugin, que anteriorment s'ha carregat. D'aquesta manera el programa carrega els plugins al sistema i crea l'objecte.

- PluginLoader: S'encarrega de tota la gestió dels plugins, manté quins plugins són carregats, descarregats, invàlids, disponibles...

3.2.2.2 Instanciació del plugin amb el nucli

Un cop es disposa d'un plugin carregat en l'aplicació, els passos pels quals el plugin actua dins l'aplicació són els següents:

1. L'aplicació recorre tots els plugins carregats anteriorment.
2. Per cada plugin, recupera l'objecte creat anteriorment i executa el mètode crear.
3. Aquest mètode rep de paràmetre un objecte ChatPanel del programa principal. L'objecte representa la finestra amb el qual un usuari parla amb un altre. D'una altra forma, és la finestra contenidora de tots els plugins i funcions que es fan servir en una conversa.
4. El mètode crear, inicialitza tot el codi que farà servir el plugin, però ha de retornar un objecte gràfic (JPanel) al programa principal.
5. El programa principal rebrà un objecte, que sap que l'haurà de mostrar per pantalla, però el més important és que aquest no sap realment què està fent el plugin.

```
[...]
PluginLoader loader = PluginLoader.getInstance();
Hashtable taula = loader.getLoadedPlugins();
//Creem un iterador i recorreguem els plugins
Iterator it = taula.keySet().iterator();
while (it.hasNext()) {
//Agafem el següent plugin (Loaded)
Object element = it.next();
//Treballem amb el string
String comparar = element.toString();
//Obtenim el jar
PluginJAR jar = loader.getPlugin(comparar);
```

```
13  //Engeguem el plugin (ha d'estar carregat), retorna objecte
        plugin creat anteriorment!
14  if (jar.getLoaded()){
15  Plugin plug = jar.getPlugin();
16  //Cridem al mètode comú a tots els plugins, que retorna un
        objecte
17  Object obj = plug.crear(this);
18  //Casting per dir-li que el podem mostrar
19  Component com = (Component)obj;
20  //Es crea la vista mostrant l'objecte...
21  }
22  }
```

A alt nivell, es pot simplificar dient que el codi que s'està executant es troba realment en el codi del plugin (no pertany al nucli). A través d'events de la GUI (botons, check-boxes...) i timers programats és el que permet la interacció entre el plugin, i l'usuari. El programa principal només rep un objecte gràfic (un canvas), que l'únic que pot fer és mostrar-lo per pantalla. El plugin és qui controla el codi.

Quan un plugin està en funcionament, té un control d'una àrea de la GUI com també té accés a qualsevol missatge XMPP que l'aplicació rebi. El plugin actua o no, segons si ha estat programat per realitzar una tasca quan rep els missatges. Addicionalment, un plugin pot fer servir el nucli de l'aplicació per enviar missatges fàcilment a l'altre usuari. Freqüentment els plugins s'han de desenvolupar en parelles per expressar les diferències entre les accions i interfícies de cada rol (pacient o especialista).

Per si no ha quedat suficientment clar, a continuació es mostra un esquema amb els passos per carregar i instanciar un plugin:

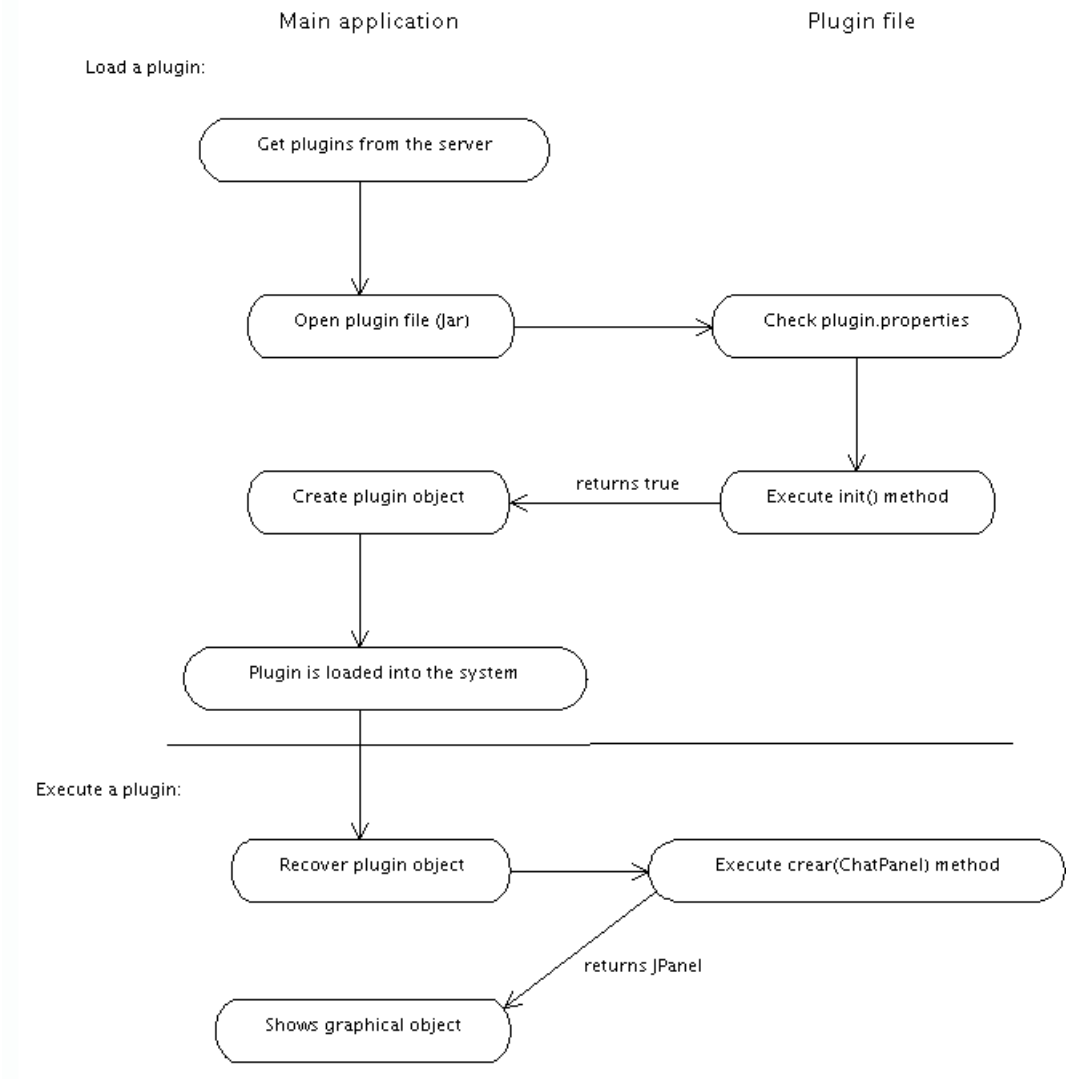

Figura 3.3: Esquema d'inicialització d'un plugin

3.3 Desenvolupament d'un nou plugin

El grup BCDS juntament amb la FEM estem contínuament pensant en noves idees i propostes per millorar l'atenció als pacients a través de la telemedicina. Una d'elles, consisteix en proposar fer algun exercici al pacient, per tal de comprovar el seu estat actual. Hi ha diversos tipus d'exercicis que es poden practicar segons les parts que es volen activar. Els més comuns representen exercicis de braços i mans, però també són molt útils els exercicis que actuen sobre les extremitats inferiors. D'aquests, malauradament, no n'hi ha molts de disponibles. La raó és que es disposa de més aparells per actuar en extremitats superiors que inferiors.

Un dels reptes del projecte consistia en dissenyar i implementar un nou plugin, que alhora fes servei a alguna d'aquestes necessitats. A més a més, aquest plugin havia de servir com d'exemple per la implementació de futurs nous plugins. D'aquí sorgeix la idea de desenvolupar un plugin que faci actuar el joystick. El joystick és un element força comú als ordinadors personals, i alhora hi ha moltes llibreries que el fan servir. És un molt bon exemple per obtenir un resultat visible sense disposar de masses complicacions. Com a element de telemedicina, no resulta molt útil perquè només treballa en concret la mà, i la coordinació entre vista i moviment.

La idea fonamental del plugin és la seva divisió en tres fases:

1. Captura de dades del dispositiu.
2. Enviament de les dades del pacient a l'especialista
3. Representació i tractament de les dades, sigui per comandes o en un entorn gràfic.

Es pot veure fàcilment l'avantatge de la divisió en fase: canviant el dispositiu es pot aprofitar parts dels punts 2 i 3, i només caldria refer la part de la captura de dades.

3.4 Implementació del plugin del Joystick

Tornant al tema de la creació del plugin, el primer que s'ha realitzat és el control d'un joystick d'ordinador. Aquest és el seu funcionament:

- El plugin del joystick és en realitat dos plugins diferents. Per una banda es disposa d'un plugin per l'especialista, i l'altra banda un pel pacient. S'ha fet d'aquesta forma per mantenir la asimetria general del programa (per exemple, en el control de la videoconferència per part de l'especialista).

