UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Centro de Control de Tierra Adaptativo para Equipos de Vehículos Autónomos Heterogéneos

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Juan Antonio Bonache Seco

Directores

José Antonio López Orozco
Eva Besada Portas

Madrid

Centro de Control de Tierra Adaptativo para Equipos de Vehículos Autónomos Heterogéneos

TESIS DOCTORAL

Juan Antonio Bonache Seco

Directores: José Antonio López Orozco y Eva Besada Portas

Facultad de Informática

Universidad Complutense de Madrid

Septiembre 2019

Centro de Control de Tierra Adaptativo para Equipos de Vehículos Autónomos Heterogéneos

Memoria que presenta para optar al título de Doctor en Informática

Juan Antonio Bonache Seco

Dirigida por los Doctores

José Antonio López Orozco y Eva Besada Portas

Facultad de Informática

Universidad Complutense de Madrid

Septiembre 2019

UNIVERSIDAD COMPLUTENSE MADRID

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D./Dña. Juan Antonio Bonache Seco,
estudiante en el Programa de Doctorado Ingeniería Informática (RD 99/2011),
de la Facultad de Informática de la Universidad Complutense de Madrid, como autor/a de la tesis presentada para la obtención del título de Doctor y titulada:

Centro de Control de Tierra Adaptativo para Equipos de Vehículos Autónomos Heterogéneos

y dirigida por: José Antonio López Orozco y Eva Besada Portas

DECLARO QUE:

La tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la Ley de Propiedad Intelectual (R.D. legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, modificado por la Ley 2/2019, de 1 de marzo, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita.

Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la tesis presentada de conformidad con el ordenamiento jurídico vigente.

En Madrid, a 19 de septiembre de 2019

Fdo.: Juan Antonio Bonache Seco

Firmado digitalmente por Juan Antonio Bonache Seco
Nombre de reconocimiento (DN): cn=Juan Antonio Bonache Seco, o=Universidad Complutense de Madrid, ou, email=jabonache@ucm.es, c=ES
Fecha: 2019.09.19 09:46:19 +02'00'

Esta DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD debe ser insertada en la primera página de la tesis presentada para la obtención del título de Doctor.

A mis padres, directores
y a todos aquellos
que me han aguantado
durante este proceso

Agradecimientos

Poco o nada sabía, hace tan sólo 5 años, a cerca del duro proceso por el que tiene que pasar un doctorando desde que comienza su investigación hasta que es capaz de plasmarla en su tesis. Ha sido un largo camino en el que me he sentido como un bebé que comienza a dar sus primeros pasos en un mundo nuevo en el que todo es curioso e interesante. Aunque he aprendido mucho en el proceso y esta tesis representa la culminación de un largo trabajo, aún me siento como ese niño lleno de curiosidad que sólo está dando sus primeros pasos en un mundo en el que, con suerte, nunca terminará de aprender.

Por suerte para mí, durante este tiempo he estado rodeado de un formidable grupo de personas que han tenido a bien irme dando empujoncitos en la dirección correcta. Y es que en el Departamento de Arquitectura de Computadores y Automática y, más concretamente, en el grupo de investigación de Ingeniería de Sistemas, Control, Automática y Robótica (ISCAR), he tenido la suerte de coincidir con un magnífico grupo de personas que, además de ser brillantes en su campo, nunca han dejado de sorprenderme por su cercanía y humildad. Gracias a todos.

Quisiera destacar a mis directores, Eva Besada Portas y José Antonio López Orozco, que han sido mis grandes apoyos durante esta investigación. Ellos han sido los encargados de darme en cada momento los estímulos necesarios para continuar este trabajo (una cariñosa reprimenda por aquí, un mensaje motivador por allá, ...) y, en definitiva, de enseñarme todo lo necesario sobre el mundo

de la investigación. Siempre os estaré profundamente agradecido por todo lo que me habéis enseñado.

Mi más profundo agradecimiento también a José Luis Risco Martín, que me ayudó en un momento difícil dando las pinceladas necesarias para continuar con un trabajo que, sin él, habría tardado mucho más en empezar a dar frutos.

Me gustaría también dedicar unas palabras a Jesús Manuel de la Cruz, una de las personas más brillantes que he conocido. Mi más profundo respeto y admiración por un gran investigador y persona, amable como pocos y rebosante siempre de una energía que era capaz de contagiar a todos los miembros del proyecto SALACOM (Sistema Autónomo Para La Localización Y Actuación Ante Contaminantes en el Mar) durante aquellas largas sesiones de experimentos en Valmayor y El Atazar.

También quisiera agredecer a mis amigos más cercanos, los de siempre, que hayan estado ahí para mí cuando lo he necesitado y, por supuesto, esa pequeña familia que hemos formado en el departamento entre los habitantes de los laboratorios y los despachos que hacen que el ambiente de trabajo sea especial.

No puedo terminar de escribir estas líneas sin dirigir unas palabras de agradecimiento a mis padres, sin cuya fe y apoyo incondicional seguramente no habría comenzado este largo proceso. Siempre han demostrado una enorme paciencia y comprensión, además de entender casi antes que yo mismo que debía completar este proceso que, aunque lleno de altibajos, iba a disfrutar profundamente.

Resumen

En la actualidad, los avances científicos y la reducción de costes en la tecnología han provocado la proliferación de vehículos no tripulados de diversas características y naturaleza (aérea, terrestre, de superficie marítima, etc.), que han comenzado a utilizarse con multitud de propósitos tanto en el ámbito civil como en el militar. Además, algunos de estos vehículos están dotados de tecnología de alta precisión que les permite realizar de forma autónoma tareas cada vez más complejas, ya sea actuando en solitario o agrupándose en equipos de vehículos que combinan sus capacidades para completar un objetivo común. Para supervisar con facilidad este tipo de tareas o misiones es necesario contar con un software que sea capaz de monitorizar los vehículos y proporcione al operador las herramientas necesarias para intervenir si es necesario. Este software, conocido como Centro de Control de Tierra (CCT), debe contar con un mapa para situar los vehículos en el entorno, los elementos gráficos necesarios para mostrar por pantalla los datos de telemetría de los vehículos y los controladores que permitan comandarlos.

El diseño e implementación de un CCT es un proceso complejo que conlleva el cumplimiento de numerosos requisitos. Estos aumentan notablemente con el número de vehículos que se debe monitorizar simultáneamente, especialmente si el equipo supervisado está formado por vehículos heterogéneos, ya que cada vehículo puede contar con unos protocolos de comunicaciones, un controlador y unos elementos gráficos de visualización diferentes. Debido a estas dificultades, lo habitual es realizar el diseño e implementación de un centro de control ad-hoc

para un vehículo o equipo de vehículos fijo que sea capaz de desempeñar unas tareas concretas en un entorno determinado. Además, la cantidad de datos que el operador de un CCT ha de monitorizar aumenta con el número de vehículos que deben ser supervisados simultáneamente, aumentando así su carga mental de trabajo y su fatiga, factores que hacen descender su rendimiento, que tarde más en realizar sus tareas o incluso que desatienda alguna de ellas. Este problema suele obligar a involucrar más operadores en la misión, dividendo las tareas de monitorización y control entre varios de ellos para que la carga mental de trabajo se mantenga en niveles aceptables. Por estas razones, un CCT debe ser también ergonómico de forma que el operador pueda realizar su trabajo fácil e intuitivamente. También es muy recomendable que incluya algún tipo de mecanismos que asistan al operador en su trabajo para ayudarle a mantener niveles estables de carga mental de trabajo, estrés y fatiga, y minimizar, en la medida de lo posible, los recursos humanos necesarios para supervisar la misión.

Por estos motivos, el objetivo principal de esta tesis consiste en acometer el problema que supone la implementación de un centro de control de tierra que pueda reconfigurarse en tiempo de ejecución para dar soporte a diferentes tipos de misiones que involucren equipos de vehículos heterogéneos variables, cambios en los objetivos de la misión o en el entorno en el que ésta debe desarrollarse y que, además, incorpore ayudas al operador que disminuyan su carga mental de trabajo.

Para ello, en primer lugar, se ha diseñado una arquitectura adaptativa dirigida a eventos que, combinando las propiedades del software auto-adaptativo y de las arquitecturas dirigidas a eventos y potenciándolas con algunos patrones como el Observador y el Modelo-Vista-Controlador, nos permita desarrollar aplicaciones adaptativas distribuidas modulares, capaces de dar soporte a comunicaciones asíncronas que provengan de diferentes sensores, vehículos autónomos u otros agentes inteligentes involucrados en una misión compleja. Utilizando esta arquitectura, se ha implementado un centro de control de tierra adaptativo ca-

paz de reconfigurarse en tiempo de ejecución para permitir cambios en el equipo de vehículos supervisado y que el sistema reaccione a cambios inesperados en los objetivos de la misión o del entorno en el que esta se está desarrollando. Además permite incluir o extraer nuevos centros de control distribuidos en tiempo de ejecución para involucrar a nuevos operadores que se hagan cargo de las tareas de supervisión que no pueden ser asumidas por los operadores participantes en la misión.

En segundo lugar, utilizando la arquitectura adaptativa dirigida a eventos desarrollada, se ha implementado un interfaz gráfico adaptativo para el centro de control que es capaz de alterar en tiempo de ejecución tanto la ubicación como la cantidad de elementos gráficos mostrados. Por una parte, esto le permite auto-adaptarse a los cambios producidos en la infraestructura del centro de control (mostrar los datos referentes a un nuevo vehículo que se involucra en la misión o eliminar aquellos que pertenecen a uno que acaba de ser extraído de la misma). Por otra, las propiedades auto-adaptativas del interfaz gráfico nos han permitido integrar dos mecanismos de ayuda al operador: 1) adaptabilidad, que consiste en cambiar de posición los elementos gráficos en la pantalla para situarlos, por orden de prioridad, en los lugares de mejor visibilidad, y 2) transparencia, que consiste en ocultar algunos elementos gráficos en aquellos instantes de la misión en los que no sean necesarios. La combinación de estas dos técnicas de ayuda permite que el operador se centre en una menor cantidad de elementos gráficos (los más importantes en cada instante) y que los localice con mayor facilidad, por lo que contribuye a disminuir su carga mental de trabajo y, por consiguiente, ayuda a minimizar, en la medida de lo posible, la cantidad de operadores necesarios para completar la misión.

Por último, se propone la utilización del centro de control de tierra adaptativo como banco de pruebas de simulación en diferentes ámbitos, ya que su naturaleza desacoplada permite cambiar, fácilmente y sin necesidad de rediseñar su estructura, algunos de los módulos que dictan su comportamiento. Esto, sumado a su

capacidad de interactuar con modelos simulados de vehículos autónomos, hace que sea una plataforma de pruebas ideal para todo tipo de experimentos, como los que han sido realizados en esta tesis para verificar la eficacia de las ayudas al operador implementadas en el interfaz gráfico del CCT. Finalmente cabe indicar que también puede ser utilizado como plataforma de entrenamiento para paliar los efectos negativos de la curva de aprendizaje de operadores inexpertos en supervisión de misiones complejas con equipos de vehículos autónomos heterogéneos.

Abstract

Nowadays, scientific research and reduction in technology costs have led to the proliferation of unmanned vehicles of different nature (air, land, sea surface, etc.), which have begun to be used for many purposes both in the civil and military scopes. In addition, some of these vehicles are equipped with high precision technology that allows them to autonomously perform increasingly complex tasks, either acting alone or grouping into teams that combine their capabilities to complete a common goal. To easily supervise these tasks or missions, it is necessary to have a piece of software capable of monitoring the vehicles and of providing the operator with the necessary tools totake it over if necessary. This software, known as Ground Control Station (GCS), must have a map to place the vehicles in the environment, the graphic elements needed to display vehicles'telemetry data on the screen, and the controllers that allow operators to command them.

The design and implementation of a GCS is a complex process that entails the fulfillment of numerous requirements. These requirements increase noticeably with the number of vehicles that must be monitored simultaneously, especially if the supervised team is made up of heterogeneous vehicles, since each vehicle can have different communications protocols, controller and visual display elements. Due to these difficulties, it is usual to design and implement ad-hoc GCSs for fixed vehicles or set of vehicles that are capable of performing specific tasks in a particular environment. In addition, the amount of data that the operator of a GCS has to monitor increases with the number of vehicles that

must be monitored simultaneously, incrementing their mental workload and fatigue, factors that decrease their performance and make them to take longer to perform his/her tasks (or to neglect any of them). This problem usually forces to involve more operators in the mission, dividing the tasks of monitoring and control among several operators so that the mental workload of each of them is kept at an acceptable level. For these reasons, a GCS must also be ergonomic to let the operator perform his work easily and intuitively. It is also highly recommended to include some mechanisms that assist the operators in their tasks to help them maintain stable levels of mental workload, stress and fatigue, and to minimize, when possible, the human resources necessary to supervise the mission.

For these reasons, the main objective of this thesis is to tackle the problem of implementing a ground control station that can be reconfigured at runtime to support different types of missions (involving variable sets of heterogeneous vehicles and changes in the objectives of the mission or in the environment) and that, in addition, can incorporate aid mechanisms to reduce operators mental workload.

To achieve this objective, we have designed an adaptive event-driven architecture that, combining the properties of the auto-adaptive software and the event-driven architectures (as well as enhancing them with some patterns such as the Observer and the Model-View-Controller) allows us to develop modular distributed adaptive applications that are capable of supporting asynchronous communications (coming from different sources like sensors, autonomous vehicles or other intelligent agents involved in complex missions). Using this architecture we have implemented an adaptive ground control station, capable of reconfiguring itself at runtime to allow changes in the supervised set of vehicles, in the objectives of the mission and in the environment. It also allows to include or withdraw, at runtime, new distributed GCSs in order to involve new operators for assisting the ones that are already participating in the mission with the su-

pervision of the tasks that can not be assumed by them.

In addition, using the adaptive event-driven architecture, we have implemented an adaptive Graphic User Interface (GUI) for the GCS. This GUI is capable of changing at runtime both the location and the number of shown graphic elements. On the one hand, this allows it to adapt itself to the changes produced in the infrastructure of the GCS (show the data of a new vehicle involved in the mission or eliminate those that belong to one that has just been witdrawn from it). On the other hand, the self-adaptive properties of the GUI have allowed us to integrate two mechanisms to assist the operator: 1) adaptability, which consists in changing the position of the graphic elements on the screen to place them, in order of priority, in the places of better visibility, and 2) transparency, which consists of hiding some graphic elements in those moments of the mission where they are not necessary. The combination of these two assisting mechanisms allows the operator to focus on less graphic elements (the most important at each moment) and to locate them more easily, which helps to reduce their mental workload and, therefore, to minimize, as far as possible, the number of operators needed to complete the mission.

Finally, we propose the use of the adaptive ground control station as a simulation test bench in different scopes, since its decoupled nature allows to change, easily and without the need to redesign its structure, some of the modules that model its behavior. This, combined with its ability to interact with simulated autonomous vehicles, makes it an ideal testing platform for all types of experiments, such as those that have been carried out in this thesis to verify the effectiveness of the operator aid mechanisms implemented in the GUI of the ground control station. Finally it isworth highlighting that it can also be used as a training platform in order to mitigate the negative effects of the learning curve of inexperienced operators involved in the supervision of complex missions with heterogeneous teams of autonomous vehicles.

Índice

Agradecimientos IX

Resumen XI

Abstract XV

1. Introducción 1

1.1. Motivación 2

1.2. Objetivos 5

1.3. Aproximación Propuesta 7

1.4. Principales Contribuciones de la Tesis 9

1.5. Estructura de la Tesis 12

2. Estado del Arte 15

2.1. Vehículos Autónomos 17

2.2. Centros de Control de Tierra 20

2.2.1. Cantidad y Tipo de Vehículos 23

2.2.2. Diseño e Implementación 28

2.2.3. Interfaz, Ergonomía y Ayudas al Operador 35

2.3. Marco Experimental 43

2.4. Conclusiones 47

3. Arquitectura Adaptativa Dirigida a Eventos 49

3.1. Arquitectura Dirigida a Eventos 51

3.1.1. Agentes: Generadores de Eventos 53
3.1.2. Canal: Comunicaciones y Gestión de Eventos 58
3.1.3. Sumideros: Consumidores de Eventos 61
3.2. Marco Experimental: Experimentos de Validación 63
3.2.1. Experimentos de Comunicaciones y Caída de Red 64
3.2.2. Experimento de Integración de Vehículos en Tiempo de Ejecución 73
3.2.3. Experimentos de Integración de CCTs en Tiempo de Ejecución 84
3.3. Caso de Uso: Adaptación del Software a Nivel Estructural 93
3.4. Conclusiones 101

4. Interfaz de Usuario: Transparencia y Adaptabilidad 105
4.1. Motor Adaptativo Dinámico 108
4.2. Controlador 115
4.3. Vista Adaptativa 117
4.4. Base de Datos de Configuración 120
4.5. Marco Experimental: Experimentos de Validación 121
4.5.1. Diseño del Marco Experimental 4 123
4.5.2. Experimentos de Recepción y Muestra de Datos 124
4.5.3. Experimentos con los Mecanismos de Transparencia . . . 130
4.5.4. Experimento con los Mecanismos de Adaptabilidad 135
4.6. Caso de Uso: Adaptación del Software a Nivel Visual 139
4.7. Conclusiones 146

5. El CCT como Banco de Pruebas: Entorno Simulado... 149
5.1. Criterios y Requisitos de Validación 151
5.2. Diseño del Simulador: Marco Experimental 154
5.3. Diseño del Experimento 160
5.3.1. Población del Experimento 161
5.3.2. Diseño de Misiones Experimentales 163

5.4. Resultados 174
5.5. Conclusiones 179

6. Conclusiones y Trabajo Futuro **193**
6.1. Conclusiones Generales 193
6.1.1. Arquitectura Adaptativa e Implementación del CCT . . . 195
6.1.2. Interfaz de Usuario 197
6.1.3. Ayudas al Operador 198
6.2. Trabajo Futuro 200

I Apéndices **203**

A. Base de Datos de Configuración **205**

Bibliografía **211**

Índice de figuras

1.1. Vehículos Autónomos del Proyecto SALACOM y Centro de Control de Tierra 6

2.1. Diferentes Tipos de Vehiculos Autónomos 17
2.2. Centros de Control de Tierra Fijos 22
2.3. Centros de Control de Tierra Portátiles 23
2.4. Carga Mental de Trabajo frente a Rendimiento, de Waard (1996) 38
2.5. Marco Experimental 43

3.1. Flujo de la Generación y Consumo de un Evento 52
3.2. Arquitectura Adaptativa Dirigida a Eventos 54
3.3. Esquema de la Capa de Comunicaciones en un Agente y Diagrama de Aceptación de la Conexión 56
3.4. Flujo de Eventos en el Sistema 59
3.5. Configuración del Marco Experimental 1 65
3.6. Tiempos de Ejecución del Algoritmo de Reconexión en el Primer Experimento del Marco Experimental 1 70
3.7. Tiempos de Ejecución del Algoritmo de Reconexión en el Segundo Experimento del Marco Experimental 1 70
3.8. Configuración del Marco Experimental 2 73
3.9. Tiempos por Vehículo en una Iteración del Primer Experimento de Repetitivad del Marco Experimental 2 79

3.10. Tiempos de Inclusión y Extracción de 20 Vehículos por Cada una de las 50 Ejecuciones del Primer Experimento de Repetitivad del Marco Experimental 2 80
3.11. Tiempos de cada una de las 10 Etapas de Inclusión y Extracción de un Vehículo para las 3 Iteraciones del Experimento 82
3.12. Configuración del Marco Experimental 3 84
3.13. Tiempos por Visualizador en una Iteración del Primer Experimento del Marco Experimental 3 88
3.14. Tiempos de Integración y Extracción de los 10 Visualizadores en cada una de las 50 Iteraciones del Primer Experimento del Marco Experimental 3 89
3.15. Tiempos de cada una de las 10 Etapas de Inclusión y Extracción de un Visualizador para las 3 Iteraciones del Experimento 91
3.16. Configuración del Hardware del Experimento del Caso de Uso 1 . 95
3.17. Configuración del Experimento Software - Fase 1 96
3.18. Maniobra Lemniscata realizada por USV1 durante Fase 1 97
3.19. Configuración del Experimento Software - Fase 2 99
3.20. Maniobra Cooperativa Líder-Seguidor Realizada por USV1 y USV2 durante la Fase 2 100
3.21. Configuración del Experimento Software - Fase 3 (Delegación de Tareas) 101

4.1. Esquema Arquitectónico del Interfaz Gráfico Adaptativo 107
4.2. Bucle de Control Externo del Interfaz Gráfico Auto-Adaptativo . 108
4.3. Algoritmo de Cálculo del Umbral de Transparencia y Fragmento del Árbol de Decisión 112
4.4. Ejemplo de una Adaptación Ejecutada por el Controlador 117
4.5. Esquema de los Vectores de Prioridad para n Vehículos 119
4.6. Configuración del Marco Experimental 4 122
4.7. Gráficas con los Resultados del Experimento de Recepción y Muestra de Datos 129

4.8. Gráficas de los Resultados del Experimento de Cambio de Umbral 134
4.9. Gráficas Resultados Experimento de Rotación de Elementos . . . 138
4.10. Hardware del Experimento Real que Involucra a 2 USV y 1 UAV 141
4.11. Configuración Inicial del GUI de la Vista (Fase 1) 142
4.12. Configuración del GUI Antes del Despegue del UAV (Fase 2a) . . 144
4.13. Configuración del GUI Despues del Despegue del UAV (Fase 2b) 145

5.1. Marco Experimental del Banco de Pruebas para Ayudas al Operador 155
5.2. Menú de Inicio para Misiones Simuladas 157
5.3. Interfaz de Usuario Durante una Misión Experimental 161
5.4. Ventanas Asociadas a Eventos en una Misión Simulada 168
5.5. Cronograma de las Misiones Simuladas de Prueba para el Grupo de Referencia 172
5.6. Cronograma de las Misiones Simuladas Definitivas para el Grupo Principal 173
5.7. Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 1-4) 180
5.8. Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 5-8) 181
5.9. Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 9-12) 182
5.10. Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 13-16) 183
5.11. Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 17-20) 184
5.12. Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 21-24) 185
5.13. Gráfica de Tiempo Empleado Promedio sin Ayudas - Tiempo Empleado Promedio con Ayudas 186
5.14. Leyenda para Gráficas Carga de Trabajo (CdT), Tiempo de Reacción (TR) y Umbral de Transparencia (UT) 186

5.15. Gráficas de Carga de Trabajo, Tiempo de Reacción y Umbral de Transparencia (Operadores 1-8) 187
5.16. Gráficas de Carga de Trabajo, Tiempo de Reacción y Umbral de Transparencia (Operadores 9-16) 188
5.17. Gráficas de Carga de Trabajo, Tiempo de Reacción y Umbral de Transparencia (Operadores 17-24) 189
5.18. Gráfica de Resultados Obtenidos con el Test NASA-TLX en los Experimentos de Ayudas al Operador 191

A.1. Esquema de Definiciones de las Tablas de la Base de Datos de Configuración 209
A.2. Esquema Relacional de las Tablas de la Base de Datos de Configuración 209

Índice de Tablas

2.1. Tabla de Comparación de Características 30

3.1. Resultados del Marco Experimental 1 69
3.2. Resultados del Primer Experimento de Repetitividad del Marco Experimental 2 78
3.3. Resultados del Segundo Tipo de Experimentos de Repetitividad del Marco Experimental 2 82
3.4. Resultados del Primer Experimento del Marco Experimental 3 . . 87
3.5. Resultados del Segundo Experimento del Marco Experimental 3 . 91

4.1. Resultados del Experimento de Recepción y Muestra de Datos . . 126
4.2. Resultados de los Experimentos de Cambio de Umbral 131
4.3. Resultados del Experimento de Rotación de Elementos Gráficos . 136

5.1. Tabla de Resultados Obtenidos con el Test NASA-TLX en los Experimentos de Ayudas al Operador 190

Capítulo 1

Introducción

"Largo es el camino de la enseñanza por medio de teorías; breve y eficaz por medio de ejemplos".

Séneca

Este capítulo comienza describiendo la motivación de la tesis, cuyo objetivo es el desarrollo de un centro de control de tierra adaptativo para la monitorización y control de equipos variables de vehículos autónomos heterogéneos, que además implemente ayudas al operador. A continuación se describirán los objetivos principales de la tesis. Finalmente se destacarán las principales aportaciones de la tesis y se concluirá detallando la estructura de la misma.

Cabe destacar que esta tesis ha sido desarrollada dentro del proyecto de investigación SALACOM (Sistema Autónomo de Localización y Actuación ante contaminantes en el mar, DPI2013-46665-C2-1-R), financiado por el Programa Español de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, cuyo principal objetivo es la contención de vertidos contaminantes en entornos marinos llevado a cabo por equipos de vehículos autónomos. Por lo tanto el software desarrollado en esta tesis ha sido testeado tanto en entorno simulado como en experimentos de campo realizados con vehículos autónomos

en los pantanos de Valmayor y El Atazar durante las fases finales de desarrollo del proyecto.

1.1. Motivación

En la actualidad los avances tecnológicos han permitido el desarrollo de vehículos autónomos de diversa índole para su uso tanto en el ámbito civil como en el militar. Estos vehículos, cada vez más avanzados, son capaces de integrar gran cantidad de sensores y actuadores que les permiten realizar multitud de tareas como pueden ser la entrega de paquetes realizada por los drones de Amazon (2016), la grabación profesional de eventos realizada con diferentes tipos de vehículos aéreos con cámaras de alta resolución como el Inspire 2 de DJI (2019), la inspección marina realizada con vehículos marinos/submarinos como el WAM-V 8 de Marine Advanced Research (2019), la búsqueda y rescate por medio del vehículo multipropósito de Milrem Robotics (2019), la exploración espacial con los vehículos Oportunity y Spirit de la NASA (2003a,b) o tareas de reconocimiento militar realizado con drones como Predator y Reaper de General Atomics (1995-2017) o Atlante de Airbus (2013).

El sistema utilizado para el desarrollo de cualquier misión autónoma puede desglosarse en cinco elementos principales: 1) las tareas que deben realizarse que constituirán el objetivo de la misión, 2) el entorno en el que deberán realizarse dichas tareas, 3) los vehículos autónomos que las llevarán a cabo, 4) el factor humano, formado por el o los operadores que se encargarán de supervisar la misión, y 5) el software que se encargará de integrar al resto de actores implicados en la misión. Los 4 primeros elementos (objetivos, entorno, vehículos y operadores) interactúan para finalizar con éxito la misión y tienen influencia directa sobre el diseño del software, cuyo diseño limitarán con una serie de restricciones y requisitos propios de cada uno de ellos. Este software (junto con el hardware en el que se ejecuta) es conocido como Centro de Control de Tierra (CCT), y su funcionamiento y ergonomía constituyen un elemento clave para el desarrollo

correcto de la misión. A continuación pasaremos a hablar de cada uno de los elementos (y sus influencias sobre el resto) con más detalle.

Las tareas u objetivos a completar son el primer requisito que se debe analizar y, junto al entorno en el que se llevarán a cabo, marcarán la pauta para seleccionar los vehículos participantes (y por tanto, se incluirán en el posterior diseño del CCT). En base a las características y restricciones propias de cada vehículo, unos serán más adecuados que otros para el entorno u objetivos de la misión, pudiéndose distinguir entre vehículos aéreos no tripulados (UAV, Unmanned Aerial Vehicle), de superficie marítima (USV, Unmanned Surface Vehicle), de tierra (UGV, Unmanned Ground Vehicle) o submarinos (UUV, Unmanned Underwater Vehicle). Además, dependiendo de la complejidad y las características de la misión, ésta podrá ser llevada a cabo de forma independiente por un solo vehículo (como en el caso del vehículo de inspección submarina Girona 500 de Ribas et al. (2012) o el desarrollo de sistemas de navegación para el vehículo de superficie Springer de Sutton et al. (2011)), o deberá entrar en acción un equipo de vehículos que se coordinen para lograr realizar la misión, como en los casos de inspección o búsqueda y rescate marítimos de Murphy et al. (2008); Lindemuth et al. (2011), en los que es muy común que los vehículos de superficie colaboren con uno o varios aéreos que ayudarán a localizar los objetivos desde el aire. Estos últimos casos son similares al que nos ocupa en el proyecto SALACOM, en cuyo marco se ha desarrollado esta tesis y que consiste en la contención de vertidos contaminantes en entornos marinos mediante la colaboración de dos vehículos autónomos de superficie marítima que despliegan una red de contención en torno a un vertido que se localiza gracias a la perspectiva aérea que aporta un cuatrirrotor autónomo. Más concretamente esta tesis se centrará en el diseño e implementación de un CCT que ayude a conseguir los objetivos del proyecto SALACOM.

El factor humano es otro de los elementos más importantes implicados en el desarrollo de una misión autónoma ya que, aunque la mayor parte del tiempo sólo intervenga en labores de monitorización y/o supervisión, puede resultar imprescindible para definir algunos aspectos de la misión o incorporar su experiencia en

caso de emergencia. En estos casos se ha demostrado que introducir a un humano en el lazo de control mejora el experimento hasta en un 50 % (Cummings et al. (2012)). Por lo tanto, las necesidades del operador u operadores involucrados en la misión determinarán también en gran medida el diseño del CCT. A este respecto, es importante que el centro de control incluya las herramientas necesarias para que el humano sea capaz de interactuar con todos los integrantes del sistema y que sea capaz de aglutinar dichas herramientas en un solo interfaz gráfico. Además, este interfaz debe ser ergonómico, sencillo e intuitivo de tal forma que el usuario pueda mantener su nivel de atención mientras realiza su labor. Es, por tanto, el interfaz gráfico un elemento esencial en el diseño del CCT debido a que influirá notablemente en la eficiencia del operador, mitigando sus niveles de estrés y carga mental de trabajo si está bien diseñado o aumentándolos en caso contrario. Además se ha demostrado que la integración de equipos de vehículos autónomos heterogéneos no solo complica el diseño del CCT y de su interfaz, sino que hace que se multipliquen los elementos que deben monitorizarse y, por tanto, hace que la carga mental de trabajo del operador aumente exponencialmente (Perez et al. (2013a,b)). Aún más, esta sobrecarga puede generar la necesidad de repartir la carga de trabajo entre varios operadores, haciendo el CCT aún más complejo. Por esta razón, en esta tesis se propone también la implementación de un interfaz gráfico adaptativo para el CCT que implementará mecanismos de ayuda al operador para mitigar los efectos negativos de la sobrecarga de trabajo.

A pesar de las dificultades anteriormente mencionadas, hoy día existen una gran cantidad de CCTs para aplicaciones tanto militares como civiles como el CCT portable de UAS (2019), la estación de control de tierra universal de UniversalGCS (2019), o la de General-Atomics (2019), utilizadas para UAVs como el Predator e incluso algunas de codigo abierto como QGroundControl (2008). Sin embargo, a pesar de la cantidad de recursos empleados para su implementación, todas estas herramientas software siguen unas pautas de desarrollo clásicas que, además de no resolver algunos de los problemas planteados, llevan al diseño ad-hoc de centros de control para misiones, vehículos y entornos muy concretos. En estos casos, si alguno de los parámetros de la misión cambia y/o se debe

reconfigurar el equipo de vehículos, se debe recurrir a largos y costosos procesos de rediseño y reimplementación del CCT y, lo que es más, si alguno de estos cambios tuviese que hacerse durante la ejecución de una misión, los objetivos de la misma quedarían incompletos casi con total seguridad. Por lo tanto, crear un centro de control de tierra que sea capaz de reconfigurarse o Adaptarse en tiempo de ejecución a cambios tanto en el entorno como en los objetivos de la misión y el equipo de vehículos involucrados supone uno de los grandes retos actuales.

En conclusión, la convergencia de la gran cantidad de requisitos derivados del diseño de una misión autónoma que implique un equipo cooperativo de vehículos autónomos heterogéneos y uno o varios operadores, nada fáciles de resolver en determinadas ocasiones aún por separado, hacen que el diseño e implementación de un CCT sea bastante complejo y, en la mayoría de las ocasiones, específico para unas determinadas condiciones, entorno, vehículo/vehículos y tareas. Por estos motivos esta tesis acomete el diseño de una arquitectura adaptativa dirigida por eventos que podrá utilizarse tanto para la implementación final de un centro de control adaptativo en tiempo de ejecución (que le permitirá reconfigurarse ante cambios en la misión, entorno o equipo de vehículos) como para la integración en el mismo de un interfaz gráfico de usuario con mecanismos de ayuda al operador.

1.2. Objetivos

Esta tesis se desarrolla en el marco del proyecto SALACOM, cuyo objetivo consiste en lograr que dos vehículos autónomos de superficie marítima desplieguen una red de contención en torno a un vertido contaminante que habrá sido previamente localizado desde el aire con ayuda de un cuatrirrotor que despega de la plataforma montada en uno de los vehículos de superficie. En la Figura 1.1 podemos observar los principales elementos involucrados en el proyecto SALACOM. En la imagen situada a la izquierda, podemos observar los tres vehículos autónomos que colaborarán en la misión desplegados en el pantano de Valma-

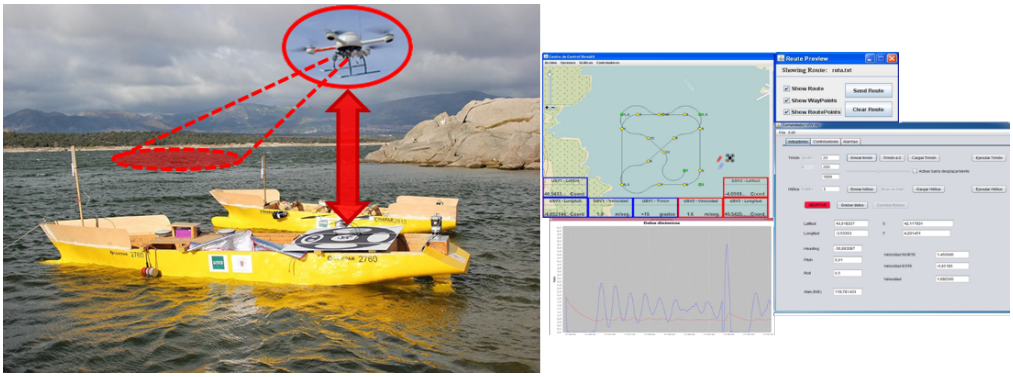


Figura 1.1: Vehículos Autónomos del Proyecto SALACOM y Centro de Control de Tierra

yor (donde se realizaron las primeras pruebas de campo). En amarillo sobre el agua se sitúan dos vehículos autónomos de superficie marítima (USV1, con la plataforma de despegue para el cuatrirrotor, y USV2). Estos dos vehículos serán los encargados de desplegar la red de contención en la zona indicada. Además sobre uno de ellos se sitúa el cuatrirrotor (UAV), que despegará para localizar el vertido contaminante, y enviarle a USV1 y USV2 las coordenadas donde se encuentra. En la parte derecha de la imagen, podemos observar una captura del centro de control adaptativo, que cuenta con un mapa en el que aparecen situados los vehículos y una ruta planificada con sus puntos de paso así como con un controlador/planificador y otras herramientas útiles para el operador como un visualizador gráfico de datos.

El objetivo general de esta tesis es el desarrollo de un centro de control de Tierra (CCT) adaptativo en tiempo de ejecución que integre las herramientas necesarias para que uno o varios operadores sean capaces de monitorizar, supervisar y/o controlar el equipo de vehículos, así como el resto de sensores que permitirán monitorizar ciertos aspectos de la misión y el entorno en el que se desarrolla. Además deberá ser capaz de adaptarse a cambios en el equipo de vehículos y objetivos de la misión así como de implementar ayudas al operador. A continuación, detallaremos los objetivos específicos de la tesis.

- Estudio del estado del arte en centros de control de tierra, diseño de software adaptativo, mecanismos de reducción de carga mental de trabajo en operadores y herramientas para medición de los mismos.
- Propuesta de una arquitectura software que permita el diseño e implementación de aplicaciones adaptativas distribuidas.
- Implementación de un centro de control capaz de adaptarse a cambios en la misión del proyecto SALACOM que puedan provocar la inclusión o extracción de elementos hardware o software en tiempo de ejecución (vehículos, sensores, visualizadores, etc.).
- Propuesta de técnicas capaces de reducir el estrés y la carga mental de trabajo de un operador del centro de control de tierra y diseño de la estructura de un interfaz gráfico de usuario para ponerlas en práctica en el centro de control
- Implementación de las técnicas de reducción de estrés y carga mental de trabajo en el interfaz gráfico del centro de control adaptativo del proyecto SALACOM
- Análisis de la utilidad de las contribuciones de la tesis tanto respecto al centro de control de tierra como a las técnicas de reducción de carga mental de trabajo del operador.

1.3. Aproximación Propuesta

Existen multitud de centros de control de tierra que dan solución al problema de monitorización y supervisión de misiones complejas llevadas a cabo por uno o varios vehículos. Sin embargo, estas soluciones clásicas, diseñadas ad-hoc para un conjunto de requisitos concreto (objetivos, vehículos, entorno, etc.), no son capaces de reaccionar satisfactoriamente a cambios en el conjunto de restricciones. Además, si la misión es compleja, las tareas del operador pueden hacerse farragosas debido a la gran cantidad de datos de los que éste debe hacerse cargo,

por lo que la solución habitual es repartir las tareas entre varios operadores (por ejemplo en misiones de reconocimiento con vehículos de altas prestaciones como el Predator de General Atomics (1995-2017)).

En esta tesis se aborda el problema que supone implementar un centro de control versátil capaz de adaptarse a cualquier equipo de vehículos heterogéneos, entorno y objetivos no solo mediante una fase de desarrollo previa, sino en tiempo de ejecución por si, debido a las circunstancias de la misión, alguno de los elementos debe reemplazarse. Propondremos una arquitectura software basada en algunos principios novedosos utilizados en el desarrollo de software auto-daptativo (Oreizy et al. (1999b); Garlan et al. (2004)) y auto-curativo (Dashofy et al. (2002)), así como en aplicaciones conectadas distribuidas (Arquitecturas Dirigidas por Eventos (EDA, Event Driven Architecture) Michelson (2006)) y algunos patrones bien conocidos como Modelo-Vista-Controlador (MVC) y Observador. Estos principios y técnicas, cuya virtud reside en que a pesar de su sencillez pueden combinarse fácilmente para potenciarse entre ellos, nos permitirán diseñar e implementar un CCT adaptativo en tiempo de ejecución que sea capaz de reconfigurarse ante cambios en la misión, entorno o incluso ante la sustitución, integración o sustracción de alguno de los vehículos autónomos involucrados.

Se propondrá también la implementación de técnicas de ayuda al usuario para tratar de paliar los efectos del exceso de su carga de trabajo mental para reducir su cansancio y el número de operadores implicados simultáneamente en una misión. Estas ayudas se incluirán en el CCT mediante la aplicación del esquema arquitectónico adaptativo al interfaz gráfico de usuario, lo que permitirá reconfigurar el interfaz durante la ejecución de la misión tanto para reubicar algunos elementos gráficos para mayor ergonomía como para aplicar la técnica de Transparencia (Mercado et al. (2016)), consistente en ocultar cierta información, considerada no relevante para la tarea en desarrollo, dejando más libre al operador para ocuparse de las tareas urgentes.

1.4. Principales Contribuciones de la Tesis

Esta sección resume las principales contribuciones y la metodología seguida durante la tesis.

Revisión bibliográfica. La investigación de esta tesis comenzó con un estudio de los centros de control de tierra existentes, haciendo especial hincapié aquellos que implementan alguna de las características deseables para nuestro proyecto. Además, se estudiaron las principales técnicas utilizadas para implementación de software adaptativo y técnicas de mitigación de carga mental de trabajo, así como su viabilidad de aplicación en un centro de control.

Centro de control de tierra adaptativo. El objetivo principal de esta tesis es el desarrollo de un centro de control adaptativo capaz de monitorizar/-controlar la misión llevada a cabo por cualquier equipo de vehículos heterogéneo, más concretamente para los involucrados el proyecto SALACOM (dos vehículos autónomos de superficie marina y un cuatrirrotor). El centro de control ha sido diseñado e implementado siguiendo las pautas marcadas por la arquitectura adaptativa dirigida por eventos, ha evolucionado a lo largo del desarrollo de esta tesis y en él se han incorporado progresivamente las técnicas y mecanismos detallados en los siguientes puntos de esta sección y en las consiguientes publicaciones.

Una primera versión de este centro de control fue presentada en una publicación en el congreso nacional:

- Juan A. Bonache Seco, Jose A. López Orozco, Eva Besada Portas, Jesús M. de la Cruz, *Centro de control versátil para equipos de vehículos heterogéneos: estado actual y mejoras futuras*,Actas XXXVII Jornadas de Automática, Madrid 2016.

Arquitectura adaptativa dirigida por eventos. En esta tesis se presenta un nuevo enfoque para el diseño e implementación de centros de control de

tierra adaptativos capaces de monitorizar y supervisar/controlar cualquier tipo de misión llevada a cabo por uno o varios vehículos autónomos heterogéneos sin necesidad de realizar una planificación y diseño ad-hoc para esa misión concreta. Además es capaz de reconfigurarse ante cambios en la misión, el entorno o el equipo de vehículos involucrado. Para ello se propone una arquitectura basada en software auto-adaptativo, capaz de reconfigurarse en tiempo de ejecución, y en arquitecturas software dirigidas por eventos, cuya naturaleza desacoplada facilitará el intercambio de módulos software/hardware pertenecientes a la infraestructura del centro de control.

Esta nueva arquitectura fue presentada por primera vez en una publicación en la revista:

- Juan A. Bonache Seco, Javier Dormido Canto, Martín Montalvo Martinez, Jose A. López Orozco, Eva Besada Portas, Jesús M. de la Cruz, *Centro de control de tierra para colaboración de vehículos autónomos marinos*, Revista Iberoamericana de Automática e Informática Industrial 2017 (Vol. 15, Num. 1).

Más adelante, la continuación del trabajo realizado sobre la arquitectura adaptativa volvió a presentarse para que su evolución y mejoras fuesen discutidas en profundidad en una publicación en un congreso internacional:

- Juan A. Bonache Seco, Jose A. López Orozco, Eva Besada Portas, José L. Risco Martín, *Adaptive event driven framework for real-time multi-agent missions*, Proceedings 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Madrid 2018.

Técnicas de ayuda al operador. En la tesis también se proponen dos técnicas de ayuda al operador que tienen como objetivo paliar los efectos adversos del exceso de carga mental de trabajo (como por ejemplo el aumento del tiempo de reacción, la distracción o la desatención a algunas tareas). La primera técnica se logrará mediante adaptabilidad, que recolocará los elementos gráficos en tiempo de ejecución para mejorar la ergonomía del CCT. La segunda,

transparencia, ocultará parte de la información durante determinados instantes en los que no sea relevante para el desarrollo de la misión para que el operador tenga menos carga de trabajo. Para ello se propone un novedoso interfaz gráfico de usuario diseñado también siguiendo las pautas de la arquitectura adaptativa dirigida por eventos. Esta estructura software dotará al interfaz de una infraestructura adaptativa cuyos módulos pueden sustituirse fácilmente, lo que lo hará una plataforma perfecta para la implementación de dichas técnicas.

Este nuevo enfoque de ayudas al operador para centros de control fue presentado en una publicación en el congreso nacional:

- Juan A. Bonache Seco, Jose A. López Orozco, Eva Besada Portas, Jesús M. de la Cruz, *Infraestructura para estudiar adaptabilidad y transparencia en el centro de control versátil*, Actas XXXVIII Jornadas de automática, Gijón 2017.

Los avances realizados con respecto a estas técnicas de ayuda fueron presentados más extensamente y discutidos en profundidad en una publicación en un congreso internacional:

- Juan A. Bonache Seco, Jose A. López Orozco, Eva Besada Portas, José L. Risco Martín, *ART-GCS: an adaptive real-time multi-agent ground control station*, Proceedings Spring Simulation Conference, Track on Complex, Intelligent, Adaptive and Autonomous Systems (CIAAS), Tucson 2019.

Por último, los experimentos diseñados para evaluar la eficacia de las ayudas al operador y los resultados obtenidos, se han presentado en un congreso nacional:

- Juan A. Bonache Seco, Jose A. López Orozco, Eva Besada Portas, Juan F. Jiménez-Castellanos, José M. Girón-Sierra, *Reducción de la carga de trabajo del operador en un centro de control adaptativo multi-vehículo*, Actas XL Jornadas de automática, Ferrol 2019.

1.5. Estructura de la Tesis

Capítulo 1: Introducción. Se plantea el problema del desarrollo de un centro de control para equipos de vehículos autónomos heterogéneos, se detallan los principales objetivos de la tesis y se resume la aproximación propuesta para resolver cada uno de ellos.

Capítulo 2: Estado del Arte. En este capítulo se realizará una revisión del estado del arte de los centros de control de tierra actuales y se compararán con el que propone en esta tesis. También se revisará la literatura sobre software adaptativo y arquitecturas orientadas por eventos. A continuación se analizarán algunas técnicas de ayuda para mitigar la carga mental de trabajo del operador y cómo cuantificarla. Finalmente se introducirán los marcos experimentales para realización de pruebas simuladas de validación de software.

Capítulo 3: Arquitectura Adaptativa Dirigida a Eventos. Se comenzará con una descripción de la arquitectura software propuesta para el diseño de aplicaciones adaptativas. A continuación se describirá cada uno de sus módulos principales así como su función dentro del esquema general de la arquitectura. Posteriormente se detallará el marco experimental para la realización de experimentos simulados para verificar la robustez y eficacia del software cuando es sometido a situaciones críticas y se discutirán los resultados obtenidos. Por último, se analizará el funcionamiento del CCT adaptativo durante un experimento de campo en el que se ha puesto a prueba la infraestructura modular del software así como el flujo de datos entre cada uno de estos módulos.

Capítulo 4: Interfaz de Usuario: Transparencia y Adaptabilidad. Este capítulo comenzará describiendo en qué consisten estas dos técnicas de ayuda al operador. Posteriormente se detallará el esquema arquitectónico diseñado para la implementación del interfaz gráfico de usuario del CCT, en el que se han integrado dichas ayudas. A continuación se presentará el di-

seño del marco experimental utilizado para la realización de pruebas de verificación de robustez y eficacia del interfaz gráfico cuando es sometido a situaciones críticas y se detallarán los resultados obtenidos. Finalmente se procederá al análisis de una misión de campo en la que se pondrá a prueba el funcionamiento del CCT y, especialmente, de su interfaz gráfico y ayudas al operador.

Capítulo 5: El CCT como Banco de Pruebas. Se realiza una especificación la plataforma de simulación desarrollada para la realización de experimentos de verificación de la eficacia de las técnicas de ayuda al operador implementadas. A continuación se expondrá la composición del grupo de referencia y el grupo de principal sobre los que se han realizado los experimentos. Posteriormente se detallará el diseño de cada uno de los experimentos y su propósito. Finalmente se expondrán y discutirán los resultados obtenidos.

Capítulo 6: Conclusiones y Trabajo Futuro. En este capítulo se resumirán los problemas resueltos durante el desarrollo de esta tesis, en la que se ha implementado un centro de control adaptativo para supervisión de equipos de vehículos autónomos heterogéneos, y se analizan las soluciones propuestas a cada uno de ellos así como las conclusiones obtenidas durante el proceso. Finalmente se expondrán algunas líneas de trabajo que podría ser interesante continuar en el futuro.

Capítulo 2

Estado del Arte

"La verdad es demasiado complicada como para permitir nada mas allá de meras aproximaciones".

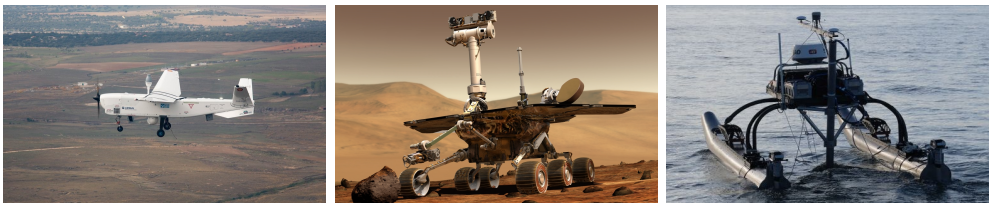
John von Neumann

En la actualidad, los avances tecnológicos permiten desarrollar componentes electrónicos, sensores y computadoras cada vez más pequeños, potentes y de bajo consumo que pueden incorporarse casi a cualquier dispositivo portátil o vehículo. Además, gracias a la mejora en las comunicaciones y en su ancho de banda, estos dispositivos pueden permanecer conectados en todo momento y comunicarse entre ellos para sincronizarse o enviar los datos generados durante el transcurso de su actividad. Esta evolución ha sido muy significativa en campos como la robótica, más concretamente en el desarrollo de vehículos autónomos, donde se ha logrado la integración de numerosos sensores y actuadores de gran potencia y precisión en chasis cada vez más pequeños. La reducción de tamaño y precio en este tipo de vehículos ayuda también a una mayor difusión de los mismos, por lo que cada día cuentan con un espectro más amplio de funciones en la sociedad.

Con la proliferación de los vehículos autónomos para todo tipo de usos, ha aumentado también la necesidad de desarrollar centros de control que sean capa-

ces de gestionar las tareas realizadas por uno o varios vehículos simultáneamente y mostrarle al operador la información relevante de la misión de la forma más sencilla y cómoda posible así como asistirle en la realización de las tareas complejas que deben ser llevadas a cabo por los vehículos. Por estas razones, los diseñadores y desarrolladores se enfrentan constantemente al reto de implementar un centro de control de tierra que además de ser robusto y seguro, cuente con un interfaz de usuario ergonómico (intuitivo y cómodo tanto a nivel visual como operativo) y, lo que es más, buscan darle valor añadido implementando mecanismos que faciliten las tareas del operador durante el transcurso de la misión. Este tipo de ayudas buscan mejorar el rendimiento del operador y por ende el rendimiento general del sistema, pudiendo llegar a ser clave en algunos casos en que la situación exige una rápida respuesta (toma de decisiones críticas, envío de consignas ante una emergencia, etc.). Esta idea se desarrolla en el trabajo de Cummings et al. (2012), donde se realizan una serie de experimentos con un equipo heterogéneo de vehículos autónomos (dos UAVs y un USV) en los que introducen al humano en el bucle de control de tareas de diferentes características (búsqueda, identificación y seguimiento de objetivos). Durante estos experimentos, queda demostrado que incluir la colaboración ágil de un humano en aquellas tareas que requieren conocimiento experto o identificación visual mejora notablemente (hasta un 50 %) el rendimiento del sistema respecto a la realización de las mismas tareas de forma automatizada, tanto en términos de área recorrida en las que el humano ayuda a replanificar y elegir zonas de búsqueda como de objetivos identificados. Por otra parte, en otras tareas que requieren gran capacidad de cálculo, la automatización total arroja mejores resultados ya que un sistema informático posee una capacidad de cómputo mayor que la de un humano.

A continuación, teniendo en cuenta la gran variedad de vehículos autónomos y de CCTs que podemos encontrar en el mercado y pueden combinarse para dar solución a las tareas que forman parte de una misión compleja, analizaremos el contexto actual de cada uno de ellos. Se comenzará haciendo una breve revisión de los diferentes tipos de vehículos autónomos que existen en el mercado, mencio-



(a) ATLANTE (UAV) (b) SPIRIT (UGV) (c) WAM-V 16 (USV)

Figura 2.1: Diferentes Tipos de Vehiculos Autónomos

nando algunos ejemplos de cada uno, describiendo brevemente sus características y el ámbito en el que se utilizan. A continuación se realizará la misma labor con los centros de control de tierra para vehículos autónomos dividiéndolos en tres grupos en base a algunas de las características que los definen: 1) la cantidad y tipo de vehículos para los que han sido diseñados, 2) el tipo de diseño e implementación, y 3) las características de su interfaz gráfico. Por último, se finalizará el capítulo presentando el concepto de Marco Experimental (Experimental Frame, EF), una herramienta formal basada en simulación muy extendida en los experimentos de validación de software, que se utilizará en esta tesis para realizar experimentos que pongan a prueba las principales características del centro de control de tierra adaptativo implementado durante su realización.

2.1. Vehículos Autónomos

Un vehículo no tripulado es aquel que no lleva personas a bordo. Puede ser controlado/guiado remotamente o realizar misiones de forma autónoma, aunque en muchas ocasiones el control o guiado remoto puede alternarse con las funciones autónomas o incluso combinarse en lo que se conoce como "control supervisado", resultado de combinar las tomas de decisión internas del vehículo con las del operador remoto.

Este tipo de vehículo debe contar con una serie de sensores que le permitan recabar información del entorno y de su posicionamiento en el mismo, como un Sistema de Posicionamiento Global (GPS) para localizarlo en el mapa, una Uni-

dad de Medición Inercial (IMU) para determinar la posición relativa del vehículo respecto al entorno (inclinación, alabeo, cabeceo, etc.) u otros sistemas de medida como pueden ser radares, lídares o cámaras. La cantidad y tipo de estos sensores, así como de los actuadores que se monten en el vehículo para interactuar con el exterior, dependerán de la naturaleza del vehículo (de tierra, aéreo, de superficie, etc.) y de las tareas para las que haya sido diseñado, llevando sensores sencillos y de precisión moderada como en un dron de bajo coste diseñado para entretenimiento y otros mucho más precisos como los diseñados para misiones complejas civiles o militares.

Se puede clasificar los vehículos no tripulados en tres grandes grupos atendiendo a su naturaleza, aéreos, de tierra y marítimos (de superficie o submarinos). A continuación describiremos cada grupo en detalle.

En primer lugar encontramos los vehículos aéreos (Unmanned Aerial Vehicles, UAVs), seguramente los más utilizados en la actualidad. Este tipo de vehículos, también conocidos como drones, se ha popularizado en los últimos tiempos gracias a su utilización para entretenimiento, y pueden adquirirse multitud de marcas y modelos a precios muy competitivos en casi cualquier superficie comercial. También se ha extendido enormemente su uso profesional para grabación de eventos, con drones que pueden adaptarse a las necesidades de cada usuario configurándose con módulos extra como cámaras de mayor resolución, módulos de estabilización de imagen y otros. En este ámbito podemos destacar algunas marcas muy conocidas como DJI con los modelos Mavic, Phantom o Spark (DJI-Technology (2019)) o Parrot con sus modelos Bebop o Mambo (Parrot-SA (2019)). En el ámbito militar, podemos resaltar algunos ejemplos de drones de largo alcance cuya tecnología de alta precisión les permite realizar misiones en emplazamientos a gran distancia. Entre los más conocidos en este ámbito podemos encontrar algunos como los modelos MQ-1, más conocido como Predator, o su variante MQ-9 Reaper (o Predator B) de General Atomics (1995-2017) o el Avión Táctico de Largo Alcance No Tripulado Español (ATLANTE) de Airbus (2013), que se muestra en la Figura 2.1a.

En el segundo grupo, situamos a los vehículos terrestres (Unmanned Ground Vehicles, UGVs) que también son utilizados en multitud de tareas civiles como el vehículo multipropósito Milrem Multiscope de Milrem Robotics (2019), un vehículo oruga diseñado para desplegarse rápidamente incluso en terrenos difíciles y cuyo diseño modular permite añadir actuadores y accesorios para adaptarse a múltiples tareas como vigilancia, búsqueda y rescate, transporte, cosecha o fertilización de cultivos o apagado de incendios. Los UGVs pueden ser usados también para agricultura de precisión, como la flota de tractores autónomos utilizados por el grupo ISCAR Universidad Complutense de Madrid (2019) para identificar en tiempo real líneas de cultivo en maíz, localizar de forma precisa rodales de malas hierbas y proceder a su eliminación. También podemos encontrar diversos ejemplos de utilización de este tipo de vehículos en el ámbito militar, como el Small Unmanned Ground Vehicle (SUGV) 310 de iRobot (2019), con una inversión de decenas de millones de dólares por parte del ejército de EEUU para formar una flota de estos vehículos que son utilizados por sus soldados, ingenieros de combate y artificieros para obtener datos del entorno y mejorar su consciencia situacional (Situational Awareness, SA) en situaciones críticas. Otro de los grandes ejemplos de UGVs realizando misiones de forma autónoma a gran distancia es el caso de los vehículos ROVER (Remotely Operated Video Enhanced Receiver) utilizados por la NASA para explorar Marte, que se pueden ver en la Figura 2.1b y cuya información se detalla en Oportunity NASA (2003a) y Spirit NASA (2003b), y cuya misión principal consiste en el análisis de rocas y suelos que puedan contener pruebas de la presencia de agua en Marte.

Respecto a los vehículos marítimos podemos encontrar dos tipos: de superficie (Unmanned Surface Vehicles, USVs) o sumergibles (Unmanned Underwater Vehicles, UUVs). Entre los primeros podemos encontrar múltiples ejemplos, como los tres modelos del vehículo modular adaptativo WAM-V (Wave Adaptive Modular Vessel) de Marine Advanced Research (2019) (Figura 2.1c), utilizado en diferentes tipos de exploraciones marítimas como la localización de obstáculos o vertidos en canales y puertos para su limpieza, la localización de posibles explotaciones mineras o la colocación de cableado submarino. En el ámbito de

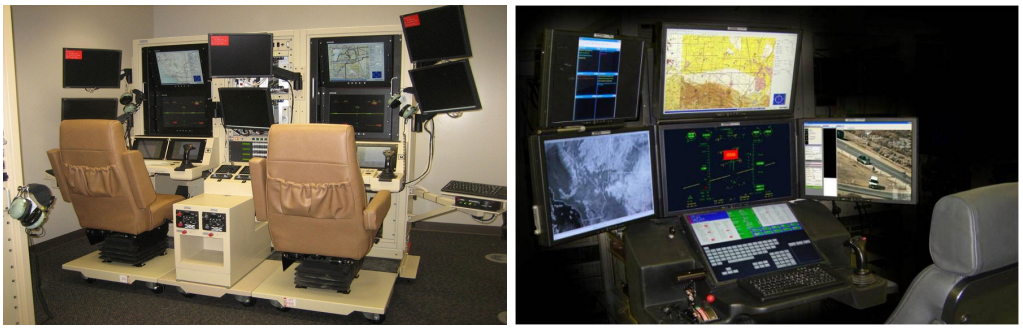
la exploración, también podemos encontrar el ESM30, de Oceanalpha (2019), un USV utilizado para localización de aguas contaminadas y que puede recorrer grandes distancias de forma autónoma recogiendo hasta 4 muestras diferentes de agua que pueden ser analizadas y sus datos enviados a la base en tiempo real. En este ámbito, podemos encontrar también los vehículos para contención de vertidos contaminantes utilizados en el proyecto SALACOM (Sistema Autónomo de Localización Y Actuación ante Contaminantes en el Mar) realizado por el grupo ISCAR–UCM (2019). Entre los segundos, los vehículos autónomos sumergibles, existen también múltiples ejemplos, como REMUS (Remote Environmental Monitoring UnitS) de Kongsberg-Maritime (2019), utilizado para tareas tan diversas como la arqueología marina, la medición de calidad del agua, o el reconocimiento y mapeado submarino, campo en el que también destaca el vehículo submarino Girona 500 de Ribas et al. (2012). También podemos encontrar algunos otros ejemplos como los modelos UUV CInspector o ROV Seaeye Falcon 12158, utilizados por la Universidad Cantabria (2019) en múltiples proyectos como pueden ser la supervisión de infraestructuras en los puertos marítimos o el mantenimiento de estructuras y cableado submarinos.

