https://doi.org/10.17979/spudc.9788497498081.0979

CENTRO DE CONTROL VERSÁTIL PARA EQUIPOS DE VEHÍCULOS HETEROGÉNEOS: ESTADO ACTUAL Y MEJORAS FUTURAS

Juan A. Bonache Seco, Jose A. López Orozco, Eva Besada Portas, Jesús M. de la Cruz
jabonache@ucm.es, jalo@ucm.es, ebesada@ucm.es, jmcruz@ucm.es
Universidad Complutense de Madrid

Resumen

Los vehículos autónomos o tripulados de forma remota hacen necesaria una estación de control de tierra que permita visualizar y, en determinadas situaciones, controlar o enviar instrucciones a dichos vehículos. Cuando estas tareas se vuelven más complejas y requieren varios vehículos en colaboración o incluso que el número o naturaleza de los mismos varíe en base a sus necesidades, el manejo de datos y su visualización se vuelve complicado. Este artículo plantea una posible solución al problema del manejo de datos, control y visualización de un equipo de vehículos autónomos o tripulados de forma remota y a los ocasionados por variaciones en el número y naturaleza de los vehículos o al entorno cambiante de la misión. Si se contempla además la relevancia del flujo de datos en el rendimiento del operador, añadimos una nueva dimensión al problema. También, analizaremos los aspectos relevantes que deben desarrollarse en el centro de control para que pueda adaptarse y aprender en tiempo de ejecución cómo cambiar la configuración del entorno gráfico de la forma más beneficiosa posible en base a los datos recopilados y las características del operador.

Palabras clave: Estación de Control de Tierra (Ground Control Station - GCS), Adaptabilidad, Transparencia, Vehículos Autónomos (UAV, USV, UGV)

1. INTRODUCCIÓN

La proliferación actual de vehículos autónomos o tripulados de forma remota para fines comerciales [1], militares [2, 3, 4] o incluso de entretenimiento [5, 6] hace necesario contar con una aplicación de interfaz gráfico que permita visualizar los datos del vehículo, los del entorno que le rodea y, en determinadas situaciones, de la misión que debe cumplir. Además, deberá permitir su control o envío de ciertas consignas o instrucciones.

Los centros de control de tierra pueden disponer de diversas configuraciones hardware y software. El software, puede ser de código abierto [7, 8], militar [9] o comercial [10]. Respecto al hardware, puede bastar un dispositivo donde instalar el software o ser necesarios sistemas más complejos y robustos que varían en tamaño, desde los fijos en un edificio, contenedor de camión o similar [11] hasta los centros de control portátiles [12, 13].

El primer problema que deberemos afrontar cuando comenzamos el desarrollo de este tipo de software es dotarlo de la capacidad de monitorizar y controlar diferentes tipos de vehículos autónomos. Se trata de un problema asumible si el objeto de la aplicación es fijo pero se complica cuando el numero de vehículos cooperantes y/o la misión pueden cambiar. Por este motivo, en el marco del proyecto DPI2013-46665-C2 (SALACOM, Sistema Autónomo de Localización y Actuación ante Contaminantes en el Mar) se ha desarrollado un Centro de Control Versátil (CCV, [14]) que se presenta en este trabajo y que permite de forma rápida y sencilla reconfigurar una estación de control de tierra para cualquier tipo de misión y un número variable de vehículos autónomos de naturaleza heterogénea.

Una vez superado el problema antes mencionado, nos centraremos en otros objetivos de estas aplicaciones, como que el usuario se sienta cómodo y no tarde en asimilar su funcionamiento, o que no se sienta abrumado con la cantidad de datos mostrados por pantalla. Para ello, la interfaz gráfica de usuario debe ser intuitiva y fácil de manejar, con iconos representativos que muestren los datos de forma clara y limpia. Cuando el ámbito de la aplicación es el ocio, cumplir estos requisitos es más fácil ya que normalmente hay un único vehículo y el centro de control hace las veces de un "control remoto"[7, 8]. Si el cometido del vehículo es realizar una misión de forma autónoma, la cantidad y precisión de los datos a mostrar suele ser mayor, por lo que aumenta la dificultad de su visualización simultánea. Además, si el flujo de datos es demasiado grande, puede provocar efectos adversos como distracciones, descenso de rendimiento, tiempos de reacción, etc. Este problema del tratamiento del flujo de datos mostrado por pantalla suele enfocarse de dos formas, por un lado las que precisan sensorización del operador y por otro las que centran su monitorización en aspectos referen-

tes al propio programa, su evolución en tiempo de ejecución y cómo el usuario lo utiliza.