- L'especialista s'encarrega d'enviar-li un exercici al pacient, d'un llistat que disposa. Aquests exercicis han estat creats segons les necessitats dels especialistes. Un exemple consisteix en moure un mico per la pantalla i ha d'anar a buscar els plàtans.

- Mentre que el pacient va fent l'exercici, l'especialista també el pot veure gairebé en temps real. D'aquest aspecte de la comunicació se'n parlarà més endavant.

- L'especialista pot parar un exercici i enviar-li un altre de diferent. Alhora, pot estar fent servir la videoconferència, xatejar o prendre notes. De cada exercici finalitzat s'emmagatzemen els resultats obtinguts per a un futur seguiment del pacient.

En la interacció persona-ordinador a través del joystick, es poden realitzar una gran diversitat exercicis. La majoria d'aquests exercicis han estat pensats per la FEM, ja que en el fons ells saben quins exercicis tindran més utilitat. La UdG, inicialment, va desenvolupar els següents exemples d'exercicis:

- Representant un mico com el cursor del joystick, el pacient ha d'anar movent-se per la pantalla per anar a cercar uns plàtans.

- Una forma un pèl més evolucionada consisteix en que el mico també va als plàtans, però ara els ha d'arrossegar a un altre destí (una caixa). D'aquesta forma també

entra en joc la part de coordinació.

Aquests exercicis es poden ajustar per dificultat (l'àrea dels plàtans pot ser més gran o petita), o per temps (un exemple seria comptar el nombre d'encerts en un temps determinat). A continuació es parlarà en detall, tant de la part del pacient, com la de l'especialista.

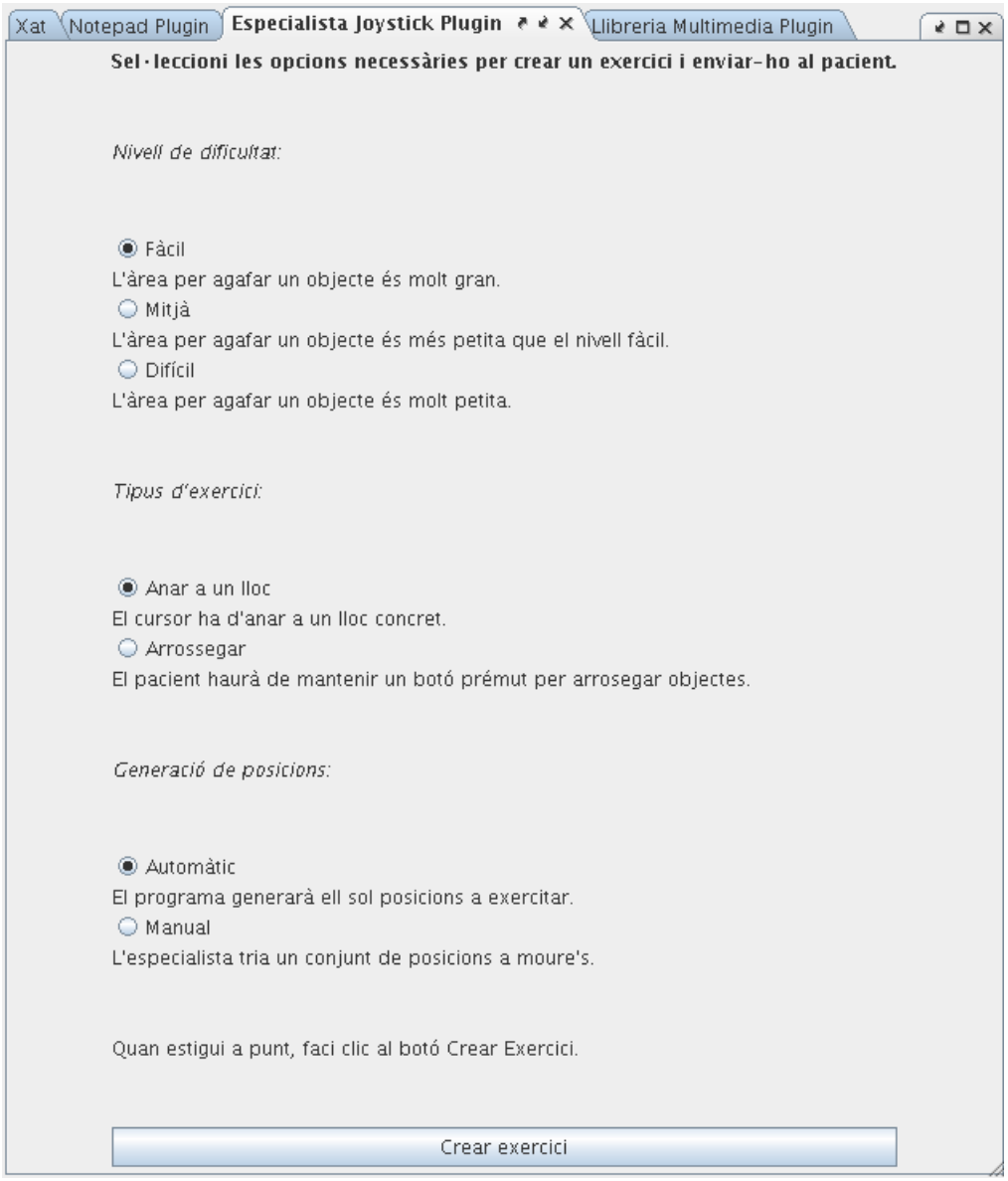

Figura 3.4: Panell on l'especialista tria les opcions de l'exercici

3.4.1 Part del pacient

El pacient primer ha d'esperar a rebre un exercici, i després actuarà movent el dispositiu. Tal i com s'ha plantejat en el punt anterior, es desglossa aquest apartat en les tres parts del plugin: adquisició de dades, enviament i representació.

Adquisició de dades:

Per fer l'adquisició de dades d'un joystick s'ha fet servir una llibreria en Java ja desenvolupada, per tal d'obtenir resultats immediats i no haver de saber programar un joystick a baix nivell. Existeixen moltes llibreries disponibles per la lectura del dispositiu, però a l'hora d'escollir, la que s'ha fet servir havia de complir els següents requeriments:

1. Ha de ser multiplataforma com el Java, és a dir, que la llibreria funcionés en Windows, Linux i Mac Os X. A més ha de ser purament escrita en Java.

2. Ha de reconèixer tot tipus de dispositius, tant actuals com varis que portin anys en el mercat.

3. Que fos fàcil d'incorporar i funcionable dins del plugin (tot el plugin només pot estar format per un jar).

4. Que estigués documentat i tenir llicència GPL.

La llibreria que hem fet servir ha estat jinput[13], que permet realitzar aquestes tres tasques sense problemes. jinput detecta dispositius com joysticks, comandament digital per jocs i/o entreteniment, volants... fins i tot el propi teclat i ratolí. En el laboratori de proves es disposa d'un joystick Genius (USB, 4 Botons, 3 eixos) que el detecta perfectament. Jinput requereix d'una altra llibreria Java anomenada Jutil[14].

Una de les petites dificultats era el punt 3, el qual per fer-lo funcionar cal descomprimir les llibreries necessàries quan es carrega el plugin. Aquest procés el fa automàticament el plugin cada cop que es carrega, totalment transparent per l'usuari. Per recollir les dades, es programa un temporitzador en Java (10ms és un bon valor) i va fent consultes al joystick (estil poll, nosaltres hem de preguntar al dispositiu). Retorna double pel valor de la X i la Y i l'estat dels botons (amb un 0.00 o un 1.00).

Enviament de dades:

Aquest apartat requereix una especial atenció ja que es tracta d'un dels més importants. Un cop s'han recollit les dades del joystick, cal disposar d'una manera per poder-les enviar a l'especialista per tal que sàpiga el que està passant en aquell moment.

Per enviar les dades, es programa un altre temporitzador de Java. Un dels problemes de l'enviament de les dades consisteix en que no es pot enviar a la mateixa freqüència que s'està llegint el dispositiu (10ms), per les característiques de la xarxa (un ADSL). Encara que s'envien poques dades, caldria enviar un nombre de missatges massa elevat perquè es puguin processar correctament. Per tant, cal trobar un equilibri entre el retard i el nombre de missatges.

Després de vàries proves, s'ha establert un valor de 250ms que es considera molt correcte. La connexió ho suporta perfectament, no genera un tràfic excessiu i es pot processar a temps. El contingut d'un missatge que va del pacient a l'especialista està format pels següents camps:

- X,Y del cursor
- X,Y d'on ha d'anar (el plàtan)

- X,Y d'on ha de portar el plàtan (la caixa)
- BotoActivat
- Nombre d'encerts (ho controla el pacient i l'envia a l'especialista)
- Temps actual, en el que hi ha hagut un encert
- IP del pacient (si es fa servir una connexió directa l'especialista es connecta al pacient)

Com es pot comprovar, un missatge només consta d'uns pocs enters i un String que és la IP. Per tant, el tràfic que es genera per missatges és mínim, i el que es controla més, és el nombre de missatges que es van enviant. Per 250ms, aproximadament, es pot generar un tràfic màxim de 1,1 KB/s si no es fa servir compressió.