2.2. Centros de Control de Tierra

Cuando hablamos de un Centro de Control de Tierra (CCT), nos referimos al conjunto de hardware y software que unidos permiten la monitorización y/o el control de vehículos autónomos. Además éste debe incluir al menos un interfaz de usuario que permita monitorizar la información de telemetría (posición del vehículo en un mapa, ángulos de cabeceo, alabeo y guiñada, batería, velocidad, etc.), un equipo de comunicaciones que le permita conectarse a los vehículos de forma remota y los elementos necesarios para enviar consignas de control. Finalmente, dependiendo de la complejidad de la misión para la que haya sido diseñado y del vehículo a monitorizar, puede incluir elementos adicionales que permitan visualizar la información de los sensores y actuadores de los vehículos.

Debido a la enorme proliferación de los vehículos aéreos no tripulados y a su naturaleza remota, se ha hecho necesario el uso de interfaces de comunicación que han desembocado en el desarrollo de estándares de comunicación entre los vehículos aéreos y sus CCTs, como son el STANAG (STANdard AGreement o acuerdo de standardización) 4586 de la OTAN (2012) o MAVLink (Micro Air Vehicle Link) de Meier (2009). Aunque esto permite cierta flexibilidad en la utilización de los CCTs cuando se trata de monitorización y control de UAVs, cuando se trata de desarrollar un CCT que incluya vehículos autónomos marinos de superficie, como en el caso que se contempla en esta tesis, su desarrollo se complica debido a la ausencia estándares. De hecho, en la actualidad, el comité F41 de sistemas de vehículos autónomos marinos de la organización de normas internacionales ASTM (2017) (American Society for Testing And Materials), una de las organizaciones internacionales de desarrollo de normas más grandes del mundo, tiene entre sus objetivos el desarrollo de un estándar o guía de comunicaciones para USVs pero hasta el momento sólo está disponible un estándar para vehículos submarinos. En consecuencia, a día de hoy para la mayoría de CCTs para USVs o que tratan de integrar un equipo de vehículos de naturaleza heterogénea que los incluya, la elaboración del software no es tan sencilla, ya que no existen protocolos ni standards de comunicaciones unificados que lo permitan. Esto deriva en la elaboración de software ad-hoc para el tipo específico de vehículos de superficie que están involucrados en una misión muy concreta.

Además, podemos distinguir entre dos grandes grupos de CCTs dependiendo de sus características y tamaño. En primer lugar encontramos los centros de control fijos o montados sobre vehículos, que se distinguen por contar con gran cantidad de hardware de precisión y por ser de un tamaño considerable. Este tipo de CCTs pueden desplazarse de un lugar a otro en un remolque o contenedor de camión o permanecer fijos en las instalaciones que hagan las veces de base. En este grupo podemos encontrar algunos de los CCTs más destacados, como los modelos Legacy o Storm, de General-Atomics (2019), que podemos observar en la Figura 2.2. El primero, a la izquierda, es un CCT fijo diseñado para pilotar remotamente aeronaves por medio de comunicaciones vía satélite. A la derecha



(a) General Atomics: Legacy GCS (b) General Atomics: Storm GCS

Figura 2.2: Centros de Control de Tierra Fijos

de la imagen observamos Storm, un CCT específicamente diseñado para potenciar su ergonomía, contando con múltiples pantallas reconfigurables y táctiles. En segundo lugar, se encuentran los CCTs portátiles, caracterizados por contar con un hardware de tamaño más pequeño que permite que sean transportados de un lugar a otro por una persona. Este tipo de centros de control tienen la portabilidad como principal característica y suelen integrarse en un maletín de protección para su transporte. Cuentan con su propio sistema de comunicaciones, batería, controles, etc., todos ellos integrados en un solo bloque. Algunas partes de este hardware pueden ser modificados por encargo para otorgar cierta flexibilidad en su diseño que les permitirá ser utilizados con diferentes vehículos o misiones. Podemos encontrar algunos ejemplos de este tipo de CCT como los modelos ALT de Alti-UAS (2019), Octopus ISR de Octopus-Systems (2019) o los modelos GCase I y II de Alpha-Unmanned-Systems (2019), que podemos ver en la Figura 2.3.

En las siguientes subsecciones, distinguiremos entre diferentes tipos CCTs, tanto en el ámbito militar como civil, además de algunos experimentales o en desarrollo, dividiéndolos en tres grupos en base a: 1) la cantidad y naturaleza de vehículos que deben monitorizar y/o controlar, 2) las técnicas aplicadas en su diseño e implementación y 3) el interfaz de usuario, ergonomía y ayudas al operador que incorporan.

(a) Alpha Unmanned Systems: GCase I (b) Alpha Unmanned Systems: GCase II

Figura 2.3: Centros de Control de Tierra Portátiles

2.2.1. Cantidad y Tipo de Vehículos

Una de las características más importantes de un CCT tiene que ver con los vehículos que debe monitorizar y/o controlar, ya que esto suele condicionar en gran medida su diseño debido a las dependencias que existen entre las características hardware y software de un vehículo autónomo, su naturaleza y las tareas para las que ha sido construido.

A modo de ejemplo, podemos encontrar misiones que son llevadas a cabo por un solo vehículo o por varios que realizan tareas independientes o formando un equipo colaborativo homogéneo o heterogéneo en el que varios vehículos de distinta naturaleza complementen sus características para lograr un objetivo común. En el grupo de misiones realizadas por un sólo vehículo, podemos encontrar ejemplos como el Girona 500, un vehículo submarino autónomo que puede ser adaptado a diferentes tipos de tareas cambiando su configuración. En su trabajo, Ribas et al. (2012), describen detalladamente las diferentes configuraciones del vehículo en las que se varía la disposición de sus elementos de flotabilidad, los sensores que porta o la disposición de sus propulsores para realizar diferentes tipos de misiones de mapeo del fondo marino y localización e intervención de

objetos por medio de un garfio o un brazo robótico con 7 grados de libertad. También en este primer grupo podemos encontrar ejemplos de vehículos que realizan misiones de experimentación como Springer, utilizado por Sutton et al. (2011), que proponen dos algoritmos de posicionamiento geográfico para vehículos autónomos, uno basado en fusion multisensor de datos (Fuzzy Multisensor Data Fusion, MSDF) y otro basado en estimación adaptativa de modelos múltiples (Multiple Model Adaptive Estimation, MMAE), determinando gracias a los experimentos realizados con este vehículo que MSDF es superior en términos de robustez y precisión en el rumbo a MMAE. Por último, podemos encontrar otros ejemplos de misiones que pueden ser realizadas en solitario por diferentes micro-vehículos de superficie marítima en el trabajo de Patterson et al. (2013), que nos ofrece una visión de misiones relacionadas con tareas de seguridad y rescate marítimo poniendo en valor el apoyo que estos vehículos pueden prestar en tareas de búsqueda, rescate y monitorización marítima (por ejemplo el micro-vehículo EMILY puede equiparse con salvavidas para tareas de rescate o con diferentes sensores para reconocimiento y toma de muestras). También muestran en este trabajo algunos usos potenciales que podrían implementarse en el futuro como el apoyo a los guardacostas mediante la implantación de vehículos autónomos que patrullen el litoral.

En el otro extremo, podemos encontrar ejemplos del uso de un gran número de vehículos (normalmente homogéneos) denominados enjambres. El ejemplo más claro es el uso de pequeños drones de ala rotatoria para la generación de imágenes luminosas en el cielo, como los 1218 drones desplegados simultáneamente por Intel (2018) en los juegos olímpicos de invierno de Pyegongchang y que batió el record Guinness, posteriormente superado por eHang (2018) con 1374 drones desplegados. En el caso de enjambres de drones de ala fija, normalmente orientados a aplicaciones militares, destaca el record de 119 vehículos desplegados por CETC (2017). Habitualmente, los CCTs utilizados para monitorizar grandes enjambres de vehículos muestran la información de una forma diferente a aquellos utilizados en misiones que pueden llevarse a cabo por un pequeño número de vehículos, centrándose en el rendimiento, la efectividad y eficiencia del

grupo mediante datos como el porcentaje completado de la misión, el porcentaje de área cubierta, el número de obstáculos encontrados, los recursos consumidos, los tiempos de ejecución, etc. Hussein et al. (2018), en lugar de mostrar datos individuales de cada vehículo (telemetría, estatus, etc.). Además, aunque aún no se ha determinado el tipo de interfaz y la información que resultaría ideal para el control e interacción de un humano con un enjambre con un gran número de vehículos, se ha llegado a la conclusión de que la interacción humana en el bucle de control mejora el rendimiento del sistema y lo hace aplicable a una mayor cantidad de ámbitos Hussein et al. (2018). Por lo tanto, a pesar de las diferencias existentes entre ambos tipos de centros de control y de que en esta tesis nos centraremos en equipos de vehículos cooperativos heterogéneos donde el número de vehículos es lo suficientemente pequeño para no ser considerado un enjambre, se considera que las técnicas desarrolladas en esta tesis podrían ser de aplicación en este ámbito, ya que podrían ayudar a mejorar el rendimiento del operador incluido en el bucle de control.

Podemos encontrar múltiples ejemplos de misiones realizadas por equipos de vehículos cooperativos. Entre ellos destacaremos, por similitud a los objetivos de esta tesis, algunos cuya misión involucra cooperación entre USVs y UAVs. En primer lugar podemos mencionar el trabajo de Murphy et al. (2008), en el que se despliega un equipo formado por un USV (modelo AEOS-1) que será apoyado desde el aire por un UAV (modelo iSENSYS T-Rex) para realizar labores de detección de daños en los muelles y embarcaderos así como ubicación de escombros sumergidos y la determinación de carriles seguros para la navegación marítima tras el huracán Wilma. Dicho trabajo trata del primer despliegue conocido de un equipo cooperativo USV-UAV no simulado tras un desastre de estas características y en él se abordan algunos temas de gran interés para este campo, como la problemática que supone maniobrar USVs cerca de la costa o el reto que supone la colaboración entre vehículos aereos y de superficie. Por otro lado se encuentra el trabajo de Lindemuth et al. (2011) en el que despliegan un equipo marsupial, definido por Murphy et al. (1999) como “una colección de robots móviles donde uno o más de ellos dependen de otro de forma física, al menos temporalmente,

para obtener directivas, comunicaciones, transporte o energía", formado por un USV (basado en el diseño del AEOS-1) y un USV (modelo iSENSYS IP3), que serán utilizados para labores de inspección del litoral con propósitos militares, medioambientales o en labores de respuesta ante desastres naturales.

En estos casos, un centro de control de tierra contará con una serie de características y restricciones que dependerán de la naturaleza del vehículo a monitorizar y éstas serán más complejas si se trata de varios vehículos. Si los vehículos son de naturaleza heterogénea, la dificultad aumenta, ya que esto implica cambios en componentes del CCT como pueden ser los diferentes indicadores gráficos para mostrar la telemetría, los controladores de los distintos vehículos y, sobre todo, protocolos de comunicaciones particulares para cada uno. Debido a estas restricciones, únicas en cada caso, normalmente el software de los CCTs acaba siendo diseñado ad-hoc para cada misión y el tratar de conseguir uno que sea capaz de adaptarse a circunstancias cambiantes en el entorno de desarrollo de la misión, a la propia misión o al equipo de vehículos involucrado es un reto abierto para ingenieros y desarrolladores. Por una parte, podemos encontrar CCTs especialmente diseñados para el manejo simultáneo de equipos de vehículos homogéneos y los más comunes suelen ser para múltiples UAVs, como el diseñado por Perez et al. (2013a), un CCT de arquitectura descentralizada basado en aplicaciones y librerias de código abierto que se utiliza para el manejo de un sistema de vigilancia formado por múltiples UAVs y que proporciona asistencia al operador para que sea capaz de monitorizar varios vehículos simultáneamente. También de código abierto y para uno o varios UAVs simultáneos, podemos encontrar QGroundControl (2008), un CCT muy utilizado que puede ser ejecutado en la gran mayoría de plataformas actuales del mercado (Windows, linux, OSX, iOS, Android) y que proporciona al operador las herramientas básicas necesarias (planificación de vuelo autónomo, posicionamiento sobre mapa, seguimiento del vuelo, navegación por waypoints, datos de telemetría, etc.) para controlar cualquier UAV que se comunique mediante el protocolo MAVLink mencionado anteriormente. También existen algunos CCTs especialmente diseñados para USVs como ASV (2019), un sistema propietario que, mediante la instalación de un núcleo hardwa-

re estandarizado, es capaz de dotar a los vehículos marinos de las herramientas necesarias para comunicarse con su aplicación y que de esta forma sean capaces de realizar misiones autónomas o ser controlados remotamente por un operador. También podemos encontrar WGSM (2019) de Liquid Robotics, otro software propietario que permite el desarrollo de aplicaciones personalizadas adaptadas a las necesidades de cada cliente (diferentes vehículos y misiones en cada caso).

Por otro lado, respecto a centros de control para múltiples vehículos heterogéneos, podemos encontrar algunos ejemplos, como el desarrollado por Heo et al. (2016), un CCT para múltiples entidades de combate autónomas (Unmanned Combat Entities, UCE) heterogéneas que permite a un solo operador monitorizar hasta 4 UCEs y que cuenta con un total de 6 pantallas que permitirán al operador alternar entre los diferentes vehículos y sus modos de ataque o reconocimiento por medio de gestos intuitivos sobre los controles que aparecen en las pantallas. Otro CCT para múltiples vehículos heterogéneos es el entorno ASMAC, diseñado por Mupparapu et al. (2004) y que busca la capacidad de poder integrar nuevos vehículos a través de un lenguaje de control común (Common Control Language, CCL) además de introducir ayudas y guiado para el operador en algunos de los aspectos que consideran clave (como un menú de asistencia para las labores de planificación y un interfaz de usuario que muestra los datos de forma amigable). En el ámbito de los sistemas de vigilancia está AMFIS de Bürkle et al. (2011), un CCT capaz de fusionar y presentar de forma ergonómica los datos de sensores heterogéneos (además de los incluidos en UAVs y UGVs, otros que pueden integrarse en el sistema) y que asiste al operador, cuya labor consideran crítica en los sistemas de vigilancia, detección y localización de vehículos o personas. También podemos encontrar el CCT para entornos marinos diseñado para el proyecto europeo Munin por Burmeister et al. (2014) y que busca mejorar la seguridad en los trayectos por las zonas cercanas a la costa de vehículos marítimos mediante un sistema de navegación automática mediante sensores que se complemente con un sistema de monitorización de operadores humanos desde la costa. Por último podemos mencionar el CCT inmersivo de Walter et al. (2004), que permite integrar diferentes tipos de vehículos de reconocimiento y combate e

incorpora al interfaz de usuario nuevas tecnologías como la realidad virtual para dotar al operador de una información más detallada y realista del entorno en el que se desarrolla la misión.

Como puede observarse, existen numerosos ejemplos de CCTs que permiten realizar misiones con las tres configuraciones más comunes: un sólo vehículo, varios vehículos del mismo tipo o varios vehículos heterogéneos. Pero en ninguno de los casos el software es lo suficientemente flexible como para adaptarse a un cambio significativo en el objetivo de la misión, el entorno o, especialmente, en los tipos de vehículos involucrados. En las siguientes líneas resumiremos las aproximaciones utilizadas en diferentes tipos de software experimentales y comerciales que usaremos para conseguir dotar al nuevo CCT que se desarrollará en esta tesis de las características necesarias para cumplir con estos requisitos.

2.2.2. Diseño e Implementación

El diseño e implementación de centros de control de tierra capaces de adaptarse a cambios en la misión, entorno o equipo de vehículos involucrados en tiempo de ejecución es un reto constante para ingenieros y desarrolladores. Además, debido al aumento en la complejidad de las misiones o tareas a realizar, el número y capacidades de los dispositivos o vehículos a monitorizar/controlar por un CCT crece constantemente, por lo que éste debe tener cada vez más capacidad de cómputo para ser capaz de sincronizar los vehículos, realizar la planificación de la misión, enviar las consignas necesarias y gestionar los datos que se recogen durante el transcurso de una misión. Por estas razones resulta útil aplicar nuevos paradigmas que resultan más adecuados a estos problemas, como la computación distribuida, que nos permite dividir las tareas en diferentes computadores de forma transparente para el usuario. De hecho, en los últimos años, gracias al estudio de nuevas técnicas y la aplicación de patrones software y diseños arquitectónicos modernos, ha sido posible implementar CCTs funcionales que cumplen con algunos de los requisitos necesarios para adaptarse en tiempo de ejecución a cambios en el entorno, misión o equipos de vehículos monitorizados.

A continuación comentaremos algunos de los enfoques más destacados que se han aplicado en el desarrollo software de nuestro CCT para dotarlo de las propiedades anteriormente mencionadas. Sus características más destacadas aparecerán en la tabla 2.1, mostrándose en la primera columna el trabajo al que se hace referencia. En las tres siguientes, respectivamente, se indica si da soporte a adaptabilidad en tiempo de ejecución (es decir, si permite cambiar los dispositivos conectados, o sus propios componentes software mientras se está ejecutando), si es reutilizable (si es facilmente adaptable a otro tipo de misión o tarea) y si es escalable (si el software puede crecer fácilmente). La quinta columna muestra si el softwaredesarrollado siguiendo cada enfoque puede usarse para aplicaciones con diferentes propósitos.

Tabla 2.1: Tabla de Comparación de Características

Trabajo	Adaptación T. Ejec.	Reutilizable	Escalable	Multi Propósito	Distribuido	Tiempo Real	Simulación T. Real	Vehículos Heterogéneos
Garlan et al. (2004); Cheng et al. (2006)	✓	✓	✓	✓	×	✓	×	✓
Oreizy et al. (1999a)	✓	✓	✓	✓	✓	✓	×	✓
Dashofy et al. (2002)	✓	✓	✓	✓	×	×	×	×
Jovanovic y Starcevic (2010, 2008)	×	✓	✓	×	✓	✓	×	×
Hong et al. (2005)	✓	✓	×	×	×	✓	×	×
Hallsteinsen et al. (2012)	✓	✓	✓	✓	✓	×	×	✓
Amoui et al. (2012)	✓	✓	✓	✓	×	×	×	×
Filipponi et al. (2010)	×	✓	✓	✓	✓	×	×	✓
Magee et al. (1995)	✓	✓	✓	×	✓	✓	×	×
Damianou et al. (2001)	✓	✓	✓	×	✓	✓	×	×

La sexta y séptima columnas indican si el CCT permite que las aplicaciones sean distribuidas y si pueden integrar hardware y aplicaciones de tiempo real. Finalmente, la octava indica si el trabajo permite la monitorización y control de dispositivos inteligentes de naturaleza heterogénea. A continuación discutiremos con más detalle los diferentes aspectos de dicha tabla.

Existen varios enfoques de entornos (frameworks) de desarrollo basados en diseño arquitectónico, como Garlan et al. (2004) y Cheng et al. (2006), que proponen la utilización de Rainbow, un entorno que permite diseñar aplicaciones auto-adaptativas basadas en una arquitectura (apoyándose también en un lenguaje formal enriquecido que permite definir cada módulo y las relaciones entre ellos) con un bucle externo de control (external control loop). Para ello se cuenta con un modulo software situado fuera del propio sistema que se encarga de monitorizar datos extraídos del mismo y realizar cambios sobre él (adaptarlo) cuando los valores de estos datos indican que cambiar su estructura resultaría beneficioso para su rendimiento. Por su parte, Oreizy et al. (1999a) utilizan un enfoque similar en el que se propone una estructura arquitectónica que cuenta, en este caso, con dos bucles externos de control que separan en dos bloques lo que en el caso de Garlan et al. se realizaba en uno solo. El primero, llamado Bucle de evolución ("Evolution Loop"), gestiona la adaptación del software a lo largo del tiempo y el segundo, el Bucle de Adaptación ("Adaptation Loop"), se encarga de los cambios reactivos que se producen cuando algún evento los activa. También podemos encontrar algunos ejemplos interesantes de software desarrollado en el campo de las Ciudades Inteligentes (Smart Cities), cuya gran virtud es la capacidad de crear sistemas complejos distribuidos que conectan, sincronizan y analizan los datos producidos por una elevada cantidad de dispositivos heterogéneos simultáneamente. Por ejemplo, el enfoque propuesto por Filipponi et al. (2010), utiliza la filosofía de las arquitecturas orientadas a eventos (Event Driven Architectures o EDAs) para implementar infraestructuras de módulos altamente desacoplados capaces de manejar grandes cantidades de conexiones y de intercambiar mensajes asíncronos, lo que puede resultar muy útil a la hora de diseñar un CCT. En este trabajo se desarrolla un sistema para espacios públicos

que permite la monitorización de sensores heterogéneos gracias a la implementación de un modulo intermedio (o middleware) que hará las veces de enlace entre cada uno de los sensores (que tienen unos protocolos diferentes debido a su heterogeneidad) y el sistema principal. Aprovechando la arquitectura dirigida a eventos, este módulo se encargará de recibir los eventos tal como se producen en los sensores (datos), procesarlos y enviarlos a los modulos consumidores. También podemos encontrar el framework de desarrollo diseñado por Hallsteinsen et al. (2012), que permite implementar sistemas de computación distribuida para conectar a clientes desde cualquier parte creando a un entorno "ubicuo" (por ejemplo una red interconectada para transporte público que permite a los clientes conectarse desde dispositivos móviles con hardware y sistemas operativos de cualquier tipo para comprobar si hay alguna incidencia en el servicio, comprar tickets, etc.). Para ello han creado un entorno de desarrollo que permite implementar aplicaciones que se adaptan a los cambios de contexto (información relevante sobre las preferencias del usuario o el entorno que puedan influir en el sistema). Estas aplicaciones incluirán un middleware que permite separar la lógica de negocio, la información del entorno y los mecanismos de adaptación y que implementa un bucle de control para computación automática conocido como MAPE (Monitorizar, Analizar, Planear, Ejecutar) definido por Kephart y Chess (2003).

Respecto a enfoques más próximos a nuestro campo de estudio, podemos encontrar algunos entornos de desarrollo específicos para implementar CCTs que tratan de alcanzar un software flexible, escalable y adaptable. Entre ellos podemos destacar algunos trabajos como los de Jovanovic y Starcevic (2008, 2010), que han desarrollado una arquitectura altamente reutilizable gracias a su estructura modular orientada a objetos, que es capaz de aislar los aspectos lógicos o estáticos de la aplicación de los propios a cada subsistema que se conecte (por ejemplo, el protocolo propio de cada vehículo autónomo monitorizado) y además puede integrar simultáneamente aplicaciones tiempo real y no tiempo real. Beneficiándose de las virtudes de los patrones software integrados en la arquitectura, han desarrollado un CCT adaptativo y reutilizable para controlar un

vehículo aéreo no tripulado, incorporando además algunas características interesantes como el renderizado 3D del terreno en el que se desarrolla la misión para mejorar la Consciencia Situacional (Situation Awareness), es decir, la representación mental que tiene el operador del terreno en el que se realiza la misión. Podemos encontrar otro enfoque similar en el trabajo de Hong et al. (2005), que han diseñado una arquitectura específicamente orientada a software en tiempo real basada en RT-Linux. En dicho trabajo, logran implementar un CCT que garantiza el cumplimiento de las tareas en tiempo real gracias a una arquitectura de 4 capas, una hardware, otra de ejecución, otra de servicios y una última con un interfaz de usuario remoto. Gracias a esta división en capas el CCT puede reconfigurarse para monitorizar un helicóptero de radio-control en diferentes tipos de misiones. Alternativamente, Amoui et al. (2012), proponen una aproximación mediante una arquitectura orientada al modelo (model-centric architecture) implementada mediante un entorno de desarrollo basado en grafos para adaptación en tiempo de ejecución (Graph-based Runtime Adaptation Framework, GRAF), especialmente diseñado para simplificar el proceso de adaptación software. Su mecanismo de adaptación consiste en redirigir el flujo de control a un intérprete que permite al sistema añadir nuevos módulos software o modificar los ya existentes durante su ejecución.

Otro enfoque bastante extendido se basa en los lenguajes de especificación formal, utilizados para describir un sistema a más alto nivel de lo que se puede hacer con un lenguaje de programación, es decir, están pensados más para definir o conceptualizar un sistema, que para implementarlo como sería el caso de un lenguaje de programación habitual. Al tratarse de un enfoque bastante alejado de nuestra propuesta, no profundizaremos mucho en su aplicación. No obstante, dentro de esta línea cabe destacar algunos trabajos como Darwin, de Magee et al. (1995), un lenguaje de vinculación que interconecta estructuras software (estáticas o dinámicas) o Ponder Policy de Damianou et al. (2001), una aproximación basada en roles y políticas activadas por eventos que suele utilizarse en seguridad informática para implementar cortafuegos (firewalls) y otro software de uso similar. También podemos destacar el trabajo de Dashofy et al. (2002),

que consiste en una arquitectura para implementar aplicaciones software que son capaces de auto-reparar dinámicamente algunos módulos de su infraestructura. Para ello definen la estructura de sus aplicaciones mediante xADL 2.0, un lenguaje de marcado basado en XML que les permite describir su sistema y los cambios que pueden ocurrir en él.

Además, resulta interesante mencionar algunos protocolos y sistemas de comunicación específicos para arquitecturas basadas en eventos, generalmente orientados a desarrollo web, que aunque se alejan del ámbito contemplado en esta sección (por lo que no han sido incluidos en la tabla comparativa), podrían aportar algunas características interesantes al sistema si se considera, por ejemplo, la transmisión de datos a módulos de nuestro software distribuidos en localizaciones remotas a través de internet. Entre ellos, podemos destacar algunos como WebSocket (2010), que proporciona un canal de comunicaciones bidireccional en un solo socket TCP y que ha sido diseñado para ser implementado en navegadores y servidores web aunque se puede utilizar en cualquier aplicación cliente servidor. También podemos encontrar WebHooks (2007), que proporciona un sistema de retrollamadas HTTP POST utilizado para notificaciones o actualizaciones en tiempo real, o Server-Sent-Events (2013), una tecnología implementada en la mayoría de navegadores web actuales y que provee al sistema de algunos mecanismos muy útiles como el de recuperación ante caídas de red (pérdidas de mensajes) y la posibilidad de identificación única de eventos. Aunque, como hemos mencionado anteriormente, estas tecnologías son de gran interés y podrían ser incorporadas al centro de control desarrollado en esta tesis en caso de considerar necesario un aumento de la seguridad debido, por ejemplo, a la difusión de datos por internet, en el caso que nos ocupa en la actualidad hemos elegido simplificar el sistema utilizado basándolo en el patrón Observador implementado sobre el sistema de Sockets TCP Java, lo que nos permitirá aligerar el flujo de comunicaciones con paquetes de un protocolo propio que son de un tamaño más pequeño, ya que estos protocolos incluyen numerosos sistemas de seguridad basados en capas que no resultan imprescindibles en el entorno de la red privada desplegada para nuestras misiones de campo.

Teniendo en cuenta las características mencionadas, cabe decir que en esta memoria propondremos una arquitectura que permite la implementación de un CCT adaptativo para que uno o varios operadores monitoricen y controlen un equipo de vehículos heterogéneos que puede cambiar durante el desarrollo de la misión. En él se combinarán algunas de las técnicas de los trabajos anteriormente mencionados, como la computación distribuida, la capacidad de integrar aplicaciones de tiempo real y no tiempo real, el bucle externo de control, la filosofía desacoplada de las arquitecturas orientadas a eventos o la utilización de patrones de desarrollo software habituales como Modelo-Vista-Controlador o Patron Observador. Combinando todas estas técnicas, a las que se podrían sumar los módulos necesarios para implementar alguno de los protocolos para arquitecturas basadas en eventos mencionadas si se desea añadir alguna de sus características al sistema de comunicaciones, se logrará desarrollar un CCT capaz de cumplir con todos los requisitos necesarios para llevar a cabo misiones complejas en entornos variables y peligrosos para el hardware (las misiones en el mar son especialmente arriesgadas ya que si hay algún accidente se corre el riesgo de estropear o perder el hardware permanentemente), además de poder cambiar durante la ejecución el equipo de vehículos autónomos e incluso la distribución de los módulos software del CCT si esto fuese necesario.

2.2.3. Interfaz, Ergonomía y Ayudas al Operador

El interfaz gráfico es uno de los elementos más importantes en cualquier aplicación o programa informático, ya que es el elemento con el que el usuario interactúa para realizar una tarea u obtener cierta información a través del dispositivo electrónico. Por lo tanto, un Interfaz Gráfico de Usuario (Graphical User Interface, GUI), debe ser *intuitivo* (para que el usuario identifique los componentes y la curva de aprendizaje sea lo más rápida posible), y *ergonómico* (para que esos componentes sean y estén colocados de la forma más cómoda posible para el usuario). También suele ser conveniente contar con una serie de mecanismos que ayuden al usuario u operador cuando su labor sea compleja, requiera una cantidad de cálculos mentales elevada o necesite tomar decisiones rápidamente.

Siendo el interfaz una parte tan importante en cualquier programa informático, lo es aún más cuando se trata de un CCT, ya que el GUI es la parte donde el operador visualiza el desarrollo de la misión y realiza la toma de decisiones en base a la información disponible.

Al igual que en el resto de componentes software que forman un centro de control de tierra, habitualmente el GUI está estrechamente vinculado con la misión, el vehículo o los vehículos involucrados en la misma, y las características del entorno donde ésta se desarrolla, siendo también diseñado en la mayoría de las ocasiones ad-hoc para estos tres bloques de variables junto con el resto del CCT. Entre los elementos gráficos esenciales para el GUI de un CCT hay que destacar un mapa del entorno donde se desarrolla la misión en el que se visualizará el vehículo o vehículos autónomos y que junto con algunos datos de telemetría (orientación, cabeceo, alabeo, guiñada, etc.) ayudará al operador a mejorar la consciencia situacional (representación mental del vehículo en el entorno), y a mejorar su tiempo de reacción en la toma de decisiones. También deberán mostrarse por pantalla otros datos relevantes para la misión como: posición de los actuadores del vehículo, objetivos, ruta a seguir, etc. Como podemos observar, se trata de una cantidad bastante grande de datos, que se multiplica por cada vehículo participante en la misión y crece con su complejidad, ya que serán necesarios más vehículos, sensores y actuadores para poder llevarla a cabo. Esta ingente cantidad de datos deberá ser monitorizada y controlada para la toma de decisiones, lo que incrementa la carga mental de trabajo (Mental WorkLoad o WL) y el estrés del operador, aumentando su fatiga y pudiendo producir negligencia por parte del operador respecto a alguna de las tareas. Por estas razones, es muy importante que el GUI esté optimizado, sea ergonómico y que cuente con ayudas al operador, ya que de lo contrario la misión monitorizada desde el centro de control podría verse alterada por algún error debido a la fatiga del operador.

Por lo tanto, evitar la fatiga del operador o usuario de aplicaciones informáticas es de capital importancia, ya que de ello depende el correcto desarrollo de su actividad y por consiguiente que la misión sea un éxito o no. Para ello resulta esencial medir correctamente la carga mental del trabajo del operador, ya que

esto nos indicará cómo esta desarrollando su trabajo, si las tareas que realiza son pocas (provocando distracciones por aburrimiento), muchas (provocando fatiga, estrés y llegando a sobrepasar su capacidad de trabajo) o se encuentran a un nivel adecuado. Esto, en definitiva, marcará la capacidad del operador para realizar más o menos tareas, y cómo de eficiente es nuestro GUI a la hora de ayudarle a trabajar.

Es importante resaltar que la carga mental de trabajo (Mental WorkLoad o WL) es definida por Rouse et al. (1993) como "la demanda puesta sobre los humanos". Esta definición puede resultar poco útil en nuestro contexto, así que debemos completarla aclarando que cuando *demanda* se refiere a la carga impuesta a una persona, "carga de trabajo experimentada" suele ser un mejor indicador. Además esa definición corta, sitúa toda la influencia de la carga de trabajo en una fuente externa, la tarea que exige esfuerzo, cuando está demostrado que depende además de las capacidades del individuo y otros factores como su motivación, estado de ánimo, cansancio previo, etc. En definitiva podemos tomar la carga mental de trabajo como la capacidad del individuo para realizar un trabajo, medida por ejemplo en un porcentaje, que va descendiendo según las diferentes tareas van demandando esfuerzo mental.

Existen diversos estudios sobre carga mental de trabajo realizados en diferentes ámbitos que proponen cómo modelarla, medirla y utilizar la información extraída. De entre ellos, cabe destacar aquellos realizados sobre pilotos de avión ya que, debido a la similitud de los controladores y elementos gráficos que deben monitorizar, resultan especialmente útiles para nuestro propósito. A este respecto podemos destacar el libro de de Waard (1996), un completo estudio en el que se modela la carga mental de trabajo definiendo las 4 regiones que podemos ver en la Figura 2.4: A) la zona donde el operador tiene un alto rendimiento, B) el espacio donde el rendimiento desciende cuando el operador tiene más demanda, C) la región donde el rendimiento está al mínimo debido a altas cargas de trabajo, y D) la zona donde la exigencia es mínima provocando descenso de atención en el operador. A su vez, la zona A es dividida en 3 regiones: A1) es la zona en la que, debido a la escasa demanda de trabajo o a la monotonía del

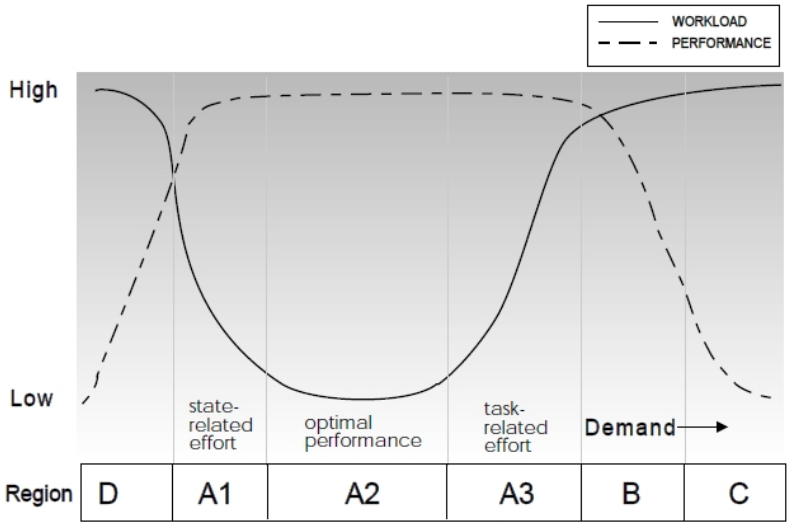


Figura 2.4: Carga Mental de Trabajo frente a Rendimiento, de Waard (1996)

mismo, el operador debe aumentar su esfuerzo si quiere mantener un rendimiento aceptable (sólo durante breves períodos de tiempo), A2) es la región donde el operador puede fácilmente con la carga de trabajo administrada (rendimiento óptimo), y A3) similar a A1, es la zona donde al aumentar la demanda el operador aún puede mantener un rendimiento aceptable durante breves periodos de tiempo pero aumentando su esfuerzo para completar las tareas. De Waard también hace una clasificación de las formas de cuantificar la carga de trabajo mental, que pueden dividirse en tres grandes ramas (las subjetivas tipo autoinforme, las de rendimiento y las psicológicas para las que suele ser necesario recurrir a sensorización). De este trabajo extraemos información muy útil para nuestra investigación (además del modelo mencionado anteriormente) como en qué características de una cuantificación para medir carga de trabajo hay que fijarse para saber si ésta es buena y funciona correctamente. Más concretamente, para una medición correcta de la carga mental de trabajo, se recomienda una medida subjetiva a posteriori, otra de carga mental de trabajo de tareas primarias y una tercera de carga de trabajo de tareas secundarias, estas dos últimas realizadas durante la ejecución de la misión.

Respecto a los cuantificadores subjetivos realizados a posteriori, podemos destacar algunos trabajos de interés que nos han permitido elegir el cuantificador de trabajo subjetivo que será utilizado en esta tesis, el índice de carga de tareas de la NASA (NASA Task Load IndeX, NASA-TLX). Por ejemplo, Rubio et al. (2004) relizan una comparación entre tres de los métodos de evaluación subjetiva de carga mental de trabajo más conocidos, el NASA-TLX, la técnica de asesoramiento de carga de trabajo subjetiva (Subjective Workload Assesment Technique, SWAT) y Perfil de carga de trabajo (Workload Profile, WP). Esta comparación se realiza en base a diferentes índices: intrusividad (cuanto influye el cuantificador en el experimento), sensibilidad (capacidad de deteccion de cambios), diagnóstico (capacidad de determinar la procedencia de un cambio) y selectividad (capacidad de reaccionar sólo ante los cambios que resulten relevantes). Los tres resultan igualmente inocuos en términos de intrusividad, ya que se realizan a posteriori. Además, WP resulta ligeramente superior en términos de sensibilidad mientras que NASA-TLX es superior en términos de diagnóstico y selectividad. Por último, dados los resultados obtenidos, ofrecen una recomendación de utilización para cada uno de estos cuantificadores subjetivos, proponiendo NASA-TLX como mejor cuantificador para determinar el rendimiento de un individuo en una tarea particular, como es el caso que nos ocupa. Por su parte, Hart y Staveland (1988) profundizan en los matices de cómo realizar un NASA-TLX correcto, y qué elementos tomar como medidores, y proponen una escala multidimensional en la que se combina información sobre la magnitud y fuentes de seis factores relacionados con la carga de trabajo y cómo combinarlos para realizar una estimación lo suficientemente sensible y fiable. Además, en un trabajo realizado 20 años despues, Hart (2006) analiza 550 estudios en los que se ha utilizado NASA-TLX probando que es una herramienta de un uso razonablemente sencillo para medir con suficiente sensibilidad y fiabilidad la carga de trabajo en campos más allá de aquel para el que fue diseñado, la aviación. Por último, también aporta información útil el manual de buenas prácticas para realización de estimación de carga mental de trabajo con el método NASA-TLX de Arquer y Nogareda (2000).

Existen otros tipos de cuantificadores de carga mental de trabajo, como los índices fisiológicos medidos mediante sensorización. Estos métodos son más difíciles de implementar debido a la necesidad de una mayor cantidad de recursos hardware (sensores y demás equipo necesario para monitorizar constantes vitales, parpadeos, gestos faciales, impulsos cerebrales, etc.) y a la mayor dificultad de tratar los datos obtenidos. Además, la obtención de este tipo de índices requiere de métodos físicamente más intrusivos para el operador, que tendrá que portar dichos sensores mientras realiza la misión, lo que puede aumentar su sensación de incomodidad y por tanto su estrés y fatiga mental. Estas razones, sumadas a los argumentos detallados en párrafos anteriores que indican que se puede realizar una estimación fiable y eficaz de la carga mental de trabajo mediante una medida subjetiva tomada a posteriori y dos medidas sobre cargas primarias y secundarias, nos han hecho decantarnos por esta segunda opción, mucho más directa y sencilla de implementar en el software de nuestro CCT.

Aunque no serán utilizados en este trabajo, cabe destacar algunos trabajos en el campo de los cuantificadores fisiológicos, como el realizado por DiNocera et al. (2007), que utiliza la distribución de zonas en la que se fijan los ojos de un piloto durante una simulación como medida indirecta de carga mental de trabajo para predecir el aumento o descenso en la misma durante el desarrollo de una misión simulada. De los resultados obtenidos durante la realización de las pruebas, se deduce que se produce mayor dispersión ocular durante las fases más estresantes de la misión (despegue, aterrizaje, etc.) por lo que se determina que esta medida puede ser utilizada como una cuantificación en tiempo real de la carga mental de trabajo del piloto y, por lo tanto, puede ser utilizada para activar mecanismos de adaptación automática. También podemos encontrar otros trabajos como el de Liao et al. (2005), que utilizan un sistema de medida no invasivo basado en reconocimiento visual de factores como el movimiento ocular, expresiones faciales o movimientos de la cabeza. Posteriormente utilizan una red bayesiana dinámica para modelar el estrés sufrido por el usuario. También podemos destacar el trabajo de Roscoe (1993), que realiza una distinción entre la carga de trabajo mental (más relacionada con el nerviosismo o estrés) de la

carga de trabajo físico. Este trabajo discute sobre la utilización de las medidas de respiración, pulso y variación del pulso en pilotos tanto durante vuelos reales como simulados.

Respecto a los cuantificadores que pueden tomarse durante la ejecución de una misión, existen varios tipos, de entre los que cabe destacar algunos de los detallados en el trabajo de Sebastián y delHoyo (2004). En este trabajo se mencionan algunos cuantificadores directos, como por ejemplo consultar al operador su carga mental de trabajo subjetiva en un instante determinado de la misión, y otros indirectos, como pueden ser la frecuencia de respuesta correcta, la tasa de error, o el tiempo de reacción en una determinada tarea. Estas recomendaciones han sido tenidas en cuenta a la hora de estimar la carga mental de trabajo de un operador mientras utiliza el CCT desarrollado en esta tesis, en el que además de la medición de carga mental de trabajo subjetiva realizada a posteriori mediante el test NASA-TLX, se han implementado en el interfaz los mecanismos necesarios para la medición de las dos medidas que se recomienda recoger durante la ejecución de una misión utilizando algunos de los índices mencionados anteriormente. En primer lugar se realiza la consulta de carga mental de trabajo subjetiva durante algunas tareas primarias críticas mediante la aparición de un formulario sencillo diseñado para mantener el nivel de intrusividad lo más bajo posible. En segundo lugar, se han incluido también formularios en los que el operador deberá realizar una tarea adicional secundaria en ciertos instantes de la misión. En ella se deberá realizar un sencillo cálculo mental sobre el que se medirá el tiempo de reacción, clave en este tipo de misiones y que será indicativo de la capacidad mental que el operador está empleando en las tareas primarias y secundarias.

Una vez definida la carga mental de trabajo, sus consecuencias en el operador y la forma en la que se medirá en el CCT implementado en esta tesis, nos centraremos en el diseño de los mecanismos de los que se debe dotar al GUI para ayudar al operador a que el indicador que mide la carga mental de trabajo no se dispare y pueda seguir el desarrollo de la misión el tiempo suficiente para acabarla con éxito.

Para ello nos basamos en algunos de los conceptos del software auto-adaptativo que se han detallado en la sección 2.2.2, por ejemplo en los trabajos de Garlan et al. (2004); Cheng et al. (2006); Cheng y Garlan (2007); Cheng et al. (2004). A continuación puntualizamos algunas de sus principales características en las que se inspirará nuestro trabajo (estructura modular, bucle externo de control, etc.) y que, aplicadas a los componentes del interfaz de usuario, nos permitirán alterarla dinámicamente para que se adapte a las necesidades ergonómicas derivadas del usuario (preferencias de visualización como lugares prioritarios en la pantalla, colores que visualiza con mayor facilidad, etc.), misión (cantidad y tipo de vehículos) y del entorno.

También contaremos con las ideas aplicadas en el campo de la transparencia. Aunque estudios como el de Buxton et al. (2000), analizan si el rendimiento de los usuarios de GUIs mejora al modificar la transparencia literal de sus componentes (es decir, alterando lo transparentes que éstos se vuelvan para dejar ver la información que se encuentra tras ellos), en este contexto tomaremos este concepto como la transparencia del programa hacia el operador, es decir, la cantidad de información que debe mostrarse para que éste sea capaz de realizar sus tareas sin que aumente la desconfianza en el software (si se oculta demasiada información, al no saber los razonamientos internos de toma de decisiones de los agentes inteligentes, el operador pierde confianza y duda) pero que no sea tan alta como para que quede abrumado por una cantidad excesiva que le impida realizar su trabajo. En este campo, Mercado et al. (2016) han desarrollado un estudio aplicado a un grupo heterogéneo de operadores para determinar su nivel de rendimiento, su nivel de carga de trabajo y el nivel de confianza en el programa dependiendo del nivel de transparencia utilizado para varias misiones distintas. En el experimento se propone a cada operador varias misiones sobre las que tiene que tomar decisiones, variando en cada caso el grado de transparencia, que se estructura en tres niveles. Estos tres niveles proporcionan cantidades distintas de información (a mayor transparencia mayor cantidad de información mostrada) con los que el operador tendrá que decidir si acepta el consejo propuesto por la aplicación o si cambia la estrategia a seguir para el desarrollo de la misión

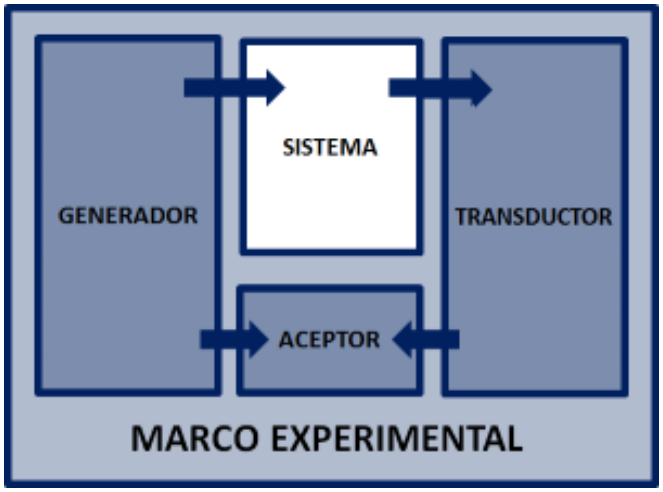


Figura 2.5: Marco Experimental

según su propio criterio. En base a los resultados de este experimento se puede extraer la conclusión de que, en contraste con lo expuesto por Helldin (2014), el rendimiento del operador, el nivel de confianza y la sensación de usabilidad de la aplicación aumentan con el nivel de transparencia sin repercutir en la carga de trabajo o el tiempo de respuesta. Esto se traduce en que, según aumentamos la cantidad de información proporcionada (de forma que el operador pueda deducir por qué el software aconseja un plan de ruta u otro para la misión y pueda razonar si éste es bueno o no), se aumenta el rendimiento del operador, como también se afirma en el trabajo de Chen et al. (2014). Es muy importante, por tanto, encontrar un compromiso que nos permita mostrar la información considerada relevante en cada instante de una misión pero sin que esta exceda la capacidad máxima que puede procesar el usuario, ya que así se conseguirá disminuir la carga mental de trabajo del operador sin perder su confianza en el software.

2.3. Marco Experimental

En las diferentes fases de desarrollo de una aplicación informática, ésta debe ser validada para comprobar que cumple con los requisitos para los que ha sido

diseñada y depurar posibles fallos. Esta tarea puede acometerse de diferentes formas, por ejemplo desarrollando la aplicación por completo antes de comenzar con la fase de pruebas (lo que podría hacernos perder mucho tiempo si algún fallo grave ha pasado desapercibido y la aplicación ha de ser rediseñada), o ir realizando tests sobre cada una de las partes de nuestra aplicación (una práctica mucho más recomendable siempre que sea posible). Respecto a las diferentes pruebas a las que hay que someter al software, podemos encontrar dos aproximaciones diferentes, someter el programa a estímulos de forma manual (disponiendo un grupo de personas que lo pruebe) o diseñar una serie de pruebas automatizadas que nos ayuden en esta fase de validación mediante una simulación. Además las simulaciones utilizadas para realizar experimentos son clasificadas por Hild (2000) en tres grupos: 1) emulación, utilizada cuando el principal objetivo es comprobar si el sistema funciona o no (en las últimas fases de desarrollo, el sistema estará modelado con todo detalle), 2) simulación cuántica, para probar cada una de las partes del sistema (en este caso algunas funcionalidades han de ser modeladas a alto nivel ya que aún no se conocen sus detalles), y 3) simulación dirigida, a medio camino entre las anteriores (algunas funciones modeladas en detalle, otras a alto nivel). En nuestro caso, combinaremos fases de pruebas simuladas (principalmente emulación) con pruebas realizadas por humanos tanto en entorno simulado como en experimentos de campo con vehículos reales.

Respecto a la metodología a seguir para realizar las pruebas simuladas, cabe destacar el concepto de marco experimental (Experimental Frame, EF), introducido por Zeigler et al. (2000), que es una especificación de un conjunto limitado de circunstancias bajo las cuales un sistema (ya sea un sistema real o un modelo del mismo) puede ser observado y sometido a experimentos. En otras palabras, un marco experimental nos ayuda a rodear al sistema (o modelo del sistema) con el que queramos realizar experimentos de un entorno simulado que nos permitirá probarlo en condiciones similares a la realidad. De esta forma, definiendo diferentes marcos experimentales seremos capaces de realizar diferentes pruebas sobre un sistema. Existen dos formas de ver los EFs, la primera consiste en tomarlos simplemente como la definición de los tipos de datos que interactuarán

con nuestro sistema. La segunda, más útil en el contexto que nos ocupa, consiste en considerar el marco experimental como un sistema que interactúa con el sistema sobre el que se quiere experimentar para obtener datos bajo unas determinadas circunstancias. En esta segunda aproximación, como puede observarse en la Figura 2.5, podremos dividir el EF en tres bloques: 1) el generador, que será el módulo encargado de estimular al sistema de la forma deseada (genera las señales o datos de entrada que se introducirán al sistema bajo estudio), 2) el transductor, que se encargará de recopilar y almacenar los datos de salida del sistema, y 3) el aceptor, que se encargará de monitorizar el sistema y validar los resultados (este módulo no siempre es necesario, dependerá de como se deseen interpretar los datos de salida). Este tipo de arquitectura cuenta con multitud de ventajas, ya que podemos cambiar cualquiera de los módulos obteniendo un nuevo conjunto de experimentos. Por ejemplo, si cambiásemos el generador podríamos cambiar los datos de entrada, sometiendo al sistema a unos estímulos diferentes para estudiar su reacción ante esos eventos concretos (o cambiar los datos de entrada para modelarlos en más detalle debido a algún cambio en la implementación del sistema). También podríamos cambiar la forma de almacenar los datos alterando el transductor o incluso realizar el mismo experimento varias veces para comprobar si se cumplen diferentes objetivos cambiando en cada ocasión el aceptor.

Podemos encontrar ejemplos de modelos de sistemas validados por medio de la creación de marcos experimentales definidos con diferentes herramientas software, como el de Veena (2005), en cuya tesis marca como objetivo averiguar cuantas llamadas pueden ser establecidas o rechazadas en una red de comunicaciones entre aeronaves y centros de control de tierra. Para ello contará con información sobre las rutas de las aeronaves y datos sobre el efecto de la ionosfera en las comunicaciones de radio de aeronaves que se encuentren en su zona de influencia. Para realizar los experimentos implementa dos modelos. El primer modelo simulará la red de comunicaciones entre aeronaves y centros de control de tierra que implementan el protocolo de establecimiento de enlaces automático. El segundo será el marco experimental, que proporcionará los datos de entrada

de las rutas que seguirán las aeronaves, las posibles interferencias sobre comunicaciones, cantidad de mensajes generados, etc. y se encargará de guardar los datos de salida. Estos modelos se generan con un programa específico para especificación de sistemas de eventos discretos (Discrete Event System Specification, DEVS).

Por su parte, Foures et al. (2013), proponen un método para validación de sistemas mediante simulación que consta de 4 fases en el que el marco experimental se aplicará en varias iteraciones sobre diferentes partes del software: 1) bucle del modelo, en el que se describirá el comportamiento del sistema que debe ser validado mediante simulación, 2) bucle del software, en el que se implementa el modelo del sistema descrito, 3) bucle del controlador, en el que se debe validar la equivalencia del comportamiento del sistema tras implementar los controladores, y 4) bucle hardware, en el que poco a poco se sustituirán los sistemas simulados por los reales. Además también aportan una metodología basada en un lenguaje de modelado para simulación (Simulation Modeling Language, SiML) que permitirá controlar mejor el desarrollo de la simulación y su validez.

Otra aproximación diferente a la implementación de marcos experimentales puede verse en el trabajo de Daum y Sargent (2001), en el que se presentan modelos desarrollados con el prototipo para modelado y simulacion de sistemas en Java (Hierarchical Modeling and Simulation System-Java, HiMASS-j), que utiliza el paradigma de modelos de grafos de control de flujo (Hierarchical Control Flow Graph Model, HCFG) que permite el modelado de sistemas mediante un interfaz visual interactivo con el que podrán crearse módulos sencillos (similares a autómatas) que se agruparán formando módulos más complejos hasta conseguir el comportamiento del sistema deseado.

En nuestro caso, hemos realizado el diseño de varios marcos experimentales que crearán un entorno simulado para probar la robustez y eficacia del CCT durante la realización de las tareas para las que ha sido diseñado. Estos marcos experimentales emularán las entradas al sistema durante una misión con varios vehículos heterogéneos, sometiendo a cada una de las partes del sistema a situa-

ciones críticas de forma sistemática y repetitiva, guardarán los datos generados y, en algunos casos, los validarán mediante un módulo aceptor.

2.4. Conclusiones

El desarrollo de centros de control de tierra es un reto constante debido a la dificultad que implica la monitorización y control de equipos de vehículos heterogéneos que cooperan para realizar una misión compleja y que, en la mayoría de ocasiones deriva en la implementación ad-hoc de centros de control específicos para una misión concreta realizada por un equipo de vehículos definido en un entorno conocido. Por estas razones, sería deseable conseguir desarrollar un centro de control que sea capaz de adaptarse a diferentes equipos de vehículos, nuevos objetivos en la misión o cambios en el entorno en tiempo de ejecución.

Para poner este problema en contexto, se ha comenzado el capítulo realizando un análisis de los vehículos autónomos que existen actualmente en el mercado basándonos en su naturaleza (de tierra, de superficie marítima, aéreos y submarinos) y sus diferentes configuraciones, que los harán compatibles con diferentes tipos de misiones y entornos.

A continuación, se ha llevado a cabo un análisis de los diferentes centros de control de tierra existentes, clasificándolos en grupos en base a sus características (fijos o portátiles) y su objetivo, del cual dependerán en gran medida las restricciones a la hora de implementarlo, ya que éstas vienen determinadas por la misión, el entorno y, sobre todo, el equipo (tipo y número) de vehículos que deba monitorizar.

Una vez puestos en contexto los centros de control, así como sus limitaciones, se ha realizado un análisis de algunas técnicas y marcos de desarrollo arquitectónicos utilizados en software auto-adaptativo, software para entornos distribuidos conectados y otras aplicaciones que, implementados en un centro de control, pueden hacer que un CCT cuente con algunas de las características deseadas (adaptable en tiempo de ejecución ante cambios en vehículos, objetivos o entorno).

Además, se ha realizado un análisis de la importancia que tiene el interfaz gráfico en un centro de control, ya que es una herramienta básica que debe ser ergonómica e intuitiva, además de contar con herramientas que ayuden al operador a mitigar los efectos adversos de un exceso de carga de trabajo, estrés y fatiga. Esto puede lograrse mediante la implementación de técnicas de ayuda al operador como adaptabilidad y transparencia.

También se ha realizado un repaso a la literatura sobre carga mental de trabajo y cómo medirla, estudiando diferentes métodos de cuantificación y cuales son más adecuados para ser integrados en un centro de control de tierra teniendo en cuenta su eficacia, sencillez de implementación y un nivel de intrusividad lo menor posible.

Para finalizar se ha introducido el concepto de marco experimental, una forma habitual de validación de software consistente en el desarrollo de un ecosistema en torno al programa sobre el que se quiere experimentar y que permitirá simular las entradas al sistema y guardar (y, en algunos casos, validar) las salidas producidas por el mismo.

De todo lo anterior se extraen las dificultades que conlleva el diseño e implementación de un centro de control capaz de adaptarse a diferentes equipos de vehículos autónomos, misiones o entorno en el que se realizan, así como de la importancia de un interfaz gráfico que asista al operador en la realización de sus tareas. En esta tesis se propone un marco de desarrollo arquitectónico, así como la implementación de un centro de control que será capaz de adaptarse en tiempo de ejecución a los cambios anteriormente mencionados mientras reduce la carga de trabajo del operador gracias a la incorporación de los mecanismos de ayuda integrados en su interfaz gráfico.

Capítulo 3

Arquitectura Adaptativa Dirigida a Eventos

"Es de importancia para quien desee alcanzar una certeza en su investigación el saber dudar a tiempo".

Aristóteles

En la nueva era del internet de las cosas (Internet of Things - IoT), los sistemas deben ser capaces de conectar simultáneamente una gran cantidad de dispositivos hardware diferentes. Esto es extrapolable a los vehículos autónomos, que progresivamente cuentan con más capacidades que les permiten resolver misiones cada día más complejas. Por estas razones, los centros de control de tierra son un elemento esencial para supervisar y controlar vehículos autónomos (además de otros sensores y agentes inteligentes) que realizan misiones complejas en tiempo real. Este tipo de misiones demanda una elevada potencia computacional para gestionar correctamente la sincronización de todos los dispositivos, el almacenamiento y tratamiento de datos generados, la toma de decisiones automatizada, el envío de consignas de control, etc. Además, debido a la complejidad creciente de las misiones y a que éstas pueden realizarse en un entorno hostil para el hardware de los dispositivos implicados (por ejemplo, en el entorno marítimo),

se hace necesario que el software sea capaz de integrar, además de dispositivos heterogéneos reales, otros simulados para poder comprobar el correcto desarrollo de la misión minimizando los riesgos de pérdida de hardware. Estos dispositivos adicionales son modelados y simulados en tiempo real utilizando complejas plataformas formadas a partir de la integración de diferentes herramientas y lenguajes de programación (Mittal y Risco-Martín (2017b)). Las soluciones tradicionales para la integración de simulaciones y aplicaciones en tiempo real que en el pasado han facilitado enormemente la implementación de este tipo de software, no cumplen con los requisitos de viabilidad, fiabilidad y escalabilidad necesarios en los sistemas complejos actuales (Mittal et al. (2009)). Además, para facilitar la implementación de plataformas y herramientas de desarrollo, las simulaciones distribuidas deben ser ligeras, fáciles de gestionar y orientadas a servicios, por lo que es necesario diseñar nuevos paradigmas de simulación que sean capaces de integrar estos requisitos en los nuevos estándares de computación distribuida (Mittal y Risco-Martín (2017a)).

Debido a la necesidad de cumplir todos estos requisitos, el desarrollo de un Centro de Control de Tierra (CCT) distribuido que soporte tal carga computacional y sea, además, robusto, reutilizable y adaptable en tiempo real conlleva numerosos retos, tanto desde el punto de vista de la ingeniería y el diseño de la arquitectura software como de la investigación y aplicación de nuevas técnicas y paradigmas de programación que permitan implementar este tipo de software.

En este capítulo se presenta una arquitectura adaptativa dirigida a eventos (Adaptive Event-Driven Architecture) específica para implementar centros de control de tierra (aunque sus conceptos son extrapolables a cualquier otro tipo de aplicación adaptativa), que permite superar los problemas derivados de la realización de misiones con equipos de vehículos autónomos heterogéneos reales y/o simulados, que pueden cambiar según las necesidades de la misión. Para resolver los problemas de heterogeneidad en el equipo de dispositivos conectados y de co-simulación, la aproximación propuesta combina ideas de las arquitecturas dirigidas a eventos (Event-Driven Architectures - EDAs) y del software auto-adaptativo.