Los estudios que requieren sensorización del operador (y, en ocasiones, del entorno) deciden cómo modificar el flujo de datos mostrado en base a sus aptitudes físicas y psicológicas o su estado en el momento de desarrollo de su actividad (cansancio, distracciones, estrés, etc.). Por ejemplo, Amditis et al. proponen AIDE (Adaptive Integrated Driver-vehicle interfacE, [15, 16]), una solución holística basada en el patrón Modelo-Vista-Controlador aplicada al sector de automoción para realizar un sistema adaptativo de asistencia al conductor que decide en base a características y monitorización del conductor, además de las características del entorno. También en automoción, A. Polychronopoulos et al. introducen una solución al déficit de atención (hipovigilancia) con el sistema AWAKE [17] que monitoriza al conductor y le muestra alertas en caso de detectar una disminución de atención. Además NietoLee et al. proponen Virtual Control Room (VCR, [18, 19]), un sistema pensado para entornos industriales con un intenso flujo de datos. De Graaf et al. presentan también el sistema CAMMI (Cognitive Adaptive Man Machine Interface), llevado a cabo por la plataforma de desarrollo tecnológico ARTEMIS (Advanced Research and Technology for EMbedded Intelligence and Systems) con aplicaciones en aviónica, automoción y emergencias civiles diseñado para mitigar la carga de trabajo soportada por los jefes de bomberos.

En los estudios que no requieren sensorización del usuario se propone, mediante diferentes técnicas, un sistema de arquitectura modular capaz de adaptarse en tiempo de ejecución. En esta línea Magee et al. [20] proponen Darwin, un lenguaje de tipo declarativo para especificación de diseño de sistemas distribuidos. Por su parte, Nicodemus Damianou et al. [21] presentan PONDER, otro lenguaje declarativo de especificación orientado al manejo de permisos y seguridad en sistemas distribuidos. Además Oreizy et al. proponen una infraestructura adaptativa [22] y una serie de cuestiones para definir los módulos de la misma. Por su parte, Cheng, Garlan et al. proponen una serie de estudios que les han llevado al desarrollo de RAINBOW [23, 24, 25, 26], un sistema basado en una arquitectura auto-adaptativa con infraestructuras reutilizables.

La futura versión de nuestro CCV se engloba dentro de esta segunda linea de investigación, y en este trabajo se presenta un análisis de aquellas características más relevantes que deberemos tener en cuenta durante el desarrollo del mismo para convertirlo en un centro de control que se adapte a las necesidades de la misión y del usuario. También se introducirá el concepto de "transparencia" entre el software y el operador, es decir, cuánta información ha de mostrar el sistema para que la confianza del operador en el programa aumente y al utilizarlo tome las mejores decisiones en base a los consejos que el software le propone [27].

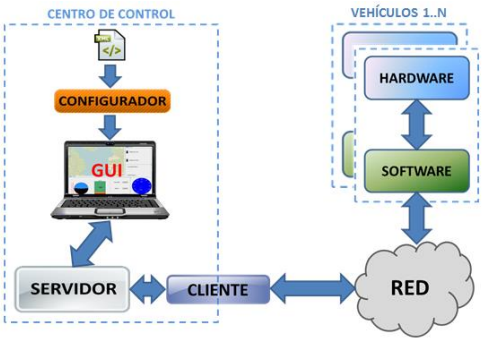


Figura 1: Esquema General del CCV

2. ESTADO ACTUAL DEL CCV

En esta sección se detalla el estado actual de nuestro Centro de Control Versátil, se desarrollará su estructura y su funcionamiento, y se pondran en relieve sus limitaciones al ser utilizado en un caso real, el control de los barcos no tripulados del proyecto SALACOM, en el que es necesario monitorizar un flujo de datos considerable.

2.1. DESCRIPCION DEL CCV

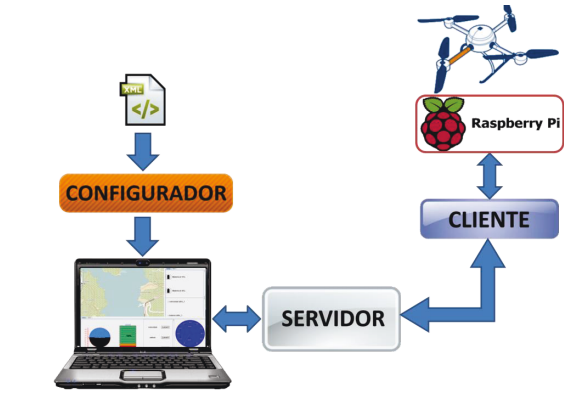
En esta sección expondremos nuestra solución actual al problema que supone diseñar un centro de control de tierra versátil y de bajo coste adaptable a las necesidades que pueden surgir de 1) la monitorización y control de un conjunto variable de vehículos no tripulados de naturaleza heterogénea (diferentes tipos de comunicaciones, datos de telemetría distintos, etc.), y 2) de la ubicación de sus componentes en base a criterios ergonómicos y preferencias del operador. Algunos de estos componentes podrán moverse, mostrarse u ocultarse y extraerse de la ventana principal del interfaz gráfico. Para ello se propondrá una infraestructura que constará de tres módulos que pueden verse en la figura 1: un interfaz gráfico modular (GUI), un sistema de configuración y un esquema de comunicaciones flexible capaz de comunicarse tanto con vehículos de cualquier naturaleza como con visores y otros centros de control realizando mínimas modificaciones.