Per enviar els missatges, s'ha implementat de dues maneres diferents. La primera consisteix en enviar els missatges a través del servidor de XMPP (ejabberd[15]). En canvi, la segona opció permet fer una connexió directa entre les dues màquines i enviar els missatges entre elles sense que intervingui el servidor.

El protocol XMPP és un protocol de missatgeria estandarditzat, robust i de codi obert basat en XML (anteriorment en Jabber). Es pot consultar la seva documentació sense cap problema. Hi han moltes aplicacions que el fan servir, la més coneguda és el Google Talk. La nostra aplicació fa servir aquest protocol, però a més a més existeixen vàries formes diferents a l'hora d'implementar-ho. La implementació feta servir són les llibreries SMACK [16], concretament la versió 2.0.0.

Aquest treball no explicarà el funcionament intern d'aquest protocol (per això ja hi ha extensa documentació), però sí que cal donar algunes nocions sobre el mateix. Una dada

important és que és descentralitzat, per tant, qualsevol persona es pot muntar el seu propi servidor de Jabber (port estàndard 5222). La idea més important és com s'organitzen els usuaris i servidors en el sistema. Cada usuari de la xarxa XMPP disposa d'un identificador únic (Jabber ID o JID). La seva estructura és com una adreça de correu, de l'estil nomusuari@domini.com. Els recursos són prioritats assignades a un valor numèric que es poden posar als usuaris, de la forma nomusuari@domini.com/recurs.

- Servidor: Per poder enviar els missatges a través del servidor, aquest només accepta missatges escrits en format XMPP. És a dir, en el fons és con si enviéssim missatges de xat però aquests no es mostren, sinó que es parsejen i serveixen pel tema de la representació gràfica del joystick.

  El següent codi ens mostra com fer un missatge que viatgi pel servidor:

```
1  import org.jivesoftware.smack.XMPPConnection;
2  import org.jivesoftware.smack.packet.Message;
3  import org.jivesoftware.smack.packet.Packet;
4
5  public static void EnviarMissatge(JoystickSession joy,
       String user)
6  {
7  XMPPConnection con =
8    BuddyList.getInstance().getConnection();
9  Message missatge = new Message();
10 missatge.addExtension(joy);
11 missatge.setType(Message.Type.NORMAL);
12 missatge.setTo(user);
13 con.sendPacket(missatge);
14 }
```

Notes sobre el codi:

1. Cal importar les classes XMPP de les llibreries SMACK.
2. Com que anteriorment ja ens hem connectat al servidor, es recupera aquest objecte creat al qual enviarem les dades.
3. JoystickSession, és l'objecte que conté els atributs que es volen enviar. Està definit amb camps XML.
4. El missatge és de tipus Normal, i se li passa l'usuari.
5. User és un String que s'importa de la classe BuddyList. Aquesta classe ens dóna l'adreça en Jabber del destinatari.

Queda clar que el pacient envia les dades capturades del joystick, però també ha de poder rebre dades de l'especialista ja que és ell qui li envia el tipus d'exercici, la dificultat... De la part de recepció es parlarà en l'especialista.

- Avantatges: Amb aquest sistema no cal saber cap adreça IP, ni s'ha d'obrir cap port al router ja que tot es fa a través del propi servidor.
- Inconvenients: L'ample de banda està limitat a la restricció establerta pel servidor. En les proves realitzades, es va haver d'ampliar el límit per defecte per tal que els missatges arribessin amb fluïdesa. També cal vigilar la freqüència del nombre de missatges que es van enviant per no saturar el servidor.

- Directa: Per poder realitzar la connexió directa, el quid de la qüestió consisteix en com obtenir la IP de l'altre equip. En l'estructura del plugin, el pacient fa les funcions de servidor i l'especialista de client. L'especialista s'ha de connectar al client i cal saber la seva adreça IP.

  Es van considerar les següents propostes per obtenir l'adreça externa:

- El propi servidor de Jabber ens ofereixi l'adreça IP amb qui parlem, ja que amb l'adreça JID no es pot realitzar una connexió directa.
- Fer servir un sistema STUN (*Simple Transversal of UDP over NATs*) per descobrir-la.
- Realitzar una consulta a una màquina externa i que ens retorni l'adreça.
- Agafar directament d'un fitxer de configuració la IP que ha inserit l'usuari prèviament.

Totes aquestes opcions, cal que se li enviï l'adreça a l'altra persona a través d'un missatge Jabber.

La primera opció sembla la més bona i fàcil d'utilitzar, ja que el propi servidor sap l'adreça correcta. Però segons la pròpia filosofia del XMPP, el servidor mai dóna cap adreça IP sota cap circumstància.

> "The IP address and method of access of clients MUST NOT be made public by a server, nor are any connections other than the original server connection required. This helps to protect the client's server from direct attack or identification by third parties."[17]

L'opció de la configuració manual ha de ser evitada, ja que és una complicació extra a l'usuari i aquest no s'ha de preocupar de res. En tot cas, es fa servir com a segon recurs si fallen les altres.

Descartada aquesta opció, es va plantejar STUN Vs. consulta externa. Existeixen vàries implementacions de STUN, per exemple en Java el JSTUN[18]. Es va decidir fer servir la consulta externa per aquests dos motius:

- Implementar un STUN és molt pesat, ja que cal fer moltes connexions per saber la nostra IP externa.
- Al fer moltes connexions, també tarda molt en descobrir-la, al voltant de 1 i 2 minuts. En canvi la consulta externa només requereix una sola connexió, i és molt més immediata.

Per demostrar aquests fets, es va fer un test real en un ADSL de Telefònica, analitzant tots els paquets necessaris amb els dos mètodes. En el propi servidor de Jabber es va muntar un breu script php que retornés l'adreça externa de qui la consultava.

```
<?php
if(isset($_SERVER["HTTP_X_FORWARDED_FOR"]))
{
//Proxy detectat
$ip=$_SERVER["HTTP_X_FORWARDED_FOR"];
}
else
{
$ip=$_SERVER["REMOTE_ADDR"];
}
echo $ip;
?>
```

Com es pot observar, també es fa comprovacions pel cas que el nostre ordinador estigui darrera d'un proxy. Aquest sistema no és 100% fiable, ja que si es vol, un pot enganyar la seva adreça IP, però en el nostre cas no té sentit, que el propi usuari vulgui saber la seva ip i alhora l'estigui intentant ocultar.

Sobre el test, en STUN va realitzar 20 connexions i va trigar 20 segons en saber la ip externa. En canvi, la connexió al servidor extern només necessita fer 1 sola connexió (un GET d'una web HTML), i triga menys d'un segon.

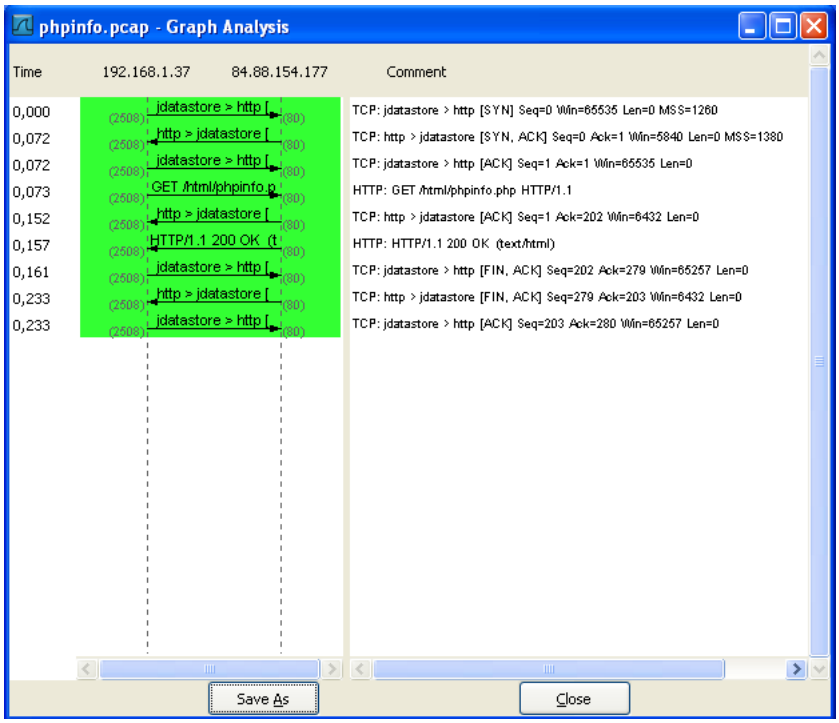

Figura 3.5: Diagrama de flux de la connexió directa al servidor

Un cop establertes les opcions disponibles, per realitzar la connexió directa:

- A través de missatges XMPP, l'especialista li demana la IP al pacient.
- El pacient (servidor) realitza la consulta, i li envia aquesta IP a l'especialista.
- L'especialista es connecta a través d'un socket de Java.

```
1 import java.io.*;
2 import java.net.*;
3
4 public static void ConnectarDirecte(String ip,int port){
5 try {
6 Socket echoSocket=new Socket(ip, port);
```

```
7  PrintWriter out = new PrintWriter(echoSocket.
       getOutputStream(), true);
8  BufferedReader in = new BufferedReader(new
       InputStreamReader(
9  echoSocket.getInputStream()));
10 }
11 catch (UnknownHostException e){
12 System.err.println("Error en saber el host");
13 }catch (IOException e) {
14 System.err.println("Error de I/O en la connexió.");
15 }
16 }
```

- L'especialista li envia l'exercici al pacient.
- Espera a rebre dades del pacient per mostrar-les. L'especialista pot parar l'exercici quan vulgui.