Un análisis de los conceptos clave de las EDAs, así como sobre el flujo de procesamiento de eventos y de los principales componentes que formarán su estructura (y como deben ser implementados), puede encontrarse en Michelson (2006). Respecto al software auto-adaptativo, podemos mencionar el trabajo de Laddaga y Robertson (2004), en el que se encuentra su definición formal y una revisión sobre los principales retos y metodología de implementación.

La combinación de ambas ideas es fundamental en la arquitectura del CCT propuesta en esta tesis. Por un lado, las arquitecturas orientadas a eventos darán soporte a componentes software asíncronos y de naturaleza desacoplada, que potenciarán la integración simultánea de diferentes tipos de sensores y vehículos autónomos. Por el otro, el software adaptativo proporcionará las herramientas necesarias para modificar dinámicamente el comportamiento de ciertos módulos software que permitirán introducir, cambiar o quitar dispositivos del sistema durante el desarrollo de la misión. Además, se aprovecharán estos conceptos para la implementación de técnicas de ayuda al operador en el GUI, cuyos componentes cambiarán en tiempo de ejecución con el objeto de mejorar la ergonomía y eliminar parte de la información, considerada poco relevante, para disminuir la carga mental de trabajo del operador (y por tanto su estrés y fatiga) y mejorar su tiempo de reacción ante tareas y toma de decisiones.

3.1. Arquitectura Dirigida a Eventos

Hemos diseñado un entorno de desarrollo arquitectónico adaptativo que aprovecha las ventajas de combinar la filosofía de las arquitecturas dirigidas a eventos (EDAs), con módulos desacoplados que reaccionan ante los mensajes (eventos) recibidos y el patrón Observador (Observer pattern) que permite la implementación sencilla y robusta de un sistema de comunicaciones asíncrono. Este entorno puede ser usado para diseñar infraestructuras distribuidas complejas y adaptativas para casi cualquier ámbito en el que se necesite integrar múltiples dispositivos heterogéneos (vehículos autónomos, sensores, agentes inteligentes autónomos, etc.), monitorizarlos y controlarlos.

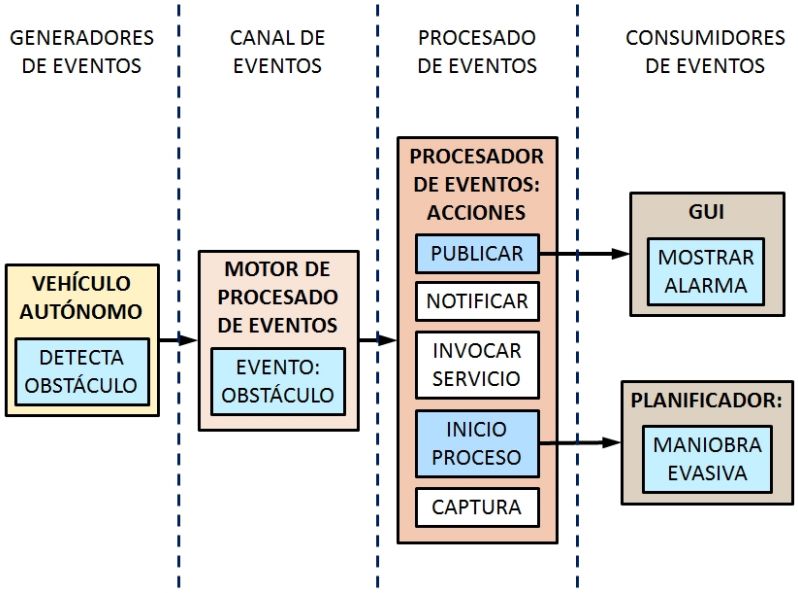


Figura 3.1: Flujo de la Generación y Consumo de un Evento

Tradicionalmente, como señala Michelson (2006), las arquitecturas dirigidas a eventos están divididas en 3 grandes bloques de módulos: 1) los Generadores de eventos, también conocidos como Agentes (Agents), 2) Consumidores de eventos, también conocidos como Sumideros (Sinks) y 3) Canal gestor de eventos (Channel Manager) que propagará los eventos desde los productores a los consumidores. En la Figura 3.1, podemos ver un diagrama sencillo del flujo de eventos inspirado en el trabajo de Michelson (2006) y aplicado a nuestro CCT. En él podemos ver como un generador crea un evento (por ejemplo, un vehículo detecta un obstáculo) cuyos datos son interpretados y encapsulados en un evento (obstáculo detectado) para ser procesado y posteriormente transmitido a los consumidores de eventos que activan los comportamientos programados para cuando se reciba ese tipo de evento (el interfaz mostrará una alarma al operador y el planificador activará el protocolo de maniobra evasiva).

La arquitectura dirigida a eventos ayuda al entorno de desarrollo a desacoplar los agentes generadores de eventos (en este caso, sensores, vehículos, operadores, etc.) de los consumidores de eventos (en este caso, los diferentes módulos de nuestro CCT) y permite a los procesadores de datos recibir mensajes asíncro-

nos en tiempo real, de distintas procedencias, y de forma robusta y segura, lo que supone una gran ventaja a la hora de supervisar un equipo de dispositivos heterogéneos que pueden enviar sus datos en cualquier instante de la misión. El desacoplamiento es también una característica muy importante para lograr la Adaptabilidad, ya que permite que cualquier infraestructura diseñada siguiendo este esquema añada, quite o cambie en tiempo real cualquiera de los Agentes (sin importar su hardware, software o protocolo de comunicaciones) o Sumideros de la infraestructura. Todo esto, hace nuestro software tolerante a cambios en el entorno, en los objetivos de la misión o en la tarea actual, ya que permite introducir otros dispositivos, vehículos o sensores para adaptarnos a ello. Además, el desacoplamiento potencia la simulación distribuida en tiempo real, permitiendo a una computadora remota generar datos (por ejemplo de la planta que modela el comportamiento de un vehículo autónomo) y unirse al sistema en tiempo de ejecución. Como podemos observar en la Figura 3.2, la arquitectura esta formada por los 3 bloques de módulos anteriormente mencionados: Agentes, Sumideros y Canal gestor de eventos. A continuación explicaremos cada uno de los bloques en detalle.

3.1.1. Agentes: Generadores de Eventos

En nuestra arquitectura dirigida a eventos, un agente es un módulo que "siente" algo (información procedente de sensores, actuadores o disparadores externos) y, en consecuencia, genera eventos. La arquitectura cuenta con una infraestructura que permite a cualquier dispositivo autónomo con capacidad de conexión remota unirse al sistema en tiempo de ejecución. Esto es posible gracias a una capa de comunicaciones añadida en el hardware embebido del dispositivo (o en una micro-computadora como Raspberry Pi, Intel Edison, Arduino, etc. si no fuese posible) y un fichero de configuración XML donde se guardan los datos que indican al sistema cómo configurarse y comunicarse con cada agente. En nuestro caso implementaremos un protocolo de comunicaciones basado en TCP/IP, pero podría adaptarse el sistema de forma sencilla a otros protocolos como los mencionados en la Sección 2.2.2 (WebSocket (2010), WebHooks (2007), Server-

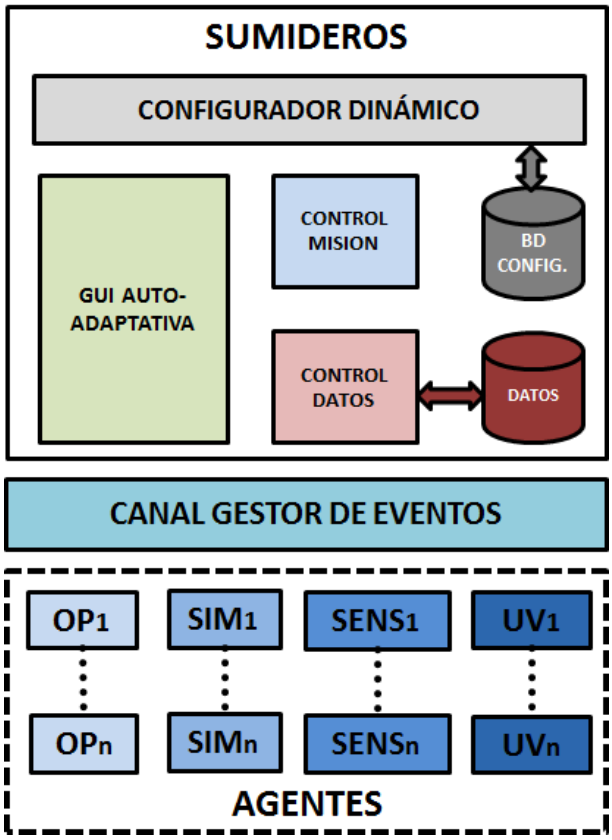


Figura 3.2: Arquitectura Adaptativa Dirigida a Eventos

Sent-Events (2013)), ya que gracias al diseño modular de la arquitectura, se pueden añadir los modulos software necesarios sin alterar el funcionamiento del resto del centro de control. En la Figura 3.3 podemos ver un esquema de comunicaciones y el diagrama del sistema de *negociación de conexión* (handshake) entre un agente y el canal gestor de eventos. En primer lugar, cuando el agente (un vehículo autónomo, por ejemplo) se activa, un proceso automatizado alojado en su computadora embebida comienza el proceso de conexión al sistema. Inicialmente, el programa extrae los datos del fichero XML de configuración (identificador único del agente, protocolo de comunicaciones e identificador y tipo de datos que enviará) y elabora una clave para la negociación de conexión (handshake) que ayudará al canal gestor de eventos a: 1) aceptar o rechazar la conexión, 2) configurar el software adaptador intermedio (Middleware Adapter)

que realiza la traducción del protocolo, y 3) configurar la forma de encapsular los datos que acompaña a cada evento. Una vez elaborada esta clave, será enviada al puerto por defecto del servidor (también configurado en el documento XML) del Canal que, al recibirlo, lo corroborará por medio del validador de claves (que extraerá los datos necesarios de la base de datos habilitada para ello). Si la clave es correcta, el canal establecerá un puerto único para ese agente y le enviará un mensaje de validación que le indicará que puede comenzar a enviar datos (eventos). Por el contrario, si la clave no es validada correctamente, se enviará un mensaje de error al agente (que reaccionará según el protocolo establecido reintentando la conexión o poniéndose en modo espera) y se mostrará una alarma que indicará al operador que ha habido un intento de conexión que ha sido rechazada y el identificador del vehículo que ha sido rechazado.

Consideraremos, a partir de ahora, los 4 tipos de agentes que podemos ver en la parte inferior de la Figura 3.2: vehículos autónomos, sensores externos (ajenos a los vehículos), motores de simulación y operadores.

Vehículos Autónomos pueden ser de varios tipos, aéreos (UAVs), de superficie (USV), de tierra (UGVs) o submarinos (UUVs). Son un tipo especial de agente que debe enviar datos (eventos) a través del canal, una vez conectados siguiendo el protocolo mostrado en la Figura 3.3, y recibir ciertos comandos (consignas, rutas, etc.) que el operador puede enviar a través del planificador y/o comandador integrado en el interfaz de usuario del CCT. Este tipo de agente genera información que es encapsulada en dos tipos de eventos: a) *eventos de datos* (Data Events), en los que se incluye la telemetría y la información del resto de sensores del vehículo, y b) *eventos de alarma* (Alarm Events), que representan un mensaje urgente con información crítica que debe ser mostrada al operador y procesada por el módulo de control de misión (Mission Control Module) que actuará en consecuencia (según el proceso que se detallará en la Sección 3.1.3).

Sensores Externos generan eventos relacionados con cualquier tipo de información sensada del entorno (por ejemplo viento, temperatura exterior o

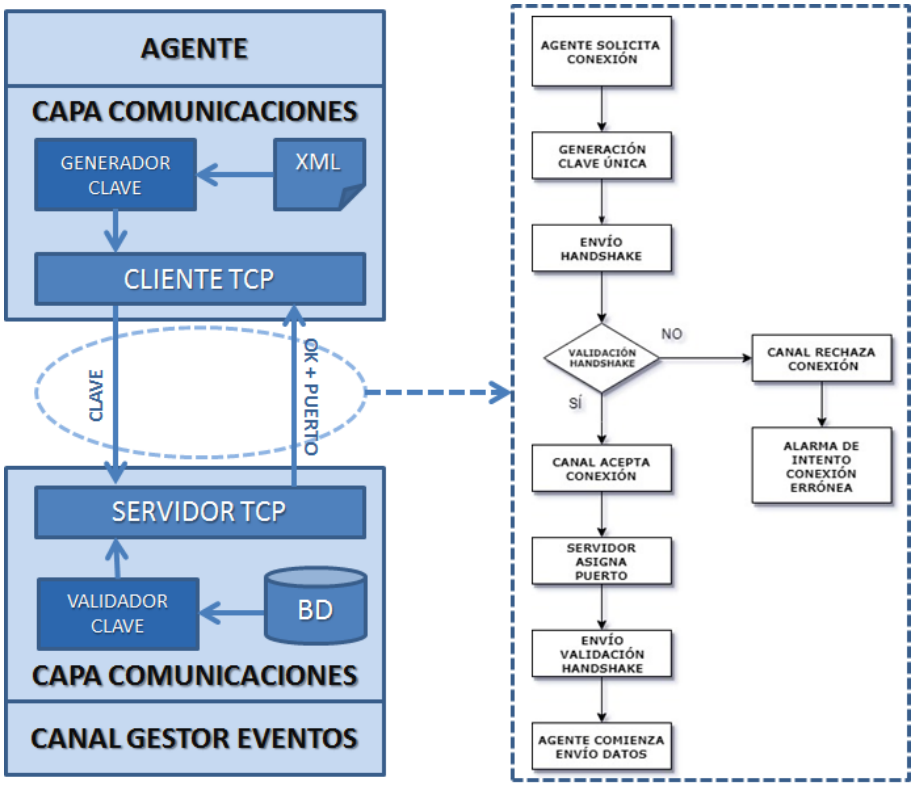


Figura 3.3: Esquema de la Capa de Comunicaciones en un Agente y Diagrama de Aceptación de la Conexión

corrientes marítimas), datos concretos que deban ser monitorizados en elementos que participen en alguna tarea y que no estén integrados en los vehículos, imágenes o detección de movimiento que provenga de un sistema de vigilacia, etc.

Motores de Simulación son computadores remotos en los que se ha instalado la capa de comunicaciones de nuestro entorno de desarrollo adaptativo y que cuentan con el modelo de comportamiento de alguno de los agentes mencionados previamente (por ejemplo, el modelo de navegación de un vehículo autónomo), que permite al sistema llevar a cabo simulaciones distribuidas en tiempo real. Los datos generados en el computador remoto son enviados a los sumideros a través del canal de eventos siguiendo los mismos mecanismos que si se tratase de un agente real. Por lo tanto, la simula-

ción es transparente para el resto de componentes de la infraestructura, quedando reflejado como si de un agente convencional se tratase.

Operadores. En el ámbito que nos ocupa, consideraremos también a los operadores como un tipo particular de agente, ya que pueden producir ciertos tipos de eventos que llamaremos solicitudes de operador (Operator Requests). Este tipo de evento es usado frecuentemente por un operador que solicita información al sistema, como puede ser el estado de un vehículo, la progresión de una tarea, etc. También existe un tipo especial de solicitud de operador llamada Solicitud de cambio estructural (Infrastructure Change Request, ICR), una de las características más importantes de nuestro CCT adaptativo. Un ICR es un mensaje que sirve para sincronizar todas las partes involucradas en un cambio crítico en la infraestructura del CCT, como puede ser un nuevo agente que solicita su ingreso en el sistema o un operador que solicita delegar tareas en otro porque está sobrepasado. Por ejemplo, cuando un nuevo agente se conecta al sistema en tiempo de ejecución, éste solicita su ingreso al sistema a través del canal gestor de eventos. En ese momento el módulo de control genera un ICR que será recibido por el sumidero apropiado, en este caso el GUI adaptativo. El interfaz mostrará un mensaje al operador principal, que deberá dar permiso para realizar los cambios estructurales. El módulo de configuración dinámica (Runtime Configuration Manager) activará los mecanismos adaptativos necesarios para cambiar la infraestructura basándose en la información almacenada en la base de datos de configuración (Configuration DataBase), donde está almacenado el conocimiento necesario para recomponer la infraestructura software con los módulos necesarios para integrar al nuevo agente. Un ICR también puede ser generado debido a una solicitud de delegación de tareas. Por ejemplo, si el Operador 1 se encuentra sobrepasado por la cantidad de tareas acumuladas, deberá pedirle a otro operador disponible, por ejemplo el Operador 2, que se haga cargo de las tareas relacionadas con alguno de los vehículos a su cargo. Si el Operador 2 da su consentimiento, se iniciará el proceso de delegación de tareas. En ese momento, los elementos gráficos

relacionados con las tareas delegadas aparecerán en el GUI del Operador 2, que pulsará el botón "Comenzar" cuando esté preparado. En este punto, dichos elementos gráficos desaparecerán del GUI del Operador 1, liberándole de esa parte de su carga de trabajo.

3.1.2. Canal: Comunicaciones y Gestión de Eventos

Los canales son los módulos software que utilizan las arquitecturas dirigidas a eventos para transferir un evento desde un agente al sumidero apropiado. En nuestra arquitectura adaptativa dirigida a eventos, solo hay un canal gestor de eventos multipropósito (Channel Manager) que realizará todas las tareas relacionadas con los canales de distribución de eventos. Las conexiones entre los módulos distribuidos de la infraestructura software se realizan mediante Sockets TCP/IP para asegurar la compatibilidad entre todo tipo de computadoras independientemente de su hardware y sistema operativo. Para la difusión de mensajes entre agentes y sumideros, el canal de eventos se beneficia del patrón Observador, quedando configurado como elemento observado (Subject), mientras que los sumideros quedan configurados como observadores (Observers). Podemos observar este esquema en la Figura 3.4, sobre la que también indicamos el flujo de información desde que un evento es generado hasta que es recibido por el sumidero adecuado.

El canal gestor de eventos es el elemento observado, y está diseñado como un modulo multi-capa con un servidor, adaptadores software intermedios (Middleware Adapters) y un generador de eventos (Event Generator), cuyas tareas dentro de la estructura de comunicaciones de nuestro entorno de desarrollo detallaremos a continuación.

Servidor es la capa software del Canal de Eventos que gestiona las peticiones de conexión. Se encarga de establecer y mantener a lo largo del desarrollo de la misión un canal de comunicaciones seguro y robusto entre los agentes y los sumideros. El servidor tiene la capacidad de recibir peticiones de conexión por Socket TCP/IP y crear un canal único de entrada/salida para cada

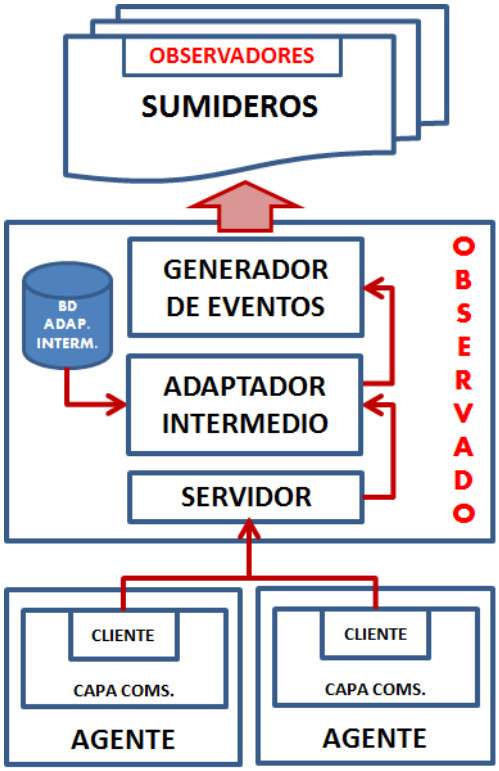


Figura 3.4: Flujo de Eventos en el Sistema

agente. Una vez los mensajes enviados por los agentes lleguen al servidor, éstos serán enviados a los Adaptadores Intermedios para su tratamiento.

Adaptadores Intermedios son los módulos que decodifican los diferentes protocolos usados por cada uno de los agentes, ayudando a desacoplar a los elementos "productores" de los elementos "consumidores" de nuestra infraestructura. El adaptador intermedio se configura con ayuda de la información enviada en la petición de conexión inicial del agente (handshake) vista en la Sección 3.1.1. Las claves para decodificar los protocolos de cada agente son almacenados previamente en la base de datos para adaptadores intermedios (MiddleWare DataBase), de donde cada adaptador obtendrá la información correspondiente. Tras ser decodificada, la información llegada de los agentes es trasladada al Generador de Eventos.

Generador de Eventos: es la estructura encargada de recibir la información recientemente decodificada por los adaptadores intermedios y encapsularla en un Objeto Evento que será propagado por el sistema gracias al patrón Observador. Cuando el software del canal gestor de eventos (observado) detecta que un evento ha sido creado, activa el disparador que notifica a todos los observadores (sumideros) que dicho elemento está en el sistema para que, en base a los datos de qué tipo de evento es, éste sea recibido y tratado por el sumidero adecuado, que reaccionará con los mecanismos programados al efecto.

Para ilustrar el flujo de los datos desde que llegan al canal hasta que son recibidos por un sumidero, nos apoyaremos en la Figura 3.4, utilizándola como soporte visual. En primer lugar, el agente debe realizar una solicitud de conexión a la infraestructura enviando una petición al servidor del Canal por medio del Socket cliente alojado en la capa de comunicaciones introducida en su hardware (o micro-computadora agregada). Esta primera conexión se realiza con los datos extraídos del archivo de configuración XML y envía un mensaje para el inicio de la conexión, que automatiza un proceso de negociación (handshake) entre las dos partes implicadas en un canal de telecomunicaciones utilizado para validar ambas partes así como para intercambiar los datos necesarios para establecer las comunicaciones. En este caso, por ejemplo, se envían datos sobre el protocolo de comunicaciones propietario que utiliza dicho agente. Una vez recibida la petición del agente, el servidor del canal validará la conexión, configurará el adaptador intermedio adecuado al protocolo recibido y abrirá un socket para comunicarse con ese agente. A partir de este punto, el Canal recibe los mensajes de ese agente por la vía establecida y los transfiere al adaptador intermedio adecuado. El adaptador recibe los datos, los decodifica utilizando la información almacenada en la base de datos de protocolos (MiddleWare Database) y los entrega al generador de eventos. En esta capa, los datos se encapsulan en un Objeto Evento y se activa el disparador del patron Observador que notifica a todos los módulos observadores que se ha creado un nuevo evento. A partir de aquí, el patron Observador (implementado por medio de Interfaces Java), se encarga de

difundir el Evento a los sumideros, que son capaces de discriminar si deben recibirlo y, en caso afirmativo, iniciar los procesos programados para tratar los datos encapsulados en su interior.

3.1.3. Sumideros: Consumidores de Eventos

Los sumideros son procesadores de eventos y se encargan de identificar un evento y reaccionar a él con los mecanismos apropiados. Como hemos señalado anteriormente, los sumideros cumplen el rol de observadores en el patrón Observador. Esta característica permite que los módulos de nuestra infraestructura puedan desarrollarse fácilmente ya que el único requisito que deben cumplir es implementar el método de actualización (update) del Observador que provee el interfaz para programación de aplicaciones (Application Programming Interface, API) de Java. Esta característica dota a nuestro entorno de desarrollo de una gran flexibilidad y reusabilidad, ya que puede ser utilizado para múltiples propósitos con algunos cambios puntuales bastante sencillos. Además, en el ámbito que nos incumbe, un CCT desarrollado siguiendo esta filosofía, se adaptará a cualquier cambio que se produzca en el entorno en el que se realiza la misión (o en sus objetivos) añadiendo, cambiando o quitando módulos en tiempo de ejecución.

En el entorno de desarrollo propuesto hay cuatro tipos de sumideros: Control de Datos , Control de Misión, Configurador Dinámico e Interfaz de Usuario Auto-Adaptativo.

Control de Datos es el sumidero que gestiona el almacenamiento y el tratamiento de los datos recibidos de los Agentes. Filtra los datos relevantes para el entorno de desarrollo, y los almacena en una base de datos para que posteriormente puedan ser consultados y analizados para realizar futuros experimentos y mejoras. Hay tres bloques principales de datos que se almacenan de forma independiente. El primer bloque está formado por los datos relacionados con la misión (telemetría de los vehículos, datos del entorno, objetivos cumplidos, etc.). El segundo bloque está formado por

los datos relacionados con el operador (tiempos de reacción, carga mental de trabajo, tiempo que tarda en realizar las tareas relacionadas con la misión, solicitudes de información al sistema, etc.). En tercer lugar está el bloque de datos referentes al interfaz de usuario (el lugar que ocupa en cada instante cada uno de los elementos gráficos mostrados en el GUI, la cantidad de elementos gráficos mostrados por cada vehículo, las solicitudes del operador para que se muestren por pantalla datos no disponibles, etc.).

Control de Misión es el módulo encargado de procesar y reaccionar ante los eventos relacionados con tareas que deben ser realizadas por los Agentes, así como aquellas relacionadas con los datos monitorizados por los sensores de la infraestructura. En este módulo están implementados los mecanismos relacionados con todas las maniobras reactivas que deber ser realizadas ante la recepción por los agentes de ciertos eventos. Por ejemplo en un CCT diseñado para múltiples vehículos autónomos heterogéneos, como es nuestro caso, este modulo se encarga de activar los protocolos reactivos ante ciertas alarmas (lanzar una maniobra evasiva si un vehículo se encuentra con un obstáculo, reducir la velocidad de un vehículo si se recibe una alarma de "velocidad alta", reducir el consumo de energía si se recibe una alarma de "batería baja", etc.). Este tipo de mecanismos automatizados es de gran importancia para los CCT, ya que ayudará al sistema a disminuir la carga mental de trabajo del operador y, en algunos casos, a reducir a uno el número de operadores requeridos para monitorizar/controlar una misión.

Configurador Dinámico es la estructura encargada de realizar los cambios en la infraestructura de nuestro software (un CCT en este caso). Es decir, añade, cambia o quita módulos en tiempo de ejecución para adaptar al sistema a una nueva situación. Cuando el sistema detecta un nuevo requisito que no puede ser cumplido o recibe una solicitud de cambio estructural (ICR), el configurador dinámico activa los mecanismos de adaptación del software. Las reglas para implementar los cambios automáticos están almacenadas

en la Base de Datos de Configuración (Configuration DataBase), que además puede leer algunos de ellos de un archivo de configuración XML si un nuevo agente va a unirse al sistema. Cuando el operador da su permiso y el sistema alcanza una situación no crítica que permita realizar los cambios, el configurador dinámico ejecuta los cambios solicitados.

Interfaz de Usuario Auto-Adaptativo es, como hemos comentado con anterioridad, uno de los módulos más importantes de una aplicación y más concretamente de un CCT, ya que es responsable de mostrar al usuario todos los datos relevantes y permitirle interactuar con el resto del sistema. Muestra los datos de la forma más ergonómica posible para reducir la carga mental de trabajo y, por tanto, el estrés y el cansancio del operador. Este Sumidero ha sido diseñado como un subsistema complejo, siguiendo la misma filosofía marcada por nuestra arquitectura adaptativa dirigida a eventos, que da lugar a una estructura auto-adaptativa independiente que ha sido alojada dentro de la estructura principal. Dada su importancia y complejidad, este subsistema se describirá en detalle en el Capítulo 4.

3.2. Marco Experimental: Experimentos de Validación

Una vez realizado el diseño e implementación de la infraestructura distribuida adaptativa dirigida a eventos que permite la integración de componentes en tiempo de ejecución (reales o simulados en tiempo real), se deben realizar algunos experimentos para determinar si tanto la infraestructura que forma nuestro sistema distribuido como el CCT y su interfaz gráfico son capaces de realizar las tareas para las que han sido concebidos.

Las primeras pruebas, antes de introducir el hardware físico (desplegar vehículos en el agua, por ejemplo), han sido realizadas por medio de un marco experimental específicamente diseñado para estudiar qué sucede en aquellas situaciones que se consideran más críticas durante el desarrollo de las misiones que se realiza-

rán en el marco del proyecto SALACOM. Más concretamente se han diseñado 3 familias de experimentos diferentes con el objetivo de verificar el correcto funcionamiento del CCT en las 3 situaciones críticas más relevantes para el desarrollo de nuestras misiones: 1) problemas de comunicación, 2) integración de nuevos vehículos en la infraestructura software, y 3) integración de nuevos CCTs durante una misión. En la primera se simulará una caída de red y se pondrán a prueba los mecanismos diseñados para evitar la pérdida total de comunicaciones con los vehículos, que podría resultar muy peligrosa cuando estos se encuentren en el agua a gran distancia del lugar donde se ha establecido el CCT. En la segunda se simulará la integración de nuevos vehículos a la misión en tiempo de ejecución, ya que un fallo en alguno de los mecanismos de la infraestructura que soporta dicha integración podría echar a perder la misión y ocasionar daños en el hardware por pérdida de control del mismo. En la tercera se simulará la inclusión en la infraestructura de nuevos centros de control para supervisar la misión, ya que un fallo en este punto también podría ocasionar errores inesperados en la monitorización o control de los vehículos que podrían acarrear consecuencias para el hardware o la misión.

A continuación describiremos con más detalle cada uno de los grupos experimentales, así como los objetivos que se desean alcanzar en cada uno de ellos. También se presenta el marco experimental diseñado para cada experimento y sus módulos, diseñados de forma que puedan ser fácilmente reutilizados y escalados para su utilización en futuros experimentos. Por último se analizan los resultados obtenidos durante la ejecución de cada experimento.

3.2.1. Experimentos de Comunicaciones y Caída de Red

Una de las situaciones críticas más importantes que pueden darse durante la realización de una misión en un entorno al aire libre es la pérdida de comunicaciones, ya que puede ser causada por aspectos ajenos al diseño de la misión tan diversos como: las condiciones meteorológicas que influyen en la cobertura de la red de comunicaciones, que uno de los vehículos quede fuera de rango de la antena de comunicaciones por errores en la planificación de las rutas,o la pér-

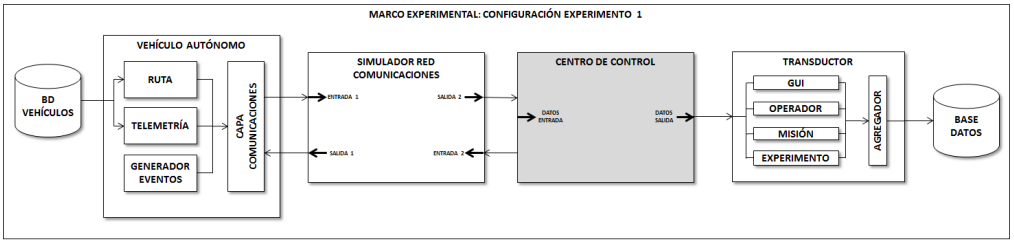


Figura 3.5: Configuración del Marco Experimental 1

dida momentánea del flujo de corriente que alimenta alguno de los componentes hardware involucrados. Si esto ocurre, debemos estar seguros de que el sistema reaccionará adecuadamente avisando al operador y tratando de reestablecer las comunicaciones con los componentes remotos (los vehículos en este caso) o, en el peor de los casos, realizando una parada de emergencia para que la situación no empeore.

3.2.1.1. Diseño del Marco Experimental 1

Para el primer bloque de experimentos se ha diseñado un marco experimental con tres módulos que someten al sistema bajo estudio (el centro de control) a unas condiciones en las que se pierde la conexión con un vehículo durante un breve período de tiempo. Durante estas pruebas, la conexión se realiza por medio de Sockets TCP/IP en modo local para que las comunicaciones entre el vehículo y el CCT sean similares a las realizadas en los experimentos de campo. Los tres bloques que forman el marco experimental del experimento 1, junto al sistema bajo estudio (en gris), pueden verse en la Figura 3.5 y son descritos en detalle a continuación:

Vehículo autónomo es el módulo encargado de simular el comportamiento de uno de los vehículos autónomos integrado en el sistema. Para ello cuenta con cuatro bloques internos y una base de datos que guarda la información generada por un vehículo autónomo durante una misión real. El módulo de generación de rutas lee las coordenadas (latitud y longitud) de la ruta planificada para un vehículo y las transmite a la capa de comunicaciones.

El módulo de generación de telemetría también lee de la base de datos toda la información relevante sobre la telemetría del vehículo (batería, orientación, cabeceo, alabeo, datos del timón, velocidad, etc.) durante la misión real previa y la envía a la capa de comunicaciones. El tercer módulo es el encargado de generar eventos aleatorios relacionados con posibles situaciones de la misión (por ejemplo, la detección de un obstáculo). Por último, la capa de comunicaciones está implementada de forma similar a la capa de comunicaciones de un vehículo autónomo del sistema, se encarga de realizar la petición de conexión inicial al sistema (incluida la negociación de conexión) y de encapsular los datos recibidos del resto de módulos (ruta, telemetría y eventos) para que sean enviados al sistema y mostrados en el CCT. Además, en este módulo se ha implementado un mecanismo tipo "dispositivo de hombre muerto", según el cual si no se recibe información del sistema en un periodo de tiempo estipulado (en los ejemplos, 30 segundos) se realiza un paro de emergencia del vehículo y se ejecuta el algoritmo de reestablecimiento de comunicaciones, que reiniciará el canal de comunicaciones e intentará realizar la solicitud inicial de conexión cada 30 segundos hasta que logre conectarse de nuevo al sistema.

Simulador de red de comunicaciones es un módulo intermedio que hace las veces de red de comunicaciones entre el vehículo y el centro de control. Recibe las comunicaciones del CCT y del vehículo y las envía al receptor indicado salvo que se esté simulando una caída de red, en cuyo caso los mensajes no serán enviados al receptor. Este módulo permitirá probar los mecanismos de reestablecimiento de comunicaciones en el vehículo y de alerta al operador en el centro de control (que si no recibe datos de alguno de los vehículos monitorizados en un periodo de 30 segundos, muestra un aviso de pérdida de conexión para que el operador pueda realizar las comprobaciones necesarias para recuperar las comunicaciones con el vehículo). Para configurar los experimentos de comunicación, se puede introducir tanto la duración del corte de red como la probabilidad de que dicho corte ocurra. Durante los experimentos analizados en esta memoria, los cortes

tienen una duración de 30 o 60 segundos y se realizan con una probabilidad del 100 % para asegurarnos de que ocurren cuando los tenemos programados.

Transductor es el módulo encargado de recibir los datos del sistema y guardarlos en un almacenamiento persistente. Se ha estandarizado el comportamiento de este módulo, de forma que sirva para todas las pruebas que realicemos utilizando los diferentes marcos experimentales que se presentan en esta tesis. Para ello se han habilitado cuatro canales diferenciados de recepción de datos y un agregador que se encarga de tratarlos y guardarlos en un almacenamiento persistente. Dependiendo del marco experimental diseñado, se reciben los datos por uno o varios de los canales habilitados. En primer lugar, contamos con el canal del GUI, por el que se reciben datos del interfaz de usuario, para poder guardar su estado, y la cantidad y el orden de los elementos gráficos mostrados. También se pueden recibir datos por el canal del operador, como son su carga mental de trabajo, su tiempo de reacción y el tiempo que tarda en llevar a cabo algunas tareas. En tercer lugar, por el canal de misión se reciben los datos relacionados con la misión, como son los eventos relevantes recibidos de cada vehículo, información del entorno y los objetivos cumplidos. Por último, por el canal de experimento se reciben los datos relativos al experimento, como son datos de inicio de la simulación, tiempos de ejecución, etc. Más en concreto, en el caso del experimento 1 de este capítulo, los datos relevantes son: 1) el número activaciones del protocolo de hombre muerto (que consta de la parada de emergencia y el bucle de solicitud de reestablecimiento de comunicaciones) en el vehículo, 2) el número de peticiones aceptadas y rechazadas por el sistema, y 3) el número de veces que el sistema muestra correctamente el aviso de pérdida de conexión.

3.2.1.2. Diseño de los Experimentos

Para evaluar la efectividad del protocolo de hombre muerto así como la robustez del sistema frente a caídas de red mediante pruebas de repetitividad con sucesivas caídas de red (caso extremo en el que se producen breves cortes cada pocos segundos) se han realizado dos ejecuciones de la simulación diseñada con el marco experimental número 1.

En la primera ejecución se lleva a cabo una secuencia de mensajes intercalados con cortes de red que permiten interrumpir y recuperar la conexión entre el vehículo y el CCT. Más en concreto, el vehículo envía mensajes al CCT cada segundo (como en los vehículos reales), el primero llega con éxito al CCT, que responderá con una señal para indicar que sigue conectado (en este caso, también cada segundo aunque en misiones reales puede enviarse cada más tiempo). A continuación, el simulador de red de comunicaciones simulará un corte de red de 30 segundos, lo que activará el protocolo de hombre muerto (intento de reconexión por parte del vehículo y alarma de pérdida de conexión al operador por parte del CCT). Tras esos 30 segundos, el simulador de red volverá a permitir las comunicaciones entre CCT y vehículo para que éstos puedan volver a conectarse y seguir comunicándose. En este momento se volverá a enviar el paquete de datos del vehículo al CCT, que contestará con la señal de hombre muerto. Tras la recepción de estos mensajes se volverá a introducir un corte de 30 segundos. El experimento realizará este proceso sucesivamente durante 50 iteraciones.

La segunda ejecución del experimento será similar, pero esta vez se simula una caída de 60 segundos, de forma que el algoritmo de reestablecimiento de comunicaciones del vehículo tenga ocasión de realizar el intento una primera vez, fallar, y volver a intentarlo con el objetivo de probar que en caso de durar la caída de red más de 30 segundos, el vehículo se parará y seguirá intentando conectarse hasta que lo consiga o se quede sin batería.

Para estos experimentos se supone una red de comunicaciones ideal en la que se cuenta con un ancho de banda suficiente y con una cobertura perfecta salvo por los cortes introducidos por el simulador de red. Por lo tanto, no se tienen

Tabla 3.1: Resultados del Marco Experimental 1

# IT	% MR	%AM	IR	RE	%E	TDP (seg.)	TTDP (seg.)	TTE (seg.)
50	100%	100%	50	50	100%	< 1	37,4	1637,4 (~27 min.)
50	100%	100%	100	50	50%	< 2	1º. 36,86 2º. 37,82	3174,68 (~53 min.)

en cuenta retardos en la conexión o fallos de recepción de mensajes que no sean los forzados para el experimento. Los tiempos y retardos propios de la red, se consideran irrelevantes para las conclusiones que se extraerán del experimento, ya que en los experimentos de campo pueden variar notablemente dependiendo del hardware de comunicaciones disponible, del número de elementos software conectados, de la distancia entre los elementos que establecen comunicaciones, del entorno en el que se realiza, etc. Además, el objetivo de este experimento consiste en probar la robustez del CCT frente a caídas en la red (algoritmo de reestablecimiento de comunicaciones y sistema de avisos al operador), no en evaluar la propia red, por lo que las comunicaciones establecidas serán, en este caso, un medio para medir los algoritmos de recuperación ante caídas y una forma de demostrar que, a pesar de ellas, el sistema es capaz de seguir funcionando de forma correcta y de recuperarse en un breve periodo de tiempo siempre que la red lo permita. Por estas razones, las comunicaciones se configuran en un entorno local ya que en este experimento sólo se realiza el intercambio de mensajes entre módulos vía TCP/IP y la simulación de cortes o caídas momentáneas de la red de comunicaciones.

3.2.1.3. Resultados

A continuación analizamos los resultados obtenidos en ambas ejecuciones, que pueden verse en la Tabla 3.1 y en la Figuras 3.6 y 3.7. En la tabla encontramos, de izquierda a derecha, las columnas siguientes: #IT, representa el número de iteraciones realizadas; %MR, el porcentaje de mensajes recibidos con éxito; %AM, el porcentaje de alarmas mostradas; IR, el número de intentos realizados de reestablecimiento de comunicaciones; RE, el número de intentos reali-

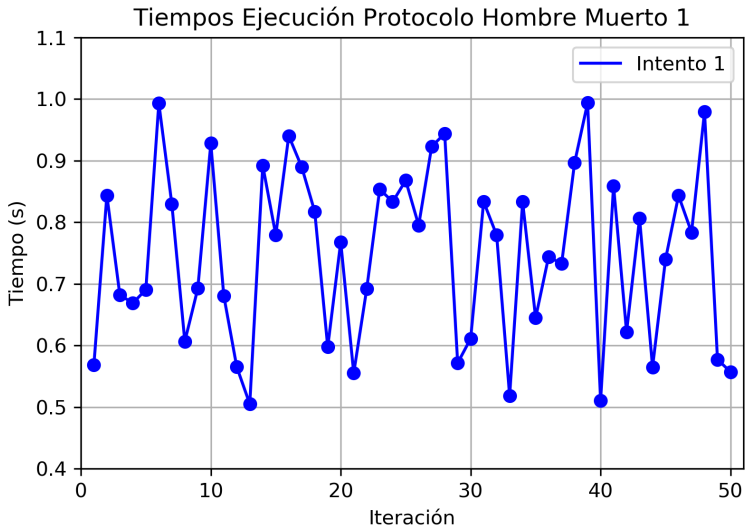


Figura 3.6: Tiempos de Ejecución del Algoritmo de Reconexión en el Primer Experimento del Marco Experimental 1

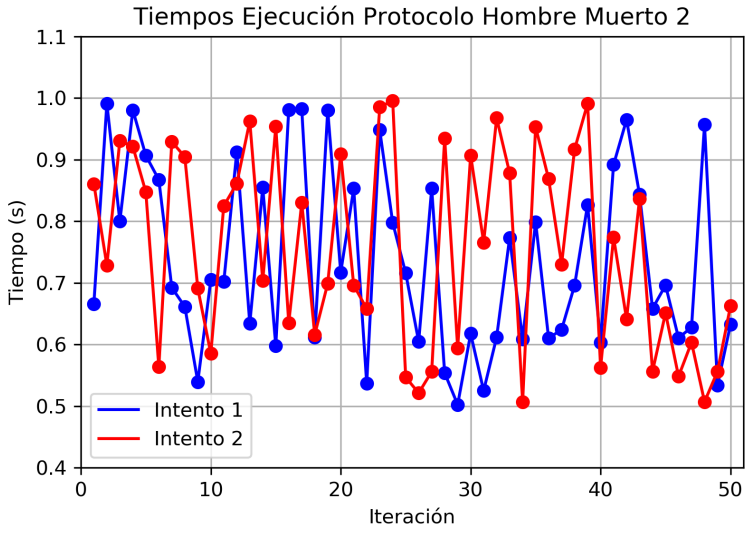


Figura 3.7: Tiempos de Ejecución del Algoritmo de Reconexión en el Segundo Experimento del Marco Experimental 1

zados con éxito; %E, el porcentaje de éxito en el reestablecimiento de comunicaciones; TDP, el tiempo de duración del protocolo de parada y reconexión; TTDP,

el tiempo total de duración del protocolo (suma de cada uno de los anteriores); y TTE, el tiempo total de ejecución del experimento. En las Figuras 3.6 y 3.7 podemos observar, respectivamente, los tiempos de ejecución del algoritmo de reestablecimiento de comunicaciones en cada iteración de la primera ejecución del experimento, y los tiempos del primer y segundo intento de ejecución del algoritmo en cada iteración de la segunda ejecución del experimento.

Durante la ejecución de ambas pruebas simuladas, se han obtenido los resultados esperados ya que tanto el algoritmo de reestablecimiento de comunicaciones como el mecanismo de avisos al operador han mostrado ser lo suficientemente robustos, realizando su labor correctamente en todas las iteraciones. Por un lado, el mecanismo de aviso de pérdida de comunicaciones con el vehículo del CCT ha sido exitoso, ya que el mensaje de alarma que se muestra en el interfaz de usuario si no se reciben mensajes con ese identificador de vehículo en un tiempo igual o superior a 30 segundos se ha mostrado correctamente en todas las ocasiones. Éste mensaje desaparece automáticamente una vez se recupera la conexión. Por otro lado, el algoritmo de reestablecimiento de comunicaciones implementado en los vehículos también ha demostrado la suficiente robustez y eficacia, logrando 50/50 intentos en la primera ejecución (100 % de éxito) y 50/100 en la segunda ejecución (50 % de éxito) y unos tiempos de ejecución que aseguran el buen desarrollo del mecanismo de reconexión. Respecto al resultado de conexiones exitosas durante la segunda ejecución de la simulación, a pesar de que figure un 50 % de éxito, se trata en realidad de un éxito equivalente al 100 %, ya que con el diseño del experimento se pretendía probar que el bucle sigue intentando reconectarse hasta que la red vuelve a estar disponible. En este caso, al haber introducido un corte de 60 segundos, el bucle itera en dos ocasiones, produciendose 2 intentos consecutivos en los que sólo el segundo (una vez pasados los 60 segundos de corte programado en el simulador de comunicaciones) puede tener éxito.

Los tiempos de ejecución del algoritmo de reestablecimiento se encuentran, en todos los casos, por debajo de un segundo en el proceso de obtener la clave para la negociación de conexión inicial, enviarla al CCT y reiniciarse el algoritmo de reconexión para una segunda iteración (o continuar con el envío de datos

si se tiene éxito a la primera). Los tiempos oscilan entre 0.5 y 1 segundo en todas las iteraciones. Para poder observar información más detallada de estos tiempos, las Figuras 3.6 y 3.7 muestran, para los 50 intentos de conexión, los tiempos de la primera y segunda ejecución del algoritmo de reestablecimiento. El tiempo total de la suma de las 50 iteraciones del algoritmo de reestablecimiento de comunicaciones puede verse en la penúltima columna de la tabla 3.1. Finalmente, el tiempo total de ejecución de cada uno de los experimentos figura en la última columna de la tabla. Este proceso incluye los envíos del primer paquete de datos del vehículo al CCT (1 seg.), la respuesta del CCT (1 seg.), los 30 seg. de corte (o 60 seg. en la segunda ejecución) y el tiempo de ejecución del protocolo de hombre muerto. El primer experimento ha sido completado en unos 27 minutos mientras que la segunda ejecución se completó en 53 minutos aproximadamente. La ejecución de los experimentos, que involucran un software con varios hilos trabajando simultáneamente, se ha realizado en un ordenador con doble núcleo a 1.2 Ghz y 8Gb de RAM, una configuración que no resulta muy potente en la actualidad. Por esta razón, es muy probable que los tiempos de ejecución del protocolo de hombre muerto puedan ser mejorados utilizando un equipo más potente. No obstante, como queremos poner el sistema a prueba en condiciones adversas, resulta adecuado maximizar los tiempos de ejecución debidos al hardware para comprobar si aún en estas condiciones los tiempos son adecuados (en este caso, además, hay que tener en cuenta que tanto la simulación del software del vehículo como el software del centro de control y del transductor para la adquisición de datos, es decir, el marco experimental completo, se ejecutan sobre el mismo hardware). En nuestro entorno experimental, podemos concluir que a pesar del hardware utilizado para la ejecución de los experimentos, resulta razonable que el mecanismo de reconexión se realice en un tiempo inferior a un segundo, por lo que los resultados temporales del experimento han sido favorables en ambas ejecuciones. Además, se ha probado que, a pesar de someter al sistema a la situación crítica que suponen los cortes de red continuos en intervalos de 30 a 60 segundos durante periodos de 27 a 53 minutos, tanto la infraestructura como los mecanismos de respuesta ante caídas de la red son lo

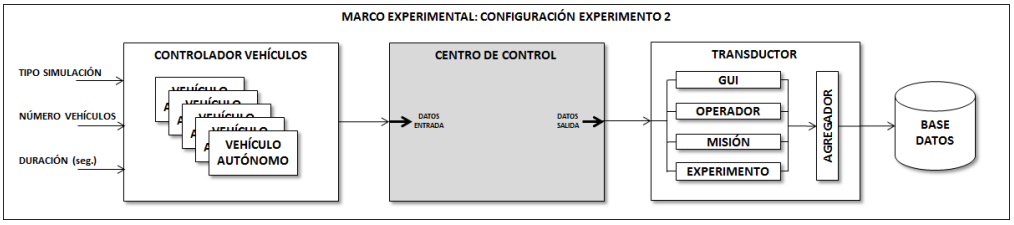


Figura 3.8: Configuración del Marco Experimental 2

suficientemente robustos para aguantar este tipo de problemas técnicos durante largos períodos de tiempo.

3.2.2. Experimento de Integración de Vehículos en Tiempo de Ejecución

La incorporación de vehículos en tiempo de ejecución a la infraestructura de nuestro CCT es otra de las situaciones críticas que deben ser probadas antes de llevarlas a cabo en los experimentos de campo, donde el vehículo puede escapar de nuestro control e incluso resultar dañado si hay algún error grave en los mecanismos diseñados para su ingreso en el sistema. En este caso, y dado que ya se ha probado el mecanismo de seguridad que permitirá la parada de emergencia y reestablecimiento de comunicaciones de los vehículos si llegase el caso de quedar incomunicados, las pruebas se centrarán en el estudio de dos nuevos aspectos: 1) determinar cuántos vehículos pueden integrarse en el CCT sin que haya fallos en la supervisión de la misión (retardos, pérdida de datos, etc.), y 2) realizar pruebas intensivas de integración de vehículos para comprobar que no se producen fallos en ninguno de los módulos software ni en los mecanismos implicados en esta labor.

3.2.2.1. Diseño del Marco Experimental 2

El marco experimental diseñado para estas pruebas puede verse en la Figura 3.8. Está formado por un bloque generador (control de vehículos), el sistema bajo estudio (el centro de control de tierra), y el bloque transductor estándar utilizado en todos los experimentos realizados por medio de marco experimental.

Control de vehículos es el módulo encargado de simular el desarrollo de una misión en la que uno o varios vehículos serán introducidos y/o extraídos del sistema. Su comportamiento puede configurarse mediante 3 variables: 1) tipo de simulación, 2) el número de vehículos y 3) la duración de la misión. Mediante la primera variable se determina si en el experimento, a) se introducen sucesivamente diferentes vehículos (hasta alcanzar el número indicado en la segunda variable y comprobando entre dos inclusiones sucesivas, que los datos de los vehículos son recibidos correctamente en el CCT), o si b) se simula sucesivamente la entrada y salida de un vehículo en el sistema. La segunda variable, utilizada solo para el primer tipo de simulación, indica el número de vehículos que deben crearse y conectarse al CCT. La tercera, utilizada también sólo en el primer tipo de simulación, indica cuánto tiempo debe mantenerse el sistema funcionando con el número total de vehículos para poder determinar si todo funciona correctamente. Ademas, es importante indicar que el controlador de vehículos utiliza el módulo de vehículos autónomos creado para el experimento 1 (y que puede verse en la Figura 3.5) para crear tantos vehículos simulados como sea necesario y para extraerlos del sistema cuando el experimento así lo requiera. El protocolo para integrar un nuevo vehículo es el explicado en la Sección 3.1.3, con la salvedad de que la aprobación para la inclusión del vehículo en el sistema ha sido automatizada (en lugar de ser llevada a cabo por parte del operador). Por simplicidad, por cada vehículo introducido en el sistema, en el CCT sólo se representa el icono que muestra su posición en el mapa y un elemento gráfico que muestra su orientación. Esto nos permite introducir un mayor número de vehículos (ya que no hay espacio físico para introducir más elementos gráficos por vehículo) y lograr una ejecución más rápida de cada una de las simulaciones.

Transductor: es el utilizado en el experimento 1, ya que este ha sido diseñado e implementado de forma estándar para que pueda ser utilizado en cualquiera de los experimentos realizados en el contexto de nuestro marco experimental. En este caso se guarda información relacionada con la canti-

dad de vehículos integrados en el sistema, las repeticiones del experimento y los tiempos de ejecución.

3.2.2.2. Diseño de los Experimentos

Para evaluar la robustez de la infraestructura de nuestro CCT adaptativo ante la integración (inclusión y eliminación) sucesiva de vehículos, se han diseñado dos bloques de experimentos en los que se utilizará el marco experimental descrito anteriormente.

En primer lugar se diseña un experimento para determinar el número máximo de vehículos que se pueden incluir en el sistema antes de que se produzca un error. Este experimento se configura solicitando que el marco experimental introduzca (sin eliminar) un número muy grande de vehículos (1000) y dejando que éste itere hasta que el software capture una excepción (error del sistema). Para las limitaciones del hardware (PC con Windows 8.1 y un microprocesador de doble núcleo 1.2GHz con 8Gb de RAM) y del software (memoria asignada a la maquina virtual Java) disponible, se ha determinado que el número máximo de vehículos que se pueden introducir con éxito en el sistema es 413 (número bastante elevado que permite mostrar que el entorno del CCT puede funcionar con un enjambre de vehículos aunque, como ya se ha mencionado, la funcionalidad para la que se ha diseñado el interfaz gráfico es para un número más pequeño de vehículos que cooperan para realizar una misión), ya que al intentar introducir un nuevo vehículo se produce un error de desbordamiento de memoria. Sin embargo, hay que tener en cuenta que la infraestructura no sólo debe mantener un canal de entrada/salida abierto por cada vehículo, si no que además debe reservar memoria y espacio en el GUI para los gráficos, hecho que limitará notablemente el número de vehículos que se pueden monitorizar realmente. De hecho, se realizó una prueba del funcionamiento de CCT con los 413 vehículos simulados, observándose durante la misma fallos que podrían afectar a la supervisión de una misión real, como amplios retardos en el refresco de la imagen o la imposibilidad de incluir (sin superponerlos en la interfaz gráfica) la cantidad de datos requerida para esos vehículos. Teniendo en cuenta estos aspectos (y

la incapacidad del operador de realizar tareas relacionadas con un número de vehículos tan alto), un tope de vehículos razonable para la supervisión con un solo operador sería 20 vehículos. Aún asi, es útil determinar el número máximo de vehículos a incluir en el sistema antes de que se produzca un fallo software, ya que, aunque esta situación no se dará en los experimentos reales, permite incluir mecanismos de reacción ante este tipo de fallo (mostrando un aviso al operador para que no acepte la integración de un vehículo en el sistema cuando se haya superado el número máximo considerado adecuado para llevar a cabo la misión).

Tras estas pruebas iniciales para determinar el número máximo de vehículos que puede incluirse sin fallos ni retardos en la visualización del CCT, se lleva a cabo el diseño de dos experimentos de repetitividad ante la integración y extracción de vehículos en el sistema.

El primer experimento de repetividad consiste en la integración de 20 vehículos secuencialmente en un periodo muy corto de tiempo. La integración se realiza de forma secuencial debido a que en las misiones reales es muy difícil que se de la integración simultánea de dos vehículos en un CCT y, si esto ocurriese, el operador tendría que aceptar uno de ellos y después el otro, ya que cada uno generaría una petición de conexión independiente. Si dos operadores en dos CCTs integrados en nuestra infraestructura distribuida tuviesen la oportunidad de aceptar la supervisión de un vehículo, éste pasaría a estar bajo control de aquel que aceptase su integración en primer lugar (lo que se notifica además al otro operador mediante un aviso en su interfaz gráfico). Se considera la integración secuencial de 20 vehículos, ya que como se ha mencionado anteriormente, es un tope razonablemente elevado para que un único operador intente supervisarlos desde un CCT. De hecho, para misiones en las que un operador deba monitorizar un número de vehículos superior, sería necesario diseñar un CCT diferente, que solo mostrase la información de ciertos vehículos o que realizase automáticamente la distribución de la carga de trabajo entre varios operadores, lo que no figura entre los objetivos de estos experimentos. Además para poner a prueba el funcionamiento del CCT, tras integrar los 20 vehículos, el programa se debe mantener en funcionamiento durante el tiempo suficiente para comprobar que no se producen

errores en el software, ni retardos o pérdidas en la muestra de información por parte del CCT. En este caso se ha considerado que 60 segundos es un tiempo suficiente para este objetivo. Tras los 60 segundos, el controlador de vehículos extraerá del sistema los 20 vehículos y volverá a comenzar el proceso de integración, y así sucesivamente hasta llegar a las 50 iteraciones contempladas en este experimento. Con esta prueba se determinará la robustez de la infraestructura ante fases extremas de integración y extracción de vehículos a una velocidad mucho más alta de lo que se daría en una misión real, probando que no se produce ninguna degradación de la infraestructura adaptativa al introducir y extraer grandes cantidades de vehículos que pueda acabar provocando errores en el CCT, o pérdidas en los paquetes de datos que los vehículos envían al interfaz del CCT para ser supervisados.

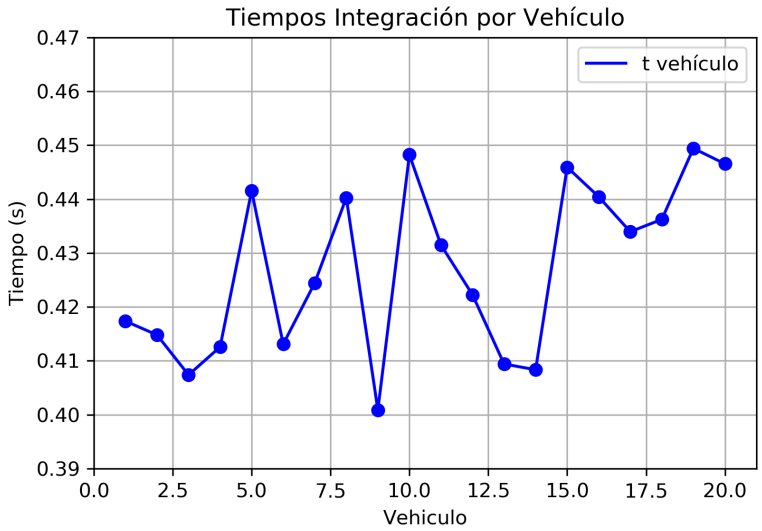
Con el segundo bloque de experimentos de repetitividad, se desea determinar la capacidad de la infraestructura adaptativa para introducir y extraer un vehículo en repetidas ocasiones sin que se produzcan errores que puedan afectar a la supervisión de la misión, como son la degradación de la infraestructura que pueda impedir la ejecución del software, o algún tipo de retardo o pérdida en los paquetes de datos enviados. Para ello se comenzará el experimento arrancando dos vehículos autónomos e integrándolos en el sistema, que comenzará a mostrarlos en el CCT por medio de su interfaz gráfico (como siempre, cada vehículo enviará un paquete de datos de telemetría cada segundo). A continuación, se comenzará a introducir y extraer un tercer vehículo en 10 ocasiones consecutivas mientras el CCT continúa en ejecución. Se realizarán tres ejecuciones del experimento que consistirán en introducir/extraer el vehículo en intervalos de 10 segundos mientras se observa que no se producen errores en el interfaz gráfico del CCT. Como en el bloque anterior, el objetivo de este experimento es probar la robustez y eficacia de nuestro CCT ante la situación crítica que supone continuar con la supervisión de dos vehículos autónomos mientras la infraestructura adaptativa responde a la introducción y extracción sucesiva de un tercer vehículo en un periodo de tiempo corto.

