Propuesta de arquitectura distribuida de control inteligente basada en políticas de calidad de servicio

José Luis Poza Luján

EDITORIAL
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA

Tesis Doctoral

Propuesta de arquitectura distribuida de control inteligente basada en políticas de calidad de servicio

Febrero 2012

Autor: D. Jose Luis Poza Luján

Directores: Dr. D. José E. Simó Ten

Dr. D. Juan Luis Posadas Yagüe

Esta editorial es miembro de la UNE, lo que garantiza la difusión y comercialización de sus publicaciones a nivel nacional e internacional.

Primera edición, 2012

© de la presente edición:
Editorial Universitat Politècnica de València
www.editorial.upv.es

ISBN: 978-84-8363-859-0 (versión impresa)

Propuesta de arquitectura distribuida de control inteligente basada en políticas de calidad de servicio

Autor

D. José Luís Poza Luján

Profesor Colaborador

Departamento de Informática de Sistemas y Computadores

Instituto de Investigación de Automática e Informatización Industrial

Universidad Politécnica de Valencia

Directores

Dr. D. José E. Simó Ten

Profesor Titular de Universidad

Departamento de Informática de Sistemas y Computadores

Instituto de Investigación de Automática e Informatización Industrial

Universidad Politécnica de Valencia

Dr. D. Juan Luis Posadas Yagüe

Profesor Titular de Universidad

Departamento de Informática de Sistemas y Computadores

Instituto de Investigación de Automática e Informatización Industrial

Universidad Politécnica de Valencia

Febrero de 2012

Deseo que mis primeras palabras de agradecimiento sean para todas aquellas personas que han contribuido de alguna forma a la realización de esta tesis doctoral y no son mencionadas a continuación, cada vez que reviso estos párrafos añado a más personas por lo que he tenido que detenerme en algún punto. Si estás leyendo estas líneas te agradezco el apoyo que me has dado para la realización de la tesis.

Por supuesto, las primeras palabras de agradecimiento deben ir dirigidas a mis directores José Simó y Juan Luis Posadas por su trabajo, tiempo, dedicación y amistad empleados en la realización de esta tesis doctoral.

Deseo mencionar a José Miguel Corberán y a Rafael Royo por mostrarme la pasión por la investigación en el periodo que, como becario estudiante, compartí con ellos en el departamento de Termodinámica Aplicada. Asimismo a Ana Pont y Lenin Lemus que me mostraron, también como estudiante becario, las bondades y las dificultades de la docencia Universitaria. No debo olvidar a todos los integrantes del proyecto GAMA por demostrarme que la investigación en equipo puede ser además un placer especialmente a Raúl Simarro y a Pedro Jorquera por enseñarme a superar los malos momentos y a disfrutar de los buenos.

Especial mención a Juan Carlos Cano y Félix Buendía, compañeros de Departamento y de deporte, a los que no sé de qué les hablaré a partir de ahora y que han estado dándome ánimos vuelta a vuelta cada vez que aparecía el tema de la tesis.

A todos mis compañeros del Departamento de Informática de Sistemas y Computadores, que ya no tendrán que preguntarme lo mismo por los pasillos, especialmente a mi compañero de despacho Pascual Pérez por aguantarme todos estos años. A Alfons Crespo, Pedro López, Salvador Petit, Vicent Lorente, Sergio Sáez, Juan Carlos Pérez, Patricia Balbastre, Silvia Terrasa, Paco Blanes, Ginés Benet, Pedro Cruz, Joaquín Ruiz, Julio Sahuquillo, Carlos Calafate, José Benlloch, Floreal Acebrón, Marina Alonso, Toni Martí, Paco Rodríguez, Vicente Atienza y el resto de integrantes que han estado constantemente preguntando el estado de la tesis y dándome ánimos para finalizarla.

A mis compañeros de Universidad: Carlos Carrascosa, Vicente Julián, Vicente Botti, Miguel Rebollo, Martí Navarro, Ángeles Calduch, Esther Sanabria. Y, como no, a mis compañeros de andanzas en el Claustro de la Universidad: Alberto Conejero, José Gonzálvez, Gemma Peña, María De Miguel y Fernando López.

Es necesario nombrar a todas las personas que a lo largo de estos años han estado constantemente mostrándome el interés y dándome los ánimos sin los cuales hubiese sido muy difícil concluir este trabajo: Manolo, Bernardo, Ricardo, Ángela, Javi, Amparo, Paco, Raquel, Marisa, Sito, Mila, Toni, Mila, Pepe, Noelia, Jesús, Víctor, Lorena, Ángeles, Ernesto, Antía, Víctor, Beatriz, Paco, María, Javi, Laura, Juan, María, Gema, Carlos, Alicia, Antonio, Fina, Javier, Jose y M. Carmen.

A mi Madre y mis hermanas Ana y Carmen.

A Elisa

Y, por supuesto, a mi Padre.

Resumen

Esta tesis se enmarca en el estudio de las arquitecturas inteligentes de control distribuido y de los sistemas de comunicaciones empleados. Más concretamente el trabajo se centra en la optimización del sistema de control por medio de la evaluación del rendimiento en el middleware a través de los parámetros de calidad de servicio y de la optimización del control empleando políticas de calidad de servicio.

El principal objetivo de este trabajo ha sido estudiar, diseñar, desarrollar y evaluar una arquitectura de control distribuido, basándose en el estándar de comunicaciones *Data-Distribution Service for Real-Time Systems* (DDS) propuesto por la organización *Object Management Group* (OMG).

A medida que aumentan los requisitos de los sistemas inteligentes de control distribuido, aumentan los requisitos de las comunicaciones. El papel de los middleware que dan soporte a los sistemas de control distribuido ha ido evolucionando incrementándose las características de los mismos. Entre las características más destacables que el middleware requiere se encuentran el soporte a la gestión temporal y el control del flujo de los mensajes.

Para poder ofrecer estas características, es necesario que las arquitecturas dispongan de fórmulas que permitan la monitorización y la valoración de dichas características. Estas fórmulas dan lugar a los parámetros de calidad de servicio (QoS). Para gestionar los parámetros la arquitectura debe proporcionar los mecanismos que permitan configurarlos en función de los requisitos. Estos mecanismos se conocen como políticas de QoS. Además de poder evaluar el servicio ofrecido, las arquitecturas deben poder evaluar el cumplimiento interno de los objetivos de control, con este objetivo aparece el concepto de parámetros de calidad de control (QoC).

La variedad de sistemas donde se aplican las arquitecturas inteligentes de control distribuido es muy amplia. De los múltiples entornos, destacan los sistemas inalámbricos de redes de sensores (WSN) por la cantidad de información que debe transmitirse y los sistemas de control en red (NCS) por las restricciones en el procesamiento. Ambos tipos de sistemas confluyen en las redes inalámbricas de sensores y actuadores (WSAN) como paradigma de sistema que debe dar soporte a una gran cantidad de información con estrictos requisitos de procesamiento. Las tendencias actuales en arquitecturas distribuidas de control están orientadas a sistemas WSAN.

Teniendo en cuenta los requisitos anteriores, como tema central de la tesis se propone el diseño de una arquitectura distribuida de control inteligente que de soporte a la QoS, tanto en la medición por medio de los parámetros, como en la gestión por medio de las políticas de QoS. Las políticas deben permitir la variación de las características de la comunicación en función de los requisitos de control, expresados estos últimos por medio de los parámetros de QoC. A la arquitectura desarrollada se le ha llamado FSACtrl.

Para determinar los requisitos de la arquitectura FSACtrl, se han estudiado las revisiones de los autores más relevantes acerca de las características más destacadas de las arquitecturas distribuidas de sistemas de control. A partir de estas características se han diseñado los elementos de la arquitectura FSACtrl. Los elementos que dan soporte a las comunicaciones se han basado en los del estándar DDS de la OMG, mientras que los elementos de control se han basado en el estándar *Sensor Web Enablement* (SWE) del *Open Geospatial Consortium* (OGC).

El modelo DDS está basado en el paradigma de comunicaciones de Publicación y Subscripción. En esta tesis, se propone ampliar el modelo *Data Centric Publish Subscribe* (DCPS), parte del estándar DDS, con una serie de elementos que permitan la gestión de los eventos que puedan producirse a lo largo del funcionamiento del sistema. El modelo SWE está basado en la estandarización del control por medio de elementos básicos que se encadenan para realizar un procesamiento más complejo. En esta tesis se propone ampliar el modelo de control con la inclusión de elementos de comunicaciones del modelo DCPS y el soporte a la QoC.

Para la validación de la arquitectura se ha implementado un entorno de diseño y simulación del control. El entorno lo forma una aplicación de diseño visual de los elementos de la arquitectura que permite la ejecución de los algoritmos de control y un simulador de robots móviles que permite incluir diversos tipos de sensores, fuentes de datos y obstáculos en el medio en el que el vehículo desarrolla la navegación.

Como ejemplo de uso de los elementos de la arquitectura FSACtrl se han implementado los cinco primeros vehículos de Braitenberg. Los tres primeros vehículos se caracterizan por carecer de funciones de control propiamente dichas, por lo que son adecuados para poder evaluar el funcionamiento de la arquitectura como middleware. El cuarto vehículo incluye funciones de control básicas, por lo que resulta conveniente para comprobar cómo embeber parte del control en la capa de comunicaciones. El quinto vehículo incluye el procesamiento lógico y es el que se emplea para comprobar cómo la gestión dinámica de las comunicaciones basada en el control permite optimizar el funcionamiento del sistema.

Los resultados del trabajo experimental desarrollado son satisfactorios. Se demuestra la viabilidad de la arquitectura como middleware con soporte al control, y principalmente se comprueba cómo la gestión en las comunicaciones de parte de las tareas de control permite optimizar el funcionamiento del sistema.

Las aportaciones principales de la tesis son diversas. Se ha desarrollado una importante labor de estudio de caracterización de las arquitecturas distribuidas de control. Asimismo se han estudiado los parámetros de las comunicaciones que dan soporte a las arquitecturas de control. Se ha diseñado una arquitectura que, basándose en estándares, permite alcanzar las características del sistema, evaluar mediante parámetros el rendimiento del funcionamiento del sistema y, por medio de las políticas de QoS permite optimizar el funcionamiento en función de las características del control.

Finalmente, destacar que la tesis se enmarca dentro de los proyectos de investigación KERTROL y SIDIRELI en los que participa el grupo de investigación de Informática Industrial del Instituto de Automática e Informática Industrial y en el que desarrolla su labor de investigación el autor. El trabajo desarrollado ha sido publicado en diversos congresos del área, tanto nacionales como internacionales.

Resum

Aquesta tesi s'emmarca en l'estudi de les arquitectures intel·ligents de control distribuït i dels sistemes de comunicacions emprats. Més concretament el treball se centra en l'optimització del sistema de control per mitjà de l'avaluació del rendiment en el middleware a través dels paràmetres de qualitat de servei i de l'optimització del control emprant polítiques de qualitat de servei.

El principal objectiu d'aquest treball ha sigut estudiar, dissenyar, desenvolupar i avaluar una arquitectura de control distribuït, basant-se en l'estàndard de comunicacions *Data Distribution Service for Real-Time Systems* (DDS) proposat per l'organització *Object Management Group* (OMG).

A mesura que augmenten els requisits dels sistemes intel·ligents de control distribuït, augmenten els requeriments de les comunicacions. El paper dels middleware que donen suport als sistemes de control distribuït ha anat evolucionant incrementant les seues característiques. Entre les característiques més destacables que el middleware requereix es troben el suport a la gestió temporal i el control del flux dels missatges.

Per a poder oferir aquestes característiques, és necessari que les arquitectures disposen de fórmules que permeten el monitoratge i la valoració d'aquestes característiques. Aquestes fórmules donen lloc als paràmetres de qualitat de servei (QoS). Per a gestionar els paràmetres l'arquitectura ha de proporcionar els mecanismes que permeten configurar-los en funció dels requisits. Aquests mecanismes es coneixen com a polítiques de QoS. A més de poder avaluar el servei oferit, les arquitectures han de poder avaluar el compliment intern dels objectius de control, amb aquest objectiu apareix el concepte de paràmetres de qualitat de control (QoC).

La varietat de sistemes on s'apliquen les arquitectures intel·ligents de control distribuït és molt àmplia. Dels múltiples entorns, destaquen els sistemes sense fils de xarxes de sensors (WSN) per la quantitat d'informació que ha de transmetre's i els sistemes de control en xarxa (NCS) per les restriccions en el processament. Tots dos tipus de sistemes conflueixen en les xarxes sense fils de sensors i actuadors (WSAN) com a paradigma de sistema que ha de donar suport a una gran quantitat d'informació amb estrictes requisits de processament. Les tendències actuals en arquitectures distribuïdes de control estan orientades a sistemes WSAN.

Tenint en compte els requeriments anteriors, com a tema central de la tesi es proposa el disseny d'una arquitectura distribuïda de control intel·ligent que de suport a la QoS, tant en el mesurament per mitjà dels paràmetres, com en la gestió per mitjà de les polítiques de QoS. Les polítiques han de permetre la variació de les característiques de la comunicació en funció dels requisits de control, expressats aquests últims per mitjà dels paràmetres de QoC. A l'arquitectura desenvolupada se li ha cridat FSACtrl.

Per a determinar els requisits de l'arquitectura FSACtrl, s'han estudiat les revisions dels autors més rellevants sobre les característiques més destacades de les arquitectures distribuïdes de sistemes de control. A partir d'aquestes característiques s'han dissenyat els elements de l'arquitectura FSACtrl. Els elements que donen suport a les comunicacions s'han basat en els de l'estàndard DDS de la OMG, mentre que els elements de control s'han basat en l'estàndard *Sensor Web Enablement* (SWE) del *Open Geospatial Consortium* (OGC).

El model DDS està basat en el paradigma de comunicacions de Publicació i Subscripció. En aquesta tesi, es proposa ampliar el model *Data Centric Publish Subscribe* (DCPS), part de l'estàndard DDS, amb una sèrie d'elements que permeten la gestió dels esdeveniments que puguen produir-se al llarg del funcionament del sistema. El model SWE està basat en l'estandardització del control per mitjà d'elements bàsics que s'encadenen per a realitzar un processament més complex. En aquesta tesi es proposa ampliar el model de control amb la inclusió d'elements de comunicacions del model DCPS i el suport a la QoC.

Per a la validació de l'arquitectura s'ha implementat un entorn de disseny i simulació del control. L'entorn ho forma una aplicació de disseny visual dels elements de l'arquitectura que permet l'execució dels algorismes de control i un simulador de robots mòbils que permet incloure diversos tipus de sensors, fonts de dades i obstacles en el mitjà en el qual el vehicle desenvolupa la navegació.

Com a exemple d'ús dels elements de l'arquitectura FSACtrl s'han implementat els cinc primers vehicles de Braitenberg. Els tres primers vehicles es caracteritzen per mancar de funcions de control pròpiament aquestes, per la qual cosa són adequats per a poder avaluar el funcionament de l'arquitectura com middleware. El quart vehicle inclou funcions de control bàsiques, per la qual cosa resulta convenient per a comprovar com embeure part del control en la capa de comunicacions. El cinquè vehicle inclou el processament lògic i és el que s'empra per a comprovar com la gestió dinàmica de les comunicacions basada en el control permet optimitzar el funcionament del sistema.

Els resultats del treball experimental desenvolupat són satisfactoris. Es demostra la viabilitat de l'arquitectura com middleware amb suport al control, i principalment es comprova com la gestió en les comunicacions de part de les tasques de control permet optimitzar el funcionament del sistema.

Les aportacions principals de la tesi són diverses. S'ha desenvolupat una important labor d'estudi de caracterització de les arquitectures distribuïdes de control. Així mateix s'han estudiat els paràmetres de les comunicacions que donen suport a les arquitectures de control. S'ha dissenyat una arquitectura que, basant-se en estàndards, permet aconseguir les característiques del sistema, avaluar mitjançant paràmetres el rendiment del funcionament del sistema i, per mitjà de les polítiques de QoS permet optimitzar el funcionament en funció de les característiques del control.

Finalment, destacar que la tesi s'emmarca dins dels projectes d'investigació KERTROL i SIDIRELI en els quals participa el grup d'investigació d'Informàtica Industrial de l'Institut d'Automàtica i Informàtica Industrial (ai2) i en el qual desenvolupa la seua labor d'investigació l'autor. El treball desenvolupat ha sigut publicat en diversos congressos de l'àrea, tant nacionals com a internacionals.

Abstract

This thesis is part of a study of intelligent architecture for distributed control and communication systems. The study focuses on optimising control systems by evaluating the performance of middleware through quality of service (QoS) parameters and the optimisation of control using QoS policies.

The main aim of this work is to study, design, develop, and evaluate a distributed control architecture based on the Data-Distribution Service for Real-Time Systems (DDS) communication standard as proposed by the Object Management Group (OMG).

Communication requirements are increasing in line with requirements for intelligent distributed control systems. The role of middleware in supporting distributed control systems has also evolved in terms of characteristics. Among the characteristics now required by middleware are time management support and message flow control.

To offer these features, architectures must have formulas that enable monitoring and assessment of these characteristics. These formulas also give rise to the parameters of quality of service (QoS). To manage these parameters, the architecture must provide mechanisms so that they can be configured according to requirements. These mechanisms are known as QoS policies. In addition to evaluating the service provided, the architecture should be able to assess internal compliance with the control objectives, and this aim introduces the concept of control of quality (QoC) parameters.

There is a wide variety of systems that use intelligent distributed control architectures. Of these many environments, we can highlight wireless sensor networks (WSN) for the amount of information that can be transmitted; and networked control systems (NCS) for the processing restrictions. Both types of systems converge in wireless sensor and actuator networks (WSAN) as a paradigm of a system that supports a large amount of information with strict processing requirements. Current trends in distributed architectures are orientated towards WSAN systems.

Considering the above requirements, the principal theme of the thesis is a design for a distributed architecture for intelligent control that supports QoS through the measurement of parameters and through QoS management policies. These policies must enable a variation in the characteristics of communication in terms of control requirements, as expressed through the QoS parameters. The developed architecture has been called FSACtrl.

To determine the requirements of FSACtrl architecture, we have studied the reviews of key authors about the most important features of distributed architectures for system control. We have designed the FSACtrl architectural elements with these features in mind. The elements that support communications are based on the OMG DDS standard, while the control elements are based on the Sensor Web Enablement (SWE) standard produced by the Open Geospatial Consortium (OGC).

The DDS model is based on the publication and subscription communication paradigm. In this thesis, an extension of a series of elements that enable the management of events during system operation is proposed for that part of the DDS standard known as Data Centric Model Publish Subscribe (DCPS). The SWE model is based on the standardisation of control through basic elements that are linked to perform more complex processing. This thesis proposes an extension to the control model with the inclusion of DCPS model communication components and support offered for QoC.

A design and control simulation environment has been implemented for validation of the architecture. The environment consists of a visual design application of the architectural elements that enable the execution of control algorithms; as well as a mobile robot simulator that can include various types of sensors, data sources, and obstacles in the environment in which the vehicle navigates.

Five Braitenberg vehicles have been implemented as an example of the use of FSACtrl architectural elements. The first three vehicles lack proper control functions, and so they are suitable for evaluating the performance of the architecture as middleware. The fourth vehicle includes basic control functions and is used to test the embedding of the control in the communications layer. The fifth vehicle includes logical processing and is used to test how control-based dynamic communication management enables the optimisation of system performance.

The results of experimental work developed are satisfactory. We have demonstrated the feasibility of the architecture as middleware for support control, and it has been demonstrated how system performance is optimised by the communications management of part of the control tasks.

There are many and varied principal contributions in this thesis. We have made a major study of distributed control architectures. We have also studied the communication parameters that support control architectures. We have designed an architecture based on standards that can reach system characteristics, evaluate system performance using parameters, and use QoS policies to optimize performance relative to the control characteristics.

Finally, it is noteworthy that the thesis has been developed as part of the KERTROL and SIDIRELI research projects in which the author works as a member of the industrial computing research group at the Institute of Industrial Automation and Computer Engineering. The research work has been published in various conferences in the area – both nationally and internationally.

Índice de contenidos

Índice de figuras

Índice de tablas

"Al ritmo al que aumenta la capacidad de los ordenadores, doblándose cada año, aproximadamente, dentro de dos o tres décadas contaremos con suficiente potencia como para realizar el trabajo de un gran sistema nervioso. Ahora bien, disponer de esa potencia es tan solo la mitad del problema. La otra mitad es organizar correctamente los mecanismos internos del soporte lógico."

[Moravec, 2001]

1 Introducción

1.1 Motivación

Uno de los campos de investigación de actualidad es el control de los sistemas distribuidos basados en las redes de computadores o NCS (*Networked Control Systems*). El control de los sistemas distribuidos ha ido evolucionando de los sistemas iniciales basados en buses de comunicaciones orientados a las distancias cortas entre componentes, a los sistemas industriales amplios basados en redes de ordenadores. A medida que aumenta la complejidad del sistema, aumenta la complejidad de las comunicaciones y se hace necesario el empleo de sistemas de intermediación, o middleware, entre los componentes.

La tendencia actual aúna todos los aspectos de los sistemas distribuidos de control inteligente en el concepto de sistemas ciber-físicos [Lee, 2008]. Los sistemas ciber-físicos cubren una amplia variedad de sistemas distribuidos de control, entre los que se incluyen las redes de sensores inalámbricos o WSN (*Wireless Sensor Networks*) y los sistemas de redes de sensores y actuadores inalámbricas o WSAN (*Wireless Sensor/Actuator Networks*) [Xia, 2008].

La conveniencia de emplear sistemas distribuidos de control por computador se basa en la capacidad de computación con algoritmos complejos que proporcionan los actuales ordenadores y la variedad de posibilidades de distribución de la información que proporcionan las actuales redes de computadores. Escoger la infraestructura correcta de comunicaciones, así como los aspectos de configuración de los elementos de control, son dos de las acciones necesarias para lograr un rendimiento satisfactorio del sistema distribuido [Hristu-Varsakelis and Levine, 2005].

Debido a la naturaleza asíncrona de las redes de comunicaciones que no ofrecen un soporte a tiempo real estricto, se hace necesaria una coordinación muy estrecha entre los componentes de control y los componentes de comunicaciones, lo que implica que exista una relación directa entre el rendimiento de las operaciones de control y el rendimiento de las comunicaciones [Yang, 2006]. Es importante conocer y controlar esta relación entre control y comunicaciones para poder garantizar unas mínimas prestaciones [Antsaklis and Baillieul, 2007]. Esta coordinación abarca cuestiones que complementan a las de soporte temporal y que se basan en la gestión de los eventos que se suceden en el sistema.

Por medio de los sistemas de control basados en eventos se obtiene un mejor balance entre el comportamiento del control y otros aspectos del sistema, como la carga del procesador y los retardos del sistema de comunicaciones [Dormido et al., 2008]. Por ello, se requieren arquitecturas que permitan implementar sistemas de control inteligente basados en eventos, aportando un valor añadido al enfoque clásico de control basado en la planificación de tareas.

Para optimizar un sistema distribuido de control basado en eventos se requiere gestionar los recursos del sistema. Debido a que la gestión de los recursos de comunicaciones influye directamente en la distribución y en la cooperación de los algoritmos de control, esta tesis propone emplear los parámetros de configuración y rendimiento de las comunicaciones y del control. La elección correcta de los parámetros, así como la formulación de los parámetros derivados y la aplicación de éstos en el sistema es una de las claves para lograr una solución optimizada.

Estos parámetros son conocidos comúnmente como parámetros de Calidad de Servicio [Kumar, 2001] en el caso de las comunicaciones y de Calidad de Control [Dorf and Bishop, 2008] en el caso de los algoritmos de control.

Las funciones que permiten gestionar un sistema por medio de los parámetros de calidad de servicio se conocen como políticas de calidad de servicio. Usar políticas de calidad de servicio en sistemas distribuidos de control inteligente involucra una variedad muy amplia de características y operaciones a implementar en el sistema, como por ejemplo la negociación de los parámetros entre los componentes, la compatibilidad de los parámetros o la reacción del sistema frente a los incumplimientos de los valores de la calidad de servicio negociados.

1.2 Hipótesis

Basándose en los aspectos anteriores, en esta tesis se determina la siguiente hipótesis: el desempeño de un sistema de control distribuido inteligente basado en un sistema distribuido de computadores, puede evaluarse, tanto cuantitativa como cualitativamente, y gestionarse a partir de los parámetros de calidad de servicio y de calidad de control de los componentes de dicho sistema (Figura 1).

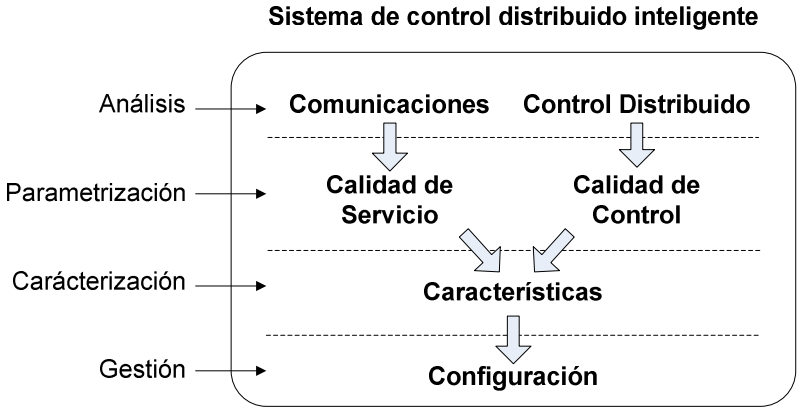


Figura 1. Hipótesis principal en la que está basada la tesis.

1.3 Objetivos

Para comprobar la validez de dicha hipótesis y como objetivo general de esta tesis doctoral se propone el diseño de una arquitectura de control distribuido inteligente basado en un sistema distribuido de computadores que proporcione los parámetros de calidad de servicio, calidad de control y las funciones necesarias para evaluar la viabilidad en el desempeño del sistema y así poder optimizarlo. Para ello, los objetivos concretos planteados son:

- **Revisión de las arquitecturas de control distribuido inteligente, para extraer las características más relevantes de las mismas**. Para cubrir el rango más amplio de sistemas, la revisión debe centrarse en diversos tipos de arquitecturas. Desde las arquitecturas de los sistemas de automatización del hogar, comúnmente conocidos como domóticos, hasta las arquitecturas más complejas de control de la navegación de robots. Además, se deben analizar los aspectos de comunicaciones asociados a los sistemas distribuidos de control inteligente, ya que la elección del paradigma de comunicación más adecuado incide directamente en la eficiencia en el control. Los resultados de la revisión se

muestran en los apartados “2.2. Arquitecturas de control” y “2.3. Sistemas de comunicaciones” del segundo capítulo. En el caso de las arquitecturas de control se encuentran todos los detalles en [Poza et al., 2009c]. En el caso de los paradigmas y sistemas de comunicaciones los detalles de los sistemas revisados se encuentran en [Poza, 2009a].

- **Análisis de los parámetros de rendimiento de los sistemas y establecimiento de la relación de los mismos con la calidad de servicio para ser empleados en la optimización del rendimiento de los sistemas**. El análisis se realiza a partir de los parámetros cuantitativos de los elementos que constituyen el sistema, para ir extendiendo el estudio hacia el ámbito de los parámetros cualitativos. Es importante relacionar las características de los sistemas, expresadas en términos cualitativos, con los parámetros cuantitativos, para obtener una base sólida de formulación del sistema que permitirá optimizarlo. Los resultados del análisis se muestran en el apartado “2.4. El papel de la calidad de servicio” en el segundo capítulo, mientras que los detalles del análisis se han publicado en [Poza, 2009b].

- **Diseño de una arquitectura de control distribuido basada en estándares que de soporte a la calidad de servicio y la calidad de control por medio del acceso a los parámetros y permita la gestión del sistema a través de políticas de calidad de servicio**. Teniendo en cuenta la complejidad implícita de los sistemas distribuidos de control inteligente, es recomendable el uso de estándares en su diseño. Los estándares de control y de comunicaciones proporcionan una plataforma común que simplifica problemas habituales como la escalabilidad del sistema o la compatibilidad entre elementos heterogéneos del mismo. El diseño del control debe permitir la gestión de los mensajes incluso en ausencia de un soporte de tiempo real estricto, característica implícita a los sistemas basados en eventos. El diseño de las comunicaciones debe basarse en el paradigma más adecuado que proporcione una plataforma confiable a los eventos de control. La arquitectura debe proporcionar una gestión dinámica de los elementos basándose en los eventos producidos durante el funcionamiento del sistema. La descripción de los estándares seleccionados se encuentra en [Poza, 2009c] y [Poza, 2009d].

- **Formulación de los parámetros que permitan modelar y comparar las características del sistema basándose en parámetros cuantitativos, para medir, evaluar y optimizar un sistema de control inteligente distribuido**. Mediante el análisis de los parámetros empleados por los componentes de control de la arquitectura propuesta se formularán los índices de optimización que permitirán alcanzar las características deseables de un sistema distribuido de control inteligente.

- **Validación de las ventajas que proporciona la arquitectura diseñada en comparación con las arquitecturas empleadas en los sistemas de ámbito similar de aplicación**. La comparación debe realizarse cualitativamente en los campos donde aplicar la arquitectura propuesta, en concreto los sistemas de redes de sensores, los sistemas de control en red y aquellos sistemas que empleen los mismos estándares que los empleados en la tesis. Por medio de esta comparación cualitativa se contextualizará la tesis en los campos anteriormente citados.

- **Validación de la arquitectura por medio de la implementación de un entorno de simulación.** La validación se realizará por medio de las correspondientes pruebas en un entorno adecuado de simulación y de control. El entorno de simulación debe proveer los datos al entorno de control para comprobar la validez y utilidad de las fórmulas de obtención de los parámetros de QoS. Además se debe disponer del entorno que proporcione todos los elementos de la arquitectura que permitan diseñar el sistema de control.

1.4 Organización del documento

En el presente capítulo se ha establecido el entorno en el que se enmarca esta tesis. Se ha determinado la hipótesis a desarrollar así como los objetivos a alcanzar para comprobar la validez de la hipótesis. El resto de capítulos del documento presentan los siguientes contenidos:

En el segundo capítulo, se analiza la evolución y situación del entorno actual de los sistemas de control (Figura 2). Para ello, primero se exponen las conclusiones más relevantes del análisis de las principales arquitecturas de control de los sistemas domóticos y de las arquitecturas de control más complejas de los sistemas de navegación de robots. Se realiza un análisis de los sistemas de comunicaciones que conectan los componentes control.

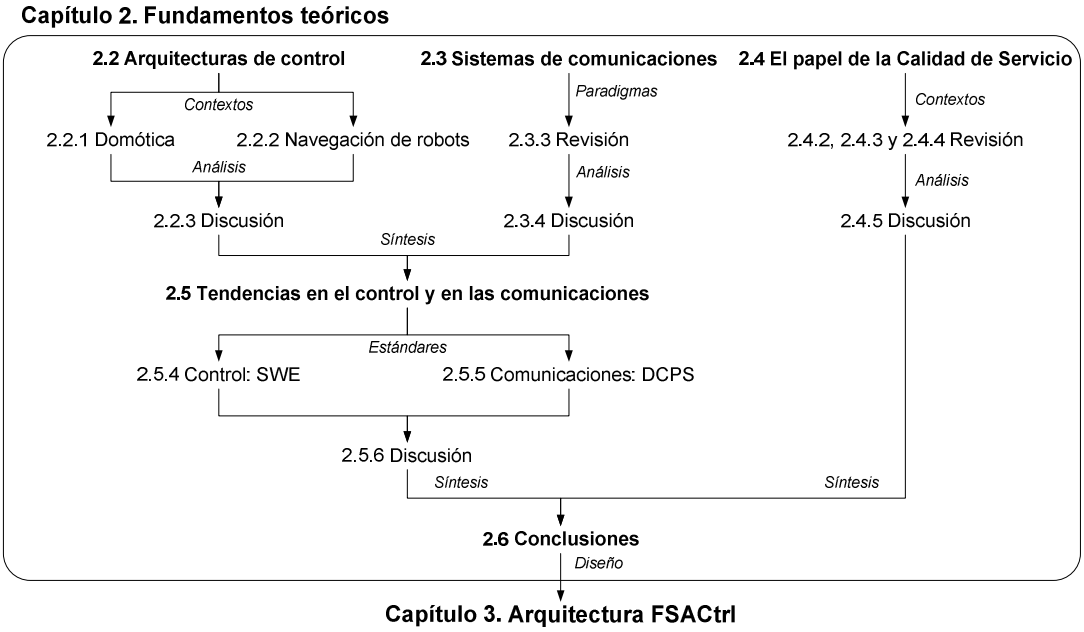


Figura 2. Diagrama conceptual del capítulo 2.

A continuación se analiza el papel de la calidad de servicio en los sistemas de comunicaciones. Se comienza por los parámetros cuantitativos para ir aumentando la complejidad hasta los parámetros cualitativos. Se evalúa el uso de los parámetros de calidad de servicio para servir de base para la obtención de parámetros más complejos. Además, se estudia el uso de los parámetros de calidad de servicio para optimizar un sistema de control a partir de las políticas de calidad de servicio. Finalmente, se realiza una discusión acerca de las tendencias en los sistemas de control y los sistemas de comunicaciones, y de esta forma, determinar cuáles son los estándares más adecuados para el diseño de la arquitectura objetivo de la tesis.

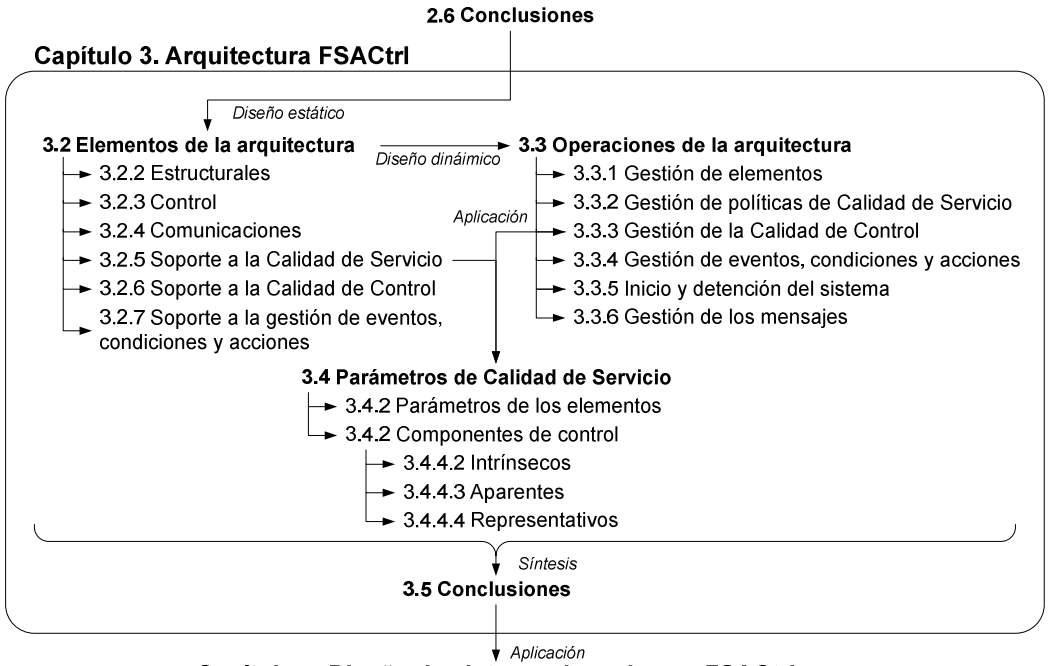


Figura 3. Diagrama conceptual del capítulo 3.

En el tercer capítulo se presenta la propuesta de arquitectura que proporciona el soporte al uso de los parámetros de calidad de servicio y de calidad de control, para establecer las acciones adecuadas para optimizar el sistema de control distribuido (Figura 3). Inicialmente se describe la arquitectura por medio de un modelo conceptual para presentar los elementos que la forman y cómo interactúan entre ellos. A continuación se presenta la especificación formal por medio de diagramas UML (*Unified Modelling Language*) que proporcionan la visión estática (elementos) y dinámica (relación entre elementos) de la arquitectura propuesta. Finalmente, se expone la propuesta de las fórmulas que permiten asignara parámetros a las características de un sistema distribuido de control por medio de las políticas de calidad de servicio.

El cuarto capítulo expone el diseño de sistemas basados en la arquitectura (Figura 4). Para ello se muestran las operaciones necesarias para la inserción, modificación y eliminación de los elementos, así como las relaciones entre ellos. Asimismo, se muestra la asignación de políticas de calidad de servicio y de parámetros de calidad de control a los elementos. Seguidamente, se muestra la gestión de los eventos del sistema y de las acciones asociadas a los eventos y las consecuencias en el sistema. Finalmente, se muestra cómo asociar los elementos de la arquitectura con un sistema multi-agente distribuido, como ejemplo de aplicación práctica de los elementos de la arquitectura desarrollada.

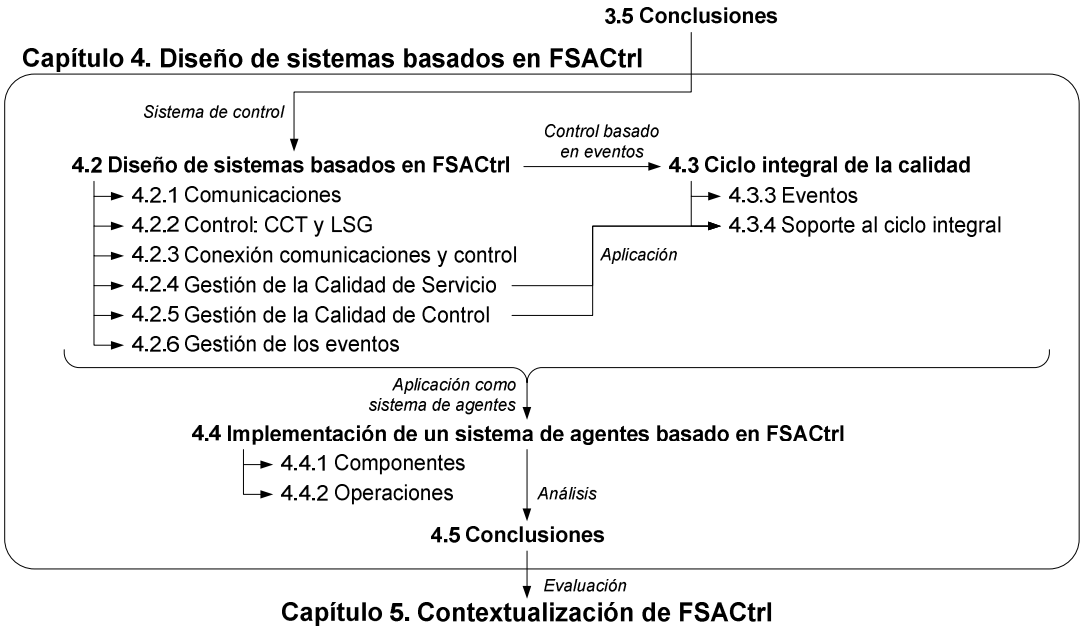


Figura 4. Diagrama conceptual del capítulo 4.

El quinto capítulo (Figura 5) realiza una comparativa de la arquitectura propuesta con sistemas similares. Esta comparativa contextualiza FSACtrl en las diversas áreas en las que puede aplicarse. Primero se describe cómo las características de la arquitectura cumplen con los requisitos para los sistemas distribuidos de control que se concluyeron en el capítulo 2.

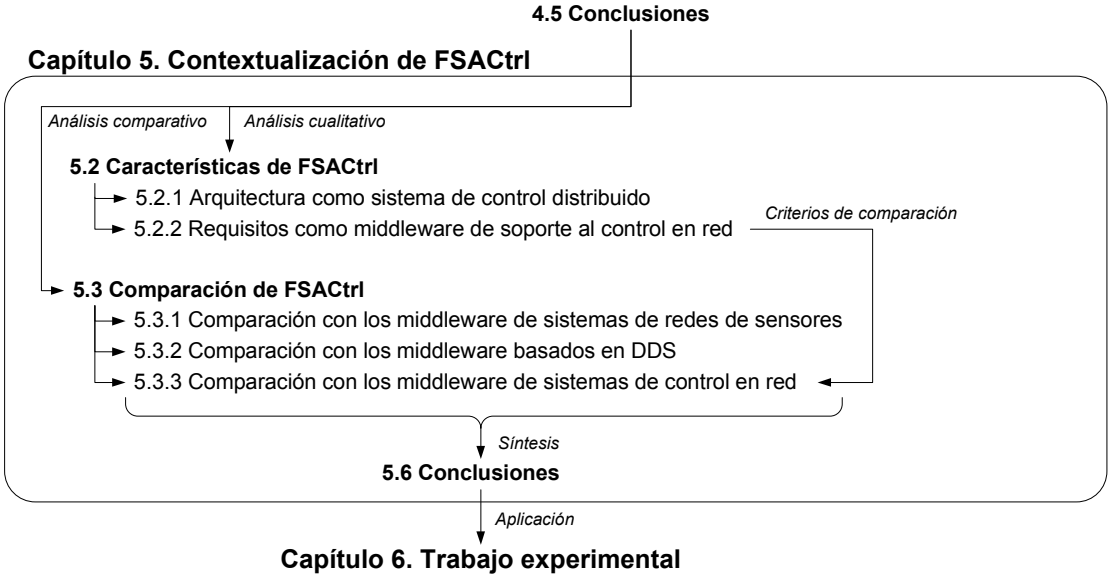


Figura 5. Diagrama conceptual del capítulo 5.

A continuación se compara cualitativamente FSACtrl con los middleware de redes de sensores (WSN), como ejemplo de sistemas con altos requisitos de volumen de información; con los middleware basados en el modelo DCPS (*Data Centric Publish Subscribe*) del estándar DDS (*Data Distribution Service for Real-time Systems*), por la similitud en el estándar empleado y con los middleware de sistemas de control en red (NCS), como principal área de aplicación.

El sexto capítulo (Figura 6) presenta el trabajo experimental desarrollado. Como plataforma de prueba se emplean los primeros cinco vehículos de Braitenberg. Los cuatro primeros vehículos emplean un control reactivo básico y permiten comprobar la viabilidad de la arquitectura como sistema de control. La optimización se obtiene a partir de la gestión de los mensajes por medio de políticas de QoS o de la selección de mensajes desde el middleware al sistema de control. El quinto vehículo de Braitenberg aporta el razonamiento lógico y se emplea para comprobar cómo la arquitectura optimiza conjuntamente las comunicaciones y el control a partir de la configuración de las políticas de QoS.

