UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Implementació VRGP: adaptador

TREBALL FI DE GRAU

Grau en Enginyeria Informàtica

Autor: Dolz Mensua, Joan Ramon

Tutor: Baydal Cardona, María Elvira
Lafond, Sebastien

Curs 4t, Enginyeria de Computadors

Agraïments

Primerament agrair la seua paciència i ajuda a la meua tutora, la doctora María Elvira Baydal Cardona. També a tots els professors que m'han acompanyat en les diferents etapes educatives, gràcies per fer-me estimar la informàtica, però també la història, les llengües, les matemàtiques, l'art i el coneixement en general.

Agrair també a tots els meus amics. Sense vosaltres la vida no és vida, només existència. No hauria pogut completar el grau sense compartir les penúries i alegries amb vosaltres.

Eternes gràcies a la meua família, que és la meua xarxa de seguretat. Tot i que aquests últims quatre anys han sigut especialment difícils, ací estem, encara sobre els nostres peus.

Per útlim, i més important, a Amiel i Ixchel.

Resum

Aquest treball es centra en la creació d'un adaptador per donar suports als vaixells per a un protocol de guiat remot de vaixells (VRGP per les seues sigles en anglés). Més concretament en el desenvolupament i posterior implementació de l'adaptador entre buc i port. El treball s'encapsula dins d'un projeccte coordinat entre l'*Åbo Akademi* i *Åboamare* (Turku, Finlàndia) amb l'objectiu de desenvolupar un vaixell autònom i de reduir les lesions entre els pràctics (capitans que condueixen els bucs en aigües perilloses o de trànsit intens).

Amb la implementació d'aquest protocol s'espera poder establir comunicació molts a molts entre ports i vaixells, enviant bidireccionalment informació sobre l'estat de les embarcacions i instruccions per a la navegació i atracament de les mateixes.

Paraules clau: Protocol, Remot, Guiat, Buc, Adaptador, Port, VRGP

Resumen

Este trabajo se centra en la creación de un adaptador para dar soporte a barcos para un protocolo de guiado remoto de barcos (VRGP por sus siglas en ingés). Más concretamente en el desarrollo y posterior implementación del adaptador entre el barco y el puerto. El trabajo se encapsula dentro de un proyecto coordinado entre la *Åbo Akademi* y *Åboamare* (Turku, Finlandia) con el objetivo de desarrollar un barco autónomo y de reducir las lesiones entre los prácticos (capitanes que conducen las embarcaciones en aguas peligrosas o de tráfico intenso).

Con la implementación de este protocolo se espera poder establecer comunicación muchos a muchos entre puertos y barcos, enviando bidireccionalmente información sobre el estado de las embarcaciones e instrucciones para la navegación y atraco de las mismas.

Palabras clave: Protocolo, Remoto, Guiado, Barco, Adaptador, Puerto, VRGP

Abstract

This work is focused on the creation of an adapter that gives support to vessels for a vessel remote guidance protocol (VRGP). More specificaly on the devvelopement and implementation of the adapter between the vessel and the harbor. This is a sub-project of a bigger one coordinated by *Åbo Akademi* and *Åboamare* (Turku, Finland) with the goal of developing an autonomous boat and reducing the accidents among the port pilots.

With this implementation we try to establish a many-to-many communication between harbors and vessels, sending both ways information about the state of the boat and instructions for its navigation and dock.

Key words: Protocol, Remote, Guidance, Vessel, Adapter, Harbour, VRGP

Índex

Índex ix

Índex de figures xi
Índex de taules xii
1 Introducció 1
1.1 Motivació . . . 3
1.2 Objectius . . . 3
1.3 Metodologia . . . 3
1.4 Estructura . . . 4
2 Context del projecte 7
2.1 Estat de l'art . . . 7
2.2 Projecte Åboat . . . 9
2.3 Nivell d'autonomia de l'Åboat . . . 10
2.4 JSON . . . 11
2.5 WebSockets . . . 12
2.6 WebRTC . . . 12
2.7 OpenDLV . . . 13
3 Proposta 15
3.1 Estructura dels diferents nivells . . . 16
3.1.1 Estructura de l'adaptador . . . 17
3.1.2 Estructura del MOC . . . 18
3.2 Protocol . . . 19
3.3 Disseny de l'adaptador . . . 23
3.4 Implementació de l'adaptador . . . 27
4 Entorn experimental 31
5 Seguretat 33
6 Conclusions 37
6.1 Problemes durant el projecte . . . 38
6.2 Aprenentatge . . . 39
6.3 Relació amb els estudis . . . 41
Bibliografia 43

Índex de figures

1.1 Imatge de la web de l'Åboamare del seu simulador 2

2.1 Missatge exemple JSON 11
2.2 Missatge de comprobació de l'estatus de connexió de l'*Åboat* 13

3.1 Diagrama de classes de l'adaptador 17
3.2 Diagrama de components del MOC 18
3.3 Diagrama d'activitat de la comunicació entre vaixell (adaptador) i MOC (port) 20
3.4 Format de la direcció *URL* per a obrir el canal entre vaixell i MOC . 21
3.5 Missatge exemple d'informació sobre el vaixell 21
3.6 Missatge exemple de demanda d'informació sobre el vaixell 22
3.7 Missatge exemple de latència del MOC 22
3.8 Missatge exemple de latència del vaixell 22
3.9 Missatge exemple de requeriment del nivell de control *recommendation* 22
3.10 Missatge exemple de despedida 23
3.11 Components de l'*Åboat* 24
3.12 Components condició necessària de l'*Åboat* 25
3.13 Diagrama de classes de l'adaptador actualitzat 25
3.14 Diagrama de components de l'adaptador 26
3.15 Missatge d'acceleració1 27
3.16 Funció per extraure les dades del missatge *odvd* de l'acceleròmetre . 28

4.1 Fotografia de la web FiTech de l'Åboat 32

5.1 Matriu de riscos del sistema VRGP 34

Índex de taules

2.1 Nivells d'autonomia segons Lloyd's Register 10

3.1 Missatges amb resposta 19
3.2 Paraules clau als missatges del MOC al vaixell 29

CAPÍTOL 1
Introducció

Els treballadors als ports, com a totes les grans àrees industrials, s'enfronten a riscos diaris que posen en perill la pròpia condició física i la aliena. No són pocs els accidents laborals entre estibadors i pràctics, i un accident d'aquest tipus pot resultar devastador no només a nivell individual, sinó que pot causar grans pèrdues econòmiques i ecològiques. En aquest treball ens centrarem en el disseny i implementació d'un adaptador que dona suport als vaixells per un protocol de comunicació entre el vaixell i el port que pretén evitar part d'aquests accidents: els dels pràctics.

Els pràctics són capitans de buc especialitzats en la navegació dels vaixells en aigües complicades, com poden ser els ports o els canals. Degut a la diferent distribució que tenen els ports, moltes vegades és necessari que un capità coneixedor de l'entorn prenga el control de la nau per atracar-la de manera segura, igual que passa en els canals per fer els bucs travessar-los sense perill. Per fer-se una idea de la importància dels pràctics podem observar el gran embús causat per l'*Ever Given* al Canal de Suez i les seues conseqüències en l'economia a nivell global, amb increments de més 6% en els preus del cru [17].

Per a que els pràctics puguen exercir aquesta important tasca, a dia de hui, han de prendre el control del vaixell de manera física, pujant per unes escales situades als costats. Aquesta acció és la causa de la majoria dels accidents laborals entre els pràctics. L'objectiu del protocol que es desenvoluparà durant aquest treball és el de controlar de manera remota els vaixells de forma que els pràctics puguen exercir el seu treball des del port enlloc d'arriscar la vida abordant els vaixells.

Contem a més amb l'evolució tecnológica ens els darrers anys, que condueix a una major automatització en la navegació. Aquesta automatització pot suposar una ventatja i un impediment a l'hora de dissenyar i implementar el protocol, ja que cada buc està dissenyat i constuït amb un equipament específic. És per això que aquest treball de fi de grau és centrarà en profunditat en el disseny i implementació d'un adaptador. L'adaptador és una peça de *software* que actua de traductor en el diàleg entre el vaixell i el port.

Amb la implementació de l'adaptador ens assegurem una comunicació fluida possibilitant l'intercanvi d'informació entre el vaixell i el port. Aquesta informació poden ser missatges sobre l'estat general del buc, *streaming* de vídeo de les càmeres o inclús instruccions de control des de terra, un control remot.

És important establir ara la definició d'alguns termes que s'utilitzaran durant la resta del treball. El primer de tots és VRGP, que està al propi títol del treball, i correspon a les sigles en anglés per a Protocol de Guiat Remot de Vaixells (*Vessel Remote Guidance Protocol*). És el protocol sobre el que es basarà tot el treball i que possibilitarà que aquesta comunicació siga eficaç.

Un altre terme que serà freqüentment utilitzat en les pròximes pàgines és MOC, que correspon a les sigles en anglés per a Centre Marítim d'Operacions (*Maritime Operations Center*). Aquest es refereix al lloc físic del port on el pràctic es comunicarà amb els bucs i prendrà el seu control si fos necessari. Podem imaginar-nos el lloc com un simulador com en els que es fan els entrenaments per a aviació, pilotatge de carreres o navegació marítima. La diferència és que la informació mostrada a les pantalles és real, enviada directament des del vaixell, i els controls tenen efecte sobre el comportament del mateix i no sobre un simulat. L'espai físic és similar al que es pot observar en la següent imatge (figura 1.1) que correspon al simulador de les instalacions de l'Åboamare

Figura 1.1: Imatge de la web de l'Åboamare del seu simulador

També és important anomenar als clients i col·laboradors a partir dels quals sorgeix aquest projecte. La idea de crear un protocol de comunicació estàndard sorgeix a l'Åboamare, una acadèmia marítima i centre d'entrenament per professionals marítims a Turku, Finlàndia (https://www.aboamare.fi). Amb aquesta idea contacten a l'*Åbo Akademi*, la universitat suèca de Finlàndia, també a la ciutat de Turku (https://www.abo.fi). Aquestes dues organitzacions actuen durant tot el projecte com a clients nostres, requerint-nos l'adaptador i un sistema que simule el MOC per provar les comunicacions.

1.1 Motivació

La motivació a l'hora de realitzar aquest treball i no un altre sorgeix de la possibilitat de col·laborar amb l'*Åbo Akademi* i *Åboamare* en una assignatura de projectes durant la meua estància a Turku, Finlàndia. Donada la situació actual del comerç internacional, on més del 90% es du a terme per via marítima[21], apareixen conflictes econòmics, ecològics i de seguretat laboral als ports. Encara que aquest projecte naix en un principi només com a mode de prevenció d'accidents laborals entre els pràctics, conforme avançàvem podiem observar que la seua aplicació podria extendre's a la prevenció d'embussos en zones portuàries i de risc, com passà al Canal de Suez en 2021 [22].

Al principi la relació entre el control remot del vaixells i la prevenció d'embussos pot sonar un poc abstracta, al cap i a la fi és un humà qui està en control de la nau en abdós casos. Aquesta possible aplicació que descobrim una vegada ja començat el projecte es deu a la visió global que tenen els controladors portuaris de la situació. Podem fer l'analogia amb els controladors aeris, que gestionen el trànsit a les rutes aèries gràcies a la visió extensa que tenen de la situació i la comunicació directa amb els pilots i primers oficials. Això seria, salvant les distàncies, el mateix. Els controladors portuàris, podrien gestionar aquest trànsit marítim d'una manera eficient, ajustant la velocitat de arribada dels diferents bucs per reduir al màxim els temps d'espera. Aquesta potencial aplicació tindria com a conseqüència una reducció de l'ús del combustible i del temps d'espera i, per tant, de despesses i pol·lució.

L'altre punt a destacar del projecte és la voluntat per part de les universitats col·laboradores de crear un protocol *open source*. A dia de hui no existeixen protocols per al guiat remot d'embarcacios que no siguen completament propietaris.

1.2 Objectius

Els objectius del present treball es centren en establir una connexió entre el centre d'operacions al port (MOC a partir d'ara) i el vaixell, fent que ambdues parts siguen capaces de comprendre els missatges enviats per l'altra part. No ens correspon a nosaltres que el vaixell seguisca o no les instruccions donades pel MOC, ja que el treball que ens correspon és només establir la connexió i traduir els missatges. El contingut dels missatges, encara que ara vorem que no és vanal, només ens interessa per efectuar l'adaptació entre les parts, no com a instruccions pròpiament dites. Tampoc avaluem nosaltres si els valors donats són lògics o no, dissenyem i desenvolupem un traductor i missatger, no un executor.

1.3 Metodologia

La metodologia a l'hora de dur a terme l'actual treball és complexa i va haver de ser decidida per tot l'equip al començament del projecte. Aquesta complexitat es dona pel fet de formar part d'un projecte major i de treballar en un equip format exclusivament per estudiants durant el desenvolupament de l'adaptador.

El desenvolupament i integració de l'adaptador, com s'ha anomenat abans, és només una part d'un projecte que pretén la creació d'un buc autònom. Aquesta col·laboració es va fer a mode de client-desenvolupador, on els clients eren tant l'*Åbo Akademi* com *Åboamare*, que ens varen requerir als alumnes de *Project Course* per desenvolupar les comunicacions entre MOC i vaixell. L'equip de treball estava composat per set persones que ens varem repartir el treball depenent de les nostres capacitats i coneixements prèvis. durant tot el període del projecte la comunicació amb els clients va ser fluida, intercanviant documentació i còdi per aconseguir els objectius, tenint accés a les seues oficines per efectuar reunions i testejos.