Tabla 3.2: Resultados del Primer Experimento de Repetitividad del Marco Experimental 2

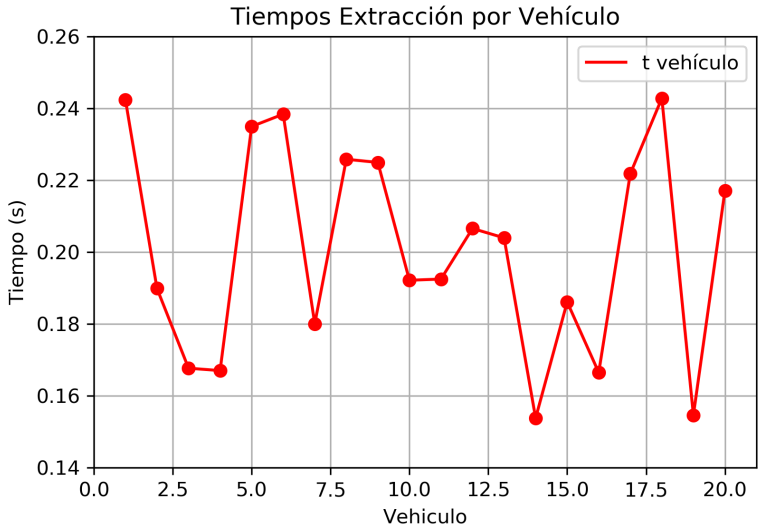
# IT	VEHÍCULOS	TMEDIO (seg.)	σ	TPRUEBA (seg.)	TMEDIO TOTAL (seg.)	MENSAJES RECIBIDOS	TTOTAL (seg.)
50	20	8,479	0,295	60	72,467	100%	3623,394 (~60 min.)
		3,988	0,537				

3.2.2.3. Resultados

El primer bloque de experimentos de repetividad tiene como objetivo determinar si el sistema es robusto y eficaz a la hora de introducir y eliminar grandes cantidades de vehículos en periodos de tiempo cortos. En la Tabla 3.2, pueden verse los tiempos medios de ejecución de las 20 iteraciones para integrar (cifra superior) y extraer (cifra inferior) cada uno de los 20 vehículos, así como las desviaciones estándar de dichos tiempos. El tiempo medio de integración de 20 vehículos está en 8.479 segundos y el de extracción en 3.988 segundos, con unas desviaciones estándar de 0.295 y 0.537 respectivamente. A estos tiempos se han añadido a los 60 segundos en los que se realizan las comprobaciones de ausencia de errores (TPRUEBA) y pérdida o retardo en la recepción de paquetes, deja un tiempo medio total de 72,467 segundos por ejecución de cada iteración. En la penúltima columna se muestra el porcentaje de mensajes recibidos respecto al total de los que fueron enviados. Finalmente, en la última columna podemos ver el tiempo total que se tardó en ejecutar el experimento, que duró unos 60 minutos (50 iteraciones de integración y extracción de 20 vehículos del sistema con 60 segundos de simulación intermedia para comprobar la ausencia de errores). Se han recopilado más datos sobre la ejecución de los experimentos en las Figuras 3.9a y 3.9b, en la que se encuentran, respectivamente, los tiempos de introducción y extracción de cada uno de los vehículos en una iteración del experimento. Además en las Figuras 3.10a y 3.10b, podemos observar un resumen de los tiempos que la simulación tardó en introducir y extraer respectivamente los 20 vehículos en cada una de las 50 iteraciones.



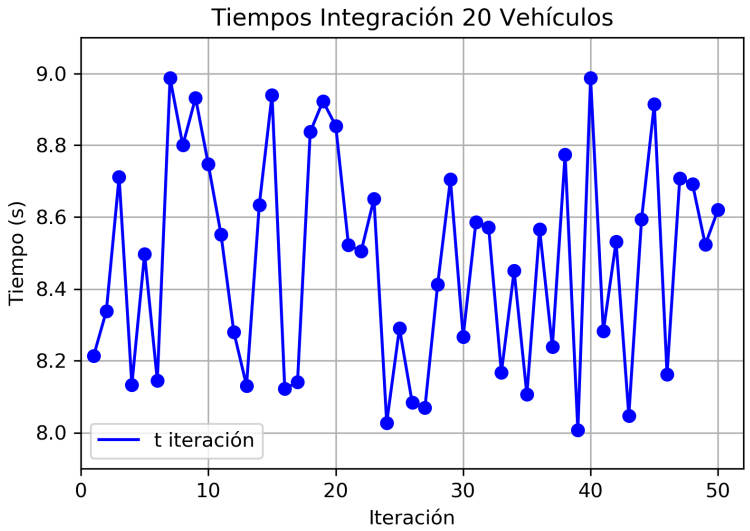
(a) Tiempos de Integración por Vehículo en una Iteración



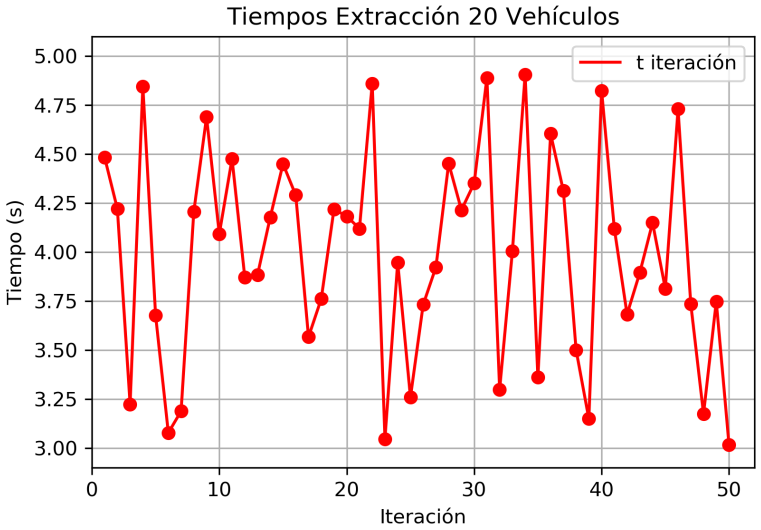
(b) Tiempos de Extracción por Vehículo en una Iteración

Figura 3.9: Tiempos por Vehículo en una Iteración del Primer Experimento de Repetitivad del Marco Experimental 2

Tras la ejecución del primer bloque de experimentos, se determina que la infraestructura es capaz de introducir y extraer un alto número de vehículos



(a) Tiempos de Integración de 20 Vehículos por Iteración



(b) Tiempos de Extracción de 20 Vehículos por Iteración

Figura 3.10: Tiempos de Inclusión y Extracción de 20 Vehículos por Cada una de las 50 Ejecuciones del Primer Experimento de Repetitivad del Marco Experimental 2

en el CCT sin que se produzca degradación en la infraestructura, ya que todos los módulos que es necesario replicar para introducir un vehículo tanto a nivel estructural como gráfico se crean y destruyen sin dejar restos que ocasionen fallos. Además, se observa que no se producen retardos o pérdida de los datos a supervisar, ya que el número total de paquetes enviados es igual al total de datos recibidos, por lo que la eficacia es del 100 %. Dado que para estas simulaciones se considera una red de comunicaciones ideal, y se realiza la conexión por Sockets en modo local, la red no produce errores en las comunicaciones y la pérdida sólo podría ser ocasionada por defectos del software del CCT. Al igual que en las pruebas del Marco Experimental 1, no es nuestro objetivo probar la estabilidad de una red de comunicaciones que puede depender de muchos factores, sino probar la robustez de nuestro software ante cambios en el número de vehículos monitorizados en tiempo de ejecución. Por lo tanto, en esta ocasión se considera la ausencia de pérdida de mensajes recibidos y mostrados en el CCT como una prueba de robustez y eficacia ante dicha situación crítica, ya que la infraestructura adaptativa ha sido capaz de crear cada uno de los nuevos canales de comunicación de forma automática en un tiempo razonable y mantenerlos estables durante la ejecución de la misión. Además, los tiempos empleados por el sistema para introducir y extraer un vehículo (o 20 en este caso) son lo suficientemente reducidos como para que el operador no sufra distracciones ni pierda datos relevantes en la monitorización, ya que en todos los casos se sitúa en valores próximos al medio segundo la etapa de integración y valores inferiores, en torno a un tercio de segundo, la extracción. Estos tiempos, además, podrían ser acortados en un equipo más potente, lo que mejoraría la experiencia del operador. En caso de verse obligado a ejecutar el CCT en un equipo aún más limitado, los tiempos podrían aumentar, pero aún se situarían dentro de límites aceptables para la supervisión de una misión ya que el margen es amplio.

A continuación se realiza el segundo bloque de experimentos de repetitividad, que tiene como objetivo realizar pruebas ante la inclusión y extracción de un vehículo durante la ejecución de una misión, y comprobar que no se producen errores en la infraestructura ni en la recepción y muestra de mensajes cuando se

Tabla 3.3: Resultados del Segundo Tipo de Experimentos de Repetitividad del Marco Experimental 2

# IT	VEHÍCULOS	REPS.	TIN (seg.)	σ_{IN}	TOUT (seg.)	σ_{OUT}	TEJECUCIÓN (seg.)	TTOTAL (seg.)
1	2+1	10	4,284	0,014	2,222	0,039	200	206,506
2	2+1	10	4,274	0,012	2,136	0,038	200	206,41
3	2+1	10	4,257	0,014	2,2	0,042	200	206,457

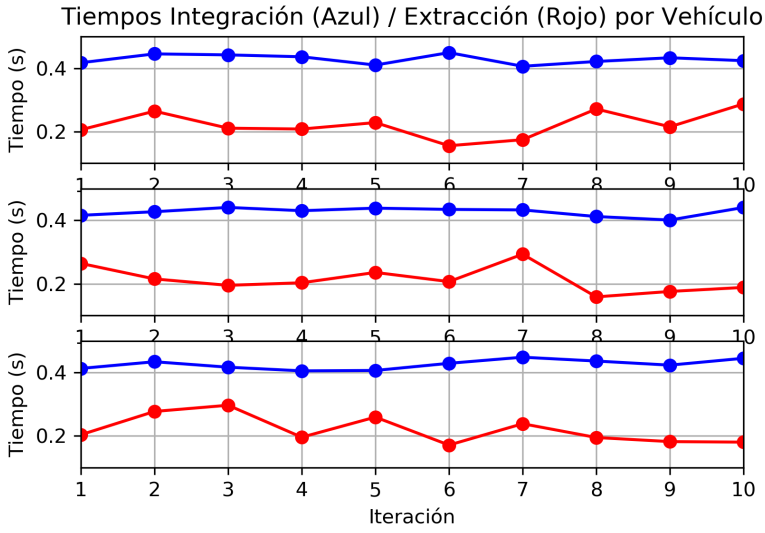


Figura 3.11: Tiempos de cada una de las 10 Etapas de Inclusión y Extracción de un Vehículo para las 3 Iteraciones del Experimento

está realizando una misión con varios vehículos autónomos (dos en este caso) y un tercero solicita entrar y salir de la misión repetidas veces en un corto periodo de tiempo. Una vez más se prueba en simulación una situación que en una misión real no se daría de forma tan rápida ni repetitiva.

En la Tabla 3.3 y en la Figura 3.11 se resumen los datos de cada iteración de las 3 ejecuciones del segundo bloque del marco experimental 2. En este bloque de experimentos se cuenta con 2 vehículos enviando datos simultáneamente mientras un tercero se introduce y extrae 10 veces en el sistema, dejando un

margen de 10 segundos entre cada paso del proceso para comprobar que no hay pérdida ni retrasos en la recepción y muestra de datos. Los tiempos totales, para las 10 etapas de introducción y extracción de los vehículos, son relativamente cortos (unos 4 segundos en la etapa de introducción y unos 2 segundos en la de extracción), lo cual nos permite incluir un vehículo rápidamente (en torno a 0.4 seg.) para colaborar en tareas de una misión real o extraerlo de la misma forma (en torno a 0.2 seg.) si ya no es útil o esta generando algún problema. Podemos observar también la desviación estándar de cada uno de estos valores es muy pequeña. El tiempo total de cada una de las ejecuciones se sitúa en torno a los 206 segundos, es decir, poco menos de tres minutos y medio, lo que supone en torno a un 3 % de aumento del tiempo del experimento debido a las diez repeticiones). Como en el experimento anterior, se está poniendo a prueba el sistema en un ordenador bastante limitado, por lo que se maximizan los tiempos recogidos en la tabla. Además, como se ha comentado anteriormente, los tiempos pueden mejorar con un equipo más potente (hecho que no conllevaría ninguna dificultad añadida) o empeorar con uno más limitado (aunque los márgenes son lo suficientemente amplios para permitirlo). Por último, y para complementar la información de la Tabla 3.3, en la Figura 3.11 se muestran los tiempos de las 10 etapas de inclusión (en azul) y extracción (en rojo) de las tres ejecuciones del experimento.

Tras las tres ejecuciones del experimento, no se ha producido ningún error en el CCT ni en los mecanismos de la infraestructura que se encargan de incluir y extraer el nuevo vehículo de la misión. Una vez más se considera la ausencia de retardos o pérdidas de información como prueba de la consistencia del mecanismo encargado de adaptar el sistema a la rápida inclusión y extracción de nuevos vehículos en la infraestructura. Esto incluye al sistema de comunicaciones que se encarga de establecer el nuevo canal, mantenerlo para el envío de mensajes y destruirlo cuando el vehículo sale del sistema; al configurador dinámico que decide cómo configurar el CCT para mostrar los datos en su interfaz gráfico, y a los adaptadores intermedios que se encargan de la traducción de protocolos del vehículo y del encapsulamiento de mensajes. Además, todo este proceso se

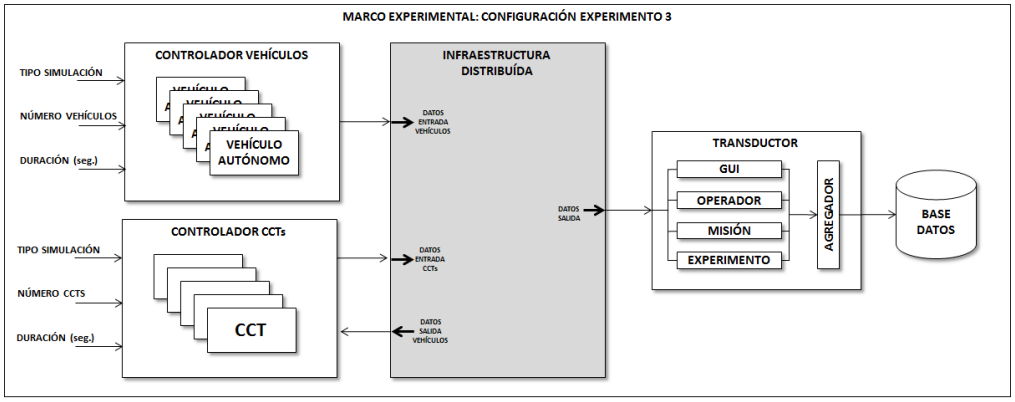


Figura 3.12: Configuración del Marco Experimental 3

consigue llevar a cabo sin que afecte al resto de módulos (vehículos, CCTs, etc.) en funcionamiento (gracias a la naturaleza desacoplada de la arquitectura) y, como hemos comentado anteriormente, sin que se observe degradación alguna en la infraestructura del CCT (que crea y elimina correctamente los módulos necesarios para introducir nuevos vehículos).

3.2.3. Experimentos de Integración de CCTs en Tiempo de Ejecución

El tercer punto crítico que debemos probar antes de realizar pruebas de campo es la integración de nuevos CCTs en tiempo de ejecución. En este caso los consideraremos sólo visualizadores, ya que servirán sólo para monitorizar la misión hasta que se realice la transferencia de competencias o delegación de tareas de un operador a otro explicada en la Sección 3.1.3. El diseño de experimentos para probar esta funcionalidad se ha realizado de forma análoga a la del segundo bloque, ya que consiste en incluir y extraer un módulo (en este caso el CCT) de la infraestructura mientras el sistema continua ejecutando una misión. Los objetivos en este caso son: 1) determinar si la infraestructura responde correctamente (manteniéndose estable y eficaz) a la agregación de un número alto de CCTs, y 2) realizar pruebas intensivas de integración/extracción de un CCT mientras otro continúa en ejecución.

3.2.3.1. Diseño del Marco Experimental 3

El marco experimental diseñado para este experimento se esquematiza en la Figura 3.12 y consta de un bloque generador formado por dos módulos (controlador de vehículos y controlador de CCTs), el bloque transductor que ya hemos visto en los dos marcos experimentales anteriores, y el sistema bajo estudio (en este caso la infraestructura adaptativa diseñada con la arquitectura adaptativa orientada a eventos). A continuación describiremos cada módulo con más detalle.

Control de vehículos, que funciona de forma similar al descrito en la Sección 3.2.2.1. Se encarga de crear las instancias de cada uno de los vehículos que se conectarán al sistema para ser monitorizados por los CCT que se integrarán en el sistema. Los vehículos, utilizados en los dos experimentos anteriores, funcionan como un vehículo autónomo que envía sus datos de telemetría y eventos referentes a sucesos relacionados con la misión.

Control de CCTs es el módulo encargado de realizar una labor similar al módulo de control de vehículos en el Marco Experimental 2. Para configurar el comportamiento de este modulo, se utilizan tres variables que indican el tipo de experimento, el número de CCTs que deben crearse y el tiempo que debe mantenerse funcionando el experimento para comprobar si es realizado con éxito. En los experimentos tipo 1, se introducirá en bloque un numero de CCTs que deberán mantenerse funcionando durante el período de tiempo estipulado, para a continuación ser extraídos. En el experimento tipo 2, se introducirá un CCT que se mantendrá en funcionamiento mientras un segundo CCT se incluye y extrae en repetidas ocasiones para comprobar si esto produce algún fallo en el sistema.

Transductor es el mismo que ha formado parte de los Marcos Experimentales 1 y 2. En este caso guarda datos sobre la cantidad de CCTs integrados en el sistema, repeticiones del experimento y tiempos de ejecución.

3.2.3.2. Diseño de los Experimentos

Se han realizado dos bloques de experimentos con el marco experimental diseñado para la integración de CCTs visualizadores en el sistema.

El primer bloque de experimentos consiste en pruebas de repetitividad para determinar la eficacia y robustez de la infraestructura distribuida del CCT. Consiste en la integración de un numero elevado de centros de control (10 en este caso) en un corto periodo de tiempo, tras lo cual el sistema se mantiene funcionando el tiempo necesario para comprobar que la visualización y entrega de paquetes de datos se realiza correctamente, para a continuación proceder con la extracción de los centros de control. Este proceso de inclusión en bloque de 10 centros de control, simulación y extracción, se realizará 50 veces consecutivas. El hardware utilizado es exactamente el mismo que en los experimentos anteriores, un PC con Windows 8.1 y un microprocesador de doble núcleo 1.2GHz con 8Gb de RAM. En el caso de los CCTs no vamos a realizar el experimento del número máximo que pueden ejecutarse simultáneamente ya que lo habitual es que no se tenga más de un centro de control por ordenador y, aunque si es habitual dividir la monitorización en varios visualizadores distribuidos en diferentes pantallas, el caso queda cubierto ya que durante el experimento se van a mantener en funcionamiento 10 centros de control simultáneamente. Por lo tanto, en este primer bloque de experimentos el objetivo es someter al sistema a la situación extrema que supone la inclusión y extracción de 10 centros de control en breves periodos de tiempo, con lo que podremos probar que la infraestructura crea y destruye las instancias de los módulos necesarios para la integración de los CCTs sin producir degradación en el sistema ni perder o sufrir retardos en la entrega o visualización de datos.

El segundo bloque de experimentos se ha diseñado de forma análoga al segundo bloque de experimentos de repetitividad del Marco Experimental 2. Con él se quiere determinar si la infraestructura adaptativa es capaz de dar soporte a la introducción y extracción continuada de CCTs, sin que esto produzca errores que afecten a la monitorización de la misión. El experimento constará de dos

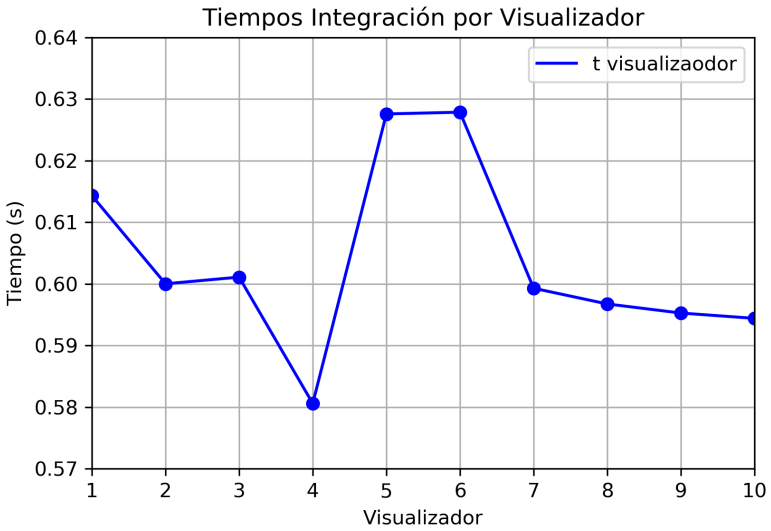
Tabla 3.4: Resultados del Primer Experimento del Marco Experimental 3

# IT	CCTs	TMEDIO (seg.)	σ	TPRUEBA (seg.)	TMEDIO TOTAL (seg.)	MENSAJES RECIBIDOS	TTOTAL (seg.)
50	10	6,062	0,125	60	69,708	100%	3485,414 (~58 min.)
		3,646	0,214				

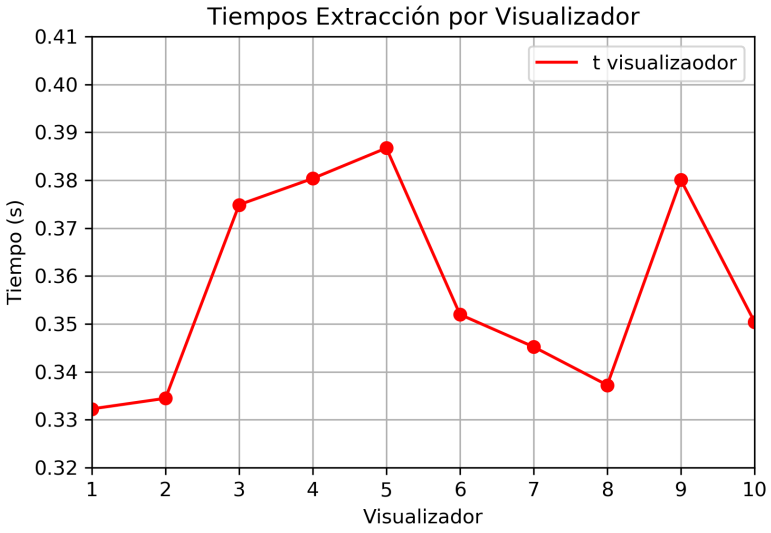
vehículos autónomos simulados por el módulo de control de vehículos, que los integrará en el sistema. Una vez introducidos los vehículos, se integrará un CCT que mantendrá su funcionamiento mientras el módulo de control introduce y extrae un CCT en 10 ocasiones sucesivas, con un intervalo de 10 segundos entre cada operación para que el sistema pueda comprobar si cada CCT incluido en el sistema ha recibido y mostrado todos los mensajes enviados. Esta ejecución se repetirá 3 veces. El objetivo de este experimento es probar la robustez y eficacia de nuestro CCT ante la situación crítica que supone la posible degradación del sistema si los CCTs integrados o extraidos no se crean o eliminan del sistema correctamente. Además, la visualización y recepción de datos en el CCT que permanece fijo también debe ser adecuada.

3.2.3.3. Resultados

El primer bloque de experimentos tiene como objeto determinar si el sistema es robusto y eficaz a la hora de introducir y eliminar grandes cantidades de CCTs en cortos períodos de tiempo. En la Tabla 3.4 podemos observar los tiempos medios de ejecución de las 10 integraciones (cifra superior) y extracciones (cifra inferior) de los CCTs, así como las desviaciones estándar de cada grupo de tiempos. El tiempo medio de integración es de unos 6.062 segundos y el de extracción de 3.646 segundos, lo que en conjunto con los 60 segundos que se mantendrá funcionando simultáneamente los 10 CCTs en cada una de las iteraciones, dará lugar a un total de 69.708 segundos de tiempo medio por iteración. El experimento concluirá, tras realizar las 50 iteraciones contempladas, en unos 58 minutos. También podemos encontrar en la penúltima columna el número



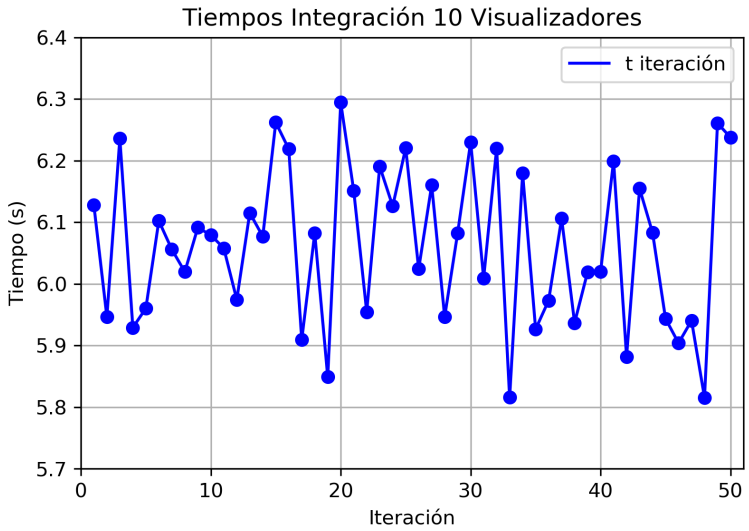
(a) Tiempos de Integración por Visualizador en una Iteración



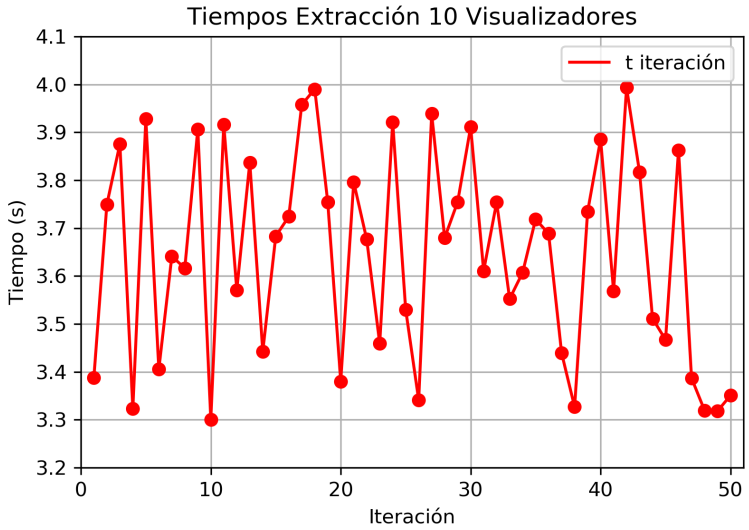
(b) Tiempos de Extracción por Visualizador en una Iteración

Figura 3.13: Tiempos por Visualizador en una Iteración del Primer Experimento del Marco Experimental 3

total de paquetes de datos que fueron recibidos correctamente, que asciende a su totalidad (100 %).



(a) Tiempos de Integración por Iteración



(b) Tiempos de Extracción por Iteración

Figura 3.14: Tiempos de Integración y Extracción de los 10 Visualizadores en cada una de las 50 Iteraciones del Primer Experimento del Marco Experimental 3

En las Figuras 3.13a y 3.13b se han resumido gráficamente los tiempos de inclusión y extracción de cada uno de los CCTs en el sistema durante una de las 50 iteraciones. Además en las Figuras 3.14a y 3.14b podemos observar el tiempo que la simulación tardó en introducir y extraer los 10 CCTs en cada una de las 50 iteraciones del experimento.

Tras la ejecución de este primer bloque de experimentos, se determina que la infraestructura adaptativa es capaz de integrar y extraer un número elevado de CCTs sin que se produzca degradación en la infraestructura, y que es capaz de instanciar y destruir correctamente todos los módulos necesarios para integrar y eliminar CCTs del sistema sin dejar restos que ocasionen fallos de funcionamiento. Además, no se observan errores en la visualización de elementos gráficos o muestra de alarmas, retardos en el refresco de pantalla o pérdida de datos, ya que el número total de paquetes enviados es igual al total de datos recibidos, por lo que la eficacia es del 100 %. Al igual que en los experimentos anteriores, no se produce ninguna pérdida de datos derivada de un fallo de comunicaciones, ya que se ha considerado un sistema de comunicaciones ideal implementado con el sistema de Sockets local del equipo en el que se ejecuta el experimento. Por esta razón, podemos asegurar, como en las pruebas realizadas en los Marcos Experimentales 1 y 2, que los elementos propios de nuestra infraestructura (el canal gestor de eventos que se encarga de la entrega de mensajes, así como los mecanismos que integran los nuevos CCTs en el sistema por medio del configurador dinámico) no han perdido ni dejado de mostrar ninguno de los mensajes enviados a partir de la conexión de cada uno de los CCTs. Esto nos ha permitido demostrar la robustez y eficacia del sistema distribuido con varios CCTs ante la situación crítica a la que se le ha sometido en este experimento, ya que la infraestructura adaptativa ha sido capaz de reconfigurarse en repetidas ocasiones durante el largo período de tiempo que ha durado la ejecución del experimento. Además, se ha comprobado que los tiempos empleados por el sistema para la integración y extracción de CCTs son lo suficientemente reducidos como para poder incorporar a un operador al sistema en cualquier momento si fuese necesario, pudiendo realizarse esta operación sin poner en peligro la misión ni perder

Tabla 3.5: Resultados del Segundo Experimento del Marco Experimental 3

# IT	CCTs	REPS.	TIN (seg.)	σ_{IN}	TOUT (seg.)	σ_{OUT}	TEJECUCIÓN (seg.)	TTOTAL (seg.)
1	1+1	10	6,048	0,016	4,594	0,023	200	210,642
2	1+1	10	6,050	0,012	4,661	0,027	200	210,711
3	1+1	10	6,076	0,011	4,555	0,032	200	210,631

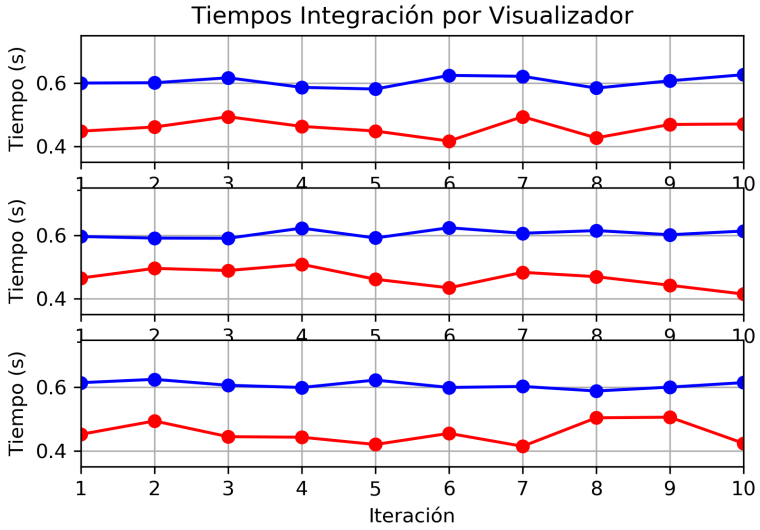


Figura 3.15: Tiempos de cada una de las 10 Etapas de Inclusión y Extracción de un Visualizador para las 3 Iteraciones del Experimento

datos de supervisión que pudiesen ser relevantes para llevarla a cabo. Debido a las limitaciones del equipo en el que se ha realizado el experimento, es razonable pensar que los tiempos podrían mejorar en un equipo más potente. Además en el caso peor de que se disponga de un equipo menos potente, con los tiempos observados existe un margen suficiente como para que la operación pueda realizarse con éxito.

A continuación se realiza el segundo bloque de experimentos que consiste en realizar pruebas de repetitividad ante la inclusión y extracción de un CCT durante la ejecución de una misión en la que ya hay otro CCT activo. Estos

experimentos tienen como objeto comprobar que no se producen errores en la infraestructura del sistema ni en la recepción y muestra de datos durante la realización de una misión en la que se introduce y extrae repetidas veces un CCT. Con estas pruebas se someterá al sistema a una situación límite que en condiciones normales no se dará en una misión real (ni por número de repeticiones ni por velocidad de integración de los CCTs).

En la Tabla 3.5 se encuentra un resumen de los datos de cada una de las ejecuciones de las tres iteraciones de este experimento. Podemos observar los tiempos medios de las 10 etapas de inclusión y extracción del nuevo CCT que se encuentran, respectivamente, en torno a los 6 y 4 segundos (esto hace que el tiempo de una única inclusión y extracción del CCT sea, aproximadamente, de 0.6 y 0.4 segundos). A la derecha de estos tiempos se encuentran las distribuciones estándar que nos indican la baja dispersión de estos tiempos respecto a la media. Tras cada inclusión y extracción del CCT se dejará funcionando al sistema 10 segundos para comprobar que funciona correctamente, teniendo una duración total de unos 210 segundos (poco más de tres minutos y medio) en cada una de las 3 iteraciones del experimento. Como siempre, asumimos unas comunicaciones ideales implementadas por medio de Sockets locales en un equipo con doble núcleo a 1.2Ghz y 8Gb de RAM. Al igual que en el resto de experimentos, consideramos que los tiempos podrían mejorar si la ejecución se realiza en un equipo más potente, pero es útil considerar el caso de un equipo poco potente para comprobar que aún así los tiempos son lo suficientemente cortos como para que el operador no sufra distracciones ni pérdida de datos en su supervisión. Además el margen de tiempo es lo suficientemente amplio como para que no haya problemas si el equipo empeora un poco. Finalmente, indicar que estas pruebas nos han ayudado a demostrar que el sistema es lo suficientemente robusto y rápido como para incorporar, en un instante dado, a un nuevo operador con un nuevo CCT en un corto margen de tiempo, de forma que este pueda asistir a otro operador que se sienta desbordado por el elevado número tareas que debe realizar durante una fase determinada de la misión.

Para complementar los resultados anteriores, en la Figura 3.15 se representan los tiempos de inclusión (en azul) y extracción (en rojo) de los 10 CCTs en cada una de las tres iteraciones del experimento.

Finalmente, cabe indicar que tras las tres ejecuciones del experimento, no se ha producido ningún error en el centro de control ni en los mecanismos de la infraestructura que se encargan de incluir y extraer el nuevo visualizador del CCT de la misión. Una vez más, se considera la ausencia de retardos o pérdidas de información como prueba de la consistencia del mecanismo encargado de adaptar el sistema a la rápida inclusión y extracción de nuevos visores del CCT en la infraestructura. Además, todo este proceso se consigue llevar a cabo sin que afecte al resto de módulos en funcionamiento gracias a la naturaleza desacoplada de la arquitectura. Como hemos comentado anteriormente, tampoco se observa degradación en la infraestructura del CCT, ya que los módulos que se crean y destruyen para albergar nuevos visualizadores se eliminan correctamente, sin ocasionar errores en el funcionamiento del CCT adaptativo.

3.3. Caso de Uso: Adaptación del Software a Nivel Estructural

El entorno de desarrollo adaptativo dirigido a eventos diseñado para la implementación de CCTs permite la creación de infraestructuras distribuidas adaptativas para monitorizar/controlar múltiples dispositivos heterogéneos (vehículos, sensores, etc.) que pueden ser reasignados si cambian las tareas o el entorno en el que se esta desarrollando la misión.

A continuación describiremos un caso de uso ilustrativo que muestra los mecanismos más importantes que se utilizan durante una de las misiones llevadas a cabo con nuestro CCT adaptativo en el ámbito del proyecto SALACOM (Sistema Autónomo de Localización y Actuación ante Contaminantes en el Mar). La misión se ha realizado donde se llevan a cabo la mayor parte de las pruebas de este proyecto, el pantano de El Atazar, que proporciona una superficie de agua lo suficientemente amplia como para realizar múltiples maniobras con el equipo

de vehículos disponible. En esta misión se cuenta con un vehículo autónomo de superficie marina (Unmanned Surface Vehicle) que denominaremos USV1. Este vehículo debe realizar una serie de maniobras pre-planificadas (por ejemplo, una lemniscata o una circunferencia) descritas en los trabajos de de la Cruz et al. (2014, 2015, 2016). Una vez estas trayectorias son completadas satisfactoriamente y el USV1 alcanza un punto de paso (waypoint) determinado, un modelo simulado de otro vehículo de superficie (USV2) debe unirse a la infraestructura software para realizar una maniobra conjunta lider-seguidor. Los objetivos principales de este experimento son: 1) verificar que el USV1 completa correctamente las maniobras pre-planificadas sobre el agua, 2) verificar que el USV1 (seguidor) sigue correctamente a USV2 (líder) en la maniobra lider-seguidor, 3) comprobar que la infraestructura adaptativa responde adecuadamente a las exigencias de la misión durante el desarrollo de todo el experimento, y 4) comprobar, en las pruebas de campo, que la transición entre los escenarios correspondientes a la misión con un vehículo involucrado y 2 vehículos involucrados es viable. Adicionalmente, se contemplan varios objetivos secundarios como testear la alarma de colisión que se activa cuando USV1 y USV2 están demasiado cerca, confirmar que un solo operador es capaz de hacer frente a las tareas de monitorización y control de ambos vehículos y, sobre todo, completar el primer conjunto de maniobras con dos vehículos de superficie aprovechando las ventajas de la simulación distribuida en tiempo real, que puede ser incluida en el sistema gracias al diseño basado en nuestra arquitectura adaptativa dirigida a eventos. Esta característica nos permite realizar este conjunto de maniobras sin que se de el peligro de una colisión entre los USVs, que podría ser fatal para el hardware si alguno de los vehículos se hundiese.

El experimento, como podemos observar en la Figura 3.16, ha sido configurado con un vehículo autónomo real (USV1): un modelo a escala de 4 metros de longitud de un barco real diseñado por el Canal de Experiencias Hidrodinámicas de El Pardo (CEHIPAR) y un modelo que simula en tiempo real y en una computadora remota el comportamiento de un barco real (USV2). Además, por razones de seguridad, el USV2 ha sido configurado como vehículo líder y el

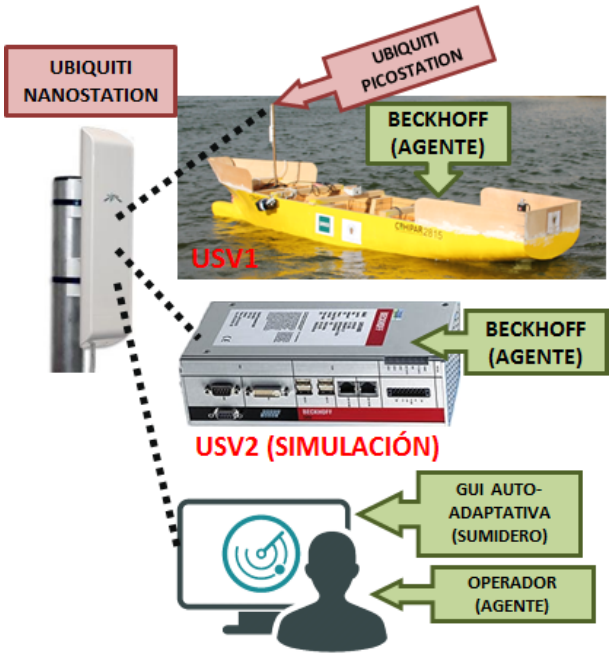


Figura 3.16: Configuración del Hardware del Experimento del Caso de Uso 1

USV1 como vehículo seguidor durante la realización de la maniobra coordinada. El USV1 es controlado y el modelo de USV2 simulado en un Beckhoff C6920, un PC industrial con TwinCAT, que es un sistema de tiempo real con un sistema operativo Windows modificado al efecto (similar al que se utiliza en el barco real USV1). Para la gestión de la telemetría, el USV1 cuenta con una IMU que incorpora una unidad GPS. El sistema de comunicaciones está formado por una antena Wi-Fi Ubiquiti PicoStation M en cada uno de los vehículos (aunque el vehículo simulado podría estar también conectado por medio de un cable al sistema) y una antena Ubiquiti NanoStation M2 como punto de acceso global, con un rango máximo de comunicaciones de 1.5 a 2 Km.

Al comenzar el experimento de campo, el USV1 es desplegado sobre la superficie de agua para la fase 1: planificación y seguimiento de trayectoria en solitario. El CCT adaptativo se auto-configura, tal y como se esquematiza en la Figura 3.17, con una infraestructura simple formada tan solo por un GUI, un vehículo autónomo (USV1), y el canal gestor de eventos que gestiona las comu-

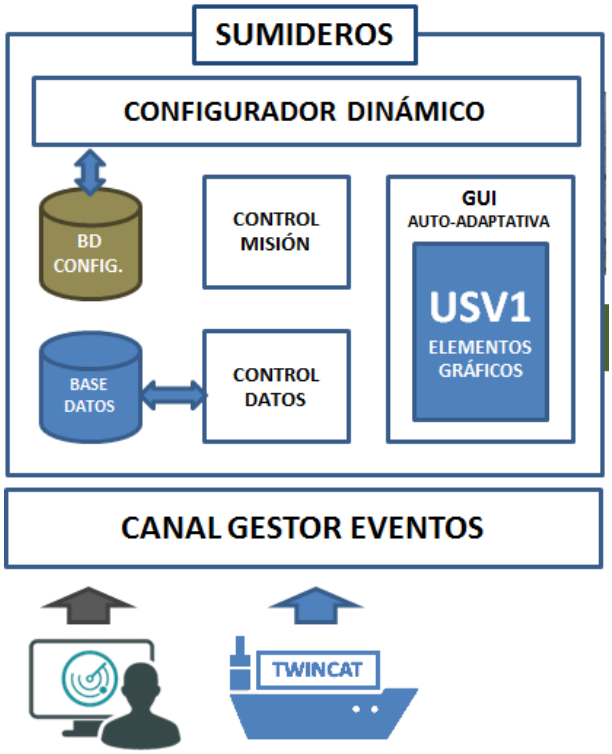


Figura 3.17: Configuración del Experimento Software - Fase 1

nicaciones y los sumideros necesarios para la gestión de datos y mecánicas de la misión.

El interfaz de usuario incluye un planificador, un controlador y todos los elementos gráficos (posición en el mapa, puntos de paso, telemetría, gráficas de datos, etc.) necesarios para que un solo operador monitorice y controle el vehículo autónomo. El canal gestor de eventos configura la conexión por Sockets TCP/IP entre los agentes (USV1, en este caso) y los sumideros (GUI, módulo de control de datos, módulo de control de misión, etc.).

En primer lugar, la capa de comunicaciones perteneciente a nuestro entorno de desarrollo instalada en el vehículo autónomo solicita conectarse al sistema comenzando la negociación de conexión que se envía al canal gestor de eventos visto en la Sección 3.1.1. El Canal verifica la conexión y abre un canal de comunicaciones TCP/IP entre el Agente (USV1) y los sumideros, y a partir de ese

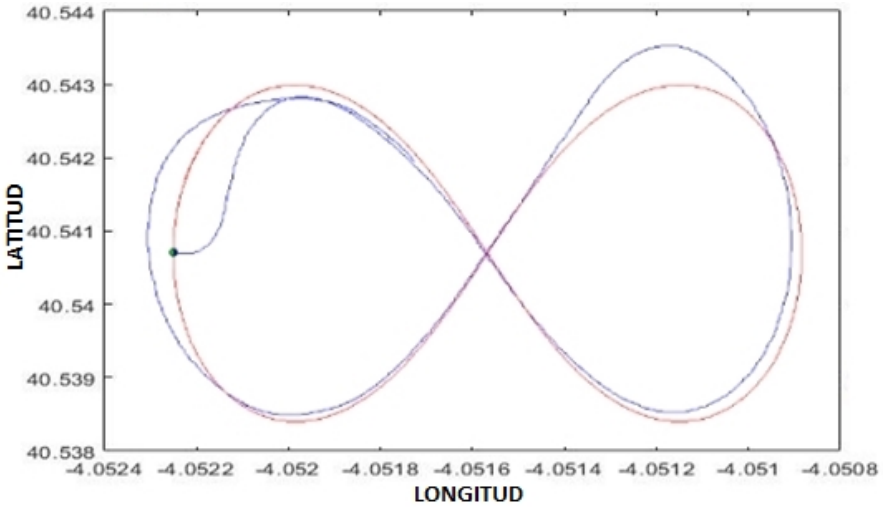


Figura 3.18: Maniobra Lemniscata realizada por USV1 durante Fase 1

momento se encarga de mantenerlo de forma segura y robusta. En este momento el Agente comienza a enviar mensajes a través del canal de comunicaciones, que los procesa, decodifica y encapsula la información extraída de los mismos en una estructura de Evento que cualquier sumidero es capaz de entender. Los eventos son difundidos por toda la infraestructura gracias al patrón Observador (descrito anteriormente). Al llegar a los sumideros, éstos serán capaces de distinguir los eventos ante los que deben reaccionar.

Durante la fase 1, el operador supervisa al USV1 para que realice varias maniobras que han sido introducidas a través del planificador incluido en el interfaz de usuario. Si hubiese que enviar alguna consigna al vehículo, podría hacerse a través del controlador integrado en el GUI del centro de control. En la Figura 3.18 podemos observar una de las trayectorias paramétricas completadas con éxito durante la fase 1, la lemniscata. La línea roja representa la trayectoria planificada y la línea azul es la trayectoria real completada por el USV1.

Cuando la fase 1 ha terminado y las trayectorias del USV1 en solitario han sido completadas, el operador introduce un punto de paso prefijado en el comandador y el USV1 se dirige hacia él. Una vez lo alcanza, comienza la fase 2. En este momento, el USV2 arranca y su capa de comunicaciones envía la solicitud

de conexión al canal gestor de eventos (ver esquema de negociación de conexión en la Figura 3.3), que establece un canal de comunicaciones agentes-sumideros de forma similar a la vista en la fase 1 con el USV1. Una vez que se ha establecido el canal de comunicaciones, el USV2 solicita su incorporación a la misión y la infraestructura adaptativa envía un ICR que necesita ser validado por el operador. Cuando el operador da permiso, el configurador dinámico accede a las reglas de cambio almacenadas en la base de datos de configuración y realiza los cambios necesarios en el software adaptativo. En este caso, el configurador añade en el GUI los elementos gráficos relacionados con el USV2 (el icono que representa al USV2 en el mapa, su telemetría, sus gráficas de trayectoria, puntos de paso, ruta, etc.), los datos y controladores para la gestión del vehículo durante la misión, los módulos de almacenamiento y gestión de datos generados durante la misión y las comprobaciones que deberán hacerse durante su desarrollo. Un esquema básico de la nueva configuración se puede observar en la Figura 3.19.

Durante la fase 2 se llevan a cabo las comprobaciones planeadas para este experimento de campo. En primer lugar, probamos satisfactoriamente la maniobra líder-seguidor donde el USV1, físicamente desplegado sobre el agua, hace las veces de seguidor y el USV2, simulado en tiempo real, actúa como líder. Esto nos permite llevar a cabo la maniobra sin ningún peligro ya que uno de ellos no está físicamente en el agua. Además, esta simulación nos ayuda a comprobar que el software adaptativo es capaz de reconfigurar su estructura en tiempo de ejecución sin que se produzca ningún error, ya que los mecanismos utilizados por el configurador dinámico son exactamente los mismos que se habrían usado con el USV2 físico. Además, utilizando un USV2 simulado, hemos sido capaces de comprobar una serie de alarmas (velocidad alta, peligro de colisión) sin poner en peligro el hardware de los barcos autónomos. En esta fase hemos completado satisfactoriamente una maniobra líder-seguidor, cuya trayectoria se puede observar en la gráfica de la Figura 3.20, donde la línea azul representa la trayectoria de USV2 (líder), simulado en un PC industrial Beckhoff remoto en tiempo real, y la línea roja representa al USV1 (seguidor), físicamente desplegado sobre el agua.

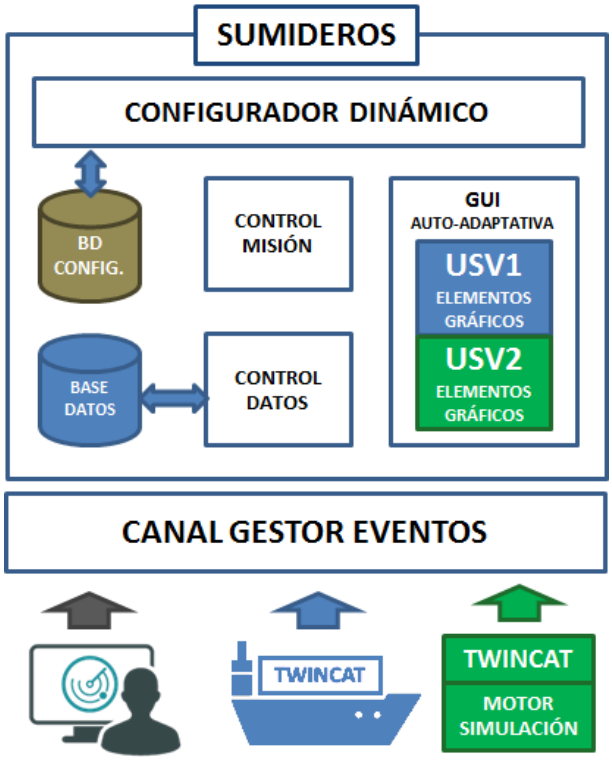


Figura 3.19: Configuración del Experimento Software - Fase 2

Adicionalmente, cuando la fase 2 casi ha concluido, se comprueba el mecanismo de delegación de tareas. Para ello se despliega un segundo GUI en otro ordenador portátil que se usa como segundo CCT y que es gestionado por un segundo operador. Este segundo GUI se conecta a la infraestructura software adaptativa al arrancar, conectándose al servidor en el puerto indicado en su archivo de configuración XML. Una vez conectado a la red, el nuevo CCT es configurado como sumidero para la recepción de datos en el esquema de arquitectura orientada a eventos que da soporte a la infraestructura. El nuevo CCT comienza su actividad monitorizando los vehículos predefinidos en su documento de configuración XML, que en este caso serán el USV1 y el USV2. Una vez el sistema distribuido ha sido conectado (es decir, hay 2 CCTs funcionando simultáneamente en 2 computadoras distintas de forma totalmente transparente para los operadores, un computador industrial de Beckhoff generando en tiempo

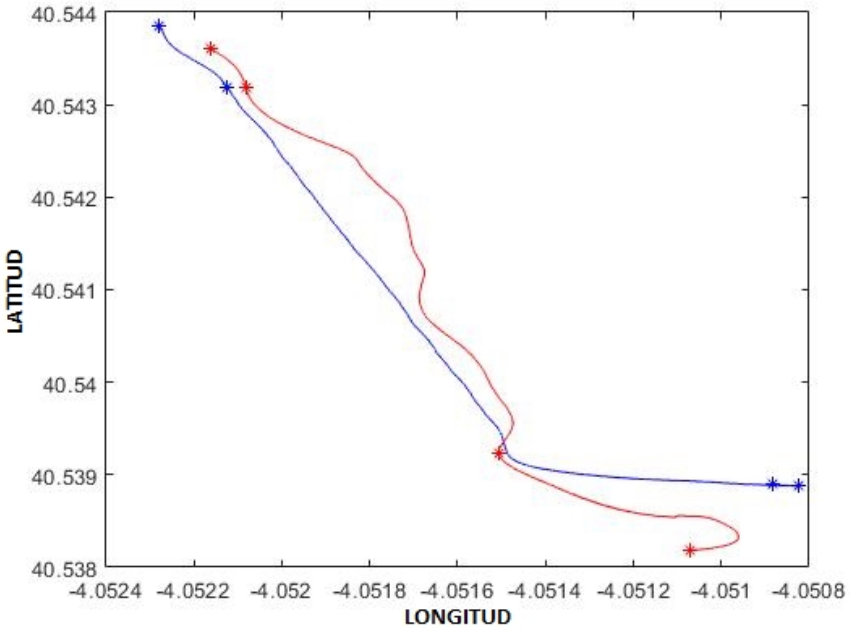


Figura 3.20: Maniobra Cooperativa Líder-Seguidor Realizada por USV1 y USV2 durante la Fase 2

real datos simulados del comportamiento de un modelo del USV2 y un USV1 físicamente desplegado en el agua con otro computador industrial de Beckhoff), se comienza la prueba de delegación de tareas. Para ello el operador 1 acude a la barra de menú superior del CCT y pulsa el boton "Solicitar Asistencia". En ese momento se abre una ventana con un desplegable que permite seleccionar un vehículo de los que se encuentran integrados en la misión en ese instante y un operador de entre los disponibles en el sistema. Una vez seleccionado el operador y el vehículo, el operador 1 pulsa el botón de aceptar y el mecanismo de delegación de tareas le envía un mensaje al operador seleccionado (operador 2 en este caso), al que se muestra una solicitud de asistencia con el vehículo seleccionado (USV2, en este caso). Cuando el operador 2 acepta, su CCT envía la orden al CCT del operador 1. A continuación, los elementos gráficos pertenecientes al USV2 desaparecen del interfaz gráfico del operador 1 (haciendo uso de los mecanismos implementados para las políticas de transparencia utilizadas como

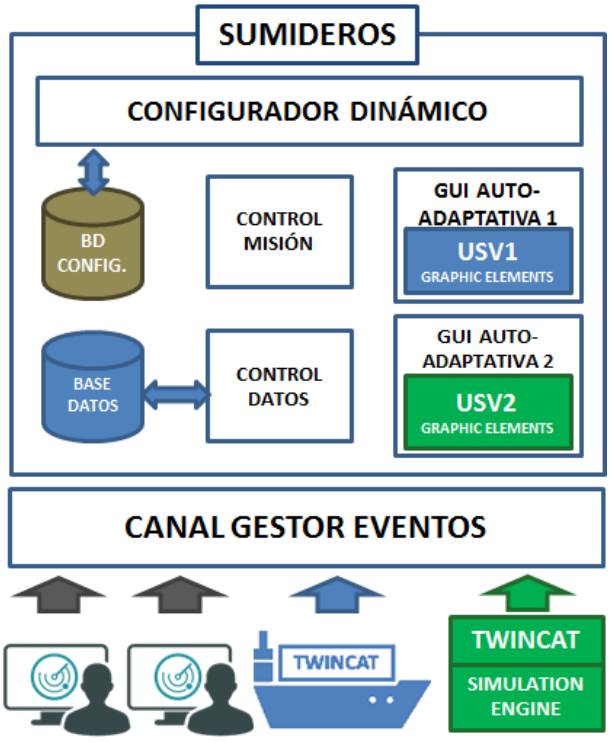


Figura 3.21: Configuración del Experimento Software - Fase 3 (Delegación de Tareas)

ayudas al operador) y se comienzan a mostrar en el interfaz del operador 2, que puede supervisar y controlar el vehículo desde el instante en el que pulsó aceptar en su interfaz. Si el operador 2 rechazase asistir al operador 1, no se produciría ningún cambio y el rechazo se notificaría al operador que ha solicitado ayuda en su interfaz. En el esquema de la Figura 3.21 podemos observar la configuración de la infraestructura adaptativa del sistema tras la reorganización realizada para albergar dos CCTs para sendos operadores.

3.4. Conclusiones

En este capítulo se ha descrito en detalle la arquitectura adaptativa dirigida a eventos que ha sido diseñada para dar soporte al entorno de desarrollo para

centros de control adaptativos en tiempo de ejecución que se presenta en esta tesis.

El diseño arquitectónico ha sido dividido en tres grandes bloques (agentes, canal gestor de eventos y sumideros) siguiendo la filosofía de las arquitecturas dirigidas a eventos clásicas. Esto permite un alto nivel de desacoplamiento, propiedad ideal para CCTs que deben añadir/quitar/cambiar algunos de sus módulos en tiempo de ejecución. La versatilidad del diseño ha sido potenciada con el patrón observador, que permite gestionar las comunicaciones entre agentes y sumideros de forma sencilla y asíncrona (un vehículo o sensor puede enviar datos en cualquier instante de la misión), configurando los sumideros como observadores y el canal que recibe y encapsula los eventos como ente observado. Además, para mejorar las capacidades de nuestro CCT se ha dotado a esta arquitectura de los módulos necesarios para auto-adaptarse en tiempo de ejecución (configurador dinámico), automatizar algunas de las tareas ayudando al desempeño de la labor del operador (control de misión) y almacenar, de forma persistente y para su análisis posterior, los datos referentes a la misión, interfaz y operador (control de datos).

Para analizar las prestaciones del centro de control de tierra adaptativo desarrollado se han definido diferentes marcos experimentales. Éstos nos han ayudado a realizar los experimentos necesarios para validar nuestro software en varias situaciones específicamente seleccionadas para detectar posibles errores en ocasiones críticas. Concretamente, se han realizado pruebas ante tres situaciones potencialmente peligrosas tanto para el hardware como para el cumplimiento de los objetivos de una misión: 1) la recuperación ante caídas en la red de comunicaciones, 2) la inclusión y exclusión de un número elevado de vehículos autónomos en el centro de control adaptativo en tiempo de ejecución, y 3) la inclusión y exclusión de un número elevado de centros de control en la infraestructura adaptativa que forma nuestro sistema durante una misión. Los resultados de estas pruebas determinan que tanto la infraestructura adaptativa como el centro de control diseñado con ella son robustos y eficaces en estas situaciones, ya que no se produce la degradación de la infraestructura (que podría provocar errores du-

rante la realización de las misiones) y porque los tiempos necesarios para llevar a cabo cada una de las tareas probadas son adecuados para que el operador no sufra distracciones ni pierda datos relevantes durante la supervisión de la misión. Por estas razones, consideramos que la robustez del CCT desarrollado permite que sea utilizado sin peligro en misiones de campo.

Por último, se ha comprobado el funcionamiento del CCT en experimentos de campo. En esta tesis, y con el fin de ilustrar el funcionamiento del CCT adaptativo en las pruebas de campo, se detalla uno de los experimentos reales realizados, donde se ha verificado que es capaz de supervisar misiones en las que se combina un vehículo físico y uno simulado en tiempo real, y en las que sucesivamente un vehículo y un CCT son añadidos en tiempo de ejecución. Finalmente, es importante destacar que todas las operaciones de integración de vehículos en maniobras colaborativas y de delegación de tareas entre operadores han sido realizadas con éxito, y que los operadores involucrados en la misión han sido capaces de monitorizar el vehículo y enviarles las consignas de control (o trayectorias planificadas) desde el interfaz gráfico del CCT.

Capítulo 4

Interfaz de Usuario: Transparencia y Adaptabilidad

"Recuerda que no hay código más rápido que la ausencia de código".
("Remember that there is no code faster than no code").