2.1.1. SISTEMA DE CONFIGURACIÓN

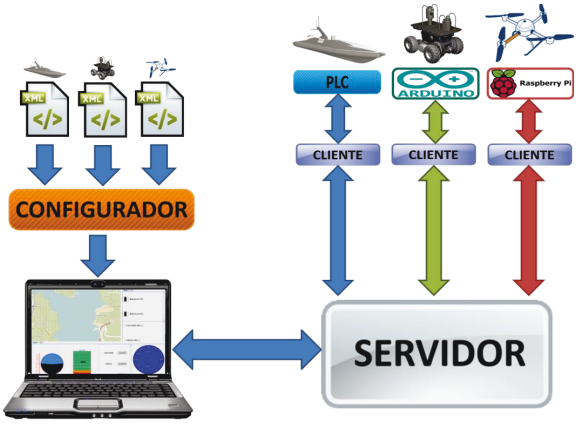
El punto clave de nuestro centro de control es el sistema que permite configurar su diseño. Para ello se ha de realizar un documento de configuración en un lenguaje de marcado standard (XML, eXtensible Markup Language) siguiendo unas reglas predefinidas en un DTD (Data Type Document) para establecer el tipo de vehículo, datos a recibir y forma de mostrarlos. Además, sirve para validar el vehículo en el comienzo de las comunicaciones.

La correcta elaboración de este documento es esencial tanto para el establecimiento de comunicaciones como para el enlace entre datos y elementos gráficos o su colocación en el interfaz de usuario, por lo que hay que dotar al sistema de la robustez necesaria para evitar errores por parte del diseñador del documento. Esta comprobación se realiza mediante un software que se encarga de la lectura del documento, comprueba la corrección de las reglas (orden correcto, sintaxis correcta, etc.) e indica, en su caso, los lugares donde se han producido los errores.

Una información más detallada sobre este documento puede encontrarse en [14], donde se podrán encontrar las reglas de construcción del archivo XML, ejemplos detallados de archivos correctos y el DTD en el que se basa el software, que hace las veces de sencillo compilador, para comprobar su corrección.



(a) Un centro de control con un vehículo



(b) Un centro de control con varios vehículos

Figura 2: Esquema de Comunicaciones

2.1.2. ESTRUCTURA DE COMUNICACIONES

Para el sistema de comunicaciones se ha implementado un software standard y reutilizable. Se ha elegido el protocolo TCP, que permitirá establecer comunicaciones wi-fi o cableadas independientemente de la plataforma y arquitectura software que cada vehículo porte, ya sea en su hardware de control integrado, en un PLC (por ejemplo Beckhoff + TwinCAT [28]) o un terminal que lleve acoplado (normalmente hardware de bajo coste y potencia de cómputo limitada como Raspberry PI, Arduino, Intel Edison, Beaglebone Black, etc.). Se establece un sistema Cliente-Servidor en el que el Centro de Control, que hace las veces de Servidor, acepta conexiones de los clientes y desempaqueta los datos que llegan por el socket asignado a cada uno de ellos. Así, cada variable se enlaza a la información del vehículo correcto y se muestra como se indicó en el fichero de configuración.

Por su parte, los Clientes dotan de flexibilidad al sistema de comunicaciones. De esta forma, en unos casos los clientes pueden desplegarse en el PC que alberga el Centro de control si el software integrado en el vehículo en cuestión es capaz de manejar peticiones de datos y recepción de consignas que lleguen del exterior (o proporciona algún tipo de librería o software para ello) como puede verse en el esquema general de la figura 1, evitando incluir ningún tipo de hardware o software adicional en el vehículo. En otros casos hace las veces de "Middleware" lanzándose éste en el Sistema Operativo o Firmware propio del vehículo (si esto fuese posible) o en el PLC o terminal hardware acoplado para comunicarse con él. Un ejemplo de la primera configuración se puede observar en la figura 2(a) (caso con un sólo vehículo) y otro en 2(b) (caso con varios vehículos heterogéneos con distintos terminales: PLC, Arduino y Raspberry PI).