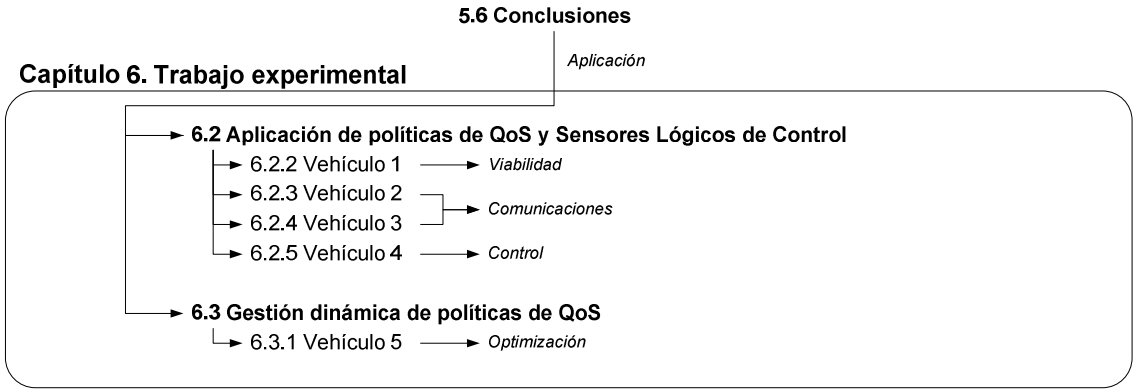


Figura 6. Diagrama conceptual del capítulo 6.

Por último, en el séptimo capítulo (Figura 7), se presentan las conclusiones mostrando el trabajo desarrollado y los objetivos alcanzados. Además, se exponen las aportaciones de la tesis en los diversos campos de investigación y en concreto en las arquitecturas de control. Posteriormente, se propone una serie de trabajos futuros, como la autoconfiguración de un sistema o la definición de parámetros de rendimiento del mismo. A continuación se exponen las publicaciones en las que se divulga el trabajo desarrollado en la tesis y, finalmente, se relacionan los bloques de la arquitectura propuesta con los proyectos de investigación en los que se ha utilizado.

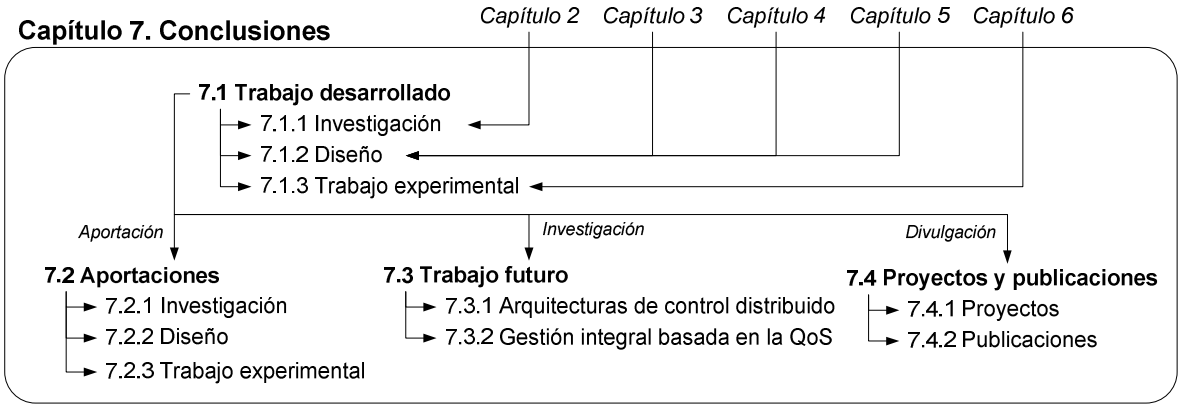


Figura 7. Diagrama conceptual de las conclusiones de la tesis.

2 Fundamentos teóricos

2.1 Introducción

2.1.1 Motivación

Para responder a la hipótesis planteada en la presente tesis doctoral es preciso conocer con detalle las características más relevantes de las arquitecturas distribuidas de control inteligente. Para ello se han analizado diversas arquitecturas, incidiendo especialmente en las arquitecturas de los sistemas de control ambiental y en los sistemas control de robots móviles. Dado que el ámbito de actuación de la tesis son los sistemas distribuidos, también es preciso conocer las características de los sistemas de comunicaciones ya que son la base sobre la que desarrollan su función las arquitecturas de control.

El presente capítulo cubre los primeros dos objetivos de la tesis: revisar las arquitecturas de control distribuido inteligente y de los sistemas de comunicaciones que les dan soporte para extraer las características más relevantes de las mismas, y analizar los parámetros de rendimiento de los sistemas para establecer la relación de los mismos con la calidad de servicio. A partir de la revisión y el análisis desarrollado se obtienen las bases de la arquitectura propuesta en la presente tesis doctoral.

2.1.2 Arquitecturas

Definir qué es una arquitectura implica deliberar sobre la idoneidad de ubicarla en un ámbito concreto, como por ejemplo arquitectura de un sistema físico, arquitectura software o incluso arquitectura de la información de un sistema. No es intención de esta tesis profundizar en el término de arquitectura, habiendo trabajos que abordan esta cuestión detalladamente [Ronda, 2008].

En el estándar IEEE-1471, la arquitectura se define como la organización fundamental de un sistema, que incluye sus elementos, las relaciones entre sí y el ambiente, y los principios que gobiernan su diseño y evolución [IEEE, 2000]. En [Bass et al., 2003] se define como la estructura de estructuras de un sistema, la cual abarca componentes de software, propiedades externas visibles de estos componentes y sus relaciones.

Hay definiciones más cualitativas de lo que es una arquitectura. La arquitectura del software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. La arquitectura, por tanto, es el resultado de ensamblar un cierto número de elementos de forma adecuada para satisfacer la mayor funcionalidad y requisitos de desempeño de un sistema, así como requisitos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad [Kruchten, 1995].

En general, desde un punto de vista teórico, la especificación de una arquitectura está definida por sus propias características, lo que le otorga una categoría de abstracción del sistema real sobre el que actúa. Desde un punto de vista más práctico, una arquitectura se puede entender como la descripción de un sistema, lo que se hace desde un punto de vista estático, y su funcionamiento, lo que se lleva a cabo desde un punto de vista dinámico.

2.1.3 Control inteligente

Se entiende como control de un sistema a la disciplina donde los métodos de control están orientados a emular las características importantes de la inteligencia humana [Antsaklis, 1999]. Genéricamente se entiende el control como el proceso por el que se guía un sistema para lograr unos objetivos, expresados como resultados. Los resultados deberán estar dentro de unos parámetros definidos previamente para que el control se considere positivo.

Es importante tener en cuenta que el control puede tener sus objetivos acotados temporalmente. Cuando las restricciones temporales son a corto plazo, se habla de control reactivo. Por ejemplo, en un sistema domótico consistiría en lanzar una alarma en caso de una intrusión en un hogar y en el caso de un sistema de navegación de un robot evitar un obstáculo. A medida que la acotación temporal del control del sistema se va ampliando, se va tendiendo al control deliberativo con la consecuente ampliación del procesamiento de los datos. Por ejemplo, en el caso de un sistema domótico un control deliberativo sería el ahorro energético a lo largo del tiempo y en el caso de un sistema de navegación de un robot navegar de forma segura de un punto a otro para obtener un mapa del entorno a partir del cual planificar misiones.

El ofrecimiento de las funcionalidades que puede tener un sistema suele realizarse por medio de servicios, por lo que las arquitecturas deben incorporar estos aspectos a su funcionalidad. En estos casos se habla de arquitecturas orientadas a servicios o SOA (*Service Oriented Architecture*) y para la evaluación del cumplimiento por parte del sistema de unos requisitos especificados, se habla de la calidad de los servicios o QoS (*Quality of Service*).

Cabe destacar que los elementos de una arquitectura, incluyendo la información que estos requieren, cambian ampliamente desde el ámbito reactivo al ámbito deliberativo. En el ámbito reactivo la información requerida es, sobre todo, información instantánea y local proporcionada por los sensores. A medida que los requisitos del ámbito del sistema a controlar aumentan, los requisitos de información que se precisan son mayores, tanto en cantidad como en calidad de la misma. Por ejemplo, para detectar una intrusión en un hogar, el sistema domótico sólo debe reaccionar a la activación de uno, o pocos más, sensores de presencia; sin embargo, cuando se trata de realizar una gestión para optimizar el consumo energético del hogar, se debe tener en cuenta los valores de muchos sensores y de muchas mediciones a lo largo del tiempo, por lo que el volumen de información aumenta y, consiguientemente, la complejidad de la información a procesar por parte del sistema.

A medida que se desea dotar de mayor autonomía al sistema, los componentes de control pueden coincidir con los componentes de arquitecturas orientadas a otros ámbitos más deliberativos, este es el motivo de la gran relación entre las arquitecturas de control y las arquitecturas de sistemas de inteligencia artificial. Esta relación tan estrecha, hace que las implementaciones de las arquitecturas también sean muy similares, por ello es normal encontrar implementaciones de sistemas de control inteligente de sistemas distribuidos basadas en arquitecturas de agentes empleadas en la resolución de problemas de inteligencia artificial.

Para realizar un control inteligente de un sistema es necesario poder efectuar una gran cantidad de mediciones, cálculos y las correspondientes acciones. Por ejemplo, para el control energético de un sistema domótico será necesario tomar mediciones del consumo de los componentes del sistema, realizar cierta predicción a partir de la

comparación con los patrones establecidos, y tomar decisiones acerca de qué componentes del sistema tienen prioridad para consumir energía, y cuales deben reducir su consumo. Por tanto, la arquitectura del sistema es la que dota de la capacidad de alcanzar los objetivos, tanto a corto como a largo plazo, de un sistema [Orebäck and Christensen, 2003], siendo el diseño de la arquitectura la que le proporcionará funcionalidad y eficiencia al sistema [Coste-Manière and Simmons, 2000].

2.1.4 Escenarios de aplicación del control inteligente distribuido

En la Figura 8, se muestra una organización de los escenarios de aplicación de las arquitecturas de control inteligente en función de las restricciones temporales del sistema y del volumen de datos que se deben manejar en el sistema.

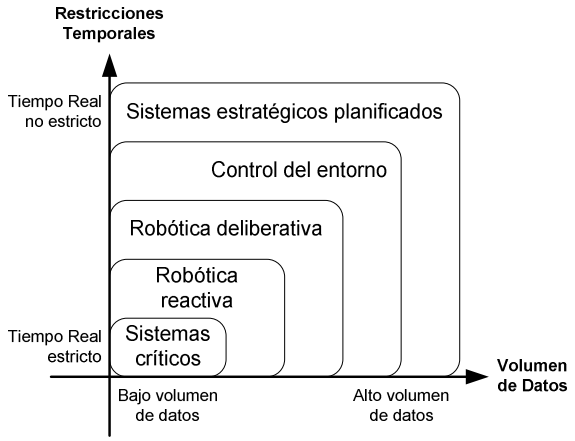


Figura 8. Ámbito de los escenarios de aplicación de las arquitecturas de control inteligente.

En el nivel con menores restricciones temporales y menor volumen de datos se enmarcan los sistemas domóticos. En este tipo de sistemas, el retraso en la reacción ante un cambio de las entradas no es crítico y es habitual trabajar con escala temporal de segundos. Además, el volumen de información es escaso, ya que el número de sensores de los que obtener información relevante suele ser escaso, lo que no implica que se trabaje con pocos sensores, sino que cada acción depende del conocimiento de escasos datos, así como del envío de acciones a un escaso número de actuadores.

El nivel intermedio lo conforman las aplicaciones robóticas. En las arquitecturas de navegación de robots las restricciones temporales pueden llegar a ser muy fuertes, como es el caso de la navegación reactiva, o tener un cierto margen, como es el caso de la navegación deliberativa. La navegación deliberativa tiene restricciones temporales menos fuertes que la navegación reactiva, lo que se refleja en que la navegación deliberativa se presta a la aplicación de algoritmos de inteligencia artificial, mientras que la reactiva se centra en el control de tiempo real. Un paso más en la complejidad se da cuando el volumen de datos que requiere el sistema va aumentando progresivamente.

El nivel más amplio lo forman los sistemas estratégicos planificados. Estos sistemas cubren casi todo el rango de restricciones temporales y de volumen de datos. Cuando las restricciones temporales son muy estrictas es habitual hablar de sistemas críticos como centrales nucleares o sistemas aeroespaciales. Generalmente la necesidad de una respuesta válida en un plazo concreto hace que se reduzca la cantidad de información con la que el sistema deba trabajar.

En el otro lado están los sistemas con menores restricciones temporales, pero habitualmente con un gran volumen de datos, bien por la cantidad de sensores y actuadores involucrados como por la necesidad de mantenimiento de un gran número de datos para la toma de decisiones. En éste último caso suele hablarse de sistemas de planificación estratégica, como las terminales de contenedores o los sistemas de automatización de industrias complejas.

En todos los escenarios se precisa de una infraestructura de comunicaciones conocida como sistema distribuido. Un sistema distribuido se define como: sistema en el que los componentes, conectados en red, comunican y coordinan sus acciones únicamente mediante el paso de mensajes [Coulouris et al., 2001].

2.1.5 Descripción del capítulo

Para realizar una propuesta de arquitectura que pueda dar soporte a los requisitos expuestos anteriormente, es preciso analizar las características presentes en las arquitecturas de control distribuido inteligente dependiendo de su ámbito de aplicación. Para ello, se realiza una revisión de las arquitecturas actuales orientadas a los sistemas ciber-físicos.

No es el objetivo de este capítulo la recopilación y exposición de dichas arquitecturas, sino la extracción de las características más relevantes que faciliten el diseño de la arquitectura propuesta. Los detalles de todas las arquitecturas analizadas se encuentran en [Poza et al., 2009c], la discusión correspondiente conformará el apartado 2.2 de Arquitecturas de control.

Conocer las características de las arquitecturas es tan sólo uno de los dos pilares necesarios para justificar el diseño de la arquitectura propuesta, el otro es la elección del paradigma de comunicaciones más adecuado, para ello, en el apartado 2.3 se encuentra la discusión acerca de los diferentes paradigmas de comunicaciones. Los detalles del análisis de los sistemas de comunicaciones se encuentran en [Poza, 2009a].

La tendencia actual al desarrollo de sistemas basados en servicios hace que la calidad del servicio suministrado sea otro de los puntos importantes a los que prestar atención. Para ello en el apartado 2.4 se realiza un repaso tanto del concepto de calidad de servicio como de los parámetros más empleados y del concepto de política de calidad de servicio. Los detalles del análisis del papel que desempeña la calidad de servicio en las comunicaciones se encuentran en [Poza, 2009d].

En el entorno de control y de comunicaciones en que se desarrolla la presente tesis destacan dos estándares en los que se apoya el trabajo: son el modelo SWE (Sensor Web Enablement) propuesto por la OGC (Open Geospatial Consortium) y el modelo DCPS propuesto por la OMG (Object Management Group). La descripción de estos estándares se desarrolla en el apartado 2.5, mientras que los detalles de ambos modelos se encuentran en [Poza, 2009b] y [Poza, 2009c].

Finalmente, en el último apartado se exponen las conclusiones de las revisiones presentadas en los apartados anteriores. Las características analizadas en el presente capítulo han guiado el diseño de la arquitectura propuesta: en la definición de los elementos que la componen, en la interacción entre dichos elementos y en la parametrización del sistema.

2.2 Arquitecturas de control

2.2.1 Arquitecturas de control de entornos

2.2.1.1 Evolución del control inteligente de entornos

En los sistemas de control, hay dos formas de distribuir los sensores y los actuadores en el entorno: ramificarlos o concentrarlos en el escenario donde actuarán. En el primero de los casos la densidad de sensores y actuadores por superficie es muy escasa, sin embargo es cercana a la fuente de datos, es lo que se conoce como el control de entornos. En el segundo caso la concentración de sensores y actuadores es alta generalmente en instrumentos, que sí cumplen un cometido concreto y se habla de robótica. El control de entornos ha ido evolucionando en la misma medida que el área de control en que se aplican, partiendo de la domótica se llega al concepto actual de control inteligente de entornos.

La domótica proviene de la unión de las palabras domus, casa en latín, y robótica del término *robota* o esclavo en checo, aunque algunas fuentes hablan de automática en lugar de robótica como segunda acepción. Actualmente se entiende por domótica al conjunto de sistemas que automatizan un hogar, aunque poco a poco se está extendiendo a la automatización de un hogar a un entorno habitable, como oficinas o edificios. En éste último caso, en lugar de hablar de *domótica*, se habla de *inmótica*.

Tanto los sistemas domóticos como los inmóticos se componen de sensores comunicados por redes de comunicación [Petriu et al., 2000]. En cuanto a las referencias en investigación, el convenio actual es llamar *home automation* a la automatización de un entorno habitable, y *smart home* cuando a dicha automatización se le añade un control inteligente de un hogar.

La línea de investigación más actual es la conocida como *entornos inteligentes* o *smart environment*, término que es más genérico que los anteriores. Con éste último término se pueden englobar muchos más sistemas de control inteligente, abarcando un ámbito de aplicación más extenso.

El control inteligente de entornos es, en la mayor parte de las ocasiones, un control reactivo, como puede ser la activación de alarmas, o el control de luces en función de la detección de presencia. Sin embargo, dada la particular configuración de sensores distribuidos de los sistemas domóticos, hace que sean unos escenarios idóneos para la aplicación de nuevas tendencias de inteligencia ambiental y computación ubicua [Ruyter et al., 2005]. Actualmente, la tendencia en investigación de domótica consiste en la integración de los sensores dentro de sistemas de computación avanzada para determinar entornos que se adapten a los requisitos de seguridad, comodidad o eficiencia energética definidos por el usuario. A medida que evolucionen los sistemas domóticos se podrá llegar a obtener sistemas auto configurables con capacidad de aprendizaje.

2.2.1.2 Características de los sistemas de control del entorno

La heterogeneidad de los sistemas de control del entorno dificulta la determinación de unas características comunes. En [Aiello, 2005] se destacan las siguientes características que se deben evaluar en un sistema domótico para determinar las capacidades potenciales del mismo.

A partir de estas características se pueden obtener las necesidades de diseño de una arquitectura que pueda cubrir la funcionalidad del control inteligente de entornos.

- Apertura. Se entiende como la publicidad en el sistema que se da del protocolo, y la posibilidad de implementarlo en cualquier localización. Un sistema abierto permite su instalación en el mayor número de escenarios con el menor número de cambios.
- Escalabilidad. Se habla de escalabilidad de un sistema domótico como la posibilidad de agregar o eliminar dispositivos a una red doméstica sin afectar a sus funcionalidades ni a su rendimiento. Se habla de escalabilidad dinámica en el caso de que se puedan agregar o quitar dispositivos cuando el sistema está en funcionamiento normal, mientras que la escalabilidad estática es la apertura, característica expuesta anteriormente.
- Heterogeneidad. Un sistema domótico es heterogéneo cuando la infraestructura del sistema soporta diferentes dispositivos hardware, redes, sistemas operativos o lenguajes de programación del control sin imponer unas restricciones. También se habla de sistema heterogéneo cuando los detalles anteriores son evitables, generalmente por medio de capas intermedias de traducción o de protocolos estándar.
- Topología. La topología de un sistema domótico es la forma en que los dispositivos están conectados unos con otros. Se habla de topología física cuando se expone la forma de conexión real, como punto a punto, bus y similares y se habla de topología lógica cuando se expone cómo se relacionan los componentes entre ellos.

Históricamente unas combinaciones se han dado con más frecuencia que otras. Aunque es difícil clasificar un sistema domótico, el punto de vista de evolución temporal es adecuado para tener una serie de escenarios desde los que poder ubicar o encuadrar una arquitectura concreta [Cook and Das, 2007].

Inicialmente los sistemas se basaban en un único bus, por lo que eran poco abiertos y sin posibilidades de ser fácilmente escalables. La escalabilidad del sistema era la del bus y la posibilidad de crear un sistema heterogéneo mínima ya que solo los dispositivos compatibles con el bus tenían posibilidad de conectarse. Los componentes se enviaban mensajes directamente y con una semántica específica del origen, el destino y el contenido.

A medida que las necesidades de escalabilidad aumentaban, los sistemas pasan a tener un servidor. En el caso de los sistemas basados en un único bus la aparición de un servidor añadía cierta complejidad, por lo que se disponía de un servidor único desde el que controlar todos los componentes.

Para permitir una mayor apertura del sistema, el servidor único pasa a ser un servidor abierto, permitiéndose la aparición de diversos servidores, lo que implicaba la aparición de protocolos más complejos a cambio de una gran capacidad de distribución.

El último de los pasos, donde la escalabilidad y la apertura son grandes son los sistemas basados en los servicios de los componentes, de tal forma que un componente adquiere el doble rol de proveedor de servicios y cliente a su vez de otros proveedores.

2.2.1.3 Revisión

Hay una gran diversidad de arquitecturas de control domótico por lo que se deben destacar aquellas que provean de las características más relevantes. Los detalles de la revisión realizada entre las arquitecturas existentes se puede obtener en [Poza et al., 2009c].

Centrándose en el ámbito del control domótico destacan diversos aspectos. El uso de componentes y de agentes es generalizado en todas las arquitecturas. Aunque algunas arquitecturas se basan en el control centralizado, la tendencia es emplear una arquitectura de control distribuido. Puede parecer que el uso del control centralizado no sea una ventaja tecnológica, especialmente teniendo en cuenta la tendencia al control distribuido, pero en sistemas con escasez de requisitos temporales y bajo volumen de datos no es tan necesaria la cercanía de los algoritmos de control a los componentes físicos, por lo los sistemas centralizado prevalecen sobre los distribuidos.

Las arquitecturas domóticas organizan los componentes de control de diversas formas. C@sa [De Carolis and Cozzolongo, 2004] divide el control por áreas, sin embargo ACHE [Mozer, 1998] introduce el concepto de *dominio de control* dividiendo el ámbito de control y centralizando cada dominio en el correspondiente componente. MavHome [Cook et al., 2003] organiza los componentes de forma jerárquica, en consonancia con el control jerárquico, lo que le proporciona la posibilidad de distribuir el control entre diversos agentes.

Sin embargo, lo más destacable es que en la práctica totalidad de las arquitecturas el control se basa en eventos, este hecho, unido a la distribución del control y el soporte al modelo multi-agente deberá ser soportado de forma eficiente por los sistemas de comunicaciones que se empleen, por lo que será un factor importante en el diseño de una arquitectura de control distribuido.

En el ámbito de las comunicaciones en las arquitecturas de control domótico se aprecia claramente la evolución en el uso de diversos paradigmas. Tan sólo ACHE emplea explícitamente una arquitectura de bus. El resto de las arquitecturas emplean servicios para que los componentes puedan solicitar acciones unos de otros. En ciertos casos, el soporte de comunicaciones está limitado, es el caso de HomeAPI [Bizarri, 1999] basado en el modelo COM/OLE (Component Object Model / Object Linking and Embedding) o el caso de MASSIHN [Cheng-Fa and Hang-Chang, 2002] basado en RMI (Java Remote Method Invocation). Sin embargo HAS [Chao-Lin et al., 2004] busca la integración de las diferentes tecnologías de red y DomoNet [Miori et al., 2006] homogeniza los servicios ofrecidos desde los componentes.

El direccionamiento de los componentes es otro de los aspectos clave en las arquitecturas. Aladdin [Wang et al., 2000] emplea un servicio de nombres y atributos. HomeAPI amplía el servicio de nombres y localiza los componentes por medio de una jerarquía de espacio de nombres. Es interesante destacar cómo las arquitecturas precisan de un medio de direccionamiento de los componentes para jerarquizar el sistema en áreas específicas de control.

En cuanto a las funcionalidades de las arquitecturas domóticas destacan la predicción y el aprendizaje, ambas funcionalidades buscan la adaptabilidad del sistema a los diferentes entornos a controlar. Para ello, las arquitecturas se sustentan fundamentalmente en el uso de componentes y especialmente de agentes debido a la flexibilidad funcional de los mismos.

El uso de control centralizado o distribuido depende más de las particularidades del sistema a controlar que de la arquitectura en sí misma. Por ello, es importante que la arquitectura disponga de la posibilidad de distribuir sus componentes en función de las necesidades del entorno a controlar. Es imprescindible que la arquitectura soporte el control basado en eventos. La diversidad de medios por los que los mensajes deberán circular hace que también sea común que un sistema de control domótico tenga soporte a diversos protocolos de comunicación.

2.2.2 Arquitecturas de navegación de robots

2.2.2.1 Navegación de robots

En los sistemas de navegación de robots, lo más habitual es que una arquitectura se desarrolle para un modelo de robot concreto que debe realizar sus actividades en una serie de entornos específicos. Si se tiene en cuenta que la mayoría de las arquitecturas están inspiradas en modelos naturales, el hecho de que exista una variedad tan amplia de arquitecturas no es nada diferente de lo que ocurre en la naturaleza, donde también hay una gran variedad de especies dependientes de las funciones y entornos en los que habitan.

En la navegación controlada por el nivel reactivo, el robot debe reaccionar rápidamente ante los cambios del entorno. Por este motivo, los requisitos de información de los algoritmos son mínimos. Además las fuentes de datos (generalmente los sensores) están muy cerca de los agentes que deben procesar la información. En el nivel reactivo, o los niveles cercanos a éste, los requisitos del sistema de comunicaciones son principalmente: tiempo real en las transmisiones y fiabilidad en la información transmitida, ya que una retransmisión de un dato supone un consumo temporal muy grande.

Sin embargo, en el nivel deliberativo, o a medida que los niveles de control se acercan más al nivel deliberativo, el robot necesita reaccionar inteligentemente ante el entorno. Es por esto que el nivel deliberativo requiere de una información más elaborada y compleja. Las fuentes de datos (sensores internos o externos de otros robots) están más distantes, tanto en el tiempo como en el espacio, que en los niveles reactivos. Los requisitos del sistema de comunicación en la capa deliberativa, principalmente son flexibilidad y adaptabilidad. La diversidad de agentes usados en la navegación deliberativa, debido a la diversidad de comportamientos inteligentes que requiere el nivel, implica que los datos de una misma fuente (por ejemplo del mismo sensor) pueden ser usados de formas distintas dependiendo de los requisitos de los agentes. Esta diversidad implica una variedad de procesado de los datos a medida que se acercan a la capa deliberativa. El sistema de comunicaciones puede colaborar con las capas, a medida que se incrementa su función deliberativa, distribuyendo, organizando, filtrando, integrando y preprocesando la información que transporta.

2.2.2.2 Características de las arquitecturas de navegación de robots

Diversos autores han destacado las características comunes de las arquitecturas de navegación de robots, ya sea para hacer un análisis y posterior evaluación de las mismas o para hacer una taxonomía más o menos compleja. Intentar destacar o aislar puntos comunes por los que clasificar o evaluar arquitecturas es complicado ya que cada una de ellas tiene objetivos, componentes y funcionalidades diferentes. A continuación se muestran las principales revisiones.

La primera aproximación se encuentra en [Brooks, 1991a] y se amplía en [Brooks, 1991b] donde se desarrollan las características específicas para los niveles de comportamientos en los que se basa su arquitectura. Las características son las siguientes:

- Generalidad (*Generality*) y versatilidad (*Versatility*) La generalidad se refiere a la amplitud de escenarios donde la arquitectura del robot puede actuar. La versatilidad muestra la capacidad para adaptarse a las diversas situaciones.
- Racionalidad (*Rationality*). Esta característica se refiere a la actuación racional del robot, o lo que es lo mismo, la capacidad que le permite actuar de acuerdo a unos principios establecidos previamente.
- Creatividad o capacidad para adquirir o asimilar nuevos conocimientos (*Ability to add new knowledge*). Se refiere a la capacidad para recoger información del entorno y procesarla de manera que sea útil para la navegación del robot.
- Capacidad para aprender (*Ability to learn*). Como capacidad de aprendizaje se entiende la capacidad de generación de nuevos comportamientos a partir de la información adquirida o de otros comportamientos ya existentes.
- Capacidad de descomposición en tareas (*Taskability*). Esta característica se refiere a la posibilidad de que un robot tenga la facultad de organizar una tarea de cierta complejidad en sub-tareas de menor complejidad. Además las sub-tareas deben ser activadas en los instantes precisos, tanto secuencialmente como concurrentemente, para que se obtenga el resultado previsto para la tarea original.
- Escalabilidad (*Scalability*). La escalabilidad se define en términos de posibilidad de poder ampliar la potencia del sistema, lo que implica poder ampliar tanto componentes como capacidad de procesamiento y comunicación.
- Reactividad (*Reactivity*). La reactividad es la capacidad de reacción en el tiempo que se considere adecuado ante los cambios del entorno. No se debe confundir con la capa reactiva de navegación ya que esta característica se refiere a un concepto y no a un comportamiento.
- Eficiencia (*Efficiency*). Esta característica se refiere a la capacidad de lograr realizar las tareas planificadas optimizando unos parámetros definidos en función del logro de unos requisitos.

A partir de la aproximación de Brooks, en [Arkin, 1998] se agrupan los criterios que se emplean para evaluar y comparar las arquitecturas, dando lugar a las siguientes características:

- Modularidad (*Modularity*): Esta característica trata acerca de la independencia entre los distintos componentes. La modularidad incluye características intrínsecas como la posibilidad de reutilización de los módulos y la escalabilidad del sistema.
- Adaptabilidad (*Adaptability*): La adaptabilidad se entiende como la capacidad de adecuar la arquitectura a las aplicaciones para las que ha sido diseñada minimizando los cambios a realizar.

- Portabilidad (*Portability*): La portabilidad es la facilidad de hacer funcionar la arquitectura en diferentes sistemas. Esta característica se diferencia de la adaptabilidad en que la portabilidad implica el cambio completo de sistema sobre el que se aplica.
- Robustez (*Robustness*): La robustez se entiende como la capacidad de la arquitectura de resistir a los errores que puedan aparecer en su funcionamiento.

La siguiente aproximación a las características que debe tener una arquitectura para controlar la navegación de un robot móvil es la que se encuentra en [Alami et al., 1998]. Aunque algunas son coincidentes con las anteriormente expuestas, es interesante comprobar cómo se amplían los conceptos.

- Capacidad de programación (*Programmability*). Esta característica se refiere a la capacidad que tiene la arquitectura de ser programada ya que es importante que un sistema autónomo pueda desarrollar el mayor número de funciones en el mayor número de entornos.
- Autonomía y adaptación (*Autonomy and Adaptability*). Estas dos características se encuentran unidas ya que cuanta más capacidad de adaptación se tenga en un sistema, de más autonomía se dispondrá. La autonomía, por tanto, se considera la capacidad de funcionar sin necesidad de actuaciones externas.
- Reactividad (*Reactivity*). La capacidad de reaccionar en los límites temporales correspondientes debe ser compatible con la realización de los objetivos que el sistema tenga asignados. Lo anterior implica que el robot no debe dedicar todo el tiempo a las misiones programadas olvidando la reacción ante situaciones que pongan en peligro su propia seguridad.
- Comportamientos coherentes (*Consistent behavior*). La característica de coherencia se refiere a que la arquitectura debe dar soporte a que el robot realice las tareas orientadas a los objetivos asignados y adecuadamente en el entorno en que las desarrolla.
- Robustez (*Robustness*). El concepto de robustez es similar a los vistos anteriormente, en el caso de la arquitectura de un robot se debe poder soportar la redundancia en el procesamiento de los datos. En el caso de los sistemas distribuidos se asocia la robustez a la descentralización del control, y por tanto la distribución de tareas y de datos para poder delegar en diferentes componentes las mismas tareas en caso de un mal funcionamiento del sistema.
- Extensibilidad (*Extensibility*). La arquitectura debe soportar la escalabilidad del sistema, de manera que la incorporación de tareas no debe implicar cambios en la estructura y la composición del sistema.

Como se puede deducir de las revisiones anteriores de características, resulta difícil concretar algunas como relevantes frente a otras, ya que en los sistemas de control de navegación de robots, las arquitecturas cubren un espectro muy amplio. Dependiendo del ámbito de funcionamiento del sistema serán deseables unas u otras características.

2.2.2.3 Revisión

La gran cantidad de arquitecturas de control de la navegación de robots existentes da lugar a un campo de investigación muy abierto y en continua evolución. La modularidad de una arquitectura es un indicador de la capacidad de distribución que tiene.

El nivel de modularidad de cada arquitectura se ha entendido como lo divisible que puede ser el control y por tanto lo adaptado que está a una situación concreta ante pequeños cambios del entorno. Se ha valorado la modularidad de las capas deliberativa y reactiva. En la Tabla 1 se expone el análisis de la modularidad de las diferentes arquitecturas revisadas.

Tabla 1 Valoración de la modularidad de las arquitecturas revisadas

Arquitectura	Tipo	Subtipo	Nivel Reactiva	Nivel Deliberativa	Modularidad
Subsunción	Reactiva	Jerárquica	Alta	Alta	Muy alta
JPL	Deliberativa	Secuencial	Baja	Baja	Muy baja
NASREM	Deliberativa	Paralelo	Alta	Alta	Muy alta
AuRA	Híbrida	Organizativa	Alta	Baja	Media
SFX	Híbrida	Organizativa	Alta	Baja	Media
LAAS	Híbrida	Organizativa	Alta	Baja	Media
3T	Híbrida	Jerárquica de estados	Alta	Alta	Muy alta
ATLANTIS	Híbrida	Jerárquica de estados	Media	Alta	Alta
Saphira	Híbrida	Orientada a modelos	Media	Baja	Baja
TCA	Híbrida	Orientada a modelos	Media	Baja	Baja
GLAIR	Híbrida	Niveles	Baja	Media	Baja
Sharp	Híbrida	Niveles	Media	Media	Media
BERRA	Híbrida	Niveles	Baja	Media	Baja
SSS	Híbrida	Niveles	Media	Media	Media
Payton's	Híbrida	Niveles	Media	Media	Media

Como se puede deducir de la valoración anterior, una de las características destacables para que la arquitectura sea adaptable es la posibilidad de generar comportamientos avanzados sin necesidad de un alto coste en la organización de sus componentes. Por tanto, la organización de los componentes, especialmente la posibilidad de generalizar sus servicios es una de las funcionalidades que tienen las arquitecturas adaptables. A continuación se procederá a intentar extraer los aspectos particulares de algunas de las arquitecturas anteriores que tienen una valoración media o alta tanto en la capa reactiva como en la deliberativa.

La arquitectura más referenciada en las publicaciones de sistemas de control, es la de subsunción [Brooks, 1986], no en vano es una de las que implementa de forma más homogénea y elegante un control deliberativo a través de una jerarquización basada en niveles priorizados. Los comportamientos de supervivencia son los más prioritarios, mientras que los menos prioritarios son los puramente deliberativos basados en el razonamiento. Sin embargo, en este tipo de arquitecturas unas condiciones de un entorno continuamente cambiante, por ejemplo, pueden hacer que los comportamientos deliberativos no terminen emergiendo debido a la prioridad de los comportamientos reactivos.

De la arquitectura NASREM [Albus and Barbera, 2005] cabe destacar la estructuración homogénea independientemente del nivel, reactivo o deliberativo, en el que se opere, lo que la hace muy adaptable. Además el emplear un método para almacenar y compartir el conocimiento global la hace muy conveniente para el aprendizaje y el uso de agentes de control.

La reutilización de componentes diferenciados para el control reactivo es un patrón común en varias de las arquitecturas. En la arquitectura AuRA [Arkin, 1990] la división del control en módulos se realiza por medio del empleo de esquemas reutilizables en la capa reactiva, lo que hace que sea muy adaptable a diversos entornos.

Algo similar sucede en SFX [Murphy, 2000] donde los esquemas son sustituidos por comportamientos comunicados con agentes. En LAAS [Alami et al., 2000] la división se realiza directamente en los denominados módulos de control. En la arquitectura 3T [Bonasso, 1997] esa división se realiza en las denominadas habilidades, accionadas por eventos externos al sistema y planificada de una forma jerárquica.

En lo que se refiere al control deliberativo, en la mayor parte de las arquitecturas es realizado por módulos muy específicos para cada uso, por ejemplo la generación de mapas o la planificación de una ruta. Sin embargo, algunas de las arquitecturas tienen módulos genéricos, lo que permite mejorar el diseño del sistema y la distribución de los componentes.

En subsunción el comportamiento cercano al deliberativo aparece como comportamiento emergente a partir de los comportamientos de los módulos reactivos. Un ejemplo de modularidad genérica es la implementada en la arquitectura NASREM donde los módulos deliberativos son tratados de la misma forma que los reactivos, lo que hace que esta arquitectura tenga una homogeneidad muy elegante. En la arquitectura 3T la modularidad deliberativa se logra por medio de una jerarquía de objetivos cuya sucesiva transformación en subobjetivos conlleva la asignación automática de tareas. En ATLANTIS los módulos son homogéneos y se implementan por medio de procesos LISP (LISt Processing).

2.2.3 Discusión

A continuación se organizan los conceptos que se han presentado en los apartados anteriores. A partir de las principales características se deducirán los requisitos que se consideran más relevantes para el diseño de una arquitectura distribuida de control inteligente.

2.2.3.1 Sobre las arquitecturas

De la revisión de las arquitecturas domóticas, se deduce cómo éstas están más adaptadas a la distribución de sus componentes que las arquitecturas de navegación de robots. Esto se debe fundamentalmente a la aparición posterior de éstas y el consecuente aprovechamiento de las tecnologías de comunicaciones que han evolucionado de forma paralela. Además de la funcionalidad a la que está destinada la arquitectura, se debe tener en cuenta la topología del control, o lo que es lo mismo la ubicación de los componentes que implementan el control. El hecho de tener los componentes de control distribuidos implicará tener en cuenta las comunicaciones como parte fundamental de la arquitectura y por tanto del diseño de la misma.

En las arquitecturas domóticas, la tendencia es disponer de una infraestructura que permita un sistema distribuido, heterogéneo, con escalabilidad. Esto se logra por medio de los sistemas basados en servicios.

En las arquitecturas de navegación de robots el paradigma híbrido es la tendencia más empleada, ya que contiene las ventajas del paradigma reactivo, con el cumplimiento de las restricciones temporales, y las ventajas del paradigma deliberativo, con la capacidad de realización de misiones complejas

2.2.3.2 Sobre las características

En la Tabla 2 se ha organizado la frecuencia de aparición de las características de las arquitecturas de control de los sistemas domóticos y de navegación de robots revisadas en los apartados anteriores.

Tabla 2. Características de los sistemas de control distribuido.

Característica	[Aiello, 2005]	[Brooks, 1991]	[Arkin, 1998]	[Alami, 1998]	Relevancia
Apertura	Sí				Baja
Escalabilidad	Sí	Sí		Sí	Alta
Heterogeneidad	Sí				Baja
Topología	Sí				Baja
Generalidad		Sí			Baja
Versatilidad		Sí			Baja
Adaptabilidad		Sí	Sí	Sí	Alta
Racionalidad		Sí			Baja
Aprendizaje		Sí			Baja
Planificación		Sí			Baja
Reactividad		Sí		Sí	Media
Eficiencia		Sí			Baja
Modularidad			Sí		Baja
Portabilidad			Sí		Baja
Robustez			Sí	Sí	Media
Programabilidad				Sí	Baja
Coherencia				Sí	Baja

Determinar las características más adecuadas de una arquitectura óptima es una tarea complicada, ya que algunas están enfrentadas, como la escalabilidad y la heterogeneidad. Conjuntar las características y elaborar los parámetros que permitan optimizar un sistema constituye un interesante reto de investigación.

2.2.3.3 Características para la optimización de sistemas

La propuesta de características de sistema óptimo de la Tabla 2 es la base para el diseño de la arquitectura que se propone en esta tesis, además forman parte de los objetivos del proyecto SIDIRELI [SIDIRELI, 2008]. A continuación se proporciona la definición detallada de cada una de las características:

- Autonomía temporal. La autonomía temporal es un concepto referido a los componentes del sistema que consiste en la capacidad que tienen éstos de poder prescindir de un reloj común, y sin embargo poder obtener información temporal de los mensajes o del propio funcionamiento, y ofrecer esa información temporal al usuario para cumplir los requisitos temporales que se hayan determinado.
- Autonomía espacial. Esta característica tiene dos significados. El primero se refiere a la posibilidad que tienen los componentes de poder continuar trabajando independiente de parte de la información que requieren para su funcionamiento. A mayor independencia espacial, se podrá minimizar el impacto de la ausencia de la información en el rendimiento del sistema. El segundo se refiere a poder trabajar desconociendo la ubicación de los componentes de los que se recibe, o a los que se envía información.

- Confiabilidad. Esta característica es una síntesis de las dos anteriores. Un sistema que proporciona un control temporal y tenga una cierta autonomía del resto de los componentes es un sistema en el que se puede confiar el control. La confiabilidad es una extensión del concepto clásico de fiabilidad. La fiabilidad se refiere a la ausencia de errores en el sistema, mientras que la confiabilidad implica la independencia ante los efectos de los errores.
- Recuperabilidad. La recuperabilidad es la capacidad de reorganizar el sistema ante cambios de condiciones de funcionamiento. Para reorganizar el sistema, los componentes deben ser capaces de moverse o de transferir sus servicios a otros componentes.
- Reutilización. Esta característica se refiere a la posibilidad de que el sistema ofrezca la posibilidad de poder emplear el mismo componente para situaciones distintas. Es una característica que requiere del sistema una flexibilidad para adaptar los componentes de control y de comunicaciones en función de la variabilidad de los escenarios en los que se aplican.
- Adaptabilidad. La adaptabilidad es la capacidad de proporcionar una respuesta robusta del sistema ante un cambio en las condiciones del mismo. Los cambios de condiciones se refieren tanto a las condiciones de realidad física sobre la que el sistema trabaja, como a las condiciones lógicas referentes a las comunicaciones o al control.
- Estabilidad. La estabilidad se entiende como la capacidad del sistema de mantenerse dentro de unos márgenes de funcionamiento que se consideran estables. Para ello el sistema debe poder formular sus características de forma cuantitativa y permitir la adaptación del mismo para cumplir los requisitos.
- Movilidad. La movilidad es la capacidad de poder reubicar espacialmente los componentes. De esta forma se puede realizar un reparto de roles y tareas en el sistema. Para lograr la movilidad de componentes o las partes del mismo, se debe proporcionar un mecanismo para referirse a los componentes abstrayendo los detalles del sistema. Como valor añadido, los componentes deben tener la capacidad de descubrir las ubicaciones donde poder ubicarse.