L'equip de desenvolupadors estava format per un encarregat de burocràcia, legalitat i empresa; un encarregat de la gestió de les ferramentes i del *setup* del *pipeline*; tres encarregats de la creació de l'entorn que simularia el port; i dos encarregats del costat del vaixell. El treball de fi de grau que ens ocupa es basa en el desenvolupament del costat del vaixell, on un company s'encarregava de la comunicació en cru amb el port i jo m'encarregava de l'adaptador. Per tant, la gestió del treball ha sigut molt important durant l'evolució d'aquest projecte ja que l'equip era considerablement gran. Per abordar aquesta gestió utilitzarem una metodologia àgil com és *scrum* [18].

Scrum és una metodologia de treball pensada per treballar en equip on es realitzen entregues parcials de manera regular del producte final. Durant l'exercisi d'aquesta metodologia es realitzen *sprints* (cicles temporals curts i de durada fixa) que en el nostre cas eren d'una setmana. Per a això ens reuniem setmanalment exposant a la resta de l'equip els avanços, intercanviant possibles sol·lucions i, posteriorment, mostrant els progressos als clients.

1.4 Estructura

Aquets projecte es divideix en cinc capítols, contant el capítol introductori. Cadascun dels capítols té alhora seccions on s'aprofundeix en el contingut.

El capítol introductori és poc més que el que sogereix el títol, una vista per damunt del que serà el contingut del treball. Ací s'expliquen els termes que s'utilitzaran durant les properes pàgines, s'explica el perquè s'ha escollit un projecte sobre el protocol de guiat remot de vaixells i què és un protocol de guiat remot de vaixells. A més, també s'explica quins són els objectius a aconseguir al terme del treball i quina metodologia es seguirà per aconseguir aquests objectius.

Al context del projecte, el segon capítol, ens endinsem en tot l'entorn que rodeja el project, des de l'estat actual dels protocols de comunicació entre bucs i ports fins a la tecnologia utilitzada per implementar l'adaptador. El context del projecte es pot considerar una introducció avançada, una col·lecció d'informació necessària per comprendre què estem fent i com ho fem. El primer apartat del capítol és l'estat de l'art, on es parla de les diferents tecnologies i protocols ja existents en la comunicació marítima. Després tenim el capítol sobre el projecte que engloba aquest treball, l'Åboat, un projecte de navegació autònoma establert a Turku, Finlàndia. A més hi ha un espai dedicat a delimitar el nivell d'autonomia del vaixell desenvolupat al projecte. En els següents apartats introduïm

breument les quatre tecnologies diferents utilitzades durant tot el transcurs del projecte: *JSON, WebSockets, WebRTC* i *OpenDLV*.

El tercer capítol està enfocat en el desenvolupament i implementació del propi adaptador. Comencem veient l'estructura dels diferents components que formen el sistema, ja que és necessari comprendre com funciona l'altra part, el MOC, per entendre l'evolució de l'adaptador. Després entrem al terreny del protocol, fonamental per fer funcionar tota la maquinària i que ha de ser suficientment robust com per a abarcar la gran varietat de bucs existents. Una vegada vist el protocol es pot començar a dissenyar l'adaptador i passar-lo a una fase més avançada. L'últim pas, no és un altre que la implementació del mateix.

El següent capítol està dedicat a explorar l'entorn experimental on es testeja el producte. Aquest capítol no té seccions diferenciades dins d'ell ja que és una explicació dels passos seguits per fer les proves.

El cinqué capítol tracta sobre les consideracions en matèria de seguretat informàtica que s'han pres i les que es deurien prendre a l'hora d'abastir un projecte d'aquestes característiques.Veurem que degut a les limitacions tècniques i temporals, aquest apartat del projecte no s'ha tingut en compte amb tanta consideració com la resta d'apartats.

I per últim tenim les conclusions, on analitzem el recorregut seguit durant el projecte. Aquest capítol està dividit en tres seccions diferents, apart d'una introducció en la que es fa un breu resum de tot el que s'ha exposat durant el treball. Aquests apartats són de característiques més personals, ja que parlen dels problemes que han sorgit en el temps en que s'ha treballat en el projecte, els coneixements que s'han adquirit i la relació del conjunt amb els estudis cursats.

CAPÍTOL 2
Context del projecte

Com ja s'ha anomenat en el capítol introductori, el comerç marítim és la principal via de comerç en l'actualitat, i això presenta diversos problemes que, gràcies als avanços en tecnologia, es poden solventar o amainar. Durant aquest capítol aprofundirem en el estat actual tant de les tecnologies de comunicació disponibles com en els protocols existents per al guiat remot de vaixells, que com vorem són privatius i no molts.

2.1 Estat de l'art

A dia de hui les tecnologies per comunicacions remotes són moltes i molt variades, des de la ràdio fins internet passant pels infrarrojos o l'ultrasó. Cada tipus de comunicació té les seues pròpies característiques que el fan més efecctiu en un escenari o en un altre.

La ràdio és una tecnologia que ens permet efectuar comunicacions a molt llarga distància i en condicions complicades. Existeixen quatre àrees geografiques definides pel *Global Maritime Distress Safety System* depenent de les quals es necessita un sistema de comunicació per ràdio o un altre. No entrarem en profunditat en les diferències entre les diferents àrees ja que abarquen paràmetres com la latitud a la que es troba el vaixell, que no són útils per al propòsit d'aquest treball. Però sí que cal remarcar que aquestes àrees abarquen fins una distància de 150 milles nàutiques des de la costa (228Km aproximadament)[19]. El problema que es presenta amb la ràdio són les pèrdudes d'informació que es poden donar, i quan es tracta de dirigir un buc en temps real, aquesta limitació és molt important.

Respecte a mètodes de comunicació a distància com els infrarrojos o els ultrasons, presenten impediments obvis per al control a distància de grans vaixells. Els infrarrojos, com qualsevol que haja utilitzat alguna vegada un comandament de televisió pot saber, ha d'estar dirigit directament al receptor, el que fa que a distancies suficientment llargues siga molt complicat establir la comunicació. Els ultrasons, per contra, el que necessiten és un medi estable pel que poder propagar-se, l'aigua és un d'aquests medis, però no assegura tampoc un correcte enviament del màxim número de paquets.

Per últim queden les microones. Les microones són la tecnologia en la que es basen els protocols 4G i 5G. És la tecnologia que senta la baseper a les comunicacions sense fil que tenen lloc a internet. Amb les antenes 4G, que tenen un rang de entre 30Km i 35Km i una senyal menys propícia a interferències [4]. A més tenim el protocol 5G, a pesar de no tindre un rang tan ample com el del 4G, també pot ser de gran utilitat a l'hora d'establir comunicacions amb els vaixells. Part d'aquesta ventaja es dona pel fet que, en contra de la radio, els infrarrojos o els ultrasons, no és necessari establir una comunicació directa entre port i buc, només és necessari que el buc estiga connectat a una antena, sense importar el punt del món on aquesta estiga (en aquest cas, degut a la latència, seria bastant més complicat controlar-lo de manera remota en temps real).

Aquesta és la situació actual, amb una visió molt genèrica, en quant a tecnologies de comunicació. El panorama canvia bastant si ens centrem en la situació dels protocols de guiat remot de vaixells, ja que no existeix cap. El més paregut que hi ha a l'objectiu que es pretén durant aquest projecte en la actualitat és un guiat remot desenvolupat per *Rolls-Royce*. [5]

El protocol dissenyat per *Rolls-Royce* és en base molt paregut al nostre: un protocol de comunicació entre el port i el vaixell, amb especial èmfasi en els vehicles autònoms. De fet, aquesta és una de les principals diferencies entre el protocol de la gran companyia britànica i el nostre, ja que amb el nostre protocol s'ofereix un servei no únicament per a vehicles autònoms, sinó també per a vehicles tripulats que requereixen d'assistència per maniobrar. És per això que al nostre projecte es pot observar el nom *Maritime Operating Center*, que es tradueix com a Centre Operatiu Marítim, mentres que al protocol de *Rolls-Royce* el centre anàlog és el *Remote Operating Center*, que es tradueix com a Centre Operatiu Remot. Aquesta diferència lèxica és important per entendre la dimensió que abarquen ambdós protocols. El VRGP busca una comunicació global, amb tots els vaixells que la requerisquen adaptant-se al seu grau d'autonomia, el protocol *Rolls-Royce* per contra només actua amb vaixells autònoms.

En la resta d'àmbits, el protocol de la marca anglesa i el nostre és bàsicament el mateix. Sense conèixer, per suposat, el funcionament intern de les comunicacions, podem dir que el funcionament en nivells alts és el mateix, s'intercanvia informació entre vaixell i port en ambdues direccions, podent des delport prendre control remot del vaixell.

2.2 Projecte Åboat

Dins d'aquest context de falta d'un protocol establert per a la comunicació d'instruccions entre port i vaixells entren la *Åbo Akademi* i *Åboamare* amb l'objectiu de millorar la seguretat dels pràctics al port, a més de reduir costos per possibles accidents o embussos. L'objectiu general és el de dissenyar i construir un buc completament autònom, sense la necessitat d'un pràctic o capità que ho controle des de terra. Però en el camí per aconseguir aquest objectiu, el projecte es diversifica a diferents nivells per sota, donant via també a un control remot en cas de situacions complicades o d'emergència. És aquest cas sobre el que es sustenta el present treball. Aquestes situacions d'emergència es comuniquen mitjançant una connexió HTTP al port, que és qui decideix en última instància si estableix o no connexió amb el vaixell, això ho fa mitjançant uns missatges definits pel protocol que analitzarem més endavant. Una vegada establerta la connexió, totes les comunicacions de l'*Åboat* es fan mitjançant *WebSocket Secure* en cadenes de caràcters amb format JSON, també aprofondirem més en la construcció i traducció d'aquests missatges en les pròximes pàgines.

Al final, l'importància que té l'*Åboat* en aquest treball és que fa d'entorn, un entorn controlat i que estableix uns límits clars per al desenvolupament de l'adaptador. Aquests límits es tradueixen en un requeriment per part dels clients en utilitzar certes tecnologies, com són *OpenDLV*, *WebRTC* i *WebSocket Secure*.

2.3 Nivell d'autonomia de l'Åboat

Per acabar d'entendre quin és el paper de l'adaptador dins de l'*Åboat* hem d'entendre en quin nivell d'autonomia ens movem. No és el mateix adaptar els missatges per a què siguen llegibles per humans a fer-lo per a màquines. Per a aclarar el nivell d'autonomia del que parlem, podem fixar-nos en diversos estàndards. No hi ha un consens clar sobre els diferents nivells d'autonomia ni en vaixells ni en cap altre tipus de vehicles, l'estàndard més comú i citat és el de *Lloyd's Register* que estableix set nivells d'automatisme començant pel nivell 0 fins el 6 [13]. Però com hem dit abans, *Rolls-Royce* també té els seus pròpis protocols i estàndards, en aquest cas la seua definició d'autonomia entra en més profunditat definint fins 10 nivells des de l'1 fins al 10 [5]. Per establir el nivells d'autonomia ens fixarem en la taula de set nivells de *Lloyd's Register*.

Nivell d'autonomia	Descripció
AL 0: Control manual	Sense funció autònoma. Tota acció i presa de decisions es fa manualment, això és: els humans control·len totes les accions
AL 1: Suport de decisions a bord	Totes les accions són preses per un operador humà, però les ferramentes de decisió poden presentar opcions o, per contra, influenciar les accions preses. La informació és donada pels sistemes a bord
AL 2: Suport de decisions a bord i a distància	Igual que amb el sistema de decisions a bord amb l'afegit que la informació pot ser donada per sistemes a bord i fora borda
AL 3: Humà actiu al bucle	Les decisions i les accions són executades amb supervisió humana. La informació pot ser donada per sistemes a bord i fora borda
AL 4: Humà al bucle	Operador / Supervisor: les decisions i accions són executades de forma autònoma amb supervisió humana. Les decisions de gran impacte s'implementen de manera que dona als operadors humans la oportunitat d'intercedir i sobreescriure
AL 5: Autònom	Rarament supervisat, les decisions són preses i executades pel sistema
AL 6: Autònom complet	Sense supervisió, les decisions són completament preses i executades pel sistema

Taula 2.1: Nivells d'autonomia segons Lloyd's Register

Veient la taula podem observar que el projecte de l'*Åboat* podria entrar en diferents categories. Descartem d'inici les categories AL 5 i AL 6, donat que aquest nivell d'autonomia sí que es persegueix al projecte, però no al subprojecte al qual s'engloba l'adaptador VRGP. Tenint en compte que les instruccions s'envien des de port per un operador especialitzat, també podriem descartar la categoria AL 4. El nostre treball llavors es trobaria en un punt intermig entre l'AL 2 i l'AL 3,

això és: les accions són executades pel sistema però amb una supervisió humana. En el cas de les instruccions enviades des de port al vaixell ens trobariem en el nivell AL 2, on les accions son preses per un operador humà amb informació fora borda. En el cas de la comunicació des del vaixell al port, ens trobariem al nivell AL 3 on les accions són executades pel sistema però amb supervisió humana.

2.4 JSON

JSON és un format de text (no un llenguatge de programació) lleuger per l'intercanvi de dades. L'estructura amb que es formatetgen les dades dins del document fa que siga fàcil de llegir per humans i fàcil d'interpretar per màquines. Aquest format està compost per dos senzilles estructures: objectes, i llistes ordenades.