Así se establece un protocolo de comunicaciones entre Cliente y Servidor en el que se envía cada dato con un identificador y tipo, de tal forma que el Servidor sabe asignarlo a un vehículo válido que se le haya indicado y mostrarlo de forma adecuada. De momento, el protocolo utilizado ha sido diseñado con el propósito de ser ligero y sencillo, pero sería posible adaptar los módulos del CCV a protocolos standard de forma prácticamente inmediata. Por ejemplo podría adaptar-

se a MAVLink (Micro Air Vehicle communication protocol [29]) añadiendo variables por medio de definiciones XML, generando el código y añadiendo en nuestro centro de control las funciones de empaquetado y desempaquetado de mensajes tal como se describe en su documentación.

También se han integrado algunos elementos de seguridad como un "handshake" (comprobaciones básicas durante la apertura de comunicaciones) que comprueba que los parámetros en Cliente y Servidor son correctos y no habrá fallos a la hora de visualizar los datos en el centro de control. Esta comprobación asegura que desde el cliente se cuente con un identificador de vehículo válido y que las variables solicitadas esten disponibles.

2.1.3. INTERFAZ GRÁFICO

Otro aspecto clave para el CCV es el diseño de un interfaz de usuario lo suficientemente versátil y robusto que sea capaz de mostrar los datos clave para cualquier tipo de vehículo, independientemente de si se trata de un UAV (Unmanned Aerial Vehicle: vehículo aéreo no tripulado), un USV (Unmanned Surface Vehicle: vehículo de superficie no tripulado) o un UGV (Unmanned Ground Vehicle: vehículo de tierra no tripulado).

Para ello, se ha diseñado un software modular jerárquico en el que hay ciertos elementos contenedores obligatorios que pueden alterar su posición dentro de la ventana principal, y una serie de elementos de control y visualización que se introducen en estos contenedores. Además, algunos elementos gráficos de visualización son concretos para ciertas tareas (por ejemplo el velocímetro o el nivel de batería en la figura 3(A)) y otros más flexibles que permiten la asignación de un tipo de dato, un nombre y una forma de representación para mostrar cualquier dato que podamos necesitar (por ejemplo, longitud y latitud en la figura 3(B)).

Los elementos contenedores principales son tres: un panel vertical donde se pueden introducir alarmas, valores numéricos y accesos directos a otros elementos gráficos que queramos poder extraer de la ventana principal; un panel horizontal donde se pueden introducir elementos gráficos o numéricos; y un mapa donde se muestra en todo momento la posición de los vehículos además de otros elementos opcionales como marcadores, simulación de ruta de vehículos, etc. Todos estos visualizadores gráficos se agrupan por vehículo en pestañas dentro de cada elemento contenedor y se les asigna un color e identificador. Además, existe la posibilidad de poner los elementos considerados más importantes por el diseñador/operador en una pestaña común a todos los vehículos que se mostrará

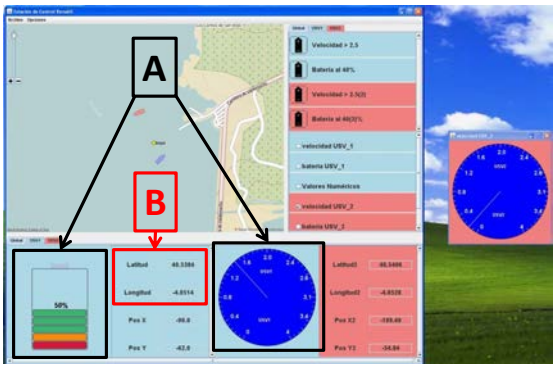


Figura 3: Interfaz Gráfico del CCV

en primer plano al iniciar la aplicación (ver paneles horizontal y vertical junto al mapa de la figura 3, donde se ven mezclados elementos de los vehículos representados por los colores azul y rojo).

La modularización del interfaz gráfico permite su reutilización y modificación teniendo en cuenta necesidades de la misión, variaciones en los vehículos que participen en la misma o incluso preferencias del operador para cada una de ellas.

2.2. ANALISIS DEL FUNCIONAMIENTO DEL CCV SOBRE UN CASO REAL

Tras comprobar que el CCV propuesto en la sección anterior cumple con los objetivos establecidos de forma robusta y eficaz, comenzó a evaluarse en términos de eficiencia y rendimiento del operador.