Representació gràfica:

Encara que la representació de les dades es podria fer en forma de nombres, no és tan intuïtiva com fer una representació gràfica del moviment del dispositiu. Per fer-ho, s'han fet servir les llibreries AWT Graphics del propi Java. Aquestes disposen d'objectes 2D els quals es poden pintar directament al JLabel (el panell que es mostra).

Un truc que s'ha fet servir és el tema de la detecció d'objectes (quan el mico cursor "toca" el seu objectiu). Les llibreries AWT disposen d'un mètode intersect, que comprova si dos objectes s'estan tocant. La idea feta servir és que es dibuixen les imatges, però aquestes en realitat ocupen l'espai d'un rectangle. És a dir, el quadrat hi és en el Panel,

però no es pinta. Llavors tota la detecció es fa a partir dels quadrats, i fa la sensació que realment és el mico qui toca els plàtans. Per ajudar al pacient, a cada encert s'ha afegit un reforç visual i un de sonor. Per exemple, en arribar el mico al plàtan, es mostra una mà en forma de OK, i es sent un "dong" afirmant l'encert.

Parlant del mico i els plàtans, totes les imatges són GPL, i algunes d'aquestes s'han extret del Tango Project [19]. El projecte Tango ofereix un conjunt d'icones força estàndards per fer-les servir, i varis programes com els SO les fan servir.

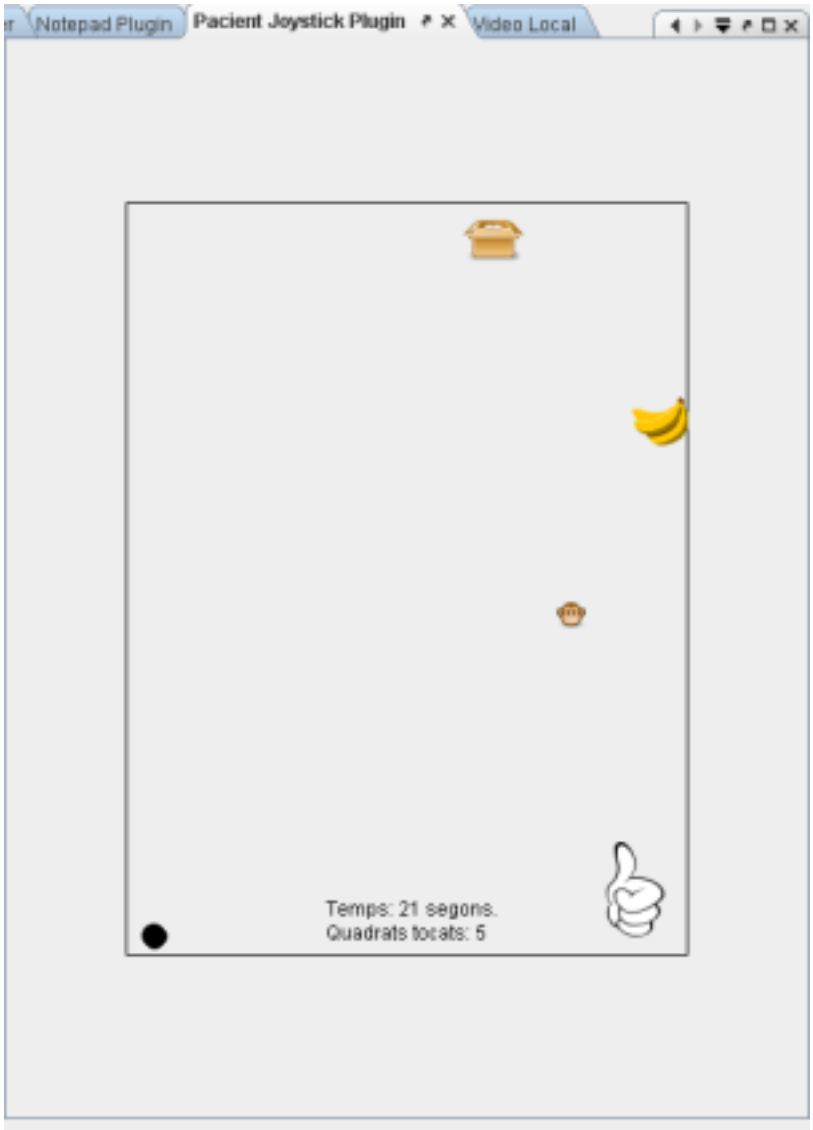

Figura 3.6: Representació gràfica de l'exercici per part del pacient.

3.4.2 Part de l'especialista

Adquisició de dades:

La part de l'especialista no ha de llegir cap dispositiu, per tant no existeix.

Recepció de dades:

En la part del pacient s'ha parlat de les dades que enviava el pacient, però l'especialista també li envia dades al pacient:

- Dificultat de l'exercici (varis nivells).
- Tipus d'exercici (si ha d'anar a un lloc, arrossegar un objecte, ...).
- Les posicions es defineixen aleatòriament o manualment.
- Parar un exercici (1 o 0).

A l'hora de rebre missatges, hi ha dues opcions possibles. La primera és fer un polling i anar demanant nous missatges cada interval de temps. La segona, és a través d'interrupcions. Es considera millor la segona opció, ja que evites gastar recursos inútilment intentant endevinar si s'han rebut. Tant de la connexió a través del servidor, com de la directa estan implementades amb les interrupcions.

- Servidor: Es programa un filtre de paquets (en aquest cas, els nostres paquets del joystick), el qual quan en rep un, dispara un mètode. Aquest mètode s'encarregarà de cridar a les funcions de pintar la pantalla i actualitzar l'estat del dibuix.

```
1 import org.jivesoftware.smack.PacketListener;
2 import org.jivesoftware.smack.XMPPConnection; import
3 org.jivesoftware.smack.packet.DefaultPacketExtension;
```

```
4  import org.jivesoftware.smack.filter.PacketExtensionFilter;
5  import org.jivesoftware.smack.packet.Message;
6  import org.jivesoftware.smack.packet.Packet;
7
8  public static void RebreMissatge(final MyPanel panell){
9  XMPPConnection con = BuddyList.getInstance().getConnection
       ();
10 PacketListener listener = new PacketListener(){
11 public void processPacket(Packet packet){
12 final Message message = (Message) packet;
13 DefaultPacketExtension joy =
14   (DefaultPacketExtension) message.getExtension
15      ("xsow", "joysticksession");
16 if (joy != null){
17 //Fem les accions d'actualització...
18 }
19 }
20 };
21 PacketExtensionFilter filterExt = new PacketExtensionFilter
       ("xsow", "joysticksession");
22 con.addPacketListener(listener, filterExt);
23 }
```

Notes: Mypanel és la classe gràfica de representació. Els passos que es realitzen són:

1. Recuperem la connexió XMPP.
2. Es crea un PacketListener, una classe per escoltar paquets que es reben.
3. Definim el procés que executarà la interrupció. Aquí dins, el packet el definim

com un DefaultPacketExtension. Com que un missatge d'aquest és un XML, es poden recuperar fàcilment els seus camps i fer les actualitzacions.

4. Al final de tot, després de definir el mètode, s'afegeix a la connexió el PacketListener del tipus joysticksession (els nostres paquets).

- Directa: Per la connexió directa, cal crear un nou thread encarregat de la lectura de missatges. El mètode per rebre en un socket (readLine()) és bloquejant i fins que no rep un missatge, no segueix. Per tant, fent un nou thread funcionarà perfectament i serà a mode d'interrupció.

```
import java.io.*; import java.net.*;

public void run(){
String inputLine;
while (true){
try {
inputLine = in.readLine(); //mètode bloquejant!
//Fem l'acció que volguem actualitzar, ex:
panell.setencerts(LlegirEtiquetaXML(inputLine,"quadrats"));
}catch (NullPointerException e) {
//Encara no s'ha establert la connexió
}catch (IOException e) {
System.err.println("Error de I/O en la connexió.");
}
}
}
```

Representació gràfica:

En la part de l'especialista, sí que pren una major importància a l'hora de mostrar els moviments rebuts. Cal pensar que és possible que els paquets amb la informació de les coordenades es perdin, arribin tard, o amb desordre. Una mesura que s'ha adoptat és un control de paquets. Cada paquet va numerat, i només s'accepten els paquets següents a l'últim. Si arriben paquets anteriors, no ens interessen ja que el cursor ja s'haurà mogut a una posició més avançada.