Els objectes són estructures definides entre claus d'apertura i de tancament, dins d'ells els noms van seguits de dos punts i posteriorment del valor asignat. Aquestes parelles de nom i valor estan separades entre si per comes.

Per altra banda tenim les llistes ordenades de valors, que com a gran part dels llenguatges de programació es defineixen entre corxets amb els elements separats per comes.

Així expressat pot sonar un poc confús, però si veiem un exemple, com el de la figura 2.1, ens adonem que és una estructura molt senzilla no només per a la lectura humana, sinó per la interpretació per part de la màquina.

```
1  {
2    "information": {
3      "coordinates": [60.392581, 22.104095],
4      "call": 3LXY
5    }
6  }
```

Figura 2.1: Missatge exemple JSON

És aquesta senzillesa la que ha fet que ens decantarem per l'ús de JSON sobre altres com *XML* per al format de la informació intercanviada entre el vaixell i el MOC. A més té l'advantatge de que hi ha una gran col·lecció de llibreries disponibles per al processament de fitxers JSON en els llenguatges utilitzats durant el projecte (C++, *JavaScript* i *Java*).

2.5 WebSockets

WebSockets és una tecnologia utilitzada per establir una connexió interactiva entre un client i un usuari, aquesta és una comunicació *full-duplex* sobre un únic *socket TCP*. El protocol *WebSocket* està definit per l'IETF [9], organització que publica els estàndards que regulen la comunicació en internet. És àmpliament utilitzat en diferents aplicacions web. A diferència de l'HTTP, els *WebSockets* neixen amb la intenció de que qualsevol de les parts puga envia la informació necessaria sense que aquesta siga requerida prèviament per l'altra part. Aquesta característica fa que siga molt més útil en aplicacions amb trànsit de dades en temps real, ja que evita que el client haja d'estar interrogant permanentment al servidor per si aquest necessita enviar-li dades. Al projecte els *WebSockets* són utilitzats per establir la connexió entre el vaixell i el MOC.

El fet de que siga tan popular, ja que té suport en els principals navegadors com Firefox, Safari, Google Chrome, Opera o Edge [15], és un punt a favor a l'hora d'escollir-lo per establir les connexions entre vaixell i MOC. A més, la seua estructura client-servidor s'adapta perfectament a la relació que hi ha entre ambdues parts, junt a la ventatja sobre HTTP de no ser un protocol dirigit pel client. El problema és que per defecte, els *WebSockets* no estan xifrats, la connexió no és segura. I la seguretat en un sistema crític, com és el control d'una nau, és bàsica.

Per tant, es necessita una capa de seguretat sobre aquesta tecnologia, i ací és on entren els *WebSocket Secure*. Aquestos no són una altra cosa que *WebSockets* xifrats. Per comprendre això necessitem saber que els *WebSockets* estableixen la connexió mitjançant un *handshake* sobre HTTP (no xifrat), per xifrar aquesta comunicació, el que fan els *WebSocket Secure* és establir una capa de xifrat per sota amb el protocol TLS (*Transport Layer Security*).

2.6 WebRTC

El propi nom *WebRTC*, si expandim les sigles (*Real Time Communication*), ens pot donar una idea de per a què serveix. És una especificació de comunicació en temps real de codi obert mitjançant APIs.

Aquesta tecnologia està desenvolupada en dues parts, per una banda, l'especificació del protocol està feta per l'IETF, mentres que la especificació de l'API *JavaScript* està definida pel W3C [3]. Aquest protocol serveix per intercanviar qualsevol tipus de missatge o informació a través del camí més directe possible entre els dos participants, però és especialment útil a l'hora d'enviar vídeo i audio, és a dir, contingut multimèdia. I és justament aquesta la raó per la qual *WebRTC* va ser escollida per aquest projecte, per l'enviament de contingut multimèdia en temps real, que és una de les necessitats bàsiques de les comunicacions entre el vaixell i el MOC.

WebRTC és compatible amb qualsevol interfície HTML5 o *WebSocket* i pot ser implementada en una aplicació nativa sempre que tinga suport per als protocols IPv4 i IPv6 , qualitat que ens convé en un projecte com aquest en el que es pretén generalitzar la comunicació de qualsevol tipus de vaixell amb el port. També és important el fet de que per les seues especificacions, *WebRTC* està xifrat d'extrem a extrem ja que la sessió *WebRTC* ocorre habitualment sobre tecnologies web TLS, qualitat important de cara a la seguretat de les comunicacions.

2.7 OpenDLV

Potser *OpenDLV* siga la tecnologia menys coneguda de les utilitzades en el desenvolupament de l'adaptador. Tècnicament no és una tecnologia, sinó un entorn software utilitzat pel desenvolupament de vehicles autònoms. Aquest entorn es basa en sessions UDP *multicast* en les adreces 255.0.0.X on X pertany al rang [1, 254]. D'aquesta manera, tots els serveis dins del mateix grup UDP poden comunicar-se entre ells [1]. A això se li anomena sessió.

Els missatges enviats dins d'aquestes sessions tenen un format comú, aquestos missatges intercanviats són *envelopes*, que contenen la informació estructurada com a *protobuf*. Un exemple de definició d'un missatge de connexió seria el de la figura 2.2:

```
1  message ConnectionStatus [id = 3] {
2    string url     [id = 1]
3    string message  [id = 2]
4    uint32 code    [id = 3]
5  }
```

Figura 2.2: Missatge de comprobació de l'estatus de connexió de l'*Åboat*

Podem observar que el format és molt intuitiu, cada tipus de missatge té un identificador, i dins el missatge, cada argument té també un identificador. Per tant, si rebem un paquet *id* = 3, amb la propietat 1 = "www.prova.com" i la 2 = "-Hola món!", ja sabem que és un *ConnectionStatus* que ve de la *url* "www.prova.com" i amb un missatge que diu "Hola món!".

Aquesta és la propietat més important de OpenDLV de cara a entendre els missatges, a l'hora d'escriure i llegir els missatges, és tan fàcil com seleccionar aquells atributs que volem llegir o modificar amb el seu identificador.

Amb aquest software el que es pretén és l'intercanvi de dades entre les interfícies *hardware* i *software* en un nivell alt, sense haver d'acudir a nivells més baixos de programació. Aquest entorn s'utilitza en tot tipus de vehicles autònoms, des de vaixells fins a cotxes o drons. És una manera de simplificar l'obtenció de dades de diferents sensors i alhora enviar instruccions als diferents actuadors com poden ser motors, ja que estàn tots reunits dins la mateixa adreça UDP.

CAPÍTOL 3
Proposta

Una vegada comprés el context del projecte i analitzada la situació en la que aquest es troba, és l'hora de proposar un sistema funcional per a que aquest funcione. Recordem que l'adaptador és no més que una part del projecte complet, és per això que cal entendre la proposta completa abans d'entendre el disseny de l'adaptador per si mateix. Es varen proposar en principi tipus molt diferents de comunicació, només estaven clars els requeriment dels clients (*Åbo Akademi* i *Åboamare*) que eren utilitzar missatges en format JSON entre MOC i vaixell mitjançant tecnologia de *WebSocket Secure*, *WebRTC* i, dins del propi vaixell utilitzar missatges mitjançant *OpenDLV* donat que era el sistema prèviament implementat. A partir d'eixos requeriments es plantejava crear una arquitectura capaç de complir i fer-lo d'una manera tan eficient com fóra possible.

Durant aquest capítol entendrem per què es va escollir una arquitectura basada en tres classes per al MOC, distribuides en: *websocket*, *message* i *state*. Veurem la classe *websocket* en profunditat per entendre la comunicació entre aquesta i l'adaptador VRGP. També entrarem en el disseny propi de l'adaptador VRGP, per al qual haurem de comprendre els llenguatges i tecnologies utilitzades, a més del protocol i el format dels missatges *odvd*, que és el format propi de la tecnologia *OpenDLV*.

Abans d'entrar a la matèria específica, és important entendre com funciona la comunicació que requereixen els clients entre MOC i vaixell. Tota connexió comença per requeriment del vaixell, que informa al MOC de la seua situació i capacitats per les operacions a distància (això és, una llista amb sensors i actuadors disponibles al buc), i opcionalment la seua necessitat de guiat remot. El MOC llavors actua com a servidor i reconeix el missatge rebut, després obre la comunicació. Durant aquesta fase s'estableix el canal de comunicació entre el buc i el MOC.

Quan ho considere necessari, el MOC sol·licitarà el vaixell un fluxe d'una o més fonts (sensors, càmeres, GPS...) amb informació a bord. Les fonts d'informació que el buc pot transmetre al MOC inclouran sempre les dades de navegació actuals com poden ser la posició, el rumb, la velocitat, la velocitat de gir... En cas de que el buc sol·licite orientació, el MOC pot enviar missatges amb les indicacions necessaries a través del canal creat prèviament. Tant el buc com el MOC poden decidir en quin moment posar fi a la comunicació, i això és fa d'una ma-

nera ordenada mitjançant uns missatges específics del protocol que veurem més endavant.

Tot aquest procés és similar a una videocridada, com la que es faria per *Skype* o *Zoom*. Posem per exemple que Alice i Bob estan en una videocridada per *Zoom*, per a establir la connexió Alice ha de donar-li el seu contacte a Bob (missatge del vaixell al MOC) i llavors Bob pot començar la cridada. Alice té un problema, ha d'instal·lar una aplicació i no sap com, així que Bob li demana que compartisca pantalla i encenga el micròfon (el vaixell envia un *stream* de les dades dels sensors). Si només amb les indicacions no aconsegueixen instal·lar l'aplicació, Bob sempre pot demanar a Alice el control remot de l'ordinador. Per últim, qualsevol dels dos pot terminar la comunicació només donant-li al botó de penjar.

3.1 Estructura dels diferents nivells

L'adaptador compleix una funció fonamental en les comunicacions entre el vaixell i el MOC, sent l'element que funciona de traductor entre els formats dels diferents actors. Així doncs, l'adaptador és un element capaç de rebre missatges per part del port, traduïrlos al format del vaixell i acctuar en conseqüència. El que l'adaptador pot fer una vegada rep i processa un missatge és decidir si traduir i remetre el missatge al vaixell (per exemple en cas d'una instrucció), o enviar informació enviada pel vaixell i traduida pel propi adaptador al MOC (en el cas d'un requeriment).

Amb la idea de com funciona la comunicació, és hora de desenvolupar una arquitectura que complisca amb aquesta funció. La primera idea que pot sorgir és la de dos terminals (vaixell i MOC) que es comuniquen sense més. És una idea senzilla que podria funcionar si en cada extrem hi haguera humans controlant el sistema, però tenim la situació que el vaixell no té cap humà darrere mentres que al MOC sí que hi ha operadors humans, així que la informació que s'envia ha de traduir-se en algun punt. Es presenten dues opcions llavors, instal·lar l'adaptador (el traductor) al costat del MOC o al del vaixell. Si analitzem ambdues opcions, podem observar que a la llarga instal·lar l'adaptador al MOC seria poc pràctic, ja que per cada sistema diferent que hi haguera, s'hauria d'ampliar l'adaptador al port, cosa que acabaria per crear diferències entre els ports, i el que es busca amb aquest projecte és crear un protocol universal i unificat, així que no seria d'utilitat. Per tant, l'única opció viable és la d'instalar l'adaptador directament al buc, sent aquest específic per a cada tipus de vaixell. Al ser un projecte de codi obert aquesta idea pot suposar un gran avantatge, ja que les navieres poden instal·lar l'adaptador sense haver de canviar tot el sistema de control, mentres que els ports no han de preocupar-se de res més que establir connexió amb el vaixell, sense haver de saber com funciona internament cada nau.

3.1.1. Estructura de l'adaptador

Així, tenim dos sistemes separats, l'adaptador i el MOC, on dins de l'adaptador tenim dues classes separades, una per comunicarse amb el hardware del vaixell, i una altra per establir la connexió via *WebSockets* amb el MOC. Això ens presenta automàticament la necessitat de crear també una classe que es dedique a controlar ambdues i fer d'enllaç, un *manager*. En la figura 3.1 podem observar el diagrama de classes corresponent a l'adaptador. Aquest es divideix en tres classes: *Adapter manager*, *OpenDLV handler* i *WebSocket client*. Les classes per si mateixes són extremadament simples en quant a quantitat de funcions, després vorem com la classe *OpenDLV handler* necessita d'una classe suplementària que guarde l'estat del vaixell, però per ara aquest diagrama és més senzill per fer-nos una idea del funcionament de l'adaptador.

La classe *WebSocket client* estableix la connexió *WebSocket* mitjançant la funció *run()*. Cada vegada que rep un missatge del MOC (VRGP *service* al diagrama) el passa directament al *manager* que a la seua vegada ho passa al *handler*. Per contra, quan s'ha d'enviar un missatge, la funció utilitzada és *send*, i és el *manager* qui la crida.

Si ens movem a la part esquerra de la figura podem observar la classe *OpenDLV handler*, aquesta és la classe encarregada de comunicar-se amb el vaixell, mes concretament la classe encarregada de rebre les senyals dels diferents sensors i enviar aquelles que són requerides. El mètode *run()* s'encarrega d'obrid una sessió *OpenDLV* en el canal especificat per rebre les diferents senyals, mentres que el mètode *on_receive()* selecciona de la informació recollida aquella que s'ha demandat. És per això, que per simplificar aquesta tasca el mètode *run()* actualitza una classe de estat ja en el format *json* requerit, de manera que quan siga demandada, l'enviament de l'informació siga automàtic sense haver de traduir o esperar a que el sensor en qüestió envie un missatge nou.