A medida que van aumentando las prestaciones que permiten optimizar un sistema, también aparecen algunas dificultades. Por ejemplo, la movilidad de componentes permite clonar los mismos para aumentar la fiabilidad de un sistema, sin embargo, esa clonación implica un aumento en la redundancia de la información. En la Figura 9 se muestra las relaciones entre las características anteriores.

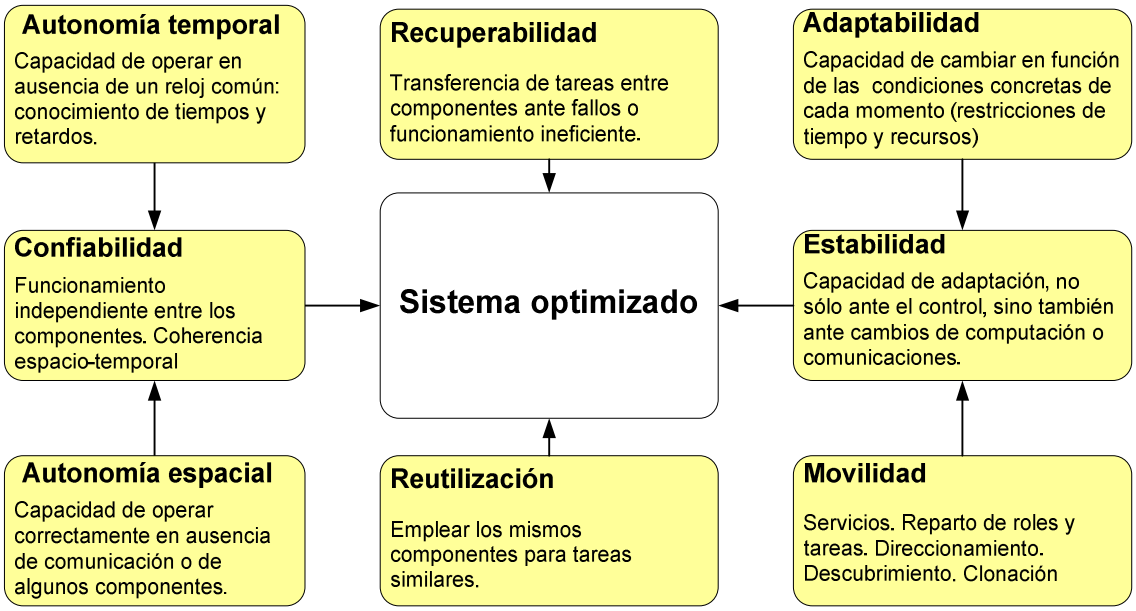


Figura 9. Principales características de los sistemas ciber físicos con relaciones entre ellas.

Una vez determinadas las características idóneas de un sistema de control distribuido se debe analizar cómo debe ser el sistema de comunicaciones que le de soporte y proporcione los parámetros que permitan optimizar el sistema. A continuación se analizarán los sistemas de comunicaciones y la parametrización de los mismos para la medición de la calidad.

2.3 Sistemas de comunicaciones

En las arquitecturas distribuidas de control inteligente, las comunicaciones son uno de los pilares fundamentales en los que se basan los componentes por lo que resulta de gran interés conocer sus características. Para obtener las características más relevantes se han analizado una gran cantidad de sistemas de comunicaciones [Poza, 2009a]. A continuación se expone la contextualización de las comunicaciones en los sistemas distribuido y los resultados más relevantes del análisis realizado.

2.3.1 Sistemas distribuidos

2.3.1.1 Definición

Un sistema distribuido se define como aquel en el que los componentes, conectados en red, comunican y coordinan sus acciones únicamente mediante el paso de mensajes [Coulouris et al., 2001]. Es habitual que los sistemas distribuidos se caractericen por la concurrencia de los componentes, la ausencia de un reloj común y la independencia de los fallos. La concurrencia permite que los recursos disponibles, incluidos los propios componentes, puedan ser utilizados simultáneamente. La ausencia de un reloj común, o global, se debe a que la transferencia de mensajes entre los componentes no tiene una base temporal común sino que la gestión temporal está distribuida en los componentes. Finalmente el aislamiento del sistema de los fallos de los componentes implica que cada componente debe tener una independencia con respecto al resto de manera que pueda seguir funcionando aunque otro componente no se encuentre disponible.

Para alcanzar los objetivos que se hayan definido en el sistema, los componentes deber realizar diversas tareas de manera independiente y servirse concurrentemente de los recursos que tienen a su disposición. La comunicación entre componentes obliga a que exista un intercambio de información de forma coordinada entre los mismos. Este intercambio de información se debe realizar a través de los medios de comunicación que el sistema distribuido proporciona a cada componente y además implicará una ubicación espacial de los componentes en los correspondientes nodos de procesamiento. El acceso de los componentes a los medios de comunicación disponibles obliga a la existencia de uno o varios protocolos de comunicaciones y de las interfaces que forzosamente deberán conocer los componentes que se comuniquen.

El intercambio de información entre componentes supondrá que deberá existir una forma de enviar la información que sea comprensible por los componentes implicados en el intercambio. Consecuentemente el contenido de los mensajes deberá disponer de un lenguaje y de unas normas conocidas por los agentes que intervengan en el proceso de intercambio de información. La existencia de un lenguaje, supone la existencia de una gramática y consiguiente de una sintaxis. Estos aspectos del lenguaje de intercambio de información deberán ser tenidos en cuenta a la hora de diseñar un sistema distribuido y de intentar lograr un alto grado de estandarización del mismo [FIPA, 1997].

2.3.1.2 Características de los sistemas distribuidos

Entre las propiedades deseables de un sistema distribuido destacan las siguientes [Coulouris et al., 2001].

- Heterogeneidad. Se entiende como la variedad y la diferenciación de los componentes de un sistema distribuido. La variación de los diversos componentes de un sistema justifica la creación de estándares que permitan a los componentes conocer las reglas de funcionamiento.
- Extensibilidad. Este concepto se refiere a la compatibilidad que el sistema puede ofrecer a los nuevos componentes. La extensibilidad del sistema también justifica el uso de estándares. La extensibilidad de un sistema también se conoce como apertura.
- Seguridad. Es una de las propiedades que proporciona mayor valía a un sistema distribuido. Hay tres aspectos que son importantes en cuanto a definir la seguridad de un sistema: disponibilidad, confidencialidad e integridad. La disponibilidad se refiere a que los recursos del sistema están accesibles para aquellos componentes que lo requieran, en el instante en que sea necesario. Confidencialidad es la protección que el sistema proporciona para no ser accesible por componentes no autorizados. Asimismo la confidencialidad debe proporcionar la suficiente seguridad a un componente, que los componentes con los que está trabajando son realmente los que él requería para trabajar, es decir, además de evitar la intrusión, un sistema confiable debe proporcionar suficientes garantías para evitar la suplantación. Finalmente la integridad se refiere a la protección que el sistema suministra contra la alteración de los datos, ya sea accidental o provocada.

- Escalabilidad. Un sistema distribuido es escalable en la medida en que se pueden aumentar sus componentes sin que suponga una pérdida de efectividad debida a los cambios. El objetivo final de un sistema, con respecto a la escalabilidad, es que no se deban realizar cambios en el sistema pese al aumento de los componentes.
- Soporte a fallos. Un fallo en uno o varios componentes hará que el resultado del sistema sea diferente o de menor calidad que el esperado, por ello la prevención y tratamiento de los fallos es un aspecto que deberá tener muy en cuenta un sistema distribuido. Algunas características que deben tener los sistemas distribuidos ante la aparición de un fallo son: la detección, el enmascaramiento, la tolerancia y la recuperación. La detección del fallo debe realizarse lo antes posible para evitar que los errores en el procesamiento tengan consecuencias colaterales en el resto de los componentes. Una vez detectado el fallo, éste debe enmascararse o atenuarse, es decir, que aunque el fallo es conocido los componentes no se ven afectados. La tolerancia a fallos implica que los componentes conozcan el fallo y reaccionen frente al mismo de una manera conocida y controlada. Finalmente, la recuperación es la fase en la que el sistema, una vez solucionado el fallo, vuelve a funcionar según los requisitos.
- Concurrencia. La concurrencia es la capacidad que tiene un sistema para que un recurso pueda ser utilizado simultáneamente. La simultaneidad puede no ser completa, sino a efectos de los componentes que acceden al recurso. La concurrencia supondrá el uso de diversas estrategias de acceso al recurso con el objetivo de mantener la integridad de la información.
- Transparencia. La transparencia de un sistema es la ocultación al usuario de los detalles que son propios de la gestión interna del sistema. De esta manera, los usuarios de un sistema no lo perciben con más detalles que el estrictamente necesario. La transparencia se logra de diversas maneras, pero fundamentalmente a partir del encapsulado de componentes y del uso de interfaces.

A medida que un sistema distribuido de soporte a las propiedades anteriores, sus componentes y el sistema en su conjunto verán aumentada su eficiencia e incrementado su valor.

2.3.2 Paradigmas de comunicación

Se entiende como un paradigma la ordenación formal de los elementos que componen un sistema. En el caso de las comunicaciones, los paradigmas describen las conexiones de las comunicaciones. Desde este punto de vista se pueden distinguir diversos modelos, tal como se describe en [Liu, 2004].

En la Figura 10, se puede observar la clasificación de los paradigmas de comunicaciones. Como se puede observar hay tres paradigmas, en función del papel que toman cada uno de los componentes de la comunicación. Estos son, de menor a mayor complejidad, el modelo de paso de mensajes, el modelo cliente-servidor y los sistemas de mensajería.

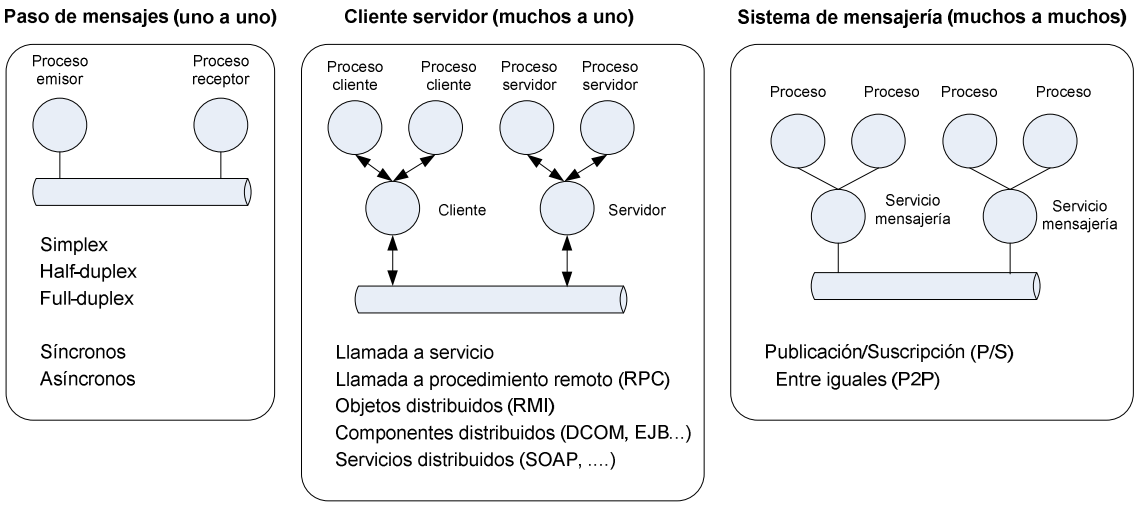


Figura 10. Paradigmas de comunicación en función de los componentes involucrados.

El modelo de paso de mensajes es el primero cronológicamente en aparecer. Es el más sencillo y sienta las bases para los modelos siguientes. En este paradigma es donde aparecen los conceptos principales en función del sentido de la comunicación y la simultaneidad de la misma (simplex, half duplex y full duplex). También aparece el concepto de mensajes síncronos, cuando existe un mecanismo para interconectar los relojes de los componentes y asíncrono cuando cada componente mantiene su información temporal. En el paradigma de paso de mensajes la gestión de las comunicaciones es responsabilidad de los procesos involucrados, lo que disminuye la eficiencia y carga al control con la responsabilidad de la gestión del protocolo de las comunicaciones.

El siguiente paradigma en aparecer es el modelo cliente-servidor, donde aparecen unos componentes responsables de ofrecer unos servicios de comunicaciones y otros componentes responsables de acceder a los servicios. Este modelo implica la especialización de los componentes, lo que añade un grado de complejidad al sistema, sin embargo aísla a los componentes de control de los detalles y complejidades de las comunicaciones.

Finalmente, aparece el modelo basado en sistemas de mensajería. Estos sistemas unifican cada nodo para actuar como cliente y servidor simultáneamente, lo que implica que en todos los nodos se debe tener un proceso especializado en la comunicación, que normalmente es común en todos los nodos. En este último caso se conoce como middleware el conjunto de procesos que intervienen en la gestión del sistema de mensajería [Gaddah and Kunz, 2003].

2.3.3 Revisión

La mayor parte de los componentes de los sistemas distribuidos precisan recibir datos de diferentes fuentes y distribuirlos a diferentes destinos. Por tanto, las arquitecturas de comunicaciones de los sistemas de control deberían emplear sistemas de comunicaciones orientados a distribuir datos, no sólo a invocar servicios remotamente como cliente/servidor o a realizar llamadas de funciones tal como hace RPC (Remote Procedure Call).

De entre los modelos de comunicaciones, uno de los métodos en los que se basan las comunicaciones, de manera más frecuente, es el uso de mensajes, también conocido como MOM (Message Oriented Middleware). Los servicios de mensajería proporcionan una comunicación asíncrona entre componentes de un sistema. Los servicios de mensajería ocultan el origen del mensaje, por lo que sirven tanto para intercambiar mensajes entre componentes de un mismo servidor como los mensajes que provienen de otros servidores.

Para que un servicio de mensajería pueda emplearse en un sistema distribuido, los nodos que se comunican deben estar programados para reconocer un formato de mensaje común. Actualmente se están desarrollando sistemas donde el contenido de los mensajes es código, con lo que los programas que se ejecutan en los servidores varían, no solo en la información con la que trabajan, sino en los algoritmos y conocimientos con los que procesan dicha información.

Las aplicaciones que emplean los servicios de mensajería para intercambiar información no precisan de un acoplamiento tan dependiente como en el paso de mensajes. El emisor y el receptor pueden cambiar o ser sustituidos sin que ello afecte a las aplicaciones. El servicio de mensajería es el responsable de organizar y gestionar los mensajes en las colas internas hasta que sean procesados por los componentes, así como de la gestión de la entrega y de la coherencia de los mismos.

En la Figura 11, se muestran las arquitecturas revisadas en función del año de aparición de la tecnología y del modelo principal de comunicaciones que emplean. Se observa cómo la evolución puede ser dentro de un mismo paradigma, como hacen NML (Neutral Messaging Language), IPC (Inter-Process Toolkit) y RTC (Real-Time Communications) que se basan todos ellos en el paso de mensajes de TCX (TCA eXchange) [Gowdy, 2000], o cambiar la tecnología aunque la funcionalidad sea similar, como el caso de Microsoft con DDE (Dynamic Data Exchange), OLE, COM y DCOM (Distributed COM). En otras tecnologías se ha preferido no cambiarla y mantener la funcionalidad, es el caso de CORBA (Common Object Resource Broker Architecture) y DDS, ambos propuestos por la OMG.

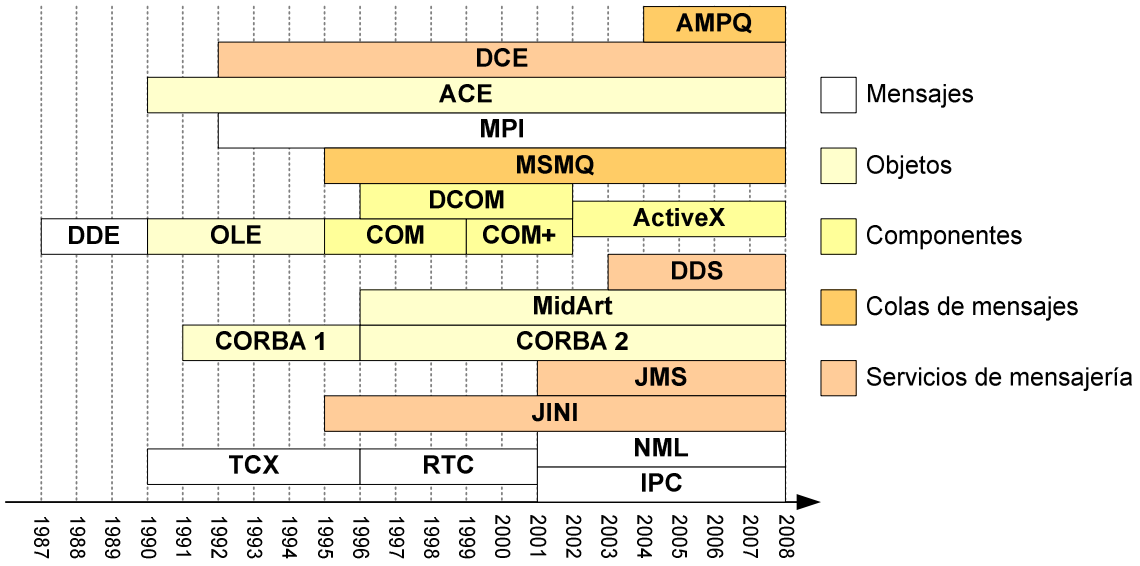


Figura 11. Ubicación de las diferentes tecnologías en función del paradigma que implementan.

En la Figura 11, se observa cómo los modelos de comunicación están relacionados con los modelos de programación empleados. Hasta la generalización de la programación orientada a objetos, el modelo empleado se basaba en librerías que proporcionaban funciones de comunicación basadas en mensajería, es el caso de TCX, DDE o MPI (Message Passing Interface). Además, el modelo de cliente-servidor hacía que los servidores de mensajería también tuviesen aparición como modelos iniciales. Son los casos de DCE (Distributed Computing Environment), y de JINI que aparece como servicio de mensajería de JAVA, simultáneamente a la aparición del lenguaje.

Con la generalización de la programación orientada a objetos, aparecen modelos de comunicación basados en dicho paradigma, como ACE o CORBA. La programación basada en componentes tiene su sistema de comunicaciones paradigmático con COM y DCOM de Microsoft. Como evolución de los modelos se tiene la combinación de los servicios de mensajería y objetos, que da lugar al modelo DDS donde se busca simplificar la programación de las comunicaciones, ofreciendo la misma potencia que los anteriores modelos. Finalmente, la evolución de los middleware da lugar a la aparición de capas intermedias de procesamiento como JMS, MSMQ o el modelo más reciente AMPQ.

2.3.4 Discusión

Los modelos vistos anteriormente cubren todas las características deseables en un sistema de comunicaciones que dé soporte a una arquitectura de control inteligente. Cabe destacar que algunos de los sistemas de comunicaciones permiten implementar diferentes paradigmas por lo que, por ejemplo, es posible implementar fácilmente un modelo cliente-servidor con un modelo de publicación-subscripción.

Todos los modelos deben realizar un intercambio de mensajes. Lo que diferencia a cada modelo es cómo gestiona los mensajes y qué componentes emplea para ocultar los detalles de la gestión. Los aspectos anteriores se agrupan bajo la denominación de middleware. En la capa más baja de procesamiento, los modelos emplean colas de mensajes y ocultan los detalles de la gestión a los usuarios.

En muchas ocasiones, conviene actuar directamente sobre las colas de mensajes, especialmente cuando se desea un comportamiento concreto de los aspectos temporales o del flujo de los mensajes. Por tanto, las implementaciones de los modelos deberán contemplar la gestión de las colas, bien sea dando acceso directamente a ellas o por medio de funciones de gestión.

En la Figura 12, se puede observar la ubicación de los principales modelos vistos anteriormente en función del soporte que ofrecen al tiempo real. Se puede apreciar cómo los sistemas de mensajería directa, que no implementan un paradigma de forma implícita, son los que ofrecen mayoritariamente soporte al tiempo real, mientras que los sistemas basados en servicios o colas de mensajes ofrecen un mínimo control temporal.

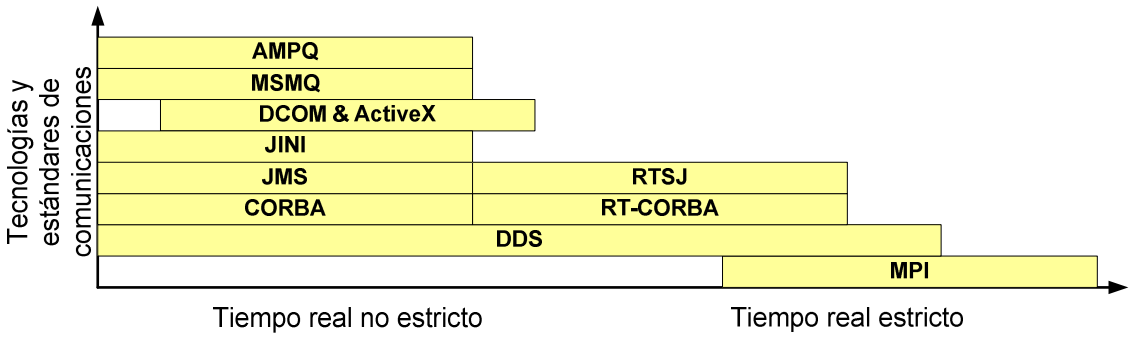


Figura 12. Ubicación diferentes tecnologías en función de los aspectos de tiempo real que cubren.

Cabe destacar que para los modelos JMS o CORBA existen implementaciones con control temporal y control de flujo. Estos modelos están desarrollados directamente para cubrir las carencias en estos aspectos que tienen los modelos iniciales. Sin embargo, hay algunas evoluciones como MidArt (propuesta de ampliación de CORBA) en el modelo de objetos o DDS en el modelo de mensajería que sí implementan un control de tiempo real estricto como característica destacable.

Analizando las características anteriores de los sistemas, se puede afirmar que las características más deseables de un sistema de comunicaciones son las siguientes.

- Soporte a diversos paradigmas de comunicaciones. Dependiendo de las circunstancias del sistema, puede ser más interesante emplear un paradigma u otro. Por ejemplo, ante ciertas restricciones temporales, el paso de mensajes puede ser más ventajoso que un sistema cliente-servidor.
- Gestión de las comunicaciones síncronas y asíncronas. Los componentes que empleen el sistema de comunicaciones deben poder comunicarse de forma asíncrona, o lo que es lo mismo a iniciativa del componente, o de forma síncrona a iniciativa del sistema de comunicaciones.
- Soporte a la organización y localización espacial de los componentes. Lo que implica dar soporte a algunas características como el agrupamiento de los componentes, la identificación de los mismos o la localización en el sistema.
- Configuración dinámica del sistema. Es importante que el sistema de comunicaciones permita variar la mayor cantidad de características posibles, sin necesidad de detener la actividad de los componentes que lo emplean.
- Descubrimiento del sistema. El sistema de comunicaciones debe ofrecer la posibilidad de informar de los cambios de configuración a los componentes que lo emplean, o informar de la configuración actual a los nuevos componentes.
- Control temporal y soporte a tiempo real. Los componentes deben conocer los aspectos temporales de los mensajes, como por ejemplo el tiempo que tarda un mensaje en llegar, o la capacidad de solicitar al sistema de comunicaciones la entrega de un mensaje en un periodo concreto de tiempo.
- Sencillez en la interfaz de comunicaciones, en componentes y en protocolos empleados. En algunos modelos de comunicación, la complejidad de estos aspectos ha provocado que su uso no se extienda pese a ser modelos aparentemente válidos.

- Parametrización. El aspecto del comportamiento del sistema en función de unos parámetros es importante puesto que permite sintonizar las comunicaciones con los requisitos de los componentes. Además, la parametrización es necesaria para monitorizar el funcionamiento del sistema.

Para la configuración del comportamiento del sistema de comunicaciones en función de los requisitos de los clientes aparece el concepto de calidad de servicio (QoS). A continuación se analizará el papel que la calidad de servicio toma en la configuración de las comunicaciones para poder reconocer qué características son relevantes en la definición de parámetros de comportamiento del sistema de comunicaciones.

2.4 El papel de la calidad de servicio

2.4.1 Definición

La calidad se define como un conjunto de propiedades inherentes a algo, que permiten juzgar su valor. Por tanto, la calidad de servicio es un concepto que trata de definir las propiedades por medio de unos parámetros por los que evaluar un servicio ofrecido. De entre las diversas definiciones de calidad de servicio que se localizan en el campo de los sistemas de comunicaciones destacan las siguientes.

- Efecto global de las prestaciones de un servicio que determinan el grado de satisfacción de un usuario al utilizar dicho servicio [ITU, 1994]
- Conjunto de las características, tanto cuantitativas como cualitativas, de un sistema distribuido necesarias para alcanzar las funcionalidades requeridas por una aplicación [Vogel et al., 1995].
- Conjunto de requisitos del servicio que debe cumplir la red en el transporte de un flujo [Crawley et al., 1998].

La medición de la calidad de servicio se realiza por medio de los llamados parámetros de calidad de servicio. Los parámetros son las variables que, en una familia de elementos, sirven para identificar cada uno de ellos mediante su valor numérico. En el campo de las comunicaciones no hay un consenso sobre el conjunto de parámetros que pueden emplearse para determinar si un servicio cumple con unos requisitos de calidad definidos. Por ello, es conveniente revisar distintos autores de distintos ámbitos para delimitar los más habituales y posteriormente definirlos. Cabe destacar que en la RFC 2330 [Paxson et al., 1998] se da una visión en la que se diferencia la composición de parámetros entre una composición espacial de métricas empíricas y una composición temporal. Esta visión permite clasificar los parámetros, ubicándolos en el ámbito espacial (gestión del flujo de mensajes, localización de componentes, etc.) o en el ámbito temporal (cumplimiento de plazos, tiempo de validez de un mensaje, etc.).

Los parámetros de calidad de servicio se emplean para describir características del servicio como el ancho de banda o el plazo temporal que se requiere de los mensajes, por medio de la lectura del mismo. Además los parámetros se emplean para configurar el servicio, dentro de las posibilidades que el servicio ofrezca, por ejemplo estableciendo el ancho de banda o el plazo temporal requerido del servicio. Cuando la configuración de diversos parámetros se realiza por medio de funciones que además negocian y comprueban la coherencia de la configuración de los parámetros, se habla de políticas de calidad de servicio.

Es habitual que los parámetros de calidad de servicio y las políticas que las gestionan, sean específicos a los sistemas en los que se aplican. Por ejemplo, en el ámbito de las redes de comunicaciones la calidad de servicio se entiende en términos como la capacidad de asegurar una tasa de datos en la red o ancho de banda, un retardo de los mensajes o una variación del retardo o *jitter*. Sin embargo, en el ámbito del control la calidad del servicio se entiende en términos de tiempos de computación o de reacción del sistema ante eventos.

Dada la importancia que los parámetros de calidad de servicio tienen, es beneficioso que en los sistemas distribuidos de control inteligente, se permita al sistema añadir información a los elementos de control del sistema, como los agentes componentes de control o a los elementos de comunicaciones, que determine unas configuraciones de calidad necesarias para la consecución de los objetivos definidos.

Para conocer la funcionalidad de los parámetros de calidad de servicio, a continuación se realiza una revisión de cómo los definen los distintos sistemas que los emplean: sistemas basados en las colas de mensajes, sistemas distribuidos y entornos middleware.

2.4.2 Parámetros de los sistemas de colas de mensajes

Las colas de mensajes constituyen la base de un sistema de mensajería, por lo que la aproximación mínima a la parametrización de las comunicaciones es la que se da desde la teoría de colas. La bibliografía es amplia, aunque una visión orientada al análisis de sistemas se da en [Stuck and Arthurs, 1984] donde los parámetros de las colas se emplean para determinar el valor del rendimiento, considerándose este último como el parámetro más relevante. Los parámetros de las colas de mensajes son históricamente los primeros parámetros de rendimiento que aparecen en los sistemas de comunicaciones. Conocer qué indicadores se tienen y cómo se definen es importante como punto de partida para conocer cómo obtener el resto de parámetros. A continuación se describirán brevemente los parámetros más importantes de las colas de mensajes.

- Tiempo de respuesta. Representa el tiempo total en procesar un mensaje en la cola, desde que llega hasta que finaliza su procesamiento. Como parámetro puede ser instantáneo a un mensaje concreto o bien calculado como el promedio de los tiempos en un intervalo de tiempo concreto.
- Demanda de servicio. Representa la cantidad de solicitudes del servicio en función de un intervalo de tiempo.
- Tasa de servicio. La tasa de servicio representa el promedio de mensajes servidos en un intervalo de tiempo por parte de la cola de mensajes.
- Carga. La carga es un parámetro referido a la cantidad de mensajes que en un instante de tiempo, o durante un periodo, hacen uso de la cola. La carga representa la utilización que se está haciendo de la cola. En algunos casos se le conoce también como tráfico. Desde un punto de vista del usuario, la carga se puede ver como la demanda que se tiene de la cola de mensajes.
- Productividad. En los sistemas de colas, la productividad se refiere a la eficiencia en el uso. En las colas la productividad tiene una relación directa con la utilización. Formalmente la productividad se calcula a partir de las probabilidades de carga de los componentes o lo que es lo mismo la disponibilidad de la cola.

En las colas de mensajes, los parámetros anteriores definen una serie de ecuaciones como la condición de estabilidad o las leyes de Little que relacionan los promedios de números de mensajes con los promedios temporales de las colas [Averill and Kelton, 2000].

2.4.3 Parámetros de los sistemas distribuidos

Debido a que los sistemas de comunicaciones son modelables mediante colas, los parámetros empleados con los sistemas de colas de mensajes pueden extenderse fácilmente a los sistemas de comunicaciones. Dependiendo de la bibliografía y del ámbito que se trate, la denominación del parámetro varía. Para este estudio se han contemplado las definiciones de colas de mensajes revisadas en [Stuck and Arthurs, 1984], las definiciones de calidad de los computadores [Jain, 1991] y las definiciones de calidad de los sistemas distribuidos [Coulouris, 2001]. En la Figura 13 pueden contemplarse las relaciones entre los diferentes parámetros.

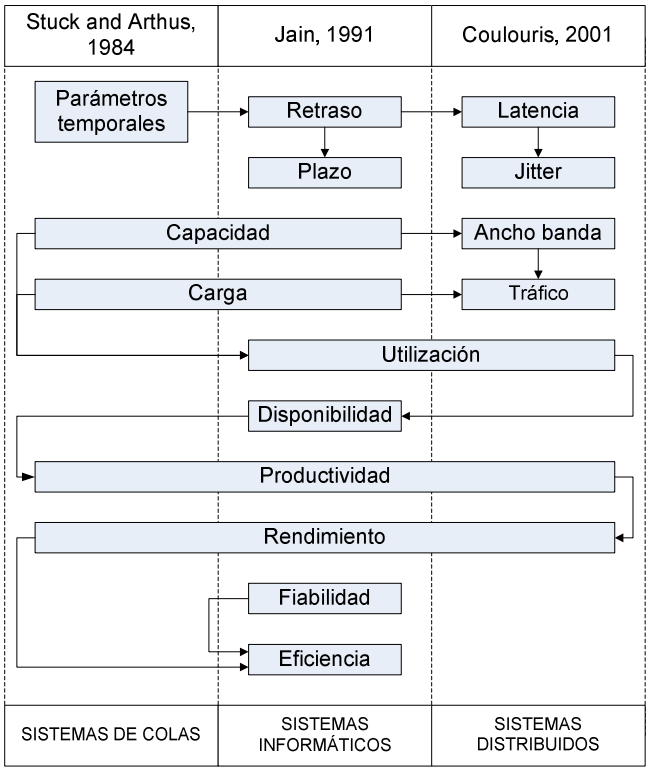


Figura 13 Parámetros de calidad de servicio en función del ámbito en el que se aplican

En las colas de mensajes se contemplan como parámetros temporales el tiempo de espera (retardo que se tiene al acceder a un servicio, desde que se solicita el acceso) o el tiempo de servicio. En [Jain, 1991] se definen diversos tiempos como el tiempo de respuesta, tiempo de reacción, tiempo de operación, tiempo de servicio o tiempo de ocio. Estos parámetros permiten controlar temporalmente cualquier proceso en un sistema. Los más importantes son el retraso de un mensaje, que es inherente al sistema de comunicaciones, y el plazo temporal que es un parámetro que determina el cliente para ser cumplido por el retraso. En lo que a parámetros temporales, en los sistemas distribuidos se habla de latencia como el tiempo total de respuesta de un servicio y el *jitter* como la desviación del cumplimiento de los requisitos temporales del mensaje.

Cuando se trata de medir características del servicio, en las colas de mensajes de habla de capacidad como la cantidad de peticiones que un servicio es capaz de atender, mientras que en las comunicaciones se emplea el concepto de ancho de banda como la capacidad de transmitir las solicitudes de servicio. Dependiendo de la capacidad y de la demanda que se tenga de un servicio se pasa a definir la carga del servicio, que desde un punto de vista de comunicaciones se conoce como tráfico. La carga de un servicio, en función de la capacidad del mismo determina la utilización. Es habitual que la utilización se emplee desde un punto de vista relativo mientras que carga y capacidad son parámetros absolutos. Como se verá más adelante un servicio puede estar sobrecargado, pero no "sobreutilizado". El uso que se hace de un servicio determina la disponibilidad del mismo, de tal forma que una utilización del 100% no lo deja disponible. La disponibilidad va más allá de la utilización ya que para su cálculo es habitual tener en cuenta tanto el tiempo de que el servicio no puede usarse por estar sirviendo normalmente como el tiempo que el servicio no puede usarse por haber tenido un fallo.

En función de la disponibilidad y de los recursos del sistema se obtienen los dos parámetros más empleados para la medición de la calidad de servicio: la productividad y el rendimiento. Estos dos parámetros son muy similares, de manera que es habitual considerarlos como sinónimos. Sin embargo, desde un punto de vista formal, la productividad tiene en cuenta el resultado, mientras que en el rendimiento es habitual que, además del resultado, se tenga en cuenta los recursos empleados.

La fiabilidad se define en función de los fallos en el servicio, de forma que un sistema sin errores es un sistema fiable por completo. A partir de la fiabilidad y del rendimiento es como se obtiene la eficiencia de un sistema de tal manera que un sistema que ha alcanzado la máxima eficiencia es un sistema optimizado.

En la Figura 13 se observa la evolución de los principales parámetros en función del tipo de sistema sobre el que se aplican. Es destacable la variación de la nomenclatura entre distintos entornos. Por ejemplo, en los sistemas distribuidos se habla de latencia en lugar de retrasos y de tráfico en lugar de la carga. Esto no es incongruente, ya que en los sistemas distribuidos, además de los parámetros de calidad de servicio propios de los nodos (sistemas informáticos y de colas a todos los efectos) se debe añadir la medición de los parámetros propios de la red. Debido a que los sistemas distribuidos de control inteligente aúnan conceptos de los sistemas distribuidos y de los sistemas informáticos, es conveniente definir de forma conjunta los parámetros. Seguidamente se proponen las definiciones más detalladas obtenidas a partir del análisis de los parámetros de calidad de los diferentes entornos expuestos:

- Tiempo de espera *(waiting time)*. El tiempo de espera es uno de los parámetros basados en los sistemas de colas. Se calcula siempre en función de uno de los aspectos del sistema. Por ejemplo, en el caso del procesamiento se suele referir al tiempo que se tarda entre que se solicita un procesamiento y se obtiene el resultado. En el caso de las comunicaciones suele hablarse del tiempo que tarda la respuesta en llegar ante el envío de un mensaje. A un nivel más alto, en los sistemas de control se puede tener diferentes visiones dependiendo de lo que se deba esperar, en el caso del resultado de una acción se puede hablar del tiempo de espera como el tiempo transcurrido entre que se recibe un cambio en la entrada de información, se procesa la entrada, se toma una decisión y se actúa sobre la salida de información. Como parámetro se considera una magnitud de dimensión temporal.

- Retardo *(delay)*. Tiempo que transcurre entre el instante de tiempo esperado y el instante real en el que sucede un acontecimiento. El retraso tiene muchos significados pero, cuando se enmarca en el contexto de la calidad de servicio, se suele relacionar con alguna de las características sobre las que se trabaja. Por ejemplo, en computación el retardo de procesamiento es el tiempo que transcurre entre que un componente recibe un mensaje, el mensaje es procesado y se realiza la acción correspondiente. Desde el punto de vista de las comunicaciones se entiende como retraso el tiempo que se tarda desde que un dispositivo recibe una trama hasta que la trama es retransmitida hacia el destino correspondiente. Se suele hablar también de *retraso de serialización*, como el tiempo que se tarda en transmitir un paquete o trama. Y retraso de extremo a extremo (*end to end*) como el retraso total que los paquetes experimentan desde la fuente al destino, es decir, desde que inician su trayecto hasta que llegan al destino final.
- Plazo temporal *(deadline)*. Aunque no es considerado como factor de calidad de servicio propiamente dicho, es el cumplimiento del *deadline* lo que debe considerarse como un factor de calidad de servicio. Para comprobar si un sistema cumple los plazos temporales, se deben cumplir los retrasos correspondientes de transmisión y de procesamiento.
- Latencia (*latency*). Tiempo que transcurre entre que se da una acción y se produce la respuesta. En el caso de las comunicaciones es habitual calcularlo como el tiempo transcurrido mientras una unidad de datos entra en el componente correspondiente y lo abandona. En ocasiones se trata como sinónimo de *retraso*, aunque formalmente sean distintos conceptos ya que el retraso implica una desviación sobre una expectativa de tiempo, mientras que la latencia es un valor proporcionado por el sistema.
- Inestabilidad (*jitter*). Se define como la medición de las variaciones en función de una referencia. En el ámbito de las redes se habla de un cambio o variación de la latencia. También se considera la variación del retardo. En [Demichelis and Chimento, 2002] se propone emplear PDV (Packet Delay Variation) como parámetro de calidad de servicio.
- Capacidad (*capacity*). La capacidad suele entenderse como un máximo alcanzable en un sistema. En [Jain, 1991] se diferencia la capacidad nominal (máximo rendimiento) y la capacidad de uso (máximo rendimiento para un tiempo de respuesta dado). Esta diferenciación es importante, ya que la capacidad de uso es la medida más realista sobre la capacidad que puede proporcionar el sistema que cumpla los requisitos temporales que se le exigen al servicio.
- Ancho de banda (*bandwidth*). Se define como la capacidad de producción o rendimiento prorrateada de una red, un medio de comunicación o protocolo de comunicaciones. En el caso de la calidad de servicio, se entiende como el reparto del ancho de banda, ya que como calidad de servicio no se tiene la capacidad de acceder al medio. En ocasiones se llama ancho de banda a la velocidad o al rendimiento de un medio de comunicación.
- Carga (*load*). La carga se refiere a la cantidad de trabajo que un componente tiene en un instante concreto o para un intervalo de tiempo. Como medida proporciona una estimación tanto de la utilización como de la capacidad.

- Tráfico (*trafic*). Es un término orientado a las comunicaciones y se entiende como la carga pero desde un punto de vista dinámico, es decir, la carga a lo largo de un tiempo en función de la unidad de medida de la cantidad de información.
- Utilización (*utilization*). Se define como la tasa de empleo o uso de algo. Se calcula como el porcentaje de tiempo durante el cual un servicio (o recurso) está ocupado. En el ámbito de las comunicaciones se emplea para localizar zonas de congestión y cuellos de botella a partir de valores elevados de utilización de componentes.
- Productividad (*throughput*). La productividad es uno de los parámetros más empleados y con más interpretaciones. Se define como la capacidad o el grado de producción por unidad de trabajo y suele calcularse como la relación entre lo producido y los medios empleados o, si se desea obtener en función del tiempo, se calcula como la cantidad de trabajo realizado en un periodo de tiempo. En el ámbito de las comunicaciones, suele hablarse de la cantidad de información que una red es capaz de entregar durante un intervalo de tiempo. También se encuentra definido como rendimiento (*performance*). Formalmente el rendimiento se define como la proporción entre el producto o el resultado obtenido y los medios utilizados. En ocasiones se expresa el rendimiento como un porcentaje de la productividad.
- Disponibilidad (*availability*). La disponibilidad se define como la cualidad o condición de poder disponer libremente de un servicio. Se suele calcular como el tiempo mínimo que se asegura que el servicio estará en funcionamiento, en ocasiones se calcula como el tanto por cien del tiempo en el que se puede obtener el servicio si éste se solicita. También se plantea en términos de probabilidad de lograr el servicio.
- Fiabilidad (*reliability*). Se define como cualidad de que un servicio (o recurso) ofrezca seguridad o buenos resultados. Generalmente se calcula como la probabilidad del buen funcionamiento de algo. Para el caso de las comunicaciones, se suele concretar al campo sobre el que se aplica. Formalmente se proporciona como la probabilidad del buen funcionamiento de un elemento.
- Eficiencia (*efficiency*). Es un término muy similar al de rendimiento, y en muchas ocasiones suelen utilizarse indistintamente. En [Jain, 1991] se define como la ratio entre el máximo rendimiento obtenible (capacidad de uso) y la capacidad del sistema (capacidad nominal). La eficiencia se refiere especialmente a la optimización de los recursos empleados, y generalmente se diferencia del rendimiento en que este último se expresa en función de una unidad, generalmente temporal, mientras que la eficiencia se proporciona como una tasa.

2.4.4 Parámetros de los entornos middleware

Para realizar una conexión transparente, ocultando los detalles de los sistemas de comunicaciones a las aplicaciones de un sistema distribuido, aparece el concepto de middleware. El concepto de middleware es muy amplio, aunque comúnmente se entiende como la capa intermedia entre la aplicación y el sistema de comunicaciones que facilita la conexión entre ambas capas [Gaddah and Kunz, 2003].

La variedad de arquitecturas e implementaciones consideradas como middleware es muy amplia, por lo que el concepto de calidad de servicio, en éste ámbito, también lo es.

En un middleware la calidad de servicio es un objetivo, por lo que la arquitectura debe proveer funciones que permitan gestionar las comunicaciones y lograr así un cierto nivel de eficiencia. La medición del nivel de calidad de servicio puede hacerse por medio de los parámetros de calidad de servicio. Sin embargo, cada middleware proporciona unas funciones específicas, lo que implica que la gestión de la calidad de servicio es muy dependiente del middleware que se utiliza. A continuación se revisará el soporte a la calidad de servicio que ofrecen los principales entornos middleware

2.4.4.1 JMS

El sistema de mensajería de Java (JMS) no especifica directamente la calidad de servicio. Sin embargo diversas implementaciones basadas en JMS, cambiando los atributos de los componentes o de los mensajes pueden gestionar diversos atributos a partir de los cuales controlar la calidad del servicio [Sun. 2002].