Kevlin Henney

El interfaz gráfico de usuario (GUI) es uno de los módulos más importantes en una aplicación, ya que es el responsable de mostrar al operador los datos y resultados de las tareas ejecutadas por la máquina y de permitirle interactuar con ella. Cuando se trata de un Centro de Control de Tierra (CCT) para monitorizar y controlar equipos de vehículos autónomos heterogéneos, un interfaz bien diseñado puede ser un aspecto fundamental que marque la diferencia entre un CCT útil o uno inservible.

El interfaz de un CCT debe mostrar grandes cantidades de información (telemetría de los vehículos, alarmas, datos del entorno, objetivos de la misión, etc.) y debe permitir al operador comandar el vehículo a diferentes niveles (por ejemplo, testear sus actuadores, realizar maniobras básicas o acoplar nuevos vehículos a la estructura durante el desarrollo de una tarea de la misión). En misiones com-

plejas con vehículos autónomos de tecnología puntera, lo habitual es que sean necesarios varios operadores para supervisar las tareas relativas a monitorización y control. Si la misión implica la cooperación de un equipo de vehículos de naturaleza heterogénea, los datos se multiplican por cada vehículo involucrado, ya que los operadores necesitan monitorizar, al menos, los datos más relevantes de cada uno de ellos. Como consecuencia, según resaltan Perez et al. (2013a) en su trabajo, la carga de trabajo del operador crece exponencialmente con el número de vehículos, porque según aumenta la cantidad de datos a monitorizar, los operadores necesitan concentrar su atención en más elementos gráficos y realizar más tareas, hecho que puede derivar en un incremento de su estrés y fatiga mental. Sebastián y delHoyo (2004) definen la fatiga mental como la disminución de la capacidad física y mental de un individuo, después de haber realizado un trabajo durante un periodo de tiempo determinado y señalan que una vez el individuo está mentalmente agotado, la única forma de continuar realizando correctamente su trabajo es reemplazarlo por otro que esté descansado. Esto nos indica la gran importancia que puede tener en un CCT la reducción de la carga mental de trabajo, que ayudará a prevenir la fatiga mental y por lo tanto permitirá al operador continuar realizando la supervisión de la misión durante más tiempo (idealmente, hasta que ésta finalice). Esto, además, ayuda a reducir la cantidad de operadores implicados, ya que al estar más descansados, necesitarán menos relevos y/o ayudas con sus tareas. Por estas razones, desarrollar GUIs que ayuden a mantener en el operador niveles aceptables de carga mental de trabajo es un aspecto clave y un reto constante en el modelado e implementación de CCTs. Más concretamente, en el centro de control adaptativo implementado durante el desarrollo de esta tesis, se propone un GUI ergonómico y amigable que es capaz de adaptarse dinámicamente a las preferencias del operador y a los cambios en la misión, de forma que no solo se incremente el rendimiento de los operadores, sino que también se ayuda a minimizar la cantidad de operadores involucrados en cada misión.

A continuación presentaremos el diseño e implementación del GUI de nuestro CCT, un software modular auto-adaptativo que reconfigura su vista en tiempo

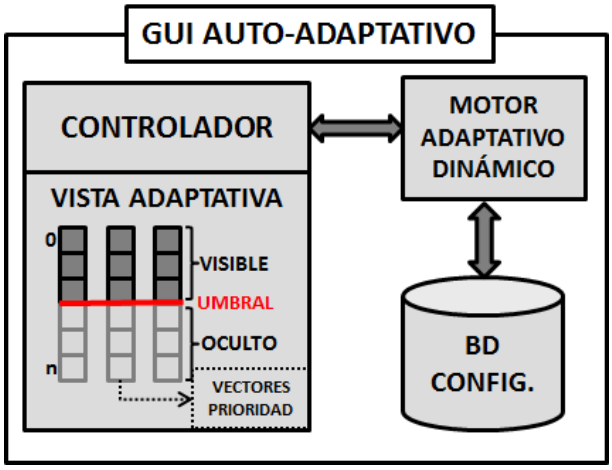


Figura 4.1: Esquema Arquitectónico del Interfaz Gráfico Adaptativo

de ejecución con el objetivo de reducir la carga mental de trabajo del operador. Esto ha sido posible gracias a la implementación de mecanismos adaptativos que cambian la posición de los elementos gráficos con propósitos ergonómicos, basándonos en conceptos clave de arquitecturas adaptativas como las vistas en el trabajo de Oreizy et al. (1999a) y de Garlan et al. (2004), donde se utilizan módulos que se encargan de realizar la evaluación y reestructuración del software mediante bucles externos de control. Además se aplican políticas de transparencia que ocultan aquellos elementos considerados menos relevantes en un instante concreto de la misión, de forma similar a las que podemos observar en los trabajos de Chen et al. (2014) o de Mercado et al. (2016).

Más en concreto, el interfaz gráfico auto-adaptativo está formado por los 4 módulos que pueden verse en la Figura 4.1: el motor adaptativo dinámico (Adaptive Runtime Engine, ARE), el controlador (Controller), la vista adaptativa (Adaptive View) y la base de datos de configuración (Configuration DataBase). Estos módulos se relacionan a través del flujo de datos del bucle de control externo, representado en la Figura 4.2 y en la que podemos observar cómo el ARE (motor adaptativo dinámico) recolecta medidas sobre la situación de la vista, sobre el operador que interactúa con ella y sobre el estado de la misión. Después el ARE toma decisiones basándose en esa información y activa los

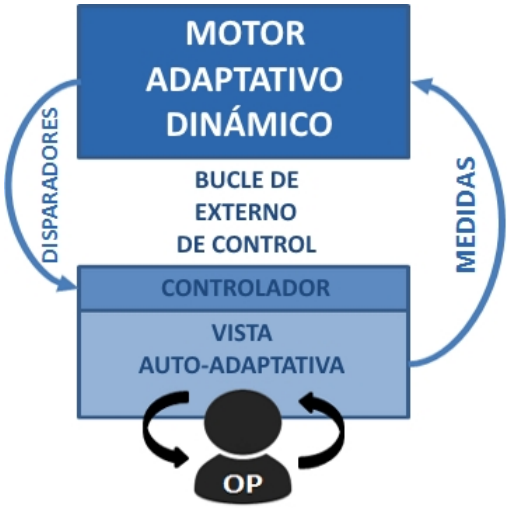


Figura 4.2: Bucle de Control Externo del Interfaz Gráfico Auto-Adaptativo

disparadores necesarios en el controlador, que realizará las modificaciones físicas en la vista. Los cambios en la vista afectan directamente a la carga de trabajo y estrés sufridos por el operador, e indirectamente al desarrollo de la misión y, por tanto, a las futuras decisiones del ARE. La base de datos de configuración interactúa con diferentes módulos para reorganizar los elementos en la vista de acuerdo a las preferencias del operador y a los parámetros globales almacenados sobre la misión. Detalles adicionales sobre la funcionalidad de estos módulos serán presentados en las siguientes secciones.

4.1. Motor Adaptativo Dinámico

El motor adaptativo es el módulo que decide cuando es necesario un cambio en la vista y el que se lo notifica al controlador para que lo ejecute. Es una parte esencial del bucle de control externo, ya que está a cargo de recopilar la información recibida del sistema y analizarla para determinar las modificaciones que deberán llevarse a cabo para que se adapte la vista y se reduzca la carga mental de trabajo del operador. Sus tareas principales son:

Monitorización de datos relacionados con la conducta del operador durante el desarrollo de una misión, como son el tiempo de reacción, el tiempo empleado en una tarea comparado con el tiempo estimado para esa tarea, la carga mental de trabajo (subjetiva cuestionada al operador y estimada en base a un estudio previo de las tareas a realizar), etc. También hay que monitorizar datos referentes a la vista y a los elementos gráficos que en ella se representan, como son su localización en cada instante de la misión, su número o su prioridad. Por último también hay que observar otros datos relacionados con la misión en desarrollo, como son el número y tipo de vehículos involucrados, la información más relevante que se debe mostrar para cada tarea, etc.

Análisis y Procesado de los datos recolectados procedentes de la vista y el operador. Con ellos el ARE realizará un análisis sobre la que se fundamenta la toma de decisiones que desencadenará los cambios realizados por el controlador. Además este módulo realiza un preprocesado de los datos recibidos, para guardarlos en una base de datos y poder analizarlos y mejorar la herramienta.

Las decisiones del ARE, que dependen de los datos recopilados y de sus efectos en el rendimiento del operador, pueden ser programadas ad-hoc o generadas mediante técnicas de Inteligencia Artificial (IA) o aprendizaje automático (Machine Learning, ML). Aprovechando también la naturaleza desacoplada de la arquitectura adaptativa dirigida a eventos, el ARE ha sido implementado como un módulo independiente que permite cambiar fácilmente su comportamiento así como incorporar diferentes algoritmos de IA o ML, simplemente sustituyendo su módulo software por otro similar. Esta característica hace del CCT propuesto una plataforma de experimentación ideal para probar diferentes técnicas de ayuda al operador aplicadas al interfaz gráfico y sus componentes. Estas técnicas mantendrán los niveles de carga mental de trabajo y fatiga en valores razonables, disminuyendo la presión que el operador debe soportar cuando la cantidad de datos o tareas durante una misión son excesivos.

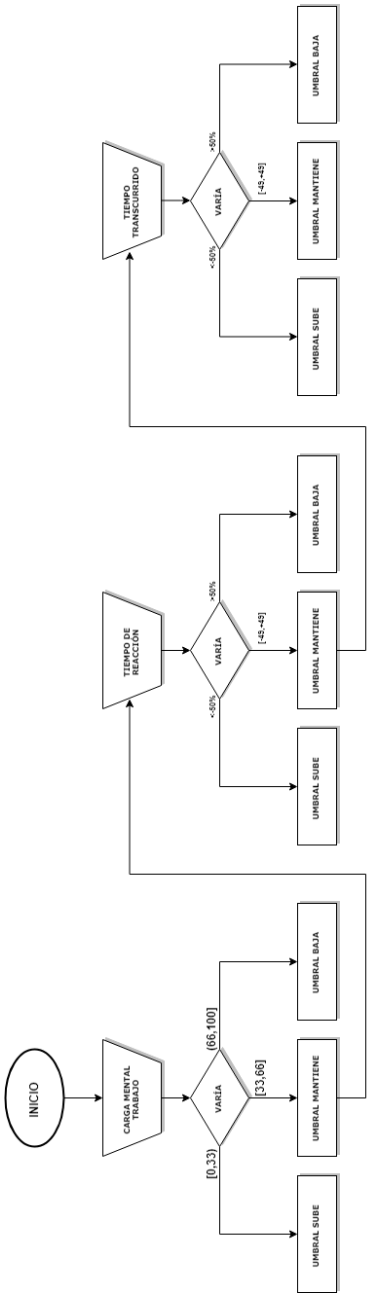
Durante las fases de experimentación realizadas durante esta tesis, los mecanismos implementados en el ARE son los siguientes:

Estimador de Carga de Trabajo evalúa la carga mental de trabajo del operador a partir de tres factores que se miden de forma periódica durante la ejecución de la misión. Estos factores son: 1) el Tiempo de Reacción (Tr), 2) la carga mental de trabajo subjetiva, y 3) el tiempo transcurrido (Tt), que se obtiene como el tiempo estimado para la tarea - tiempo empleado en la tarea. La carga mental de trabajo, además de medirse de forma subjetiva, depende de varios índices indirectos. Como muestran Sebastián y delHoyo (2004), esta carga mental puede ser calculada en base al tiempo de respuesta del operador frente a una tarea y a la fatiga que éste sufre en un instante determinado. Si el tiempo de reacción sube, suele ser indicador de que la fatiga en el operador está aumentando, y por tanto también está creciendo la carga mental de trabajo. Esta carga mental también depende de la dificultad de la tarea realizada y del tiempo disponible para la misma (cuanto más tiempo disponible, menos presión y por tanto menor carga mental de trabajo). Por estas razones, en nuestro sistema se toman medidas directas de la carga mental de trabajo subjetiva durante el transcurso de la misión, así como del tiempo de reacción (Tr) y del tiempo transcurrido (Tt), para que el ARE pueda poder decidir cuando la vista debe ser modificada para mitigar la carga mental de trabajo.

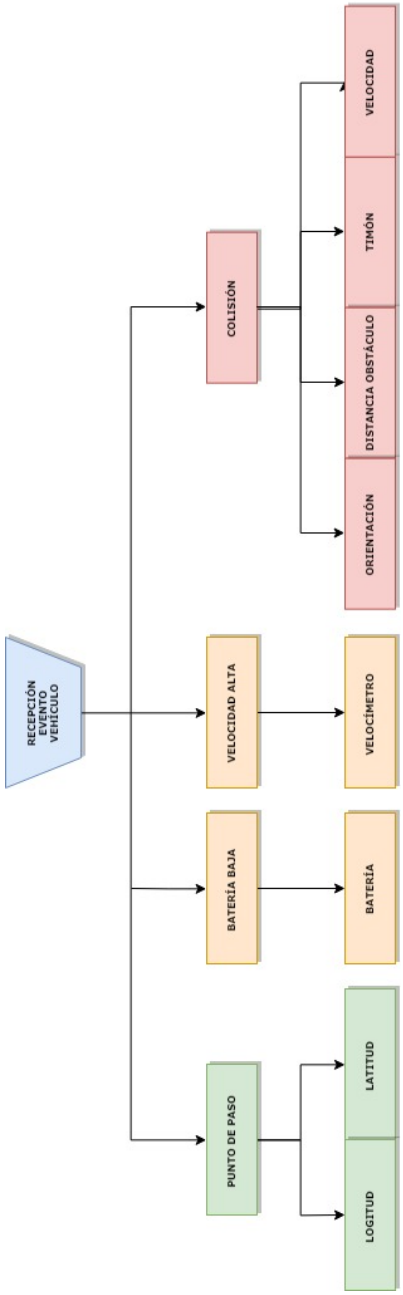
Calculador del Umbral de Transparencia es una de las herramientas más importantes de las que dispone la vista para disminuir la carga mental de trabajo del operador. Calcula el valor del umbral que mostrará más o menos información al operador basándose en dos factores combinados: 1) la estimación de la carga mental de trabajo vista anteriormente, y 2) el numero de elementos gráficos mostrados en la vista. De esta forma, para cada perfil de operadores, se puede considerar un número máximo y mínimo de elementos gráficos que deben ser mostrados por defecto. Además, el número de elementos gráficos mostrados por la vista en cada instante

dependerá de la tarea en progreso y del nivel de carga mental de trabajo de un operador concreto en dicho instante. Así, a menor nivel de carga mental de trabajo se mostrarán más elementos gráficos y a mayor carga se mostrarán menos, siempre dentro del valor mínimo necesario para realizar correctamente cada tarea.

Arbol de Decisión define el comportamiento de la vista, asignando valores de beneficio y peso a cada elemento gráfico basándose en la tarea que se está realizando (qué elementos gráficos son más importantes en este instante) y en las preferencias del operador. De este modo el sistema se asegura que los elementos gráficos mostrados en un instante determinado son, al menos, los que tienen mayor prioridad en ese instante (durante la ejecución por parte del operador de una tarea concreta).



(a) Ejemplo del Algoritmo para el Cálculo del Umbral de Transparencia



(b) Ejemplo de un Fragmento del Árbol de Decisión

Figura 4.3: Algoritmo de Cálculo del Umbral de Transparencia y Fragmento del Árbol de Decisión

En la Figura 4.3a podemos ver un sencillo algoritmo para el cálculo del umbral de transparencia que se ha utilizado en las fases de experimentación. En este caso se prioriza la carga mental de trabajo sobre el resto de valores medidos, el siguiente valor con más peso es el tiempo de reacción medido en ciertas tareas y, en tercer lugar, el tiempo transcurrido. La carga mental de trabajo que se introduce en el algoritmo es una combinación de la carga mental de trabajo estimada (que depende de Tr y Tt) y de la carga de trabajo subjetiva consultada al operador durante algunas fases de la misión. Esta combinación puede modificarse fácilmente variando los pesos específicos asignados a cada una de las medidas. En este caso, como el propio algoritmo tiene en cuenta los valores de Tr y Tt, la mayor parte del peso se concentra en las medidas subjetivas de carga mental de trabajo. Además, si la carga mental de trabajo aumenta por encima del 66 % de la capacidad del operador, se baja en un nivel el umbral de transparencia (lo que conlleva ocultar un elemento gráfico por cada vehículo monitorizado), si baja por debajo del 33 % se aumenta en un nivel el umbral (lo que hace que se muestre un elemento gráfico más por cada vehículo monitorizado) o, si se mantiene entre estos valores, el umbral conserva su valor. En este último caso (cuando se mantiene el valor del umbral) el algoritmo consulta el siguiente índice, el tiempo de reacción. Si su valor aumenta/disminuye por encima/debajo del +50/-50 %, el umbral aumenta/baja su valor en una unidad. En caso de no superar esos valores, el cambio del umbral se mantendrá a la espera de consultar el tercer índice, el tiempo transcurrido. Si la diferencia entre el tiempo estimado para la tarea y lo que el operador tarda en realizarla aumenta/disminuye su valor en más de un +50/-50 %, el valor del umbral aumentará/disminuirá un nivel. En caso de no producirse esos cambios, el umbral permanece con su valor actual hasta que se produzca la siguiente ejecución del algoritmo de cálculo del umbral de transparencia. Cada ejecución de este algoritmo se realiza periódicamente (cada 60 segundos, por ejemplo) o ante la llegada de un evento que lo dispare.

Por otra parte, en la Figura 4.3b se encuentra un fragmento esquematizado del árbol de decisión que se utiliza en el motor adaptativo dinámico. En él se puede observar la entrada de eventos que nos dirigirán a una zona u otra del

árbol. En concreto, se observan 3 tipos de eventos remarcados con 3 colores diferentes. Los eventos coloreados en verde simbolizan comprobaciones rutinarias que no implican la realización de una tarea urgente por parte del operador (por ejemplo, verificaciones rutinarias del estatus de la misión como la llegada a un punto de paso). Los eventos coloreados en ámbar simbolizan alarmas de aviso en las que si no se actúa se puede alcanzar una situación crítica (por ejemplo, un nivel bajo de batería o una velocidad excesivamente alta en alguno de los vehículos que podrían derivar en la pérdida de control sobre el mismo). Los eventos coloreados en rojo simbolizan las alarmas de mayor prioridad, sobre las que hay que actuar de inmediato para evitar un accidente (por ejemplo, una alarma de detección de obstáculo, sobre la que si no se actúa ya sea activando automáticamente un protocolo de evasión o conduciendo el vehículo manualmente, se puede producir una colisión del vehículo). El motor dinámico recibe el evento y lo introduce en el algoritmo que guía al sistema por las ramas del árbol de decisión, que finaliza en los nodos que indican qué elementos gráficos deben ser promocionados (aumentar su nivel de prioridad) para que el sistema sepa que deben ser visualizados por parte del operador. Una vez llegado a este nodo, el sistema activa el disparador que modifica el índice de prioridad de movimiento (según el valor de visualización y coste de movimiento asignado por el árbol de decisión) para que los elementos involucrados se coloquen en las mejores posiciones de los vectores de prioridad (por defecto arriba/izquierda, aunque éstas pueden ser modificadas en base a las preferencias del operador). Así, por ejemplo se promocionará la longitud y latitud moderadamente cuando se reciba un evento que indique que un vehículo ha alcanzado un punto de paso (waypoint), de forma que éstos se mantengan en la parte visible de la vista, pero no necesariamente en el lugar más prioritario. Si el evento recibido es ámbar, los elementos gráficos correspondientes serán promocionados a alguno de los lugares más prioritarios (por ejemplo ante una alarma de batería baja, programable para activarse cuando se alcanza el porcentaje deseado, el indicador de batería subirá varios puestos para quedar situado en las posiciones más prioritarias. Por otro lado, si el evento recibido es rojo, el algoritmo se asegura de que los elementos gráficos necesarios

se coloquen en los puestos más prioritarios de la vista (por ejemplo, en el caso de una alarma de colisión, la orientación, la distancia al obstáculo, la posición del timón y la velocidad del vehículo aparecerían por ese orden en las cuatro primeras posiciones organizadas de mayor a menor).

4.2. Controlador

El controlador es el módulo ejecutor que realiza en la vista los cambios decididos por el motor dinámico, es decir, es la estructura que aplica los cambios que marcan las políticas de transparencia y adaptabilidad. Además actualiza los Vectores de Prioridad que se pueden ver en la Figura 4.1, donde los elementos gráficos asociados a cada vehículo en la misión son ordenados de acuerdo a su prioridad, calculada a partir del ratio beneficio/coste. Información detallada sobre los vectores de prioridad se presentará en la Sección 4.3.

Cuando el motor adaptativo dinámico decide que ha de producirse un cambio en la vista, debe enviar al controlador los nuevos valores que se han calculado y un disparador que active los mecanismos implementados para realizar cada uno de esos cambios.

Si se produce un cambio en el umbral de transparencia, el controlador recibe el nuevo valor del umbral y un disparador que activa el mecanismo que hace que los componentes se reajusten. En este caso, el disparador activa el algoritmo que comprueba cada uno de los vectores de prioridad presentes en la vista (uno por vehículo), introduce el nuevo valor del umbral y activa la política de transparencia, que hace que los elementos gráficos situados por encima del umbral sean mostrados al operador mientras que aquellos que se encuentran por debajo queden ocultos, es decir, si el valor del umbral sube mostrará más datos sobre la misión al operador y si el valor del umbral baja mostrará menos para que éste se pueda concentrar en aquellos que son más relevantes en el instante concreto en el que se desarrolla cierta tarea. Gracias a esta reducción de datos, el operador tardará menos en localizar, entre la información disponible, la necesaria para llevar a cabo una determinada tarea, reduciendo su tiempo de reacción y

el tiempo necesario para cada tarea y, por ende, reduciendo su carga mental de trabajo y su fatiga.

Cuando el motor dinámico recibe una notificación de evento y decide que debe cambiar la prioridad de alguno de los elementos gráficos de la vista, el proceso es análogo al que sucede cuando hay un cambio en el umbral de transparencia. Se reciben los nuevos valores junto a los identificadores de los elementos gráficos y el vehículo al que pertenecen, además del disparador que activa el proceso de adaptación de la vista en el controlador. Entonces el disparador activa el algoritmo que cambiará dichos valores en los vectores de priorioridad, de forma que a continuación éstos sean reordenados de mayor a menor prioridad para proceder después a actualizar la vista. Este proceso recoloca los elementos gráficos en el interior de los márgenes accesibles de la vista y en orden según el tamaño asignado para cada elemento gráfico del vector. En otras palabras, las políticas de adaptación del GUI acomodan la posición y tamaño de los elementos de la vista en tiempo de ejecución de acuerdo a las necesidades del operador y la tarea en proceso. En la Figura 4.4 podemos ver un ejemplo de lo que ocurre durante este proceso, en el que un USV llega a un punto de paso, el árbol de decisión aumenta el ratio beneficio/peso de los elementos gráficos de Longitud y Latitud para que se coloquen en los lugares elegidos como más prioritarios (arriba o a la izquierda por defecto) dentro de los límites establecidos para el Vector de Prioridad de ese vehículo autónomo. Podemos ver que el elemento gráfico Latitud ya se encontraba en el segundo puesto de mayor prioridad, mientras que el que representa la Longitud estaba situado en la tercera posición. Al ser promocionado, pasa al primer lugar, intercambiando el puesto con el elemento gráfico que representa la batería.

Las políticas de adaptación del Controlador tienen también en cuenta las preferencias del operador para mostrar los elementos en la Vista. Por ejemplo, la posición prioritaria preferida para los elementos (arriba o abajo, izquierda o derecha) o los colores que visualiza más fácilmente (podría elegir rojo, verde, negro, etc.). En otras palabras, los elementos gráficos asociados a los Vectores de Prioridad con el mismo umbral de transparencia pueden ser representados de

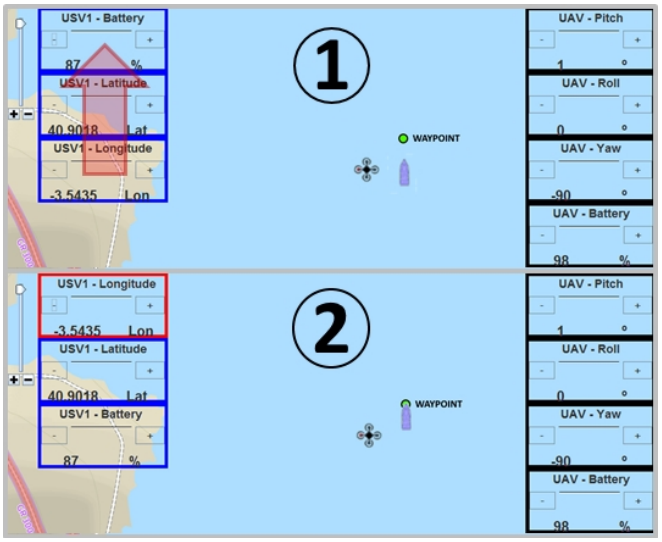


Figura 4.4: Ejemplo de una Adaptación Ejecutada por el Controlador

forma muy diferente en la vista para adaptarse no solo al estado actual de la misión, sino también al estrés y las preferencias ergonómicas del operador. Además si en algún momento el usuario necesita consultar un dato que no se encuentra en la vista, puede solicitarlo al sistema pulsando el botón [+] de cualquiera de los elementos gráficos de ese vehículo. Esta acción desplegará un menú en el que aparecen todos los datos disponibles del vehículo y si pulsa en cualquiera de ellos, éste será automáticamente promocionado a la última posición visible.

4.3. Vista Adaptativa

La vista es el módulo software que muestra la información al operador y le permite interactuar con el resto del sistema (comandar vehículos, solicitar datos, etc.). Está formada por una serie de controladores y/o comandadores que han sido integrados en el interfaz gráfico para enviar consignas a los vehículos y a cualquier otro dispositivo involucrado en la misión. También cuenta con un conjunto de elementos gráficos que representan la información recibida de cada uno de los dispositivos involucrados en la misión que sea considerado relevante para el desarrollo de la misma. Por último, en la vista de un CCT se hace

imprescindible contar con un mapa que ayude al operador a hacerse una imagen mental de la disposición del equipo de vehículos en el entorno en el que se realiza la misión. Esto ayuda a mejorar notablemente la consciencia situacional del operador y, por lo tanto, a que la misión se lleve a cabo con éxito. En este mapa se representan los vehículos autónomos, las rutas, los puntos de paso y cualquier otro elemento que sea necesario situar durante la misión. Como hemos apuntado con anterioridad, el número de elementos que se muestran en la vista, así como su localización en la pantalla, tamaño y color, cambian dinámicamente a lo largo del desarrollo de la misión. Estos cambios son llevados a cabo por el controlador de acuerdo a las decisiones tomadas en el motor dinámico con el objetivo de estabilizar (y si es posible disminuir) la carga mental de trabajo y la fatiga del operador. Estos cambios son posibles gracias a un elemento arquitectónico clave, los vectores de prioridad, en cuyas celdas se enlazan los elementos gráficos que muestran los datos recibidos de cada uno de los vehículos involucrados en la misión. Cada elemento gráfico tiene un ratio de prioridad que irá cambiando en tiempo de ejecución por medio de los valores o fórmulas que se encuentran en los nodos del árbol de decisión descrito en la Sección 4.1, a los que se llegará dependiendo del evento o tarea que se esté realizando en ese instante. Además, los vectores de prioridad tienen un umbral de transparencia (que puede ser común o individual para cada uno de los vehículos) y que indica a la vista cuántos de los elementos gráficos enlazados al vector deben ser mostrados en ese instante. Esta adaptación dinámica se logra gracias al bucle externo de control, en el que el motor dinámico monitoriza los índices que le permiten tomar las decisiones sobre adaptación y transparencia, enviando los disparadores al controlador que modificará los vectores de prioridad, reordenándolos de mayor a menor prioridad (en caso de activar el disparador de adaptabilidad) o cambiando el número de elementos por vector que se muestran (si se envía el disparador de transparencia). En la Figura 4.5 podemos observar un esquema de la disposición de los vectores de prioridad para n vehículos autónomos (Unmanned Vehicles, UV), así como el funcionamiento de su umbral de transparencia (que indica a partir de donde los

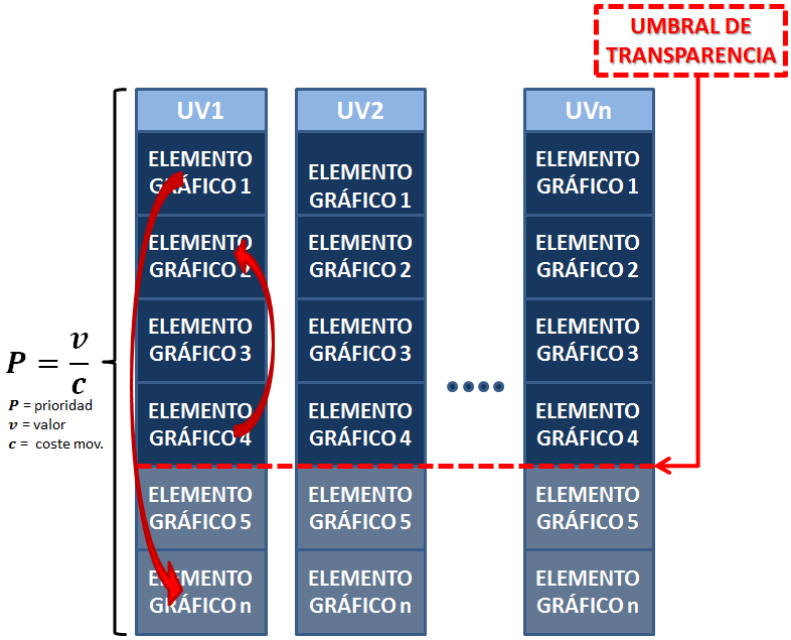


Figura 4.5: Esquema de los Vectores de Prioridad para n Vehículos

elementos dejan de mostrarse) y del ratio de prioridad (que cambiará de orden los elementos cuando se active la ordenación del vector de prioridad).

El propósito final de estos cambios es ayudar a los operadores en cuatro aspectos fundamentales: 1) mejorar la ergonomía de la vista para que realicen sus tareas lo más cómoda e intuitivamente posible, 2) localizar más rápidamente la información más importante en cada instante de la misión, 3) eliminar la información menos relevante para que el operador pueda concentrarse en las tareas importantes, y 4) mejorar la consciencia situacional en misiones con múltiples vehículos heterogéneos. Finalmente, merece la pena mencionar que la vista también puede adaptarse en tiempo de ejecución a peticiones directas del operador relacionadas con qué información mostrar y donde mostrarla. Estas peticiones directas pueden solapar, si es necesario, las decisiones tomadas por el ARE y ejecutadas por el Controlador. Además, estas peticiones serán almacenadas en la Base de Datos de Configuración para ser analizadas a posteriori y determinar si puede ser beneficioso incluir los comportamientos asociados en sus políticas de Transparencia o Adaptabilidad de futuras misiones.

4.4. Base de Datos de Configuración

Se trata del módulo de almacenamiento persistente que guarda los principales parámetros de configuración del interfaz gráfico y los datos de los perfiles del operador. En resumen, provee información de cómo mostrar los elementos gráficos de la forma más ergonómica posible para un operador específico que lleva a cabo una tarea/misión específica.

Los parámetros principales de configuración son una guía básica sobre cómo posicionar los elementos gráficos en la vista en líneas generales. Se almacenan datos como los límites de la vista, las áreas donde colocar los elementos gráficos o las posiciones prioritarias por defecto (por ejemplo, arriba o a la izquierda). También almacena datos sobre los nodos de los árboles de decisión (ratios beneficio/coste iniciales de cada elemento gráfico) y el tiempo estimado para que un operador realice cada tarea.

Además, cuando un operador es registrado en el sistema, se crea un perfil que almacena su nombre de usuario y contraseña, así como otros datos relacionados con sus preferencias ergonómicas como su elección de posiciones prioritarias, sus colores preferidos para los elementos gráficos prioritarios, etc. Durante el desarrollo de las misiones, la base de datos también almacena información sobre el comportamiento del operador para analizarlo a posteriori y, si se considera que puede producir una mejora, adaptar las decisiones del motor dinámico a las peticiones que ha realizado el operador durante la ejecución de una tarea o misión específicas (ver Apéndice A). Pongamos como ejemplo la monitorización de una misión en la que un vehículo autónomo debe completar una ruta previamente planificada. Para comprobar que el vehículo sigue la ruta correctamente, además de visualizar su posición relativa en el mapa, el operador deberá verificar las coordenadas de latitud y longitud cada vez que llegue a un punto de paso de la ruta. En este hito de la misión, al llegar el vehículo a un punto de paso, se genera un evento que se envía al CCT. Dicho evento será procesado por el sistema y hará que árbol de decisión del motor dinámico llegue al nodo en el que suben los ratios de prioridad beneficio/peso de los elementos gráficos de

coordenadas (latitud y longitud). Al subir el ratio, estos dos elementos gráficos se mostrarán en posiciones de mayor prioridad y el operador podrá comprobar si las coordenadas coinciden con las de la ruta planificada. En este punto de la misión, al operador se le puede ocurrir que fuese útil ver también la orientación del vehículo para mejorar su consciencia situacional, averiguando de esta forma si el vehículo ha llegado al punto de paso con una orientación adecuada para seguir la ruta preestablecida. Para poder observar dicha información, el operador pulsará el botón habilitado en el interfaz para mostrar todos los datos que se están recibiendo del vehículo en ese instante de la misión y podrá pulsar sobre cualquiera de ellos para que se muestre de inmediato. Esta opción solicitada por el operador será almacenada en la base de datos de configuración junto con la marca temporal, el vehículo y la tarea en ejecución (junto con otros datos referentes a la misión). Si en el análisis posterior se detecta que ese usuario ha solicitado ese dato en un instante concreto para una tarea concreta en varias ocasiones, el sistema podría determinar que ese dato es relevante, añadirlo el nodo de evento de "punto de paso" del árbol de decisión junto con los que ya estaban, de forma que la orientación se muestre automáticamente al operador al alcanzar un punto de paso en la siguiente misión.

4.5. Marco Experimental: Experimentos de Validación

El interfaz gráfico de usuario auto-adaptativo ha sido puesto a prueba para determinar si es capaz de llevar a cabo las tareas para las que ha sido diseñado.

Las primeras fases de pruebas sobre el interfaz gráfico de usuario del CCT serán llevadas a cabo por medio de un marco experimental, que, de forma semejante a los marcos experimentales utilizados en el capítulo anterior simula una situación específicamente seleccionada para analizar el funcionamiento de la herramienta antes de utilizarla en misiones reales. En este capítulo se han diseñado 3 familias de experimentos con el objetivo de verificar las principales funciones del interfaz así como las situaciones más críticas que pueden producir-

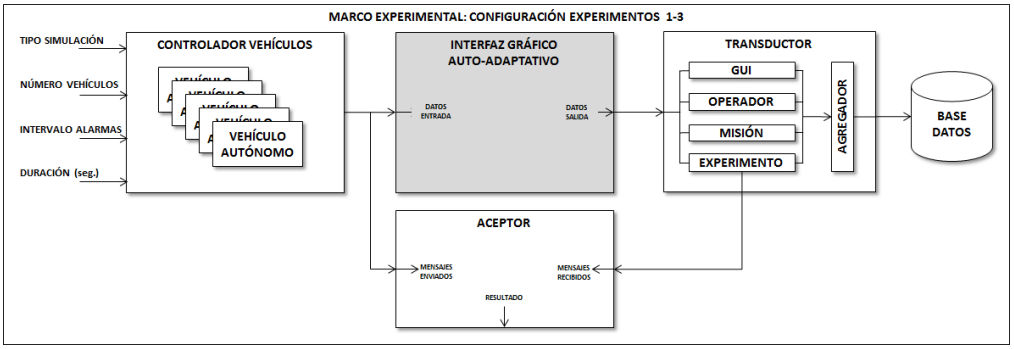


Figura 4.6: Configuración del Marco Experimental 4

se durante una misión, relacionadas con: 1) mostrar correctamente, sin retardos ni pérdidas, la información recibida de los agentes (vehículos, sensores, etc.), 2) ejecutar y mostrar correctamente las políticas de transparencia , y 3) ejecutar y mostrar correctamente las mecánicas de adaptabilidad. Para estas tres familias de experimentos, se utilizarán tres vehículos que serán monitorizados durante una misión. Además, en la primera familia se contabilizarán los mensajes enviados y los recibidos, así como las alarmas de situación crítica mostradas con la intención de determinar si el interfaz muestra correctamente toda la información. En la segunda se simularán cambios en el umbral de transparencia que obligarán al interfaz a ocultar y volver a mostrar algunos elementos gráficos con el objetivo de demostrar que el interfaz no produce fallos ni pierde información en el proceso. En la tercera famila se realizarán pruebas sobre el mecanismo de adaptabilidad que reordena los elementos gráficos para probar que el proceso se realiza correctamente.

A continuación describiremos el marco experimental utilizado para estas 3 familias de experimentos, que en este caso será el mismo para los tres, así como las características más relevantes de los tres tipos de experimentos, sus objetivos y los resultados obtenidos en cada uno de ellos.

4.5.1. Diseño del Marco Experimental 4

Se ha diseñado un nuevo marco experimental que genera la información y las condiciones necesarias para la realización de la serie de experimentos sobre el sistema que se desea probar, el interfaz gráfico auto-adaptativo. Dicho marco, el cuarto de esta tesis, constará, además del módulo que se desea probar, de los tres bloques que pueden observarse en la Figura 4.6: el controlador de la simulación de los vehículos, el transductor y el aceptor. Éste último módulo ha sido incorporado para facilitar la validación de los resultados de este bloque de experimentos, y tiene como tarea comparar automáticamente el número de mensajes enviados por el módulo controlador de simulación y los recibidos y mostrados que serán recogidos por el transductor. A continuación detallaremos las funciones de cada uno de los módulos del marco experimental:

Controlador de vehículos es el módulo encargado de gestionar las instancias de los vehículos que forman parte de la simulación. Se puede configurar su comportamiento por medio de las siguientes variables: el tipo de simulación (para las tres familias de experimento de este capítulo), el número de vehículos que se debe crear, el intervalo de tiempo entre el envío de alarmas al interfaz y la duración del experimento. Una vez seleccionadas las variables, el controlador creará las instancias indicadas del módulo vehículo autónomo utilizado en experimentos anteriores (ver Figura 3.5). Cada vehículo se conectará al sistema y comenzará a enviar mensajes (paquetes de datos) cada segundo para que sean mostrados en el interfaz. Adicionalmente, cada vehículo enviará una alarma urgente de forma periódica (de acuerdo con el tiempo configurado) que deberá ser mostrada en el interfaz del CCT. También, por cada vehículo se enviarán 4 datos de telemetría que serán mostrados en el interfaz en sus correspondientes elementos gráficos, entre ellos, las coordenadas para que el interfaz los muestre en el mapa. Una vez transcurrido el tiempo de ejecución indicado previamente, se destruirán las instancias de los vehículos y se concluirá el experimento.

Transductor es el mismo módulo que ha sido utilizado en los marcos experimentales del capítulo anterior. Está a cargo de recibir los datos del sistema y guardarlos en un almacenamiento persistente. Cuenta con cuatro canales independientes de recepción de datos para el interfaz (GUI), el operador, la misión y el experimento; y un agregador que se encarga de tratarlos y guardarlos en un almacenamiento persistente. Durante este bloque de experimentos, se encarga de guardar el número de mensajes recibidos y mostrados por el interfaz, el número de alarmas mostradas, los cambios en el umbral de transparencia y los cambios de orden de los elementos gráficos relacionados con la adaptabilidad.

Aceptor: es el módulo encargado de recibir el número de mensajes (paquetes de datos) enviados por el controlador de la simulación y compararlo con el número de mensajes recibidos y mostrados por el interfaz. Se encuentra implementado mediante un disparador que aumenta el contador de mensajes enviados cada vez que uno sale del controlador de simulación y mediante otros dos contadores que incrementan su valor cada vez que el transductor recibe los datos sobre un mensaje almacenado y mostrado en el interfaz. Este módulo devuelve las cantidades totales y el porcentaje de éxito.

4.5.2. Experimentos de Recepción y Muestra de Datos

Uno de los aspectos más importantes de un interfaz gráfico de usuario es la capacidad de mostrar correctamente toda la información proporcionada por el sistema. Esta capacidad es todavía más crítica en el interfaz de un CCT, ya que de esta información puede depender no solo la consecución del objetivo de una misión, sino también la posibilidad de evitar situaciones que puedan acarrear peligros serios para los equipos desplegados de forma remota (vehículos, sensores, etc.). Por estas razones, se ha diseñado un experimento con el objetivo de determinar si el interfaz de nuestro CCT recibe y muestra toda la información enviada por los agentes integrados en el sistema.

4.5.2.1. Diseño del Experimento

Durante la ejecución de este experimento, el simulador de vehículos creará tres instancias de vehículos autónomos, ya que éste es el número máximo de vehículos que pueden tener involucrarse en las pruebas de campo que pueden tener lugar en el marco del proyecto SALACOM. Además, no se considera relevante la introducción de un mayor número de vehículos para los resultados de este experimento, ya que el correcto funcionamiento de la infraestructura y el interfaz ante la introducción de un mayor número de vehículos, así como la ausencia de retardos provocados por los mismos, ya han sido analizados en los experimentos detallados en la Sección 3.2. Una vez creadas las instancias de los vehículos y establecidos los canales de comunicaciones, se comienza el envío periódico de paquetes de información por parte de cada vehículo, consistente en un total de 100 paquetes de datos (que encapsulan 4 campos de información de telemetría) y 10 alarmas. Además, los datos y las alarmas son respectivamente enviados con una frecuencia de 1 y 0.1 Hz. Debido a esta sincronización, los tiempos de recepción de los paquetes son prácticamente simultáneos, por lo que se somete al sistema a una situación más crítica de lo que se daría en las condiciones habituales de una misión. Cada paquete de datos enviado, así como cada paquete recibido y cada dato mostrado es contabilizado por el marco experimental, y una vez los vehículos terminan el envío de datos, la simulación se detiene y se procede a realizar un análisis de los datos recogidos. También se almacena el tiempo que tarda cada mensaje y alarma en ser mostrado (computado desde que el paquete de datos es recibido por el interfaz hasta que, después de actualizar los datos de cada componente, se activa el refresco de pantalla que mostrará los cambios). Para este experimento, se supondrá una conexión ideal (realizada mediante Sockets TCP/IP en modo local) con suficiente ancho de banda y sin retardos ni interrupciones, ya que el interés del experimento es valorar únicamente la eficacia del interfaz gráfico. Por lo tanto, las medidas mostradas serán referentes al interfaz, tanto en tiempos de retardo como en paquetes recibidos/perdidos.

# UV	PAQUETES ENVIADOS	PAQUETES RECIBIDOS	DATOS MOSTRADOS	% DATOS MOSTRADOS	RETARDO MEDIO (seg.)	σ	RETARDO TOTAL (seg.)
1	100	100	400	100	0,16697	0,00293	16,697
2	100	100	400	100	0,16710	0,00292	16,710
3	100	100	400	100	0,16764	0,00324	16,764

(a) Información de los Paquetes de Datos

# UV	ALARMAS ENVIADAS	ALARMAS RECIBIDAS	ALARMAS MOSTRADAS	% ALARMAS MOSTRADAS	T RETARDO MEDIO (seg.)	σ	T RETARDO TOTAL (seg.)
1	10	10	10	100	0,1672	0,00305	1,672
2	10	10	10	100	0,1686	0,00277	1,686
3	10	10	10	100	0,1659	0,00203	1,659

(b) Información de las Alarmas

Tabla 4.1: Resultados del Experimento de Recepción y Muestra de Datos

4.5.2.2. Resultados

Tras la ejecución del experimento, se han analizado los datos y se han resumido en dos tablas, en las que respectivamente se representan los datos referentes a los paquetes y a las alarmas.

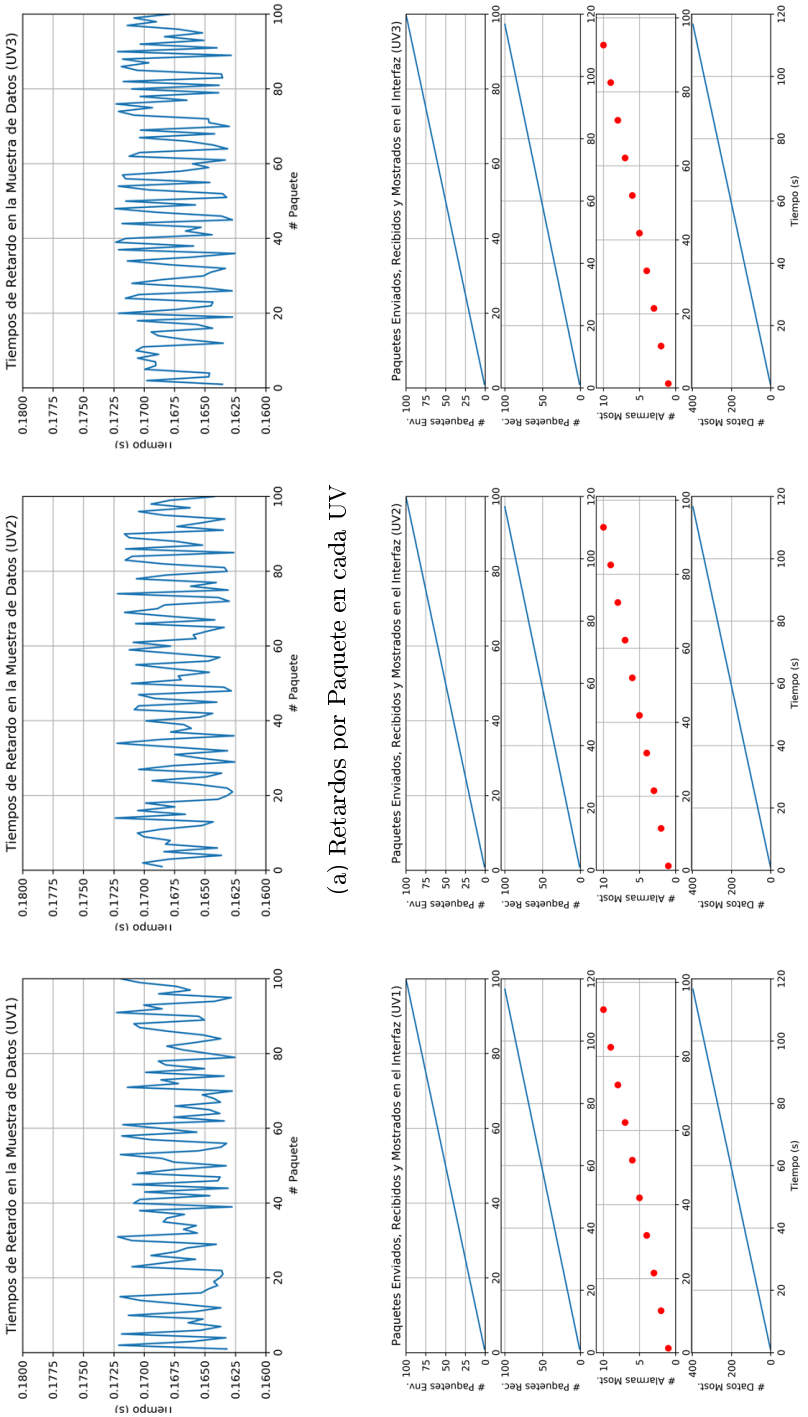
En la Tabla 4.1a se resume el número de paquetes enviados, recibidos y datos mostrados, así como el tiempo medio que se tarda en mostrar un paquete de datos, la desviación estándar de los tiempos en torno a esta media y el total que se ha tardado en mostrar los 100 paquetes de datos. De forma análoga, en la tabla de la Figura 4.1b, se encuentra el resumen del número de alarmas enviadas, recibidas y mostradas, y del tiempo medio, desviación estándar y tiempo total que se tarda en mostrar las diez alarmas.

Los resultados obtenidos son muy positivos ya que, gracias al patron observador, el envío de datos asíncrono desde cada vehículo se realiza de forma robusta y eficaz. Los tiempos de procesamiento de los paquetes de datos (correspondientes al desempaquetado, distribución y actualización de gráficos) son razonablemente

pequeños (del orden de décimas de segundo), lo que nos permite recibir y mostrar gran cantidad de datos sin que se aprecien retardos notables en el refresco de la pantalla. Estos datos son realmente positivos dentro del ámbito del proyecto SALACOM, ya que tienen en cuenta el hecho de que durante esta simulación el interfaz gestiona la recepción de los paquetes envíados simultáneamente desde los tres vehículos autónomos simultáneamente. Además, la velocidad de cómputo está limitada por el equipo, por lo que si los experimentos fuesen ejecutados en uno más potente, los tiempos podrían reducirse más.

En las gráficas que aparecen en la Figura 4.7 podemos encontrar un resumen de los tiempos obtenidos durante la ejecución de este experimento. En las tres imágenes presentes en la Figura 4.7a se resume gráficamente el tiempo de procesamiento de cada uno de los paquetes de datos recibidos por los vehículos (UV1, UV2 y UV3 respectivamente). En las tres imágenes de la Figura 4.7b, podemos encontrar las gráficas, de arriba a abajo, de la cantidad de paquetes enviados, la cantidad de paquetes recibidos, la cantidad de alarmas mostradas y la cantidad de datos (4 por paquete) mostrados por cada uno de los vehículos. En ellas se puede observar que se han recibido y mostrado el total de paquetes, datos y alarmas en cada vehículo, por lo que consideramos que el interfaz es eficaz, y también eficiente, dado que los tiempos de procesado son bastante pequeños. Los datos de cada vehículo, enviados en intervalos de un segundo, son mostrados en el interfaz en décimas de segundo, por lo que los datos encolados en el canal de comunicaciones no dejan de ser mostrados durante la ejecución de una misión. Si el sistema funcionase más despacio debido a una sobrecarga de red o a algún otro problema ajeno a nuestro software, se produciría una ralentización en la entrada del buffer que acumula los datos, que quedaría vacío. Esto haría que los datos permaneciesen fijos más tiempo en el interfaz gráfico del CCT. En caso de llegar demasiado rápido (cosa que no ocurrirá en condiciones normales, ya que es prácticamente imposible para un operador monitorizar tal cantidad de datos en un tiempo tan reducido), los datos tampoco se perderían, ya que se acumularían en el buffer e irían mostrándose en el interfaz en intervalos de décimas de segundo. Este mecanismo dota al interfaz de robustez frente a la sobrecarga de

datos, ya que éstos podrían acumularse e ir mostrándose en el interfaz todo lo rápido que permita el hardware en el que se utiliza. Gracias a este experimento, podemos determinar que el interfaz gráfico esta preparado para ser utilizado en pruebas de campo reales ya que, siempre que la red cuente con el ancho de banda y la estabilidad suficientes, no se perderá la información proporcionada por los agentes remotos (vehículos, sensores, etc.) ni se sufrirán retardos notables debidos a la ejecución de los algoritmos de procesado o refresco de pantalla.



(a) Retardos por Paquete en cada UV

(b) Mensajes Recibidos y Mostrados en cada UV

Figura 4.7: Gráficas con los Resultados del Experimento de Recepción y Muestra de Datos

4.5.3. Experimentos con los Mecanismos de Transparencia

Una vez probada la eficacia y robustez en la recepción y la muestra de datos en la vista, debe determinarse si el interfaz está listo para poner en práctica las políticas de transparencia implementadas en el CCT adaptativo. Para ello se debe poner a prueba que los mecanismos implementados con este objetivo pueden ser aplicados en el interfaz gráfico sin que esto ocasione ningún problema, tanto a nivel estructural (los módulos de los elementos gráficos deben aparecer y desaparecer de la pantalla en tiempo de ejecución) como a nivel de eficiencia (los cambios deben poder llevarse a cabo sin que se produzcan retardos prolongados).

4.5.3.1. Diseño del Experimento

Para determinar si el interfaz gráfico es capaz de ejecutar los cambios asociados a los mecanismos de transparencia en tiempo de ejecución, se ha diseñado un experimento de repetitividad cuyo objetivo es controlar las subidas y bajadas del umbral de transparencia, que cambiará con una frecuencia alta durante la ejecución del experimento, y controlar si los cambios de umbral se plasman correctamente en la vista del interfaz gráfico. Con este propósito, el marco experimental lleva a cabo el control y almacenamiento de los tiempos de ejecución de los cambios de umbral. Como en el experimento anterior, se supone una conexión ideal con suficiente ancho de banda y sin retardos ni interrupciones para el envío de datos que deben ser mostrados en los elementos gráficos que aparecen/desaparecen al cambiar el umbral de transparencia.

Para llevar a cabo este estudio, se realizan tres ejecuciones con el marco experimental 4. En cada ejecución hay 3 instancias de los vehículos autónomos que envían paquetes de datos cada segundo para que sean visualizados en el interfaz. Además, el controlador de vehículos envía cada 10 segundos un nuevo valor de umbral de transparencia (cuyo valor oscilará entre 4 y 5) y el disparador que activa los algoritmos que realizan los cambios en el interfaz gráfico. Por lo tanto se realizarán 10 cambios con un intervalo de 10 segundos entre ellos, un ritmo mucho mayor de lo que podría producirse durante el desarrollo de una

Tabla 4.2: Resultados de los Experimentos de Cambio de Umbral

# ITERACIÓN	CAMBIOS UMBRAL	CAMBIOS EJECUTADOS	% CAMBIOS MOSTRADOS	RETARDO MEDIO (seg.)	σ	RETARDO MAX. (seg.)	RETARDO TOTAL (seg.)
1	10	10	100	0,7259	0,0299	0,7688	7,259
2	10	10	100	0,7214	0,0286	0,7677	7,214
3	10	10	100	0,7435	0,0255	0,7750	7,435

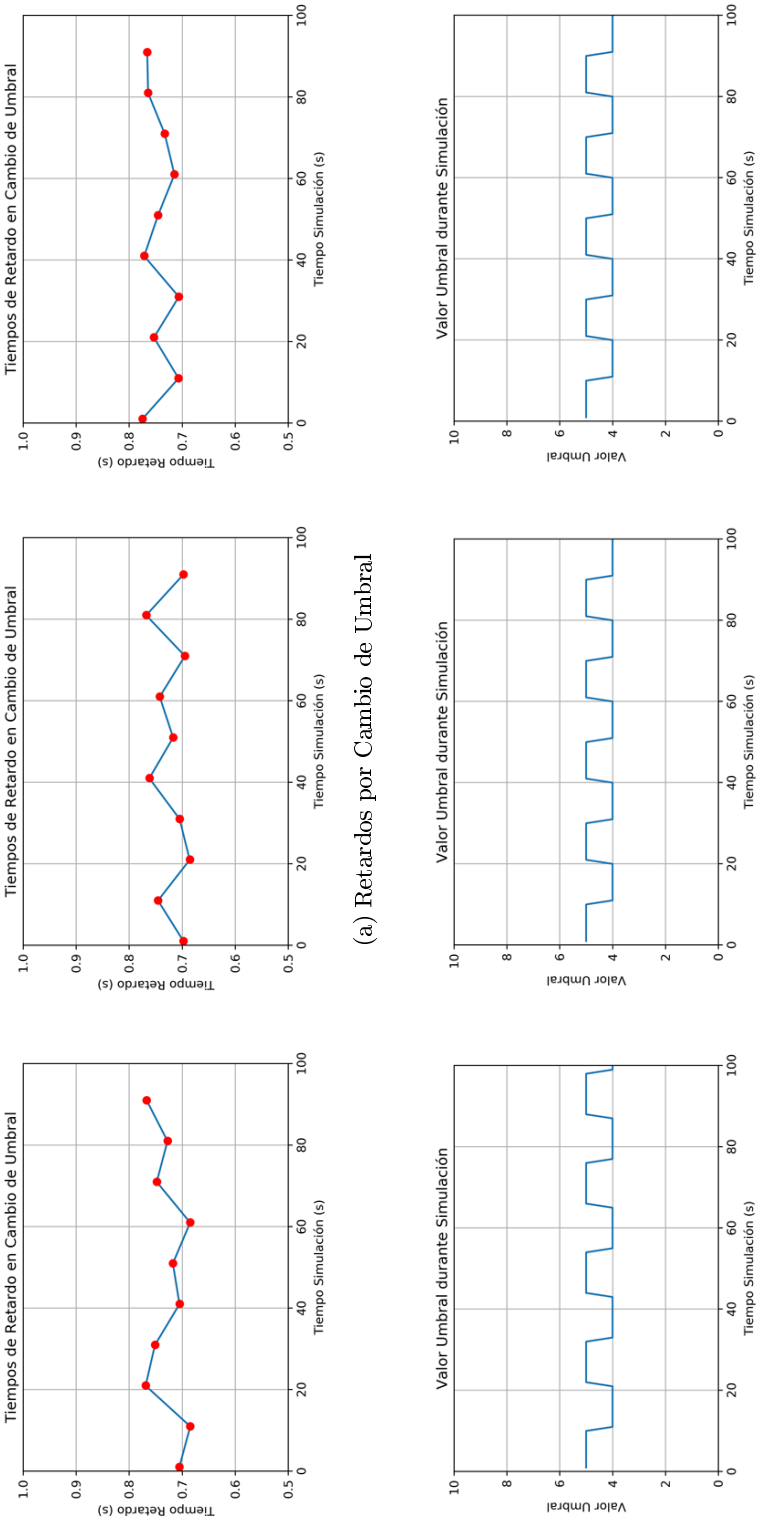
misión en las condiciones habituales, ya que se desea poner al sistema en un estado crítico a propósito. Además, por el diseño del algoritmo de cambios en el umbral, nunca se produce un cambio de más de una unidad hacia arriba o hacia abajo, ya que después de cambiar, el sistema espera el tiempo estipulado para volver a comprobar la carga mental de trabajo del operador antes de subir o bajar otra unidad dicho umbral. El marco experimental también almacena el tiempo que tarda cada cambio en plasmarse en el interfaz (computado desde que el nuevo umbral y el disparador son recibidos por el interfaz hasta que se activa el refresco de pantalla que mostrará los cambios). Se ha elegido un valor de oscilación del umbral entre 4 y 5 elementos, ya que esto hace que se muestre por pantalla entre 4 y 5 elementos por cada vehículo. Esto hace que se muestren total de 12 a 15 elementos, lo que se considera un numero medio-alto de elementos sin que el operador sufra un exceso de carga de trabajo. Como se ha demostrado en los experimentos realizados en la Sección 3.2, el sistema no sufre retardos en el refresco de pantalla debidos al exceso de vehículos mostrados hasta la monitorización de un numero simultáneo de ellos que se sitúa muy por encima de lo que se dará en las condiciones habituales producidas en una de las misiones para las que este tipo de CCT ha sido diseñado. Por lo tanto, no se considera necesario incluir en el interfaz un numero mayor de vehículos ni de elementos gráficos durante el presente experimento.