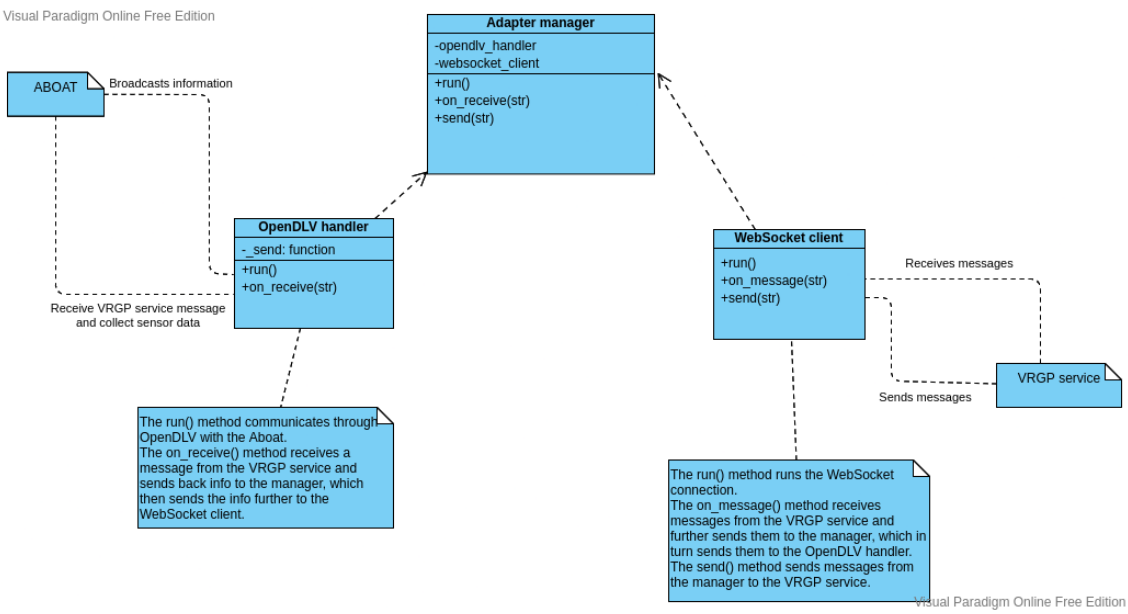

Figura 3.1: Diagrama de classes de l'adaptador

3.1.2. Estructura del MOC

Però per entendre la funció de l'adaptador no val només amb conèixer la seua estructura, també és necessari tindre una idea, encara que siga superficial, de com funciona el MOC a l'altre costat. El MOC es divideix en tres classes: *websocket*, *message* i *state*. D'aquestes tres classes ens interessa únicament la classe *websocket*, que és la que establirà la comunicació amb l'adaptador, més concretament el component *WebSocketMessageHandler*, com es pot veure a la figura 3.2.

WebSocketMessageHandler és la classe que interpreta el missatges que arriben per part de l'adaptador. És el receptor i discriminador de missatges del MOC, s'encarrega de discernir quina és la demanda de l'adaptador i passar-lo a la classe *VrgpMessageType* per satisfer la demanda. La importància d'aquesta classe a l'hora de desenvolupar l'adaptador radica en que els missatges enviats han de tindre el format adequat per a que *WebSocketMessageHandler* puga interpretar-los de forma correcta, per tant s'ha de conéixer el format acceptat per part d'aquesta classe i quines són les demandes que estan programades a la mateixa. Aquestes comandes són les especificades pel protocol, que veurem a la secció 3.2.

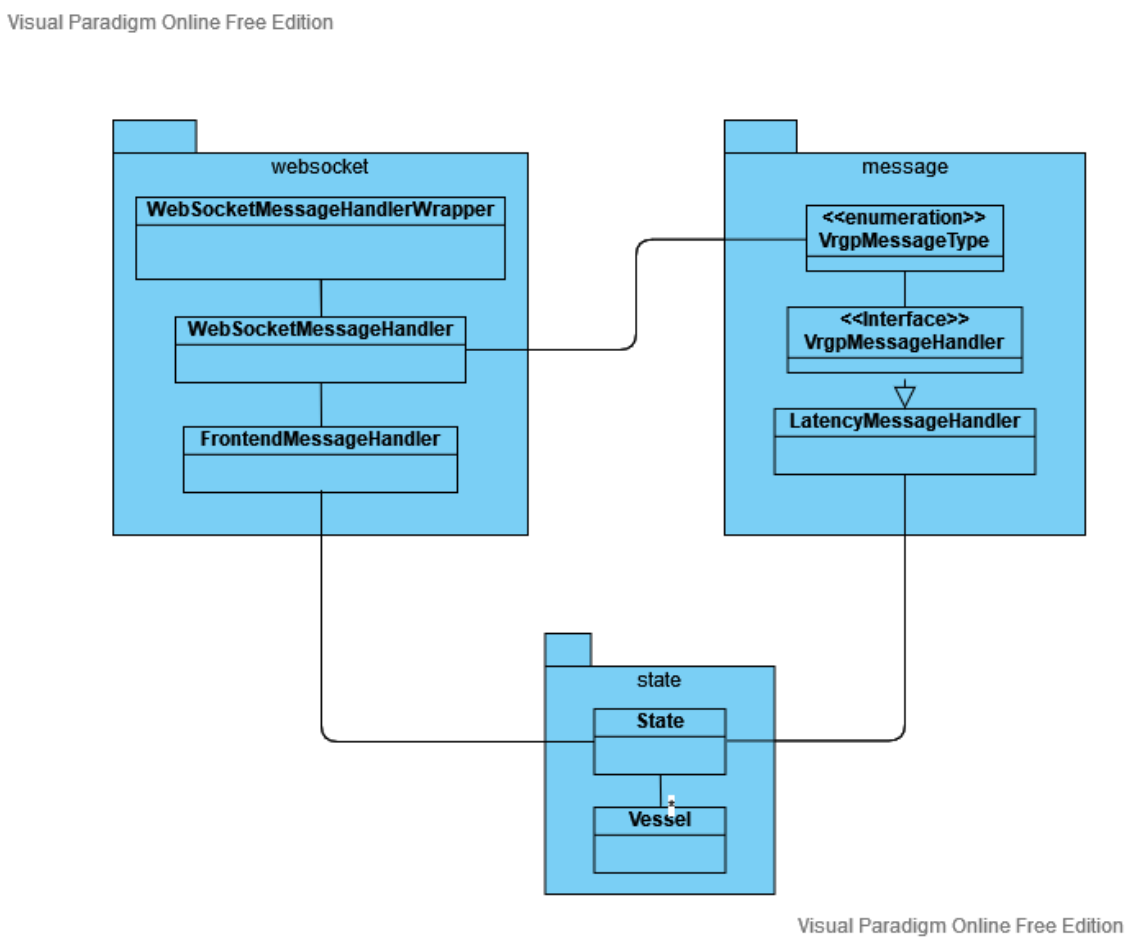

Figura 3.2: Diagrama de components del MOC

3.2 Protocol

El protocol és potser l'element més complex de tot el treball ja que s'ha de procurar que siga el més genèric i modular possible sense oblidar els detalls. Aquest protocol el podriem dividir en dos seccions: aquells missatges que requereixen d'una resposta i aquells que no. Els missatges que requereixen resposta són un grup tancat, els de registre, els de autenticació, els de requeriments i els de terminació, i els podem veure a la taula 3.1. La resta de missatges, aquells que no requereixen resposta (encara que la poden tindre) són els de enviament d'informació. Donat que els missatges amb resposta són un grup limitat, començarem veient aquestos, dels missatges independents veurem només els més importants.

Missatge	Descripció
Registre	Registre del vaixell al port
Autenticació del vaixell	Mitjançant un identificador únic de cada vaixell, s'autentica l'identitat del vaixell al port
Autenticació del MOC	Igual que l'autenticació del vaixell al port però en l'altre sentit
Requeriment d'informació	Missatge enviat pel MOC quan necessita certa informació (e.g. Dades de l'acceleròmetre)
Control del vaixell	Instruccions de navegació enviades des del port al vaixell
Terminació	Requeriment de terminació de la comunicació que pot ser enviat per qualsevol de les parts

Taula 3.1: Missatges amb resposta

D'aquests missatges, només els de registre i terminació són essencials i es dónen a totes les comunicacions que s'estableixen. En el següent diagrama d'activitat podem observar com seria la seqüència en la comunicació entre ambdues parts, com s'observa a la figura 3.3.

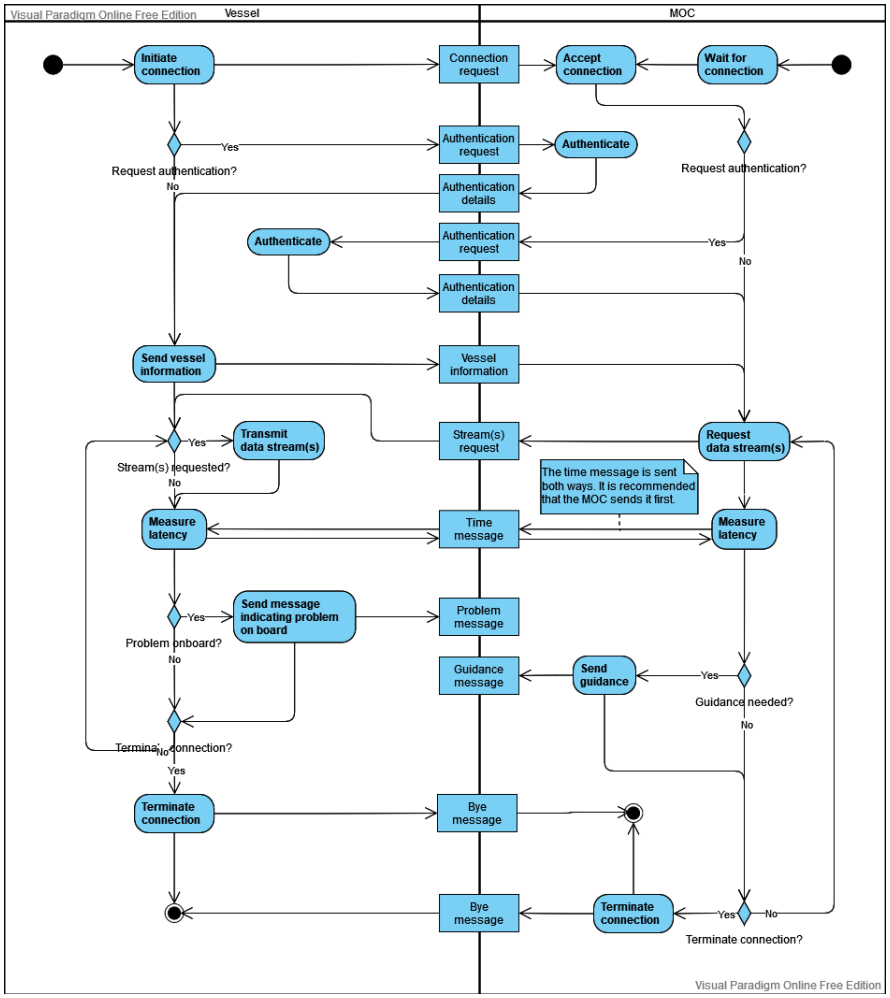

Figura 3.3: Diagrama d'activitat de la comunicació entre vaixell (adaptador) i MOC (port)

El diagrama pot resultar un poc confús a primera vista, així que anem a explicar els elements més importants d'aquest. Tota comunicació comença sempre amb la proposta de connexió per part del vaixell. Aquest missatge és el de registre, important no confondre'l amb el d'autenticació, que és opcional. Per efectuar el registre es fa en dos passos: primer s'obre el canal mitjançant *WebSockets* i després s'envia un missatge amb l'estatus actual del vaixell i les seues capacitats (sensors disponibles, possibilitat de control remot, situació geogràfica...). Per obrir el canal, el vaixell ha de conèixer la direcció *URL* del MOC de bestreta. Aquesta direcció requereix també del MMSI del vaixell.

L'MMSI és el número d'identificació del servei mòvil marítim, que identifica a cada vaixell a efectes de seguretat i radiocomunicacions i la seua assignació depén de les autoritats de cada país. Aquest número, a Espanya, es solicita a la Capitania Marítima en cas de vaixells amb eslora menor de 24 metres i a l'Àrea de Radiocomunicacions de la Direcció General de la Marina Mercant per al vaixells amb eslora igual o superior als 24 metres. A Finlàndia, país on s'ha realitzat aquest projecte, el registre de l'MMSI depén de l'Agència Finesa de Transport i Comunicacions (Traficom) i del Departament d'Estat d'Åland (província autònoma de Finlàndia).

D'aquesta manera, la direcció per obrir el canal de comunicació entre vaixell i MOC queda com a la figura 3.4.

```
wss://[amoc].aboamare.net/navigation/[mmsi]
```

Figura 3.4: Format de la direcció *URL* per a obrir el canal entre vaixell i MOC

On *[amoc]* seria el nom concret del MOC al que connectar-se i MMSI seria l'identificador del vaixell.

Després del registre cada part decideix si vol que l'altra part s'autentique, el recomanable és que sempre es demane l'autenticació de l'altra part. Aquesta autenticació es fa mitjançant certificats digitals, no té més misteri que el requeriment per les parts del certificat i la seua posterior verificació. En cas de que alguna de les parts demane l'autenticació de l'altra i no es donara, podria terminar la comunicació.