Esto se pudo llevar a cabo gracias a los ensayos realizados sobre un sistema real en el marco del proyecto SALACOM, en el que se cuenta con dos barcos autónomos y una infraestructura de comunicaciones que nos permitirá analizar el comportamiento del CCV. En la figura 4 podemos ver un ejemplo de configuración para un experimento con dos barcos y un CCV cuyo objetivo es monitorizar sus trayectorias y datos de telemetría en tiempo real durante el transcurso de las maniobras. En el equipo en el que se encuentra el CCV, se despliegan dos clientes que se comunican con los PLC's de los dos barcos no tripulados mediante una antena wifi de 1km de alcance. También se cuenta con el apoyo de una herramienta desarrollada para el proyecto, que hace las veces de controlador de alto nivel permitiendo al operador enviar a los barcos consignas sencillas y maniobras complejas preprogramadas.

Hemos podido comprobar que un software de las características del CCV ayuda en gran medida al correcto desarrollo de la actividad del operador y le permite prever posibles errores o peligros. No obstante, durante el despliegue de distintos expe-

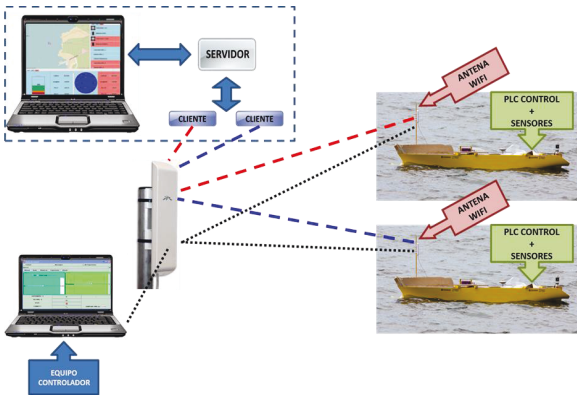


Figura 4: Ejemplo de Configuración de un Experimento en el Proyecto SALACOM

rimentos (por ejemplo, agregando un cuatrirrotor o realizando maniobras diferentes con cada uno de los barcos), se han observado algunos posibles puntos de mejora ocasionados, entre otros, por un aumento del flujo de datos o la evolución de una misión que puede requerir un cambio de configuración en tiempo real de los elementos gráficos del interfaz humano-máquina.

De esta forma se han seguido realizando experimentos con el objetivo de extraer conclusiones sobre cómo un flujo de datos demasiado intenso puede saturar el interfaz gráfico además de entorpecer la labor de la persona que lo usa. En ciertas ocasiones es necesario resaltar algunos datos importantes (como pueden ser alarmas ante situaciones críticas), disminuir la cantidad de datos mostrados o reubicar algunos elementos que en un momento dado pueden resultar irrelevantes y que distraen la atención de otros más importantes. En un sistema que también puede asistir al operador en determinados aspectos de la misión, la cantidad de información desplegada también podría ser relevante dependiendo de la persona que deba tomar la decisión.

Por estas razones, se ha decidido continuar con el desarrollo del CCV para dotarlo de los mecanismos necesarios para paliar en la medida de lo posible los efectos adversos anteriormente mencionados, producidos por las circunstancias cambiantes de las misiones realizadas y de las características de los diferentes vehículos que integran el equipo colaborativo.

3. MEJORAS FUTURAS DEL CCV

En esta sección se analizan un conjunto de aspectos fundamentales que consideramos especialmente relevantes a la hora de aumentar la versatilidad del centro de control y se introducen las posibles etapas de mejora que se realizarán próximamente.

3.1. ASPECTOS RELEVANTES ESTUDIADOS

Para tratar de mitigar los aspectos negativos de estas circunstancias, comienzan a estudiarse posibles vías de mejora con las que seguir haciendo que la aplicación evolucione. Los dos campos de investigación en los que nos centraremos son: Transparencia y Adaptabilidad.

3.1.1. TRANSPARENCIA

Dado que la cantidad de datos a mostrar crece considerablemente con cada vehículo añadido al equipo, es importante plantearse cuál es la mejor manera de mostrar su información (tipo de visualización, colocación, tamaño, etc.) y si mostrar el flujo de datos completo mejora el rendimiento del operador o, por el contrario, lo empeora.

El objetivo es lograr mostrar los datos necesarios (nos referimos, en este caso, tanto a datos de los vehículos como a información referente a la misión desplegada y el entorno en el que se desarrolla) para que el rendimiento del operador sea lo mejor posible, es decir, tenga toda la información necesaria para comandar el equipo de vehículos y tomar las decisiones adecuadas en base a las circunstancias derivadas de la misión.