- Prioridad. Por medio del atributo *priority* se puede dar una prioridad a la comunicación entre unos nodos o a los mensajes que se envían o reciben sobre otros mensajes.
- Persistencia. A través del atributo *deliveryMode* se permite variar entre el modo *persistente*, donde el mensaje siempre es almacenado y está disponible, y el modo *no persistente* donde el mensaje no se almacena.
- Duración. Por medio del atributo *timeToLive* se especifica el tiempo en milisegundos en los que un mensaje estará disponible para ser solicitado por otro componente.
- Plazo temporal. Se puede especificar el *timeout* que pueden tener los mensajes en cualquier localización de la comunicación, es decir tanto del solicitante, como del receptor o de una sola transacción.
- Tipo de conexión. Especificada en el atributo *transacted*, es una forma de tratar la conexión de la comunicación. En el modo *InOnly* los mensajes circulan en un solo sentido, lo que se considera un evento. En el modo *InOut*, los mensajes circulan en los dos sentidos, es decir hay una confirmación por cada mensaje enviado o recibido.
- Estampación de tiempos en mensajes. Por medio del atributo *messageTimestampEnabled* se almacena en cada mensaje el instante de tiempo en que es generado, lo que permite disponer de un control temporal de las comunicaciones a nivel de mensaje.

2.4.4.2 CORBA

Common Object Request Broker Architecture (CORBA) define políticas de calidad de servicio y funciones para la negociación de la misma [OMG, 2001]. Las políticas de calidad de servicio, agrupadas por los ámbitos en que actúan, son las siguientes.

- Políticas de reconexión. Únicamente se define la política *RebindPolicy* que especifica cómo actuar ante la pérdida de conexión. Los modos en que se puede gestionar son: reconexión transparente, a petición o sin reconexión.

- Políticas de sincronización. Al igual que en el caso anterior sólo se define una política: *SyncScopePolicy*, que especifica el alcance de la sincronización. Los alcances que se pueden configurar son: ninguna sincronización, sincronización con el servidor, con el destinatario o por mensaje.
- Políticas de prioridad. Las políticas de prioridad especifican la prioridad de la conexión.
- Políticas de tiempos de espera. Las políticas de tiempo de espera especifican los tiempos que esperarán los componentes tanto para la conexión entre ellos como para los mensajes.
- Políticas de enrutamiento. Definen cómo actuar con los mensajes en los puntos de interconexión. Se define cómo actuar ante un mensaje que hay que redirigir y cuántas veces puede ser redirigido entre componentes.
- Políticas de orden de entrega. Especifica el orden por medio del cual se pasan los mensajes a los componentes.

2.4.4.3 FIPA

En el modelo de la Foundation for Intelligent Physical Agents (FIPA) la calidad de servicio se considera una cuestión opcional, por lo que se deja como responsabilidad del programador desarrollar las funciones para emplear los parámetros para gestionar las políticas basadas en la calidad de servicio que gestionen la comunicación entre agentes. Sin embargo, en [FIPA, 2002] se realiza una propuesta de los siguientes parámetros de QoS como relevantes en la comunicación:

- Ratio de la línea (parámetro *line-rate*). Se considera como el ancho de banda en una sola dirección sobre la conexión entre dos agentes.
- Rendimiento (parámetros *throughput* y *throughput-std.dev*). Rendimiento y desviación estándar del rendimiento, considerados como el número de bits de usuario transmitidos correctamente en una dirección para un intervalo de tiempo.
- Tiempo del recorrido completo (parámetros *rtt* y *rtt-std-dev*). Tiempo requerido, y desviación estándar, para enviar un dato y que se reciba la confirmación de llegada correcta del mismo.
- Retraso (parámetros *delay* y *delay-std-dev*). Retraso, y desviación estándar, considerado como el tiempo nominal para que un segmento de datos se transmita a través de la conexión.
- Tiempo de medio de conexión (parámetro *mean-up-time*). Tiempo de funcionamiento correcto previsto para una conexión, expresa la capacidad de que algo funcione sin interrupciones.
- Tasa de omisión y tasa de errores de segmentos (parámetro *omission-rate* y *frame-error-rate*). Probabilidad de que un segmento de datos no sea transmitido correctamente a través de un enlace.
- Tasa de bits erróneos (parámetro *ber*). Tasa de bits erróneos del total de bits transmitidos en un intervalo.
- Retraso de establecimiento de conexión (parámetro *conn-setup-delay*). Tiempo muestreado que se tarda en establecer la comunicación entre entidades.

- Probabilidad de error en la conexión. (parámetro *conn-setup-failure-prob*): Ratio del total de llamadas atendidas con la ratio del total de llamadas fallidas con el total de las llamadas intentadas en una población (de agentes) determinada.
- Estado (parámetro *status*). Estado de la conexión: conectado, desconectado o conectando.

2.4.4.4 DCPS

En el modelo Data Centric Publish Subscribe (DCPS) perteneciente al estándar Data Distribution Service (DDS) del Object Management Group (OMG), las políticas de calidad de servicio se implementan como una lista de calidades de servicio que debe cumplir el componente al que se asocie. Todos los componentes de un sistema de comunicaciones pueden tener un conjunto de calidades de servicio asociadas.

El servicio será quien negocie y determinará si los requisitos de calidad entre componentes son compatibles. Excepto en unos casos concretos, los servicios ya negociados pueden ser cambiados aun ya establecida la conexión.

Para mejor entendimiento, se ha decidido organizar las políticas de QoS en grupos, atendiendo a la funcionalidad que ofrecen o el ámbito de la comunicación en el que actúan, en concreto se puede hablar de cuatro grupos principalmente: gestión de meta datos, gestión de los aspectos temporales, gestión del flujo de mensajes y gestión de los componentes DDS.

La gestión de los meta datos consiste en proveer a los componentes de la capacidad de enviarse mensajes entre sí con información adicional que no se puedan enviar por medio del resto de las políticas de calidad de servicio. No hay limitaciones en cuanto al contenido se refiere.

El siguiente conjunto de políticas de calidad de servicio agrupa aquellas que gestionan los aspectos temporales de las comunicaciones.

- Durabilidad (*Durability*). La durabilidad es una política referida al tiempo que los datos deben permanecer en el componente. Aunque es una política de ámbito temporal se gestiona por medio del número de muestras de mensajes que se deben mantener en cada componente. Por medio de esta política se puede mantener un historial de mensajes.
- Plazo temporal (*Deadline*). Es la política por medio de la cual se especifica el tiempo límite de espera que define la validez de un mensaje desde el punto de vista del componente. Es decir, el plazo temporal no especifica el tiempo que debe tardar un mensaje, sino cuánto tiempo está el componente dispuesto a esperar la llegada de un mensaje.
- Latencia (*Latency*). La latencia especifica el tiempo máximo de retraso desde que el mensaje se escribe en el medio de comunicación hasta que el dato es leído por los receptores. En términos de comunicaciones es un plazo temporal dependiente del medio y no del componente que espera el mensaje.
- Vivacidad (*Liveliness*). Esta política de calidad de servicio es la responsable de indicar al sistema que los componentes se encuentran activos y con capacidad de gestionar mensajes. En el caso de DCPS el método empleado es el envío de mensajes de aviso cuando el componente no ha enviado mensajes en un intervalo de tiempo definido.

- Filtro temporal. (*Temporal filter*). Esta política de calidad de servicio especifica la separación mínima temporal que deben tener los mensajes consecutivos desde el punto de vista del componente. Es el parámetro complementario al *plazo temporal*, que define la separación temporal máxima que se permite entre mensajes.
- Vida útil (*Lifespan*). Por medio de esta política de calidad se especifica el tiempo que un dato puede ser considerado válido una vez ha llegado al componente. Una vez se cumple el tiempo de vida útil, el dato no puede ser proporcionado al resto de los componentes.

El siguiente grupo de políticas de calidad de servicio lo forman las que dan soporte a la gestión del flujo de mensajes entre los componentes. Las políticas agrupadas en éste ámbito son:

- Fiabilidad (*Reliability*). La fiabilidad es la política de calidad de servicio orientada a minimizar el impacto producido por la pérdida de mensajes. En DCPS se especifica si se deben entregar todos los mensajes del historial de mensajes enviados, de esta forma se pueden reenviar aquellos mensajes que se hayan perdido.
- Prioridad (*Transport priority*). La prioridad, entendida como política de calidad de servicio, especifica un orden en el tipo de mensajes. En DCPS se deja a la aplicación la determinación de qué valores son los prioritarios, así como la determinación del número de niveles de prioridad.
- Orden (*Destination order*). Como política de calidad de servicio define el orden en que se deben procesar los mensajes desde el punto de vista de los componentes. En el caso de DCPS se dan dos posibilidades, el orden de salida de los componentes, o el orden de llegada. Por medio de esta política de calidad de servicio se permite que los componentes puedan trabajar en redes multi-camino.
- Historial (*History*). El historial define el número de muestras que se deben mantener en cada componente y por tanto ser proporcionados al resto. La cota inferior de ésta política de calidad de servicio es uno, lo que implica que sólo el último mensaje es válido. La cota superior consiste en mantener todos los mensajes desde que se inició la sesión, lo que puede emplearse para trazar el sistema, aunque puede provocar problemas por los recursos consumidos.
- Límite de recursos (*Resource limits*). Esta política de calidad de servicio determina los límites de los recursos que el componente puede consumir. Las limitaciones se pueden dar en diversos ámbitos como el número de mensajes o el número de conexiones.
- Presentación (*Presentation*). La presentación se refiere a las características relativas a cómo se presentarán los datos a la aplicación final. Por medio de ésta política de calidad de servicio se gestionan características como el acceso en un orden concreto a los mensajes, por parte de los componentes.

Además de las mencionadas anteriormente, DCPS proporciona políticas de calidad de servicio orientadas a cómo se gestionan las ubicaciones de los componentes en el sistema. La más interesante es la que permite dividir el sistema en áreas aisladas, o particiones. La partición permite especificar conjuntos de componentes entre los que se envían mensajes a los que no pueden acceder los componentes ajenos a la partición.

2.4.4.5 Análisis

La calidad de servicio es una herramienta de gran utilidad para la evaluación y el ajuste en el funcionamiento del middleware de un sistema de comunicaciones. La evaluación por medio de parámetros facilita la cuantificación de la calidad de servicio, tanto la ofrecida por un sistema como la requerida por los componentes. Las relaciones existentes entre parámetros de calidad de servicio cuantitativos permiten afrontar la obtención de parámetros cualitativos, por lo que la calidad de servicio es válida como herramienta de apoyo a la evaluación y el funcionamiento de un sistema.

Para un mejor análisis de los middleware, en la Tabla 3 se ha organizado las áreas de comunicación que cubren las políticas de calidad de servicio en cada uno de los middleware.

Tabla 3. Revisión de los parámetros y políticas de QoS de los middleware genéricos.

Área QoS	Parámetro/Política	JMS	FIPA	CORBA	DCPS
Control del flujo de mensajes	Prioridad	Sí	--	Sí	Sí
	Persistencia	Sí	--	--	Sí
	Sincronización	--	--	Sí	--
	Enrutamiento	--	--	Sí	--
	Orden de entrega	--	--	Sí	Sí
	Tasas de errores (pérdidas)	--	Sí	--	Sí
Control temporal de los mensajes	Plazo temporal (Timeout)	Sí	--	Sí	Sí
	Plazo temporal (Deadline)	--	--	--	Sí
	Retraso (Delay)	--	Sí	--	Sí
	Control temporal (Latency)	--	Sí	--	Sí
	Control temporal (Time Stamp)	Sí	--	--	--
	Rendimiento (throughput)	--	Sí	--	Sí
	Velocidad (bandwidth)	--	Sí	--	--
Gestión de la conexión de los componentes	Tipo o modo de conexión	Sí	--	--	--
	Reconexión	--	--	Sí	--
	Retrasos	--	Sí	--	--
	Errores	--	Sí	--	--
	Estado	--	Sí	--	--
	Actividad (liveliness)	--	--	--	Sí
	Gestión topología	--	--	--	Sí
Soporte a meta-datos	Información adicional	--	--	--	Sí

En la Tabla 3 se observa cómo los middleware cubren, en mayor o menor medida, las áreas de la calidad de servicio. La variedad de parámetros de los modelos más comunes empleados en las comunicaciones de sistemas distribuidos indica que no hay un consenso sobre la gestión de la calidad de servicio. El hecho de que algunos sistemas (JMS o FIPA) definan parámetros y otros (CORBA o DDS) definan políticas ya indica una visión diferente. En la Figura 14 se muestra la ubicación, dentro de los componentes de un sistema distribuido, en la que actúa la QoS de los sistemas revisados anteriormente.

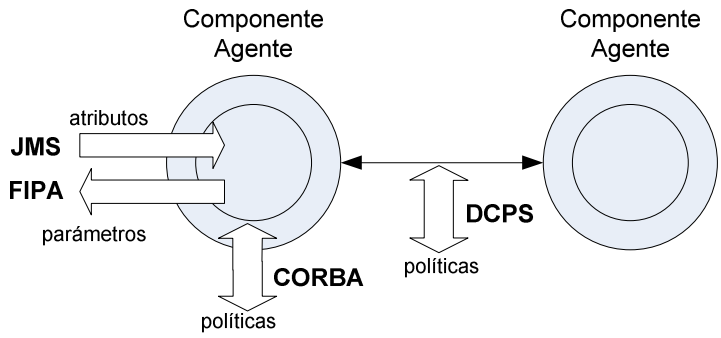


Figura 14. Ubicación dentro del sistema distribuido de la QoS de diversos middleware.

JMS está orientado a proporcionar los atributos del componente que permitan controlar la calidad de servicio del mismo, mientras que la propuesta de FIPA está más orientada a la obtención de los parámetros del agente. CORBA añade al control de FIPA las políticas de calidad de servicio, aunque los atributos sobre los que trabaja están más orientados a la conexión que a la comunicación en sí misma. Finalmente, DCPS está centrado en la gestión de la comunicación abarcando, por ello, todo el sistema distribuido.

2.4.5 Discusión

Respecto a la calidad de servicio, se observa cómo las colas de mensajes están presentes en todos los niveles, independientemente de la complejidad del modelo de comunicaciones. Las colas de mensajes proporcionan parámetros sencillos y cuantitativos, lo que permite parametrizar desde un nivel bajo de la calidad de servicio. Poder proporcionar estos parámetros a los niveles más altos de las comunicaciones permitirá parametrizar con una base cuantitativa la calidad de servicio. Asimismo, poder variar los atributos de las colas de mensajes permitirá un control a bajo nivel de las comunicaciones por parte de los niveles más altos. Éste control, aunque puede resultar complejo, proporciona una gran potencia al sistema de comunicaciones.

La mayor parte de los sistemas de comunicaciones coinciden en los mismos tipos de parámetros de calidad de servicio, generalmente centrados en el rendimiento y en la fiabilidad de la comunicación. Cabe destacar que, a diferencia de las colas de mensajes donde la calidad de servicio está perfectamente formulada y parametrizada de forma cuantitativa, en los sistemas de comunicaciones los parámetros de calidad de servicio se formulan de manera cualitativa, dejando a la implementación de las comunicaciones, por medio de un middleware, la formulación de la calidad de servicio en función de las características particulares del sistema de comunicaciones.

La calidad de servicio va tomando relevancia a medida que los organismos de estandarización o los fabricantes proporcionan una mayor información de las comunicaciones. En cuanto a los middleware se refiere, la calidad de servicio se emplea como parámetros o como políticas de calidad de servicio. Cuando se emplean políticas de calidad de servicio, éstas gestionan de forma activa el rendimiento de las comunicaciones.

A medida que crece la complejidad de un sistema, crece la complejidad de los parámetros que determinan su buen funcionamiento. Por ello, la correcta determinación de parámetros de calidad en función del objetivo en el que se aplican es un campo de investigación de gran interés.

2.5 Tendencias en el control y en las comunicaciones

2.5.1 Estándares de control y comunicaciones

En los anteriores apartados se han analizado las características del control y de las comunicaciones de los sistemas de control inteligente distribuidos. A partir de este análisis se ha determinado la importancia de la calidad de servicio en los middleware de comunicaciones y de la calidad de control en los sistemas basados en eventos (Figura 15) y suponen el punto de partida para el diseño de una arquitectura que integre el control y las comunicaciones

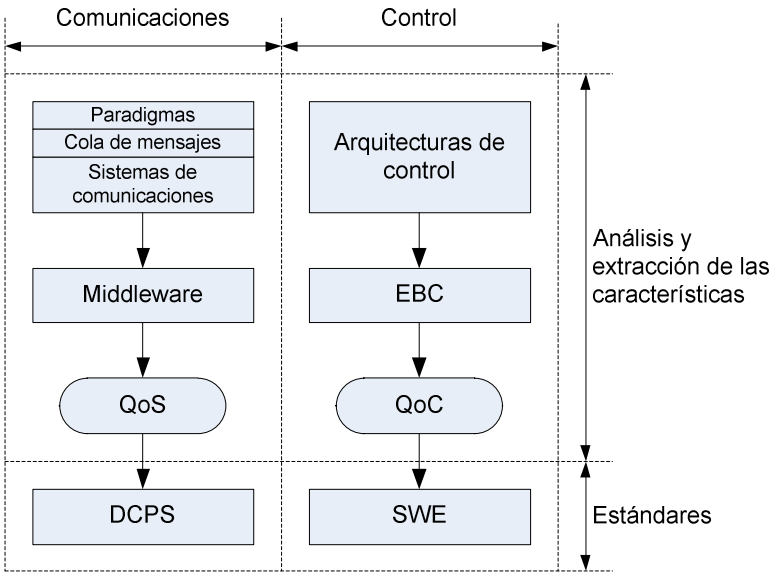


Figura 15. Tendencias en el control y en las comunicaciones.

Tal como se ha mostrado en los apartados anteriores, la estandarización no es una de las características más habituales en los diseños de las arquitecturas de control. Desde un punto de vista histórico, las arquitecturas se han diseñado en función del sistema a controlar, sin haberse centrado en buscar un modelo genérico que pueda ser empleado como base para otros sistemas y para estandarizar, en la medida de lo posible, las arquitecturas de los sistemas. El uso de estándares supone una ventaja importante a la hora de diseñar un sistema. Debido al trabajo previo realizado en el proceso de estandarización por parte de los diseñadores, el estándar proporciona una seguridad y una validez de los modelos empleados.

A diferencia del control, en el ámbito de las comunicaciones el uso de estándares está más extendido. Esto se debe a que los diseños de arquitecturas de control se han centrado más en los componentes de control que en los medios de transmisión empleados, lo que ha provocado la reutilización de un mismo medio convirtiéndolo en estándar de facto.

2.5.2 Control basado en eventos

El control basado en eventos o *Event Based Control* (EBC) es una aproximación del control donde los eventos son los que determinan la acción de control en lugar de que ésta esté determinada por un intervalo de tiempo constante, conocida como control por muestreo o *Time-driven based control approach* [Sánchez et al., 2009].

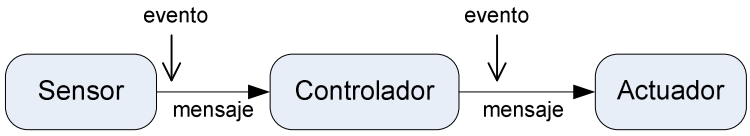


Figura 16. Sistema de control basado en eventos.

En el modelo EBC, los mensajes entre sensores, controladores y actuadores se envían únicamente cuando se produce una condición relevante (Figura 16) llamada evento. Las condiciones que generan eventos pueden ser muy diversas.

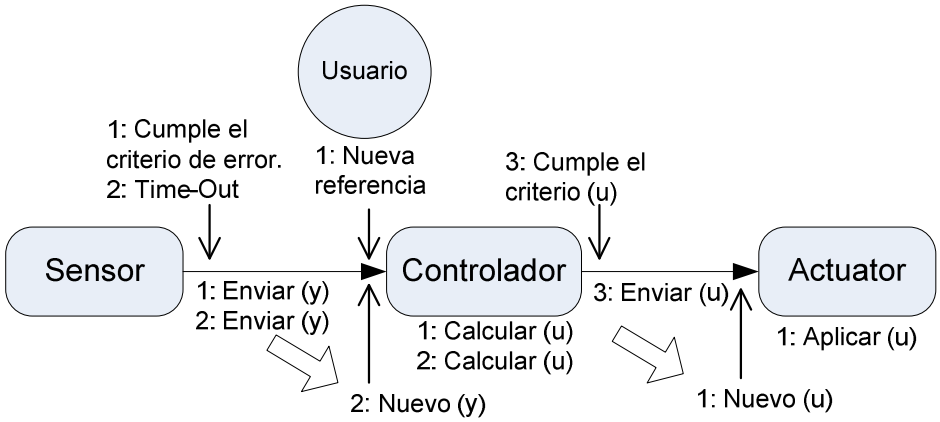


Figura 17. Desencadenantes de los eventos en el modelo EBC.

Por parte del sensor, los desencadenantes más habituales son la detección de un error con la referencia proporcionada o bien que ha transcurrido el tiempo de espera y por tanto hay que enviar un mensaje (Figura 17), en ambos casos se envía un mensaje con el valor actual del sensor (y). Los eventos que pueden llegar al controlador son dos: una nueva referencia por parte del usuario o bien un valor del sensor (que se ha enviado por cualquiera de las dos causas anteriormente descritas), en ambos casos el evento desencadena el cálculo de una nueva acción de control (u). El controlador debe comprobar si es necesario enviar la acción de control, ya que ésta puede ser la misma que se envió con anterioridad o bien puede no producir efecto, en caso de cumplir el criterio, se envía un mensaje que llegará al actuador para que procese la acción de control enviada.

El modelo EBC requiere de una infraestructura de comunicaciones basada en eventos, por lo que el control temporal de los mensajes y el control del flujo de los mismos deberán ser proporcionados por el middleware que se emplee en el sistema de control. Asimismo, se deberá proporcionar toda la infraestructura de filtrado de mensajes en función de la relevancia del envío que se pueda requerir.

2.5.3 Calidad de control

De la misma forma que para evaluar la eficiencia en las comunicaciones se emplean los parámetros de QoS, el control debe proporcionar los correspondientes parámetros, conocidos como parámetros de calidad de control o QoC (Quality of Control). Se considera un control perfecto cuando la señal que se envía al actuador hace que la señal medida por el sensor sea idéntica a una señal de referencia, por lo tanto no existe error entre la señal medida y la señal de referencia. El error de control se emplea para modificar la señal enviada al actuador.

Los parámetros de QoC más comunes son la integral del valor absoluto del error o IAE (Integrated Absolute Error) y la integral del tiempo por el valor absoluto del error o ITAE (Integrated in Time Absolute Error). Ambos parámetros permiten al sistema conocer cómo evoluciona el error y generar la acción de control para corregirlo. En el modelo EBC, la QoC debe incluir los aspectos relacionados con la gestión de los eventos [Dorf and Bishop, 2008] lo que implica la existencia de parámetros comunes como la productividad o el retardo.

2.5.4 Arquitectura basada en servicios: SWE

SWE parte como una iniciativa del OGC con el propósito de disponer de los datos de sensores vía Servicios Web. El modelo SWE lo propuso la OGC en el año 2004. El modelo propuesto está siendo actualmente empleado en la NASA y está tomando una gran importancia en cuanto a su uso para la monitorización y gestión de redes de sensores.

El objetivo fundamental de SWE es disponer del marco de referencia para el conjunto de estándares que posibilitan la explotación de sistemas y/o sensores conectados en Internet [Botts et al., 2006]. El sistema debe cumplir los siguientes requisitos funcionales:

- Descubrimiento. Se entiende el descubrimiento como la capacidad de descubrir los sensores del sistema y los sistemas de observación que proporcionan.
- Caracterización. Se entiende la caracterización del sistema como la capacidad para determinar las características de los sensores y poder averiguar los detalles de cada uno de ellos.
- Configuración. Consistente en permitir el acceso automático a los parámetros del sensor. Por medio de esta característica se proporciona al usuario la posibilidad de poder configurar los sensores para adecuarlos a los requisitos.
- Publicación. La característica consiste en permitir el acceso en tiempo real a los datos con las características temporales de los sensores. Es decir, el usuario debe poder conocer los valores de los sensores, así como los instantes temporales de las mediciones efectuadas.
- Planificación. La planificación se entiende como la posibilidad de que el sistema permita a los usuarios demandar que los sensores adquieran datos en un tiempo y lugar determinado.
- Subscripción. Consiste en la posibilidad de poder subscribirse a un servicio donde se publicarán las alertas en función de los valores observados por los sensores.

Para lograr estas características, SWE divide las funcionalidades en diversos componentes que se relacionan entre sí. Estas especificaciones van desde los componentes de proceso hasta las especificaciones de la información que se debe transmitir entre componentes. A continuación se describen los componentes del estándar SWE.

2.5.4.1 Componentes SWE

Los componentes de SWE se dividen en dos grupos: los modelos de información y los servicios. Los modelos de información son las especificaciones XML (*Extensible Markup Language*) con las que trabaja el sistema, mientras que los servicios son los componentes encargados de trabajar con dichos modelos. Los modelos propuestos por SWE emplean XML como lenguaje para la especificación y son los tres siguientes:

- O&M: *Observations and Measurements Schema* (Observaciones y medidas).
- SensorML: *Sensor Model Language* (Lenguaje de Modelo de Sensor).
- TML: *Trasducer Markup Language* (Lenguaje de Marcado de Transductores).

La función de O&M es la de servir como modelo para representar las medidas o las observaciones de los sensores. Se considera una observación a un evento cuyo resultado es un valor que describe un fenómeno. Las observaciones se deben asociar a un instante de tiempo en que se realizan, por lo que debe existir un control temporal, así como la información espacial de las mismas que permita localizar a las mediciones. Las observaciones se realizan por medio de procedimientos en los que intervienen los siguientes componentes: sensores, observadores, encadenado de procesos e incluso simulaciones, por lo que una observación no es una medida de un sensor, sino que también puede ser el resultado del procesamiento de una serie de sensores.

SensorML proporciona la información necesaria acerca de los sensores. Esta información debe contener la localización de los mismos, la información necesaria para su análisis y procesamiento de las observaciones que se puedan dar y que permita caracterizar la fiabilidad y calidad de las mismas. Finalmente debe proporcionar la información que permita almacenar las características fundamentales de los mismos. Se debe tener en cuenta que un sensor no es sólo un sensor físico, sino cualquier proceso que recibe y proporciona datos, por lo que SensorML modela también el procesamiento del sistema.

TML es el lenguaje empleado para el intercambio de datos entre sensores y sistemas, el lenguaje comunica tanto los datos del sensor, como la meta-información asociada al mismo. TML es independiente de los sensores que comunica, por lo que la meta-información que pueda contener debe ser común a todos los tipos de sensores. Esto hace que se pueda emplear un procesamiento común a todos los sensores. TML debe permitir intercambiar y operar entre diferentes datos de diferentes sensores heterogéneos y poder fusionarlos.

Los servicios que proporciona SWE se especifican como servicios que deben ofrecer operaciones a los clientes. La información necesaria para estas operaciones se debe intercambiar usando los modelos de información descritos anteriormente. Los servicios que proporciona SWE son los siguientes:

- SOS (*Sensor Observation Service*). El servicio SOS proporciona la descripción de los sensores, la posibilidad de registrarse, las observaciones y todas las características necesarias para interpretar los resultados que se proporcionen.
- SAS (*Sensor Alert Service*). El servicio SAS facilita las funciones necesarias para la configuración y generación de avisos en función de unos parámetros que configuran alarmas. Estos avisos se generan como consecuencia de que la observación de un sensor excede o no alcanza un umbral configurado. Como alerta se envía, además del evento que la ha producido, la fecha y la localización.

- WNS (*Web Notification Service*). Servicio de Notificación Web. El WNS proporciona la posibilidad de suscripción a un servicio de notificaciones en función de un criterio establecido en relación con los sensores de los que se desea recibir la información.
- SPS (*Sensor Planning Service*). El SPS es el servicio responsable de generar y responder a las solicitudes de conjuntos de datos de distintos sensores y las transacciones correspondientes.
- SCS (*Sensor Collection Service*). El SCS es el servicio responsable de las observaciones por medio de los modelos de datos O&M y de las especificaciones del sistema de sensores por medio de *SensorML*.

Los servicios propuestos están muy relacionados con la intención de la arquitectura de ofrecer la posibilidad de trabajar completamente con un sistema de sensores, es decir, los servicios están diseñados para una adquisición y procesamiento distribuido en paralelo y asíncrono. Un análisis detallado del estándar SWE se puede encontrar en [Poza, 2009b].

2.5.4.2 Soporte al procesamiento de datos

Aunque el ámbito original de aplicación de SWE es el de las redes de sensores, el soporte a procesamiento que ofrece se acerca a la funcionalidad requerida de una arquitectura de control inteligente. Este procesamiento se basa en unos componentes esenciales conectados entre ellos [Robin and Botts, 2006] que por medio de funciones proporcionan un procesado de la información entrante y la ofrecen como información ya procesada. En la Figura 18 se observan los componentes de SWE y las relaciones entre ellos.

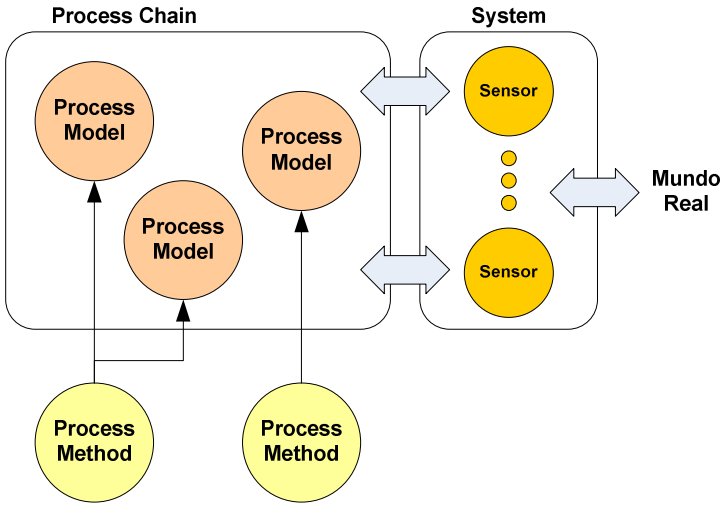


Figura 18 Componentes del modelo de control propuesto por la arquitectura SWE

Desde el punto de vista de SWE un componente es un proceso físico que transforma información. Ejemplos de componentes son: detectores, actuadores y filtros de procesos físicos. En el caso de los sensores físicos, se especifica un tipo concreto de componente llamado *detector*. Un detector es la parte atómica de un sistema de medición que define el muestreo y la respuesta de un dispositivo. Un detector tiene sólo una entrada y una salida y ambas trabajan con valores escalares. Los sensores complejos pueden definirse como una composición de múltiples detectores con múltiples entradas y múltiples salidas.

Un detector es un caso particular de un componente llamado *Process Model*. Un *Process Model* es un bloque atómico de proceso, normalmente empleado dentro de una composición llamada *Process Chain* lo que permite hacer composiciones tan complejas como se requiera. Un *Process Model* está asociado a un *Process Method* que define el interfaz y la forma de ejecución del *Process Model*, además de definir sus entradas, salidas y parámetros de funcionamiento.

Un *Process Chain* es un bloque compuesto de procesamiento, consistente en un conjunto de subprocesos interconectados. Estos subprocesos pueden ser sencillos *Process Model* o combinaciones más complejas de *Process Chain*, lo que proporciona la capacidad de modelar un procesamiento complejo por medio de componentes más sencillos. Un *Process Chain* incluye las fuentes de datos, así como las conexiones explícitas con los elementos que lo componen y las conexiones entre ellos. También se especifican sus entradas, su salidas y los parámetros de una forma similar a como lo hace el *Process Model*.

Un *Process Method* es la definición de la interfaz y del comportamiento de un *Process Model*. Pudiendo ser almacenado en forma de librerías para que sea utilizado por diferentes instancias de *Process Model* asociando un *Process Model* a un *Process Method*. Fundamentalmente describe el interfaz del proceso y el algoritmo que se empleará.

Finalmente, un sistema es un modelo compuesto a partir de un grupo de componentes. Los componentes pueden incluir detectores, actuadores o subsistemas. Un Sistema relaciona un *Process Chain* con el mundo real, por lo que proporciona características adicionales a los componentes, como posiciones relativas o interfaces de comunicación.

2.5.5 Arquitectura basada en la calidad de servicio: DCPS

El modelo de servicio de distribución de datos (DDS) es una especificación para sistemas de distribución de datos basada en el modelo de publicación-subscripción propuesto por el OMG para las comunicaciones con soporte a la calidad de servicio [Pardo-Castellote, 2003]. Todos los detalles del modelo, con los diagramas UML correspondientes y los detalles de las políticas de calidad de servicio se pueden encontrar en [Poza, 2009c].

DDS cubre los sistemas de comunicaciones con necesidades de tiempo real [OMG, 2007]. El modelo se basa en la conexión de productores de información (publicadores) con consumidores de información (suscriptores) desacoplando los componentes en tiempo, espacio y flujo de mensajes [Houston, 1998] con la comunicación basada en la negociación y la gestión de las calidades de servicio.

DDS se divide en dos capas: el Data-Centric Publish-Subscribe (DCPS), que es el responsable de la distribución de los datos, y el Data Local Reconstruction Layer (DLRL), que es la capa responsable de adaptar los datos a las aplicaciones locales. En un sistema DDS la capa DCPS es obligatoria, mientras que la capa DLRL es opcional.

En la capa DCPS, las políticas de calidad de servicio se emplean para describir el comportamiento de una conexión entre los productores y los consumidores. En la capa DCPS de DDS es donde se especifican las políticas de calidad de servicio. Los componentes más relevantes de la capa DCPS se verán con detalle en el siguiente apartado.

2.5.5.1 Componentes DCPS

En la Figura 19 se muestran los componentes de la capa DCPS y las interacciones entre ellos. Los productores son las aplicaciones, o partes de la aplicación, que producen información. La producción de información puede ser bajo demanda, es decir a consecuencia de una solicitud recibida, o por iniciativa propia del componente, a consecuencia del resultado de un proceso. Los consumidores de información son los que reciben datos de los productores, bien como respuesta de una solicitud previa o como consecuencia de un evento.

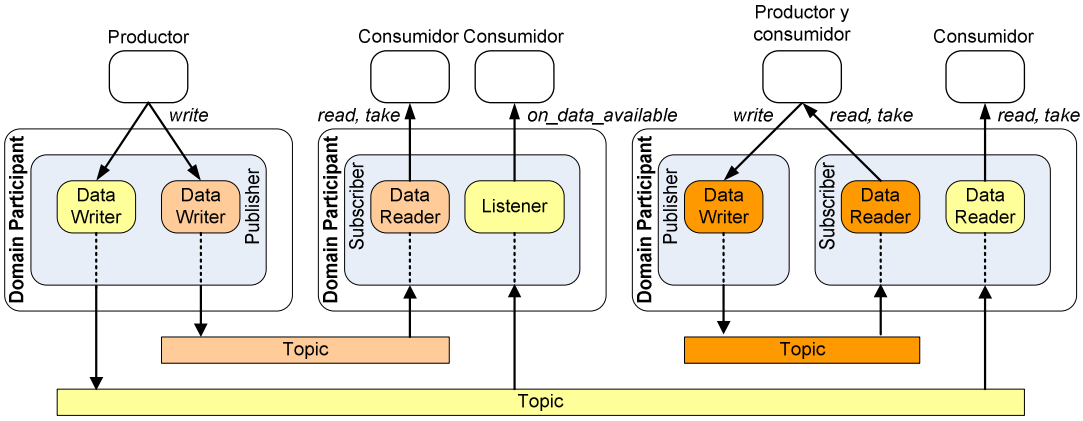


Figura 19 Elementos del modelo de comunicaciones DCPS de DDS

Los productores se consideran escritores de datos y los consumidores se consideran lectores de datos. A continuación se da una breve descripción de cada componente, para seguidamente describir el comportamiento:

- Dominio (*Domain*). Agrupación que une a las aplicaciones individuales con las comunicaciones. Todos los escritores o lectores de datos se comunican en el mismo dominio. En DDS se pueden crear múltiples dominios, aislados unos de otros.
- Componente de dominio (*Domain Participant*). Es un objeto genérico del que se derivan todos los componentes del modelo. Al ser el objeto base de todos los componentes, permite especificar unos valores de calidad de servicio por defecto para todos los componentes.
- Lector (*Data Reader*). Es el punto inicial de acceso para una aplicación a los datos recibidos por un subscriptor. Una vez se ha creado y configurado con la correspondiente calidad de servicio, una aplicación puede consultar los datos disponibles mediante dos métodos: *take* y *read*. El primero elimina el dato del *middleware*, una vez es leído. El segundo método deja el dato en el *middleware* para poder ser leído en más de una ocasión.
- Oyente (*Listener*). Este componente realiza una función similar a la del *DataReader*, pero se diferencia en que funciona por evento y no por consulta, es decir, es el *Listener* el que recibe un aviso por cada mensaje que llega al *Subscriber*.
- Escritor (*Data Writer*). Es el punto de acceso de una aplicación para publicar datos en un *Publisher*. Una vez se han configurado todos los parámetros de calidad de servicio, la aplicación llama a la función *write* de un *DataWriter* para enviar la información.

- Asunto o tema (*Topic*). Es el mecanismo por medio del cual los productores y consumidores se comunican, constituye el punto de conexión entre los *Publisher* y los *Subscriber*. Debe tener un identificador (*Topic name*) y un tipo (*Topic type*). El nombre es una cadena de caracteres que identifica unívocamente al tema dentro de un dominio. El tipo define el carácter del contenido.
- Clave del *Topic* (*Topic Key*). Dentro de un *topic*, se puede escoger uno de los campos internos como clave, este campo se emplea para ordenar los datos entrantes, de manera que se puede utilizar para seleccionar datos.
- Publicador (*Publisher*). Es una entidad que contiene un grupo conjunto de *DataWriters*. Un desarrollador puede especificar un comportamiento por defecto basado en los parámetros de calidad de servicio, que se aplicará a los *DataWriter* que pertenezcan al *Publisher*.
- Subscriptor (*Subscriber*). Los subscriptores se emplean para agrupar componentes *Listener* y *DataReader*, esto permite configurar calidades de servicio por defecto al conjunto de componentes.

Partiendo del *Topic* se puede justificar el resto de componentes. El *Topic* es una forma de localizar la información en el sistema, similar al nombre de una variable dentro del código de un programa.

Los componentes que escriben y leen en un tema son el *Publisher* y el *Subscriber* respectivamente. Estos dos últimos componentes de comunicación son el nexo con los componentes que quieran emplear el sistema de comunicaciones, es decir, el *Publisher* y el *Subscriber* son la interfaz que los componentes ven del sistema de comunicaciones.

Los componentes *Publisher* y *Subscriber* son los responsables de realizar las operaciones de comunicación propiamente y se comunican con un *DataWriter* y un *DataReader* respectivamente, de forma que aíslan los detalles de las comunicaciones al resto de los componentes.

La separación entre los componentes que dialogan con el control y los que dialogan con el medio de comunicación permite organizar y gestionar las comunicaciones de manera eficiente, y dota al sistema del concepto de middleware que aísla el control de las comunicaciones y viceversa.

Las operaciones principales de comunicación que DCPS proporciona son las siguientes:

- Escritura por consulta a petición del componente en un *DataWriter*.
- Lectura por consulta, a petición del componente desde un *DataReader*
- Lectura por evento, o aviso, a petición del medio de comunicación por medio de un *Listener*.

Estas operaciones ofrecen un punto de vista muy eficiente similar al protocolo SNMP (Simple Network Management Protocol) de gestión de redes [Stallings, 1999]. En el modelo SNMP la comunicación por consulta se basa en dos operaciones: lectura (*get*) y escritura (*set*) y la comunicación por evento en la operación de aviso (*trap*), que consiste en el envío de un mensaje. SNMP está diseñado para gestionar de forma sencilla cualquier dispositivo en red. Esta gestión puede aplicarse a diversos sistemas de control. En [Giladi, 2004] y [Gang et al., 2000] se pueden ver diversos ejemplos de cómo gestionar un sistema de control domótico con las tres operaciones fundamentales de SNMP.

2.5.5.2 Soporte a la calidad de servicio

En el modelo DCPS, las políticas de calidad de servicio se implementan como una lista de calidades de servicio que debe cumplir el componente al que se asocie. Todos los componentes de un sistema de comunicaciones pueden tener un conjunto de calidades de servicio asociadas.

Las calidades de servicio que se solicitan por parte de un Subscriptor, deben ser cumplidas por un Publicador. Para la negociación se sigue la secuencia Subscriptor solicita y Publicador ofrece. El servicio será quien determinará si los requisitos son compatibles. Excepto en unos casos concretos, los servicios ya negociados pueden ser cambiados aun ya establecida la conexión.

Las políticas de calidad de servicio de DDS se han organizado previamente en áreas atendiendo a la funcionalidad que ofrecen o el ámbito de la comunicación que ofrecen, tal como se ha realizado en la Tabla 3 (página 62): gestión de meta datos, gestión de los aspectos temporales, gestión del flujo de mensajes y gestión de los componentes DDS. La descripción de las políticas se realizó anteriormente en el subapartado 2.4.4.4.

La gestión de los meta datos consiste en proveer a los componentes de la capacidad de enviarse mensajes entre sí con información adicional que no se puedan enviar por medio del resto de las políticas de calidad de servicio. No hay limitaciones en cuanto al contenido se refiere.

Finalmente, en DCPS se tienen políticas de calidad de servicio orientadas a gestionar las ubicaciones de los componentes en el sistema. La más interesante es la que permite dividir el sistema en áreas aisladas o partición. La partición permite especificar conjuntos de componentes entre los que se envían mensajes a los que no pueden acceder los componentes ajenos a la partición.

2.5.6 Discusión

La tendencia en el control distribuido inteligente consiste en emplear arquitecturas orientadas a componentes que intercambian tanto la información como las funcionalidades mediante el ofrecimiento y la utilización de servicios. Por ello, los componentes precisan de un modelo que proporcione de forma general y estandarizada, los servicios y los patrones necesarios para poder ser implementado con garantías en un sistema. El modelo SWE, especialmente en lo relativo a los procesos físicos, da una aproximación sencilla y fácilmente adaptable a sistemas de control distribuido, ya que los servicios son uno de los métodos de trabajo más eficientes en este tipo de sistemas.