Després d'autenticar-se (o no), el vaixell envia informació sobre si mateix de nou, aquesta vegada un missatge més complet on s'especifiquen les dimensions del buc. Aquest missatge, en format *json*, inclou les dimensions (*loa* i *breadth*) i una descripció en *SVG* de la forma més definida del vaixell (*from_above* i *from_abaft*), a part del ja anomenat MMSI, un atribut per definir la senyal de ràdio assignada al vaixell (*call*), el seu nom (*name*), la seua altura (*height*), la seua profunditat (*draft*) i un atribut *simulation* per definir si la nau és real o no. Així doncs, un missatge d'aquest tipus podria ser com el següent:

```
{
  "vessel": {
    "mmsi": "230172000",
    "call": 3LXY,
    "name": "Titanic II",
    "loa": 50.0,
    "breadth": 10.0,
    "height": 12.0,
    "draft": 7.0,
    "from_above": "M 5 0 Q 10 3 [...]",
    "from_abaft": "M 5 20 Q 0 20 [...]",
    "simulation": true,
  }
}
```

Figura 3.5: Missatge exemple d'informació sobre el vaixell

Al rebre aquest missatge, el MOC pot dibuixar el perfil del vaixell sobre el mapa de manera que el pràctic que treballa en remot coneix la situació amb el major detall possible.

Després d'haver efectuat la connexió i intercanviar la informació necessaria per al control remot, el MOC pot sol·licitar informació específica o *stream* dels sensors, donat que aquesta informació ja és coneguda per part del port. El format del missatge en aquest cas és més flexible que l'anterior, podent demanar

tants elements com es desitge en un sol missatge. Un exemple d'aquest tipus de missatge seria el següent:

```
{"request":
  {
    "conning": {
      "priority": true,
      [...]
    },
    "camera0": true,
    "sensor2": true,
  }
}
```

Figura 3.6: Missatge exemple de demanda d'informació sobre el vaixell

Més tard s'envien missatges per sincronitzar els rellotges i controlar la latència, aquest és un missatge que s'envia de manera regular cada cinc segons. Un exemple d'aquest missatge enviat pel MOC (primer en enviar el missatge) seria el de la figura 3.7, i la resposta del vaixell seria com a la figura 3.8.

```
{"time": {"sent": 1608550017650}}
```

Figura 3.7: Missatge exemple de latència del MOC

```
{"time": {"sent": 1608550017650, "received": 1608550017680}}
```

Figura 3.8: Missatge exemple de latència del vaixell

Els següents passos que podem veure al diagrama d'activitat de la figura 3.3 són prescindibles per al transcurs del projecte, només és necessari entendre que hi ha certes paraules clau a l'hora de notificar un problema a bord (*debug*, *info*, *caution*, *warning*, *alarm* i *emergency*) depenent de la gravetat i prioritat del problema.

També hi ha certes paraules clau per part del vaixell al MOC a l'hora de requerir guia (*advice* per demanar informació útil per la navegació, *recommendation* per demanar recomanacions de navegació i *control* per demanar al MOC que prenga el control complet de la nau) depenent del nivell de control necessari per part del MOC. El vaixell sempre ha de requerir un d'aquestos tres modes de control en el primer missatge que intercanvie amb el MOC i el mateix MOC deu satisfer-lo dins de les seues capacitats. Per tant, un missatge en el que el vaixell desitjara tindre recomanacions per part del MOC per a la navegació dins del port quedaria com el de la figura 3.9.

```
{ "guidance": "recommendation" }
```

Figura 3.9: Missatge exemple de requeriment del nivell de control *recommendation*

Abans d'anar amb el missatge de terminació és important també conèixer un tipus especial de missatge que no apareix al diagrama, ja que es considera informació com podria ser qualsevol fluxe de dades que es demanara. Aquest és el *conning* que apareix a la figura 3.6 a la línia 3. Aquest tipus de missatge ha d'incloure la informació del GPS actual (latitud, lonngitud, direcció, velocitat sobre el fons i rumb sobre el fons), i si estan disponibles, també les dades sobre la velocitt del vent, el timó i la propulsió. A més pot afegir-se qualsevol informació que es considere d'interés.

Per últim queda la despedida, la terminació de la comunicació, que es fa amb un simple *bye*, o adéu en anglés. A part, també s'especifica la *URL* del MOC amb el que es termina la comunicació, en cas de que fóra el MOC qui terminara la connexió, en lloc de la *URL* s'especificaria l'MMSI del vaixell. Així doncs quedaria un missatge com el següent si el vaixell decidira terminar la connexió:

```
1 {"bye": {"url": "wss://[amoc].aboamare.net "}}
```

Figura 3.10: Missatge exemple de despedida

3.3 Disseny de l'adaptador

El disseny de l'adaptador es basa en el protocol vist anteriorment i alhora en l'estructura del dispositiu físic, el vaixell sobre el que es treballa, l'*Åboat*. Com el protocol ja l'hem vist, ens centrarem ara en les connexions entre el vaixell i l'adaptador, que com veurem són molt senzilles, per a això hem de conèixer els components de l'*Åboat* que s'observen a la figura 3.11. Aquests són poquets, ja que el vaixell sobre el que treballem és una construcció de dimensions reduïdes. Observem que tenim dos sensors IMU (un extern i un intern), que són sensors d'acceleració. Tenim també LIDAR, que és un sensor que genera núvols de punts on cada punt representa la distància de l'objecte que hi ha en eixa direcció, dit d'un altra manera, és com un RADAR en 3 dimensions que funciona amb tecnologia làser. Hi ha una brúixola (*compass*), un GPS, telemetria sobre la connexió a la xarxa LTE, una càmera d'infrarrojos i una càmera 360º.

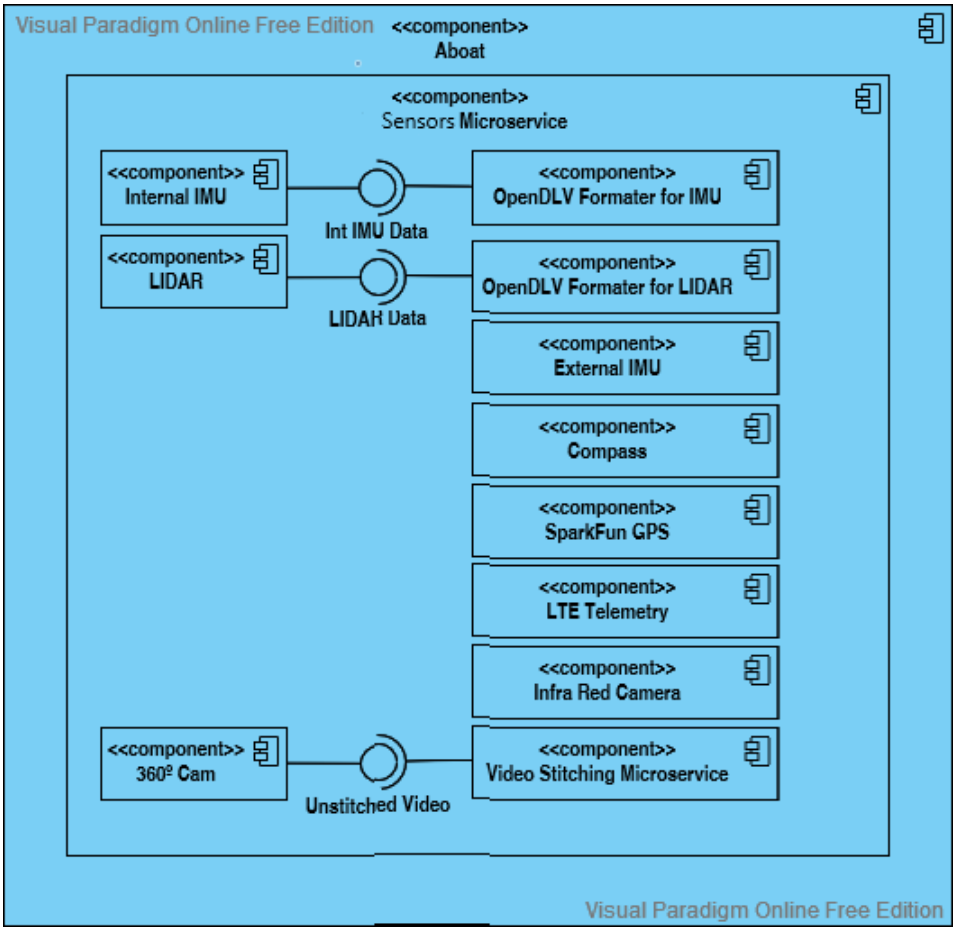

Figura 3.11: Components de l'*Åboat*

Apart dels components catalogats com sensors, disposem també d'un rellotge i del router, aquests dos elements són condició necessària per al correcte funcionament de l'adaptador en tots els vaixells en els que s'instal·le (es pot veure a la figura 3.12). Aquests missatges, els generats pel rellotge i el router, passen directament des del vaixell al MOC sense tractar-se com informació a traduir per l'adaptador, ja que formen part de l'establiment de la connexió, i no de la informació requerida. Dit d'una altra manera, els missatges no estan formatetjats en *odvd*. El seu tractament doncs ha de ser diferent al dels missatges que es generen a partir dels sensors i càmeres[1]. Aquest tractament es fa directament a la classe *websocket_client*, ja que els possibles missatges enviats pel router són el de demanda de connexió, a la figura 3.4, i el de terminació de la connexió, a la figura 3.10. Per altra banda, el rellotge envia només el temps exacte, i és l'adaptador qui emmagatzema els temps de recepció per enviar-los de tornada com s'observa a la figura 3.12.

[1]Les càmeres són en realitat un tipus de sensor també

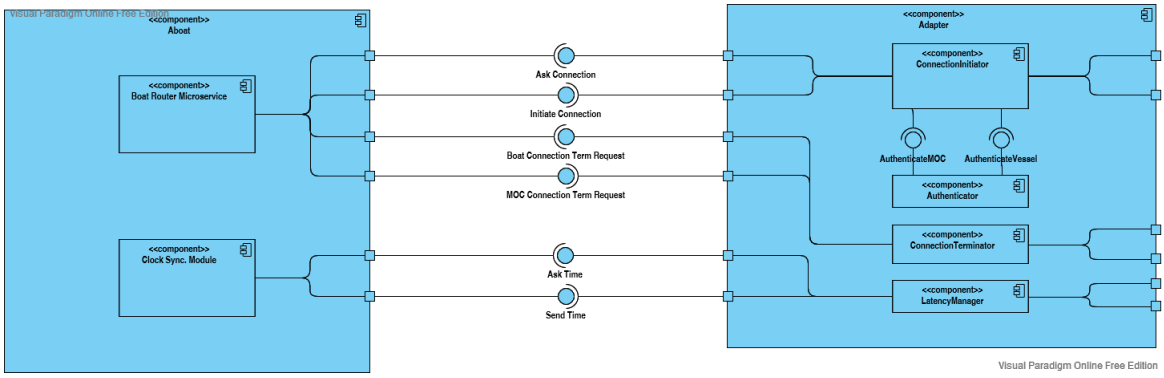

Figura 3.12: Components condició necessària de l'*Åboat*

Tots els components envien la seua informació en *broadcast* dins de la mateixa direcció UDP, per tant és l'adaptador qui ha de discernir a quin element correspon cada missatge depenent del seu identificador i actuar en conseqüència. Com tots els elements envien informació sense importar si aquesta ha sigut requerida o no, i amb una latència diferent, la millor opció per tindre aquesta informació llesta per si es demanada és crear una classe de estat, la classe *state*. Per tant el diagrama de classes que observem a la figura 3.1 quedaria actualitzat al de la figura 3.13