4.5.3.2. Resultados

Tras la ejecución de las tres iteraciones del experimento, se han analizado los datos, que se muestran en la Tabla 4.2. En ella podemos ver un resumen del número de cambios de umbral ordenados por el controlador de simulación y los ejecutados con éxito en el interfaz. También podemos ver el tiempo medio de retardo desde que se reciben los datos sobre el nuevo umbral hasta que éstos se muestran en la vista del interfaz gráfico y su desviación estándar respecto a dichas medias y los tiempos totales de retardo de los 10 cambios en cada ejecución. Además, en la penúltima columna indica el tiempo máximo de retardo de cada una de las tres ejecuciones, señalando en rojo el tiempo máximo de los 30 cambios.

Se considera que los resultados obtenidos son adecuados, ya que se han ejecutado correctamente todos los cambios en el umbral de transparencia (alternando el número de elementos gráficos mostrados por cada vehículo entre 4 y 5 en intervalos de 10 segundos). Esto indica que los algoritmos implementados y los módulos software implicados en la realización de estos cambios (elementos gráficos, vectores de prioridad, etc.) funcionan correctamente incluso en situaciones de máxima exigencia (cambios continuados en intervalos muy cortos), sometiendo al sistema a una frecuencia de cambios mucho más alta de la que se dará habitualmente. Además, los tiempos de retardo para mostrar la nueva configuración de elementos gráficos son cortos (el tiempo máximo de las 30 ejecuciones es 0,7750 seg.), siempre por debajo de un segundo, a pesar de que es necesario realizar estos cambios sobre los elementos visuales de tres vehículos. Esto permite que el operador se concentre en los elementos gráficos relevantes, ya que el tiempo de refresco es prácticamente inapreciable para él, situándose entre siete y ocho décimas de segundo por cambio. Si el equipo en el que se ejecuta el CCT fuese más o menos potente, los tiempos podrían disminuir o aumentar ligeramente. A pesar del posible aumento de tiempo, se considera que el margen es suficiente como para que no afecte al operador durante el desarrollo de sus tareas de supervisión.

En las gráficas de la Figura 4.8, se muestran los tiempos de retardo de los cambios de umbral y el valor de umbral a lo largo del tiempo. En la Figura 4.8a se encuentran los tiempos de cada una de las ejecuciones de cambio ordenadas por el número de la simulación (1, 2 y 3). En la Figura 4.8b encontramos el valor del umbral que se introduce en cada instante de tiempo para cada simulación. Gracias a la realización de este experimento, se ha podido determinar que los cambios en el umbral se llevan a cabo de forma robusta (aunque se someta al interfaz adaptativo a una frecuencia elevada de cambios) y eficaz (al haberse ejecutado la totalidad de cambios solicitados al sistema). Además, los tiempos de ejecución son adecuados para que un operador consiga mantener la atención centrada en los elementos gráficos que está supervisando, ya que se encuentran por debajo de un segundo en todos los casos, lo que apenas será apreciable por parte del operador mientras realiza una misión.



(a) Retardos por Cambio de Umbral

(b) Valor del Umbral

Figura 4.8: Gráficas de los Resultados del Experimento de Cambio de Umbral

4.5.4. Experimento con los Mecanismos de Adaptabilidad

Otro de los elementos clave que se debe probar en el interfaz gráfico del CCT es la capacidad de auto-adaptarse en tiempo de ejecución, es decir, cambiar el lugar de los elementos gráficos que se muestran por cada vehículo en base a su relevancia para un instante determinado de la misión. Esto se consigue mediante los mecanismos de adaptabilidad implementados en el interfaz, que deben ser puestos a prueba para demostrar que el visor del CCT desarrollado tiene la capacidad de alterar la posición de sus elementos sin ocasionar errores ni retardos prolongados en el refresco de los datos.

4.5.4.1. Diseño del Experimento

En este caso se trata de determinar si el interfaz gráfico es capaz de ejecutar los cambios en tiempo de ejecución mediante un experimento de repetitividad que se realizará con el marco experimental descrito en la sección 4.5.1. El objetivo del experimento es controlar los cambios en el orden de los elementos gráficos mostrados por cada vehículo y comprobar si estos cambios se muestran correctamente en la vista del interfaz gráfico y se llevan a cabo en un tiempo adecuado. Como en los experimentos anteriores, se supone una conexión ideal con suficiente ancho de banda y sin retardos ni interrupciones para el envío de datos que se mostrarán en los elementos gráficos que cambiarán de lugar en la vista del interfaz.

Para llevar a cabo este experimento, se realiza una ejecución del marco experimental 4, que consta de 3 instancias de vehículos autónomos que enviarán paquetes de datos cada segundo para ser mostrados en el interfaz. Cada uno de estos paquetes contiene 7 datos que deben ser mostrados en 7 elementos gráficos, de forma que se maximice el número de vehículos y de datos que deben ser alterados simultáneamente para someter al interfaz a una carga de trabajo mayor de la que se produce habitualmente durante el transcurso de una misión. Esto es debido a que por cada disparador de cambio se deben mover los 21 elementos gráficos simultáneamente mientras que, en condiciones normales, en un instante

Tabla 4.3: Resultados del Experimento de Rotación de Elementos Gráficos

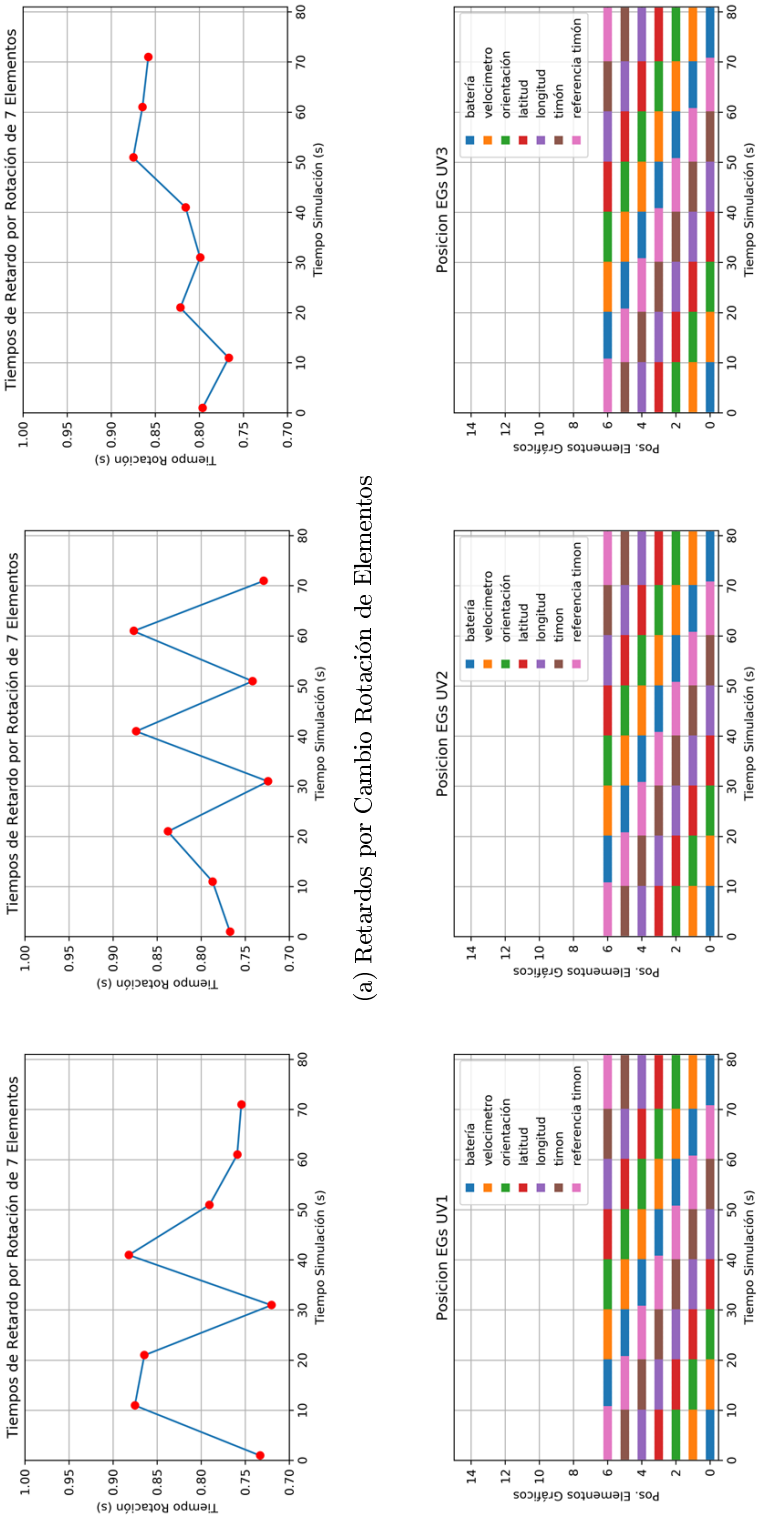
# UV	ROTACIONES	ROTACIONES EJECUTADAS	% ROTACIONES MOSTRADAS	RETARDO MEDIO (seg.)	σ	RETARDO MAX. (seg.)	RETARDO TOTAL (seg.)
1	8	8	100	0,7973	0,0624	0,882	6,379
2	8	8	100	0,7921	0,0588	0,876	6,337
3	8	8	100	0,8245	0,0356	0,874	6,596

de la misión lo habitual será mover de 1 a 3 elementos gráficos por vehículo. Además, el orden de colocación no influye en el retardo producido, ya que el algoritmo actúa de la misma forma en cualquiera de los casos. Para llevar a cabo este experimento, el controlador de simulación envía cada 10 segundos nuevos valores que provocan el cambio del ratio de prioridad de los elementos gráficos mostrados por cada vehículo, de tal forma que cada elemento gráfico en el visualizador tenga que rotar una posición. Esta rotación se realiza 8 veces, de tal forma que al finalizar la ejecución, los elementos gráficos de todos los vehículos se encuentren en la posición de inicio. Además se almacenan los valores de retardo por cada una de las rotaciones de elementos, computadas desde que se recibe un paquete de datos hasta que se finaliza la ejecución del algoritmo de actualización de la vista que mostrará el cambio por pantalla.

4.5.4.2. Resultados

En la Tabla 4.3 se muestra un resumen de las ordenes de rotación enviadas por el controlador para cada vehículo, la cantidad de rotaciones ejecutadas y el porcentaje de rotaciones mostradas con éxito. Además, se puede observar el tiempo de retardo medio por rotación en cada vehículo, la desviación estándar que indica la dispersión de dichos tiempos respecto a la media y los retardos máximos en una rotación de cada vehículo. En la última columna se puede encontrar el tiempo total que se tardó en realizar las 8 rotaciones de elementos gráficos de los tres vehículos.

El total de rotaciones enviadas se ejecutó y mostró correctamente en los tres vehículos, por lo que se considera que el experimento ha tenido éxito (porque el operador no ha dejado de ver ninguno de los cambios solicitados). En los tres grupos de elementos gráficos (uno por vehículo) se realizaron las 8 rotaciones en las que cada elemento se movía un puesto hasta quedar, tras la octava, en la posición inicial. Esto indica que tanto los elementos de la estructura (cada módulo implicado en un movimiento) como los algoritmos implementados en el interfaz reaccionan correctamente, pudiéndose extraer la conclusión de que los mecanismos auto-adaptativos de la vista de nuestro interfaz gráfico funcionan correctamente. Además, el tiempo que tarda en ejecutarse cada rotación son bastante pequeños a pesar de que el sistema esta tratando simultáneamente los datos recibidos de forma asíncrona de 3 vehículos autónomos. Estos tiempos permanecen en todo momento por debajo de un segundo (el máximo se sitúa en 0.882 seg.), por lo que el operador apenas notará los cambios. Además, se considera que este experimento se ha realizado en unas condiciones desfavorables (o al menos no favorables) ya que el hardware en el que se ejecuta el CCT no destaca por su potencia (doble núcleo a 1.2Ghz y 8Gb RAM), por lo que es probable que cuando se ejecute un equipo más potente los tiempos mejoren. En cualquier caso se considera que los cambios se realizan en un tiempo razonable, hecho que evitará que el operador pierda el foco de aquellos datos que está monitorizando.



(a) Retardos por Cambio Rotación de Elementos

(b) Valor del Umbral

Figura 4.9: Gráficas Resultados Experimento de Rotación de Elementos

Adicionalmente, en las gráficas de la Figura 4.9 se recogen los datos de tiempos y posiciones de los elementos gráficos en cada instante de la simulación. En primer lugar, en la Figura 4.9a se muestran los tiempos de retardo de cada rotación producida en cada uno de los vehículos involucrados (de izquierda a derecha respectivamente, para el UV1, UV2 y UV3). Por otra parte, en la Figura 4.9b se presenta un gráfico sobre la posición, dentro de los vectores de prioridad, de cada elemento de cada vehículo. En este caso, al tratarse de una simulación, los ratios de prioridad han sido asignados de tal forma que se logre la rotación deseada. Los elementos de mayor prioridad se encuentran en la parte superior de la gráfica e irán descendiendo según su prioridad hasta la posición 0, la de menor prioridad. Tras la ejecución de este experimento, se llega a la conclusión de que los cambios en la posición de los elementos en tiempo de ejecución se llevan a cabo de forma robusta y eficaz, además de ser realizados en un tiempo suficientemente pequeño como para no molestar al operador que supervisa la misión. Por lo tanto, se considera que las mecánicas de adaptabilidad pueden ser utilizadas en el CCT durante las pruebas de campo reales.

4.6. Caso de Uso: Adaptación del Software a Nivel Visual

En el capítulo anterior (Sección 3.3), propusimos un caso de uso donde nos concentramos en describir el comportamiento del centro de control y su infraestructura adaptativa desde el punto de vista de la arquitectura software y de cómo sus módulos principales se reconfiguran automáticamente en tiempo de ejecución mientras se realizaban una serie de maniobras. A continuación detallaremos otro experimento enfocando la atención en la adaptabilidad del interfaz gráfico y en los efectos que se producen en la vista cuando el motor dinámico activa los mecanismos de adaptación y transparencia implementados en el controlador durante diferentes etapas de la misión.

La misión elegida se desarrolla durante un experimento de campo que consiste en desplegar un equipo de dos barcos autónonomos (USV1 y USV2) y un

cuatrirrotor (UAV) para búsqueda y contención de un vertido contaminante en el agua. El UAV realiza las labores de búsqueda del vertido, mientras que los USVs llevan a cabo una maniobra cooperativa para encerrar el vertido en el interior de una red especialmente diseñada para ello. Para realizar el experimento de campo, desplegamos dos USVs físicos en el agua y los comandamos hasta que alcancen un punto de paso predeterminado. Una vez que ambos USVs han alcanzado el punto de paso, el UAV despega desde la plataforma de aterrizaje instalada en la cubierta del USV1, y trata de localizar el vertido realizando una maniobra de búsqueda en zig-zag, para retornar posteriormente a la posición del USV1 y aterrizar de nuevo sobre su plataforma. Debido a que el aterrizaje sobre una plataforma móvil e inestable sobre el agua se considera una maniobra de alto riesgo para el UAV, para evitar posibles accidentes durante las primeras fases de integración del equipo de vehículos autónomos se utiliza un modelo simulado en tiempo real del cuatrirrotor. Además, sacando partido del modelo simulado de comportamiento y sensorización del cuatrirrotor, podemos simular también la detección de un vertido contaminante durante el desarrollo de los experimentos, que se realizan por razones obvias en un entorno no contaminado. Finalmente, después de simular el retorno del UAV al punto donde se encuentra el USV1 y su aterrizaje en la plataforma, los USVs realizan una maniobra para rodear y encerrar el vertido en el interior de la red, donde en condiciones reales permanecería hasta que fuese extraída por personal cualificado.

Para la realización de esta misión se cuenta con dos USVs físicos, USV1 y USV2, de 4 y 5 metros de eslora respectivamente, ambos construidos por el CEHIPAR (ver Figura 4.10). Ambos están controlador por un PC embebido Beckhoff C6920 con TwinCAT, un sistema en tiempo real bajo Windows. El USV1 está equipado con una IMU (unidad de medición inercial) VectorNav VN-200 con GPS integrado y el USV2 lleva una IMU CodaOctopus F180, también con GPS integrado. El UAV simulado modela un dron MD4-200 de MicroDrones (nuestro UAV físico, también desplegado en la Figura 4.10 sobre la plataforma de aterrizaje instalada en el USV1). Dicho cuatrirrotor será simulado en otra computadora Beckhoff C6920 con TwinCAT. Las comunicaciones entre los UVs

Figura 4.10: Hardware del Experimento Real que Involucra a 2 USV y 1 UAV

y el CCT, este último desplegado en un portátil (que tiene un procesador de doble núcleo a 1.2GHz y 8Gb RAM) con el sistema operativo Windows 8.1, se realizan a través de una red inalámbrica Wi-Fi de 1.5 a 2 Km de rango, formada por 3 Ubiquiti PicoStation M (en cada uno de los UVs reales y en la computadora de simulación) y una antena Ubiquiti NanoStation M2 (conectada al CCT) que hará las veces de punto de conexión.

Los objetivos de este experimento son: 1) verificar que los USVs físicos pueden realizar correctamente sus diferentes maniobras, 2) poner a prueba el sistema de comunicaciones y el intercambio de información entre los vehículos reales y simulados, 3) comprobar si un operador es capaz de supervisar y gestionar todas las tareas derivadas de la misión, y 4) analizar la viabilidad del interfaz de usuario adaptativo y las políticas de Transparencia y Adaptabilidad en el ámbito de esta misión, no solo durante la transición de la vista entre las fases con 2 o 3 vehículos, sino también desde el punto de vista de la carga mental de trabajo del operador.

En la primera fase del experimento, el USV1 y el USV2 son desplegados en el agua para comenzar una maniobra cooperativa donde el USV2 hará las

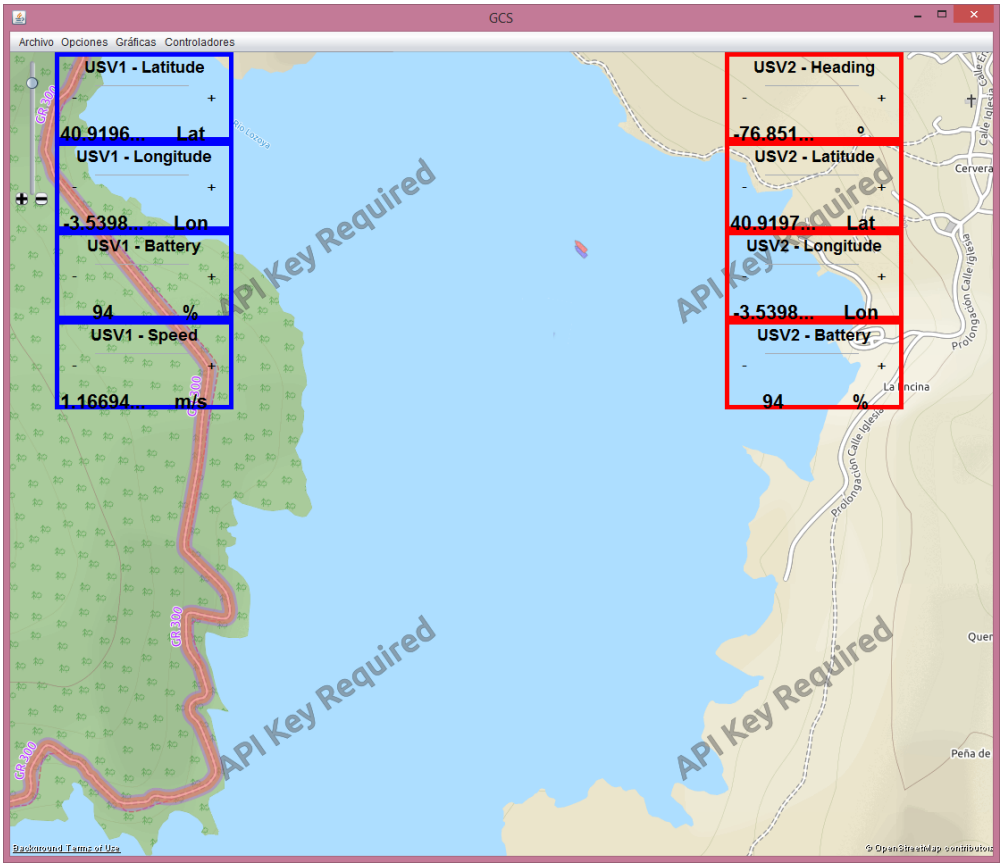


Figura 4.11: Configuración Inicial del GUI de la Vista (Fase 1)

veces de líder y el USV1 de seguidor. Durante esta fase de la misión, ambos USVs se dirigen a un punto de paso definido manteniendo una distancia de seguridad apropiada entre ellos, mientras en la vista del interfaz se muestra la información presentada en la Figura 4.11. Esta configuración se debe a que el motor dinámico configura el umbral de transparencia en 4, mostrando sólo 4 elementos gráficos por vehículo, los pertenecientes al USV1 en azul a la izquierda y los pertenecientes al USV2 en rojo a la derecha. En este instante de la misión, la carga mental de trabajo estimada para un único operador que supervisa dos vehículos es media (al tener solamente dos UVs activos desarrollando una misión en la que ya se tiene experiencia por haber sido probada en varias ocasiones).

Una vez los USVs se paran en el punto de paso objetivo, comienza la fase 2 de la misión. En esta parte del experimento, el USV1 y el USV2 permanecen parados y el UAV simulado en tiempo real despega, partiendo de las coordenadas proporcionadas por el USV1. A continuación, el UAV comienza a realizar una maniobra de búsqueda en zig-zag con el propósito de localizar el vertido contaminante para su posterior contención. Finalmente, el UAV solicitará al USV1 su posición para retornar y realizar el aterrizaje simulado sobre la plataforma de dicha embarcación. Durante esta fase de la misión se producen varios cambios en el sistema.

En primer lugar, cuando el UAV se incorpora a la misión: 1) el sistema autoadapta su estructura para incorporar al UAV en el CCT, y 2) el motor dinámico, que está monitorizando la misión y al operador en tiempo de ejecución, detecta la presencia del nuevo vehículo, añadiendo sus elementos gráficos en la vista (posición en el mapa y datos relevantes de telemetría) tal y como se muestra en la Figura 4.12. En este instante, como el Umbral de Transparencia sigue fijado a 4, se muestran, además de los 4 elementos gráficos que ya aparecían de cada USV, los 4 elementos gráficos más relevantes del UAV en esta fase de la misión (abajo en el centro de la imagen). La cantidad de datos mostrada en la vista y el estrés relacionado con el despegue del UAV (un punto crítico en la misión) provocan un aumento del tiempo de reacción (Tr) y del tiempo transcurrido (Tt) del operador. Esto hace que el motor dinámico, que monitoriza estos datos constantemente a lo largo de la misión, aumente su estimación de la carga mental de trabajo del operador y consecuentemente reduce el umbral de transparencia del interfaz gráfico. El motor dinámico envía el nuevo umbral y dispara los mecanismos de transparencia en el controlador, que reduce el número de datos mostrado en la vista por cada UV de los 4 elementos gráficos (anterior valor del umbral) que se ven en la Figura 4.12 a los 3 (actual valor del umbral) que aparecen en la Figura 4.13. Gracias a estos cambios, el operador tiene menos elementos en los que centrar su atención, y las medidas de carga mental de trabajo, Tr y Tt se estabilizan y comienzan a decrecer, permitiéndole continuar la misión como supervisor único. Además, durante el desarrollo de esta fase y según se van

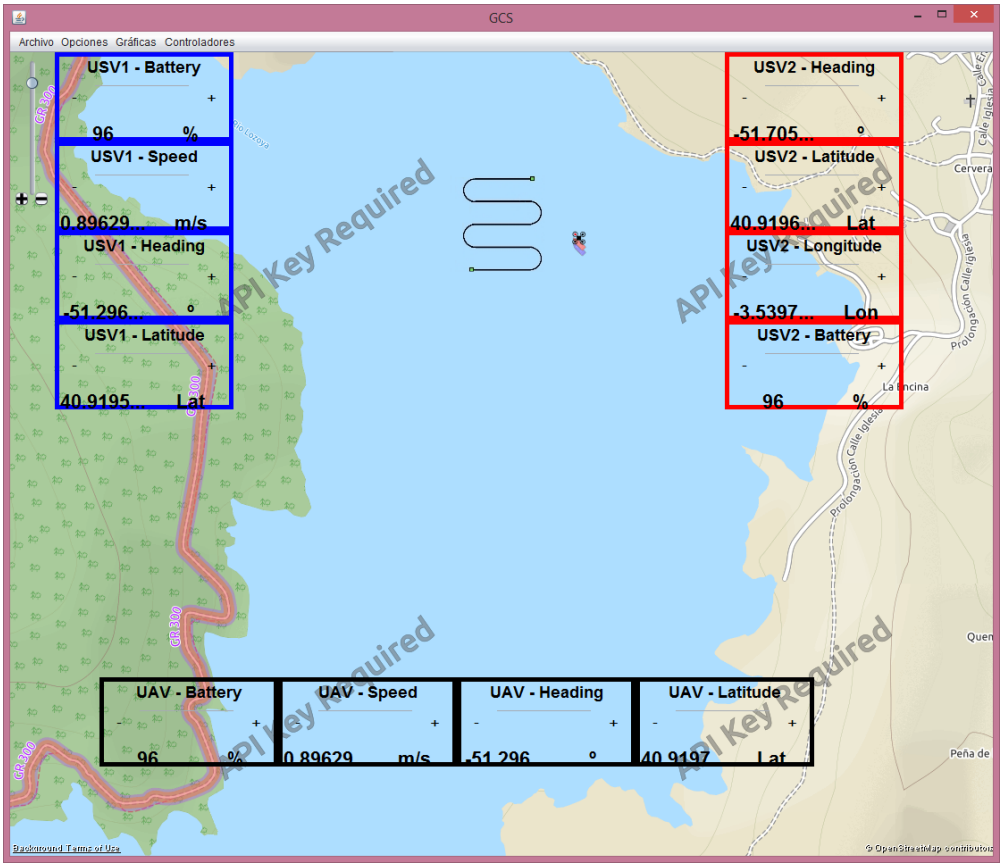


Figura 4.12: Configuración del GUI Antes del Despegue del UAV (Fase 2a)

sucediendo las tareas que deben ser completadas, el sistema envía información que modifica los ratios beneficio/coste de los elementos gráficos y dispara los mecanismos adaptativos en el controlador. Esto hace que los elementos gráficos se reordenen en los vectores de prioridad de cada vehículo (posiciones más altas para USVs y más a la izquierda para UAV), respondiendo a los nuevos valores recibidos y, finalmente, refrescando la pantalla con éstos colocados en sus nuevas posiciones en la vista.

Después de que el UAV realice la maniobra de aterrizaje simulado, el planificador del CCT calcula una ruta para los USVs a partir de los puntos de paso enviados por el UAV durante su maniobra de localización. Con esta información los USVs comienzan la maniobra que encerraría el vertido contaminante en

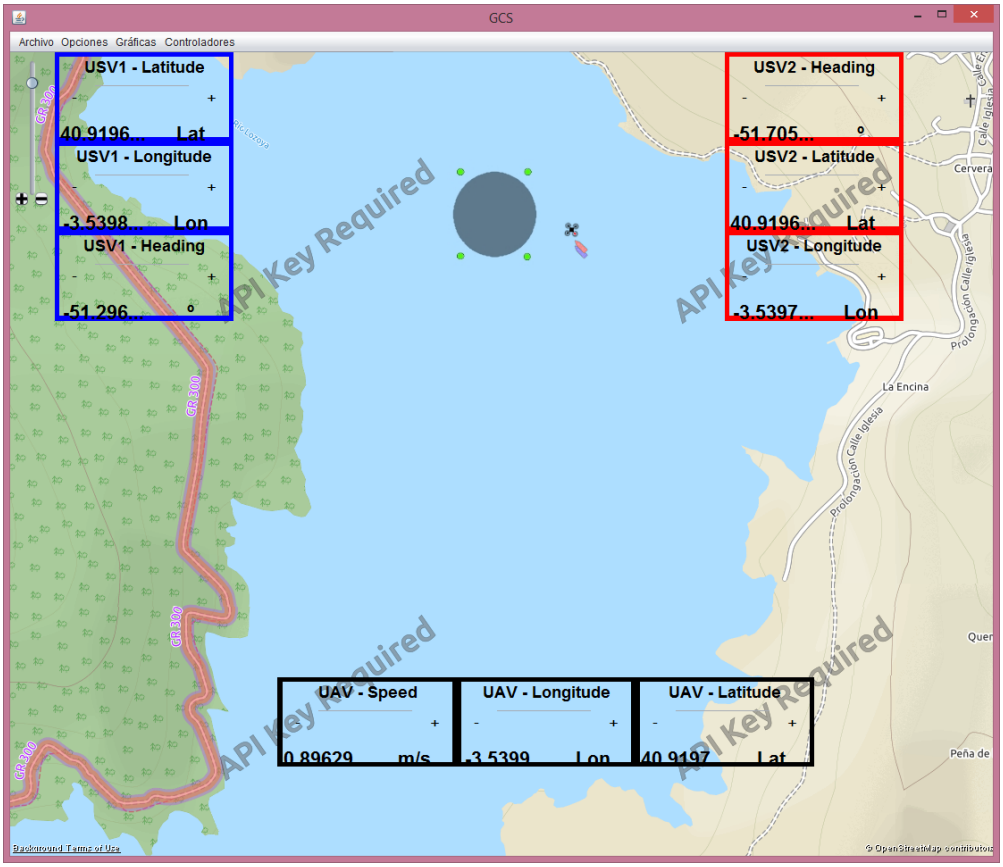


Figura 4.13: Configuración del GUI Despues del Despegue del UAV (Fase 2b)

la red. Durante este experimento, esta maniobra se realizará sin que los USVs permanezcan unidos por la red de contención, ya que la misma somete a los vehículos a unas fuerzas de arrastre que cambian por completo la dinámica de los algoritmos de control. Después del aterrizaje y apagado del cuatrirrotor autónomo simulado, el sistema retorna a su configuración inicial, donde el CCT solo supervisa la labor de dos USVs. En este punto, los elementos gráficos del UAV dejan de mostrarse en la vista de nuestro interfaz gráfico. Esto provoca un descenso en los datos a monitorizar y en las tareas derivadas de la supervisión de los vehículos. Además, al haber aterrizado ya el dron, el operador queda al cargo sólo de la maniobra de contención del vertido, ya probada con anterioridad. Estos factores unidos al descenso de tareas y elementos gráficos en los que centrar

su atención, provoca también un descenso en la carga mental de trabajo, el Tr y el Tt del operador, limitando su fatiga y moderando su estrés. Estos cambios, de nuevo monitorizados por el motor dinámico, hacen que éste vuelva a subir a 4 el umbral de transparencia, activando los mecanismos del controlador para volver a modificar la vista, que queda configurada de nuevo de forma similar a la producida durante la fase 1 de la misión (Figura 4.11). Adicionalmente, al igual que en fases anteriores, los datos recibidos sobre las tareas en ejecución, modifican el ratio beneficio/coste de los elementos gráficos durante el transcurso de las tareas, haciendo que el controlador los reordene en función de su peso específico en cada instante de la misión.

4.7. Conclusiones

El interfaz gráfico es uno de los componentes más importantes de un centro de control de tierra, ya que debe proporcionar a los operadores las herramientas necesarias para supervisar un equipo de vehículos autónomos y además facilitar su labor por medio de un diseño intuitivo y ergonómico. Además, cuando en una misión se deben controlar varios vehículos heterogéneos de forma simultánea, la carga mental de trabajo del operador crece, aumentando su fatiga e impidiéndole realizar correctamente su labor, por lo que es muy útil que el interfaz incluya algún tipo de ayuda al operador que trate de mitigar los efectos adversos de la sobrecarga de trabajo mental.

En este capítulo se ha propuesto un interfaz gráfico adaptativo modular diseñado siguiendo las pautas arquitectónicas detalladas en el Capítulo 3. Su diseño ergonómico implementa dos mecanismos para reducir la carga de trabajo mental del operador: políticas de transparencia (responsables de ocultar la información menos relevante para liberar la atención del operador) y de adaptabilidad (que mueven los elementos gráficos con datos más importantes a los lugares de mejor visibilidad de la pantalla para que el operador los localice fácilmente). Además, el interfaz gráfico se ha estructurado en 4 módulos: 1) el motor adaptativo dinámico (encargado de monitorizar los datos y tomar las decisiones que auto-adaptarán

el interfaz) 2) el controlador (responsable de realizar los cambios indicados por el motor adaptativo), 3) la vista (que muestra los elementos gráficos seleccionados por los módulos anteriores) y 4) la base de datos de configuración (que almacena de forma persistente datos sobre la configuración del interfaz y las preferencias ergonómicas del operador).

Para evaluar el funcionamiento del interfaz del CCT desarrollado y comprobar si las políticas de ayuda al operador puede ser aplicadas, se ha diseñado un nuevo marco experimental que pone de manifiesto tres capacidades clave del interfaz: 1) la de visualizar correctamente la información recibida, 2) la de aplicar las mecánicas de transparencia, y 3) la de aplicar las mecánicas de adaptabilidad. Las pruebas realizadas permiten determinar que el interfaz gráfico es robusto y eficaz, ya que permite mostrar los datos y alarmas al operador sin que se produzcan pérdidas o retardos notables. Además se ha podido comprobar que los mecanismos implementados para las ayudas al operador son capaces de adaptar los elementos gráficos de la vista en tiempo de ejecución aún siendo sometidos a unas condiciones notablemente más críticas de las que se producen durante una misión que se desarrolle en las condiciones habituales para las que ha sido diseñado el CCT adaptativo.

Finalmente, se ha puesto a prueba el funcionamiento del interfaz gráfico del CCT en la realización de experimentos de campo, para lo que se analiza uno de los experimentos llevados a cabo en el marco del proyecto SALACOM. Durante este experimento se ha podido comprobar que el interfaz, además de ser robusto y eficaz en una misión que involucra dos USVs físicos y un UAV simulado, cuenta con unos mecanismos de adaptabilidad y transparencia igualmente robustos y eficaces que han permitido al interfaz auto-adaptarse tanto ante los cambios producidos en el equipo de vehículos autónomos como ante aquellos cambios producidos por los mecanismos que activan las políticas de ayuda al operador, permitiendo al operador que supervisa la misión completar sus tareas en un tiempo razonable mientras mantenía estables sus niveles de carga mental de trabajo, fatiga y estrés.

Capítulo 5

El CCT como Banco de Pruebas: Entorno Simulado de Ayudas al Operador

"Lo perfecto es enemigo de lo bueno".

Voltaire

El centro de control de tierra desarrollado durante esta tesis cuenta con una arquitectura distribuida modular que permite alterar, incluso en tiempo de ejecución, de forma sencilla sus componentes software sin necesidad de rediseñar la estructura ni reprogramar otras partes que no sean aquellas que quieren cambiarse. Esta versatilidad, sumada a la capacidad de incorporar agentes simulados (vehículos autónomos, sensores, etc.), convierte a nuestro CCT en un banco de pruebas muy útil, ya que podemos alterar el comportamiento de sus componentes modificando alguno de los módulos que contienen sus algoritmos o políticas de ejecución para probar si dichos cambios mejoran el rendimiento del software o influyen positivamente en el del operador.

Con el objetivo de sacar el máximo partido a las capacidades del CCT como banco de pruebas y plataforma de entrenamiento, se ha creado un marco expe-

rimental capaz de recrear misiones que permiten someter a un operador a los mismos estímulos y condiciones a los que estaría expuesto durante el desarrollo de un experimento de campo. Esto se logra a través de la combinación de la simulación de vehículos autónomos (tanto en tiempo real como a través de los datos guardados de misiones ya realizadas para un mayor control de los experimentos) y la generación (aleatoria o prefijada en instantes concretos del experimento) de eventos y tareas relacionadas que emularán las situaciones a las que se desea someter al operador durante la supervisión de los vehículos involucrados en la misión.

Esta potente herramienta de simulación nos permite realizar múltiples tipos de pruebas con diferentes objetivos, entre los que destacan los siguientes. Por una parte, permite realizar entrenamientos que ayudan al operador a familiarizarse tanto con el software del CCT y sus herramientas, como con el desarrollo de una misión y las tareas que en ella debe realizar. Esto le prepara para un mejor desarrollo de su labor, atenúa los efectos de la curva de aprendizaje y le enseña a reaccionar ante situaciones críticas poco comunes que, gracias al simulador, pueden probarse sin riesgo. Por otra, ha posibilitado la toma de medidas durante la realización de las pruebas iniciales con un grupo experto de referencia que nos ha ayudado a fijar los parámetros iniciales que se utilizan para controlar las políticas de ayuda al operador implementadas. Ademas, sirve para poner a prueba nuevos mecanismos de ayuda al operador sin que ello ponga en peligro el desarrollo de una misión real o el hardware que en ella se utilize. Finalmente, también permite elaborar pruebas sobre un grupo principal de individuos con el objeto de verificar si las ayudas al operador implementadas en el CCT adaptativo mejoran el rendimiento de estos durante el desarrollo de una misión que involucre equipos de vehículos autónomos heterogéneos.

En las siguientes secciones describiremos los cuantificadores y requisitos que hemos utilizado para valorar el desempeño de las ayudas al operador que han sido probadas en esta tesis. A continuación se describirán las modificaciones efectuadas para utilizar el CCT adaptativo como banco de pruebas y plataforma de

entrenamiento, así como el diseño de los experimentos realizados y los resultados obtenidos en los mismos.

5.1. Criterios y Requisitos de Validación

El trabajo de un centro de control en una misión compleja que involucre equipos cooperativos de vehículos heterogéneos no se limita a mostrarle al operador la información y los controles para supervisarla e intervenir en caso de ser necesario, sino que debe mostrar la información de forma intuitiva y ergonómica para facilitar la labor de sus usuarios. A pesar de estas características (que deben ser implementadas en el interfaz gráfico de usuario -GUI- de un CCT) es habitual que varios operadores colaboren en las labores de supervisión de este tipo de misión, dividiéndose las tareas de monitorización y control entre dos o mas personas. Cuanto mayor sea el número y la complejidad de los vehículos involucrados, mayor será la cantidad de datos y controles a supervisar y, por tanto, mayor será el numero de operadores necesarios para llevar a cabo la misión. Por lo tanto, aunque es razonable querer minimizar la cantidad de operadores involucrados en la medida de lo posible, cuanto mayor sea la cantidad de datos y tareas que debe supervisar un único operador, mayor será la carga mental de trabajo a la que esté sometido. Esto implica un aumento de su estrés y fatiga, lo que influirá negativamente en su rendimiento, haciendo que aumente su tiempo de reacción y el tiempo que emplea en realizar cada una de las tareas, y provocando incluso que desatienda alguna de ellas u ocasionando graves trastornos en el cumplimiento de los objetivos de la misión. Además, una vez que un operador se encuentra fatigado, la única forma de que su rendimiento vuelva a niveles aceptables es que descanse, es decir, debe ser sustituido por otro, aumentando así el número de operadores necesarios para llevar a cabo la misión. Por estas razones, lograr el diseño e implementación de un centro de control que, además de mostrar los datos y controles de forma ergonómica e intuitiva, sea capaz de integrar mecanismos de ayuda al operador que hagan decrecer su carga mental de trabajo, se ha convertido en uno de los grandes retos actuales en este campo.

Para el CCT implementado durante el desarrollo de esta tesis, se han elegido dos mecanismos que ayudarán a mantener la carga mental de trabajo en niveles aceptables. En primer lugar se utiliza la adaptabilidad, que permite cambiar la posición de los elementos gráficos que muestran los datos, colocando aquellos más relevantes en cada instante de la misión en las posiciones de mayor prioridad de la pantalla. En segundo lugar, contamos con mecanismos de transparencia, que permiten ocultar parte de la información mostrada de cada vehículo (aquella menos relevante en un instante dado de la misión) para ayudar al operador, que podrá centrarse en aquellos elementos gráficos y tareas más importantes. Como se detalla en la Sección 2.2.3, estas técnicas de ayuda al operador han sido estudiadas en trabajos previos (como los de Mercado et al. (2016); Helldin (2014); Chen et al. (2014)), donde se ha observado sus efectos sobre el rendimiento y carga mental de trabajo del operador mediante diferentes procedimientos. Sin embargo, estos estudios han sido realizados, de forma teórica, en otros ámbitos o con otros objetivos, por lo que en este capitulo determinaremos si estas técnicas son eficaces cuando se aplican a un CCT como el nuestro durante la ejecución de misiones complejas en las que participan varios vehículos heterogéneos.

Para ello, en primer lugar, hemos buscado una forma robusta y eficaz de medir la carga mental de trabajo, principal indicador de la capacidad que tiene un operador de realizar su trabajo y la hemos implementado en el CCT, lo que nos permite medir su valor mediante diversos indicadores durante misiones, tanto reales como simuladas. Aunque los cuantificadores utilizados para medir la carga mental de trabajo han sido descritos en la Sección 2.2.3, y la forma de implementarlos en el interfaz del CCT ha sido detallada en el Capítulo 4, a continuación recordamos los motivos por los que han sido escogidos. Para medir de forma robusta y fiable la carga mental de trabajo se recomienda la realización de una medida subjetiva a posteriori y dos durante la ejecución, una sobre tareas primarias y otra sobre tareas secundarias. Como medida subjetiva a posteriori elegimos el índice de carga de trabajo NASA (NASA Task Load Index, NASA-TLX), al ser uno de los indicadores más utilizados. Durante la misión utilizamos una medida subjetiva (solicitando al operador que indique su nivel de carga de

trabajo en instantes puntuales de la misión), y las medidas indirectas de tiempo empleado (TE) sobre tareas primarias y el tiempo de reacción (Tr) sobre tareas secundarias, ya que estos dos tiempos son muy fáciles de calcular por el CCT.

En segundo lugar, tras implementar en el CCT los mecanismos necesarios para medir la carga mental de trabajo, se ha diseñado y realizado un experimento para demostrar la eficacia de las técnicas de adaptabilidad y transparencia sobre un operador de CCT. Esto no ha resultado fácil, ya que la carga mental de trabajo no depende sólo de los factores relacionados con la forma (posición, color) y el volumen de información, sino que también lo hace de las características propias de cada operador y de su habilidad para realizar mejor unas tareas que otras (por ejemplo un operador puede ser más rápido con los cálculos mentales mientras que otro puede tener mayor capacidad para situarse en el entorno. Además, otros factores externos pueden influir en el rendimiento de un operador, como son el nivel de estrés o la fatiga con los que comienzan su trabajo. Por estas razones, los resultados de cada operador deben evaluarse por separado, ya que es difícil encontrar unos valores genéricos para valorar la eficacia de los mecanismos de ayuda para cualquier operador. Además, debemos realizar un experimento que nos permita someter a cada individuo de forma sistemática a una serie de situaciones, lo que se ha logrado mediante el diseño de una misión que genera de forma determinista una serie de eventos que nos permiten evaluar la carga mental de trabajo del operador en situaciones prefijadas con un número ajustable de tareas. Esto ha sido posible gracias a la capacidad del CCT de integrar vehículos simulados e incorporarlos a un marco experimental como se ha visto en las Secciones 3.2 y 4.5, ya que es muy dificil lograr un experimento determinista de las características necesarias en pruebas de campo debido a la imprevisibilidad del medio en el que se desarrollan las misiones en el proyecto SALACOM (los pantanos de Valmayor y El Atazar). Gracias a este simulador de experimentos, hemos sido capaces de comparar los resultados de un mismo individuo ante las mismas situaciones con y sin ayudas al operador y ver si los resultados mejoran entre ambas ejecuciones del experimento. También nos ha permitido comparar los resultados del rendimiento de los individuos entre sí

cuando son sometidos a una situación bajo las mismas circunstancias, así como realizar una evaluación global de la utilidad de las ayudas, comprobando la cantidad de individuos que mejoran gracias a las ayudas al operador y el número de usuarios cuyo rendimiento permanece igual o empeora.

A lo largo de las siguientes secciones se describirá detalladamente el marco experimental diseñado para la realización de las pruebas sistemáticas sobre operadores reales, el grupo de individuos que serán sometidos a los experimentos y los resultados obtenidos en los mismos.

5.2. Diseño del Simulador: Marco Experimental

Una vez implementado el centro de control de tierra para las misiones del proyecto SALACOM, e integrados en su interfaz gráfico de usuario los mecanismos que permiten poner en práctica las políticas de adaptabilidad y transparencia para ayudar a mitigar los efectos adversos del exceso de carga mental de trabajo en el operador y seleccionados los cuantificadores para medirla, debemos verificar su efectividad mediante el diseño de una serie de experimentos sobre diferentes individuos.

Para ello se ha diseñado el marco experimental que podemos ver en la Figura 5.1, que proporciona al centro de control un ecosistema con las herramientas necesarias para generar distintos entornos simulados con los que se podrá desarrollar las correspondientes misiones, específicamente diseñadas para someter al operador a una experiencia casi idéntica a la que se tiene durante el desarrollo de una misión de campo del proyecto SALACOM. Dichas misiones incluyen un equipo de vehículos autónomos funcionales que envían datos en tiempo real, los datos del entorno donde se desarrolla y las tareas de supervisión derivadas de una misión real (llegada a puntos de paso, desvíos de ruta por perturbaciones externas, valores anómalos en la telemetría, detección de obstáculos, etc.). Este marco experimental se ha diseñado de forma similar al descrito en la Sección 3.2.2, ya que comparte con él algunos de sus módulos principales (control de vehículos y transductor), modificando algunos aspectos como la entrada de datos y agre-

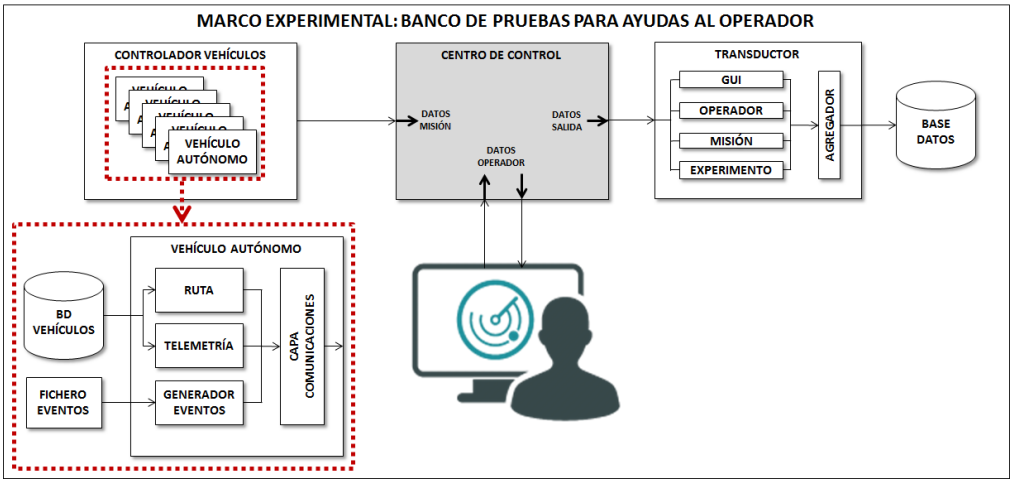


Figura 5.1: Marco Experimental del Banco de Pruebas para Ayudas al Operador

gando al operador, que interactuará con el centro de control, formando parte del sistema bajo estudio y generando datos sobre su rendimiento que han de ser tratados y almacenados por el transductor. A continuación se describe con más detalle cada uno de los módulos que forman dicho marco experimental.

Control de Vehículos es el módulo encargado de simular el desarrollo de una misión en la que se introducirán de uno a tres vehículos autónomos (para esta simulación estarán disponibles los modelos de un UAV y dos USVs), dependiendo del nivel de carga mental de trabajo a la que se quiera someter al operador. Este módulo se encarga de crear las instancias de cada uno de los vehículos involucrados en la misión, controlar el tiempo de duración de la misión y destruirlas cuando ésta acabe. Cada vehículo seguirá, como en cualquier misión real, el protocolo de conexión al sistema explicado en la Sección 3.1.3, con la salvedad introducida en las simulaciones anteriores: la inclusión del vehículo en el sistema ha sido automatizada, eliminando la etapa de aprobación del operador. De esta forma se agiliza el comienzo de las misiones en las que puede haber hasta tres vehículos solicitando su incorporación al sistema. Tras incluir los vehículos en la infraestructura, la misión comienza y el operador tiene a su disposición un CCT prácticamente idéntico al que utiliza en una misión real, salvo porque se le han

deshabilitado algunas operaciones o comandos que no son necesarios para la supervisión/control de la misión simulada.

Vehículo autónomo es el módulo encargado de simular el comportamiento de cada uno de tres vehículos autónomos integrado en el sistema. El diseño base y el funcionamiento de cada vehículo es el mismo, pero los datos que se envían (leídos de un almacenamiento persistente) y los eventos generados son diferentes para cada uno de los vehículos. Además, cada vehículo cuenta con cuatro módulos internos y una base de datos que guarda la información generada por un vehículo autónomo durante una misión real. El módulo de generación de rutas lee los datos guardados sobre la ruta de un vehículo (coordenadas: latitud y longitud) y los transmite a la capa de comunicaciones. El módulo de generación de telemetría lee también de la base de datos toda la información relevante sobre la telemetría (batería, orientación, cabeceo, alabeo, datos del timón, velocidad, etc.) del vehículo obtenida de una misión real previa, para enviarla también a la capa de comunicaciones. El tercer módulo es el encargado de generar los eventos que determinarán las situaciones de la misión a las que se quiere someter al operador (por ejemplo, la detección de un obstáculo o la llegada a un punto de paso). Estos eventos pueden ser prefijados (cargados desde un archivo de configuración que el generador de eventos leerá durante su ejecución) para someter a un grupo de individuos a los mismos estímulos durante una misión. Alternativamente, pueden ser generados al azar mediante una probabilidad establecida para cada evento en base a su frecuencia en misiones reales, cuando queramos que las misiones tengan un componente aleatorio. Además, una misión aleatoria puede reproducirse posteriormente con exactitud gracias a la inclusión de una semilla que fija los sucesos aleatorios asociados a los eventos anteriormente mencionados y muestreados teniendo en cuenta los valores de probabilidad anteriormente mencionados y que se almacenan junto a los datos de la misión. Este conjunto de módulos y los datos que en ellos se generan (o se cargan desde almacenamientos persistentes) nos permite generar un amplio espectro de misiones para evaluar

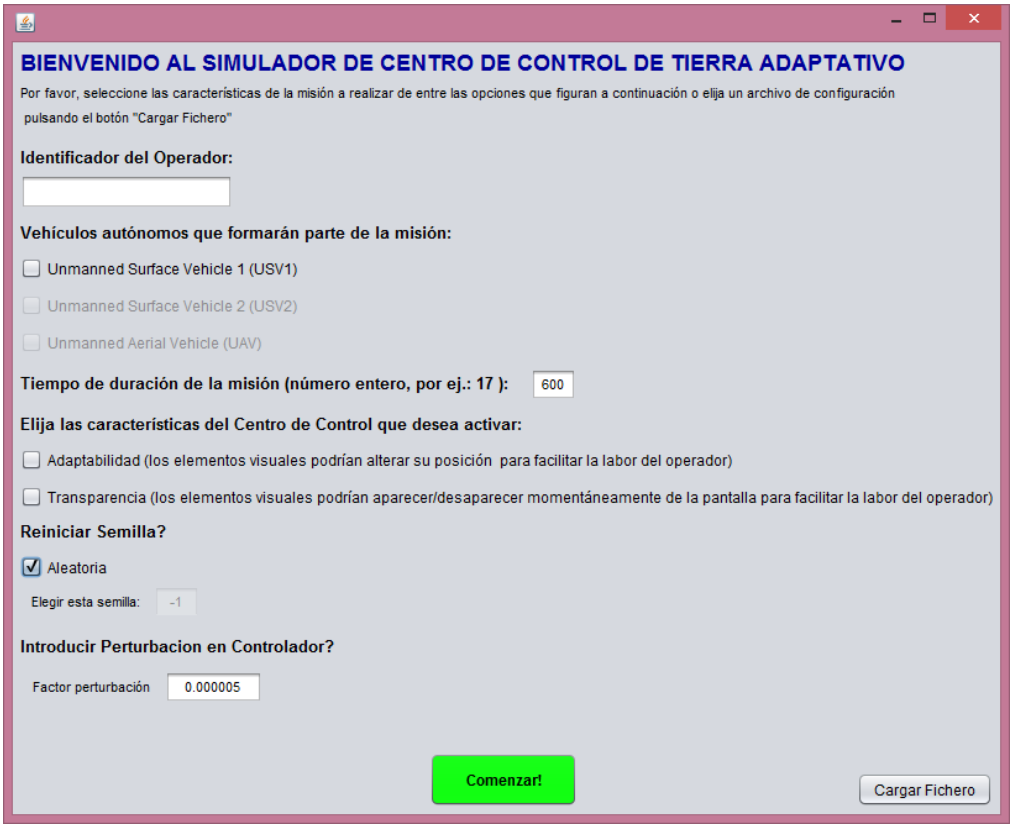


Figura 5.2: Menú de Inicio para Misiones Simuladas

el rendimiento de un individuo sometido a situaciones de mayor o menor carga mental de trabajo. Por último, la capa de comunicaciones está implementada de forma similar a la capa de comunicaciones de un vehículo autónomo utilizado en una misión real. Se encarga de realizar la petición de conexión inicial al sistema (incluida la negociación de conexión) y posteriormente de encapsular los datos recibidos del resto de módulos (ruta, telemetría y eventos). Estos paquetes son enviados al CCT cada segundo para que éste los muestre y plantee al operador los avisos, alarmas y tareas asociadas a cada evento.

Centro de Control y Operador forman el sistema bajo estudio, que estará compuesto tanto por el CCT como por el operador que interactúa con él. El marco experimental permite la recreación de diferentes tipos de misio-

nes para realizar pruebas sobre múltiples aspectos tanto del CCT (ayudas al operador, modificaciones o integración de nuevos módulos, etc.) como del operador (medida de rendimiento, entrenamiento, etc.) que deben verificarse antes de su utilización en misiones reales. Aunque buena parte de la configuración de la misión se realiza previamente y se introduce en el marco experimental mediante los datos que se cargan desde el almacenamiento persistente en los módulos de vehículo autónomo, otra parte se puede modificar al inicio del experimento mediante el menú de simulación de la Figura 5.2. Este menú permite introducir los siguientes parámetros finales para conformar la misión: 1) el identificador del operador, 2) el número de vehículos autónomos implicados en la misión, 3) el tiempo de duración del experimento, 4) las ayudas al operador que desean activarse, 5) una semilla que determina el componente aleatorio de generación de eventos (si se desea), y 6) el nivel de perturbación que quiere introducirse en el controlador de los vehículos autónomos. Mediante este sencillo menú, el diseñador del experimento puede introducir variaciones significativas sobre la base de una misión prediseñada para someter al operador a diferentes estímulos. Por ejemplo introducir más o menos vehículos autónomos somete al operador a una mayor o menor carga de trabajo, introducir una mayor perturbación hace que el operador tenga más dificultades para que los vehículos sigan la ruta establecida o, lo más importante, realizar la misma misión en dos ocasiones, una con las ayudas al operador desactivadas y otra con ellas activas permite verificar el cambio de rendimiento en el operador. Cuando se inicia el programa de la misión simulada, el operador es el encargado de introducir estos datos de entrada al marco experimental mediante el menú de simulación. Estos datos se le indican previamente al operador en función del tipo de misión que se vaya a llevar a cabo. En primer lugar el operador debe introducir su identificador, que sirve para etiquetar los datos de su rendimiento que serán almacenados por el transductor. A continuación, el operador selecciona las casillas de los vehículos que vayan a formar parte de la misión. Debe elegir de uno a

tres entre los vehículos disponibles (USV1, USV2 y UAV). Inmediatamente después, se encuentra un cuadro de texto donde se debe introducir el tiempo de la misión. A continuación el operador selecciona las ayudas que se desea proporcionar al operador (adaptabilidad, transparencia, ambas o ninguna). Después, se encuentra el apartado que permite introducir al azar o manualmente (si se desea) una semilla que controlará la generación de eventos para esa misión. Por último se puede elegir si se desea introducir un pequeño ruido o perturbación en la trayectoria de los vehículos para que éstos sean más propensos a desviarse de la ruta establecida y el operador deba estar más atento para comprobar si la trayectoria del vehículo sigue correctamente los puntos de paso.

Transductor es el módulo que realiza el tratamiento y guardado de los datos en un almacenamiento persistente. Una vez más, se utiliza el módulo transductor estándar que ha sido usado en todos los experimentos anteriores, formado por cuatro canales de recepción de datos y un agregador que se encarga de tratarlos y guardarlos. En el caso de este experimento, se reciben datos por el canal GUI, que se encargará de recibir los datos del interfaz de usuario, del que se guardarán su estado, cantidad y orden de elementos gráficos mostrados y umbral de transparencia. También se reciben datos por el canal del operador, del que se almacena su nivel de carga mental de trabajo subjetiva (solicitada en algunos instantes concretos de la misión), el tiempo empleado en las tareas primarias realizadas y el tiempo de reacción en algunas tareas secundarias. Por el canal Misión se reciben los datos relacionados con la misión, como los eventos relevantes recibidos de cada vehículo, información del entorno y los objetivos cumplidos. Por último, tenemos el canal experimento, por el que únicamente se reciben algunos datos relativos a la configuración del experimento, como la duración, la semilla aleatoria y la perturbación (en los casos en los que esta se aplicase).

5.3. Diseño del Experimento

Una vez descrito el marco experimental para generar las misiones simuladas, se realiza el diseño de una serie de experimentos con el objetivo de verificar la eficacia de las ayudas al operador implementadas en el interfaz de usuario del CCT adaptativo que se presenta en esta tesis. Estas pruebas han sido planteadas para someter a un grupo de individuos de forma sistemática a una serie de situaciones en instantes determinados, lo que nos permite evaluar su rendimiento y compararlo en situaciones similares, diferenciadas por aplicar o desactivar las ayudas al operador. Además también permite comparar los individuos entre ellos en instantes puntuales que puedan ser de interés, y comprobar si los resultados generales demuestran las ventajas de proporcionar ayudas al usuario, viendo cuántos individuos mejoran gracias a ellas. Los experimentos realizados han sometido al grupo de individuos a diferentes cargas de trabajo a lo largo de su ejecución por medio de tareas similares a las que ocurren durante las misiones reales con el objetivo de comprobar, mediante los mecanismos cuantificadores implementados en el sistema (carga de trabajo, tiempo empleado en realizar tareas primarias y tiempo de reacción en tareas secundarias), la evolución del rendimiento de los operadores. Adicionalmente, para concienciar al operador de la importancia de la realización de las tareas de la misión (ya que en una misión real, desatender una tarea podría ocasionar la pérdida o deterioro de algún vehículo), se ha implementado un sistema de puntuación que comienza en 100 puntos y cuyo valor desciende si el operador desatiende o tarda demasiado en realizar alguna tarea. Cuando el marcador llega a 0, la misión termina y se muestra al operador un aviso de que ha fracasado en el intento de completarla.