Además de esto, existe otra cuestión primordial, hallar un equilibrio ente los datos que se muestran y los que no, de forma que el operador mantenga un nivel de confianza razonable en el software del centro de control como para seguir las recomendaciones que se propongan pero que no confíe ciegamente en él. Es decir, cuantos datos se deben mostrar para que el operador tenga la información necesaria para evaluar si debe seguir o no los consejos propuestos por el software, pero sin que llegue a saturarlo provocando una bajada de su rendimiento ya sea debida a una falta de comprensión de la totalidad de la misión, una toma de decisiones tardía que provoque consecuencias adversas o una sensación de que la asistencia software es inservible o innecesaria.

Sobre el aspecto de la Transparencia, Mercado et al. [27] desarrollan un estudio aplicado a un grupo heterogéneo de operadores para determinar su nivel de rendimiento, nivel de carga de trabajo y nivel de confianza en el programa dependiendo del nivel de Transparencia utilizado para varias misiones distintas. En el experimento se proponen a cada operador varias misiones sobre las que tiene que tomar decisiones, variando en cada caso

el nivel de transparencia, que se estructura en 3 niveles que proporcionan cantidades distintas de información. El operador tendrá que decidir si hace caso al consejo propuesto por la aplicación o si cambia la estrategia a seguir para el desarrollo de la misión. En base a los resultados de este experimento se puede extraer la conclusión de que, en contraste con lo expuesto por Helldin et al. [30], 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. [31]).

3.1.2. ADAPTABILIDAD

El software adaptativo puede resultar útil en diferentes contextos y con diferentes fines. A veces puede mejorar el rendimiento del propio software que se modifica para actuar de forma más eficiente basándose en las características de un sistema cambiante (por ejemplo un servidor que acapara más recursos para poder atender a un mayor número de clientes o los libera si tiene menos peticiones). En otros casos como el que nos ocupa puede mejorar el rendimiento del usuario tratando de adaptar un interfaz gráfico para mostrar mayor o menor número de datos en base a las necesidades de una misión, para lograr un mayor nivel de confianza del operador o para resaltar los datos que sean más importantes en detrimento de aquellos prescindibles o menos importantes.

En primer lugar, para desarrollar un sistema adaptativo deberíamos preguntarnos si necesitamos utilizar un lenguaje declarativo de especificación que nos ayude a definir nuestras estructuras de forma clara y robusta, definiendo sus propiedades y cómo interactúan unas con otras. En esta línea, Jeff Magee et al. proponen Darwin [20], un lenguaje declarativo para sistemas de arquitectura distribuida que permite instanciación dinámica y flexibilidad a la hora de reorganizar los sistemas (añadir o quitar componentes) en tiempo de ejecución. Se basa en grafos de componentes que interactúan (proveen o demandan servicios) donde los conectores indican las relaciones existentes entre ellos. También podemos encontrar lenguajes de aplicación similar pero orientados a la administración de permisos y seguridad, como PONDER [21], propuesto por Damianou et al., que permite definir políticas de control (autorización, obligación, restricción, grupos, roles, jerarquías, etc.) de forma dinámica. Es flexible y extensible, y proporciona una forma uniforme de especificar políticas de seguridad o permiso de acceso con un amplio rango de módulos que pueden reutilizarse, aunque podría llegar a ralentizar el sistema en determinadas circunstancias si los requisitos de seguridad son demasiado numerosos.

Una vez establecida la forma de especificación, deberemos decidir una serie de cuestiones de importancia a la hora de diseñar la infraestructura de nuestro sistema adaptativo. Deberemos definir los módulos necesarios: al menos uno que se encargue de desplegar los elementos visuales en el interfaz gráfico de la forma adecuada, otro que monitorice el entorno y/o las variables que determinemos críticas para realizar los cambios en el visualizador, y otro que deberá clasificar la información recibida y almacenarla en bases de conocimiento u ontologías para su posterior aprendizaje. El número y funciones de cada uno de estos módulos pueden ser variables dependiendo de la arquitectura que elijamos. La información planteada por Oreizy et al. [22] nos ayudará a plantearnos una serie de cuestiones para definir nuestra estructura, como bajo qué condiciones debe cambiar nuestro sistema, cómo debemos estructurar nuestro software, el tipo de adaptabilidad que necesitamos (abierta si se pueden recoger datos y estrategias del exterior, cerrada si sólo utilizaremos recursos y datos internos), si el cambio realizado resultará rentable en términos de beneficio/coste, con cuánta frecuencia debemos comprobar si hay que realizar cambios, etc.