La cantidad de información que manejan los sensores complejos exige, por parte del sistema de comunicaciones, unas características tanto de su arquitectura física o hardware como de su arquitectura lógica o software. Entre las características más destacables está la gestión de la información, especialmente en lo que se refiere al volumen, a los requisitos temporales y el tipo de flujo de mensajes que el sistema de control requiere. Además, la tendencia actual de las arquitecturas de control se basa en la gestión de eventos, por lo que el sistema de comunicaciones empleado debe ser capaz de dar soporte a la todas las necesidades de comunicación de las arquitecturas.

El modelo DCPS es una plataforma adecuada para dar un soporte de comunicaciones eficaz a los sistemas de control distribuido, ya que organiza de forma sencilla los componentes que se precisan en las comunicaciones entre los diversos componentes del sistema.

El hecho de emplear la calidad de servicio para definir los parámetros temporales y de flujo de información sobre los que basar la comunicación de los componentes, ofrece la posibilidad de adaptar el sistema a las condiciones particulares que se puedan dar en el entorno a lo largo de todo el proceso de control.

2.6 Conclusiones

En este capítulo se ha realizado un análisis de las características, tanto de los sistemas de control como de las comunicaciones, que dan soporte al control distribuido. En la Figura 20 se presentan dichas características, las relaciones de similitud entre ellas y los requisitos a los que dan lugar. Requisitos que determinarán la base para el desarrollo de la arquitectura propuesta en la presente tesis doctoral.

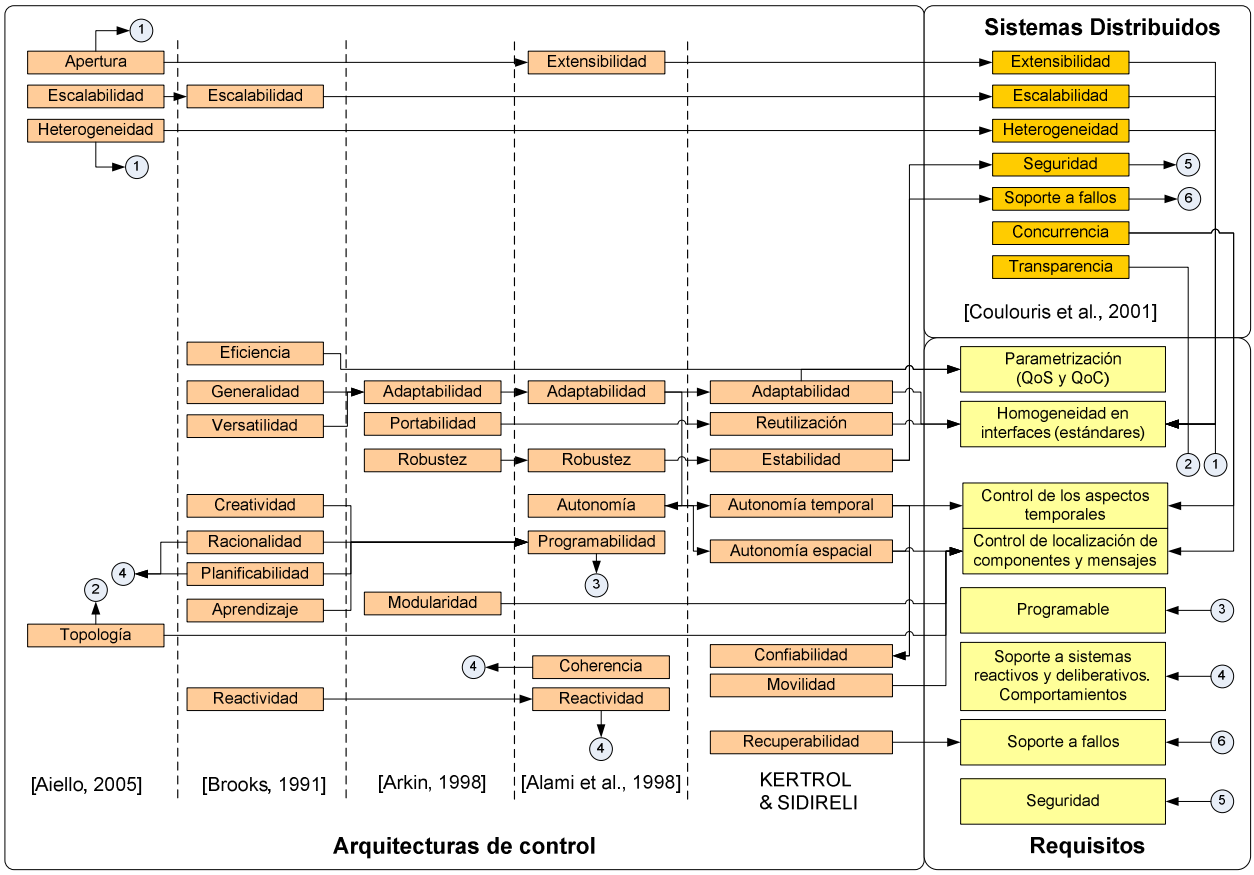


Figura 20. Relaciones entre las características de los sistemas distribuidos de control.

A continuación se describen los requisitos que deben caracterizar la arquitectura presentada en esta tesis doctoral (parte inferior derecha de la Figura 20):

- Soporte al uso de parámetros tanto internamente como externamente. La interna se refiere a la caracterización de los propios componentes con parámetros que permitan conocer su estado, como por ejemplo la carga o los tiempos de computación. La externa se refiere a la caracterización de las condiciones del entorno en el que los componentes actúan, como las tasas de llegadas de mensajes erróneos o pérdidas de los mismos. Los parámetros de calidad de servicio permiten caracterizar estos aspectos, por lo tanto el modelo propuesto deberá proporcionar soporte a la calidad de servicio.

- Homogeneidad en interfaces de los componentes, intentando que todos los componentes puedan ser definidos de forma semejante. El uso de estándares permite disponer de un entorno común, estudiado y suficientemente probado. Por ello uno de los requisitos principales será la priorización en el uso de estándares para implementar tanto el control como las comunicaciones.
- Control de los aspectos temporales. El sistema deberá proporcionar la capacidad de poder suministrar a los componentes la información temporal necesaria. Los aspectos temporales deben controlarse tanto desde los elementos de control como desde los elementos de comunicaciones. El control temporal debe ser tanto interno al componente, como por ejemplo los tiempos de cómputo de un mensaje, como externo al componente, como por ejemplo conociendo los tiempos de cómputo de los mensajes que vayan llegando al componente.
- Control de los aspectos espaciales. Entendiendo éstos como la capacidad de poder localizar los componentes en el sistema. Además también se debe ofrecer la capacidad de movimiento de los componentes entre los diferentes nodos.
- Soporte a sistemas reactivos y deliberativos. El soporte a diferentes niveles de control implica una equiparación de los componentes con el nivel de abstracción del control que aplican, es decir, permite desde componentes reactivos hasta componentes deliberativos.
- Soporte a fallos. El soporte a fallos es uno de los aspectos más complicados de gestionar, dado que implica la detección del mismo, la minimización del impacto y la restauración de las condiciones. Para poder tener un soporte a fallos eficiente se deberá proporcionar al sistema toda la infraestructura que permita la detección del mismo, generalmente suministrando los indicadores necesarios para determinar que se ha producido el fallo. Además, se deberá proporcionar la infraestructura necesaria para paliar los efectos del fallo, como por ejemplo ofreciendo un soporte a la detección de la caída de los componentes.
- Gestión de la seguridad. La seguridad, entendida como la protección de aquellos aspectos del sistema que se considere oportuno, generalmente por medio de sistemas de identificación segura de componentes o encriptación de información en el caso de los mensajes.

A medida que la variedad de sistemas de control evolucionan de los modelos centralizados a los modelos distribuidos, la importancia de las comunicaciones es mayor. Por ello resulta de especial interés estudiar la relación de las comunicaciones y el control con la eficiencia con la que la arquitectura implementa un sistema. Para ello, es preciso dotar a la arquitectura de un mecanismo de medición de la eficiencia con la que implementa un sistema.

Para la arquitectura presentada en la presente tesis doctoral se ha seleccionado el modelo de control SWE propuesto por la OGC para el procesamiento de redes de sensores debido a la gran abstracción que realiza de los procesos de control. Existen otros estándares de control como ERSP (*Evolution Robotics Software Platform*) que están orientados al ámbito de aplicación, mientras que SWE es un modelo genérico.

De entre los modelos de comunicación presentados en los apartados anteriores, el modelo DCPS del estándar DDS es el más adecuado como base de las comunicaciones de la arquitectura propuesta en la presente tesis doctoral. La elección del modelo se debe principalmente a que proporciona una taxonomía aplicada de la calidad de servicio en forma de políticas que pueden ser aplicadas para optimizar el control del sistema.

3 Arquitectura FSACtrl

3.1 Introducción

3.1.1 Motivación

En el capítulo anterior se han revisado las arquitecturas de control distribuido desde el ámbito de la domótica hasta el de la navegación de robots, además se han revisado los sistemas de comunicaciones empleados en los sistemas distribuidos y el papel de la calidad de servicio en los middleware. A partir de la revisión se han determinado una serie de características relevantes en los sistemas distribuidos de control que sientan las bases para su optimización.

En este capítulo se propone un modelo de arquitectura de sistema distribuido de control que permite medir, por medio de los parámetros de calidad de servicio, su grado de optimización. Los parámetros de calidad de servicio, y los equivalentes de calidad del control, deben proporcionar la información suficiente al diseñador del sistema para poder tomar decisiones encaminadas a la optimización del mismo.

Asimismo, el modelo emplea los valores de los parámetros de la calidad de servicio para descargar parte de las funciones del control, como el filtrado de mensajes, la generación de eventos o la ejecución de acciones, permitiendo por tanto delegar en la capa de comunicaciones parte de las decisiones que habitualmente se toman en la capa de control.

La arquitectura, denominada FSACtrl, toma como punto de partida el modelo Frame Sensor Adapter (FSA) [Posadas et al., 2002] desarrollado íntegramente en el grupo de investigación, adicionándole las características de control del modelo SWE y ampliándolo con la calidad de control y la gestión de eventos, así como con las características de comunicaciones del modelo DCPS en lo que respecta al tratamiento de la calidad de servicio.

Es habitual encontrar modelos formales en todos los ámbitos donde intervenga un proceso de computación, tal como las arquitecturas de control [Astigarraga, 2001]. Para su descripción se dispone de diversas herramientas de entre las cuales, actualmente, el modelo UML [OMG, 2003] es uno de los más adecuados y extendidos. Por ello, se emplearán los diagramas UML como método descriptivo.

3.1.2 Antecedentes: el modelo FSA

El modelo Frame Sensor Adapter (FSA) ha sido desarrollado en el grupo de investigación de Informática Industrial del Instituto de Automática e Informática Industrial (ai2) de la Universidad Politécnica de Valencia (UPV). El objetivo principal del modelo FSA es definir un método común de acceder a cualquier recurso de comunicaciones y está ampliamente probado en diversos entornos de tiempo real, principalmente en sistemas de robótica móvil [Posadas et al., 2008], y en entornos industriales ampliamente distribuidos [Poza et al., 2003].

En el modelo FSA las aplicaciones emplean una interfaz común para enviar y recibir mensajes de diversos canales de comunicaciones [Pérez et al., 2001]. En la Figura 21, se muestran los componentes y las operaciones principales. El modelo se describe con detalle en [Posadas, 2003].

El modelo define tres interfaces: IFrame, ISensor e IAdapter, que se corresponden con los tres componentes de la arquitectura: Marco, Sensor y Adaptador (Figura 21). El marco define un entorno común donde contener y gestionar (crear, modificar o eliminar) las instancias de los componentes adaptadores y los componentes sensores. Este marco común relaciona los adaptadores con los sensores y permite una coordinación más sencilla desde la aplicación o el agente que los utilice.

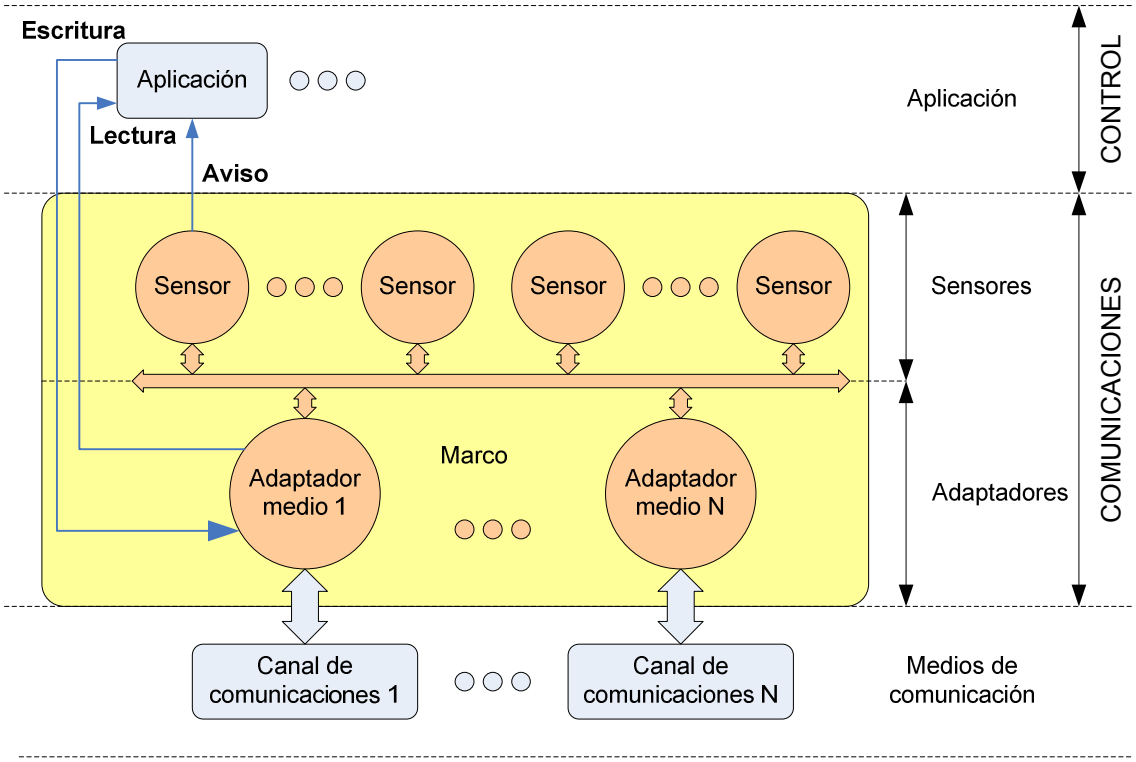


Figura 21. Componentes de la arquitectura FSA.

El sensor es el componente que permite la recepción automática de datos procedentes de los recursos de comunicaciones. El sensor recibe los mensajes a través de los adaptadores. Los adaptadores proporcionan un formato común para los mensajes y los datos independientemente del canal de comunicaciones empleado.

En la Figura 21 se muestra cómo los sensores se conectan a los adaptadores para acceder a los mensajes y no depender de las características concretas de cada canal de comunicaciones. Un mismo sensor puede conectarse a varios adaptadores.

Los adaptadores adaptan las particularidades del canal de comunicaciones a un interfaz cuya sintaxis es común a todos los componentes de la arquitectura. De esta forma, los procesos conectarán y accederán con una misma interfaz a cualquier recurso de comunicaciones utilizando el adaptador correspondiente.

El adaptador ofrece un acceso por consulta y otro por eventos del recurso de comunicaciones. Se entiende como acceso por consulta, aquel en el que la aplicación lee o escribe en el recurso de comunicaciones a iniciativa propia. Como acceso por eventos, se entiende el modo de funcionamiento en el que la aplicación es avisada a iniciativa del recurso de comunicaciones.

3.1.3 Descripción del capítulo

En el apartado 3.2 se realiza la descripción conceptual del modelo propuesto ofreciendo una visión general del mismo. Se describen los componentes y la funcionalidad. Inicialmente se describen los componentes responsables del control y de las comunicaciones que a partir de ahora se llamarán elementos. Todos ellos se describen primero desde un punto de vista conceptual y a continuación con una descripción formal basada en diagramas UML.

Seguidamente, en el apartado 3.3 se describen las operaciones de la arquitectura, las relacionadas con los elementos y con la gestión de los eventos que proporcionan el mecanismo de realización de las operaciones de optimización del sistema.

Dada la importancia que tiene la QoS en el sistema, se dedica la sección 3.4 a describir con detalle los parámetros definidos en el capítulo anterior, pero adaptados a la arquitectura.

Finalmente, en la sección 3.5 se ofrecen las conclusiones del capítulo que permiten la realización del diseño mostrado en el capítulo 4.

3.2 Elementos de la Arquitectura FSACtrl

3.2.1 Visión global

3.2.1.1 Modelo conceptual

3.2.1.1.1 Arquitectura FSACtrl

El modelo FSA cubre los aspectos de comunicación de los componentes de una arquitectura de control de sistemas. Sin embargo, es conveniente ampliar el modelo para cubrir algunas de las áreas de interés del control distribuido:

- Soporte a parte del control de los componentes dentro del mismo modelo de comunicaciones.
- Uso de modelos estándares con la validación de los organismos correspondientes que los promueven.
- Soporte a la calidad de servicio.
- Soporte a la calidad de control.
- Gestión de eventos por medio de su agrupación en condiciones y del desencadenamiento de acciones asociadas.

En el aspecto de la estandarización se ha empleado el modelo DCPS para las comunicaciones y el control está inspirado en el modelo SWE. El aspecto de la calidad de servicio lo proporciona el uso de DCPS por medio de las políticas de calidad de servicio.

En la Figura 22, se muestran las áreas principales de la arquitectura FSACtrl y la relación con los estándares SWE y DCPS. En dicha figura se aprecia cómo hay un solapamiento en las funciones de comunicaciones y de control por parte de los dos estándares. Esto se debe a que los componentes de control se emplean para implementar parte de las comunicaciones.

La arquitectura FSACtrl emplea el concepto de sensor del modelo FSA por medio de un elemento llamado *sensor lógico*, con la capacidad de interconexión entre sensores empleada por el *process chain* del modelo SWE. La conexión de sensores lógicos se organiza mediante una estructura de sensores denominada *Logical Sensor Graph (*LSG) que representa el interior de un componente de control. A su vez, los componentes de control se organizan jerárquicamente formando una estructura denominada *Control Component Tree* (CCT).

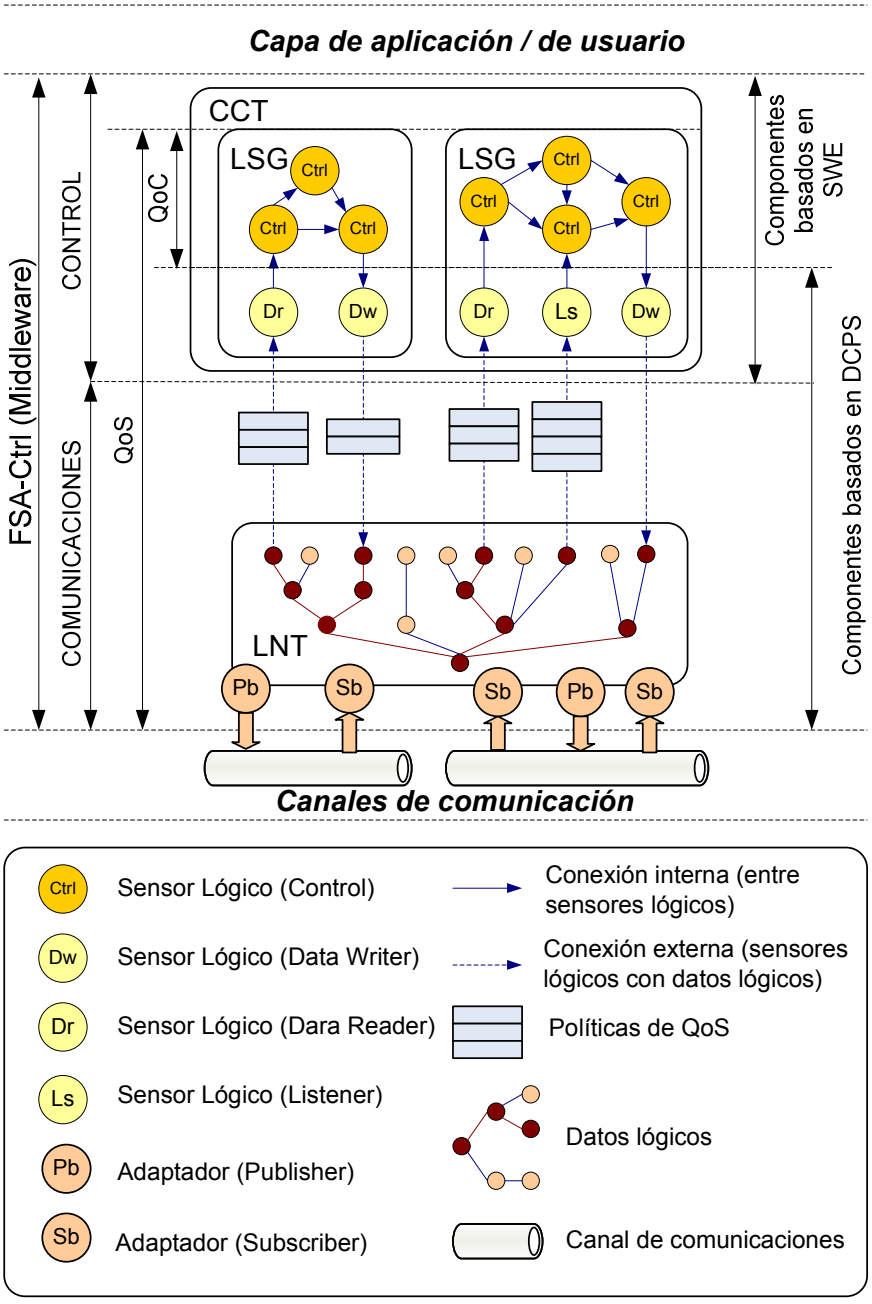


Figura 22. Áreas de la arquitectura FSACtrl.

Cabe destacar que los sensores lógicos responsables de la conexión entre las comunicaciones y el control: *DataReader*, *DataWriter* y *Listener* son a la vez procesos de control basados en SWE y componentes de comunicaciones del modelo DCPS.

En lo que respecta a las comunicaciones, la arquitectura FSACtrl extiende el concepto de adaptador del modelo FSA para dividirlo, tal como se propone en el modelo DCPS, en los elementos *Publisher*, responsables de adaptar los mensajes de los componentes de control a las características del canal de comunicaciones, y los *Subscriber*, que realizan la labor inversa a los *Publisher*, adaptando los mensajes de los canales de comunicaciones a la semántica común de los sensores lógicos.

La conexión entre el control y las comunicaciones se realiza por medio de los datos lógicos que representan unidades de información. Los datos lógicos se organizan en una estructura en árbol llamada *Logical Namespace Tree* (LNT), que relaciona todos los sensores lógicos de comunicaciones, que pertenecen al control, con los elementos de comunicaciones [Poza et al., 2008b]. Esta conexión traslada al control el concepto de *Topic* del modelo DCPS y lo organiza en un espacio simbólico jerarquizado que facilita la localización de la información el negocio de las políticas de calidad de servicio.

A modo de resumen, en la Figura 23 se muestra la ruta de datos de la arquitectura. Los mensajes se reciben del canal de comunicaciones por medio de los *Subscriber*, que los envían a los *DataReader* o *Listener* con los que compartan datos lógicos. En el caso del *Listener* el mensaje se transmite de inmediato al sensor lógico de control, mientras que en el caso de ser un *DataReader*, será el sensor lógico de control quien deberá leerlo. Los sensores lógicos procesan los mensajes y, si procede, generan otros mensajes que envían a los otros sensores lógicos con los que esté conectado. Cuando se debe enviar un mensaje a un canal de comunicaciones, el sensor lógico de control, debe enviarlo al correspondiente *DataWriter* que lo enviará al *Publisher* con el que comparta dato lógico.

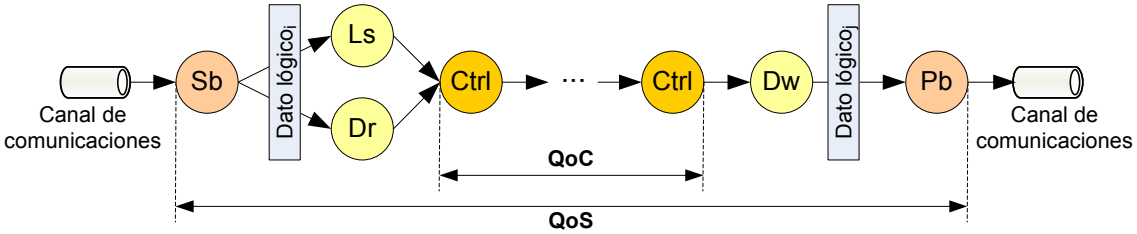


Figura 23. Ruta de datos entre los elementos de la arquitectura FSACtrl.

Debido a que la calidad de servicio se encuentra presente en todos los elementos de la arquitectura FSACtrl, tal como propone el modelo DCPS, los mismos parámetros de calidad de servicio se emplean para evaluar o mejorar la eficiencia tanto de las comunicaciones como del sistema de control.

3.2.1.1.2 Nomenclatura

Dado que la arquitectura emplea una terminología específica, y a modo de resumen, a continuación se define y ubica la terminología empleada para los elementos de la arquitectura (Figura 24).

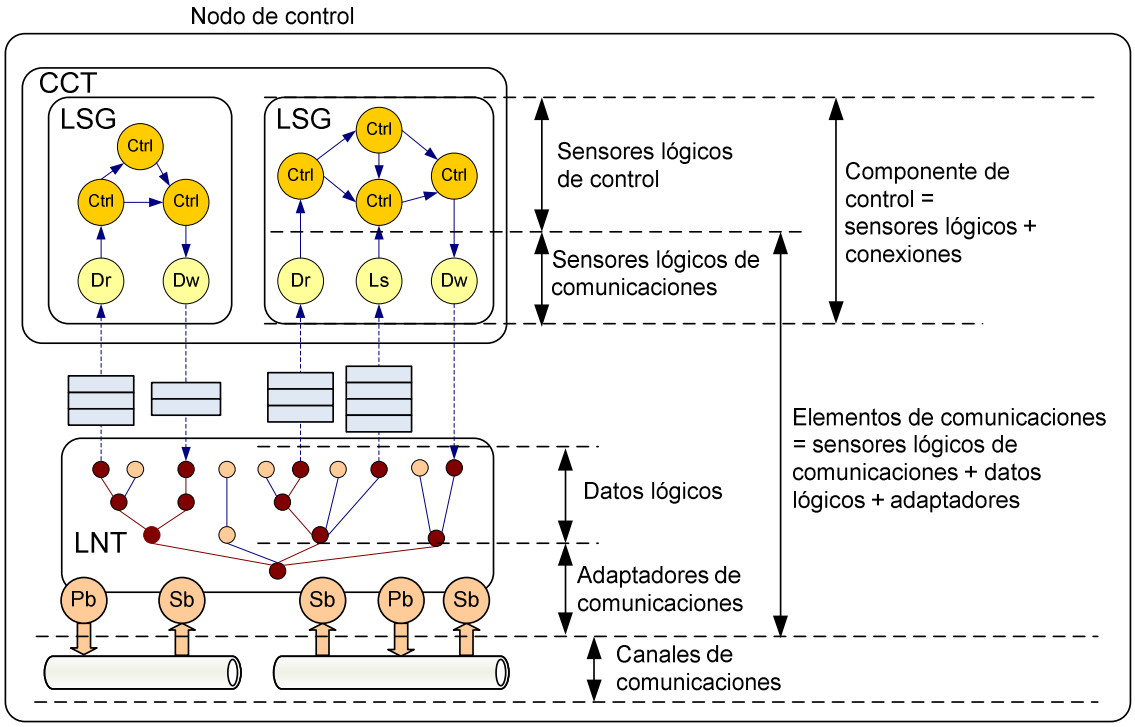


Figura 24. Terminología empleada en la arquitectura FSACtrl.

La definición de los elementos mostrados es la siguiente

- Elementos. Todas las unidades mínimas que pertenecen a la arquitectura. También se consideran elementos las características asociadas, es decir: políticas de QoS, los parámetros de QoC, las condiciones, los eventos y las acciones.
- Componentes de control. Son unos sensores lógicos con una labor estructural, contienen sensores lógicos, agrupándolos y generando una jerarquía a partir de la inclusión de unos componentes de control en otros componentes de control.
- Sensores lógicos de control. Son los sensores lógicos que procesan los mensajes entrantes realizando las operaciones para los que hayan sido programados. Se basan en el modelo de los *ProcessModel* y *ProceesMethod* del modelo estándar SWE.
- Sensores lógicos de comunicaciones. Son los sensores lógicos que se conectan a los adaptadores para intercambiar los mensajes de los sensores lógicos de control de un componente de control con los adaptadores de comunicaciones. Se corresponden con los *DataReader*, *DataWriter* y *Listener* del modelo estándar DCPS
- Adaptadores de comunicaciones. Son los elementos que conocen las particularidades del medio de comunicación. Los adaptadores que escriben en el medio son los *Publisher*, los que leen del medio son los *Subscriber*, en ambos casos se corresponden con los mismos elementos del modelo estándar DCPS.
- Datos lógicos. Nombres que abstraen el sistema de comunicaciones a los componentes de control mediante la conexión de los sensores lógicos de comunicaciones con los adaptadores.

- Elementos de comunicaciones. Son los elementos que tienen responsabilidades de comunicaciones, independientemente de si son adaptadores o sensores lógicos de comunicaciones. Los elementos de comunicaciones son los principales implicados en la gestión de la QoS.
- Canales de comunicaciones. Medios de comunicaciones que se pueden emplear, ya que se dispone de los elementos de la arquitectura FSACtrl que sirven de interfaz con estos medios de comunicación.
- Nodo de control. Sistema que alberga todos los elementos de la arquitectura FSACtrl.

3.2.1.2 Especificación formal

En la Figura 25, se presenta el diagrama UML de clases de la arquitectura FSACtrl. Para describir cómo se implementa la arquitectura se han separado los elementos en los siguientes grupos de clases.

- Clases base. Son las clases que definen las características comunes de los elementos de la arquitectura y del nodo de control.
- Clases de implementación del control. Son las clases base y clases derivadas que dan el soporte al control basándose en los componentes de control y los sensores lógicos que se incluirán en cada componente de control.
- Clases de implementación de las comunicaciones. Son las clases a partir de las que se crean los adaptadores de comunicaciones. También se incluyen como clases de comunicaciones las que dan soporte al LNT.
- Clases de soporte a la calidad de servicio. Son las clases que proporcionan los parámetros de calidad de servicio y la gestión de las políticas de calidad de servicio por parte de los elementos.
- Clases de soporte a la gestión de los eventos. Son las clases que proporcionan la captura de eventos, la agrupación de los mismos en condiciones y la asociación de acciones a las condiciones.

Las clases base de los elementos son *Entity*, *Frame* y *FrameEntity*, permiten agrupar los elementos en un entorno común y agrupan las características comunes de todos los ellos.

Las clases que implementan el control parten de la clase base *LogicalSensor*. La clase *LogicalSensorControl* implementa los sensores lógicos de control que implementan los algoritmos de control del sistema. La clase *LogicalSensorComponent* es la responsable de contener los sensores lógicos de control y las conexiones entre ellos, lo que permite generar el LSG. La relación de composición de la clase *LogicalSensorComponent* con los elementos de su misma clase es la que implementa el CCT mediante el cual los componentes de control se agrupan en áreas de control. La clase *LogicalSensorCommunication* es la que implementa los sensores de control responsables de las comunicaciones dentro de los componentes de control. A partir de la clase *LogicalSensorCommunication* se derivan las clases *DataWriter*, *DataReader* y *Listener* cuyos interfaces cumplen el estándar DCPS.

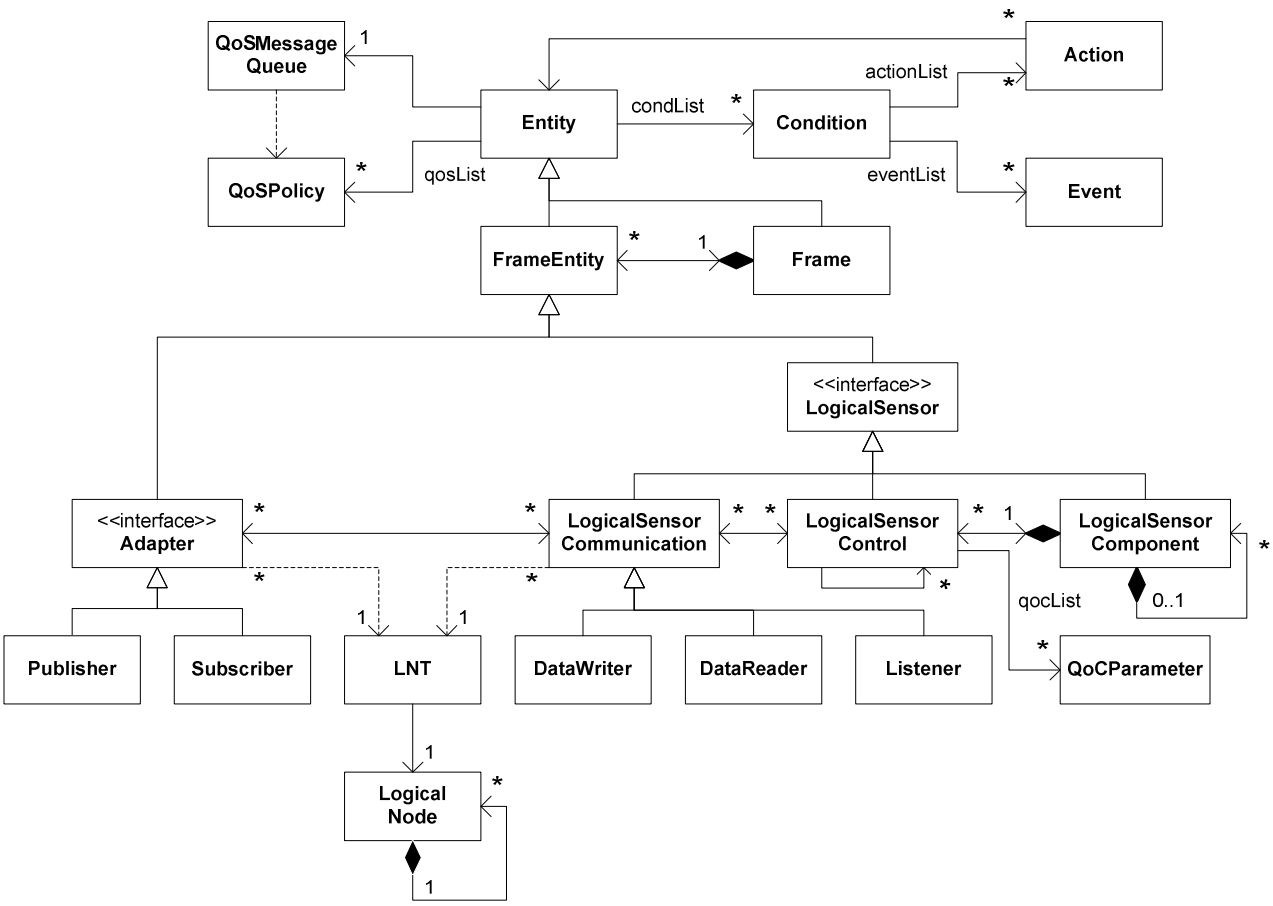


Figura 25. Diagrama de clases UML de la arquitectura FSACtrl.

Las clases que implementan las interfaces con los canales de comunicaciones parten de la clase base *Adapter* y son las clases *Publisher* y *Subscriber* que cumplen el estándar DCPS. Las clases *LNT* y *LogicalNode* son las necesarias para implementar el espacio de nombres lógicos y asocian los sensores lógicos a los elementos de comunicaciones.

Las clases de soporte a la calidad de servicio son las clases responsables de proporcionar las políticas de calidad de servicio a los elementos de la arquitectura. Cabe destacar que hay dos clases importantes para la calidad de servicio: la clase *QoSMessageQueue* que da soporte a las colas de mensajes y proporciona los parámetros de calidad de servicio, y la clase *QoSPolicy* que implementa las políticas de calidad de servicio, las cuales gestionan el flujo de mensajes en función de los parámetros de QoS obteniéndolos y aplicándolos en las colas de mensajes.

Las clases que dan soporte a la gestión de eventos son las clases *Event*, *Condition* y *Action*. La clase *Event* implementa las características del contexto que implican un evento. La clase *Condition* agrupa diversos eventos, los relaciona por medio de operaciones lógicas AND y OR y los asocia al elemento sobre los que se deben detectar. La clase *Action* especifica la acción que se deberá tomar cuando la condición se haya dado.

Hay una relación directa entre algunas de las clases de la arquitectura FSACtrl con los modelos FSA, SWE y DCPS:

- Modelo FSA: *Frame*, *LogicalSensor* y *Adapter* que implementan el Marco el Sensor y el Adaptador respectivamente.

- Modelo SWE: *LogicalSensorControl* y *LogicalSensorComponent* que implementan el *ProcessMethod* y el *ProcessChain* respectivamente.
- Modelo DCPS: *QosPolicy*, *Entity*, *FrameEntity*, *Publisher*, *Subscriber*, *DataWriter*, *DataReader* y *Listener* que implementan los componentes homónimos del modelo DCPS.

Las clases que amplían los modelos estándares son: la clase *QoSessageQueue, l*as clases *LNT* y *LogicalNode* y las clases de la gestión de eventos: *Event*, *Condition* y *Alarm*. A continuación se describen con detalle los grupos de clases introducidos anteriormente.

3.2.2 Clases base

La arquitectura FSACtrl parte de una clase principal *Entity*, todos los elementos de la arquitectura se derivan de esta clase que actúa como clase de soporte a las funciones comunes a todos los elementos de la arquitectura. La clase *Entity* tiene la conexión con el soporte a la calidad de servicio para que todos los elementos de la arquitectura puedan ser parametrizados y gestionados.

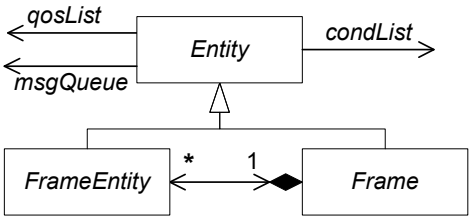


Figura 26. Clases estructurales de la arquitectura FSACtrl.

En la Figura 26 se muestra como a partir de la clase *Entity* se derivan dos clases importantes de la arquitectura: *FrameEntity* y *Frame*. La clase *Frame* es la responsable de agrupar todos los elementos de la arquitectura en un marco único. Cada *FrameEntity* sólo puede pertenecer a un mismo *Frame* para evitar la ambigüedad en la ubicación de los elementos en el sistema. El hecho de tener un único *Frame*, da al sistema una consistencia muy útil, ya que permite disponer de un entorno único desde el que poder acceder a todos los elementos del sistema.

La clase *FrameEntity* es la clase base de los elementos que se pueden agrupar en un mismo *Frame*. A cualquier objeto derivado de la clase *FrameEntity*, además de la lista de políticas de calidad de servicio, también se le puede asociar una cola de mensajes que será la responsable de comunicar al elemento con el resto de elementos del sistema, así como un conjunto de condiciones que permitan detectar eventos y actuar en función de unas circunstancias concretas, como por ejemplo filtrar mensajes en función de la detección de un patrón de contenido.

3.2.3 Clases de soporte al control

3.2.3.1 Modelo conceptual

3.2.3.1.1 Sensores lógicos

En el área de control, los elementos se conocen como sensores lógicos y son los responsables de recibir, procesar los mensajes y determinar qué acciones se deben tomar.

Los sensores lógicos pueden implementar desde sencillas operaciones atómicas como podrían ser sumas aritméticas o comparaciones lógicas, hasta operaciones complejas como el seguimiento de una trayectoria en la navegación de un robot móvil. En este último caso, la implementación se realiza por medio de la agrupación y coordinación de los sensores lógicos en elementos llamados componentes de control, creados a partir de la inclusión jerárquica de otros componentes de control para formar otros componentes de control.

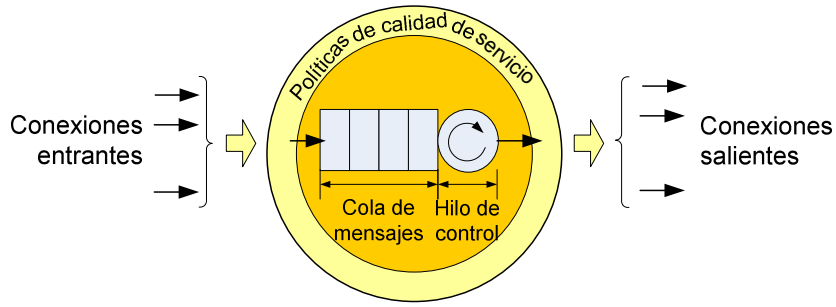


Figura 27. Configuración de los elementos de la arquitectura FSACtrl.

Todos los elementos de la arquitectura FSACtrl disponen de una cola de mensajes a la que poder aplicar unas políticas específicas de calidad de servicio (Figura 27). Cuando un mensaje llega a un sensor lógico éste es encolado en función de su prioridad. Al emplear una cola de mensajes se puede disponer de los parámetros de calidad de servicio. El hilo de control extrae los mensajes de la cola, los procesa y envía el resultado a los elementos correspondientes.En el procesamiento del mensaje es donde se puede evaluar, si procede, la relación del resultado con el valor de referencia. A partir de esa referencia se calcula el error cometido y consecuentemente los parámetros de QoC.

3.2.3.1.2 Grafo de sensores lógicos

La función principal de los sensores lógicos de control es la del procesamiento de los datos. Sin embargo, hay un tipo de sensor lógico, llamado componente de control (Figura 28), cuya función es la de contener a otros sensores lógicos conectados entre ellos.

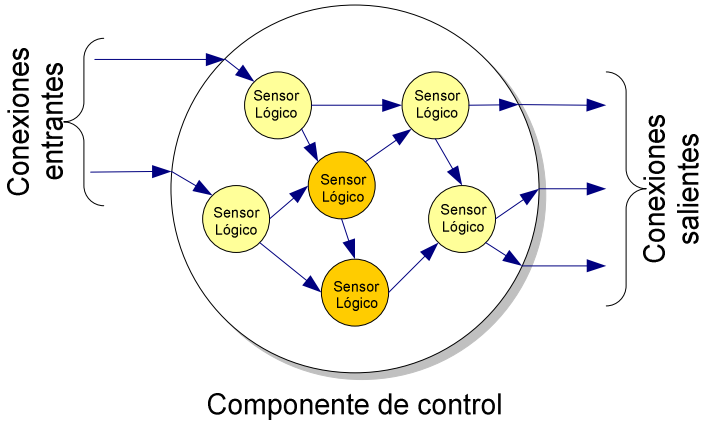


Figura 28. Concepto de Logical Sensor Graph.