Un ejemplo del interfaz de usuario durante una simulación experimental con tres vehículos autónomos puede verse en la Figura 5.3. En ella se pueden observar los elementos gráficos de dos USVs (USV1 en azul y USV2 en rojo) y de un UAV (en negro). En el centro de la imagen se pueden ver los tres vehículos realizando una maniobra con puntos de paso y, en la parte superior en verde, el marcador del sistema de puntuación de tareas. Superpuesta sobre el interfaz, también podemos

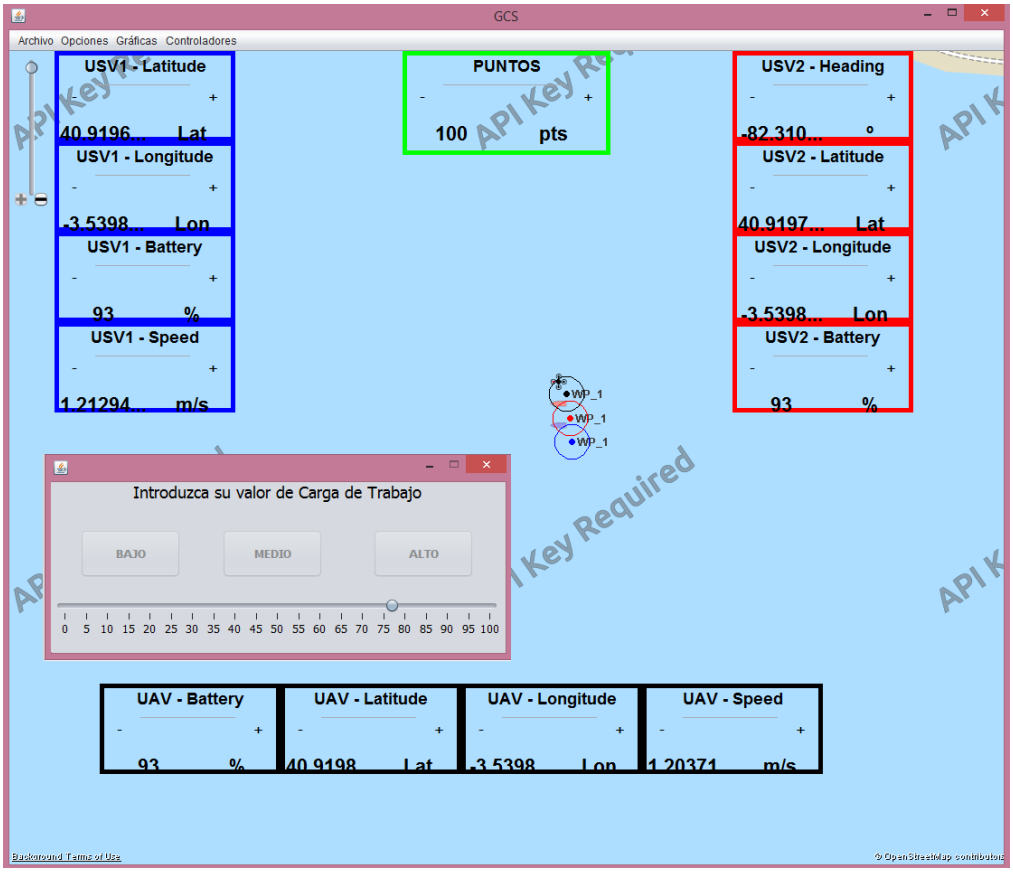


Figura 5.3: Interfaz de Usuario Durante una Misión Experimental

observar una ventana de consulta de carga mental de trabajo que el operador debe responder.

5.3.1. Población del Experimento

Para realizar los experimentos se han seleccionado dos conjuntos de usuarios: un grupo de referencia y un grupo principal. El grupo de referencia está formado por 4 individuos expertos cuyos resultados únicamente se han utilizado para calibrar algunos parámetros que han sido aplicados en el resto de experimentos. El grupo principal está formado por 24 individuos inexpertos, sobre los que se han medido los resultados de los experimentos definitivos. Más en concreto:

Grupo de Referencia: está formado por un grupo heterogéneo de 4 individuos, considerados usuarios avanzados, sobre los que se realizan las pruebas iniciales para verificar que el experimento medirá de forma efectiva la carga mental de trabajo. Además estas pruebas han servido para ajustar algunos parámetros de la simulación que han sido utilizados posteriormente con el grupo principal. Los individuos que forman este grupo tienen experiencia con centros de control o software de características similares y son instruidos previamente mediante un experimento de práctica para ayudar a paliar los efectos que pueda tener en los resultados la curva de aprendizaje. El análisis de las pruebas realizadas sobre este grupo nos ayuda a calibrar algunos parámetros con valores que serán aplicados en el resto de las experiencias de este capítulo: 1) la duración de la misión, que debe ser lo suficientemente larga como para probar ciclos de carga de trabajo a diferente intensidad, 2) la cantidad y dificultad de las tareas a resolver, ya que probar sobre un grupo experto nos orienta a la hora de saber la capacidad de trabajo que tiene un operador avanzado y de calibrar la simulación con menos tareas para un operador novato, 3) el tiempo estimado necesario para realizar las tareas, ya que deberemos dejar margen suficiente tras cada tarea para que esta pueda ser completada, y 4) los niveles de carga mental de trabajo subjetiva, tiempo empleado en una tarea y tiempo de reacción a partir de los cuales se debe modificar el umbral de transparencia para que haga efecto en los operadores.

Grupo Principal: está formado por un grupo heterogéneo de 24 individuos sobre los que se realizan los experimentos y se toman medidas para determinar si la carga mental de trabajo, tiempo empleado en tareas primarias y tiempo de reacción en tareas secundarias disminuyen al activar las ayudas al operador. Estos individuos se consideran principiantes, ya que no tienen experiencia con centros de control ni otros programas similares. Para la realización de las pruebas no se busca ningún perfil concreto de individuo, debido a que el objetivo de las ayudas es mejorar el rendimiento del operador, independientemente de sus aptitudes y sus características físi-

cas o psicológicas, ya que éstas pueden cambiar de un operador a otro sin influir de forma significativa en su rendimiento global. Es decir, un operador puede ser más diestro en un tipo de tareas concretas que en otro, o tener mayor capacidad de abstracción o de cálculo mental, compensando así algunos defectos con algunas virtudes. Por estas razones se realizan mediciones individuales en instantes determinados de la misión que se repiten en todos los experimentos para poder comparar el rendimiento de un individuo con y sin ayudas, el rendimiento de diferentes individuos en los mismos instantes y el rendimiento global del grupo en función de sus resultados individuales. Los parámetros de las misiones de este grupo han sido determinados a partir de los resultados obtenidos en los experimentos con el grupo de referencia, bajando el nivel de exigencia y aumentando los tiempos disponibles para la realización de las tareas. Con los experimentos de este grupo se ha de verificar si las ayudas al operador son eficaces en un porcentaje aceptable de la población del experimento y, por tanto, es razonable incorporarlas en las pruebas de campo realizadas en el marco del proyecto SALACOM.

5.3.2. Diseño de Misiones Experimentales

Para los experimentos con los dos grupos de individuos se han diseñado varias misiones que consisten en uno o varios vehículos autónomos realizando una maniobra mientras el operador debe ir realizando tareas de supervisión y control semejantes a las que se producen durante una misión real. Estas tareas o eventos son generados por el módulo vehículo autónomo del marco experimental y recibidas por el centro de control de tierra para que el operador pueda interactuar con el sistema y resolver la tarea asociada a cada evento. Las misiones son diseñadas en base a estos eventos, que se distribuyen a lo largo del tiempo de la misión simulada y que son mostrados al operador en diferentes ventanas para modificar su carga mental de trabajo. Dichos eventos se distribuyen en 4 grupos: 1) alarmas (que representan un aviso de valores anómalos que debe atenderse con urgencia), 2) comprobaciones rutinarias (que representan una comprobación

habitual sobre un vehículo u objetivo para verificar el desarrollo de la misión en ese instante), 3) colisiones (que son eventos urgentes que implican la detección de un obstáculo que debe ser esquivado), y 4) controles de carga de trabajo (en los que el operador debe introducir su carga mental de trabajo subjetiva durante la misión) y tiempo de reacción (que son eventos en los que el operador deberá realizar un pequeño cálculo mental para medir el tiempo que tarda en realizarlo). A continuación describiremos los eventos y tareas asociadas que pueden producirse durante la misión, y que pueden observarse en la Figura 5.4.

Alarma de Batería (AL_Bat). Las alarmas son avisos recibidos de los vehículos cuando alguno de sus datos de telemetría muestra un valor anómalo que podría ser peligroso para el desarrollo de la misión o la integridad del hardware del vehículo. Están marcados con una franja roja y cuando se reciben el operador debe darles prioridad para que puedan ser resueltos en el menor tiempo posible. En el caso de la alarma de batería, ésta simula un descenso repentino en el porcentaje de batería. Las tareas asociadas a este evento son una comprobación del valor de la batería del vehículo que ha mandado el aviso, apuntarlo en el registro junto con su identificador y, si el valor de batería es menor del 15 %, realizar una parada de emergencia del vehículo.

Alarma de Exceso de Velocidad (AL_Vel). Se trata de un aviso urgente y prioritario enviado al sistema desde el vehículo y que muestra un valor inusualmente alto de velocidad. Sus tareas asociadas serán similares a las de la alarma de batería: comprobar el valor del velocímetro de ese vehículo, anotarlo junto a su identificador en el registro y, en caso necesario, reducir la velocidad si el operador considera que está fuera del rango de seguridad.

Chequeo de Estabilidad (CK_Act). Estas comprobaciones son parte de la rutina de la misión y consisten en verificar que los datos de telemetría de los vehículos están dentro de un rango adecuado. El chequeo de estabilidad es una comprobación específica para UAVs en la que el operador deberá comprobar si la actitud del vehículo, es decir, el Cabeceo (Pitch) y Alabeo (Roll), tienen valores aceptables para un vuelo estable. El operador deberá

anotar en el registro si están por encima o por debajo de 2 grados. Ambas ventanas de chequeo pueden verse en la Figura 5.4a.

Chequeo de Timón (CK_Tim). La comprobación de los valores del timón es una verificación propia de los USVs en la que el operador debe confirmar si el valor actual del timón corresponde con el valor de referencia del timón calculado por el planificador de ruta para la maniobra que se esté llevando a cabo en ese instante. En este caso el operador debe comprobar si el valor entre el timón y su referencia difieren en más de 0.3 grados, lo cual podría implicar un desvío en la ruta producto de algún desajuste en el timón, en el algoritmo de control o por una perturbación externa (corriente marina, viento, etc.). Una vez comprobados los valores, se anotará en el registro si el valor es anómalo junto al identificador del USV que envió el aviso.

Chequeo de Ruta (CK_Ruta). La comprobación de ruta es una verificación que puede ser requerida por cualquier vehículo. Se trata de una medida rutinaria para que el operador compruebe durante algunos instantes concretos de la misión si los vehículos están cumpliendo con la planificación obtenida. El operador debe confirmar si el vehículo está pasando o va a pasar por el siguiente punto de ruta (waypoint) y, en caso contrario, comprobar si las coordenadas a las que le ha enviado el planificador son las correctas. La pantalla muestra los puntos de ruta de cada uno de los vehículos con el color en el que está representado el vehículo. Cada punto de ruta cuenta con un círculo interno que marca el lugar concreto en el que se encuentra el punto de ruta y con un círculo externo que indica el error máximo permitido para el paso del vehículo. Si el vehículo se encuentra fuera del círculo externo que marca el error máximo permitido, el operador debe indicárselo al planificador mediante un botón habilitado para ello. A continuación el planificador toma las medidas necesarias para corregir esa desviación del vehículo respecto a la ruta previamente planificada. Las ventanas de comprobación pueden verse en la Figura 5.4b.

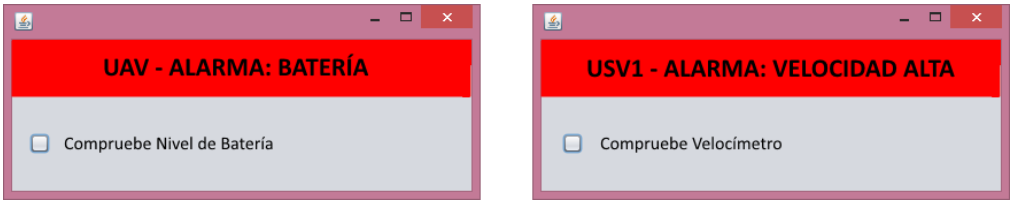
Peligro de Colisión (AL_COL). Es otro de los eventos urgentes, ya que implica que uno de los vehículos desplegados en la misión ha detectado un obstáculo y si sigue con la maniobra planificada chocará con él. Cuando se recibe este tipo de evento, el sistema despliega un panel de control simplificado especialmente diseñado para la maniobra de evasión. En este panel figura la distancia estimada al obstáculo, la velocidad actual del vehículo y la posición del timón. El operador debe comprobar los datos y seguir el algoritmo de esquiva establecido, cuyas decisiones dependen de los parámetros del vehículo en ese instante. El algoritmo indica las acciones a seguir en un orden determinado para evitar el choque (parada de emergencia si es imposible esquivar el obstáculo o reducción de velocidad y cambio de rumbo si esto fuese posible). Una vez alterados los parámetros en el panel de control, se ejecuta la maniobra y el sistema indica si ha sido exitosa o el operador se ha equivocado en alguno de los pasos del algoritmo. El panel de control simplificado para la resolución de una maniobra evasiva durante el evento de colisión, así como las instrucciones para realizarla pueden observarse en la Figura 5.4c

Tiempo de Reacción (TR). Además del tiempo empleado en las tareas primarias (alarmas, controles y peligro de colisión), mediante esta tarea, el sistema mide el tiempo de reacción del operador en tareas secundarias durante el transcurso del experimento. Estas tareas irán apareciendo en instantes concretos durante el desarrollo de la misión. La tarea secundaria para medir el tiempo de reacción, consta de un pequeño cálculo mental con cifras sencillas que nos sirve para determinar objetivamente si la carga mental de trabajo del operador es tolerable o excesiva. Que el tiempo de reacción aumente notablemente durante el transcurso de la misión o en un instante concreto de la misma, es un indicador de que la carga mental de trabajo del operador (y por tanto su fatiga y estrés) está alcanzando niveles que disminuyen su rendimiento.

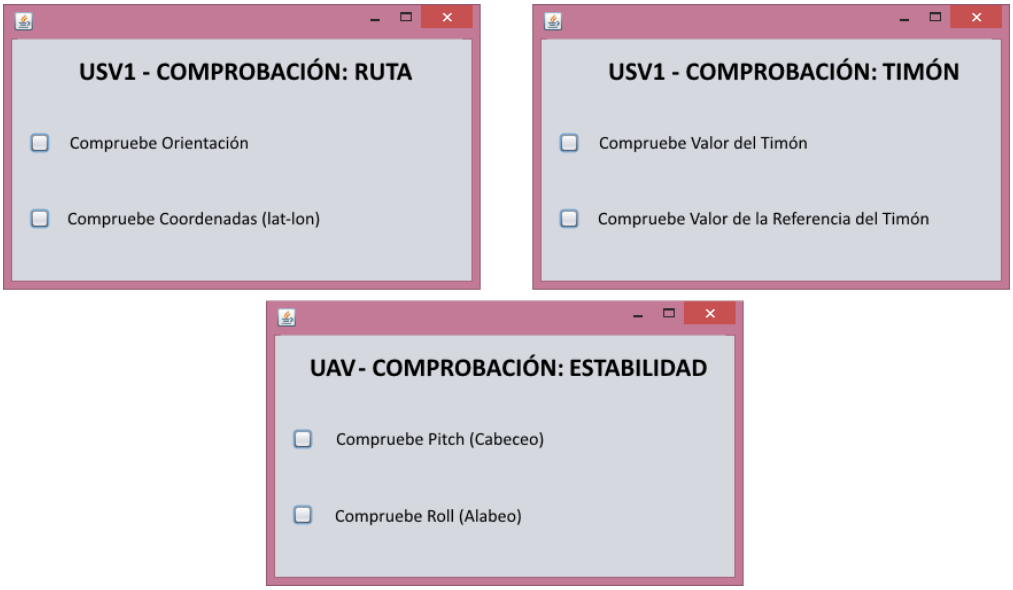
Carga Mental de Trabajo (CdT). Es la forma en la que el sistema mide durante la ejecución del experimento el valor subjetivo de carga mental de trabajo. Para ello, el operador contesta la consulta dependiendo del experimento, con un valor de 1 a 100 o pulsando un botón que permite seleccionar entre tres niveles (bajo, medio, alto). En el caso de este experimento, para una mayor precisión, se ha elegido la opción de medir el porcentaje de carga mental de trabajo. Esta información es esencial para comprobar la evolución de la capacidad de trabajo del operador a lo largo del tiempo de la misión y sirve para evaluar a posteriori qué partes de la misma resultan más complicadas y qué tareas son las más costosas en cuanto a carga mental de trabajo. Complementa al valor del test NASA-TLX que debe realizar el operador una vez finalizada la misión. Las consultas de CdT y TR pueden observarse en la Figura 5.4d.

Los eventos anteriormente descritos, generados por el marco experimental a través de los datos almacenados que se cargarán en las instancias de vehículo autónomo, son la base sobre la que se modelan las misiones experimentales. El operador realiza las tareas que el sistema le va indicando que tiene hacer a lo largo de la misión y cuya experiencia será completada por los datos de telemetría enviados también por los vehículos autónomos y que son mostrados, junto a la ubicación del vehículo en el mapa, por el CCT al operador.

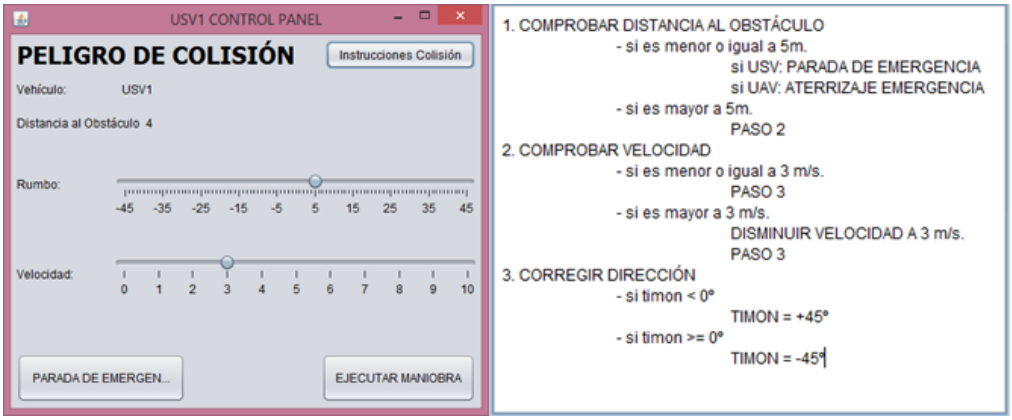
Para las pruebas iniciales con el grupo de referencia, se implementan las dos misiones que pueden verse en los cronogramas de la Figura 5.5. En los cronogramas podemos observar, a lo largo del tiempo que dura una misión experimental, el instante en el que sucede cada uno de los eventos programados. Cada evento, salvo en el caso de las consultas al operador, esta asociado a un vehículo, identificable en la Figura mediante un triángulo coloreado (rojo para USV1, azul para USV2, verde para UAV y negro para consultas) y está etiquetado con el código que aparece en la descripción de dichos eventos (por ejemplo, CK_Tim es la etiqueta que representa una comprobación rutinaria de timón y referencia de timón, y CdT representa una consulta de carga de trabajo). Estas misiones de prueba han sido diseñadas para evaluar el comportamiento del operador durante una



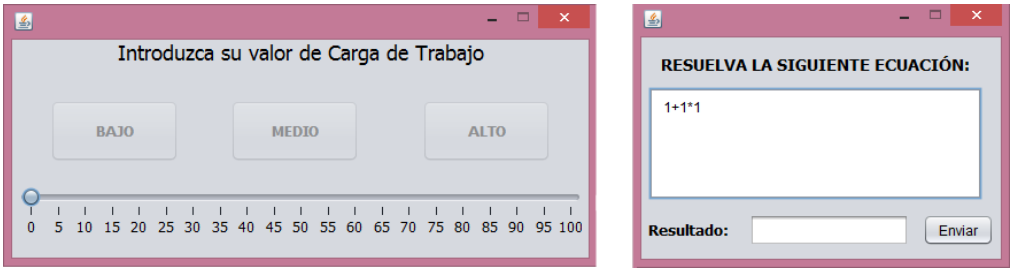
(a) Ventanas de Alarmas de Velocidad y de Batería



(b) Ventanas de Comprobaciones Rutinarias



(c) Ventanas de Colisión e Instrucciones de Esquiva



(d) Ventanas de Controles de Carga Mental de Trabajo y Tiempo de Reacción

Figura 5.4: Ventanas Asociadas a Eventos en una Misión Simulada

misión y poder calibrar los niveles de carga de trabajo (alto-medio-bajo), tiempo de reacción en tareas secundarias y tiempo empleado estimado para tareas primarias.

En primer lugar, en la Figura 5.5a, podemos ver el cronograma de la misión de prueba 1, cuya duración es de 450 segundos (siete minutos y medio) y que ha contado con 3 vehículos autónomos (USV1, USV2 y UAV). Esta misión ha sido diseñada con un control inicial de carga de trabajo y tiempo de reacción, que sirve para determinar el estado inicial del operador, tras lo cual se comienzan a suceder alternativamente eventos de los tres vehículos. Hay zonas en las que hay eventos más espaciados entre ellos y zonas en las que se pretende que la carga de trabajo sea más intensa, en la que se agrupan varios eventos con sus tareas correspondientes. A lo largo de la misión se van sucediendo las consultas de carga de trabajo (CdT) y tiempo de reacción (TR). Además, estos datos se vuelven a solicitar al operador al final de la misión para poder realizar una comparación entre su estado al principio y al final de la misión.

En segundo lugar, en la Figura 5.5b, podemos ver el cronograma de la misión de prueba 2, de una duración menor que la primera, 330 segundos (cinco minutos y medio) y que ha sido diseñada para provocar un mayor estrés en el operador, maximizando los eventos urgentes que requieren actuaciones rápidas ante sus tareas asociadas (alarmas y avisos de colisión). Desde el inicio de la misión se comienza con tareas muy exigentes para que la carga mental de trabajo del operador sea alta en la totalidad de la misión, incluso antes de las comprobaciones iniciales de CdT y TR para probar una aproximación distinta a la de la misión de prueba 1 y comprobar cuál se ajusta más a las necesidades de nuestros experimentos.

Finalmente, tras las pruebas realizadas al grupo de referencia, se ha llegado a la conclusión de que es conveniente realizar un chequeo de CdT y TR al principio y al final de la misión para tener una referencia del estado del operador cuando aún no ha comenzado la actividad mental y otra cuando finaliza la misión, de forma que pueda verse una evolución del nivel de carga mental de trabajo además del posible aumento o descenso puntual en algunos instantes del experimento.

También hemos podido ajustar la cantidad de eventos y tareas asociadas que pueden resultar adecuados a un grupo de operadores inexpertos, que será en todo caso menor al que se aplicó en las misiones de prueba en el grupo de referencia de operadores avanzados.

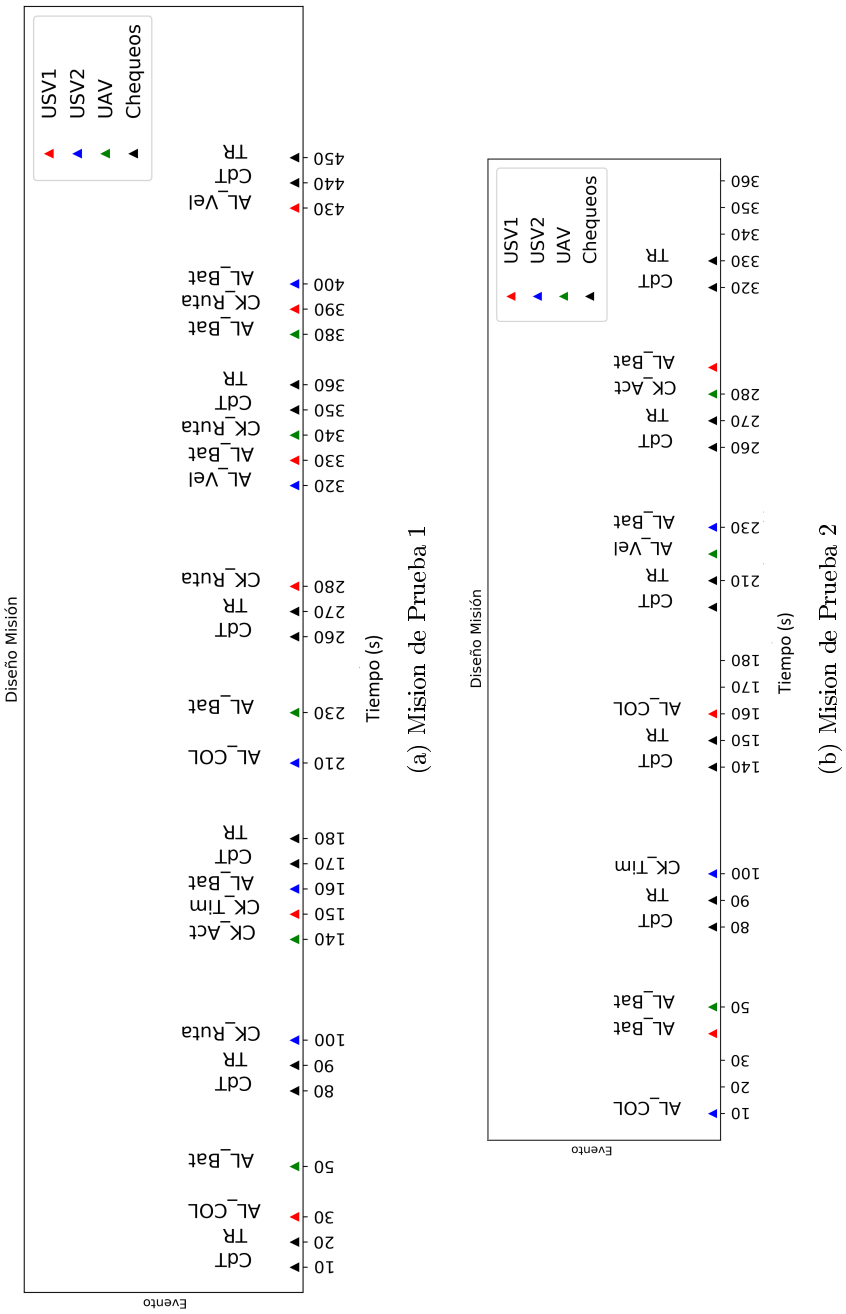
Una vez realizados los experimentos con el grupo de referencia, formado por operadores expertos, se realiza el diseño de las misiones definitivas para el grupo principal, operadores inexpertos, calibrando algunos aspectos de las mismas en un nivel menor de dificultad para obtener niveles altos de carga mental de trabajo de los individuos pero sin sobrepasar su límite. Los valores más importantes que se han calibrado son: los parámetros que activan las ayudas al operador, los tiempos estimados de realización de cada tarea, la dificultad de la misión (cantidad de tareas realizadas simultáneamente) y su duración para que los individuos no superen los límites de fatiga que ocasionan el abandono de tareas. A continuación, describiremos las dos misiones diseñadas a las que ha sido sometido el grupo principal y que pueden verse en la Figura 5.6.

En la Figura 5.6a, se encuentra el cronograma de la Misión 1, diseñada como misión de entrenamiento. Esta misión tiene una duración menor a tres minutos (170 segundos) y, aunque se han incluido tres vehículos en la misión, los eventos que se generan están muy espaciados entre sí a propósito para ayudar al operador a familiarizarse con el centro de control y aprender a realizar cada una de las tareas (se ha incluido una tarea de cada variedad disponible) que podrían surgir durante un experimento. Se da a cada operador la opción de repetir la misión cuantas veces desee para que aprenda los mecanismos que implica cada tarea y de este modo se suaviza la curva de aprendizaje hasta casi eliminar sus efectos en las dos iteraciones (sin ayudas y con ayudas) de la misión definitiva.

En la Figura 5.6b se encuentra el cronograma de la Misión 2, que es la misión definitiva a la que se ha sometido al grupo principal para cuantificar su rendimiento con ayudas y sin ayudas al operador. Su diseño es similar al de la misión de prueba 1, con chequeos iniciales y finales de CdT y TR, además de los intermedios en instantes puntuales de la misión, pero con un nivel de carga de trabajo ajustado a los operadores inexpertos. Esta misión está dividida en tres etapas

diseñadas para someter a los individuos del experimento a estímulos concretos que provoquen diferentes cargas de trabajo y así cuantificar su comportamiento en distintas situaciones. Comienza con una primera fase de calentamiento que dura hasta los 50 segundos, en la que se realiza el chequeo inicial de CdT y TR para establecer sus niveles base iniciales y se comienzan a realizar algunas acciones con márgenes de tiempo amplios para que el operador vaya entrando en situación (el operador está aún fresco y los eventos aparecen lo suficientemente espaciados como para ser atendidos adecuadamente). En la zona intermedia, hasta los 400 segundos, la carga de trabajo se aumenta hasta un nivel suficiente como para poder verificar que las ayudas son efectivas. Finalmente ocurre la fase de enfriamiento, desde el segundo 400 en adelante, en la que los eventos vuelven a estar separados como en la fase inicial y en la que se llevan a cabo las últimas consultas de CdT y TR.

Los individuos del grupo principal realizan dos iteraciones de esta última misión: la primera, sin ningún tipo de ayuda, y la segunda, con los mecanismos de ayuda al operador de adaptabilidad y transparencia activados. Durante la ejecución de ambas misiones, los resultados obtenidos de cada individuo se almacenan individualmente junto al identificador del operador y los datos de configuración de la misión. Más en concreto, se almacena el identificador de usuario, la duración de la misión, los vehículos involucrados, las ayudas activadas, la semilla aleatoria, el identificador y tipo de cada evento junto a su vehículo asociado, el tiempo en el que se produce, el tiempo estimado para su realización y el tiempo que el operador ha empleado en resolverla, además de los tiempos de reacción en tareas secundarias y el nivel de carga de trabajo subjetivo consultados durante el desarrollo del experimento. Adicionalmente también se guardan datos relativos al comportamiento del interfaz gráfico para que puedan ser analizados junto a los datos extraídos de la misión y el operador. En particular, se almacena el umbral de transparencia en cada instante, el número de elementos gráficos mostrados en cada instante, la posición de cada elemento gráfico perteneciente a cada vehículo, etc. Finalmente, tras cada ejecución, el operador realiza el test NASA-TLX, cuyos resultados son almacenados, procesados y analizados.



(a) Mision de Prueba 1

(b) Mision de Prueba 2

Figura 5.5: Cronograma de las Misiones Simuladas de Prueba para el Grupo de Referencia

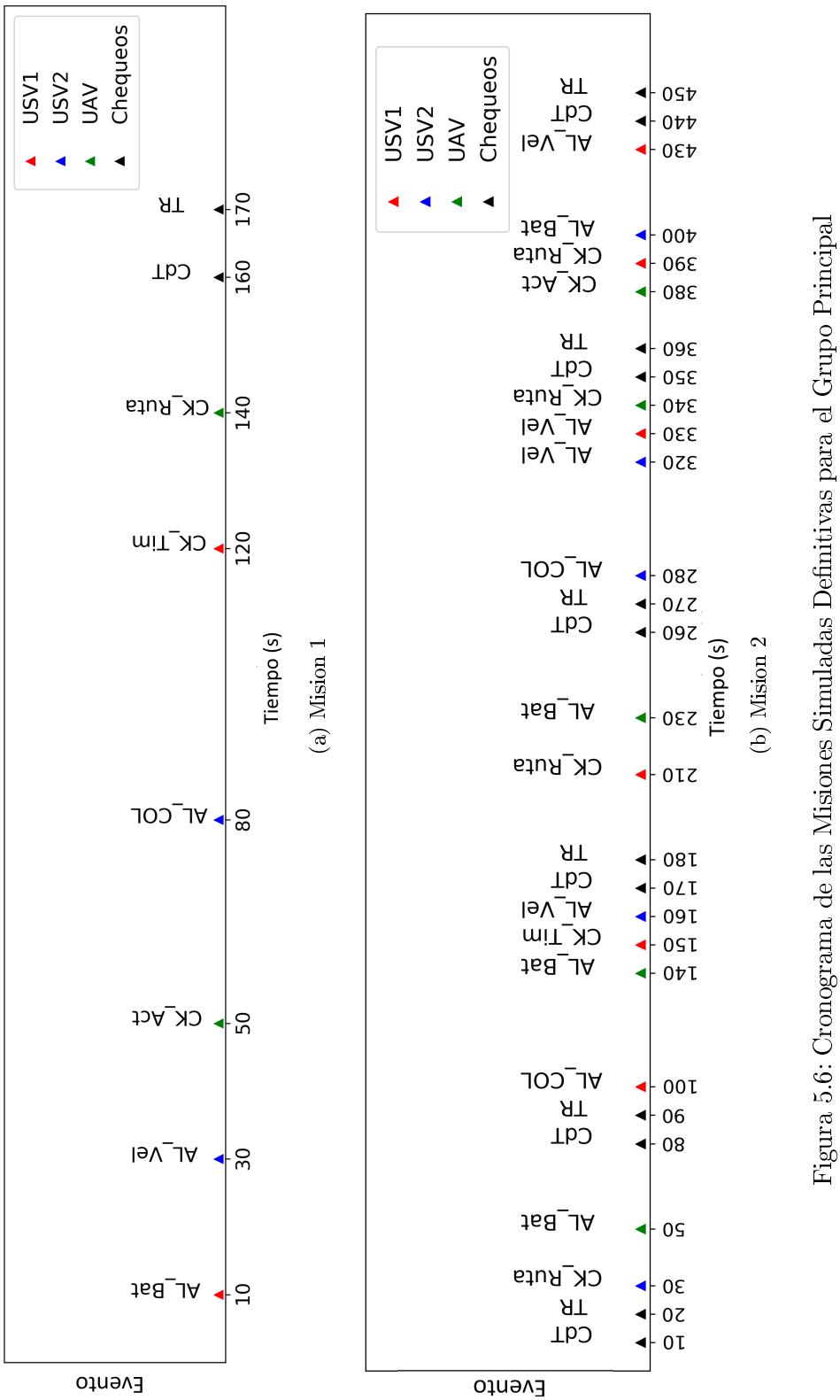


Figura 5.6: Cronograma de las Misiones Simuladas Definitivas para el Grupo Principal

5.4. Resultados

Los experimentos descritos en la sección 5.3.2 se han llevado a cabo sobre un conjunto de 24 individuos (grupo principal). En primer lugar cada individuo completa una fase de entrenamiento con la misión 1 (ver cronograma en Figura 5.6a) hasta que considera que ya tiene cierta soltura con el centro de control y las tareas que debe realizar (debido a la diferencia de aprendizaje que puede haber entre distintos operadores). A continuación, cada individuo del grupo principal lleva a cabo la misión 2 (ver cronograma en Figura 5.6b) en dos ocasiones, con ayudas y sin ayudas al operador. Cada misión se realiza de forma individual y sin comunicación entre operadores para evitar posibles interferencias o distracciones. Para cada operador se recogen durante la ejecución de la misión los cuantificadores de carga de trabajo (CdT), tiempo empleado en tareas primarias (TE) y tiempo de reacción (TR) en tareas secundarias. Al finalizar cada misión se recoge el valor de carga de trabajo a posteriori mediante la realización del test NASA-TLX. Estos resultados son almacenados y analizados de forma individual, ya que debido a las diferentes características de cada operador, los resultados pueden oscilar entre valores muy dispares. Esto hace que resulte más útil para nuestro propósito la comparación de resultados del mismo operador entre diferentes ejecuciones del experimento. No obstante, también se presentarán los resultados agrupados para poder realizar la comparación de ciertos valores entre individuos y el análisis de los resultados generales.

A continuación detallaremos, de forma gráfica para una mejor visualización, los resultados obtenidos por cada operador y compararemos los valores obtenidos para cada uno de los cuantificadores anteriormente mencionados tanto en la misión sin ayudas como en la misión con ayudas al operador.

En el primer bloque de gráficas (Figuras 5.7 a 5.12), podemos observar dos gráficas por cada operador. Encontramos en cada uno de estos cuadros un gráfico de barras que muestra los resultados de tiempo estimado para una tarea en azul contra el tiempo empleado por el operador en dicha tarea en naranja (si la barra naranja se superpone a la azul, el final de la barra azul está marcado con una

línea negra sobre la barra naranja). En la gráfica de la izquierda encontramos los tiempos obtenidos durante la primera ejecución del experimento, sin ayudas. A la derecha, podemos observar la misma gráfica, que esta vez representa los tiempos estimado y empleado para cada tarea en la ejecución del experimento con ayudas al operador. Además, para una mejor visualización, las gráficas están escaladas por pares (misma escala para cada operador), de forma que se pueda apreciar mejor la diferencia entre ambas ejecuciones. Podremos observar pequeñas diferencias entre la ejecución de un operador y otro, debidas en la mayor parte de las ocasiones al abandono de alguna de las tareas o su ejecución fuera del tiempo asignado para ella (en cuyo caso ésta no estará representada mediante las dos barras en el lugar que le correspondería de haber sido realizada correctamente). En algunas ocasiones puntuales, la carga de trabajo ha sido tan alta que al operador no le ha sido posible finalizar la misión, en cuyo caso faltan todas las tareas a partir de un instante determinado. Por ejemplo, en las dos gráficas del Operador 4 (en la Figura 5.7), puede observarse que tanto en la gráfica sin ayudas al operador como en la que sí se incluyeron ayudas, los eventos aparecen mucho más espaciados (no se realizaron todas las tareas, o al menos no se realizaron en el instante adecuado). Además en la gráfica de la derecha se puede observar que no aparece ninguna barra a partir del instante 250, lo cual indica que el operador desatendió las tareas, por lo que el sistema de puntuación llegó a 0 finalizando el experimento sin terminar la misión.

En general, en la primera ejecución del experimento, en la que el operador no cuenta con ayudas, el tamaño de las barras naranjas es, en muchas ocasiones, mayor que en el caso con ayudas, lo que quiere decir que el tiempo empleado suele superar al tiempo estimado inicialmente a pesar de la estimación realizada durante los experimentos con el grupo de referencia (incluido el aumento de tiempo agregado para los usuarios inexpertos). Sin embargo, si observamos las gráficas de la segunda ejecución, a la derecha, en la que el operador cuenta con los mecanismos de adaptabilidad y transparencia que le ayudan a localizar rápidamente los elementos importantes (colocándolos en puestos de mayor prioridad y mostrando sólo aquellos relevantes en cada instante), las barras naranjas decre-

cen, mostrando mejoría. Es reseñable que en el caso de algunas tareas puntuales, en las que el operador ha empleado un tiempo muy por encima del estimado para esa tarea, la mejora ha sido muy significativa, lo que quiere decir que este tipo de ayuda no solo mejora el rendimiento general del operador, sino que en algunas ocasiones, produce una mejora considerable en aquellas tareas que resultan más críticas para un operador concreto (donde se muestra una mayor mejora es en aquellas tareas que más le costó realizar). Por ejemplo, podemos observar las gráficas relativas al individuo etiquetado como Operador 2 (en la Figura 5.7) en las que, en general, se observa mayor cantidad de barras azules en la gráfica de la derecha (con ayudas) y, más concretamente, en las tareas que más trabajo le costaron sin ayudas (barras naranjas más altas en torno a los instantes 150 y 350) la mejora producida es muy significativa, ya que no solo ha bajado el tiempo empleado para la tarea realizada, sino que además ha pasado de superar por mucho el tiempo estimado para dicha tarea a realizarla en un tiempo mucho menor al estimado.

Durante los experimentos se ha observado que tanto el tiempo de respuesta como la percepción que cada individuo tiene sobre su carga de trabajo puede variar mucho respecto a la de otros individuos. Por lo tanto, para mostrar los resultados globales de los individuos no se realizará la media de los valores individuales, sino la media de la respuesta relativa de cada individuo. Para ello se calcula la diferencia entre el Tiempo empleado sin ayuda y con ayuda (TEsin - TEcon), que se encuentra en la Figura 5.13. En líneas generales se puede observar que los valores permanecen por encima de cero, lo que significa que se ha producido una mejora en los tiempos empleados, ya que los valores positivos indican que se ha reducido el tiempo empleado cuando las ayudas están activas. Además, los valores elevados indican que la reducción del TE es significativa.

En el siguiente bloque de imágenes (Figuras 5.15 a 5.17), podemos encontrar las gráficas de carga de trabajo (azul), tiempo de reacción (rojo) y umbral de transparencia (negro) a lo largo de la ejecución de la misión. Las líneas punteadas pertenecen a la ejecución en la que el operador no cuenta con ayudas, mientras que las líneas continuas pertenecen a la ejecución con ayudas al ope-

rador. Además, los círculos coloreados indican el instante de la misión en la que se ha realizado la consulta de CdT o TR. Podemos encontrar la leyenda de estas gráficas en la Figura 5.14, que aparece justo antes del bloque de imágenes correspondiente. Al igual que en las gráficas anteriores, pueden encontrarse algunas diferencias puntuales debidas a la alteración del orden que ha podido sufrir alguna de las consultas debido a que el operador no la haya contestado en el instante adecuado o a que la misión no ha podido ser finalizada. Por ejemplo, volviendo a tomar como referencia la gráfica del Operador 4 (en la Figura 5.15), podemos observar que al igual que en las gráficas anteriores, la ejecución de la misión con ayudas finalizaba abruptamente alrededor del instante 250, en este caso las líneas continuas (valores pertenecientes a la ejecución del experimento con ayudas), también paran alrededor de ese instante o un poco antes (en el momento en el que se realizó la última consulta antes de la finalización de la misión).

En este bloque de gráficas, puede observarse que, en general, las líneas punteadas permanecen por debajo de las líneas lisas, lo que indica que las ayudas al operador han permitido reducir los valores de carga de trabajo y los tiempos de respuesta durante gran parte de la ejecución de los experimentos. También podemos observar que, en algunos casos, al descender el umbral de transparencia, los niveles de carga de trabajo y los tiempos de reacción descienden inmediatamente. Esto significa que los mecanismos de transparencia han ocultado la información no relevante para la tarea que está siendo realizada en ese instante, permitiendo al operador mantener el foco en las labores importantes, reduciendo sus tiempos de respuesta y ayudando a disminuir su sensación subjetiva de carga de trabajo mental. Por ejemplo, si fijamos la atención en la gráfica del Operador 2 (en la Figura 5.15), podemos observar que según avanza la misión en la que las ayudas están activas (líneas contínuas), el operador percibe una carga mental de trabajo moderada, por lo que el sistema va aumentando el umbral de transparencia hasta que alcanza el nivel en el que se sitúa en la misión sin ayudas, momento en el que se produce un pico máximo en el valor de carga mental de trabajo. En ese instante, el sistema lo detecta y vuelve a bajar el umbral de transparencia

produciendo un descenso casi instantáneo en la carga mental de trabajo y en el tiempo de reacción (líneas roja y azul contínuas).

También podemos concluir, por lo observado en los valores de umbral de transparencia, que éste es un parámetro muy sensible a las características de cada operador, por lo que en este primer conjunto de experimentos no ha entrado en acción todas las veces que hubiese sido deseable. Por otro lado, la adaptabilidad ha resultado efectiva en la mayoría de las ocasiones como ayuda para disminuir la carga mental de trabajo subjetiva, el tiempo de reacción en tareas secundarias y el tiempo empleado en tareas primarias, por lo que se puede deducir que se trata de un método más estándar que puede ayudar a una mayor cantidad de operadores heterogéneos y que además necesita de un proceso de calibración menor.

A continuación, en la Figura 5.1, podemos encontrar una tabla con los resultados del test NASA-TLX realizados tras cada una de las ejecuciones de la misión por cada uno de los operadores. Para poder evaluar los resultados obtenidos, debemos saber que en este test se consideran tres niveles de carga de trabajo mental: 1) bajo, si el valor obtenido se encuentra por debajo de 500, 2) medio, si el valor obtenido se encuentra entre 500 y 1000, y 3) alto, si el valor se encuentra por encima de 1000. En la mayoría de las ocasiones se ha obtenido un valor subjetivo de carga de trabajo medio-alto, puesto que el experimento ha sido diseñado para provocar en el operador una carga mental de trabajo elevada. Además se puede observar que en el 76 % de las ocasiones (19/24), el operador tuvo una sensación de carga mental de trabajo menor en la misión realizada con ayudas al operador, mientras que en el 24 % restante (5/24) las ayudas no resultaron efectivas.

Por último, en la Figura 5.18 se encuentra la representación de la diferencia entre los índices NASA-TLX de cada operador sin ayuda y con ayuda (TLXsin-TLXcon). Si centramos nuestra atención en esta gráfica, podremos observar que en casi todos los casos (menos en los 5 anteriormente mencionados) los valores son positivos, lo que indica una mejora en los resultados del test pertenecientes a la ejecución del experimento en el que se incluyeron ayudas al operador. Además,

en algunos de los casos, el valor es alto, lo que indica que el grado de mejora es bastante significativo.

Tras la evaluación de los resultados obtenidos durante este conjunto de experimentos, podemos determinar que las ayudas al operador son eficaces en la mayoría de los casos (el 76 %) y que los parámetros que ajustan los mecanismos de adaptabilidad son menos sensibles a las características de cada operador que los parámetros que se han de ajustar para las mecánicas de transparencia. No obstante, en aquellos casos en que los mecanismos de transparencia han sido activados, se ha obtenido una notable mejoría a los pocos segundos. Por estas razones, consideramos razonable que los mecanismos de ayuda estudiados (adaptabilidad y transparencia) sean incluidos como parte del interfaz gráfico en los centros de control para misiones que involucren equipos cooperativos de vehículos autónomos heterogéneos, ya que gracias a los resultados expuestos en esta tesis, hemos podido verificar que provocan una disminución en la carga mental de trabajo del operador, lo que influye positivamente en el desarrollo de la misión aumentando el rendimiento del operador y permitiéndole realizar un mayor número de tareas durante más tiempo antes de que la fatiga comience a influir negativamente en el desempeño de su labor. Además, realizando experimentos sobre un mayor número de individuos, podremos lograr una mejor calibración de los parámetros que activan los mecanismos de adaptabilidad y, sobre todo, los de transparencia (más sensibles a las características individuales del operador), lo que podría mejorar aún más los resultados obtenidos y conseguir un aumento en el porcentaje de individuos para los que estas ayudas son efectivas.

5.5. Conclusiones

Tras el estudio de los mecanismos de ayuda al operador (adaptabilidad y transparencia) para su integración en el interfaz gráfico del CCT adaptativo, se ha diseñado un marco experimental que, con la participación de un grupo de 24 individuos, nos ha permitido verificar su eficacia a la hora de ser aplicado en misiones complejas que implican un equipo de vehículos heterogéneo (dos USVs

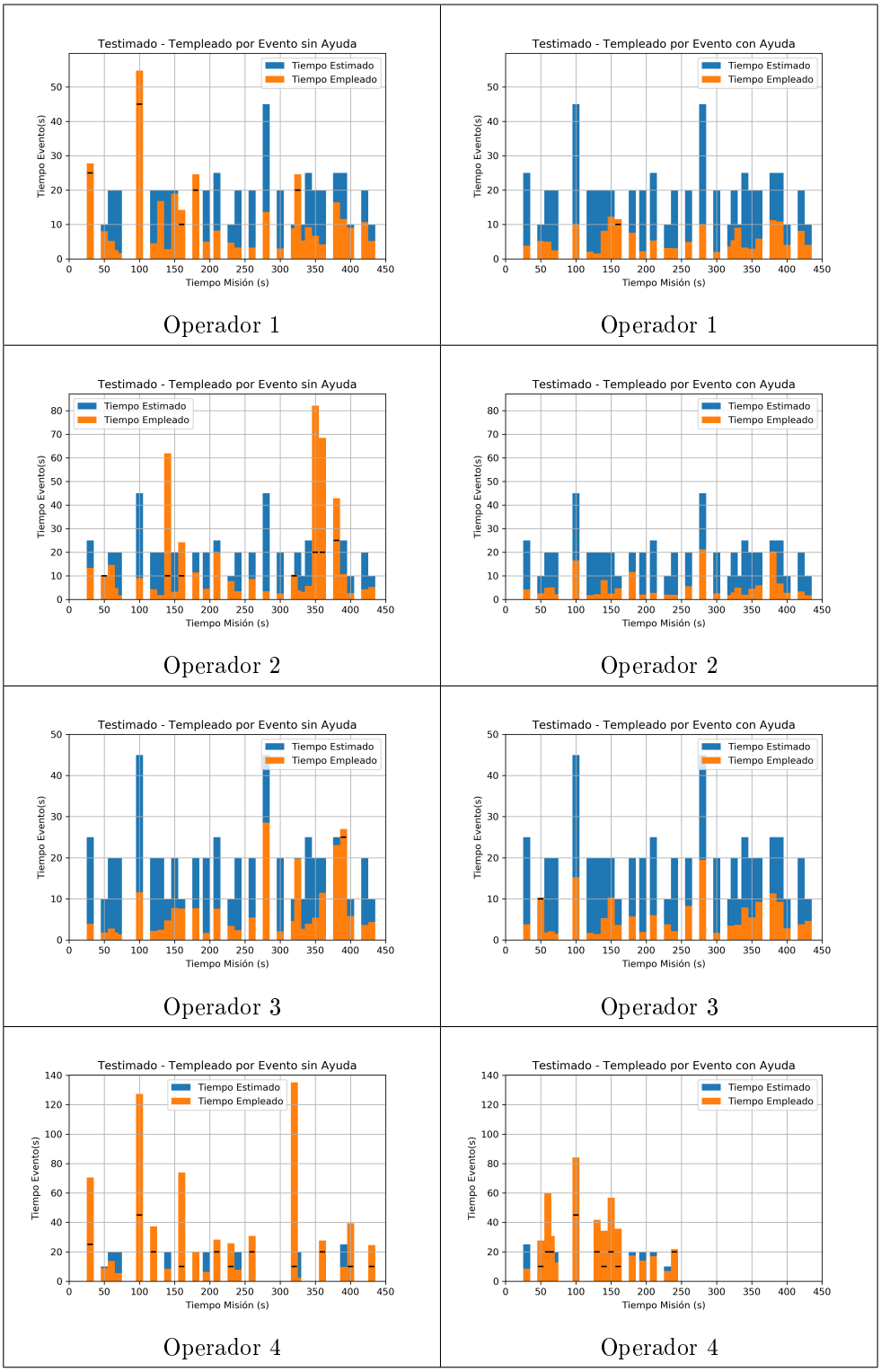


Figura 5.7: Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 1-4)

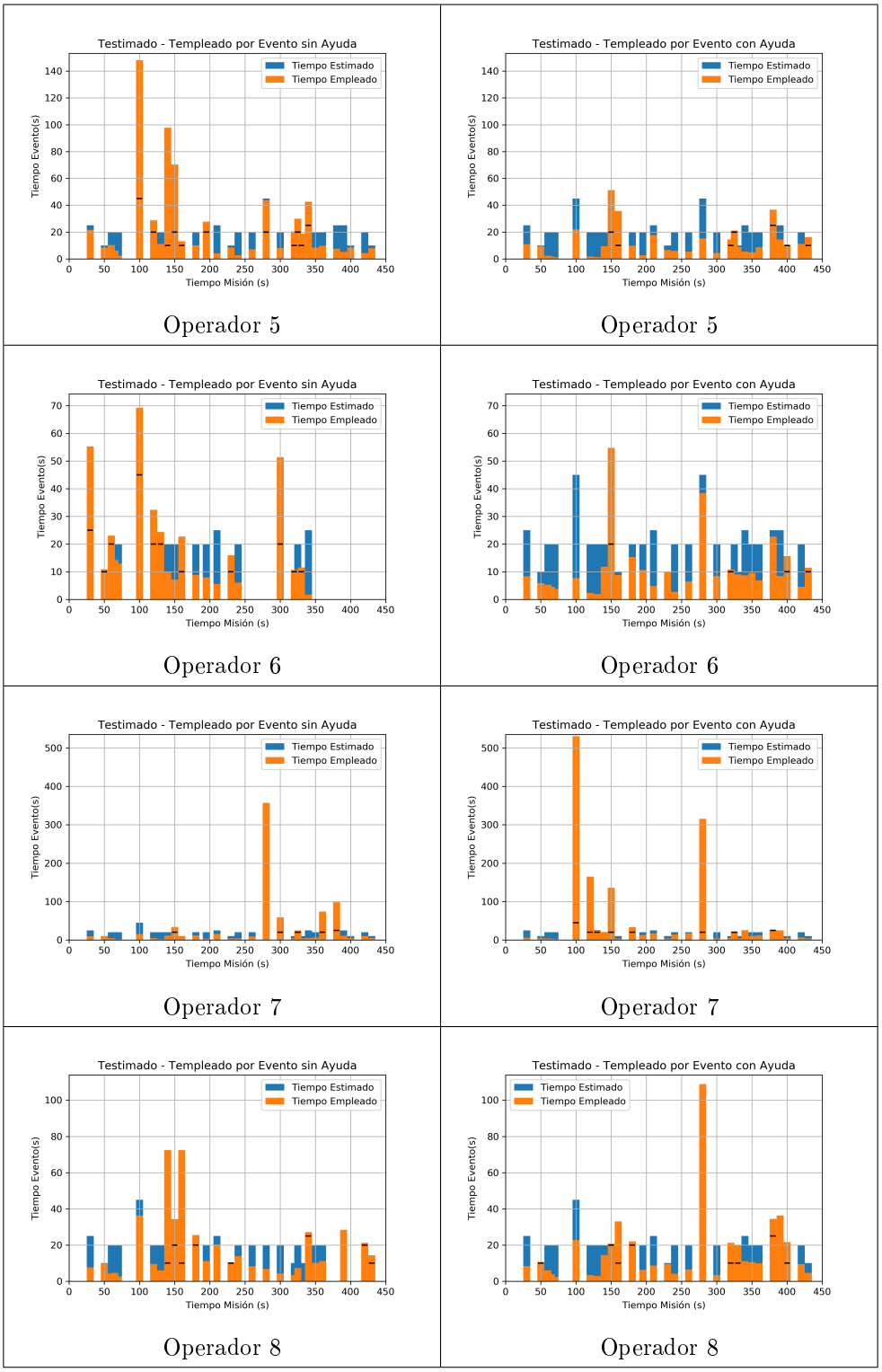


Figura 5.8: Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 5-8)

Figura 5.9: Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 9-12)

Figura 5.10: Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 13-16)

Figura 5.11: Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 17-20)

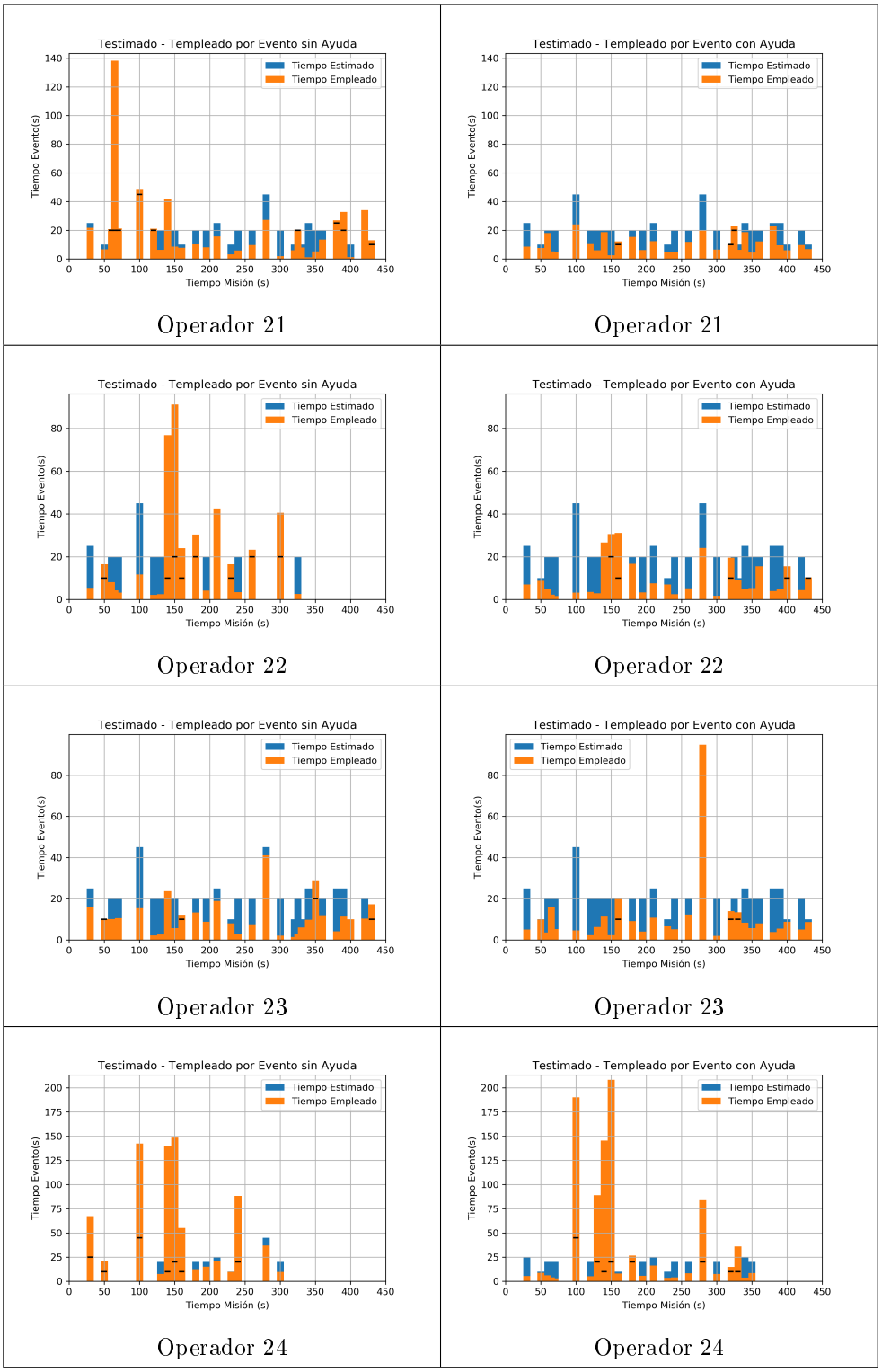


Figura 5.12: Gráficas de Tiempo Estimado y Tiempo Empleado (Operadores 21-24)