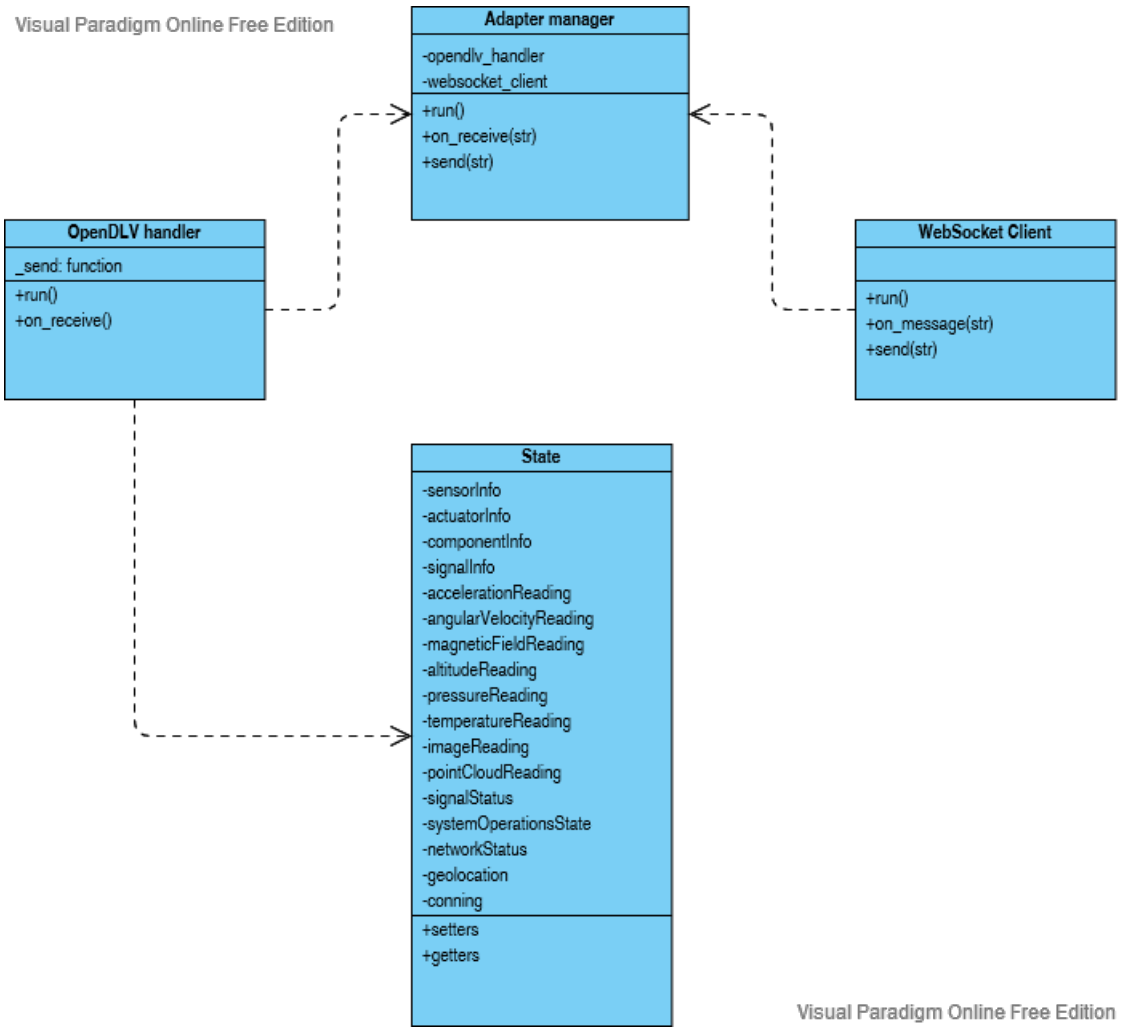

Figura 3.13: Diagrama de classes de l'adaptador actualitzat

La novetat ací és la classe *state*, que inclou tots els atributs necessaris per capturar l'informació mentres es va actualitzant. Els únics mètodes existents són per tant els *getters* i *setters* que serviran per anar modificant la informació al ritme que es va rebent i per a llegir-la quan es requerisca. A més, s'ha afegit un atribut *conning* que s'actualitza cada vegada que algun dels elements del *conning* es rep en la sessió.

Com hem dit, tots els missatges són rebuts a la mateixa sessió, però no tots han de ser tractats d'igual manera, això ens du a la necessitat de crear un gestor de missatges, que incloem directament dins de la funció *on_receive(str)* ja que la identificació per ID és molt senzilla. Una vegada tractat el missatge, s'envia a l'*AdapterManager* en el format que es requerisca depenent del tipus de missatge. Això ho podem observar al següent diagrama 3.14:

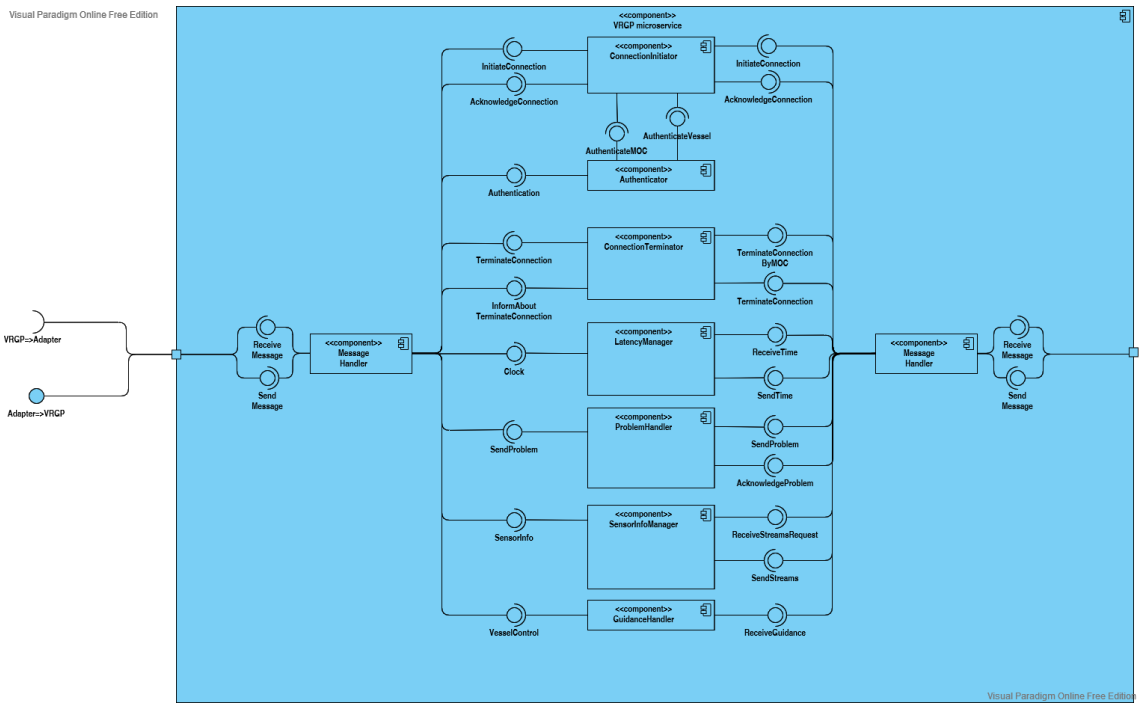

Figura 3.14: Diagrama de components de l'adaptador

Es poden observar moltes connexions que poden resultar confuses, però al final no és més que el *message handler* classificant els missatges segons el tipus i redirigint-los al gestor corresponent, després aquest l'envia a l'*AdapterManager*.

Amb això ja tindriem tots els ingredients per començar la implementació de l'adaptador.

3.4 Implementació de l'adaptador

A l'hora d'implementar l'adaptador VRGP hem de tindre en compte quines tecnologies i llenguatges anem a usar, a més d'adherir-nos als diagrames i estudis previs que hem fet abans. Una de les condicions primeres per part dels clients era utilitzar *OpenDLV*, *WebRTC* i *WebSocket Secure*, d'aquestes tres, només eren indispensables des de l'inici *OpenDLV* i *WebScocket Secure*. Això ens deixa front un panorama en que, per compabilitat a l'hora d'utilitzar OpenDLV i les llibreries capaces de llegir els missatges *odvd*, la millor opció és implementar l'adaptador en llenguatge C++, un llenguatge sòlid, orientat objectes i funcional en quasi totes les màqiunes en les que s'instalaria. També hem de tindre en compte que la implementació final es fa sobre una *Rasberry Pi 4* que ens obliga a cuidar l'ús de la memòria. Aquesta limitació seria un problema si no usarem una classe *state* a l'hora de guardar les últimes lectures dels sensors.

Tenim quatre classes a implementar, anem a enfocar-les de dalt a baix, començant pel gestor, l'*adapter_manager*. Com es pot observar al diagrama de classes (figura 3.13), el gestor és una classe molt senzilla amb tres mètodes: *run()*, *send(str)* i *on_receive(str)*. El mètode *run()* genera dos fils, un per al *WebSocket* i l'altre per a la sessió *OpenDLV*. Per l'altra banda, *send(str)* i *on_receive(str)* actuen a mode de *pipeline* entre les classes *opendlv_handler* i *websocket_client*. *Send(str)* és l'encarregada de dirigir els missatges des de la classe *opendv_handler* a la classe *websocket_client*, mentres que el mètode *on_message(str)* actua en la direcció oposada.

La classe *websocket_client* també té tres mètodes. El primer d'ells, el mètode *run()* inicia el client *websocket*, aquest mètode té certa complicació en el sentit de que no tenim control sobre el *loop* del *websocket* de manera que no el podem aturar quan nosaltres volem. Per solucionar aquest inconvenient iniciem un nou fil que corre paral·lelament a la connexió i observa constantment una variable booleana *_done*. En cas que aquesta variable siga verdadera, la connexió es tancarà (el bucle s'aturarà). Aquesta variable pot ser modificada des de la classe *opendlv_handler* quan aquesta reba un missatge del vaixell demanant terminar la connexió. Després tenim la funció *send(str)* que envia el missatge que es passe com argument mitjançant el *websocket*. Per últim està la funció *on_message*, que al rebre un missatge pel *websocket* el procesa i el redirigeix a la classe *opendlv_handler* mitjançant la funció *on_receive* de l'adaptador.

Les classes *opendlv_handler* i *state* és millor veure-les juntes, ja que la classe *state* no té cap complicació al ser un magatzem temporal de dades. L'objectiu de l'*opendlv_handler* es el de traduir els missatges *odvd* en JSON seguint les pautes que estableix el protocol. Imaginem per exemple que rebem un missatge amb informació de l'acceleròmetre com el de la figura 3.15:

```
1 opendlv.proxy.AccelerationReading {
2   accelerationX = 245.4;
3   accelerationY = -0.38;
4   accelerationZ = 15.2;
5 }
```

Figura 3.15: Missatge d'acceleracióI

Amb la llibreria *cluon* tenim una manera molt senzilla d'extraure les dades de cada sensor, primer hem de "desempaquetar"el missatge en un nou objecte, després simplement accedim als arguments d'aquest objecte un a un. La funció per extraure les dades del missatge anterior seria com es mostra a la figura 3.16:

```
1  auto onAccelerationReading = [](cluon::data::Envelope &&env){
2              opendlv::proxy::AccelerationReading msg = cluon::
                   extractMessage<opendlv::proxy::AccelerationReading>(
                   std::move(env));
3            json accelerationReading;
4
5              accelerationReading["accelerationX"] = msg.accelerationX
                   ();
6              accelerationReading["accelerationY"] = msg.accelerationY
                   ();
7              accelerationReading["accelerationZ"] = msg.accelerationZ
                   ();
8
9              stateInfo.setAccelerationReading(accelerationReading);
10         };
```

Figura 3.16: Funció per extraure les dades del missatge *odvd* de l'acceleròmetre

Un detall a remarcar és que aquesta funció és específica per a les dades d'acceleració, si necessitarem llegir un missatge del GPS hauriem de fer una nova únicament per al GPS. És per això que aquesta funció, integrada al mètode *run()*, s'activa amb un *trigger* específic per quan arriba un missatge amb l'identificador de l'acceleròmetre, i hi ha una funció i el seu corresponent *trigger* per a cada tipus de missatge. Totes tenen la mateixa estructura, creen una variable tipus *json* (que és una *string* de una llibreria externa utilitzada per simplificar-ho tot a l'hora de construir els JSON), després estableixen cadascun dels seus paràmetres amb el valor del missatge i es guarda a la classe *state*.

El segon mètode, *on_receive(str)*, analitza la *string* que rep d'entrada en busca de certes paraules clau. Sabem que el missatge rebut està formatetjat en *json*, per tant, primer buscarem si demana informació o per contra demana el control del vaixell (hi ha més comandaments al protocol, però no ens extendrem en explicar el funcionament ja que al final és el mateix patró). En cas de que demane informació, llig quina és la informació que es demana i la envia llegint-la des de la classe *state*. En cas de que el que es demane siga el control del vaixell, el que es fa és enviar les instruccions en format *odvd* al vaixell, que és el procés invers al vist en l'exemple anterior. Les paraules clau són les mostrades a la taula 3.2. Cadascuna d'aquestes paraules contitueix un bloc al qual poden haver diversos atributs. En cas de que existiren aquests atributs al missatges, també s'hauria de llegir i traduir (de *json* a *odvd*) el seu contingut. Un exemple d'això pot ser el missatge *request*, que necessita d'atributs (mínim 1) per a que l'adaptador puga enviar a informació que es requereix.

Amb això ja tindriem completada la implementació de l'adaptador VRGP, només queda testejar-lo en un entorn controlat abans de poder aplicar-lo al món real.

Paraula	Descripció
request	Demana informació, ja siga sobre els sensors, *streaming* en directe o sobre la situació general del vaixell
acquire	Demana permís per controlar el vaixell, es respon amb un missatge *controls* que dona el permís
message	Missatge informatiu genèric amb una *string*, pensat per ser llegit per humans
operator	Missatge informatiu per donar a conèixer qui és l'operador encarregat del control
command	Missatge amb un comandament de guiat
takeover	Finalització del control remot del vaixell
bye	Finalització de les comunicacions

Taula 3.2: Paraules clau als missatges del MOC al vaixell

CAPÍTOL 4
Entorn experimental

L'entorn experimental el podem dividir en dos entorns diferents. El primer és la connexió entre un vaixell simulat i un MOC simulat, en el que el arribarem a enviar missatges complexes com són la informació dels sensors o la senyal de les càmeres. El segon entorn per altra banda és un entorn on la connexió es realitza entre entre l'*Åboat* i un port simulat. Per raons de temps i de l'ús compartit del vaixell amb altres grups d'investigació i projectes, el testeig amb el vaixell real es va reduir només a l'establiment de la connexió. Per tant els missatges dels que es fa ús en el conjunt físic vaixell-simulador són els de connexió i terminació. En abdós casos, les connexions es feren utilitzant direccions reservades per a l'entorn experimental, no sent aquestes les direccions futures dels bucs reals.