Finalmente, a la hora de diseñar la nueva infraestructura del CCV, nos apoyamos también en el trabajo de Cheng, Garlan et al.[23, 24, 25, 26] en los que se define RAINBOW, una infraestructura que provee de los mecanismos necesarios para componer un software adaptativo que cambia a raíz de las necesidades observadas en la monitorización del sistema en lazo cerrado. Para ello han desarrollado su propio lenguaje lo suficientemente expresivo para definir una ontología que permita representar un sistema adaptativo definiendo tres conceptos básicos: Operador (acciones a realizar), Táctica (secuencia corta de acciones para arreglar un problema específico) y Estrategia (árbol de tácticas para resolver determinadas situaciones). Este lenguaje permite definir tácticas, asignarles costes y variar estrategias dependiendo de los datos monitorizados eligiendo la rama por la que se mueve el programa en determinadas circunstancias. Estas definiciones de estrategias de cambio, unidas a una arquitectura modular dotan a RAINBOW de la flexibilidad necesaria para su utilización en gran variedad de contextos.

3.2. ANÁLISIS DE LOS ASPECTOS RELEVANTES AL CCV

El objetivo a lograr a partir de ahora, es dotar al CCV de la infraestructura necesaria para aplicar la Adaptabilidad y la Transparencia sin perder las ventajas otorgadas por el modelo que tenemos hasta ahora. Para alcanzar estos objetivos, se comenzará con una reestructuración de la interfaz de usuario, que a pesar de contar con una notable flexibilidad a la hora de realizar su diseño, aún resulta ligeramente limitada a la hora de afrontar cierto tipo de cambios.

Para ello se prescindirá de los paneles horizontal y vertical y se ampliará la superficie del mapa a la totalidad de la ventana principal. La superficie del mapa se dividirá en una cuadrícula virtual que permitirá albergar los componentes con todas las virtudes de organización con las que se contaba anteriormente pero añadiendo las ventajas de poder variar el tamaño de los mismos en función de su importancia, hacerlos desaparecer y reaparecer, pasarlos de primero a segundo plano y moverlos libremente por la superficie del mapa en base a las necesidades de la misión o las preferencias del operador. Esta estructura de cuadrícula también nos permitirá implementar la adaptabilidad con la flexibilidad necesaria, sin tener que ajustarnos a dos marcos que impedían variar tamaños y usar ciertas posiciones. Los componentes podrán seguir colocándose de forma similar, antes del comienzo de la ejecución del programa, realizando un diseño en el archivo de configuración XML, pero además podrán adaptarse de forma automática en tiempo de ejecución.

También se implementará una nueva característica en el archivo de configuración XML, el nivel de transparencia inicial deseado, que se modificará en base a las preferencias del usuario. Hemos podido comprobar que un usuario con cierta experiencia en el manejo del CCV es capaz de determinar con suficiente exactitud su grado de confianza en el software y cuanta información necesita que despliegue para una correcta toma de decisiones, pudiendo más adelante implementar cambios en el nivel de transparencia en tiempo de ejecución.

El siguiente paso que debemos realizar es la elección de los módulos necesarios para la monitorización, recogida y almacenaje de datos, toma de decisiones y aprendizaje automático que actúe como cerebro del CCV Adaptativo. Una vez dotado el sistema de la infraestructura necesaria, sólo quedará determinar las estrategias de cambio y aprendizaje, completando así un centro de control que podrá responder mucho mejor a las necesidades derivadas de la gestión de un equipo colaborativo de vehículos autónomos de naturaleza heterogénea.

4. CONCLUSIONES

En este artículo se presenta un Centro de Control Versátil, configurable en base a las necesidades de la misión y el operador, capaz de monitorizar simultáneamente un equipo de vehículos heterogéneos en colaboración, tal y como se pone de manifiesto en las pruebas realizadas sobre las misiones de localización y recogida de vertidos del proyecto SALACOM.

Además, el artículo realiza un estudio crítico sobre las limitaciones actuales de dicho sistema, e introduce algunos aspectos relevantes que deberán ser tenidos en cuenta a la hora de desarrollar una nueva versión del CCV para asegurar un mayor grado de adaptabilidad del mismo a las necesidades del usuario, a la evolución de la misión y a la cantidad de información mostrada al operador en cada momento. Todo esto se llevará a cabo en tiempo de ejecución, mediante un sistema de aprendizaje automático y de monitorización de la eficiencia del sistema y de la forma de interactuar con los controles que tiene el usuario durante el desarrollo de cada misión.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Educación, Cultura y Deporte bajo el proyecto DPI2013-46665-C2.

Referencias

[1] (2016), Amazon Prime Air. https://www.amazon.com/b?&node=8037720011.

[2] (2016), General Atomics UAV's. http://www.ga-asi.com/aircraft-platforms.