A més a més, no només hi ha una sola forma de representar el moviment. Es pot observar en temps real o fent una animació. A continuació es parlarà dels pros i contres.

- Temps real: Tant bon punt arriben les coordenades, es mostren a la pantalla. El moviment és més real amb el que està passant, però també es pot perdre el sentit del moviment si es mou ràpid el joystick degut als intervals de temps. S'observa com si el cursor saltés d'un costat a l'altre de la pantalla.

- Animació: A les dades se li afegeix un retard, però el moviment entre dos punts s'interpola i es dibuixa un moviment continu. En altres paraules, entre que es reben unes coordenades i les següents hi ha un interval de temps. Durant aquest interval, es dibuixa un moviment rectilini entre dues coordenades guardades.

  La fórmula per calcular aquest moviment és senzilla:

  $x = \alpha * x_0 + (1 - \alpha) * x_1$

  $y = \alpha * y_0 + (1 - \alpha) * y_1$

  On alfa és un interval de temps normalitzat [0,1], Xo i Yo el punt d'origen i els altres dos punts el destí.

  Aquesta animació resulta un moviment més fluid per l'ull humà i més agradable.

Cal recordar, que el que es representa és una **aproximació**, no un moviment exacte.

També es poden realitzar altres sistemes d'animació en comptes del lineal. El problema del lineal consisteix en que els moviments amb corbes no els interpreta bé, o moviments molt petits (un exemple clar seria un pacient amb Parkinson). Altres sistemes d'animació, per exemple, serien els splines de tres punts. Encara que no seria molt difícil programar-ho, creiem que el moviment lineal és suficient per una bona aproximació.

3.4.3 Altres

En aquest apartat, es parlarà d'aspectes que no pertanyen a cap d'aquests tres punts generals, però que incorpora el nostre plugin.

- **Finestra de configuració:** Tal i com s'ha comentat anteriorment, en el nostre plugin es pot afegir una pantalla dins la configuració del programa perquè l'usuari pugui triar entre vàries opcions. Aquestes dades, es guarden en un fitxer de text dins la carpeta del seu perfil (*Profile*). En conseqüència, un plugin pot disposar de les seves pròpies configuracions.

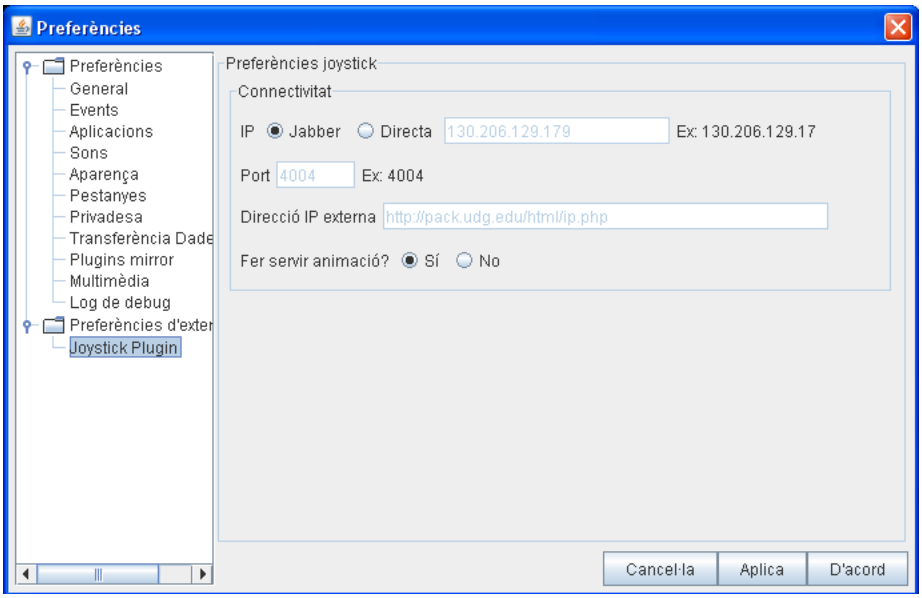

Figura 3.7: Diàleg de les preferències de l'especialista

La finestra mostra si es fa servir una connexió directa o a través del servidor. En cas de la directa li permet configurar la connexió i també pot activar o desactivar l'animació. Aquestes opcions es tenen en compte abans d'arrancar la conversa amb l'altre contacte.

- **Temps:** Tots els exercicis es cronometren per saber quan tarda el pacient a fer les activitats proposades. Per realitzar el temporitzador, s'ha creat un altre thread de

Java que s'encarrega cada segon d'actualitzar la variable del temps, i repintar el gràfic per tal que es mostri.

- **Estadístiques de l'exercici:** L'especialista guarda per cada exercici realitzat unes estadístiques. Es troben en format txt dins el seu userDir, en la carpeta resultatsJoystick:
  - Tipus d'exercici
  - Temps que ha durat l'exercici
  - Nombre d'encerts
  - Dificultat de l'exercici
  - Mitjana d'encerts
  - Desviació estàndard

- **Múltiples idiomes**: Tant la part del pacient com la de l'especialista, el plugin llegeix la configuració general de l'aplicació i obté l'idioma que s'està fent servir actualment. Es mostren els missatges en l'idioma actiu. Un plugin pot disposar de varis fitxers que contenen les cadenes de text que fa servir, un fitxer per cada idioma que es vol traduir. El propi Java disposa d'una classe específica per la internacionalització d'una aplicació (i18n) que és la classe Resources.

- **Compressió de dades:** Una característica que es volia implementar a l'hora de crear aquest plugin d'exemple, era la possibilitat d'enviar dades comprimides. En el cas concret del joystick, el tràfic que s'envia és molt petit, però pot resultar molt interessant en futures aplicacions. En Java, hi ha vàries possibilitats per enviar dades xifrades que s'explicaran a continuació, i quina és la opció escollida.
  - El propi Java disposa de llibreries per tractar fitxers comprimits (java.util.zip). El seu ús és senzill, però té dos problemes greus:

    * Les llibreries funcionen bé per fitxers (és a dir, que el tamany sigui conegut), però pels sockets el seu rendiment és molt dolent, i els enviaments no són optimitzats. Es poden fer servir les subclasses GZIPInputStream (i Output).
    * La compressió està bé, però no és de les millors. Per enviament de missatges en mode Directe, a 50ms i sense compressió, el joystick consumia cap a 6KB/s. Amb la compressió de Java 3,3KB/s.
    * El més important, era el retard que s'afegia per falta de temps a descomprimir. Quan s'envien cada 50ms, s'ha vist que per la descompressió de les dades, l'aplicació no respon amb la fluïdesa desitjada.
  - En canvi, existeix una llibreria anomenada Jzlib[20] que és gratuïta (BSD licence) i que ofereix millors resultats que la pròpia de Java. Aquesta llibreria, sí que està pensada per l'enviament comprimit de dades per sockets, i està més desenvolupada en detalls tècnics que la de Java. En les mateixes condicions, el tràfic va baixar fins a 0,24Kb/s i un retard menor que la de Java (evidentment, sí que es nota un pèl més de retard que si no comprimíssim les dades).

Un dels millors aspectes de la compressió és que és transparent al codi existent. En TCP, només cal afegir una capa a sobre del stream d'entrada i de sortida:

```
1 import java.io.*; import java.net.*;
2 import com.jcraft.jzlib.*;
3
4 static ServerSocket serverSocket = null;
5 static Socket clientSocket = null;
6 static PrintWriter out = null;
7 static BufferedReader in = null;
8
9 static class EscoltarTCP extends Thread {
```

```
10  //[...]
11  public void run() {
12     try {
13        clientSocket = serverSocket.accept();
14        // Construim un objecte Lectura que envolta el flux d
             'entrada del socket (descomprimit).
15        ZInputStream zIn=new ZInputStream(clientSocket.
             getInputStream());
16        in = new BufferedReader(new InputStreamReader(zIn));
17        // Fem un flux que comprimeix les dades a enviar.
18        ZOutputStream zOut=new ZOutputStream(clientSocket.
             getOutputStream(), JZlib.Z_BEST_SPEED);
19        zOut.setFlushMode(JZlib.Z_PARTIAL_FLUSH);
20        // Construim un objecte Escritura que envolta el flux
              de sortida del socket (comprimit)
21        out = new PrintWriter(new OutputStreamWriter(zOut));
22        } catch (UnknownHostException e) {
23        System.err.println("No coneixem el host.");
24        } catch (IOException e) {
25        System.err.println("Fallada d'acceptació.");
26        }
27     }
28  }
```

A partir dels streams d'entrada i sortida, es construeixen els objectes de les llibreries de la compressió, i a sobre d'ells ja van els objectes que es fan servir normalment com si no es fes una compressió. El codi per enviar i rebre missatges és exactament igual, tant si apliquem la compressió com no, només han canviat dues línies noves.