Comencem amb l'entorn enterament simulat. Per simular l'*Åboat* no es necessita crear un gran programa de complexitat excelsa. Com hem vist abans, les dades es reben dins d'una sessió UDP local. L'únic que es necessita llavors és enviar missatges dins de la direcció associada a eixa sessió. En el nostre cas, per establir la mateixa sessió *OpenDLV* que al vaixell real al laboratori, la direcció escollida és 255.0.0.150. Per enviar la informació obrim el *WebSocket* enviant els paquets al port 8080, on el MOC escolta per prestar el servei, i sent la direccció IP aquella que li correspon a l'ordinador que simula el MOC, que no és fixa al ser diferents ordinadors portàtils personals els que executen aquest simulador. Com l'objectiu és poder provar tant com es puga, els missatges s'envien des d'un petit programa de consola amb opció de seleccionar els diferents comandaments a enviar. D'aquesta manera enviàrem missatges personalitzats, enviant informació tant correcta com no per comprovar el correcte funcionament del programa. És important el fet d'enviar informació errònia per observar que el comportament és estable.

Les proves es basaren en, primerament, establir una connexió. Una vegada la connexió estigué establida, per part del simulador del vaixell generarem informació sobre els sensors que no sigué enviada fins que des del MOC es va requerir, veient així que la classe *state* feia la seua funció com estava previst, guardant l'últim estat dels diferents sensors i evitant així haver d'esperar a un nou enviament d'informació per satisfer les necessitats del MOC. De tant en tant s'enviaven missatges per part del MOC amb instruccions o requeriments inexistents, els quals l'adaptador simplement descartava, que és el comportament dessitjat. Per últim s'envià el missatge de tancament de la connexió i es donà per finalitzada la prova en l'entorn enterament simulat.

Després començaren les proves a l'entorn real. A l'aparell físic el concepte és el mateix que a la simulació del vaixell. Instal·lem el programa, per a major simplicitat contingut en un *Docker Container*[1], i l'executem. L'únic detall a tindre en compte és la complexitat del vaixell, el projecte ací va molt més enllà que una sessió UDP. Però al final el concepte és el mateix, instal·lar el programa, obrir la sessió i el *Websocket* i, en aquest cas, actuar només des del costat del MOC, ja que el vaixell sí que és autònom. Per a la instal·lació de l'adaptador s'ha de fer mitjançant SSH ja que la raspberry instal·lada no té cap interfície computadora-humà.

Figura 4.1: Fotografia de la web FiTech de l'Åboat

El vaixell, a la figura 4.1, està compost d'un motor (amb previsió d'ampliar a dos, un per costat) que podem observar a la dreta de la imatge com un apéndix blanc a l'embarcació; un sensor LIDAR[2] ubicat al centre al punt més alt del màstil; quatre càmeres ubicades just davall del LIDAR que donen una imatge de 360º; un GPS i dos acceleròmetres dins la caixa negra junt a la bandera, caixa on també es troben la Raspbery Pi i un router; i per últim dos antenes 4G, junt a la caixa negra que apareixen com dos cilindres blaus. Amb aquest equipament l'objectiu era provar la connexió i enviar correctament i amb el menor retard possible les senyals i informació dels diferents sensors.

Malauradament degut a les limitacions en l'accés al vaixell, les proves realitzades en l'entorn físic es varen reduir fins el punt de testejar només l'establiment de la connexió, el qual sigué satisfactori.

[1] *Docker Container* és un *software* que agrupa el codi i totes les seues dependències amb l'objectiu d'executar l'aplicació ràpidament sense haver de compilar manualment. Aquest *software* funciona tant en Linux com en Windows.

[2] LIDAR és un sensor utilitzat per mesura distàncies utilitzant un fes làser pulsat.

CAPÍTOL 5
Seguretat

La seguretat en un projecte on es tracta el control de sistemes crítics com són els vaixells és fonamental. En aquest projecte, degut a la limitació de temps i de recursos humans, ens hem centrat en agafar ferramentes ja existents per asegurar el sistema, encara que la situació ideal seria la de crear tot un sistema de seguretat específic per al protocol i les comunicacions.

A l'hora d'analitzar els riscos als que ens enfrontem hem de tindre en compte que estem treballant en un ecosistema molt fràgil, on un error pot suposar la pèrduda de vides humanes, apart de l'impacte econòmic i ecològic que aquest pot arribar a tindre. Parlem d'una infraesructura que històricament ha sigut estratègica en guerres i que per tant pot ser blanc de *hackers*, terroristes o governs contraris. Amb aquestes bases, encara que hem de tenir en compte que el projecte actual no arriba a eixa escala global ja que és experimental encara, s'ha dissenyat la següent matriu de riscos (veure figura 5.1) on s'han establert diferents escenaris depenent de la seua probabilitat d'ocurrència i l'impacte que suposaria.

També és important diferenciar els actors que hi ha dins la comunicació, ja que el sistema pot ser vulnerable per si mateix, però també cadascun dels actors (vaixell i MOC) poden ser vulnerables de manera individual.

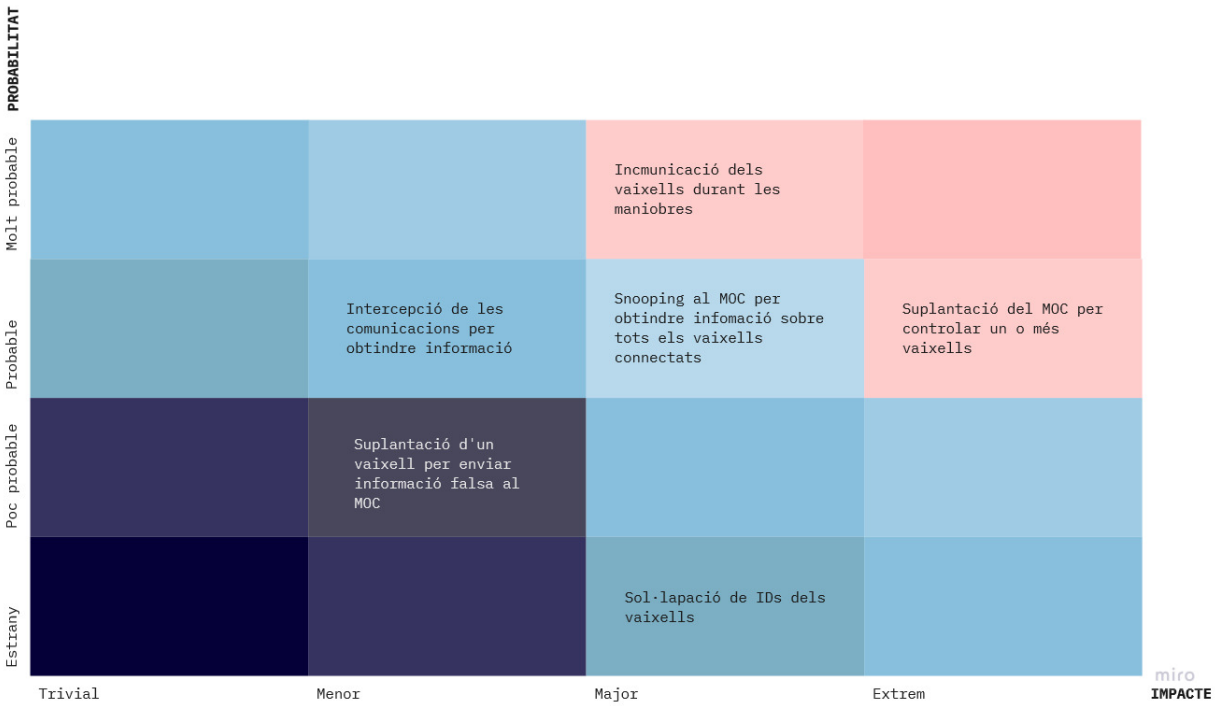

Figura 5.1: Matriu de riscos del sistema VRGP

Podem observar en la matriu que hi ha riscos tant relatius a atacs externs com a fallades del sistema. Ara explicarem com s'han abordat aquestos riscos amb les capacitats limitades que es tenien i quines solucions es podrien oferir amb un equip complet i temps.

Les situacions més crítiques, per probabilitat i impacte, són la de una incomunicació dels vaixells durant les maniobres, ja siga per atac extern o per fallada de la comunicació, i la de la suplantació del MOC amb l'objectiu de controlar els vaixells que a ell estiguen conectats. Són dos escenaris completament diferents, així que s'han de tractar com a tal.

Per evitar que una possible incomunicació dels vaixells fóra crítica el que es fa és passar automàticament a control manual en cas de que fós possible donades les característiques de la embarcació. En cas de que el vaixell siguera autònom passaria a control del vaixell amb indicacions estrictes d'aturada. Per protegir-se de possibles atacs DDoS[1] per part del MOC el que fem és rebre les propostes de connexió en un fil separat d'aquell que tracta les connexions ja establertes, aquest fil, abans de saturar, tanca l'entrada possible de noves connexions però mantenint aquelles que ja estan actives. Aquesta sol·lució no és ideal, però és la escollida donats els pocs recursos disponibles. Per part del vaixell no hi ha problema amb aquest tipus d'atac ja que sempre és ell qui demana establir connexió amb el MOC i no al revés.

Per evitar la suplantació de qualsevol de les parts el que s'utilitzen són certificats electrònics que identifiquen a cadascuna de les parts, tenint cada MOC i cada vaixell una certificació única Amb aquesta solució també sol·lucionem el problema que podria causar que dos vaixells tingueren una identificació idèntica.

[1]Un atac DDoS és un atac disribuït de denegació de servei mitjançant el qual l'atacant causa una caiguda del servei.

Tot i així, cap la possibilitat de que un atacant entrara directament al MOC i des d'allà controlara els vaixells o robara informació (*snooping*). Per aquest problema no tenim una solució desenvolupada. La idea més lògica seria aillar els sistema, fer que la xarxa fós tancada, però donat el gran trànsit de diferents vaixells que es dona a un port, aquesta seria una solució d'implementació difícil i costosa, i va més enllà del que aquest projecte abasta.

Respecte a un possible *Man in the Middle*[2] la solució passa per les tecnologies *WebSocket Secure* anomenada al capítol 2.5. Amb l'encriptació de les comunicacions dificultem l'a lectura i modificació dels missatges enviats per ambdues parts.

Com es pot veure, la seguretat no és el punt fort del projecte ja que aquest és encara experimental. Abans de dur el protocol al món real hauria de fer-se un estudi molt més incisiu en la seguretat.

[2] *Man in the Middle* és un atac on l'atacant és capaç de llegir, insertar i modificar els missatges de la comunicació.

CAPÍTOL 6
Conclusions

Durant les pàgines anteriors hem vist l'evolució del projecte de l'adaptador VRGP des del per què de la idea fins ala implementació en un medi físic. Vist en persepectiva pot semblar que no és un gran assoliment: hem establert una connexió entre dos màquines mitjançant internet. No és una fita especialment remarcable i molt menys única. Però tot el treball va més enllà d'establir una connexió entre un port i un vaixell, tampoc és crear un control remot, no al menys un qualsevol.

El resultat d'aquest treball que ha sigut el treball de tot un any i un equip de set persones és un protocol de comunicació genèric amb un adaptador per universalitzar el mateix protocol sense necessitar de canvis sigificatius als vaixells ja existents. Tot això amb la simulació del port i una proposta d'interfície d'usuari per als pràctics del port. Però anem a centrar-nos en les conclusions només de la part del projecte corresponent a aquest treball de fi de grau: l'adaptador del protocol.

Encara que l'adaptador dissenyat no és tan complex com ho seria per un vaixell de les dimensions d'un carguer transatlàntic o d'un creuer, podem estar satisfets amb el resultat, ja que l'adaptador està fet pensant en que siga de fàcil modificació per adaptar-se a qualsevol vaixell, gran o petit; simple o complex. I això sí que ho compleix, modificar les funcions que actuen de traductores és tan simple com adaptar-les als valors que s'envien des dels sensors de cada vaixell concret. Després aquestes dades s'envien ja formatetjats de manera que el port sempre tinga el mateix format de missatge.

Amb aquesta petita peça que és l'adaptador aconseguim dos objectius: que els vaixells no hajen de tindre centenars de protocols instal·lats depenent del port al que atraquen i que els ports no tinguen centenars de protocols instal·lats depenent dels vaixells que atraquen. Aquesta modularitat permet que es controle des de qualsevol estació, qualsevol buc, permetent als pràctics treballar des de sales en terra ferma i reduïnt el risc d'accidents laborals i embussos.

Tot i això, encara queda molt treball per davant, implementar l'adaptador és la part més senzilla, però ara és qüestió de l'*Åbo Akademi* i *Åboamare* demostrar la seua viabilitat econòmica i pràctica en situacions reals. Entrant en consideracions personals, crec fermament que el futur de la navegació és a base de vaixells autònoms i/o controlats de manera remota, si és amb aquest protocol o amb un altre és una qüestió difícil d'esbrinar.

6.1 Problemes durant el projecte

L'evolució del projecte ha estat plena de problemes a resoldre, tant de mètode de treball com de coneixement de les tecnologies o de funcionament del prototip. Per a recòrrer aquests de la manera més senzilla possible, anem a anar per parts, començant pels problemes en el mètode de treball.