[3] (2016), UAV Atlante. http://militaryaircraft-airbusds.com/Aircraft/UAV/Atlante.aspx.

[4] (2016), US Army UAV operator. http://www.goarmy.com/careers-and-jobs/browse-career-and-job-categories/transportation-and-aviation/unmanned-aerial-vehicle-operator.html.

[5] (2016), Parrot. https://store.parrot.com/es/.

[6] (2016), DJI. http://store.dji.com/es.

[7] (2016), QGroundControl. http://qgroundcontrol.org/.

[8] (2016), ArduPilot. http://ardupilot.com/ardupilot/index.html.

[9] (2016), UAS SkyView. http://www.uas-europe.se/index.php/products/skyview-ground-control-station-software.

[10] (2016), Freeflight 3 para skycontroller de parrot. https://play.google.com/store/apps/details?id=com.parrot.freeflight3&hl=es.

[11] (2016), General Atomics Ground Control Stations. http://www.ga-asi.com/ground-control-stations-gcs.

[12] (2016), UAV Factory Ground Control Stations. http://www.uavfactory.com/product/16.

[13] (2016), UAS Ground Control Stations. http://www.uas-europe.se/index.php/products/portable-gcs-computer.

[14] Bonache Seco, J. A. (2013), Trabajo Fin de Máster: Diseño de centro de control versatil para la monitorizacion y control de vehiculos autonomos. http://eprints.sim.ucm.es/26469/.

[15] Amditis, A., KuBmann, H., Polychronopoulos, A., Engstrom, J., and Andreone, L. (2006) System architecture for integrated adaptiveUMI solutions. *Intelligent Vehicles Symposium, 2006 IEEE* [16], pp. 388 – 393.

[16] Amditis, A., et al. (2010) Towards the automotive HMI of the future: Overview of the aide-integrated project results. *Intelligent Transportation Systems, IEEE Transactions on* [15], pp. 567 – 578.

[17] Polychronopoulos, A., Amditis, A., and Bekiaris, E. (2004) Information data flow in awake multi-sensor driver monitoring system. *Intelligent Vehicles Symposium, 2004 IEEE* [15], pp. 902 – 906.

[18] Lee, A. N. and Lastra, J. L. M. (2013) Enhancement of industrial monitoring systems by utilizing context awareness. *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2013 IEEE International Multi-Disciplinary Conference on* [19], pp. 277 – 284.

[19] Lee, A. N. N., Evchina, Y., Dvoryanchikova, A., and Lastra, J. L. M. (2013) Pro-active content managing system for efficient human machine interaction in data intensive environments. *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on* [18], pp. 790 – 796.

[20] Magee, J., Dulay, N., Eisenbach, S., and Kramer, J. (1995) Specifying distributed software architectures. *Software Engineering - ESEC 95 Lecture Notes in Computer Science*, **989**, 137–153.

[21] Damianou, N., Dulay, N., Lupu, E., and Sloman, M. (2001) The ponder policy specification language. *Policies for Distributed Systems and Networks series Lecture Notes in Computer Science*, **1995**, 18–38.

[22] Oreizy, P., Gorlick, M. M., Taylor, R. N., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D. S., and Wolf, A. L. (1999) An architecture-based approach to self-adaptive software.

[23] Garlan, D., Cheng, S.-W., Huang, A.-C., Schmerl, B., and Steenkiste, P. (2004) Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer (IEEE Computer Society)*, **7 (Issue: 10)**, 48–54.

[24] Cheng, S.-W., Huang, A.-C., Garlan, D., Schmerl, B., and Steenkiste, P. (2004) Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Proceedings of the International Conference on Autonomic Computing (ICAC 04)*, pp. 276 – 277.

[25] Cheng, S.-W., Garlan, D., and Schmerl, B. (2006) 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*, pp. 2–8.

[26] Cheng, S.-W. and Garlan, D. (2007) Handling uncertainty in autonomic systems. *Automated Software Engineering - ASE 07*.

[27] Mercado, J. E., Rupp, M. A., Chen, J. Y. C., Barnes, M. J., Barber, D., and Procci, K. (2016) Intelligent agent transparency in human-agent teaming for multi-UxV management. *Human Factors*, **58**, 401–415.

[28] (2016), Beckhoff TwinCAT. https://www.beckhoff.com/english.asp?twincat/default.htm.

[29] (2016), MAVlink. http://qgroundcontrol.org/mavlink/start.

[30] Helldin, T. (1999) Transparency for future semi-automated systems(doctoral dissertation).

[31] Chen, J., Procci, K., Boyce, M., Wright, J., Garcia, A., and Barnes, M. (2014) Situation awareness-based agent transparency(no. arl-tr-6905).