El componente de control abstrae la funcionalidad de los sensores lógicos que lo componen, formando el LSG y que es similar al concepto de componente de control habitualmente empleado en las arquitecturas de control.

Un componente de control puede ser tan complejo como sea necesario, este aspecto lo determina el conjunto de los sensores lógicos de control y de comunicaciones que lo componen y las conexiones entre ellos. Los componentes de control se incluyen recursivamente dentro de otros componentes de control, lo que da lugar al CCT.

En la Figura 29 se muestra un ejemplo de LSG en el que se observan los sensores lógicos de comunicaciones, etiquetados como Dr para los *Data Readers*, Dw para los *Data Writers* y Ls para los *Listeners*. Los sensores lógicos sin etiquetar son los sensores lógicos de control. Las conexiones entre los sensores determinan el flujo de mensajes. En el ejemplo, se trata de una detección de colisiones. En la parte inferior de la figura hay un Ls que avisa, por medio de un evento, de la variación del valor de alguno de los sensores del anillo de infrarrojos. En el caso de detectarse, se pasa a medir los Dr para determinar el valor concreto y calcular la nueva velocidad que se enviará a los Dw conectados a los motores.

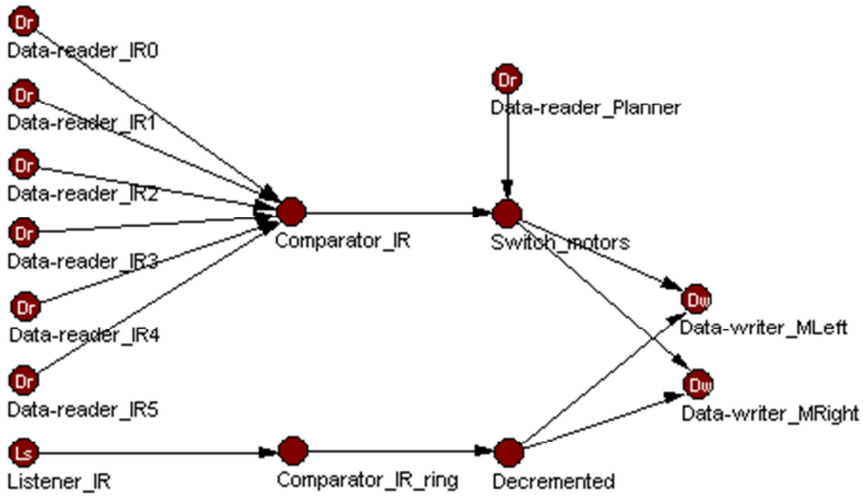


Figura 29. Ejemplo de Logical Sensor Graph.

Los componentes de control son mecanismos para ofrecer un entorno que contenga al grafo de sensores lógicos, de manera que se pueda encapsular el comportamiento del mismo. Al disponer de la posibilidad de incluir componentes de control unos dentro de otros, se crea una jerarquía de inclusiones que describe de manera funcional las áreas del control del sistema, este aspecto se verá con detalle en el siguiente subapartado.

El uso de componentes de control permite el almacenamiento de su composición y estructura mediante archivos, lo que da lugar a la reutilización de una especificación para la implementación de varias instancias del mismo componente de control.

3.2.3.1.3 Árbol de componentes de control

Dado que los componentes de control pueden incluir a otros componentes de control, las sucesivas inclusiones forman un árbol que describe intuitivamente las áreas de control que se han desarrollado.

Al árbol de componentes de control se le ha denominado *Component Control Tree* (CCT). En la Figura 30 se puede ver un sencillo ejemplo de CCT.

Un CCT tiene un nodo raíz, que representa a todo el sistema de control. A partir del nodo raíz, y a medida que se aumenta la profundidad en el CCT, se incrementa el detalle del control ya que se incrementa la especialización de los componentes de control.

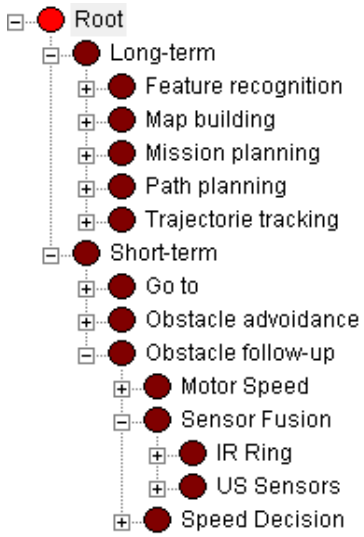


Figura 30. Ejemplo Component Control Tree.

Hay una similitud entre la posición del componente de control y el tipo de control que el componente de control realiza. Los componentes de control cercanos al nodo raíz actúan como componentes de control deliberativo, mientras que los componentes de control más alejados o finales son componentes de control reactivo. En el ejemplo de la Figura 30 se ha dividido el control en dos áreas fundamentales: largo plazo y corto plazo.

La distinción entre componente de control y agente de control es muy sutil, ya que un sensor lógico que realice un procesamiento complejo de los mensajes de entrada puede considerarse un agente de control [Shoham, 1997]. Este último aspecto se verá con detalle en el siguiente capítulo.

3.2.3.1.4 Conexiones

En la Figura 31 se muestra el modo en que se realizan las conexiones entre elementos de la arquitectura FSACtrl. Los elementos tienen conexiones entrantes y salientes, sin estar limitado el número. En todas las conexiones se negocian las características de la comunicación por medio de las políticas de calidad de servicio. Los valores negociados en las conexiones pueden variarse a lo largo de la comunicación.

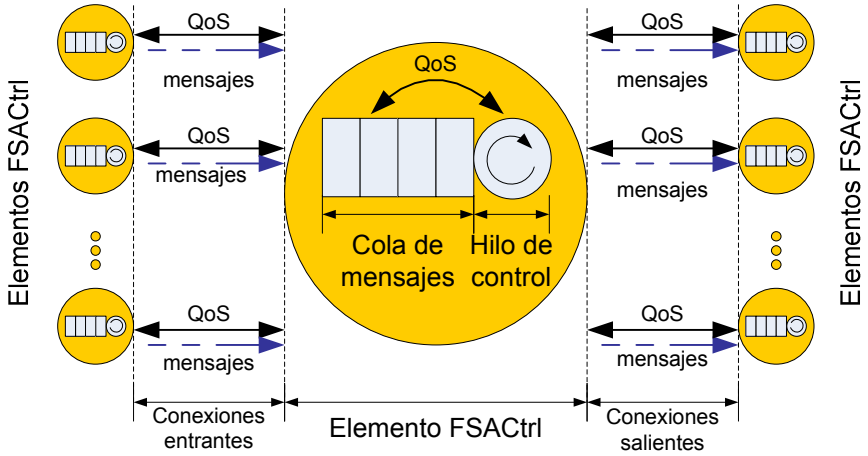


Figura 31. Elementos de FSACtrl con las conexiones entrantes y salientes.

La comunicación entre los sensores lógicos puede ser de dos tipos, dependiendo de si se encuentran en localizaciones distintas dentro del sistema distribuido, o si se encuentran en el mismo nodo. Cuando las conexiones se dan entre dos elementos localizados en dos nodos distintos, los mensajes se pasan por medio de los correspondientes canales de comunicaciones que conectan los dos nodos. Cuando las conexiones se dan en dos elementos localizados en el mismo nodo, los mensajes se transmiten por medio de los sistemas de mensajería internos que proporciona el nodo.

3.2.3.2 Especificación formal

Todos los elementos de control parten de la clase *LogicalSensor*. La clase *LogicalSensor* es la responsable de gestionar el hilo de control encargado de recibir los mensajes de entrada, procesarlos, generar el resultado y enviarlo a los sensores que estén conectados. En el caso de la arquitectura FSACtrl, se extiende el concepto de *ProcessChain* de SWE a un grafo, dado que se permiten conexiones entre elementos en cualquier sentido. En la Figura 32 se muestran las clases relacionadas directamente con *LogicalSensor*.

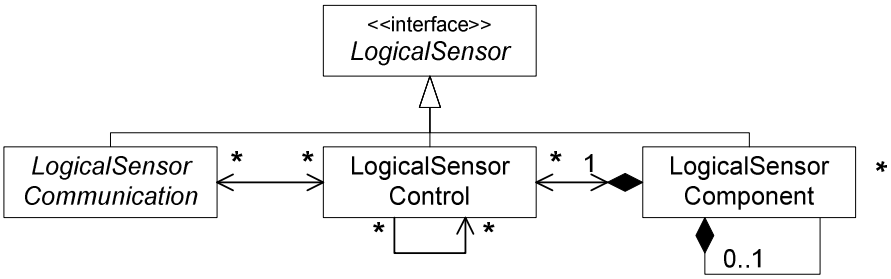


Figura 32. Clases de soporte al control de la arquitectura FSACtrl.

A partir de la clase base *LogicalSensor* se derivan tres clases con diferente funcionalidad: *LogicalSensorCommunication*, *LogicalSensorControl* y *LogicalSensorComponent*. La clase *LogicalSensorCommunication* es la clase base de las clases *DataWriter*, *DataReader* y *Listener* que implementan la gestión de las comunicaciones y sirven de interfaz entre los canales de comunicaciones (implementados por medio de las clases *Publisher* y *Subscriber*, descritas en el siguiente subapartado) y los componentes de control implementados por la clase *LogicalSensorControl*.

La clase *LogicalSensorControl* es la clase que implementa los objetos contenedores del código de control que se ejecuta a través de un hilo de control. Los sensores lógicos de control están conectados con los sensores lógicos de comunicaciones, la conexión se representa como asociación UML con los componentes de control y con los otros sensores lógicos de control. El número de sensores lógicos de comunicaciones que pueden intercambiar mensajes con los sensores lógicos de control no está limitado.

La función principal de la clase *LogicalSensorComponent* es la de contener los sensores lógicos de control y los sensores lógicos de comunicaciones para formar un componente de control. La relación de inclusión entre clases *LogicalSensorComponent* permite incluir un componente de control dentro de otro formando el CCT.

3.2.4 Clases de soporte a las comunicaciones

3.2.4.1 Modelo conceptual

3.2.4.1.1 Elementos de comunicaciones

Como se puede contemplar en la Figura 22 de la página 80, los elementos de comunicaciones se ubican en la capa de control y en la capa de comunicaciones. En la capa de control se encuentran las clases que implementan los sensores lógicos de comunicaciones: *DataReader*, *DataWriter* y *Listener*, funcionalmente similares a los propuestos en DCPS. En la capa de comunicaciones se tienen las clases que implementan a los adaptadores de comunicaciones: *Publisher* y *Subscriber*, también funcionalmente similares a los correspondientes homónimos de DCPS.

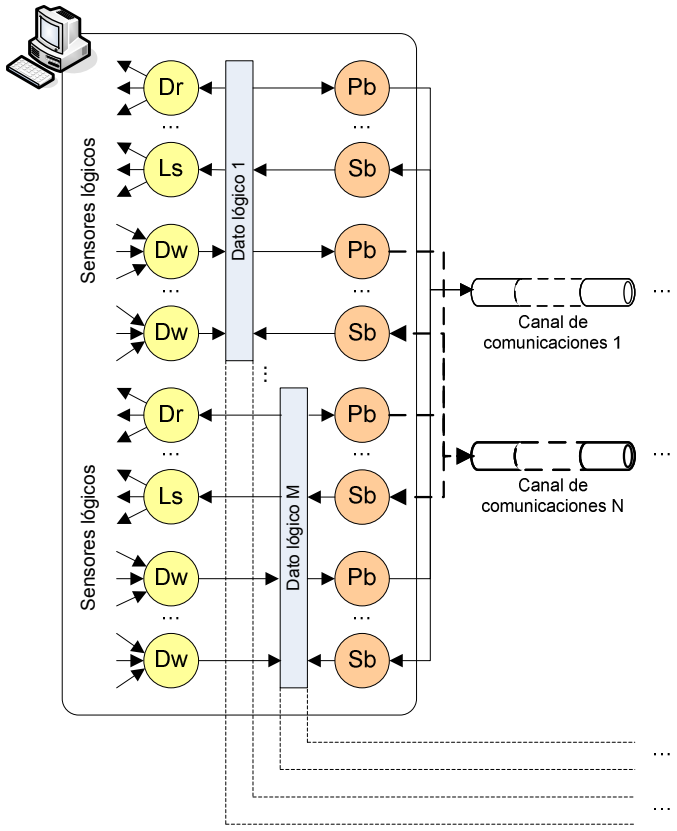


Figura 33. Elementos de comunicaciones.

Tanto los sensores lógicos de comunicaciones como los adaptadores de comunicaciones, se asocian unívocamente a un dato lógico que actúa como nexo entre ambos. El papel del dato lógico es importante, ya que todos los sensores lógicos de control del sistema distribuido que estén interesados en la información del sensor lógico que envía el mensaje están conectados al mismo dato lógico. Un solo dato lógico permite la conexión de múltiples sensores lógicos de comunicaciones con múltiples adaptadores (Figura 33).

La ruta de los mensajes a través de un canal de comunicaciones se muestra en la Figura 34. Cuando un sensor lógico de control (Ctrl en la Figura 34) desea enviar un mensaje lo hace a través de un *DataWriter* (Dw en la Figura 34) que tiene asociado un solo dato lógico por medio del cual está conectado al *Publisher* (Pb en la Figura 34), responsable de adaptar y enviar la información por el canal de comunicaciones. El *Publisher* envía el mensaje a todos los *Subscriber* (Sb en la Figura 34) que compartan dato lógico. Los *Subscriber* envían el mensaje a los *DataReader* (Dr) o *Listener* (Ls) con los que comparta dato lógico. Finalmente los *DataReader* esperarán a que el sensor de control solicite el mensaje, y los *Listener* lo enviarán directamente a los sensores de control.

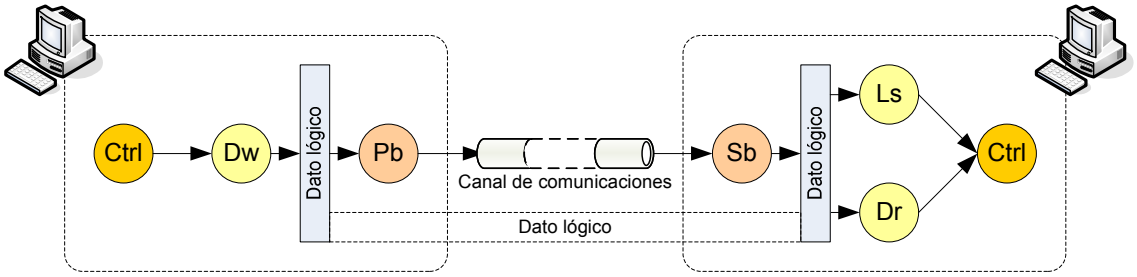


Figura 34. Conexión entre computadores por medio de FSACtrl.

Los canales de comunicaciones dependen de las características propias del canal empleado por lo que la información de configuración de la conexión de cada *Publisher* y de cada *Subscriber* es particular al medio (Figura 35).

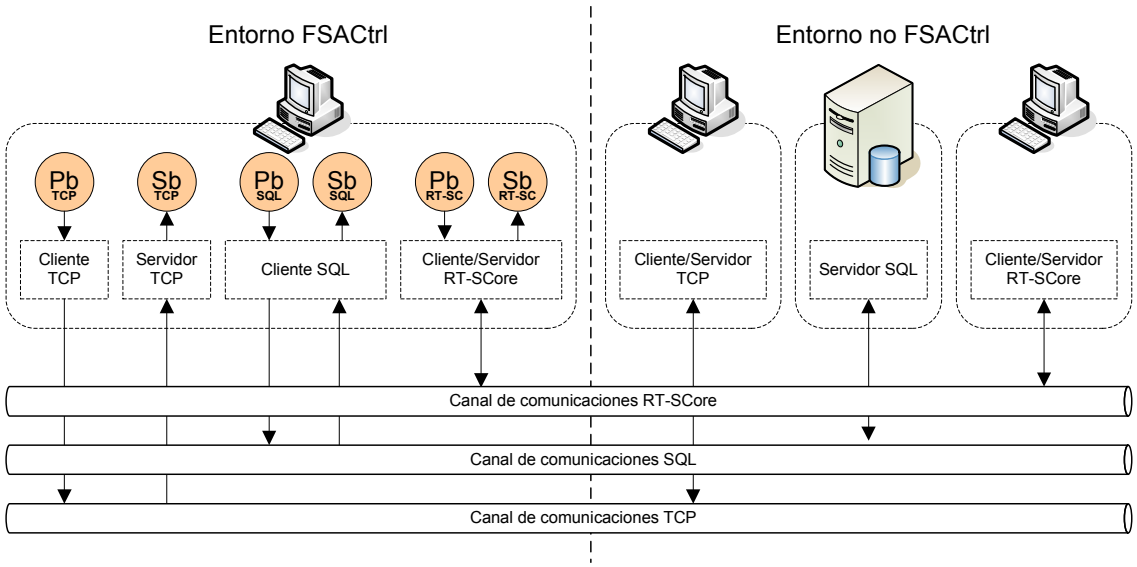


Figura 35. Conexión de adaptadores a diferentes canales de comunicaciones.

Tal como se muestra en la Figura 35, en el caso de emplear comunicaciones basadas en TCP, un *Publisher* emplea un cliente TCP que se conectará a un *Subscriber*, que a su vez está implementado por medio de un servidor TCP. No necesariamente la implementación de un *Publisher* o un *Subscriber* deber asociarse a un cliente o un servidor. Por ejemplo, si se emplea un canal de comunicaciones basado en una pizarra distribuida, como RTSCore [Posadas et al., 2002], los *Publisher* y los *Subscriber* acceden a un servicio que realiza las tareas de cliente y de servidor. En cambio, si se emplea un canal de comunicaciones de gestión de datos por medio SQL, tanto los *Publisher* como los *Subscriber* se conectan a un cliente SQL, ya que las consultas SQL son síncronas y por tanto la solicitud y la respuesta de la consulta se realizan en la misma operación.

3.2.4.1.2 Árbol de espacio de nombres lógicos

Tanto en el modelo FSA, en el que se inspira la arquitectura FSACtrl, como en la capa DCPS del modelo DDS, la información se organiza de forma abstracta en unos ítems que etiquetan y clasifican la información. En el modelo FSA se emplea un elemento llamado *dato primitivo*, mientras que en el modelo DDS se denomina *Topic*.

El modelo FSACtrl enriquece el concepto de *Topic* empleando un mecanismo similar llamado *Dato Lógico*. Los datos lógicos se organizan jerárquicamente en una estructura de datos de árbol llamada *árbol de espacio de nombres lógicos* o LNT (Logical Namespace Tree).

Un dato lógico es un canal lógico de comunicaciones. Las aplicaciones se comunican enviando mensajes mediante la escritura y lectura de los componentes asociados por medio de los datos lógicos. El nombre de un dato lógico se compone de una secuencia de palabras separadas por un símbolo. En el caso de los árboles de espacios de nombres lógicos, se ha escogido como separador el símbolo de directorio de archivos de los sistemas operativos (‘/’) y cada una de las palabras que forman el dato lógico se le ha llamado nodo lógico.

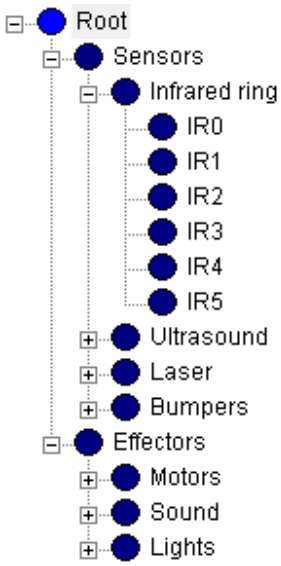


Figura 36. Ejemplo de *Logical Namespace Tree*.

En la Figura 36 se puede ver un ejemplo de LNT donde se ha dividido la información en dos áreas principales: sensores y actuadores, y a medida que se profundiza en el árbol, aumenta la concreción de la información. El nodo raíz representa el sistema completo de sensores y actuadores, mientras que los nodos terminales representan sensores o actuadores concretos Por ejemplo, en el LNT representado en la Figura 36, si se desea recibir la información del sensor de infrarrojos "IR4" basta con conectar un *DataReader* o un *Listener* al dato lógico "Root\Sensors\Infrared ring\IR4".

Una de las características del LNT es la capacidad de referirse a todos los datos lógicos contenidos en un subárbol a partir de un solo dato lógico. Para representar el subárbol se emplea el dato lógico con la secuencia de palabras desde la raíz hasta el nodo lógico del subárbol a partir del cual se desea referir a todas las ramas de datos lógicos, seguido del símbolo del asterisco ('*'). Por ejemplo, en el LNT representado en la Figura 36, si se desea recibir todos los datos del anillo de infrarrojos basta con conectar un *DataReader* o un *Listener* al dato lógico "Root\Sensors\Infrared ring*".

3.2.4.2 Especificación formal

3.2.4.2.1 Elementos de comunicaciones

Las clases responsables de las comunicaciones se organizan en dos grupos, las descendientes de la clase *Adapter* y las descendientes de la clase *LogicalSensorCommunication* expuestas en el apartado anterior.

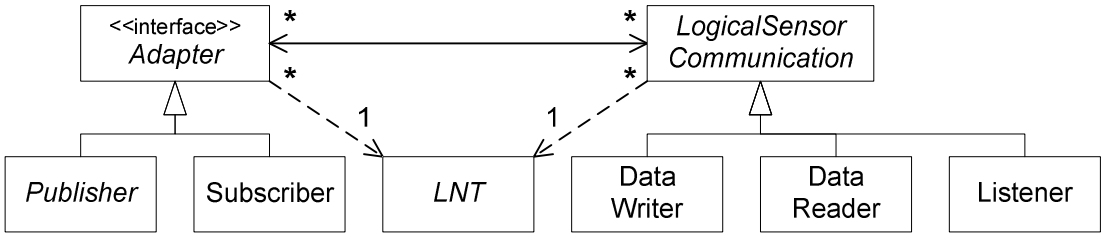


Figura 37. Clases de soporte a las comunicaciones de la arquitectura FSACtrl.

La clase *Adapter* representa la conexión del sistema con los canales de comunicación reales (Figura 37). Debe haber una clase *Adapter* por cada canal de comunicaciones que se desee emplear en el sistema. Esto se debe a que la clase *Adapter* es la que implementa los detalles del canal de comunicaciones y oculta su gestión a las clases *Publisher* y *Subscriber*, las cuales desarrollan las labores propias del estándar DCPS. Tal como se expuso anteriormente, la clase *LogicalSensorCommunication* es la clase base de los tres tipos de sensores lógicos de comunicaciones: *DataWriter*, *DataReader* y *Listener*.

3.2.4.2.2 Árbol de espacio de nombres lógicos

Las clases que implementan el LNT son *LNT* y *LogicalNode*. *LNT* es la clase que contiene el nodo raíz a partir del cual parten el resto de nodos lógicos conformando una estructura de árbol. Además la clase *LNT* se emplea como interfaz para todas las operaciones estructurales con el espacio de nombres lógicos. Para implementar los nombres de los datos lógicos, se emplea la clase *LogicalNode* y representa un nodo lógico del dato lógico.

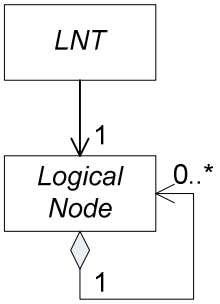


Figura 38. Diagrama de clases del espacio de nombres.

En la Figura 38 se muestran las dos clases y cómo la estructura se forma al conectarse unos nodos con otros de forma jerárquica. Debido a que la relación entre los *LogicalNode* es una agregación por asociación directa, un nodo sólo puede tener un ascendente y una cantidad ilimitada de descendentes.

3.2.5 Clases de soporte a la calidad de servicio

A todo elemento derivado de la clase *Entity* se le puede asociar una lista de políticas de calidad de servicio. Cada política de calidad de servicio se deriva de la clase *QoSPolicy*. La lista de políticas de calidad de servicio estará vacía en el caso de que no se quieran asociar políticas al elemento. Por ejemplo, es posible negociar la frecuencia de actualización de datos de un componente de control en función de las necesidades de procesamiento que se tengan o determinar si se desea recibir los mensajes en un orden concreto.

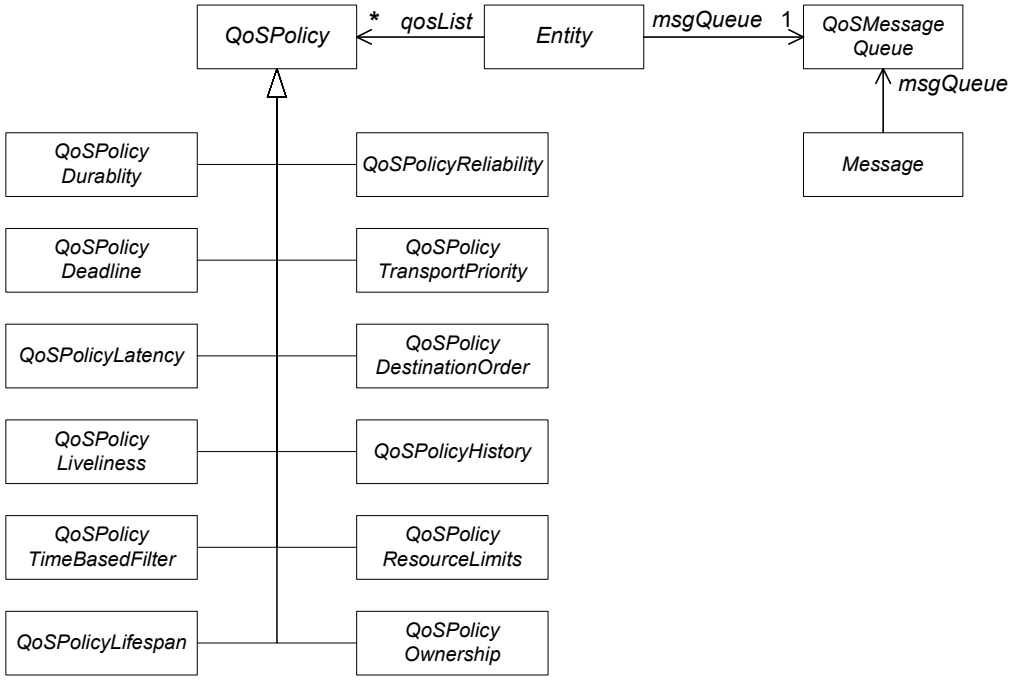


Figura 39. Clases de soporte a la calidad de servicio de la arquitectura FSACtrl.

En la Figura 39 se muestran todas las políticas de calidad de servicio que se pueden asociar a un elemento de la arquitectura FSACtrl. Las políticas seleccionadas son las pertenecientes a las áreas de gestión de los aspectos temporales y de gestión de flujo de mensajes del modelo DCPS.

La clase que da soporte a la gestión de los mensajes entre elementos, tanto de comunicaciones como de control, es *MessageQueue*. La clase *MessageQueue* gestiona los mensajes en una cola FIFO con niveles de prioridad configurables. Los mensajes sin prioridad se encolan en el mismo orden de llegada, mientras que los mensajes con prioridad son encolados en función de las mismas.

3.2.6 Clases de soporte a la calidad de control

En la arquitectura FSACtrl el soporte a la calidad de control se basa en la clase *QoCParameter*. Cada sensor lógico de control contiene una lista de parámetros de QoC por medio de la cual se accede a los parámetros para actualizarlos a la llegada de cada mensaje y ser consultados por los posibles eventos asociados (Figura 40). A partir de la clase QoCParameter se derivan las clases que dan soporte a los parámetros de calidad de control, en concreto: IAE (Integrated Absolute Error) y el parámetro ITAE (Integrated in Time Absolute Error).

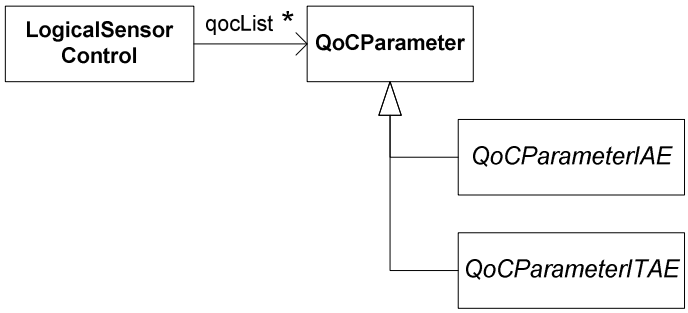


Figura 40. Clases de soporte a la calidad de control de la arquitectura FSACtrl.

La implementación de parámetros de QoC mediante clases, y no como atributos de los sensores lógicos de control, se debe a la necesidad de poder asociar los eventos de los parámetros de QoC a condiciones y poder realiza acciones a partir del cumplimiento, o incumplimiento, de la QoC especificada en el parámetro.

3.2.7 Clases de soporte a la gestión de eventos, condiciones y acciones

3.2.7.1 Modelo conceptual

Una de las características en las que basa su eficiencia el modelo de control basado en eventos es la selección de los sucesos significativos que generarán los eventos. En ocasiones no basta con detectar un evento discreto, sino que son una combinación de sucesos los que determinan el evento. Una combinación de eventos da lugar a la realización de diversas acciones asociadas al acontecimiento del evento.

Para cubrir los aspectos anteriormente citados, en FSACtrl se dispone de tres tipos de elementos (Figura 41): eventos, condiciones y acciones. Los eventos son situaciones que se monitorizan, las condiciones son agrupaciones de eventos y las acciones son los efectos en el sistema que se asocian a las condiciones.

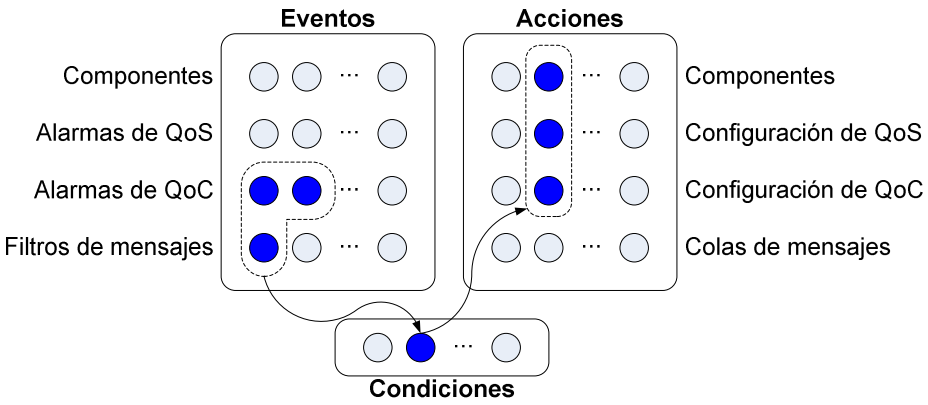


Figura 41. Elementos de la gestión de los eventos.

3.2.7.1.1 Eventos

Las circunstancias que pueden desencadenar un evento pueden ser muy variadas, en el modelo DCPS del estándar DDS se plantean principalmente las cuestiones asociadas al incumplimiento de las políticas de QoS o de errores en los elementos. Esta gestión la realiza DDS mediante los elementos *Condition* y *WaitSet*. Sin embargo, dado el carácter dinámico de la arquitectura FSACtrl es preciso ampliar el concepto planteado en DDS. A continuación se muestran las fuentes de eventos en el sistema y la descripción de los mismos.

- Eventos de elementos. Los eventos de los elementos son aquellos que se dan al ser llamado uno de sus métodos. Por ejemplo al iniciarse la actividad de control en un sensor lógico de control o al conectarse al canal de comunicaciones en un adaptador.
- Alarmas de QoS. El evento se da cuando se cumple, o no, alguno de los requisitos establecidos sobre los parámetros de una política de QoS. Por ejemplo cuando se cumple el plazo temporal o un mensaje llega antes de tiempo.
- Alarmas de QoC. El evento se da cuando se cumple, o no, alguno de los requisitos establecidos sobre los parámetros de una política de QoS. Por ejemplo cuando el contenido del mensaje genera un error mayor que el valor de referencia establecido.
- Filtros de mensajes. El evento se da cuando el contenido de un mensaje cumple, o incumple, un criterio establecido. Por ejemplo cuando un campo de un mensaje coincide con un determinado patrón.

En FSACtrl cada evento tiene asociado un estado interno que determina si se ha producido o no. El estado interno es consultado o cambiado por la condición a la que pertenece en función de las relaciones lógicas entre eventos.

3.2.7.1.1.1 Eventos relativos a elementos

Los eventos de elementos se emplean para monitorizar la actividad de éstos a lo largo del funcionamiento del sistema. Se asocian a los métodos de los elementos, y pueden ser eventos estructurales: los que se dan cuando se varía la topología, por ejemplo durante la inserción o eliminación de elementos, o eventos de funcionamiento: los que se dan durante el funcionamiento de un elemento, por ejemplo cuando se inicia o se detiene su actividad.

3.2.7.1.1.2 Alarmas de QoS y QoC

La gestión de las alarmas se basa en la monitorización de una política de QoS o de un parámetro de la QoC. La alarma puede definir un umbral superior, uno inferior o ambos umbrales.

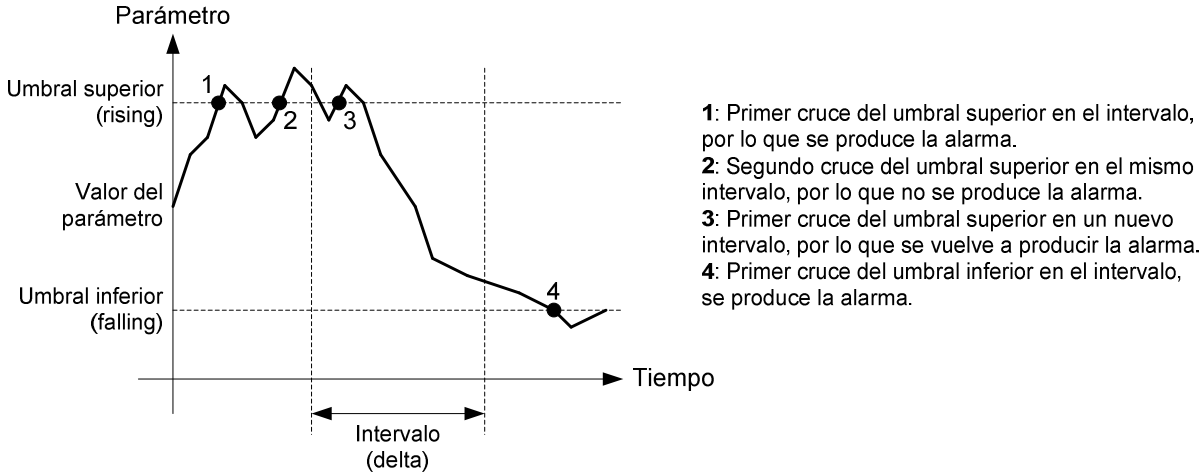


Figura 42. Definición y efecto de los umbrales en las alarmas.

La alarma se acciona cuando se sobrepasa un umbral en una dirección determinada (Figura 42). Las alarmas se definen en función de un valor absoluto para el caso de parámetros que no son incrementales, como por ejemplo el retraso de un mensaje, o en función de un valor relativo para parámetros incrementales, como por ejemplo el número de mensajes entrantes. En el caso de una alarma incremental se debe definir el intervalo de tiempo de muestreo durante el cual se debe monitorizar. El intervalo de tiempo también se emplea para configurar el tiempo de reactivación de la alarma.

3.2.7.1.1.3 Filtros de mensajes

Los eventos de filtrado son los que se aplican sobre la cola de mensajes de cada uno de los elementos. Permiten localizar mensajes entrantes a los elementos en función de su contenido. El filtrado se basa en la definición de un patrón de comparación y dos máscaras, una de coincidencia y otra de no coincidencia (Figura 43).

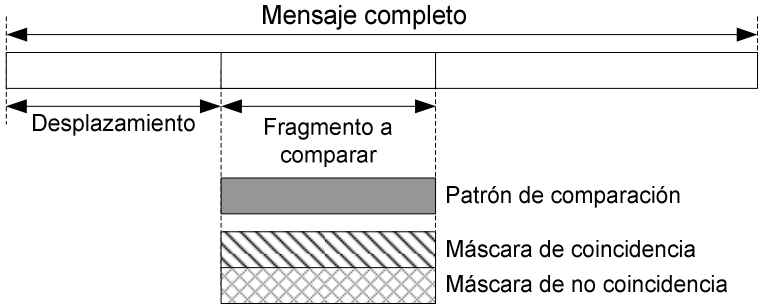


Figura 43. Definición de componentes de un filtro.

Para optimizar la aplicación del filtro, se define un desplazamiento, en bytes, desde donde se comienza el fragmento de mensaje que se compara con un patrón. Por ejemplo, si los mensajes dirigidos a un determinado elemento deben comenzar con una secuencia con un identificador del elemento que se encuentra a partir del segundo byte, el filtro deberá definir un desplazamiento de dos bytes y el patrón con la secuencia del

identificador que se desea localizar. Además se especifican dos máscaras, una de coincidencia con los bytes que se deben corresponder con el patrón y otra donde los bytes no deben corresponderse con el patrón. Por medio de las dos máscaras se determinan simultáneamente secuencias deseables junto a secuencias no deseables.

3.2.7.1.2 Condiciones

Las condiciones son agrupaciones de eventos relacionadas entre sí por medio de operaciones lógicas básicas (AND y OR). Por ejemplo, si un mensaje entra en una cola (evento de elemento) y la política de QoS deadline se ha dado (evento de QoS) previamente entonces el mensaje no debe ser procesado. Esta evaluación lógica de eventos la realiza la condición y avisa al elemento al que esté asociado para que sea este quien tome las medidas sobre el mensaje. Las acciones, que se verán a continuación, también pueden actuar sobre el elemento para descargar al programador de esta labor.

Es importante destacar que las condiciones tienen una prioridad para los casos de condiciones excluyentes, por ejemplo si una condición desea iniciar un elemento y otra desea detenerlo. El establecimiento de prioridades corresponde al diseñador del sistema. A continuación se detallan los elementos y la forma de interaccionar entre ellos.

Las condiciones son similares conceptualmente a las alarmas y filtros empleados en el protocolo de monitorización RMON (Remote Network MONitoring) [Waldbusser, 2000], debido a su sencillez y a la orientación de comunicaciones. De esta manera se logra una extensión del concepto *Condition* de DCPS empleándolo para la detección de los eventos significativos relacionados con el sistema y para desencadenar, si se requiere, la correspondiente acción.

3.2.7.1.3 Acciones

Una condición puede llevar asociada un conjunto de acciones a tomar en el caso de que la condición se de. Las acciones pueden ser de diversos tipos:

- Acciones de elementos. Son las que actúan sobre los elementos o las conexiones entre ellos.
- Acciones de QoS y de QoC. Son aquellas que actúan sobre una política de QoS o sobre un parámetro de QoC.
- Acciones de colas de mensajes. Son las acciones que actúan sobre las colas de mensajes.

3.2.7.1.3.1 Acciones de los elementos y de sus conexiones

Las acciones asociadas a los elementos son las que desencadenan debido que se ha llamado a un método del elemento como por ejemplo el inicio o la detención del procesamiento del sensor lógico. También es posible añadir o eliminar políticas de QoS a cualquier elemento o parámetros de QoC a un sensor lógico. Cuando se inserta una política de QoS o un parámetro de QoC se deben definir los valores por defecto con los que se iniciará por medio de una acción de política de QoS o de parámetro de QoC respectivamente. Los elementos de control pueden llevar asociados acciones de creación o eliminación de sensores lógicos. Las acciones de conexiones consisten en crear, o eliminar, una conexión entre dos elementos, cuando se de la condición, definiendo el elemento fuente y el elemento destino.

3.2.7.1.3.2 Acciones de las políticas de QoS y de los parámetros de QoC

Las acciones de QoS son aquellas que cambian las características de una política de QoS o un parámetro de QoC ya existente. En el caso de las políticas de QoS, la acción consiste en el cambio de los parámetros de la política. En el caso de parámetros numéricos el cambio puede ser un incremento, o decremento, de una cantidad determinada o la inserción de un valor absoluto directamente.

3.2.7.1.3.3 Acciones asociadas a las colas de mensajes

Las acciones asociadas a las colas de mensajes son las que permiten variar el comportamiento de las mismas. En el caso de la arquitectura desarrollada se plantean las funciones de apertura y cierre de las colas, lo que permite rechazar mensajes. Además también es posible el vaciado de la cola de mensajes lo que forma parte del mecanismo de reinicio del componente y permite desechar mensajes que no serán procesados.

3.2.7.2 Especificación formal

La Figura 44 muestra las clases relacionadas con la gestión de eventos. Todo elemento puede tener asociado tantas condiciones como se considere necesario. Las condiciones se agrupan en una lista de condiciones. Cada condición tiene a su vez una lista de eventos y una lista de acciones.

La clase *Condition* es la responsable de gestionar los eventos y las relaciones lógicas (AND y OR) entre ellos agrupándolos en una lista interna. La clase base *Event* permite asociar en la lista de eventos de una condición eventos de distinto tipo para así poder relacionarlos.

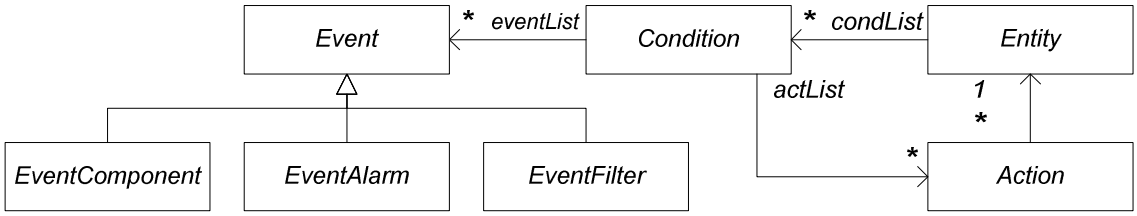


Figura 44. Clases de soporte a la gestión de eventos de la arquitectura FSACtrl.

Las acciones están asociadas a la condición por medio de la correspondiente lista. Una acción sólo puede tener asociada un elemento sobre el que actuará, mientras que un elemento puede tener tantas acciones como sea necesario.