- **TCP o UDP:** Un cop implementat l'enviament de missatges a través del servidor de XMPP o una connexió directa, i l'opció d'enviar les dades comprimides en el cas de la connexió directa, un altre pas útil de cara a futurs plugins és la possibilitat d'enviar els missatges per un protocol TCP o UDP. Com a curiositat, la compressió no es pot aplicar a UDP ja que no es tracten fluxos de dades (es pot perdre informació pel camí).

3.5 Altres plugins implementats

Encara que l'extensió principal per veure les possibilitats que pot oferir la creació dels plugins és el control del joystick, també s'han adaptat altres funcionalitats en forma de plugins. A continuació s'explicaran cada una d'elles sense entrar en molta profunditat.

3.5.1 Bloc de Notes

El bloc de notes és una eina útil de cara al perfil de l'especialista. Tal i com indica el seu nom, es mostra un senzill editor de text en el que el doctor pot agafar notes sobre el pacient. Permet editar tant text en pla, com text en format HTML (es poden afegir enllaços a un document, crear taules, ...). Es pot considerar més avançat que el famós Bloc de Notes del Sistema Operatiu Windows.

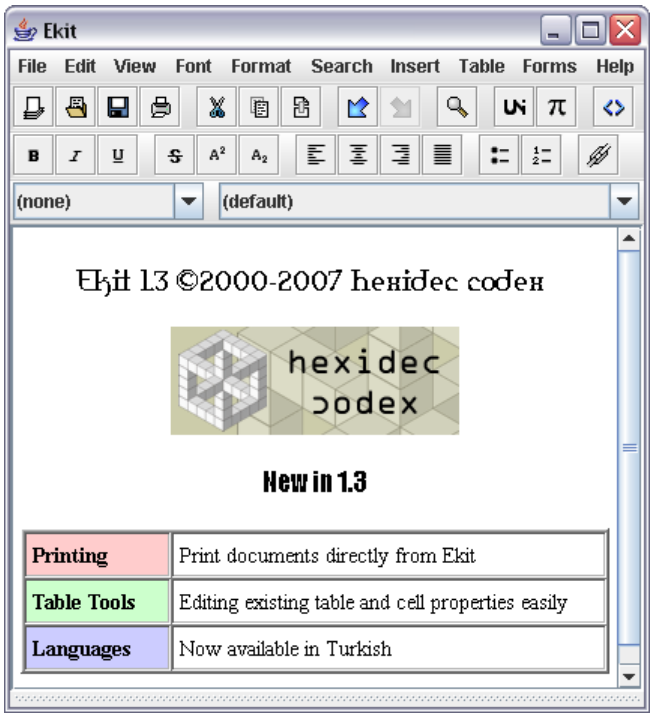

Figura 3.8: El bloc de notes en acció (imatges, taules, colors...)

Per afegir aquesta funcionalitat, s'ha fet servir l'aplicació en Java Ekit.

- Editor de text versàtil amb moltes opcions. Permet posar estil al text, guardar fitxers i fins i tot imprimir-los.
- Codi font disponible i lliure, lleuger i ocupa poc espai en disc.
- Es pot adaptar fàcilment i suporta varis idiomes. Incorpora un motor de diccionari anomenat Jazzy.

3.5.2 Llibreria Multimèdia

El plugin de la llibreria multimèdia està subdividit en tres parts ben diferenciades, cadascuna d'elles aporta una nova funcionalitat:

- **Explorador de fitxers:** Una funcionalitat molt útil de cara a l'especialista consisteix en l'organització dels fitxers que es van emmagatzemant. A la finestra es mostra un explorador de fitxers propi per tal que trobi més fàcilment les dades emmagatzemades (converses amb un pacient, les notes amb el bloc de notes, les fotos i vídeos d'una sessió de videoconferència, dades d'un exercici d'un plugin i altres). Permet navegar d'una manera còmoda a través del directori de carpetes i es poden obrir els fitxers en el mateix programa. Les converses i les notes les obrirà amb el plugin del Bloc de Notes, si es troba disponible, (en cas contrari no les podrà obrir). També pot reproduir els vídeos guardats, i mostrar les fotos fetes amb el programa.

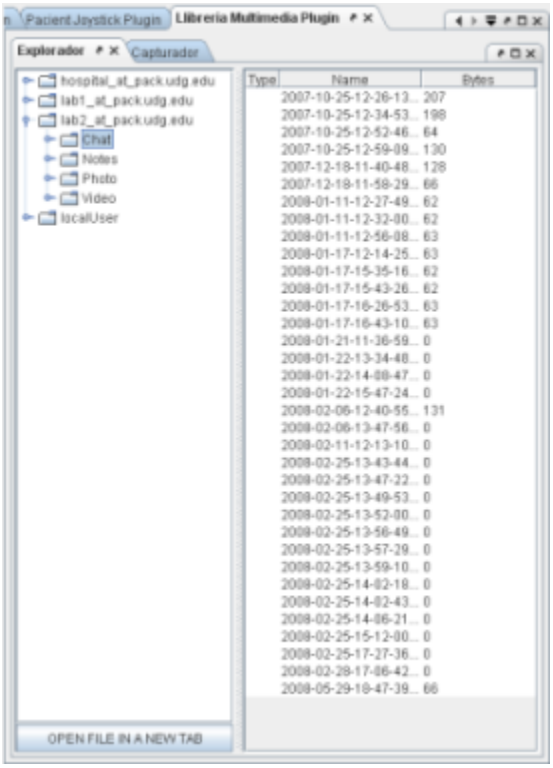

Figura 3.9: L'explorador de fitxers, amb converses entre un pacient i el seu especialista

- **Capturador:** Un requeriment que ens van demanar des de la Fundació, durant el projecte TRiEM, era la possibilitat de fer servir la webcam en mode local. En aquesta part, el pacient es pot fer fotos a ell mateix amb la webcam i gravar-se en qualsevol moment. Una característica d'aquest mode és que les fotografies que es fan són a la màxima resolució possible de la càmera, a diferència de les fotografies en una sessió de videoconferència que són a la resolució 352x288. Un pacient pot fer-se una foto d'una zona en concret i després enviar-la a l'especialista, que la podrà analitzar millor.

- **Transferència de fitxers:** Una nova funcionalitat afegida recentment és la capacitat de poder enviar fitxers entre el pacient i l'especialista. Aquest pas representa una base per futurs exercicis ja que la podran aprofitar. Tal i com s'ha comentat anteriorment en l'apartat "Requeriments del servidor", s'ha muntat un servidor de SFTP,

en el qual el pacient pot enviar fitxers al servidor i, després l'especialista se'ls pot baixar. Aquest sistema té els seus punts forts i febles:

- Punts a favor:
  - Asíncron: El punt més important, evita que les dues parts estiguin connectades alhora i dóna més flexibilitat a les dues parts.
  - Centralització: En el mateix servidor es fan totes les funcions, el qual simplifica complicacions.
  - Ample de banda de pujada: Al poder fer la tasca en qualsevol moment, s'evita possibles colls d'ampolla a l'hora d'enviar fitxers.
- Punts en contra:
  - Seguretat: Al proporcionar contes SFTP al servidor, cal disposar d'un bon sistema de seguretat tant en els accessos, com en els permisos. És possible que la informació sigui sensible, i per tant, cal tindre present temes de xifratge, versions actualitzades...
  - Escabilitat: Quan s'afegeix enviament i rebuda de dades al servidor llavors cal mesurar l'impacte en el seu ample de banda per varis usuaris.

Per poder fer una connexió SFTP, cal fer servir unes llibreries en Java que permetin treballar amb SSH. Existeixen moltes llibreries (tant de codi obert, com tancat), però en aquest cas s'ha fet servir la llibreria j2ssh (altres exemples: Jsch, JSSH, ...), perquè és fàcil de fer servir, i disposa de moltes aplicacions (una d'elles és un client de Sftp). J2ssh es troba dins del paquet d'utilitats SSHTools[21].

A pesar de disposar de les llibreries, aquestes no proporcionen cap interfície gràfica per treballar-hi amb elles. No obstant, s'ha implementat tota una interfície gràfica

que resulta ser el més simplificada possible de cara a l'usuari. Segons si és un pacient o un especialista, es comporta d'una forma diferent... Abans, però, cal parlar sobre el tema dels usuaris i permisos.

En principi, el pacient només podrà enviar fitxers al servidor (dades, imatges...). Cada pacient disposarà de la seva carpeta pròpia, i a més a més, el pacient no es podrà moure d'aquest directori. Llavors, l'usuari pacient serà un usuari engabiat dins el seu propi directori.