Per comprendre aquest punt és necessari conèixer la composició de l'equip que hem treballat al projecte, on érem tres espanyols, un rumà, un alemà, un anglés i un bangladeshià. Juntament amb els nostres clients, amb qui estavem en constant comunicació, que eren finesos. Aquesta és una combinació de cultures i maneres de treballar i enfocar els problemes molt diferents. Encara que la llengua no era barrera en cap moment degut a l'ús comú de l'anglés, la cultura sí que ho era. Per establir un mètode de treball comú i estable acudírem a *scrum*, el marc de treball per desenvolupament àgil del que hem parlat al capítol 1.3. Tot i així, l'adaptació a aquest mètode ha sigut asimètrica, amb components de l'equip que s'han adaptat molt ràpid i uns altres als que els ha costat considerablement més temps. En aquestos casos el que s'ha fet ha sigut descarregar de treball a aquells components que anaven més endarrerits en pro d'aquells que ja havien complit amb la seua part. La solució no és ideal ni molt menys, però ha servit com a pedaç momentàni fins que tot l'equip es posara al mateix ritme.

Per altra banda ha estat el problema de la comunicació inicial amb els clients. Ací entra sobretot el nul coneixement per part de tots els components de l'equip sobre el projecte. Ningú a l'equip tenia coneixement previ sobre navegació o ports, així que totes les reunions inicials eren per posar al dia a l'equip en tot allò relacionat amb els objectius del projecte, des de l'explicació de la funció dels pràctics fins al concepte del MMSI.

També en certa manera relacionat amb els clients està el fet de tindre certes tecnologies exigides que no coneixiem ningú de l'equip. Cert és que tots haviem utilitza *WebSockets* durant la carrera i que també coneixiem JSON, però ningú havia escoltat parlar de *WebRTC* o *OpenDLV*. Més que un problema va suposar un temps d'adaptació i aprenentatge que es va resoldre dividint les tecnologies entre els components de l'equip. Tres persones aprendrien de *WebRTC* i la resta de *OpenDLV*, una vegada adquirits els coneixements necessaris, ens ensenyariem entre nosaltres allò aprés. D'aquesta manera estalviariem un poc de temps al no haver de començar desde zero ambdues tecnologies.

Per útlim, l'hora de provar el prototipus va ser un constant xocar-se contra un mur, encara que a cada intent aquest xoc era més i més lleu. L'hora de provar el codi en l'entorn físic real és on es troben els errors lògics que no es veuen durant la compilació i, al primer intent, no establia connexió. Llavors haguérem de parar, revisar el codi, corregir i tornar a intentar. El fet de que les poves siguen sobre un vaixell real fa que siga molt més lent tot el procés de testeig del software ja que no parlem d'un ordinador al qual podem aturar les tasques, modificar i tornar a intentar. En aquest cas haviem de modificar els arxius al nostres ordinadors, enviar-los al vaixell i compilar-los una vegada estigueren al vaixell. Això a part de la necessitat d'anar a l'oficina d'*Åboamare* cada vegada que necessitarem provar el codi. Tot aquest procés consumia molt més temps que sobre simulacions, però també ens donava resultats reals sobre un entorn físic.

6.2 Aprenentatge

Quasi des del primer dia del projecte tot ha sigut aprenentatge, tant en la tècnica i les tecnologies com en el mètode de treball i les relacions interpersonals dins de l'equip.Començarem per les habilitats tècniques adquirides, ja que és més fàcil discernir-les.

En l'apartat tècnic, el coneixement adquirit més obvi és l'aprenentatge sobre navegació autònoma. En un context global, he après sobre l'estructura i funcionament dels ports i les normatives a l'hora d'atracar el vaixells.Però més concretament he aprés sobre tecnologies utilitzades pel control dels vehicles autònoms, com és *OpenDLV*. Utilitzant aquesta ferramenta no només he fet ús de la mateixa, sinó que m'ha forçat a utilitzar el llenguatge C++, llenguatge que havia usat en contades ocasions durant la carrera però que no dominava. No ha sigut difícil aprendre aquest llenguatge tenint en compte que sí que havia usat amb freqüència abans llenguatges com C o *Java*, però és un element a destacar.

Seguint amb les tecnologies apreses, abans de començar el projecte no coneixia *WebRTC*, i encara que en la part corresponent a aquest treball, l'adaptador, no l'he utilitzada, sí que és cert que he hagut d'estudiar-la i comprendre el seu funcionament per entendre el projecte com el conjunt que és.

A més, i gràcies també a una assignatura[1] cursada al mateix temps que el projecte, he pogut desenvolupar un coneixement més profund sobre els sensors LIDAR i acceleròmetres, que després han sigut d'utilitat al mateix projecte donades les característiques del vaixell autònom amb el que treballàvem.

En un aspecte més abstracte ens trobem amb el que a la Universitat Politècnica de València anomenem competències transversals. És difícil estabir fins quins punts de domini he arribat en aquelles i quines ja tenia apreses, però sí que hi ha certes competències que cal destacar.

La primera d'elles és la de diseny i projecte. Al treballar en un projecte real, amb uns clients reals, he aprés com funciona la dinàmica de planificació i disseny en un període de temps determinat com és un curs escolar. A més, he aprés a tractar amb clients i discutir les característiques desitjades per arribar a un punt comú. És cert que ha sigut especialment fàcil en certes ocasions el tractament amb els clients ja que l'àmbit de treball al qual es dediquen és molt similar i per tant entenen les limitacions existents.

Per suposat aquesta qualitat inherent al projecte que era treballar en un equip format per set estudiants de diferents nacionalitats ha reforçat el treball en equip. De fet, he aprés una qualitat que crec és molt útil, i que també s'englobaria en la comunicació efectiva (encara que parlarem d'aquesta un poc més avant), i és el fet de ser capaç de mediar entre parts. Amb un equip tan gran no era d'estranyar que hi haguera divergències, i per solucionar-les moltes vegades es necessitava d'algú que intercedira i aclarira el dos punts de vista. En ocasions eixe mediador era jo, en ocasions era un altre component de l'equip, però en qualsevol cas considere que és una qualitat molt útil.

[1] L'assignatura en qüestió és *Multidimensional Sensing Techniques* cursada a l'*Åbo Akademi*

Abans parlava de comunicació efectiva, i justament aquesta era un àrea que em corresponia a mi a l'equip. Durant el tanscurs del projecte hem hagut d'anar a diferents fires a fer *pitches*[2] front a empresaris i professors finesos. Escriure, practicar i finalment exposar aquests breus discursos va ser de gran utilitat per perfeccionar l'habilitat de comunicar idees complexes de la manera més senzilla possible, ja que la majoria dels presents no sabien res sobre protocols de guiat de vaixells ni sobre les tecnologies que utilitzem.

Les competències de coneixemen de problemes contemporànis i d'aprenentatge permanent han estat lligades durant el transcus del projecte. En tota la carrera mai havia estat aprenent d'una manera tan contínua. M'explique, quan s'estudia durant la carrera habitualment es tenen uns períodes d'examens ja establerts, i encara que no és una bona pràctica, la majoria hem deixat per a les últimes setmanes l'estudi dels temaris. Això no era possible mentres treballava al projecte, estava contínuament forçat a aprendre, ja fora sobre la tecnologia, sobre les implicacions econòmiques que suposaria o sobre els problemes portuaris actuals.

També cal mencionar la capacitat d'anàlisi i resolució de problemes, ja foren petits problemes lògics al codi o obstacles tècnics per les limitacions de l'embarcació, tot el projecte ha sigut una constant batalla per resoldre problemes. El més important que m'he dut d'aquesta competència és aprendre a parar. A vegades pareix contraintuitiu, però encara que còrrega presa perquè el *deadline* s'apropa, el millor és parar, allunyar-se del problema i mirar-lo amb distància, preguntar si és necessari, i continuar. Al final els problemes, amb calma, es resolen d'una manera molt més eficaç i efectiva, en lloc d'estar donant voltes sobre la mateixa idea sense arribar enlloc. Això també podria incloure en certa manera la competència de planificació i gestió del temps.

Tirant la vista enrere, el projecte ha sigut suficientment complet i exigent per aprendre, en definitiva, com és treballar en un entorn real, amb tot el que implica.

[2] Un *pitch* és un discurs breu, d'uns dos o tres minuts, amb l'objectiu de vendre el projecte o producte que es desenvolupa

6.3 Relació amb els estudis

Per últim entrem en la relació amb els estudis. Quasi tot el projecte de fet té una relació directa amb els estudis, sobretot amb les branques de xarxes i d'enginyeria dels computadors. Per suposat, el projecte ha requerit d'habilitats de programació, necessàries per dur a terme tota la implementació, habilitats adquirides durant tota la carrera en assignatures com Programació, Estructura de Dades i inclús Sistemes Operatius (ambdues assignatures). Les diverses assignatures de xarxes han sigut imprescindibles alhora de contruir els *WebSockets* i establir les connexions. També cal destacar el paper de les assignatures de programació paral·lela que, en menor mida, també han sigut útils a l'hora d'implementar tot el sistema.

Especial menció mereixen les assignatures de gestió de projectes, que tenen una relació directa amb el, valga la redundància, projecte. Des de la planificació al disseny dels diferents components, tot ha sigut una aplicació pràctica del que s'imparteix a les citades assignatures.

També considere important mencionar la relació del projecte amb les assignatures de Tècniques de Sensors Multidimensionals i de Acceleradors Hardware per Robòtica i IA, assignatures cusades a l'*Åbo Akademi* i l'Universitat de Turku respectivament i amb les que el projecte guarda una relació directa. Tant en la comprensió de les dades i informació que s'envien durant les comunicacions establertes com en l'enteniment del funcionament dels vehicles autònoms.

Bibliografia

[1] C. Berger, "OpenDLV - A modern microservice-based software ecosystem for self-driving vehicles", *OpenDLV Github*, 4 d'agost 2019. Accessible des de: https://github.com/chalmers-revere/opendlv/blob/new/README.md

[2] C. Holmberg, S. Hakansson & G. Eriksson, "Web Real-Time Communication Use Cases and Requirements", *RFC 7478*, març 2015.

[3] C. Jennings, H. Boström & J.I. Bruaroey, "WebRTC 1.0: Real-Time Communication Between Browsers", *W3C*, 26 de gener 2021. Accessible des de: https://www.w3.org/TR/webrtc

[4] E. Ezhilarasan & M. Dinakaran, "A review on mobile technologies: 3G, 4G and 5G", *Second International Conference on Recent Trends an Challenges in Computational Models*, 2017. DOI: 10.1109/ICRTCCM.2017.90

[5] E. Jokoinen, et al, "Remote and autonomous ships. The next step", *Rolls-Royce Marine*, 2016. Accessible des de: https://www.rolls-royce.com/˜media/Files/R/Rolls-Royce/documents/customers/marine/ship-intel/aawa-whitepaper-210616.pdf

[6] H. Alvestrand, "Overview: Real-Time Protocols for Browser-Based Applications", *RFC 8825*, gener 2021.

[7] H. Alvestrand, "Transports for WebRTC", *RFC 8835*, gener 2021.

[8] H. Schulzrinne, S. Casner, R. Freferick & V. Jacobson, "RTP: A transport Protocol for Real-Time Applications", *RFC 3550*, juliol 2003.

[9] I. Fette & A. Melkinov, "The WebSocket Protoco", *RFC6455*, desembre, 2011.

[10] International Telecommunication Union "Assignment and use of identities in the maritime mobile service", *ITU-R M.585-9*, maig 2022.

[11] J.F. Kurose & K.W. Ross, "Redes de computadoras. Un enfoque descendente", *Pearson*, 2017. ISBN: 978-84-9035-528-2

[12] L.C. Östergård, "A modern Maritime Training Center with a solid history", *Blue Growth in Soutwest Finland*, (s.d.). Accessible des de: https://www.sininenkasvu.fi/aboa-mare-2/

[13] Lloyd's Register, "Shipright design and construction: design code for unmanned marine systems"Febrer, 2017.

[14] M. Handley, V. Jacobson & C.Perkins, "SDP: Session Description Protocol", *RFC 4566*, juliol 2006.

[15] Mozilla, "WebSockets API", *MDN Web Docs*, (s.d.). Accessible des de: https://developer.mozilla.org/es/docs/Web/API/WebSockets_API

[16] Pilotage Maritime Dunkerque, ".Le simulateur de manoeuvre", (s.d.). Accessible des de: https://www.pilotedunkerque.fr/en/the-pilot-station/le-simulateur-de-manoeuvre

[17] Redacció BBC News Mundo, Çanal de Suez: ¿cuánto ha costado hasta ahora el bloqueo?", *BBC News Mundo*, 29 de març 2021. Accessible des de: https://www.bbc.com/mundo/noticias-internacional-56564948

[18] Scrum Org, "What is Scrum?" *Scrum.org*, (s.d.). Accessible des de: https://www.scrum.org/resources/what-is-scrum

[19] S. Tenese, "Maritime Distress and Safety System (GMDSS) Next Generation", University of South Africa, Octubre, 2019

[20] Traficom, "Ship Registration", *Traficom*, (s.d.). Accessible des de: https://www.traficom.fi/en/transport/maritime/ship-registration

[21] United Nations, "Review of Maritime Transport", *United Nations Publications, UNCTAD/RMT/2019*, 31 de gener 2020.

[22] V. Yee, "Giant Ship Blocking Suez Canal Could Take 'Days, Even Weeks' to Free", *The New York Times*, 25 de març 2021 (Actualitzat 10 d'octubre 2021). Accessible des de: https://www.nytimes.com/2021/03/25/world/middleeast/suez-canal-ship.html

[23] WebRTC, "Guies WebRTC", *WebRTC* (s.d.). Accessible des de: https://webrtc.org/getting-started/overview