3.3 Operaciones de la arquitectura FSACtrl

3.3.1 Gestión de elementos del sistema

3.3.1.1 Elementos mínimos del sistema

Un sistema diseñado a partir de la arquitectura FSACtrl contiene unos elementos mínimos iniciales a partir de los cuales se desarrolla el sistema. En la Figura 45 se observa la secuencia de operaciones que se realizan cuando se crea un sistema basado en la arquitectura FSACtrl.

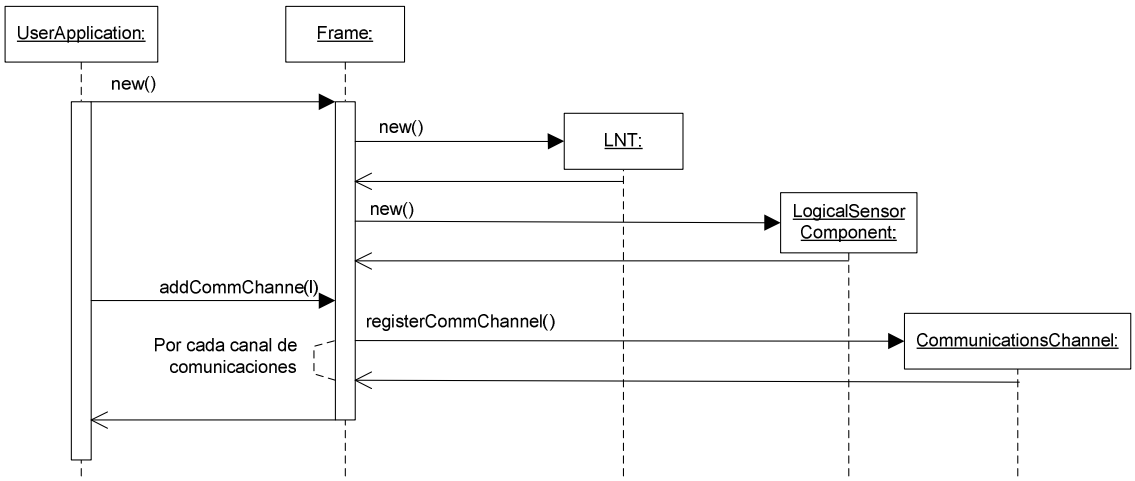


Figura 45. Diagrama de secuencia de la creación de un sistema basado en FSACtrl.

Inicialmente, el *Frame* crea el LNT con un único nodo (*Root*) del cual descenderán el resto de los nodos lógicos que formarán los datos lógicos. También se inserta el componente de control principal (*Root*), que representa el sistema de control. El resto de componentes de control formados por los sensores lógicos de control estarán incluidos dentro del nodo raíz. Finalmente, se insertan los canales de comunicaciones que se emplearán en el sistema.

3.3.1.2 Inserción de elementos en el sistema

La inserción de los elementos consiste en añadir adaptadores, datos lógicos y sensores lógicos al sistema, incluidos los componentes de control. En la Figura 46 se muestra el diagrama de secuencia de la inserción de los elementos en el sistema. La inserción de un elemento se solicita al *Frame*. No existen restricciones en cuanto al orden de inserción.

Para poder albergar sensores lógicos, es necesario insertar un componente de control que los albergue. En cuanto se tenga un componente de control se pueden crear los sensores lógicos de comunicaciones (*DataReader*, *DataWriter* y *Listener*) o los sensores lógicos de control necesarios.

Una vez se han creado los sensores lógicos se realizan las conexiones entre ellos, tanto si son sensores lógicos de control como si se trata de sensores lógicos de comunicaciones. La conexión se realiza por medio de la función *registerEntity* que añade en la lista de destinatarios de los mensajes de la entidad origen a la entidad destinataria conectada. La implementación de esta conexión es una cuestión interna de la aplicación que implemente la arquitectura.

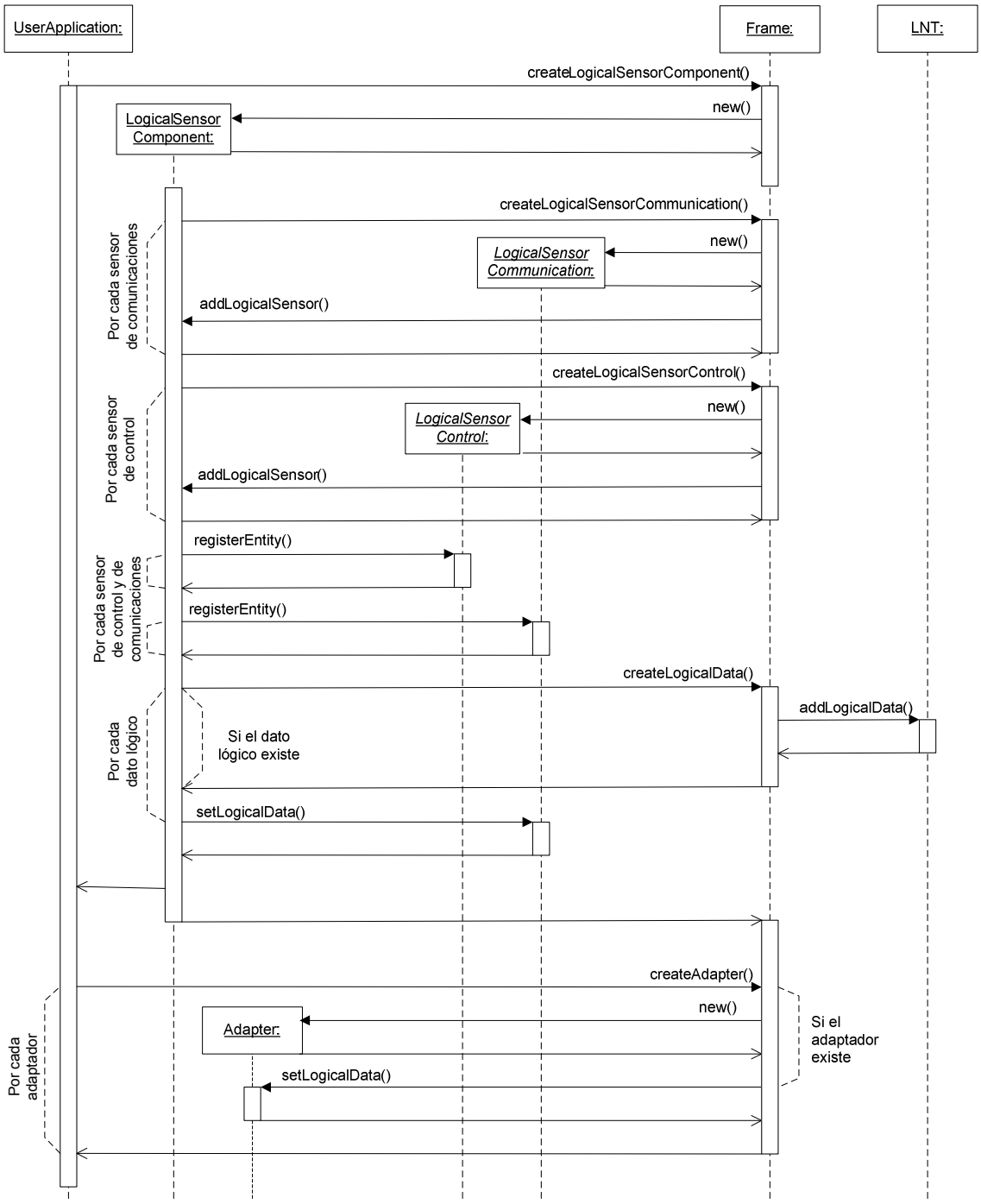


Figura 46. Diagrama de secuencia de la inserción de elementos.

El siguiente paso consiste en conectar los datos lógicos necesarios con los sensores de comunicaciones. Si los datos lógicos no existen, se solicita su creación al *Frame*. Finalmente se crean los adaptadores tanto los *Publisher* como los *Subscriber*, caso de no existir, y se les asocian a los datos lógicos correspondientes. Una vez insertados todos los elementos necesarios para funcionar, el sistema está preparado para comenzar a procesar los mensajes.

3.3.1.3 Modificación de elementos

La modificación de elementos de FSACtrl se da cuando se desea cambiar alguna de las características del elemento lo que provoca efectos en los componente de control y, por extensión en el resto del sistema. La modificación puede darse por iniciativa de usuario o como consecuencia de una acción. Las fases de la modificación se muestran en la Figura 47.

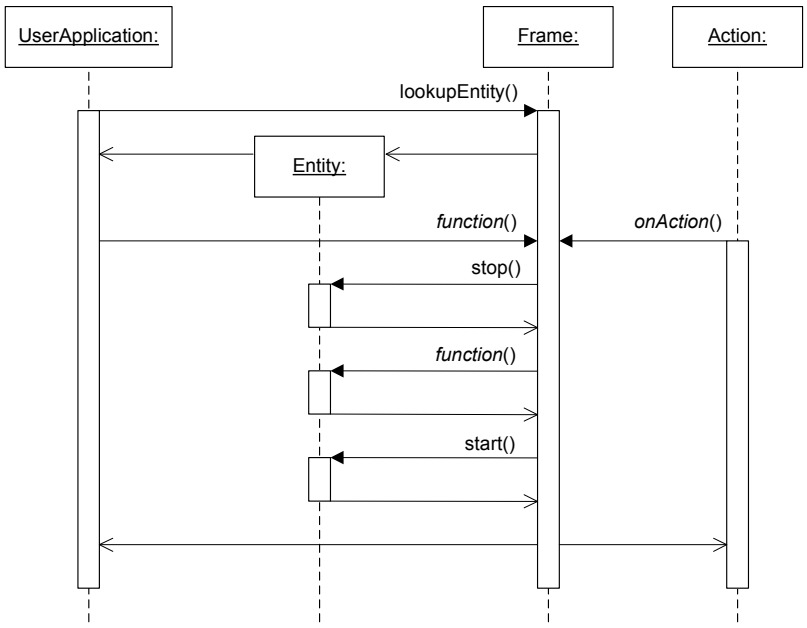


Figura 47. Diagrama de secuencia de la modificación de elementos.

Cuando la modificación es a iniciativa del usuario, se debe solicitar el acceso al elemento mediante la operación *lookupEntity*, una vez el *Frame* localiza el elemento, se solicita la modificación. Si la modificación la realiza una acción, entonces se solicita al *Frame* directamente, ya que las acciones tienen la referencia del elemento sobre el que se aplican. Para realizarla, se detiene primero el elemento, se realiza la modificación correspondiente y finalmente se pasa a reanudar la actividad del elemento modificado.

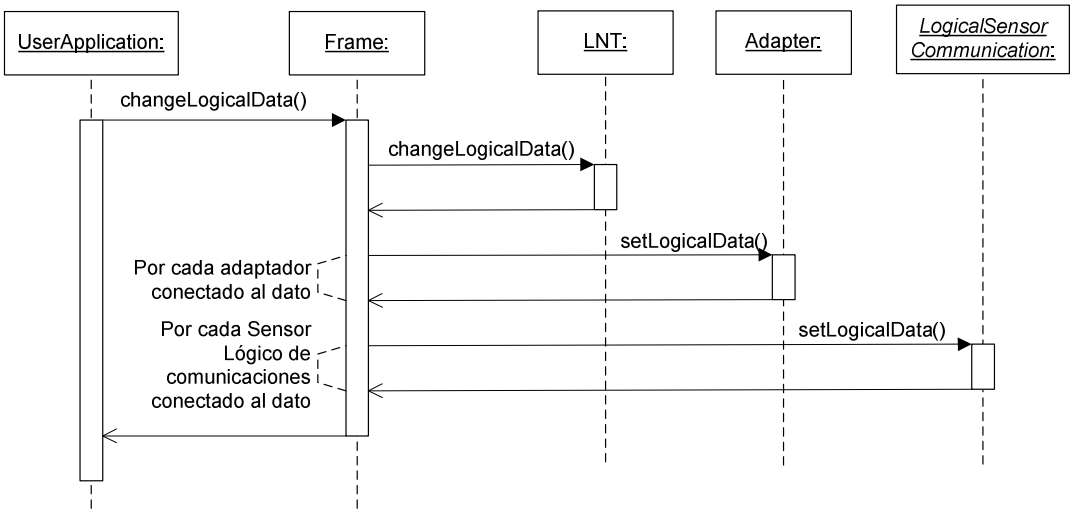


Figura 48. Diagrama de secuencia de la modificación de datos lógicos.

En el caso concreto de un dato lógico, la única modificación posible es el cambio de nombre del dato, en este caso la operación se realiza a iniciativa del usuario quien solicita el cambio al *Frame*. El *Frame* cambia el dato y actualiza el cambio a los elementos de comunicaciones que lo compartiesen (Figura 48).

En el caso de cambio de nombre de dato lógico, no se detienen los elementos involucrados ya que el dato lógico se emplea para saber qué elemento de comunicaciones debe conectarse a otro elemento de comunicaciones, una vez hecha la conexión la transferencia de mensajes es directa entre los elementos conectados.

3.3.1.4 Eliminación de elementos

La eliminación de elementos consiste en solicitar al *Frame* que lo elimine. Esta eliminación no tiene efectos que deban implicar la eliminación del resto de elementos. La eliminación de componentes de control implica tener que realizar una eliminación controlada, ya que un adaptador (*Publisher* o *Subscriber*) o un dato lógico pueden estar siendo compartidos con otro componente de control. Dado que es el *Frame* quien contiene la información compartida será el responsable de las comprobaciones y de las eliminaciones controladas. El diagrama de secuencia de esta operación se muestra en la Figura 49.

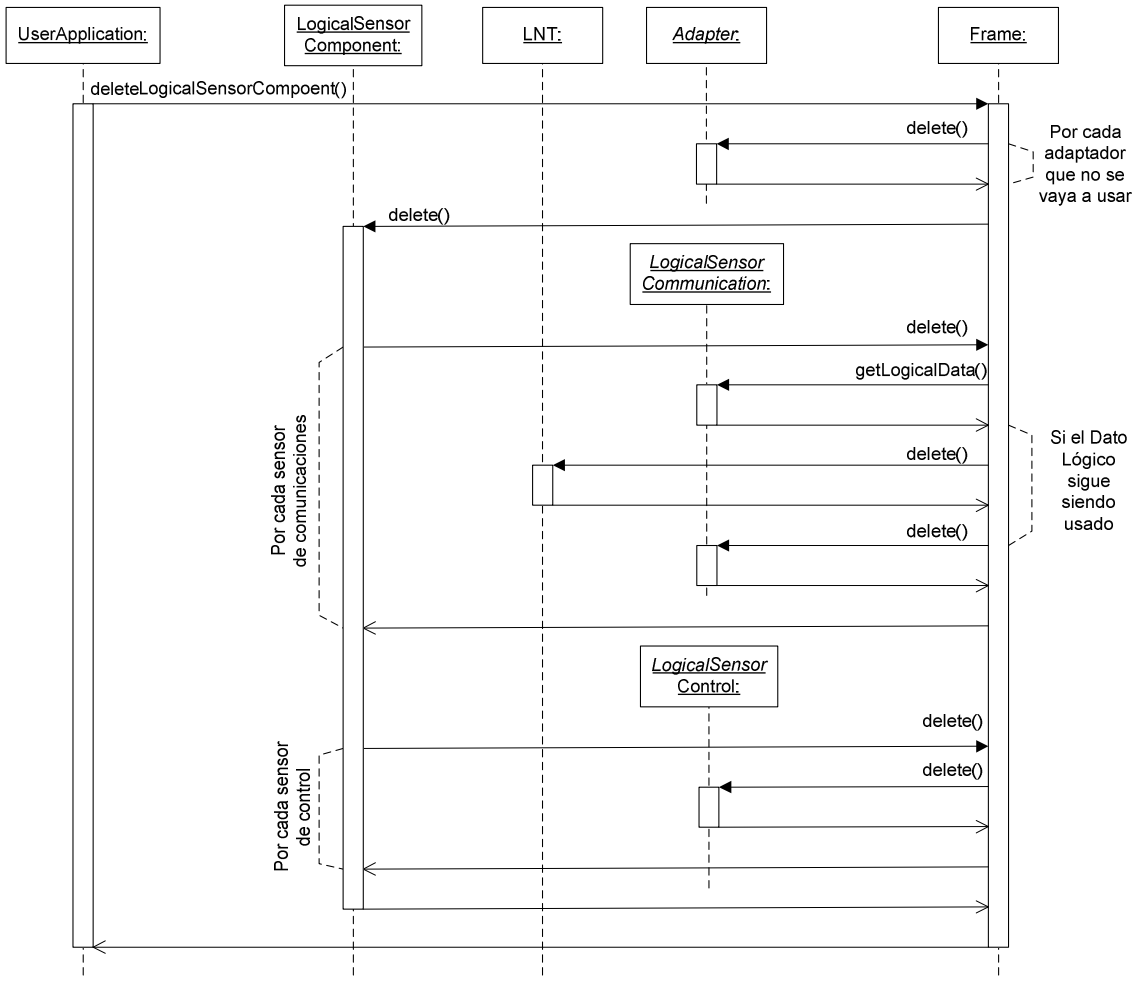


Figura 49. Diagrama de secuencia de la extracción de elementos.

El *Frame* comprueba los adaptadores que únicamente eran utilizados por el componente de control y pasa a eliminarlos. Seguidamente por cada sensor de comunicaciones del componente de control se comprueba si el dato lógico que empleaba el elemento eliminado sigue siendo usado por otros sensores de comunicaciones de otros componentes de control. En el caso de que el dato lógico no lo use ningún elemento, es eliminado. A continuación se eliminan los sensores lógicos de control del componente de control y finalmente se elimina el propio componente de control.

3.3.2 Gestión de políticas de QoS

El hecho de que todos los elementos de la arquitectura puedan disponer de colas de mensajes y de un control temporal de su procesamiento hace que se puedan obtener los parámetros de calidad de servicio, tanto de forma individual como del conjunto de un componente de control. La obtención de los parámetros de calidad más complejos es responsabilidad de los componentes de control y, en última instancia, será el *Frame* el que obtendrá los parámetros globales de optimización del sistema.

Entre algunas configuraciones de políticas de calidad de servicio pueden aparecer restricciones recíprocas, de tal forma que la configuración de una política puede ser incompatible con la configuración de otra política. Las restricciones de configuración de las políticas de calidad de servicio pueden darse tanto internamente en un mismo elemento como entre las conexiones entre dos o más elementos. En el último caso, al conectar dos elementos las posibles restricciones entre las políticas de calidad de servicio se extienden a todas las rutas de mensajes dentro del mismo componente de control. Las restricciones entre políticas de calidad de servicio pueden consultarse con detalle en [Poza, 2009c].

3.3.2.1 Inserción de políticas de QoS

Para insertar una política de QoS a un componente del sistema, se le solicita al *Frame* la creación de la política, a continuación se configuran los parámetros de la política, a la configuración del conjunto de parámetros de una política de QoS se le conoce como estado. El estado de una política de QoS se especifica por medio de la función *setState*. A continuación se detiene el funcionamiento del elemento por medio de la función *stop*, para mantener una coherencia en el procesamiento de los mensajes. Se agrega la nueva política de QoS al elemento por medio de la función *addQoSPolicy* y finalmente se reinicia el elemento por medio de la función *start*. En la Figura 50 se muestra el diagrama de secuencia de la inserción de las políticas de QoS

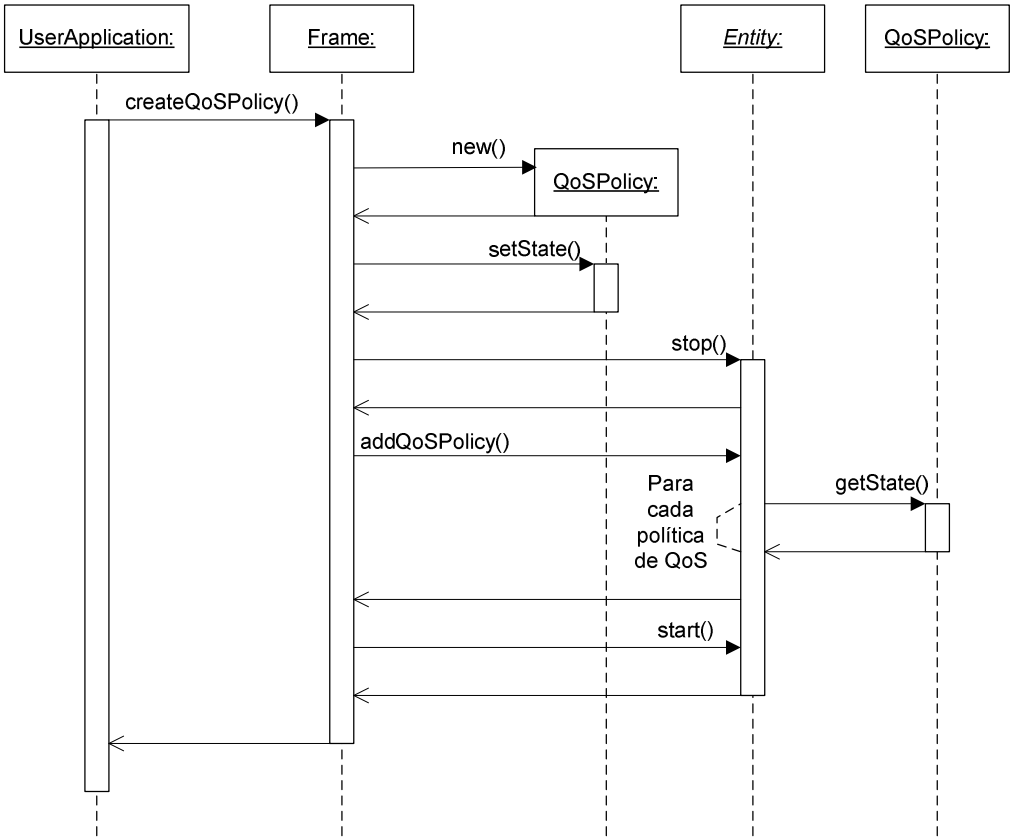


Figura 50. Diagrama de secuencia de la inserción de las políticas de QoS.

Cuando a un elemento se le agrega una política de QoS, se debe evaluar el impacto que la nueva política pueda tener en el resto de políticas de QoS y así evitar las incompatibilidades dentro de un mismo elemento. Caso de encontrar una incompatibilidad, se avisa por medio del estado de la política de QoS, que el elemento deberá procesar y avisar al elemento o usuario que solicitó la inserción de la política para que la incompatibilidad sea atendida.

3.3.2.2 Modificación de políticas de QoS

A lo largo del funcionamiento del sistema, puede ser conveniente modificar la configuración de las políticas de QoS de un elemento. Las modificaciones implican una serie de operaciones a realizar mostradas en la Figura 51.

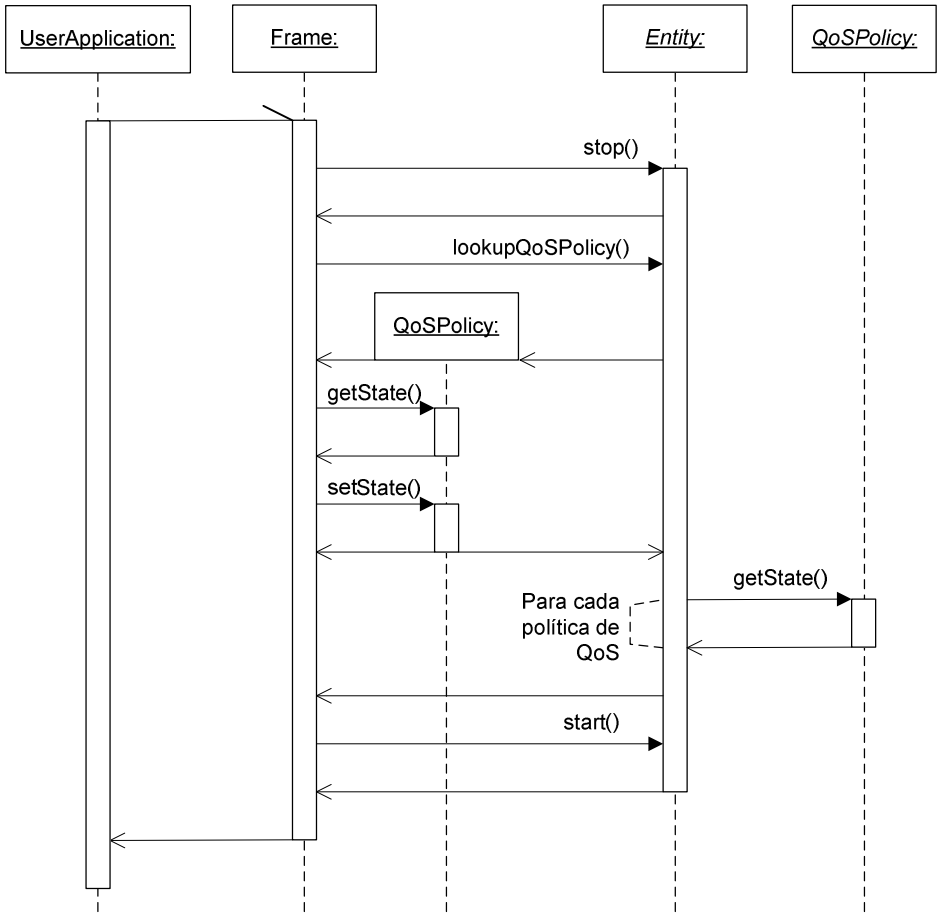


Figura 51. Diagrama de secuencia de la modificación de las políticas de QoS.

Para realizar modificaciones de la configuración de QoS de un elemento, lo primero es detenerlo por medio de la función *stop*. Una vez detenido, el elemento o usuario que desea modificar la política de QoS debe acceder a la política por medio de la función *lookupQoSPolicy*, si es necesario se leerá la configuración por medio de la función *getState* y se realizarán los cambios de configuración necesarios por medio de la función *setState*. A continuación el elemento debe comprobar que la nueva configuración es compatible con las otras políticas de QoS que el elemento pueda tener asociadas, para ello accede al estado de cada una de las políticas de QoS restantes para comprobar la compatibilidad.

3.3.2.3 Eliminación de políticas de QoS

La eliminación de las políticas de calidad de calidad de servicio, comienza solicitando al *Frame* la acción correspondiente (Figura 52). El *Frame* detiene al elemento para evitar la ambigüedad en los mensajes que se vayan recibiendo. Se elimina la política de QoS solicitada y se pasa a evaluar cada una de las políticas de QoS y condiciones restantes del elemento para comprobar el impacto que la eliminación pueda tener.

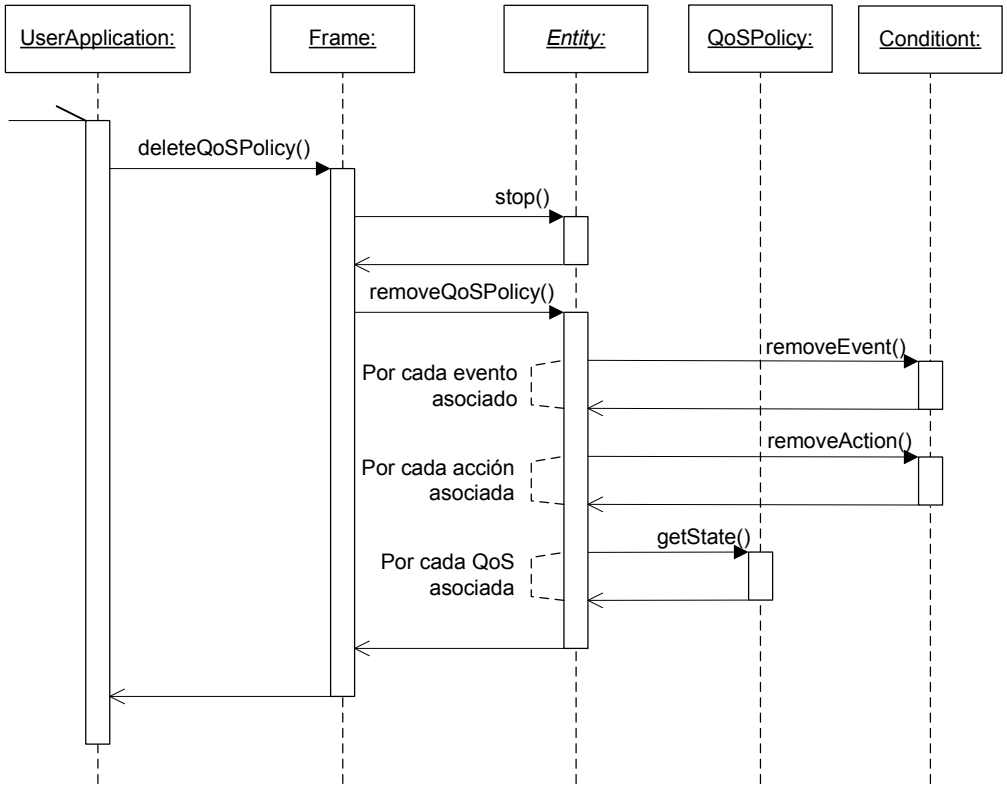


Figura 52. Diagrama de secuencia de la eliminación de las políticas de QoS.

En el caso de que la política de QoS tuviese asociada eventos o acciones, estos elementos se eliminarán consecuentemente, ya que se podría dar la circunstancia de que una condición dependiese de un estado de la política de QoS eliminada que, obviamente nunca se iba a dar. La eliminación se solicita a la condición ya que es este elemento quien gestiona los eventos y las acciones.

3.3.3 Gestión de los parámetros de QoC

3.3.3.1 Inserción de parámetros de QoC

Para insertar un parámetro de QoC, se le solicita la creación del mismo al *Frame* que lo crea y a continuación lo configura por medio de *setState*. A continuación, se debe detener, para evitar que la transición entre mensajes influya en los valores del parámetro. Seguidamente se asocia el parámetro de QoC al elemento, para finalizar volviendo a poner en marcha el elemento.

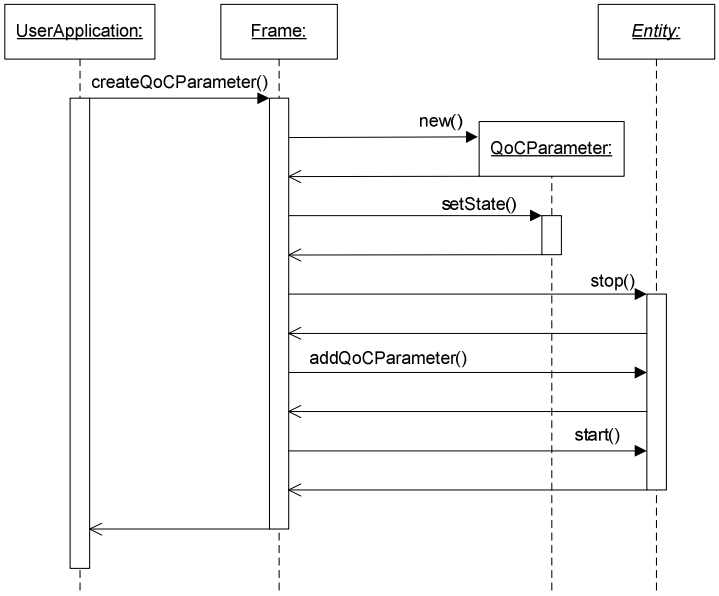


Figura 53. Diagrama de secuencia de la inserción de parámetros de QoC.

3.3.3.2 Modificación de parámetros de QoC

La modificación de un parámetro de QoC comienza por detener el elemento al que está asociado. Una vez detenido, se recupera el parámetro para leer el estado por medio de *getState* y poder cambiarlo, por medio de *setState*. Finalmente se reanuda la actividad del elemento.

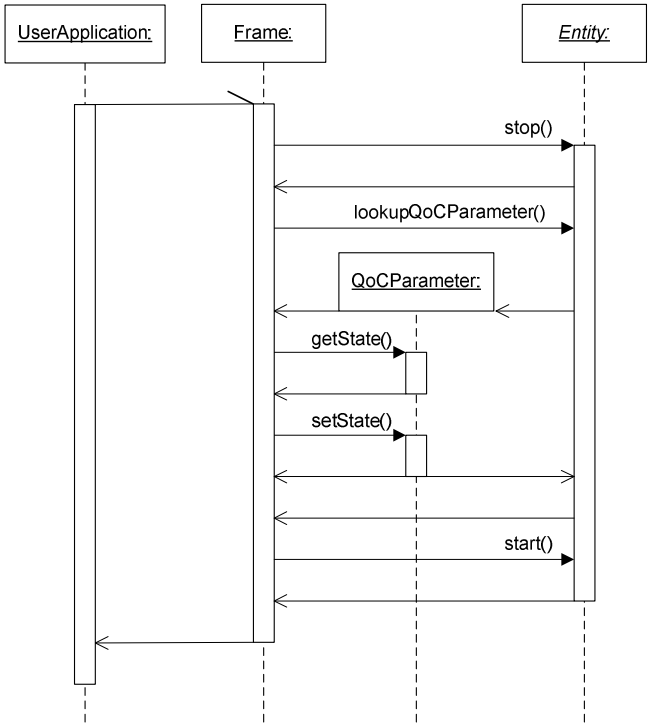


Figura 54. Diagrama de secuencia de la modificación de parámetros de QoC.

Al igual que las políticas de QoS, cuya modificación puede generar incompatibilidades con otras políticas, los parámetros de QoC pueden generar problemas caso de que se emplee para realimentar una señal de control.

3.3.3.3 Eliminación de parámetros de QoC

La eliminación de un parámetro de QoC se inicia con la detención del elemento. Una vez detenido se solicita la eliminación al *Frame*. Previamente a la eliminación, se debe revisar los eventos y las acciones asociadas al elemento, tal como se hacía con las políticas de QoS.

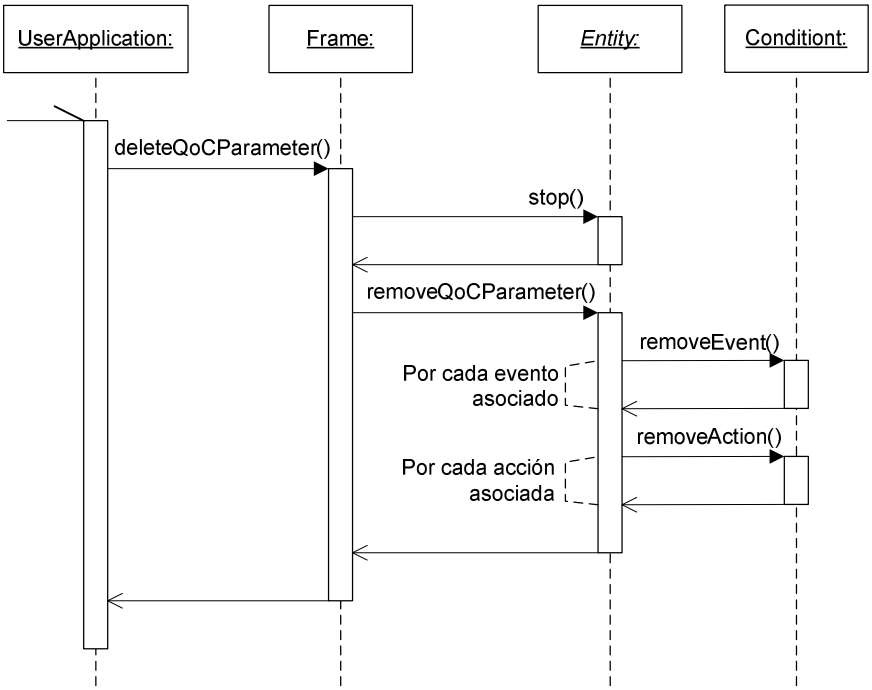


Figura 55. Diagrama de secuencia de la eliminación de parámetros de QoC.

3.3.4 Gestión de eventos, condiciones y acciones

3.3.4.1 Inserción de eventos, condiciones y acciones

Como se explicó anteriormente, los eventos y las acciones están relacionados por medio de las condiciones. Para insertar, tanto un evento como una acción se debe tener previamente una condición a la que asociarse. El proceso de inserción de las condiciones, de los eventos y de las acciones asociadas se muestra en la Figura 56.

Inicialmente debe detenerse el elemento para evitar que se una condición incompleta pueda desencadenar acciones no esperadas. Para insertar una condición, se le solicita al *Frame* la creación de la misma. Seguidamente se configura el tipo de condición por medio de la función *setState*.

Una vez se tiene la condición se pasa a crear los eventos que se detectarán y combinarán. La secuencia en la creación del evento comienza con la configuración por medio de la función *setState*, para seguidamente solicitar al elemento que lo incorpore a la condición correspondiente por medio de la función *addCondition*.

Cuando se desea asociar un evento a un elemento, se debe decidir qué tipo de evento se requiere. Los eventos asociados a las políticas de QoS se pueden asociar a cualquier elemento. Los incumplimientos de la QoC sólo se pueden asociar a los sensores lógicos de control. Los eventos relativos a los elementos y los relativos a las colas de mensajes pueden asociarse a cualquier elemento del sistema.

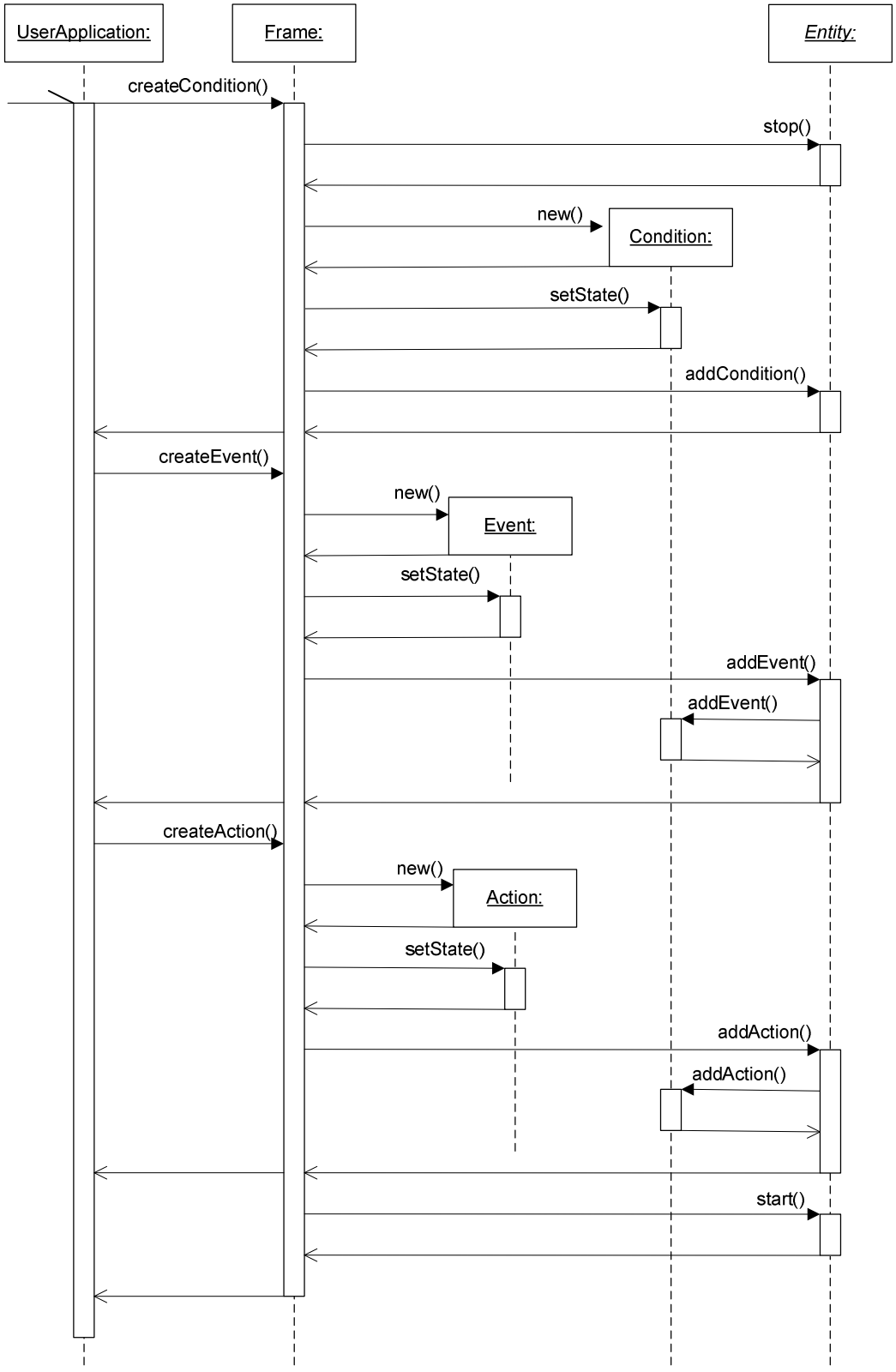


Figura 56. Diagrama de secuencia de la inserción de eventos, condiciones y acciones.

El proceso de creación y asociación de acciones a una condición es similar al proceso descrito para los eventos. Se debe crear la acción, configurarla y asociarla a la condición. La única salvedad es que las acciones no están relacionadas entre sí de la forma en que lo están los eventos.

Una vez todo el proceso finaliza se activa al componente por medio de la función *start*, de esta forma el elemento vuelve a procesar mensajes con la nueva combinación de eventos de la condición y las nuevas acciones asociadas.

3.3.4.2 Modificación de eventos, condiciones y acciones

La secuencia de acciones necesarias para modificar un evento, una condición o una acción se muestran en la Figura 57. La secuencia se inicia con la detención del elemento, al que está asociada la condición, mediante la función *stop*.