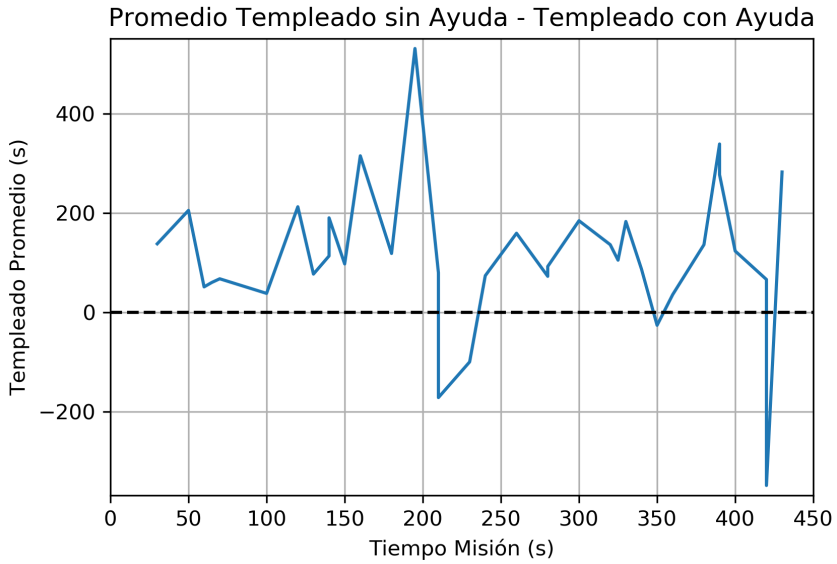


Figura 5.13: Gráfica de Tiempo Empleado Promedio sin Ayudas - Tiempo Empleado Promedio con Ayudas

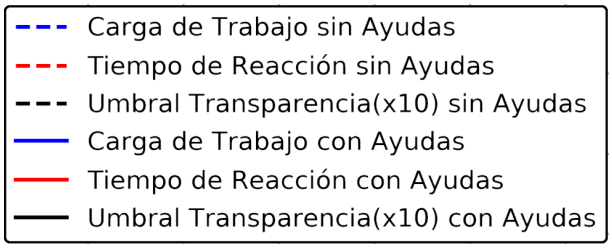


Figura 5.14: Leyenda para Gráficas Carga de Trabajo (CdT), Tiempo de Reacción (TR) y Umbral de Transparencia (UT)

y un UAV). Después de implementar el marco experimental, se ha diseñado una misión de entrenamiento para que los operadores inexpertos del grupo principal se familiaricen con el centro de control y las tareas que deben realizar durante los experimentos. Tras practicar con esta misión, se ha realizado la ejecución de otra misión diseñada para poner a prueba la utilidad de las técnicas de ayuda al operador implementadas en el CCT adaptativo. Se han realizado dos ejecuciones del experimento, la primera sin ayudas y la segunda con ayudas al operador en las que se han medido: 1) la carga de trabajo mental subjetiva durante la misión,

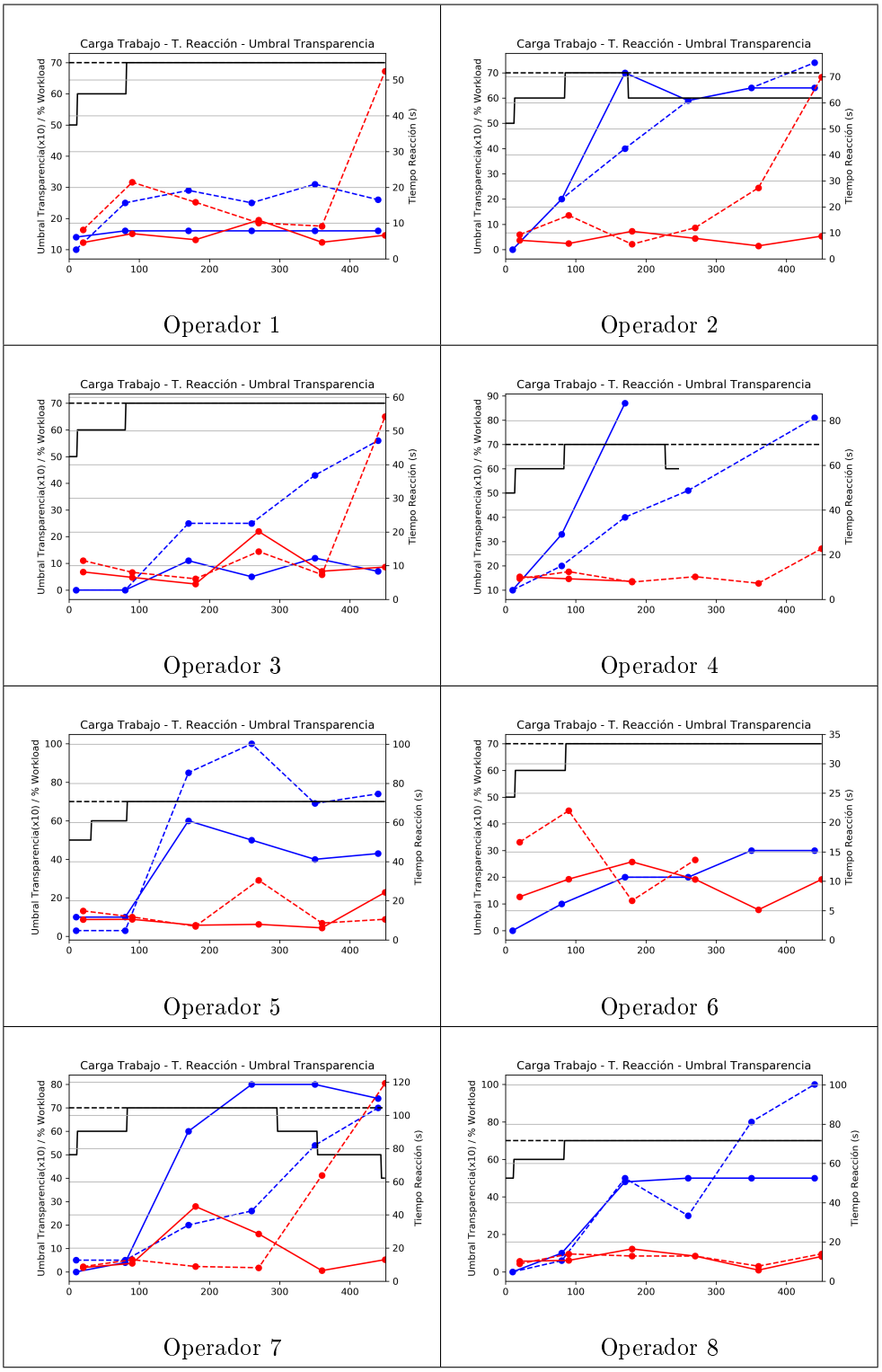


Figura 5.15: Gráficas de Carga de Trabajo, Tiempo de Reacción y Umbral de Transparencia (Operadores 1-8)

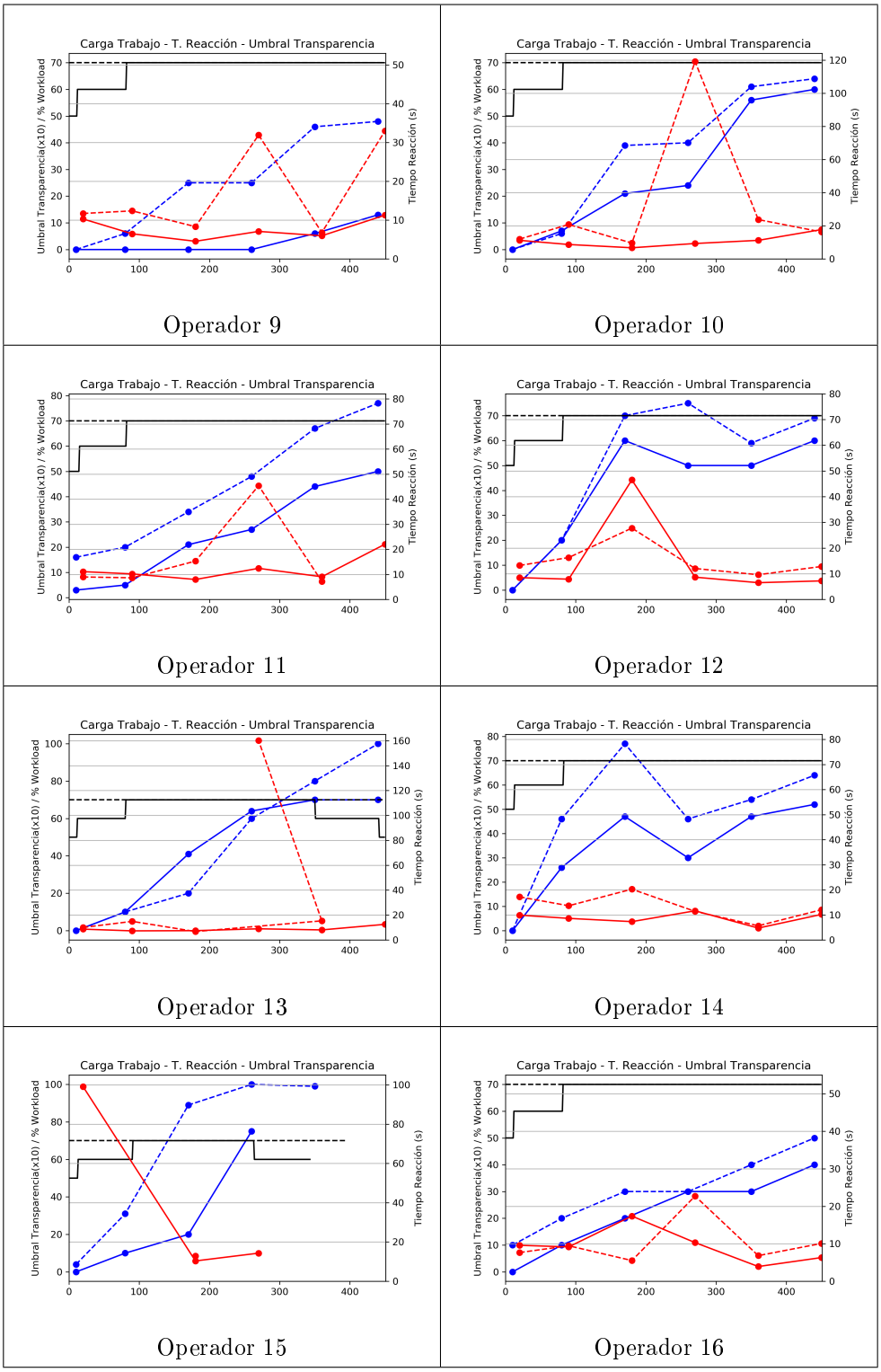


Figura 5.16: Gráficas de Carga de Trabajo, Tiempo de Reacción y Umbral de Transparencia (Operadores 9-16)

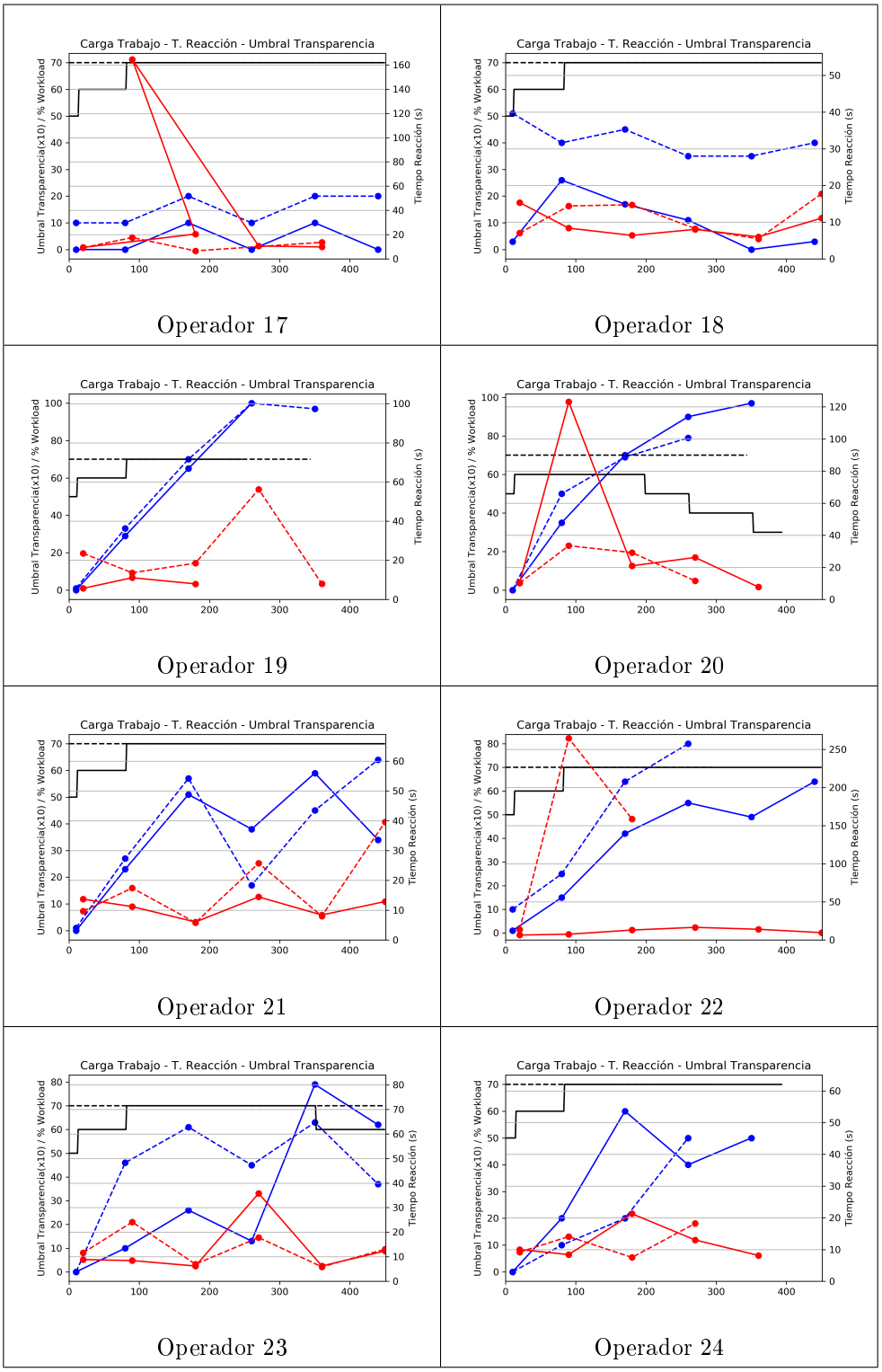


Figura 5.17: Gráficas de Carga de Trabajo, Tiempo de Reacción y Umbral de Transparencia (Operadores 17-24)

OPERADOR	# NASA-TLX SIN AYUDAS	# NASA-TLX CON AYUDAS
1	850	800
2	710	900
3	1145	960
4	1260	1040
5	1265	935
6	875	670
7	685	780
8	1175	1050
9	910	450
10	730	510
11	1310	800
12	820	445
13	910	430
14	995	775
15	820	775
16	850	1015
17	935	635
18	885	570
19	1415	1230
20	1205	1075
21	990	975
22	425	875
23	1130	1040
24	475	925

Tabla 5.1: Tabla de Resultados Obtenidos con el Test NASA-TLX en los Experimentos de Ayudas al Operador

2) el tiempo empleado en cada tarea primaria, 3) el tiempo de reacción en cada

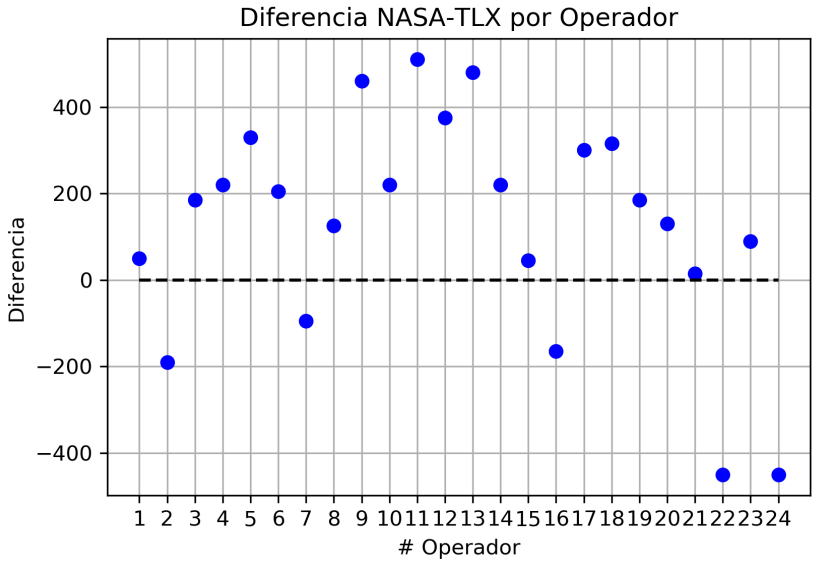


Figura 5.18: Gráfica de Resultados Obtenidos con el Test NASA-TLX en los Experimentos de Ayudas al Operador

tarea secundaria, y 4) la carga de trabajo mental subjetiva a posteriori mediante el test NASA-TLX.

De los resultados obtenidos en los experimentos se extrae la conclusión de que, en general, el tiempo empleado en las tareas primarias, el tiempo de reacción en tareas secundarias, y la carga de trabajo mental subjetiva (medidas tanto durante la misión como a posteriori) disminuyen en aquellas misiones en las que se han activado las ayudas al operador, llegando a producirse en algunos casos una mejora significativa. Por estas razones, se considera que las ayudas al operador estudiadas en esta tesis (es decir los mecanismos de adaptabilidad y transparencia) son eficaces, ya que disminuyen la carga de trabajo y la fatiga de los operadores de centros de control que llevan a cabo las tareas asociadas a misiones complejas que involucran a equipos cooperativos de vehículos heterogéneos. Además, el aumento en el rendimiento de los operadores producido gracias a las ayudas integradas en el CCT adaptativo presentado en esta tesis, les permite realizar un mayor número de tareas, aumentando así el número de

vehículos autónomos simultáneos que pueden supervisar, pudiendo reducirse en algunos casos el número de operadores necesario para llevar a cabo determinadas misiones complejas que impliquen equipos de vehículos autónomos heterogéneos.

Capítulo 6

Conclusiones y Trabajo Futuro

"Nunca sabes lo que es suficiente a menos que sepas lo que es más que suficiente".

William Blake

En este capítulo se realizará un resumen de las principales contribuciones del trabajo presentado en esta tesis, así como de las líneas de trabajo más interesantes que pueden desarrollarse en el futuro.

6.1. Conclusiones Generales

Esta tesis acomete el problema del desarrollo de un centro de control de tierra capaz de dar soporte a múltiples tipos de misiones que, debido a cambios en sus objetivos o en el entorno en el que se llevan a cabo, puedan requerir la participación de diferentes tipos de equipos colaborativos de vehículos heterogéneos. Además, debido a las dificultades que conlleva la supervisión de misiones en las que participan simultáneamente múltiples vehículos, la carga de trabajo del operador puede ser muy alta, por lo que el centro de control tiene que ser capaz de mostrar la información de forma ergonómica e incorporar ayudas al operador para paliar los efectos adversos producidos por la sobrecarga del operador.

La implementación de un CCT que permita la supervisión de múltiples vehículos heterogéneos es una labor compleja, ya que cada vehículo puede contar con un protocolo de comunicaciones, elementos gráficos de visualización y controles distintos, introduciendo una gran cantidad de requisitos que el software debe ser capaz de cumplir. Esto suele derivar en la implementación de centros de control diseñados ad-hoc para un tipo de vehículos, objetivos y entorno concretos cuyo cambio requeriría, en la mayoría de ocasiones, un costoso proceso de rediseño e implementación. Además, la supervisión de múltiples vehículos heterogéneos implica la monitorización de una gran cantidad de información (que aumenta con cada vehículo involucrado). Esto provoca un incremento significativo en la carga mental de trabajo del operador, provocando un descenso en su rendimiento, llegando incluso en algunos casos a tener a abandonar alguna de sus tareas o abortar la misión. En la actualidad, este problema suele resolverse dividiendo las tareas entre varios operadores, lo que aumenta la cantidad de recursos humanos necesarios para completar la misión.

Para abordar los problemas anteriormente mencionados, en esta tesis se ha diseñado e implementado una arquitectura software adaptativa dirigida a eventos sobre la que se ha desarrollado un centro de control adaptativo distribuido capaz de reconfigurar su infraestructura en tiempo de ejecución para dar soporte a cambios en el equipo de vehículos supervisado (entre los que pueden incluirse los modelos de varios vehículos simulados en tiempo real), en los objetivos o en el entorno en el que se realiza la misión. Además, siguiendo las pautas de esta arquitectura, se ha diseñado e implementado un interfaz de usuario auto-adaptativo, que modifica la localización y el número de componentes gráficos disponibles en función de las decisiones tomadas por un controlador que incorpora dos mecanismos de ayuda al operador (políticas de adaptabilidad y transparencia) con el objeto de disminuir su carga mental de trabajo, fatiga y estrés. Esto conlleva la mejora del rendimiento de los operadores involucrados en la misión, y, por lo tanto, puede minimizar, en la medida de lo posible, el número de operadores necesarios para llevarla a cabo con éxito.

Otro aspecto importante de esta tesis es el diseño y desarrollo de diferentes marcos experimentales para probar la robustez y eficacia de cada una de los elementos clave (arquitectura, interfaz de usuario auto-adaptativo, políticas de ayuda al operador) del centro de control de tierra propuesto. Estos marcos experimentales permiten insertar el elemento bajo estudio en un entorno de simulación que genera las entradas necesarias para cada estudio que se desea realizar y que almacena sus respuestas para poder poder analizarlas a posteriori.

A continuación se describe con más detalle la aportación realizada en torno a cada uno de los objetivos principales en los que se ha articulado esta tesis.

6.1.1. Arquitectura Adaptativa e Implementación del CCT

En esta tesis se ha propuesto una arquitectura adaptativa dirigida a eventos que nos ha permitido implementar un centro de control capaz de cumplir con todos los requisitos a los que debe someterse un centro de control de tierra tradicionales (dotar a los operadores de todas las herramientas necesarias para realizar la supervisión/control de una misión que involucre un vehículo o equipo de vehículos autónomos) y, además, cuenta con una serie de características distintivas que detallaremos a continuación y que le permiten cumplir otros requisitos que habitualmente quedan fuera de rango para un centro de control: capacidad para auto-adaptarse en tiempo de ejecución para dar soporte a cambios en el equipo de vehículos monitorizados, los objetivos de la misión, el entorno en el que la misión se lleva a cabo y, lo que es más, cuenta con un interfaz gráfico auto-adaptativo capaz de reconfigurarse para potenciar la ergonomía e integrar ayudas al operador.

Esta arquitectura combina ideas de trabajos en arquitecturas y software auto-adaptativo, arquitecturas dirigidas a eventos para entornos distribuidos conectados y algunas pautas estándar de programación como son el patrón Modelo-Vista-Controlador o el patrón Observador. Aprovechando la naturaleza modular de las arquitecturas adaptativas y su concepto de bucle de control externo, y combinándolos con la naturaleza desacoplada y la capacidad de manejo de mensajes asíncronos de una arquitectura dirigida a eventos potenciada por los

patrones anteriormente mencionados, hemos implementado una infraestructura software distribuida capaz de cambiar algunos de sus módulos software en tiempo de ejecución para adaptarse a cambios en el equipo de vehículos, objetivos o entorno en el que se realiza la misión. Además, la infraestructura distribuida desarrollada puede incorporar modelos de los vehículos autónomos simulados en tiempo real, lo que nos permite utilizar el CCT como plataforma de pruebas para maniobras o situaciones críticas que podrían poner en peligro la integridad del hardware involucrado. Estas mismas características nos permiten usar el CCT adaptativo como banco de pruebas para experimentar con diversas técnicas de ayuda al operador o incluso como plataforma de entrenamiento para operadores inexpertos.

Tras la implementación del centro de control adaptativo se ha diseñado un marco experimental que pone a prueba (mediante la simulación de aquellos eventos cuya respuesta debe ser analizada con detalle antes de realizar los experimentos reales) la robustez y eficacia del CCT frente a tres de las situaciones más críticas que pueden suceder durante la ejecución de una misión. Más en concreto, en primer lugar se han puesto a prueba los mecanismos de respuesta ante caídas de red, ya que éstas pueden provocar la pérdida de comunicación con alguno de los vehículos desplegados en la misión. En segundo lugar, se prueba el CCT adaptativo frente a la incorporación y eliminación repetitiva de vehículos en la infraestructura. Por último se prueba la robustez del sistema frente a la incorporación y eliminación repetitiva de nuevos CCTs en la infraestructura, con el objeto de permitir la monitorización distribuida, con múltiples operadores, de la misión.

Por medio de las pruebas realizadas de forma sistemática con estos marcos experimentales, se ha podido verificar que el centro de control adaptativo desarrollado es una aplicación robusta y eficaz ante cambios en la infraestructura software y ante caídas de red durante la ejecución de la misión, ya que tanto los mecanismos diseñados para la recuperación de las comunicaciones como para la adaptación de la infraestructura software ante cambios en el número de vehículos

y operadores involucrados son lo suficientemente fiables y rápidos para que el CCT pueda ser utilizado en misiones de campo reales.

Finamente, los resultados obtenidos mediante las simulaciones anteriormente mencionadas han sido verificados en experimentos reales donde se utiliza el CCT adaptativo para una misión de campo perteneciente al proyecto SALACOM. En el experimento detallado en la tesis el centro de control de tierra desarrollado es utilizado para desplegar fisicamente a un vehículo autónomo (USV) en el pantano de El Atazar e incorporar un USV simulado durante el desarrollo de la misión, para a continuación poder realizar sin riesgo una maniobra automática con dos vehículos autónomos en formación en la que el vehículo simulado actúa como líder y es seguido por un USV real.

6.1.2. Interfaz de Usuario

En esta tesis también se ha desarrollado un interfaz gráfico de usuario adaptativo que debe mostrar al operador los datos más relevantes de la misión en cada instante además de contar con las herramientas necesarias para interactuar con el resto de elementos del sistema (vehículos, actuadores, etc.). Este nuevo sub-sistema adaptativo permite al centro de control mostrar la información de forma ergonómica e incorporar mecanismos de apoyo al operador (adaptabilidad y transparencia) que ayudan a disminuir su carga mental de trabajo y fatiga durante la ejecución de una misión compleja.

Dicho interfaz ha sido implementado siguiendo las pautas de la arquitectura adaptativa dirigida a eventos anteriormente mencionada y consta de 4 módulos principales: 1) el motor adaptivo dinámico, que se encarga de monitorizar el estado del sistema y tomar decisiones sobre los cambios que deben realizarse en la vista, 2) el controlador, que recibe las ordenes de cambio que emite el motor dinámico y realiza los cambios físicos en la vista, 3) la vista auto-adaptativa, que es la parte en la que se visualizan los elementos gráficos que representan los datos relacionados con la misión, y 4) la base de datos de configuración, que guarda de forma persistente información sobre la configuración de la vista, preferencias de usuario, etc.

Tras la implementación del interfaz gráfico auto-adaptativo seha realizado el diseño de nuevos marcos experimentales con el objetivo de probar mediante simulaciones la robustez y eficacia de los mecanismos de visualización y de modificación de los elementos gráficos del interfaz. En concreto, con los nuevos marcos experimentales se pone a prueba el interfaz gráfico auto-adaptativo para verificar: 1) la correcta visualización de la información y de las alarmas enviadas por los vehículos al operador, 2) la capacidad de alterar la ubicación de los elementos gráficos para la aplicación de ayudas al operador relacionadas con la adaptabilidad, y 3) la capacidad de alterar la cantidad de elementos gráficos mostrados durante la aplicación de ayudas al operador relacionadas con la transparencia.

El conjunto de experimentos realizados nos ha permitido determinar que el interfaz gráfico es capaz de llevar a cabo de forma eficaz y robusta las tareas para las que ha sido diseñado, realizándolas además en unos tiempos moderados que se consideran aceptables para el correcto desarrollo de las tareas del operador durante una misión de campo con vehículos reales.

Finalmente, para verificar los resultados obtenidos mediante simulación, se pone a prueba el interfaz gráfico adaptativo con ayudas al operador en un experimento real dentro del marco del proyecto SALACOM. En la misión de campo detallada en la tesis, se utiliza el CCT para la supervisión de unas pruebas de integración y cooperación de diferentes tipos de vehículos, en las que se cuenta con dos USVs reales que realizan maniobras en el pantano de El Atazar. Además, durante la última fase del experimento, se muestra la correcta integración en tiempo de ejecución de un cuatrirrotor simulado al equipo colaborativo de USVs. La simulación del cuatrirrotor está justificada en este caso, ya que su equipo sensorial es el responsable de detectar un vertido tóxico (inexistente) en el pantano y, además, la peligrosa maniobra de despegue y aterrizaje del cuatrirrotor sobre un USV debe ser simulada en las fases iniciales de integración.

6.1.3. Ayudas al Operador

Para mitigar los efectos adversos que el exceso de carga de trabajo mental, el estrés y la fatiga provocan en el rendimiento del operador, se han implementado

en el interfaz gráfico del CCT los mecanismos necesarios para proporcionar ayuda al operador mediante las estrategias de adaptabilidad y transparencia.

Estas estrategias han sido seleccionadas ya que la primera (adaptabilidad) modifica la posición de los elementos gráficos (situando los más importantes en las zonas prioritarias de la pantalla) en cada instante de la misión, y la segunda (transparencia) ayuda al operador a concentrarse en los elementos más importantes en cada instante, ocultando aquellos que se consideran irrelevantes para la realización de la tarea en curso.

Para validar la eficacia de las ayudas al operador propuestas, se ha diseñado un marco experimental que nos ha permitido medir el rendimiento de un grupo heterogéneo de 24 individuos. Para ello, tras la realización de una misión de entrenamiento que tiene como objetivo paliar los efectos de la curva inicial de aprendizaje de la herramienta durante el conjunto de experimentos principales, cada individuo lleva a cabo dos misiones que le someterán sistemáticamente a una serie de situaciones en las que deben realizar tareas similares a las producidas durante una misión real. Dichas tareas tienen el propósito de someterle a una carga de trabajo suficiente para poner a prueba las ayudas al operador. Para una cuantificación fiable y robusta del rendimiento del operador, se han medido las siguientes variables durante los experimentos: 1) la carga de trabajo subjetiva del operador durante el transcurso de la misión, 2) el tiempo empleado sobre tareas primarias (aquellas asociadas a los eventos relacionados con los vehículos y el desempeño de la misión) , 3) el tiempo de reacción sobre tareas secundarias (aquellas diseñadas ex profeso para poner a prueba al operador y que no tienen relación directa con la misión), y 4) la carga mental de trabajo subjetiva a posteriori (mediante el test NASA-TLX).

Los resultados obtenidos tras la realización de estas pruebas nos indican que las ayudas al operador reducen notablemente la carga mental de trabajo, tiempo empleado en las tareas y tiempo de reacción del operador en la mayoría de los casos estudiados, y que se podría lograr aumentar sus beneficios con una mejor calibración de los parámetros que controlan los mecanismos de adaptabilidad y, en especial, de transparencia, ya que este último es más sensible a las caracte-

rísticas particulares de cada individuo. Por estas razones se determina que las ayudas al operador estudiadas son útiles para el objetivo propuesto y se considera razonable incluirlas en centros de control de tierra diseñados para supervisión de misiones complejas que involucren equipos cooperativos de vehículos heterogéneos de características similares a las propuestas durante las pruebas realizadas en esta tesis. Además se considera estos resultados extrapolables a otros tipos de misiones que, debido a sus características o restricciones, conlleven cambios sustanciales en el CCT (y su interfaz) siempre y cuando las características de los elementos gráficos que muestran la información permitan explotar los beneficios de las ayudas propuestas (adaptabilidad y transparencia), quedando los resultados sujetos a ciertas variaciones derivadas de dichos cambios.

6.2. Trabajo Futuro

A continuación se presentan algunas de las líneas de trabajo futuras más interesantes que se podrían realizar para dar continuidad a esta tesis:

- Se considera que la aplicación de políticas de ayuda al operador como las propuestas (adaptabilidad y transparencia) es una línea de investigación interesante, ya que en el conjunto de experimentos realizados durante la elaboración de esta tesis se han obtenido resultados relativos al aumento del rendimiento del operador bastante prometedores. Realizando experimentos adicionales, se podría obtener un método de calibración de los parámetros que activan los mecanismos de adaptabilidad y transparencia basándonos en las características de cada operador o en un conjunto de características que nos permitan agruparlos en diferentes conjuntos para obtener mejores resultados de los cuantificadores utilizados (descenso en los tiempos empleados para tareas primarias, en los tiempos de reacción para tareas secundarias y en la carga de trabajo mental subjetiva) para caracterizar los beneficios que los operadores obtienen al utilizar las políticas de ayuda.

- Otra de las líneas de trabajo más interesantes que podríamos seguir es la de sacar partido al centro de control adaptativo como banco de pruebas. El centro de control está formado por una infraestructura distribuida modular de naturaleza desacoplada que nos permite cambiar o integrar fácilmente módulos software que alteren su comportamiento. Esta característica es un valor añadido de nuestro CCT, ya que nos puede permitir, entre otras cosas, probar nuevas políticas de ayuda al operador, compararlas con las ya implementadas y determinar cuáles son más eficaces.

- También se considera interesante el análisis del comportamiento de los mecanismos de adaptabilidad y transparencia aplicados a centros de control que cuenten con otras características o requisitos en la información mostrada al operador, como ocurre en los casos de centros de control para enjambres de vehículos, donde la información y alertas no suele ser mostrada de forma individual, sino sobre la realización de tareas o desvío de la misión.

- Por último, se podría estudiar la integración de técnicas de inteligencia artificial y/o aprendizaje máquina en el motor de toma de decisiones del interfaz, ya que se considera que estas características dotarían al sistema del potencial de mejorar el rendimiento de casi cualquier operador que se involucre en el desarrollo de una misión compleja. Podrían integrarse, por ejemplo, mecanismos que realicen el aprendizaje de las preferencias de cada operador, analicen sus características y nos ayuden a generalizar la mejor personalización ergonómica basada en ciertas características previamente registradas en el perfil. También se considera interesante la integración de mecanismos de predicción de subidas abruptas en la carga de trabajo mental basándose en situaciones concretas previamente registradas o en cuantificadores indirectos (tiempos empleados, tiempos de reacción, tasas de error, etc.) tanto para la aplicación de medidas paliativas como para poder anticipar el momento de mayor carga de trabajo (o estrés) del usuario para tomar medidas preventivas.

Parte I

Apéndices

Apéndice A

Base de Datos de Configuración

En este apéndice se mostrará un esquema de la estructura de la base de datos de configuración descrita en la Sección 4.4 así como de la información que en ella se guarda. Además de la información que aquí aparece, contamos con un sistema de archivos en los que se guardan los resultados de los experimentos y el comportamiento del operador (interacciones con el interfaz gráfico), además de los archivos XML utilizados para configurar algunos parámetros para la conexión automática de algunos módulos remotos (por ejemplo, los vehículos) que se detallan en la Sección 3.1.1. La información que figura en la base de datos se podría ampliar si fuese necesario debido a requisitos relacionados con la misión, operador o entorno que requieran la inclusión de nuevos mecanismos o propiedades en el CCT (o su interfaz).

En la Figura A.1, se muestra la definición de cada una de las tablas de la base de datos en la que se incluyen claves primarias, externas, tipo de datos de cada uno de los campos y una tupla de ejemplo para cada una de las tablas. A continuación se describirán brevemente los datos almacenados en dichas tablas y para qué se utilizan:

Interfaz: esta tabla está constituida por tres campos numéricos enteros, un identificador único para cada una de las configuraciones del interfaz y los dos valores (horizontal y vertical) que representan el tamaño que tendrá el interfaz gráfico del centro de control dependiendo de la resolución del

equipo en el que se ejecute. Al iniciarse la ejecución del centro de control, éste lanza un comando que consulta al equipo la resolución con la que está configurado, en base a la cual se cargan los datos de esta tabla para que el centro de control se muestre correctamente dentro de los límites de la pantalla.

Grupo: una vez establecidos los límites del interfaz gráfico del centro de control en la pantalla disponible, el sistema carga de esta tabla los valores que definen los límites de los grupos de elementos gráficos (un grupo por cada vehículo) dentro del espacio del interfaz. Esta tabla cuenta con un identificador entero único de grupo para cada configuración del interfaz de la tabla anterior y cuatro valores enteros que representarán los límites superior, inferior, izquierdo y derecho del grupo. Estos valores indican al sistema como debe colocar en el interfaz los elementos gráficos, ya que dependiendo de los límites marcados, el sistema rellena el hueco disponible con filas de elementos gráficos colocados de izquierda a derecha o con columnas de elementos colocados de arriba a abajo. Además, esta tabla almacena el índice de transparencia, que marca el número de elementos gráficos de dicho grupo que son visibles por defecto al comienzo de la misión.

Prioridad: se trata de la tabla en la que se almacenan los valores de orden y coloreado prioritarios de cada grupo de elementos gráficos de la tabla anterior. Consta de un identificador entero único para cada tupla de prioridades asignado a cada grupo de elementos gráficos. Además cuenta con dos valores de prioridad representados por dos cadenas de caracteres. La cadena Prioridad1 indica en qué posición deben colocarse los elementos de más importancia dentro de ese grupo (por ejemplo "arriba" representa que deben colocarse de arriba a abajo e "izquierda" representa que deberían colocarse de izquierda a derecha). Prioridad2 indica el color preferido para mostrar los elementos más importantes (por ejemplo "rojo" indica que el color preferido sería rojo, por lo que las alarmas aparecerían destacadas con una franja roja). En esta tabla se almacenan una serie de tuplas que

indican los valores por defecto de prioridad para cada grupo y además, si se han almacenado previamente, los valores de preferencia de un usario concreto para dicho grupo y que se cargarán cuando el sistema detecte que es ese usuario el que ha ingresado en el centro de control.

Operador: esta tabla guarda elementos relacionados con el perfil de operador. El identificador único esta representado por la cadena de caracteres con el nombre de usuario. También se almacena la contraseña codificada y un enlace a la tupla que le indica al sistema sus preferencias ergonómicas, si es que ha decidido cambiarlas (en caso contrario se tomarían los valores por defecto), almacenadas en la tabla Prioridad. Por último también se guarda un enlace a los datos guardados sobre este operador durante las misiones llevadas a cabo.

Datos: esta tabla contiene tuplas formadas por tres elementos. Los dos primeros, identificador de datos (valor entero) e identificador de misión (cadena de caracteres) forman un identificador único compuesto que relaciona cada misión con el operador que la llevó a cabo. Además, el tercer valor de la tupla representa mediante una cadena de caracteres la ubicación del archivo en el que se almacenan los datos del operador durante una misión.

Arbol: es la tabla que almacena los valores de beneficio y coste de cada nodo del árbol de decisión y que serán utilizados para calcular el ratio de prioridad de cada elemento gráfico en un instante determinado de la misión en ejecución. Estos valores han sido calculados previamente para ajustar la prioridad de los elementos cuando el sistema responde a una serie de eventos (por ejemplo, si se recibe una alarma de "batería baja", se llega al nodo que contiene los valores necesarios para mostrar la batería en una de las posiciones más prioritarias del interfaz gráfico para que el operador lo localice lo antes posible). Estos valores pueden ser modificados si al analizar los datos de un operador durante las misiones realizadas se detecta una conducta del operador que indica que el elemento cuyos valores están en ese nodo debe mostrarse o no mostrarse cuando se dan unas condiciones

determinadas durante una misión (por ejemplo, si siempre que hay una alarma de velocidad el usuario solicita las coordenadas del vehículo además del velocímetro, se estudiaría incluir en el nodo unos valores de mayor prioridad para dicho elemento gráfico). La tupla está formada por un identificador único formado por la combinación de un entero y una cadena de caracteres que representa cada uno de los elementos gráficos cuyos valores han de alterarse cuando se llega a dicho nodo, a los que siguen los valores enteros de coste y beneficio elegidos para ese elemento gráfico.

TiempoEstimado: en esta tabla se almacenan los valores de tiempo estimado máximo del que se dispondrá para realizar las tareas programadas que deben realizarse cuando sucede un evento durante la misión. Cada tupla constará de un identificador único de tarea representado por una cadena de caracteres y de un entero que representa el tiempo estimado para esa tarea en segundos.

En la Figura A.2, se muestra una tabla relacional esquemática en la que se resumen las relaciones entre cada una de las tablas de la base de datos.

Interfaz = (**id_if**: integer, horizontal: integer, vertical: integer)
Ejemplo tupla → (1,640,480)

Grupo = (**id_grupo**: integer, id_if: integer, lim_sup: integer, lim_inf: integer, lim_izq: integer, lim_der: integer, transparencia: integer)
Ejemplo tupla → (1,1,10,540,10,110,4)

Prioridad = (**id_prior**: integer, id_grupo: integer, prioridad1: varchar, prioridad2: varchar)
Ejemplo tupla → (1,1,arriba,rojo)

Operador = (**id_op**: varchar, pass: md5, id_prior: integer, id_data: integer)
Ejemplo tupla → (operador1 ,ae2b1fca515949e5d54fb22b8ed95575,1,2)

Datos = (**id_data**: integer, **id_mision**: varchar, dir_archivo: varchar)
Ejemplo tupla → (2,mision1_10_06_19,c:/misiones/mision1_10_06_19)

Arbol = (**id_nodo**: integer, **id_elem**: varchar, beneficio: integer, coste: integer)
Ejemplo tupla → (1,velocimetro,10,8)

TiempoEstimado = (**id_tarea**: varchar, tiempo: integer)
(waypoint,60)

Figura A.1: Esquema de Definiciones de las Tablas de la Base de Datos de Configuración

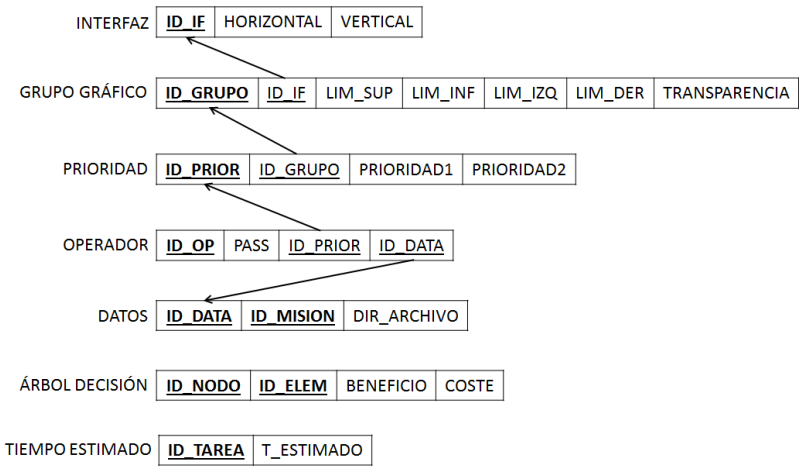


Figura A.2: Esquema Relacional de las Tablas de la Base de Datos de Configuración

Bibliografía

Y así, del mucho leer y del poco dormir, se le secó el celebro de manera que vino a perder el juicio.

Miguel de Cervantes Saavedra

DJIInspire2. https://www.dji.com/es/inspire-2?site=brandsite&from=landing_page, 2019.

UAS Ground Control Stations. http://www.uas-europe.se/index.php/products/ground-control-stations/portable-gcs, 2019.

Airbus. UAV Atlante. https://www.airbus.com/defence/uav.html, 2013.

Alpha-Unmanned-Systems. GCase alpha unmanned systems. https://alphaunmannedsystems.com/ground-control-station/?lang=es, 2019.

Alti-UAS. ALT GCSs. https://www.altiuas.com/ground-control-station/, 2019.

Amazon. Amazon prime air. https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011, 2016.

Amoui, M., Derakhshanmanesh, M., Ebert, J. y Tahvildaria, L. Achieving dynamic adaptation via management and interpretation of runtime models. *Journal of Systems and Software*, 2012.

ARQUER, I. y NOGAREDA, C. Ntp 544: Estimación de la carga mental de trabajo: el método nasa tlx. *Centro Nacional de Condiciones de Trabajo*, página 6, 2000.

ASTM. Committee F41 on unmanned maritime vehicle systems (umvs). [Online] https://www.astm.org/COMMITTEE/F41.htm, 2017.

ASV. Asview control system. [Online] https://www.asvglobal.com/autonomous-control-system/, 2019.

BÜRKLE, A., SEGOR, F., KOLLMANN, M. y SCHÖNBEIN, R. Universal ground control station for heterogeneous sensors. *Journal On Advances in Telecommunications, IARIA*, vol. 3(3), páginas 152–161, 2011.

BURMEISTER, H. C., BRUHN, W., RODSETH, O. J. y PORATHE, T. Autonomous unmanned merchant vessel and its contribution towards the e-navigation implementation: The MUNIN perspective. *International Journal of e-Navigation and Maritime Economy*, vol. 1, páginas 1–13, 2014.

BUXTON, W., HARRISON, B. y VICENTE, K. Graphical user interface with optimal transparency thresholds for maximizing user performance and system efficiency. [Online]https://patents.google.com/patent/US6118427A/en, 2000. Visitado 2019.

CETC. China electronics technology group corporation. `http://en.cetc.com.cn/`, 2017.

CHEN, J., PROCCI, K., BOYCE, M., WRIGHT, J., GARCIA, A. y BARNES, M. Situation awareness-based agent transparency(no. arl-tr-6905). 2014.

CHENG, S.-W. y GARLAN, D. Handling uncertainty in autonomic systems. En *International Workshop on Living with Uncertainty*. 2007.

CHENG, S.-W., GARLAN, D. y SCHMERL, B. Architecture-based self-adaptation in the presence of multiple objectives. *SEAMS 06 Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems*, páginas 2–8, 2006.

CHENG, S.-W., HUANG, A.-C., GARLAN, D., SCHMERL, B. y STEENKISTE, P. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Proceedings of the International Conference on Autonomic Computing (ICAC 04)*, páginas 276 – 277, 2004.

DE LA CRUZ, J. M., LOPEZ-OROZCO, A, J., BESADA PORTAS, E. y ARANDA ALMANSA, J. Control de formaciones de vehículos marinos de superficie con restricciones de entrada. páginas 1044–1051. CEA, 2016.

DE LA CRUZ, J. M., LOPEZ-OROZCO, A, J., BESADA PORTAS, E., MORENO SALINAS, D. y ARANDA ALMANSA, J. Seguimiento de caminos para formaciones de vehículos marinos de superficie. 2014.

DE LA CRUZ, J. M., LOPEZ-OROZCO, J. A., BESADA-PORTAS, E. y ARANDA-ALMANSA, J. A streamlined nonlinear path following kinematic controller. En *2015 IEEE International Conference on Robotics and Automation (ICRA)*, páginas 6394–6401. IEEE, 2015.

CUMMINGS, M. L., HOW, J. P., WHITTEN, A. y TOUPET, O. The impact of human–automation collaboration in decentralized multiple unmanned vehicle control. *Proceedings of the IEEE*, vol. 100(3), páginas 660–671, 2012.

DAMIANOU, N., DULAY, N., LUPU, E. y SLOMAN, M. The ponder policy specification language. *Policies for Distributed Systems and Networks series Lecture Notes in Computer Science*, vol. 1995, páginas 18–38, 2001.

DASHOFY, E. M., VAN DER HOEK, A. y TAYLOR, R. N. Towards architecture-based self-healing systems. *WOSS '02 Proceedings of the first workshop on Self-healing systems*, 2002.

DAUM, T. y SARGENT, R. Experimental frames in a modern modeling and simulation system. *IEE Transactions*, vol. 33, páginas 181–192, 2001.

DINOCERA, F., CAMILLI, M. y TERENZI, M. A random glance at the flight deck: Pilots scanning strategies and the real-time assessment of mental workload.

Journal On Cognitive Engineering and Decision Making, vol. 1, páginas 271–285, 2007.

DJI-TECHNOLOGY. DJI. `https://www.dji.com/es`, 2019.

EHANG. ehang aav. `https://www.ehang.com/news/365.html`, 2018.

FILIPPONI, L., VITALETTI, A., LANDI, G., MEMEO, V., LAURA, G. y PUCCI, P. Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors. *Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM)*, 2010.

FOURES, D., ALBERT, V. y NKETSA, A. Simulation validation using the compatibility between simulation model and experimental frame. *Proceedings of the 2013 Summer Computer Simulation Conference*, página 9, 2013.

GARLAN, D., CHENG, S.-W., HUANG, A.-C., SCHMERL, B. y STEENKISTE, P. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer (IEEE Computer Society)*, vol. 7 (Issue: 10), páginas 48–54, 2004.

GENERAL-ATOMICS. General Atomics Ground Control Stations. `http://www.ga-asi.com/ground-control-stations-gcs`, 2019.

GENERAL ATOMICS, U. General Atomics UAVs. `http://www.ga-asi.com/aircraft-platforms`, 1995-2017.

HALLSTEINSEN, S., GEIHS, K., PASPALLIS, N., ELIASSEN, F., HORN, G., LORENZO, J., MAMELLI, A. y PAPADOPOULOS, G. A development framework and methodology for self-adapting applications in ubiquitous computing environments. *Journal of Systems and Software*, 2012.

HART, S. Nasa-task load index (nasa-tlx); 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 50, páginas 904–908, 2006.

HART, S. y STAVELAND, L. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in Psychology*, vol. 52, páginas 139–183, 1988.

HELLDIN, T. Transparency for future semi-automated systems(doctoral dissertation). 2014.

HEO, J., KIM, S. y KWON, Y. Design of ground control station for operation of multiple combat entities. *Journal of Computer and Communications*, vol. 4, páginas 66–71, 2016.

HILD, D. R. Discrete event system specification (DEVS) distributed object computing (DOC) modeling and simulation. *PhD Dissertation, Electrical and Computer Engineering Department University of Arizona*, 2000.

HONG, W. E., LEE, J. S., RAI, L. y KANG, S. J. Rt-linux based hard real-time software architecture for unmanned autonomous helicopters. *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*, 2005.

HUSSEIN, A., GHIGNONE, L., NGUYEN, T., SALIMI, N., NGUYEN, H., WANG, M. y ABBASS, H. Towards bi-directional communication in human-swarm teaming: A survey. *Cornell University arXiv*, páginas 1–15, 2018.

INTEL. Intel ligth show. `https://www.intel.es/content/www/es/es/technology-innovation/aerial-technology-light-show.html`, 2018.

IROBOT, S. Small unmanned ground vehicle. `https://www.army-technology.com/projects/irobot-310-sugv-us/`, 2019.

ISCAR–UCM. USV SALACOM ISCAR. `http://www.dacya.ucm.es/area-isa/media/imagenes.html`, 2019.

JOVANOVIC, M. y STARCEVIC, D. Improving design of ground control station for unmanned aerial vehicle: Borrowing from design patterns. *Tenth International Conference on Computer Modeling and Simulation (uksim 2008)*, 2008.

JOVANOVIC, M. y STARCEVIC, D. Software architecture for ground control station for unmanned aerial vehicle. *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2010.

KEPHART, J. y CHESS, D. The vision of autonomic computing. *Computer*, vol. 17, páginas 41–50, 2003.

KONGSBERG-MARITIME. REMUS UUV. `https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/D5682F98CBFBC05AC1257497002976E4?OpenDocument`, 2019.

LADDAGA, R. y ROBERTSON, P. Self adaptive software: A position paper. En *SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems*, vol. 31, página 19. Citeseer, 2004.

LIAO, W., ZHANG, W., ZHU, Z. y JI, Q. A real–time human stress monitoring system using dynamic bayesian network. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, página 8, 2005.

LINDEMUTH, M., MURPHY, R., STEIMLE, E., ARMITAGE, W., DREGER, K., ELLIOT, T., HALL, M., KALYADIN, D., KRAMER, J., PALANKAR, M., PRATT, K. y GRIFFIN, C. Sea robot-assisted inspection. *IEEE Robotics Automation Magazine*, vol. 18(2), páginas 96–107, 2011. ISSN 1070-9932.

ISCAR UNIVERSIDAD COMPLUTENSE DE MADRID, G. Agricultura de precision. `http://www.dacya.ucm.es/area-isa/media/videos.html`, 2019.

MAGEE, J., DULAY, N., EISENBACH, S. y KRAMER, J. Specifying distributed software architectures. *Software Engineering - ESEC 95 Lecture Notes in Computer Science*, vol. 989, páginas 137–153, 1995.

MARINE ADVANCED RESEARCH, I. 8-wam-v marine advanced research. `https://www.wam-v.com/8-wamv-usv/`, 2019.

MEIER, L. MAVlink. `http://qgroundcontrol.org/mavlink/start`, 2009.

MERCADO, J. E., RUPP, M. A., CHEN, J. Y. C., BARNES, M. J., BARBER, D. y PROCCI, K. Intelligent agent transparency in human-agent teaming for multi-UxV management. *Human Factors*, vol. 58(3), páginas 401–415, 2016.

MICHELSON, B. M. Event-driven architecture overview. *Patricia Seybold Group Research Service (2006)*, 2006.

MILREM ROBOTICS, R. Multiscope search & rescue. `https://milremrobotics.com/multiscope/`, 2019.

MITTAL, A., RISCO-MARTÃN, J. L. y ZEIGLER, B. P. Devs/soa: A cross-platform framework for net-centric modeling and simulation in devs unified process. *SIMULATION*, vol. 85(7), páginas 419–450, 2009.

MITTAL, S. y RISCO-MARTÍN, J. L. DEVSML 3.0: Incorporating Docker and Microservices for Rapid Deployment of DEVS Farm in Cloud Environment. En *Proceedings of the 2017 Spring Simulation Multionference (SpringSim 2017)*. 2017a.

MITTAL, S. y RISCO-MARTÍN, J. L. *Guide to Simulation-Based Disciplines: Advancing Our Computational Future*, capítulo Simulation-based Complex Adaptive Systems, páginas 127–151. Springer, 2017b.

MUPPARAPU, S. S., CHAPPELL, S. G., KOMERSKA, R. J., BLIDBERG, D. R., NITZEL, R., BENTON, C., POPA, D. O. y SANDERSON, A. C. Autonomous systems monitoring and control (asmac)-an auv fleet controller. En *Autonomous Underwater Vehicles, 2004 IEEE/OES*, páginas 119–126. IEEE, 2004.

MURPHY, R. R., AUSMUS, M., BUGAJSKA, M., ELLIS, T., JOHNSON, T., KELLEY, N., KIEFER, J. y POLLOCK, L. Marsupial–like mobile robot societies. *Proceedings Int. Conf. Autonomous Agents*, páginas 364–365, 1999.

MURPHY, R. R., STEIMLE, E., GRIFFIN, C., CULLINS, C., HALL, M. y PRATT, K. Cooperative use of unmanned sea surface and micro aerial vehicles at hurricane Wilma. *Journal of Field Robotics*, vol. 25, páginas 164–180, 2008.

NASA. Mars oportunity rover. https://www.jpl.nasa.gov/missions/mars-exploration-rover-opportunity-mer/, 2003a.

NASA. Mars spirit rover. https://www.jpl.nasa.gov/missions/mars-exploration-rover-spirit-mer-spirit/, 2003b.

OCEANALPHA. Esm30 oceanalpha. https://www.oceanalpha.com/product-item/esm30/, 2019.

OCTOPUS-SYSTEMS. Octopus GCS. http://octopus.uavfactory.com/uav-payloads-equipment/ground-control-station, 2019.

OREIZY, P., GORLICK, M. M. y TAYLOR, R. N. An architecture-based approach to self-adaptive software. *Intelligent Systems and their applications*, vol. 14(3), 1999a.

OREIZY, P., GORLICK, M. M., TAYLOR, R. N., HEIMBIGNER, D., JOHNSON, G., MEDVIDOVIC, N., QUILICI, A., ROSENBLUM, D. S. y WOLF, A. L. An architecture-based approach to self-adaptive software. 1999b.

OTAN. Standard interfaces of uav control system (ucs) for nato uav interoperability, ed. 3. NATO standardization agency (nsa). [Online] http://nso.nato.int/nso/nsdd/listpromulg.html, 2012.

PARROT-SA. Parrot. https://www.parrot.com/es, 2019.

PATTERSON, M. C., MULLIGAN, A. y BOITEUX, F. Safety and security applications for micro-unmanned surface vessels. En *2013 OCEANS-San Diego*, páginas 1–6. IEEE, 2013.

PEREZ, D., MAZA, I., CABALLERO, F., SCARLATTI, D., CASADO, E. y OLLERO, A. A ground control station for a multi-uav surveillance system. *Journal of Intelligent & Robotic Systems*, vol. 69, páginas 119–130, 2013a.

PEREZ, D., MAZA, I., CABALLERO, F., SCARLATTI, D., CASADO, E. y OLLERO, A. A ground control station for a multi-uav surveillance system: Design

and validation in field experiments. *Journal of Intelligent & Robotic Systems*, páginas 1–7, 2013b.

QGROUNDCONTROL. A uav control station. [Online] http://qgroundcontrol.com/, 2008.

RIBAS, D., PALOMERAS, N., RIDAO, P., CARRERAS, M. y MALLIOS, A. Girona 500 auv: From survey to intervention. *IEEE ASME Transactions on Mechatronics*, vol. 17(1), páginas 46–53, 2012.

ROSCOE, A. Assessing pilot workload. why measure heart rate, hrv and respiration? *Biological Psychology*, vol. 34, páginas 259–287, 1993.

ROUSE, W., EDWARDS, S. y HAMMER, J. Modeling the dynamics of mental workload and human performance in complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, página 27, 1993.

RUBIO, S., DÍAZ, E., MARTÍN, J. y PUENTE, J. Evaluation of subjective mental workload: A comparison of swat, nasa–tlx, and workload profile methods. *Applied Psychology*, vol. 53, páginas 61–86, 2004.

SEBASTIÁN, O. y DELHOYO, M. *La carga mental de trabajo*, páginas 1–51. Instituto Nacional de Seguridad e Higiene en el Trabajo, 2004.

SERVER-SENT-EVENTS. Server-sent events. `https://www.w3.org/TR/eventsource/`, 2013.

SUTTON, R., SHARMA, S. y XCUAO, T. Adaptive navigation systems for an unmanned surface vehicle. *Journal of Marine Engineering and Technology*, vol. 10, páginas 3–20, 2011.

UNIVERSALGCS. Universalgcs–lockheedmartin, products. `https://www.lockheedmartin.com/en-us/products/cdl-systems/universal-ground-control-station.html`, 2019.

UNIVERSIDAD CANTABRIA, U. UUVs marine vehicles experimentation laboratory of cantabria. https://www.marinelabcantabria.unican.es/vehicles.php, 2019.

VEENA, M. Development of experimental frame and abstract devs models to support scope network expansion. *PhD Dissertation, Electrical and Computer Engineering Department University of Arizona*, 2005.

DE WAARD, D. *The Measurement of Drivers Mental Workload*, vol. 1. Groningen University, Traffic Research Centre, 1 edición, 1996.

WALTER, B. E., KNUTZON, J. S., SANNIER, A. V. y OLIVER, J. H. Virtual uav ground control station. En *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*. 2004.

WEBHOOKS. webhooks. https://developer.github.com/webhooks/, 2007.

WEBSOCKET. websocket. https://www.websocket.org/, 2010.

WGSM. Wave glider management system. [Online] https://www.liquid-robotics.com/wave-glider/software/, 2019.

ZEIGLER, B., PRAEHOFER, H. y KIM, T. *Theory of Modeling and Simulation edition*, vol. 1. Academic Press, 2 edición, 2000.