En canvi, els especialistes han de poder veure als seus pacients. Queda, per determinar, si un especialista pot veure a tots els pacients o no, però aquest aspecte és un requeriment per definir. Un especialista pot disposar d'una carpeta personal, però la funció principal serà anar a les carpetes dels pacients i podrà descarregar els fitxers.

Finalment, queda pendent el tema de l'eliminació dels fitxers del servidor. Es poden eliminar automàticament després de descarregar el fitxer, o que el servidor els elimini després d'un nombre de dies.

3.5.3 Videoconferència

L'últim plugin que descriurem és la part de la videoconferència. Sense cap mena de dubte, és un dels plugins més útils en la comunicació pacient-especialista. Tal i com indica, permet establir una sessió de videoconferència entre les dues parts. A la finestra principal es pot observar dues parts (Vídeo Local i Vídeo Remot). A la finestra local es pot veure la imatge que està captant la pròpia webcam local i en la remota, la de l'altra persona. Es pot activar un dels dos vídeos, o tots dos alhora.

No s'explicarà a fons el mecanisme de funcionament de la videoconferència, ja que no és part de la feina realitzada en aquest projecte. Si se'n vol saber més, es pot consultar el projecte final de carrera del "Projecte TRiEM"[22] realitzat per en Carles Guadall.

Anteriorment ja hem comentat que es feien servir les llibreries JMF[6]per oferir la videoconferència. Les llibreries ofereixen manipular àudio i vídeo en una aplicació Java. Permeten capturar d'un dispositiu físic o lògic, reproduir, enviar i rebre fluxos multimèdia, i recodificar varis formats. Com s'ha anat veient, moltes d'aquestes opcions les fem servir. El principal problema és que les llibreries no han estat actualitzades des de 2004 per part de Sun Microsystems. Encara que funcionen bastant bé, tenen varis defectes (sincronització, errors inesperats, pocs formats reconeguts, etc). Un dels problemes més importants és que **reconeix pocs dispositius de captura.**

Una possible alternativa que fa relativament poc que ha sorgit és una nova llibreria multimèdia anomenada FMJ[7]. El gran atractiu d'aquesta nova llibreria es que és API-compatible amb JMF. En teoria, només substituint la llibreria JMF per la FMJ hauria de funcionar tot sense tocar una sola línia de codi. La realitat és que encara està molt lluny d'aquest fet, i que la llibreria està verda. És cert que la captura de dispositius, sí ha millorat amb FMJ, però encara li queden moltes coses a implementar (no suporta el format

H263/RTP, no disposa del mòdul per enviar fluxos RTP, processament gràfic molt pobre, només deixa triar un format de captura, ...). Actualment no serveix per fer videoconferència.

Un cop observats els plantejaments de JMF i FMJ, hi ha encara una solució intermèdia combinant les dues llibreries. La llibreria FMJ fa servir unes altres llibreries, anomenades LTI-CIVIL[23]. Gràcies a aquestes, permeten capturar moltes més càmeres en entorns Linux (V4L2) i Mac Os X. La solució proposada consisteix en:

1. La webcam és capturada per la llibreria LTI-CIVIL i s'obté una font de dades.
2. La llibreria FMJ (sense la part gràfica) és encarregada de convertir les dades de LTI-CIVIL a un format Datasource.
3. El format Datasource és l'entrada a la llibreria JMF, que llavors s'encarrega de tota la feina de comprimir, enviar...

Cal fer un punt i a part pel tema del Mac Os X i JMF, perquè la llibreria no és compatible amb la càmera integrada que porten els Macbook (els ordinadors portàtils de la casa Apple). Encara que de forma nativa la videoconferència en Mac Os X és parcial (només pot rebre vídeo), existeix una solució alternativa.

Es pot instal·lar una màquina virtual que emuli, per exemple, un Windows XP. Llavors s'instal·la el Java, JMF i la nostra aplicació en aquesta màquina virtual i la càmera funciona. S'ha realitzat aquesta prova amb Parallels, un programa de virtualització específic de Mac Os X i que suporta drivers específics per la càmera integrada. El resultat de la prova ha estat positiu i, per tant, una alternativa si es disposa d'aquest ordinador. Amb VMWare Fusion (un altre programa que emula una màquina virtual) també funciona

la càmera, però té més problemes amb el so del sistema (aquest problema és degut a la virtualització).

Capítol 4

Proves i resultats

En aquest capítol, farem una recopilació de les dues proves reals que s'han realitzat i es resumirà l'experiència treta de cadascuna d'elles.

27 maig 2008: Es va realitzar una prova real entre la Universitat i l'Hospital de Dia de la FEM. Totes dues bandes disposen d'una connexió ADSL estàndard. L'ordinador de l'hospital havia estat totalment formatejat i net degut que en les últimes proves realitzades semblava que era una font de problemes. Es va instal·lar l'aplicació des de zero, i també la descàrrega dels plugins. El resultat va ser molt bo, ja que tot va funcionar a la primera sense masses complicacions. També es va provar per primer cop, el plugin del joystick en temps real (sense animació) a veure com es comportava amb la videoconferència alhora. El resultat també va ser positiu, i es va poder veure com un pacient (jo) controlava l'aparell i feia l'exercici.

29 maig 2008: Es va realitzar una altra prova com l'anterior. En la màquina de la Universitat, es va voler provar una nova videocàmara (Logitech QuickCam 5000) per veure si oferia més qualitat que amb la de les proves anteriors. Però un dels problemes que es va trobar era que els controladors de la càmera no estaven correctament instal·lats i, per

tant, no funcionava amb l'aplicació. Després d'una certa demora, es va optar per posar la càmera anterior i la connexió va ser un èxit. Un dels detalls de la sessió, va ser que hi havia un fort retard entre l'àudio i el vídeo (aproximadament d'un segon) ja des dels primers minuts de la videoconferència. Malgrat això, el moviment era força fluid amb una qualitat mitjana degut a la limitació de l'amplada de pujada de la línia.

Demostracions: A més a més de les proves, s'han realitzat moltes demostracions a varis visitants, interessats per aquest projecte de telemedicina. Des de la pròpia Fundació Esclerosi Múltiple, la Fundació i2CAT (de la Universitat Politècnica de Catalunya), una empresa farmacèutica, i també doctors d'altres universitats que han visitat el grup BCDS de la Universitat de Girona (Andrzej Jajszczyk i Ewald Graif).

També cal destacar, fruit del treball aconseguit d'aquesta remodelació dins l'aplicació AXARM, la creació d'un "paper" (article acadèmic) el qual s'ha enviat a un congrés sobre medicina a Londres (eHealth 2008) titulat: *AXARM: An Extensible Remote Assistance and Monitoring Tool for ND Telerehabilitation* i ha estat **acceptat** per fer la seva presentació, durant el dies 8 i 9 de Setembre de 2008.

Capítol 5

Conclusions

Com s'ha pogut observar en aquesta aplicació i en tot el projecte que l'envolta, la feina no està ni molt menys finalitzada. En aquest treball, s'ha resumit un període des del gener del 2008 fins l'actualitat, però encara queda molt de camí per fer. S'han complert els objectius establerts pel projecte i m'agradaria destacar, a mode de resum final, totes les característiques i aspectes positius que proporciona l'aplicació:

- L'aspecte més important és que l'aplicació és real: està implementada, testejada i no és només un prototipus. Es podria començar la seva distribució en poc temps.
- Al ser una aplicació real, s'han realitzat proves pilots reals. Aquest aspecte reforça molt l'experiència en l'entorn i dóna una important realimentació al programador.
- El possible cost per l'establiment i desplegament del sistema és molt baix (per cada pacient caldria un ordinador amb videocàmera, micròfon, cascs i una connexió ADSL). A més a més, permet ser molt escalable.
- El conjunt dels punts anteriors, fa que l'aplicació disposi de grans perspectives de futur. Encara que l'àmbit de possibles interessats és reduït (institucions públiques, i potser alguna empresa en concret), no significa que es sofreixi un augment enorme en aquest àmbit de la telemedicina i teleassistència en un futur proper.

- Sobre l'aplicació, destacar el seu control asíncron. Això significa que l'especialista permet controlar moltes accions sense dependre del pacient. Per exemple, l'acció més clara és engegar les càmeres locals i remotes de la videoconferència.

També hi ha punts en contra:

- Hi ha limitacions d'estructura en el sistema. Una de les més importants, és la limitació de les línies ADSL. Actualment, el límit de pujada d'un ADSL és de 300 kbps, i això afecta molt a la videoconferència ja que cal enviar àudio i vídeo a través d'ella. Per això s'ha de controlar amb una qualitat d'imatge baixa, i intentar una tassa de fps (imatges per segon) constant (al voltant dels 15-20 es considera suficient).
- A pesar d'intentar oferir una interfície simplificada, queden alguns passos difícils de resoldre pel propi pacient. Un d'aquests és el tema d'obrir una sèrie de ports en el router de casa del pacient. La dificultat de l'usuari novell en aquests temes, afegit a la gran gamma de routers diferents que existeixen en el mercat, fa que no sigui una tasca senzilla de fer. A favor, hi ha molta informació disponible a Internet sobre aquest tema, i que només cal realitzar-ho un cop durant la instal·lació del pacient.
  - En aquests últims anys, per evitar ocupar totes les adreces IP d'una manera ràpida (IPv4 té només 2^32 adreces disponibles), es va crear l'invent de les NAT (Network Address Translation) que permet disposar de varis ordinadors connectats a Internet amb una sola adreça externa. Degut a aquesta tipologia de xarxa, cal distingir els ports de les adreces internes de la xarxa, i per aquesta raó sorgeix aquest problema. Alguns routers són UPNP (Universal Plug and Play) i permeten que l'aplicació obri els seus propis ports, però lamentablement no tots ho incorporen.

Una pregunta clau que es pot plantejar un cop vista l'aplicació, és: què beneficia respecte a altres sistemes de videoconferència, com el Messenger o Skype? Bé, doncs en alguns aspectes que remarco a continuació:

- Asíncron: Una característica important que s'aplica a la videoconferència i en altres camps és la capacitat que l'especialista pugui controlar tots els processos del pacient, per tal de facilitar-li al màxim la seva utilització. L'especialista pot engegar la videoconferència sense que el pacient hagi de fer absolutament res.
- Codi obert: El codi obert ofereix més beneficis que perjudicis, ja que sol ser un codi que ha estat provat i que altres membres el poden millorar, o ajudar a que funcioni d'una manera eficient.
- Modificable: Disposar de tot el codi font permet disposar d'un control total de l'aplicació i no haver de dependre d'altres factors externs.
- Gravació de vídeo: A diferència d'altres programes, totes les converses es poden gravar o fer fotografies.
- Extensible: És el punt final que s'ha millorat amb aquest treball. La facilitat d'afegir noves funcionalitats amb una interfície de plugins requereix un temps molt menor que si s'hagués de tocar tot el codi font.

No ens oblidem de les aplicacions reals que pot generar l'aplicació. Pel pacient, li pot suposar una millora de la seva qualitat de vida, ja que haurà de desplaçar-se menys cops de casa. Es poden reduir costs i temps tant en el pacient com en el centre d'atenció mèdica. Gràcies a l'avenç de la tecnologia, estic convençut que la tendència de la telemedicina seguirà en augment i que cal apostar-hi per ella.

Finalment, a títol personal, l'esforç que he fet ha estat molt positiu, ja que m'ha permès ampliar els meus coneixements d'informàtica. A nivell de programació he hagut de treballar amb un codi font d'altres companys, la qual cosa ha estat una continuació de la seva tasca. També s'ha mantingut una col·laboració constant entre la part tècnica (UdG) i la part mèdica (Fundació) i he implementant noves funcionalitats. Alhora, els meus coneixements en Java han augmentat significativament (ús de varies llibreries, utilització de les classes del Java, eines de compilació, etc). A més a més, treballar en el projecte m'ha servit per explorar la meva faceta com a investigador, ja que aquest és un projecte molt lligat a la investigació. Per tot això, el projecte m'ha estat un complement molt important dels estudis efectuats durant els meus anys de la carrera d'Enginyeria Informàtica.

Capítol 6

Treball futur

Una de les característiques d'aquest projecte és la seva constant renovació. Freqüentment apareixen noves idees i propostes, el qual significa adaptar les noves funcionalitats a l'aplicació. A diferència d'altres programaris, no existeixen uns requeriments fixes però es marquen vàries metes a complir. La feina es classifica en funció de si és a curt termini (més prioritari), o a llarg termini.

A curt termini, de cara a finals de 2008 el treball es centrarà en aquests punts:

- Nous exercicis: De cares a facilitar la relació entre pacient i especialista, s'estan desenvolupant noves idees per fer amb l'aplicació. Al disposar dels sistemes de plugin, la seva incorporació no hauria de ser més complicada que desenvolupar el propi exercici en sí. Algunes idees d'exercicis impliquen un reconeixement de paraules (per tant, caldrà fer servir un motor de diccionari), o relacionar imatges entre sí.

  En aquest grup, també entren les idees de fer servir els nous dispositius dels que se n'ha parlat anteriorment (la catifa de ball o d'altres).

- Exercicis asíncrons: Una característica desitjable consisteix en que un especialista prepari una sèrie d'exercicis, i li enviï al pacient perquè els faci. Aquest, en qualsevol moment que li vagi bé, els pot fer. Un cop els hagi fet, el sistema enviaria els resultats a l'especialista i li notificaria nous resultats a examinar.

  Actualment, l'activitat del joystick és síncrona (calen que els dos estiguin connectats alhora). Fer una activitat asíncrona si es vol fer servir la idea del joystick no hauria de ser més complicada que la que s'ha fet per tal que sigui síncrona.

  - Sobre l'apartat de com l'especialista pot enviar exercicis al pacient, s'estan observant vàries possibilitats. La més senzilla és crear una web modificable per l'especialista, que li comuniqui al pacient i el programa agafi les dades. Però una idea més interessant és fer servir el servei *XEP-0060 Publish-Suscribe* del XMPP. En poques paraules, es creen uns "nodes" en que els usuaris s'inscriuen. D'aquesta manera, es poden fer combinacions d'especialistes i pacients. Un especialista pot enviar un missatge a un "node", i aquest el reenvia als seus usuaris inscrits (d'aquesta part se n'encarrega el propi servidor de XMPP).

- Aplicació estable: Encara que l'aplicació és força estable, es vol aconseguir disposar d'una versió sòlida, per tal que pacients i especialistes la puguin fer servir.

- Simplificar la interfície d'usuari al mínim: Un aspecte important és facilitar al màxim el seu ús tant a pacients, com a especialistes, per tal que es sentin còmodes amb ella. Els temes de configuracions més complicades com donar d'alta a usuaris, o la instal·lació i posada a punt dels equips ho podrà realitzar un tècnic. Una feina futura a fer és crear una interfície web que simplifiqui el procés actual de les configuracions del servidor.

- Proves reals: Es vol fer un estudi de l'ús d'aquesta aplicació amb uns 10 pacients reals. A llarg termini aquest nombre es pot veure augmentat.

- Noves funcionalitats: A més dels exercicis, nous requeriments que siguin útils en l'aplicació.

Bibliografia

[1] Universitat de Girona. BCDS - Broadband Communications and Distributed Systems. http://bcds.udg.edu.

[2] Fundació Esclerosi Múltiple. FEM. http://www.fem.es.

[3] Olsen, A. JBother - A Groovy Jabber Client. http://www.jbother.org.

[4] XMPP Standards Foundation. XMPP. http://www.xmpp.org.

[5] Internet Engineering Task Force. IETF. http://www.ietf.org.

[6] Sun Microsystems. JMF - Java Media Framework. http://java.sun.com/javase/technologies/desktop/media/jmf.

[7] Larson, K. et al. FMJ - Freedom for Media in Java. http://fmj-sf.net.

[8] Sébastien, T. i Pierre, M. MySecureShell. http://mysecureshell.sourceforge.net.

[9] Duche, G. WiiuseJ - Java API for Wiimotes. http://code.google.com/p/wiiusej.

[10] Diamond, M. WiiremoteJ. http://www.wiili.org/index.php/WiiremoteJ.

[11] Fritzsch, V. motej. http://motej.sourceforge.net/.

[12] Saint-Andre, P. i Hildebrand, J. XEP-0168: Resource Application Priority. Technical Report 0168, XMPP Standards Foundation, http://www.xmpp.org/extensions/xep-0168.html, November 2007.

[13] Djp, et al. JInput - Java Game Controller API. https://jinput.dev.java.net.

[14] Scott Ananian, C. JUtil - A Parameterized Collections Library for Java. http://cscott.net/Projects/JUtil.

[15] ProcessOne. ejabberd - High Performance Instant Messaging. http://www.process-one.net/en/ejabberd.

[16] Ignite Realtime. Smack API. http://www.igniterealtime.org/projects/smack/index.jsp.

[17] Saint-Andre, P. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 3920, Jabber Software Foundation, http://www.xmpp.org/rfcs/rfc3920.html, Octubre 2004. Pàgina 66.

[18] King, T. JSTUN - Java Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translation (NAT). http://jstun.javawi.de/.

[19] Bons, H. et al. Tango Desktop Project. http://tango.freedesktop.org/Tango_Desktop_Project.

[20] Yamanaka, A. JZlib – zlib in pure Java. http://www.jcraft.com/jzlib.

[21] Thomas, C. i Hunold, S. SSHTools. http://sourceforge.net/projects/sshtools.

[22] Guadall, C. TRiEM Project. Projecte final de carrera, Universitat de Girona, 2007.

[23] Larson Technologies Inc. LTI-CIVIL. http://lti-civil.org.

Índex de figures