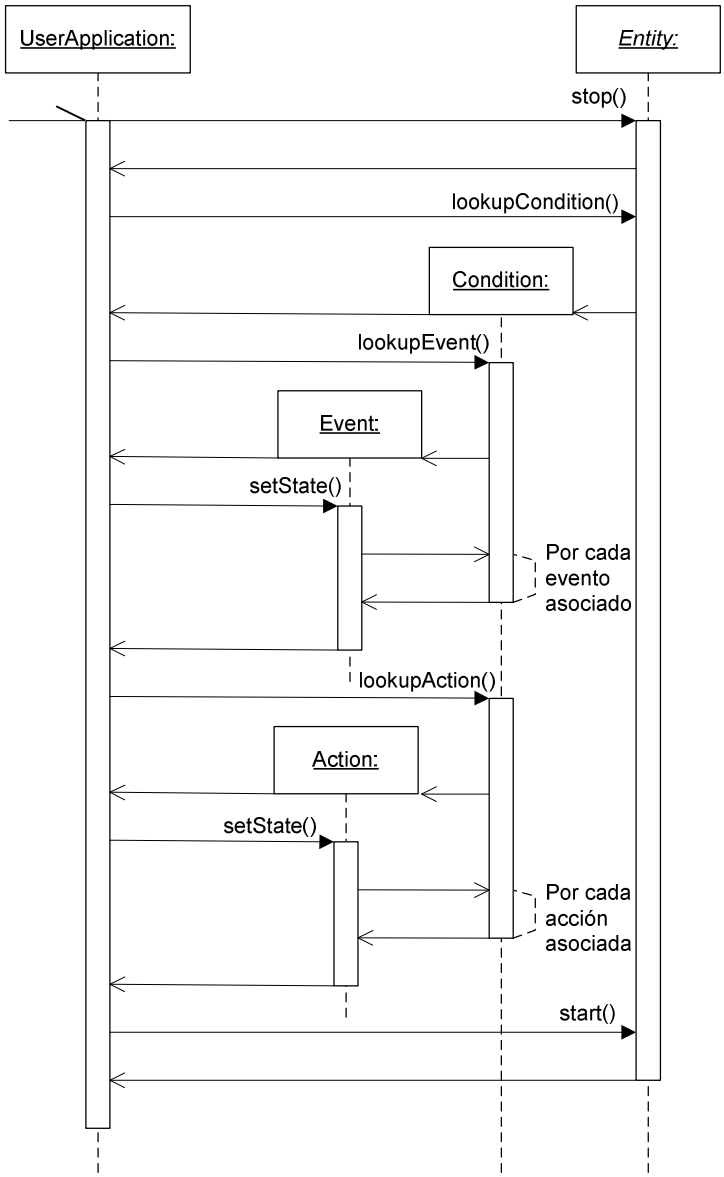


Figura 57. Diagrama de secuencia de la modificación de eventos, condiciones y acciones.

Una vez detenido se accede a la condición a la que esté asociado el evento o la acción a modificar. La modificación del evento o de la acción se realiza modificando su estado. Cuando se modifica la configuración de un evento o de una acción, la condición comprueba la compatibilidad de los cambios con los eventos o acciones que estén relacionados en la condición. Finalmente se reinicia el elemento por medio de la función *start*.

3.3.4.3 Eliminación de eventos, condiciones y acciones

La eliminación de una condición se realiza por medio de la secuencia mostrada en la Figura 58. La primera acción consiste en detener el elemento, para pasar a eliminar la condición. La eliminación de la condición se realiza eliminando los eventos y las acciones, y una vez eliminados se elimina la condición.

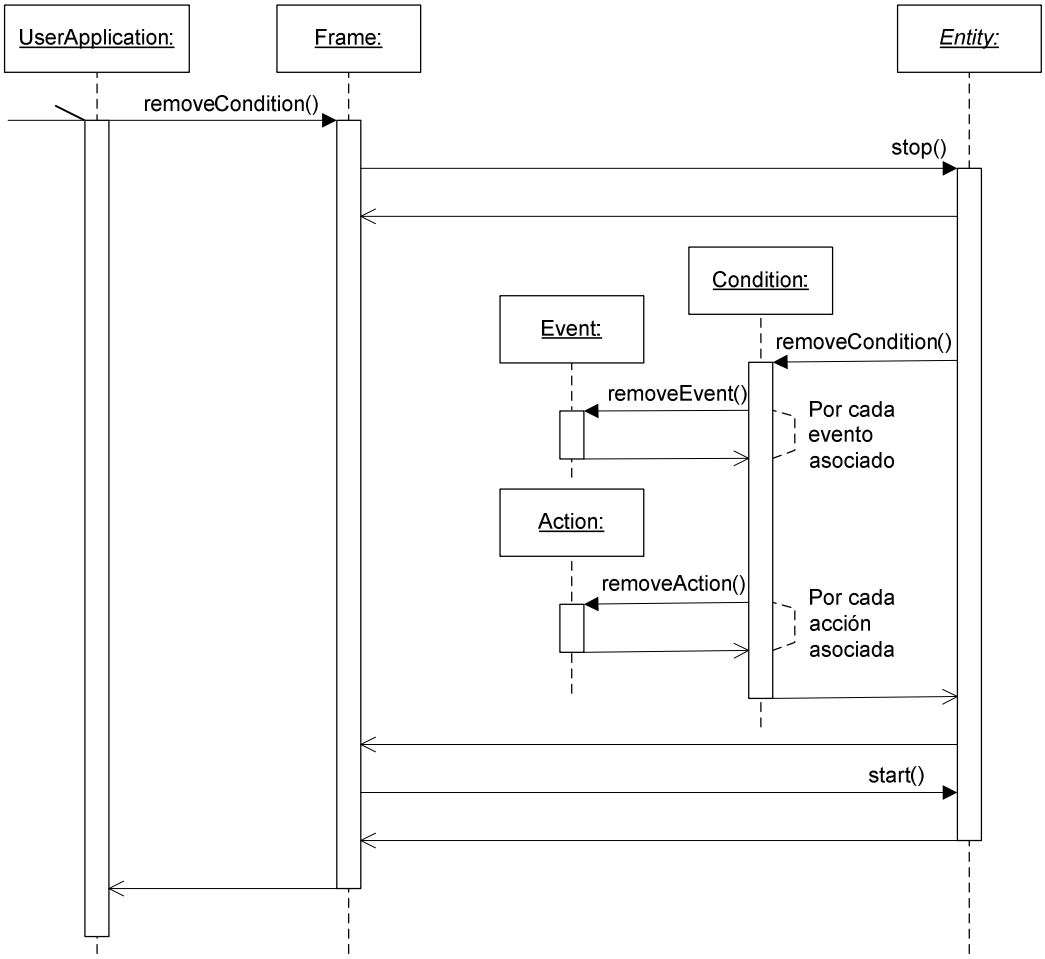


Figura 58. Diagrama de secuencia de la eliminación de eventos, condiciones y acciones.

La eliminación aislada de un evento o una acción, se realiza por medio de la misma secuencia mostrada en el apartado anterior, es decir sustituyendo la función *setState* por *removeEvent* o *removeAction*. En cualquiera de los casos anteriores, el proceso finaliza cuando se reinicia el elemento por medio de la función *start*.

3.3.5 Inicio y detención del sistema

El inicio del sistema es la acción mediante la cual los elementos comienzan a realizar las funciones para los que han sido programados. En el caso de las comunicaciones se deben realizar las tareas de conexión controlada con los canales correspondientes, mientras que en el caso del control se deben iniciar los hilos de control que procesarán los mensajes y los enviarán a los elementos correspondientes. La detención del sistema es el proceso inverso por medio del cual los elementos dejan de realizar la función de un forma controlada. Dado que el inicio del sistema es muy similar a la detención se tratarán conjuntamente ambas operaciones.

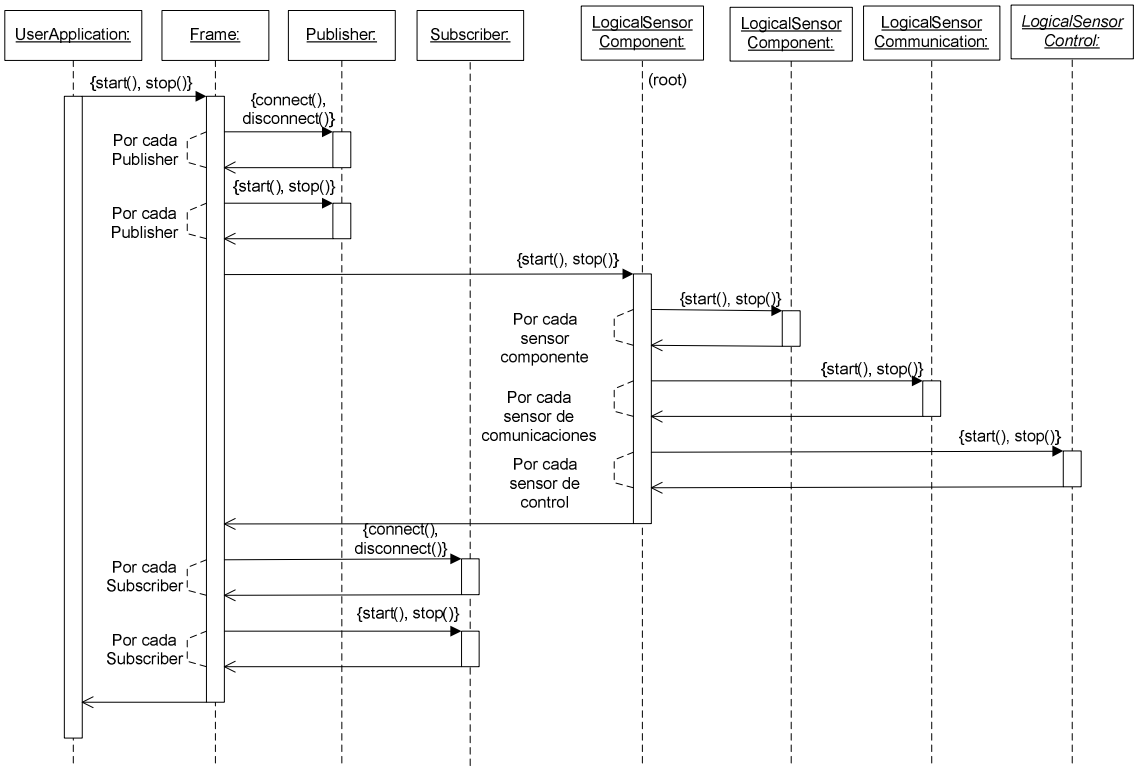


Figura 59. Diagrama de secuencia de la iniciación y detención del sistema.

En la Figura 59 se muestra el diagrama de secuencia correspondiente a la iniciación y detención del sistema, junto a las clases responsables de la gestión de los mensajes. Cuando se inicia un elemento, automáticamente se inicia su subsistema de cola de mensajes, se comprueba el resultado y se devuelve el control al marco. Las funciones que ponen en marcha o detienen el sistema son las funciones *start* y *stop* respectivamente. El *Frame*, como contenedor de todos los elementos de control y de comunicaciones del sistema, es quien recibe la orden de inicio, por medio de la función *start*. El primer efecto del inicio del sistema es la puesta en marcha de los sistemas de comunicaciones. El *Frame* llama a la función *connect* de los elementos *Publisher*, que por medio de esta función ponen en marcha sus conexiones con los canales de comunicaciones que gestionan lo que implica que el sistema ya puede interactuar con el sistema de comunicaciones enviando mensajes.

Una vez que se conoce el resultado de la conexión de los elementos *Publisher*, el *Frame* pone en marcha el sistema asociado a los mensajes que gestionará cada *Subscriber* o lo que es lo mismo, inicia los *DataReader* y los *Listener* para que puedan comenzar a recibir los mensajes del sistema de comunicaciones. Después de iniciado el sistema de recepción de mensajes se inicia, de forma análoga, el sistema de envío de mensajes iniciándose los *Publisher* y a continuación los *DataWriter*.

Finalmente, una vez que se han iniciado todos los elementos de comunicaciones, se deben iniciar los elementos de control. La iniciación de los mismos corre a cargo del *Frame*, dado que es único y conocedor del estado del inicio de las comunicaciones y por tanto de los elementos de control.

La secuencia de eventos de detención del sistema es similar a la secuencia de eventos de iniciación, es decir, el *Frame* lanza la señal de detención, pero en este caso se realiza por orden inverso, se detienen primero las entradas de mensajes de los *Subscriber* y seguidamente los *DataReader* y los *Listener* lo que bloquea el acceso de mensajes a los

sensores lógicos de control sin que éstos dejen de procesar los mensajes que ya habían sido recibidos. Por medio de la secuencia descrita anteriormente se logra una detención controlada de todos los es del sistema. De esta forma se permite avisar a las fuentes de mensajes de que el sistema de control está siendo detenido, sin que se pierdan los mensajes que las fuentes consideraban ya procesados.

Una vez vaciadas las colas de los *DataReader* o de los *Listener* los elementos de control son detenidos, de esta forma dejan de enviar mensajes a los *DataWriter*, que son detenidos una vez vacíen sus colas. Finalmente se detienen los *Publisher* y de esta forma se finaliza el proceso de envío de mensajes de control.

3.3.6 Gestión de mensajes

3.3.6.1 Envío y recepción de mensajes

En la Figura 60 se muestra la secuencia de acciones en las operaciones de envío y recepción de mensajes. Con el sistema de comunicaciones iniciado se envían y reciben mensajes a través de todos los elementos del sistema. Cuando un sensor lógico de control desea enviar un mensaje a través de un canal de comunicaciones debe emplear un *DataWriter* que, por medio del correspondiente dato lógico, esté conectado al correspondiente *Publisher*. El *Publisher* adapta y envía el mensaje a través del canal de comunicaciones a los *Subscriber* con los que esté conectado, dado que comparten el mismo dato lógico.

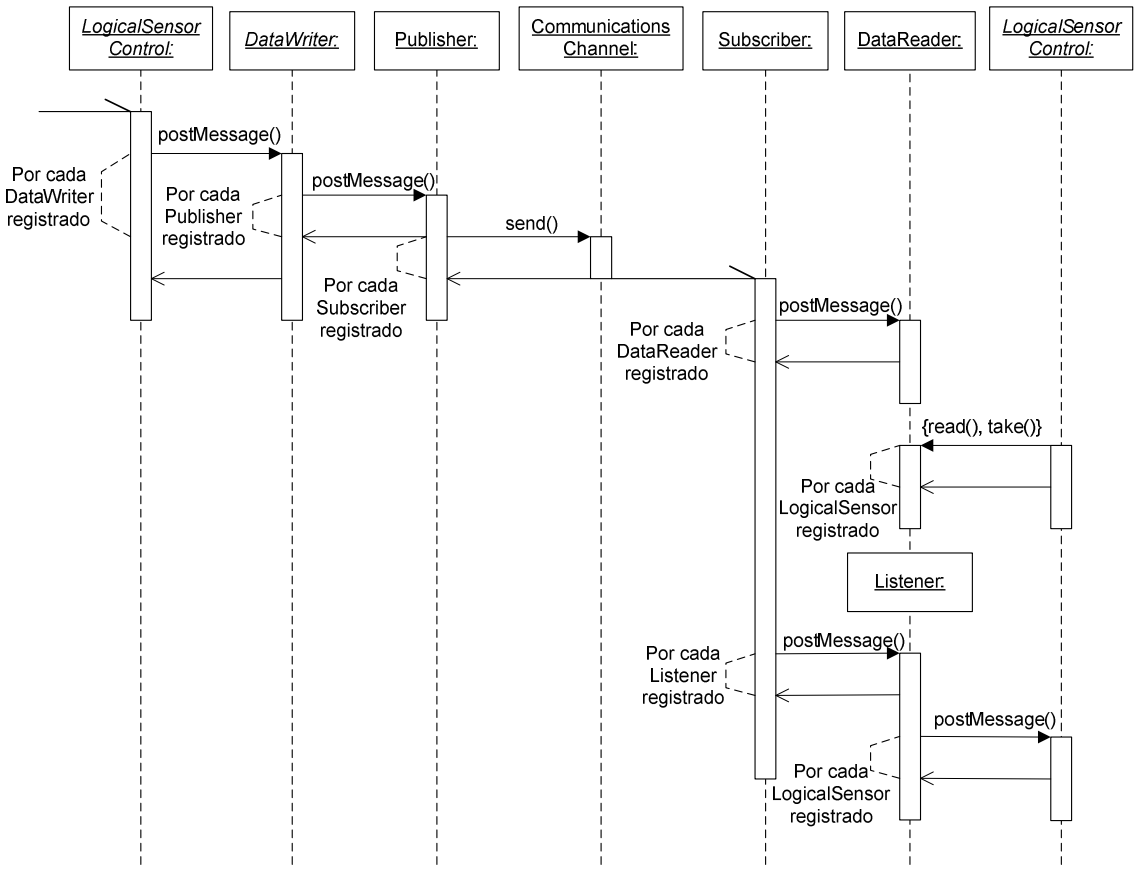


Figura 60. Diagrama de secuencia del envío y recepción de mensajes entre elementos.

Cuando el *Subscriber* recibe un mensaje del canal de comunicaciones, lo encola en los *DataReader* o los *Listener* que tenga asociados ya que comparten el mismo dato lógico.

El tratamiento que hacen del mensaje un *DataReader* y un *Listener* es diferente. El *DataReader* lo guarda en su cola de mensajes a la espera de que los sensores lógicos de control lo soliciten por medio de las operaciones *Read* o *Take*, cuando el sensor lógico de control lo considere oportuno. Por tanto el *DataReader* no produce ningún evento en el sensor lógico de control y es el sensor lógico de control, cuando esté programado, el responsable de acceder al mensaje por medio de la consulta al *DataReader*. La operación *Read* lee el primer mensaje encolado sin extraerlo de la cola de mensajes del *DataReader*, la función *Take* extrae el mensaje de la cola de mensajes del *DataReader*, dejando disponible el siguiente mensaje almacenado en la cola de mensajes del *DataReader*. En el caso de tratarse de un *Listener*, la llegada de un mensaje por parte del *Subscriber* al que esté conectado, produce un envío inmediato del mensaje a los sensores lógicos de control conectados al *Listener*. A todos los efectos, el encolado de un mensaje en el sensor lógico de control funciona de forma similar a una función de *callback* en el sensor lógico de control. Consecuentemente el sensor lógico de control debe tener programadas las acciones a realizar cuando se recibe un mensaje por parte de un *Listener* o cuando se lee el mensaje de un *DataReader*.

3.3.6.2 Encolado y procesado de mensajes

Todos los elementos de la arquitectura FSACtrl, tienen capacidad para enviarse mensajes entre ellos. La llegada de un mensaje a un elemento proveniente de otro elemento implica la realización de diversas acciones (Figura 61). Cuando un mensaje llega a un elemento, antes de ser encolado en la lista, se revisa la lista de políticas de QoS para actualizar, por medio del método *update*, los valores de los parámetros y comprobar que el mensaje cumple las condiciones que determinan las diversas configuraciones de cada política. Por ejemplo, la llegada del mensaje es el momento en el que se comprueba si el mensaje cumple los tiempos de las políticas de QoS *deadline* o *temporalfilter*.

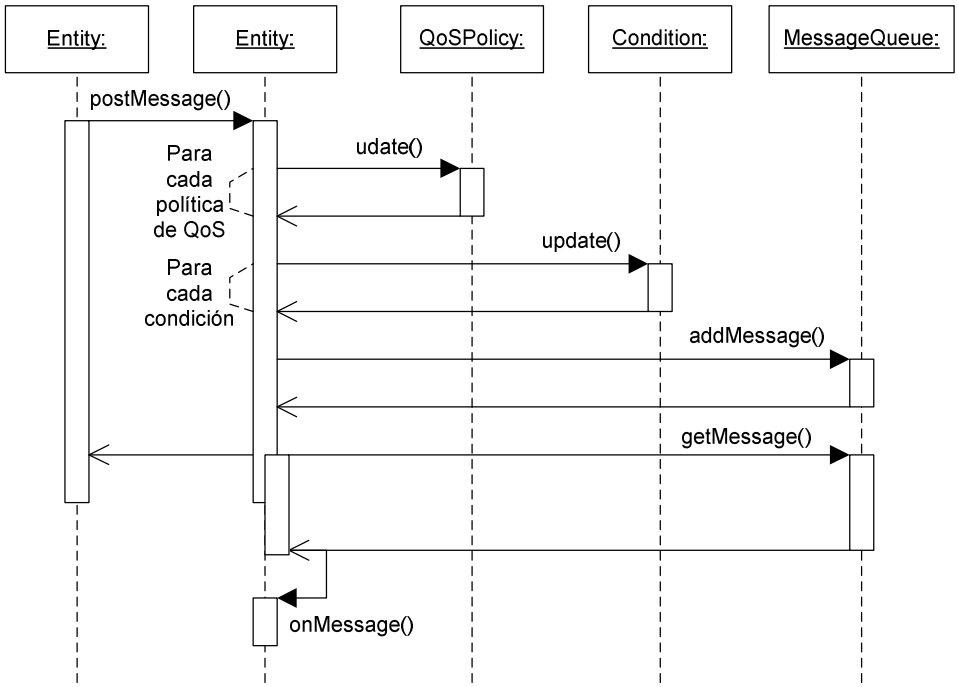


Figura 61. Diagrama de secuencia del encolado y procesado de mensajes dentro de un elemento.

Después de la actualización de los parámetros de las políticas de QoS, se pasa a comprobar, por medio del método *update*, si la llegada del mensaje debe desencadenar un evento asociado al filtrado de mensajes o a una alarma de QoS. La comprobación se realiza en todas las condiciones que tenga asociadas el elemento.

Una vez se han actualizado las políticas de QoS y las acciones asociadas a las condiciones se han procesado, el mensaje pasa a ser encolado en el elemento a la espera de ser procesado por el hilo de control. Las acciones asociadas a las condiciones se realizan de forma paralela al encolado y procesado del mensaje.

El procesado de mensajes los realiza el hilo de control, avisado por la cola de mensajes por medio de un mensaje interno del elemento. La función *onMessage* es la que recibe el mensaje desencolado de la cola de mensajes por el evento interno.

3.3.6.3 Detección y tratamiento de condiciones

3.3.6.3.1 Detección de eventos y condiciones

La detección de los eventos depende del tipo de evento de que se trate. Cuando se trata de un evento relacionado con la QoS la detección se realiza a la llegada del mensaje al elemento, ya que es en ese momento cuando se actualizan los parámetros (temporizadores y contadores). El elemento llama al método *getState* de cada una de las condiciones asociadas, que a su vez revisa las políticas de calidad de servicio para comprobar, tanto si se debe generar el evento.

Si se trata de un evento asociado a la QoC, la evaluación de la condición se realiza cuando el sensor lógico finaliza el proceso del mensaje, que es cuando la calidad de control comprueba el resultado según la referencia proporcionada.

Cuando se realiza una operación con elementos, después de realizarse la operación se comprueba si existen condiciones asociadas a las operaciones, en tal caso, dependiendo del resultado de la operación se varía el estado del evento correspondiente.

En el caso de los mensajes, el filtrado se realiza una vez se ha calculado la correspondiente QoS, ya que el hecho de que un mensaje no cumpla un criterio de filtrado no excluye el hecho de que realmente haya llegado, aunque no se propague. Si se desea calcular la QoS de los mensajes que cumplan unas ciertas características se debe realizar con un sensor lógico posterior.

3.3.6.3.2 Acciones

Cuando se da una condición a causa del conjunto de eventos correspondientes, se debe avisar al método *onCondition* del elemento que tenga asociada la *Condition*. El elemento tiene la posibilidad de implementar en dicho método la gestión de la condición. En el método *onCondition* es donde se deben realizar las operaciones lógicas entre eventos que hayan cambiado de estado, mediante la consulta de los estados internos de cada uno de los eventos (Figura 62).

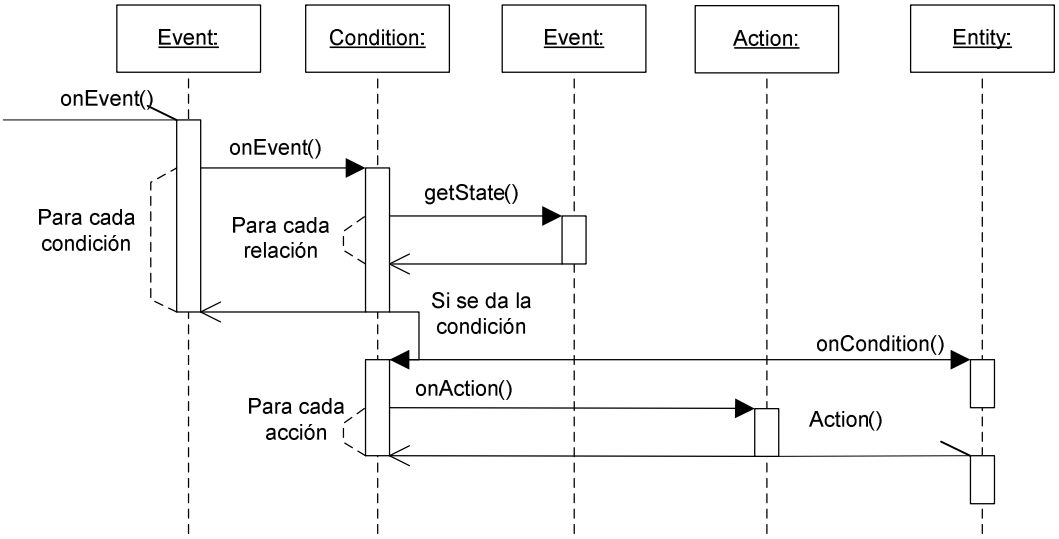


Figura 62. Diagrama de secuencia de la gestión de condiciones con eventos y acciones.

Las condiciones pueden tener asociadas acciones. Las acciones son llamadas a funciones de los elementos que se permiten hacer desde otros componentes. En FSACtrl las acciones que se pueden realizar con un componente son las llamadas a los métodos del elemento. De ellos, los genéricos a todos los sensores lógicos, tanto de comunicaciones como de control son:

- Inicio o detención de un elemento, lo que implica que éste deja de procesar mensajes.
- Abrir o cerrar un elemento, lo que implica que el elemento comienza o deja de recibir mensajes
- Añadir, cambiar o quitar políticas de QoS de los elementos y parámetros de QoC de los sensores lógicos.
- Cambio de configuración de las políticas de QoS de los elementos o parámetros de QoC, según proceda.
- Envío de un mensaje a un elemento del sistema.

Las acciones deben mantener una jerarquía, ya que no todo elemento debe poder manejar a cualquier otro elemento del sistema. En el caso de FSACtrl la jerarquía la impone por defecto el ámbito del componente de control, es decir los elementos solo pueden llamar a acciones de elementos que se encuentren en el mismo componente de control o en los componentes de control internos a ellos. Las acciones que pueden realizarse sobre los sensores lógicos pertenecientes a un componente de control son:

- Crear o eliminar sensores lógicos.
- Duplicar o mover sensores lógicos.

Estas operaciones, aunque costosas, son especialmente interesantes para la implementación de sistemas de agentes de control inteligente distribuidos, ya que serán la base para las operaciones genéricas de clonado y movimiento de agentes completos que más adelante se detallarán.

3.4 Parámetros de calidad de servicio

3.4.1 Ubicación de la calidad de servicio en el sistema

Uno de los principales objetivos de la tesis consiste en la gestión dinámica de la calidad de servicio para medir, evaluar y optimizar un sistema de control distribuido. La calidad de servicio es un concepto en el que habitualmente la formulación de los parámetros es específica al contexto en el que se aplican. En el caso de la arquitectura FSACtrl, se propone emplear los parámetros cuantitativos de las colas de mensajes y los hilos de control de los elementos para obtener los parámetros cualitativos.

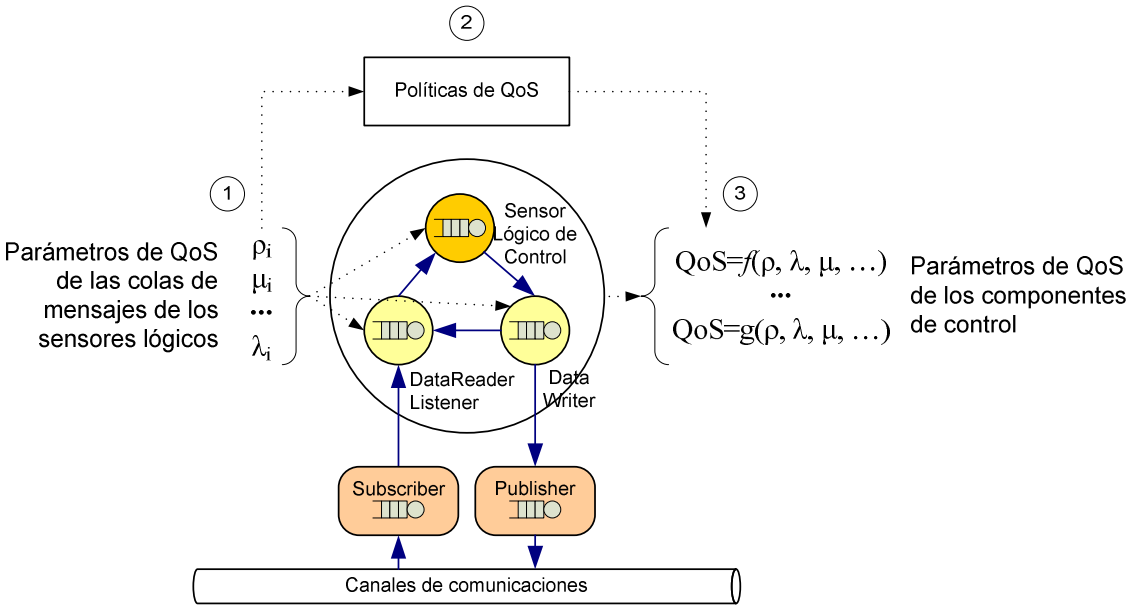


Figura 63. Obtención de parámetros globales a partir de los parámetros locales.

En la Figura 63 se expone cómo los parámetros de calidad de servicio se pueden extender desde los sensores lógicos hacia los componentes de control. De la misma forma que un componente de control oculta los detalles de los sensores lógicos que lo componen, los parámetros de calidad de servicio del componente de control son función de los parámetros de los sensores lógicos y permiten parametrizar el componente de control.

Existe una relación entre los parámetros de las colas de mensajes y los parámetros de calidad de servicio de los sistemas más complejos. Como todos los elementos de la arquitectura FSACtrl disponen de una cola de mensajes (Figura 63), pueden proporcionar los parámetros individuales cuantitativos empleados para el cálculo de los parámetros comunes cualitativos, pero con una base empírica bien definida. Al disponer de las colas de mensajes en los elementos de comunicaciones y en los sensores lógicos de control, se extrapolan al control los parámetros de calidad de servicio, tradicionalmente perteneciente al ámbito de las comunicaciones.

El hecho de disponer de colas de mensajes y de hilos de control, lejos de ser un obstáculo, supone una de las ventajas principales del sistema. Las colas y los hilos de control, además de proveer de parámetros de funcionamiento del elemento, también permiten ser configurados. Es posible ralentizar o priorizar mensajes para que se cumplan ciertos plazos temporales, o mantener mensajes en las colas para cumplir algunos requisitos del flujo de mensajes.

De forma análoga, es posible priorizar el procesamiento de los mensajes por parte del procesador en función de unas necesidades de calidad de servicio determinadas. Los aspectos anteriores pueden ser configurados por medio de las políticas de QoS, lo que hace de estas últimas un lenguaje común a todos los elementos, independiente de su composición o de su funcionalidad. De esta manera se forma un ciclo de gestión del elemento basado en la QoS, cuyas fases se muestran numeradas en la Figura 63 y que se compone de los siguientes pasos:

1. Inicialmente, la cola de mensajes del sensor lógico de control proporciona los parámetros cuantitativos (como el promedio de mensajes esperando en cola, la tasa de llegadas o la tasa de servicio), a partir de los parámetros suministrados por la cola de mensajes y el hilo de control, se obtienen los parámetros de calidad de servicio (como la eficiencia o la capacidad).
2. Los parámetros de calidad de servicio se combinan y se monitorizan, por medio de los eventos y las condiciones, para comprobar el cumplimiento de las políticas de calidad de servicio que se hayan configurado.
3. En función del cumplimiento de las restricciones de las políticas de calidad de servicio, por medio de las acciones asociadas a las condiciones se actúa sobre el sistema.

En la arquitectura FSACtrl se distinguen entre los parámetros de los sensores lógicos y adaptadores, y los parámetros de los componentes de control conectados entre sí. Los parámetros de los sensores lógicos y adaptadores son similares a los parámetros de las colas de mensajes, mientras que los parámetros de los componentes de control se elaboran a partir de las combinaciones de los parámetros de los sensores lógicos.

Los valores de los parámetros pueden estar referidos a un instante de tiempo concreto o a un intervalo de tiempo. Los valores instantáneos describen la situación actual del sistema mientras que los valores obtenidos durante un intervalo de tiempo proporcionan una visión continua del sistema. Los valores máximos y mínimos de un parámetro a lo largo de un intervalo de tiempo proporcionan la acotación del mismo en el sistema permitiendo calcular márgenes de optimización, o márgenes en los que el elemento puede negociar con otros cambios en las prestaciones de los servicios ofrecidos. La variación de los parámetros a lo largo del tiempo permite observar la tendencia a la optimización o el empeoramiento del sistema.

3.4.2 Parámetros de QoS de los sensores lógicos y adaptadores

3.4.2.1 Obtención de los parámetros

Los parámetros internos de los elementos se basan en los parámetros propuestos por la teoría de colas [Bolch et al., 2006]. En FSACtrl, los elementos contienen una sola cola de mensajes y un solo hilo de control. Esto implica que los parámetros iniciales serán los que se obtienen directamente de los elementos. En la Figura 64, se muestra un esquema de dichos parámetros y su localización.

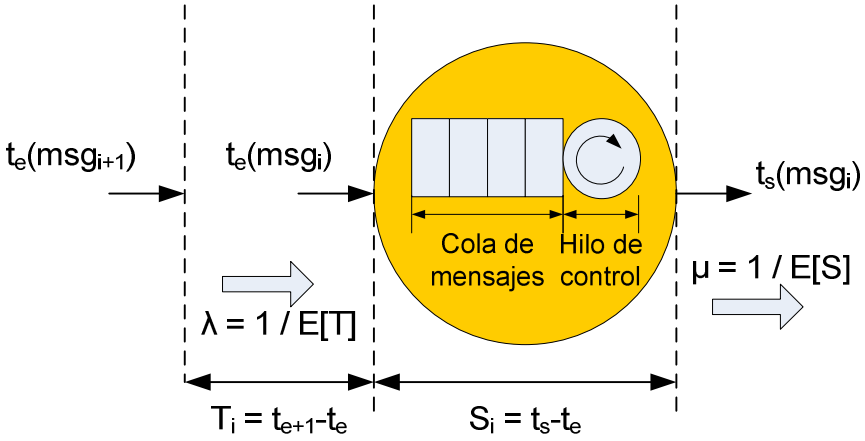


Figura 64. Parámetros de los elementos FSACtrl.

Los parámetros de los sensores lógicos y adaptadores de la arquitectura FSACtrl se obtienen a partir del flujo de mensajes de la cola de mensajes asociada y de la obtención de tiempos del hilo de control.

3.4.2.2 Formulación de los parámetros

3.4.2.2.1 Demanda del servicio

La demanda de servicio es un parámetro estimado que no depende del elemento, sino de las solicitudes que otros elementos hacen de sus servicios. Para un mensaje *i*, el intervalo de tiempo de demanda de servicio se calcula en función del mensaje anterior (fórmula 1). Siendo t_e el instante temporal en que entra un mensaje, msg_i un mensaje genérico y msg_{i+1} el siguiente mensaje en llegar.

$$T_i = t_e(msg_{i+1}) - t_e(msg_i) \quad (1)$$

Las llegada de un mensaje implica la solicitud de ejecución del hilo de control del elemento por lo que la demanda del uso de un elemento se calculará a partir del promedio del intervalo de tiempos de llegada de los mensajes que solicitan el servicio (fórmula 2). Siendo N el total de los mensajes monitorizados para la obtención del parámetro.

$$E[T] = \left[\sum_{i=1}^{N}(T_i)\right] \Big/ N \quad (2)$$

A partir de los promedios de tiempos de llegada de los mensajes de solicitud de servicio, la demanda se calcula como la tasa media de llegadas de mensajes, o lo que es lo mismo, la inversa del promedio de las demandas de servicio (fórmula 3). Las acotaciones máximas y mínimas se obtienen a partir de los valores máximos y mínimos obtenidos en el intervalo de medición y son empleados para la obtención de las cotas optimista y pesimista, lo que implica utilizar el mejor, o peor, de los valores obtenidos.

$$\lambda = 1/E[T] \quad (3)$$

En lo que a demanda se refiere, el caso más estricto es considerar el menor intervalo de tiempo en lo que a llegada de mensajes se refiere, por ello la demanda máxima de servicio se calcula como se muestra en la fórmula 4.

$$\lambda_{\max} = 1/\left[\min[T_i]_{i=1}^{N}\right] \quad (4)$$

La demanda de servicio se mide en mensajes por segundo. Al ser un parámetro externo no tiene sentido hablar de optimización del mismo si no es para optimizar los elementos de los que provienen los mensajes, es decir, no es posible actuar sobre la demanda de un elemento sin actuar sobre los otros elementos desde los que provienen los mensajes.

3.4.2.2.2 Tiempo de servicio

El tiempo de servicio, también conocido en el ámbito del control como tiempo de respuesta, es el tiempo que el elemento tarda en responder a una petición de servicio (fórmula 5) donde el servicio es calculado como el tiempo transcurrido entre que un mensaje *i* llega en el instante t_e al elemento, hasta que se termina el procesamiento del mismo en el instante t_s.

$$S_i = t_s(msg_i) - t_e(msg_i) \tag{5}$$

A efectos prácticos, el parámetro que se suministra es la cota superior de los tiempos anteriores (fórmula 6) o tiempo de servicio máximo, ya que es el más empleado para los cálculos restrictivos de sistemas de tiempo real estricto.

$$S_{\max} = \max[S_i]_{i=1}^{N} \tag{6}$$

Dado que existen variaciones en el tiempo de servicio, el parámetro que se suministra como tiempo de servicio se calcula como el promedio de los tiempos de servicio y se representa como E[S] (fórmula 7).

$$E[S] = \left[\sum_{i=1}^{N} S_i\right] \Big/ N \tag{7}$$

El tiempo de servicio se mide en las unidades temporales que se haya determinado. Por norma general se considera una optimización del tiempo de servicio la reducción del mismo.

3.4.2.2.3 Tasa de servicio

A partir del tiempo medio de respuesta se obtiene la tasa de servicio (fórmula 8) que representa el número de mensajes capaz de procesar por unidad de tiempo empleada.

$$\mu = 1 / E[S] \tag{8}$$

Tal como ocurre con el tiempo de servicio, es necesario trabajar con la acotación más estricta, por lo que se emplea la tasa de servicio mínima, obtenida a partir del tiempo de servicio máximo (fórmula 9).

$$\mu_{\min} = 1 / S_{\max} \tag{9}$$

El tiempo de servicio y la tasa de servicio ofrecen una aproximación inicial del comportamiento del elemento. La tasa de servicio se mide en mensajes por unidad temporal lo que equivale a solicitudes de servicio servidas por unidad temporal. Se considera una optimización aumentar la tasa, lo que implica disminuir el tiempo máximo de servicio.

3.4.2.2.4 Carga

La carga, o tráfico, representa la cantidad de demanda sobre el uso del elemento en función de la capacidad de servicio del mismo. Se obtiene por medio de la fórmula 10.

$$\rho = \lambda / \mu \tag{10}$$

Al ser una relación entre dos unidades idénticas, la carga es adimensional. El valor de la carga está entre cero e infinito. La carga cero indica que un elemento no realiza ninguna labor. Una carga con valor uno indica que un elemento está realizando sus funciones en todo momento y las solicitudes de servicios son exactamente las que el elemento puede asumir. Un valor de la carga mayor que uno indica un elemento sobrecargado y por tanto un elemento desbordado sobre el que se deberán tomar medidas dado que puede comenzar a perder mensajes. Para determinar la cota superior de la carga, o carga máxima, se muestra en la fórmula 11.

$$\rho_{\max} = \max[\rho_i]_{i=1}^{N} \tag{11}$$

La carga máxima teórica se obtiene a partir de los peores casos, es decir, la máxima tasa de solicitudes de servicio con la menor capacidad de procesos de los servicios solicitados (fórmula 12).

$$\rho_{\max\_t} = \lambda_{\max} / \mu_{\min} \tag{12}$$

El valor de la carga máxima es especialmente útil ya que considera que confluyen al mismo tiempo las peores circunstancias de solicitudes del servicio y de procesamiento del mismo, por lo que implica la cota absoluta de carga admisible.

3.4.2.2.5 Utilización

La utilización es un parámetro similar a la carga con la salvedad de que la carga expresa la cantidad de trabajo que está realizando el elemento, mientras que la utilización expresa el mismo concepto como porcentaje sobre la carga máxima. Un elemento puede estar sobrecargado, pero no "sobreutilizado", por ello la utilización se formula ponderando la carga con la carga máxima (fórmula 13).

$$U = \rho / \rho_{máx\_t} \tag{13}$$

La carga máxima a la que se refiere la utilización puede tener diversas procedencias: la máxima de las cargas detectadas a lo largo de la sesión o la carga máxima teórica. Escoger uno u otro origen dependerá de las características propias del sistema. Por ejemplo, la carga máxima teórica es una cota superior, en teoría inalcanzable, por lo que la utilización obtenida pasa a ser también una cota superior. Sin embargo la carga máxima de la sesión es un parámetro cuyo valor puede cambiar, por lo que la utilización obtenida es válida durante la sesión. La conclusión de lo anterior es que la utilización basada en la carga máxima teórica es útil como parámetro representativo del sistema, desde una visión más estática, mientras que la utilización basada en la carga máxima instantánea es apropiada para tomar decisiones en tiempo de funcionamiento.

3.4.2.2.6 Ocupación

La carga está calculada como una proporción entre la tasa de servicio y la demanda del servicio. La tasa de servicio puede limitarse por motivos de configuración, por ejemplo para ralentizar mensajes o para asegurar unos servicios mínimos. A la carga obtenida por medio de la limitación de tasa de servicio (μ_{lim}) de un elemento se le conoce como ocupación del elemento (fórmula 14)

$$\sigma = \lambda / \mu_{\lim} \tag{14}$$

La carga se emplea como parámetro para determinar el uso real de un sensor lógico o adaptador, mientras que la ocupación determina el uso que se hace de un elemento en función de una configuración concreta del servicio.

3.4.2.2.7 Disponibilidad

La disponibilidad de un elemento es el parámetro complementario a la carga y describe la capacidad que tiene un elemento de poder aceptar más carga de trabajo fórmula 15)

$$\delta = (1 - \rho) \tag{15}$$

Una ocupación baja implica una disponibilidad alta debido a la baja carga que soporta el elemento. A medida que la carga se acerca a uno, la disponibilidad baja hasta cero. La disponibilidad se puede expresar como porcentaje, describiendo entonces la probabilidad de poder disponer del servicio. Los valores óptimos de la disponibilidad dependen del entorno de funcionamiento del elemento. Un valor cercano a cero puede indicar un sistema muy optimizado al poder disponer siempre del elemento, pero también puede indicar un elemento infrautilizado.

3.4.2.2.8 Capacidad

La capacidad describe qué margen real de ocupación puede ofrecer el elemento (fórmula 16).

$$\kappa = (1 - \sigma) \tag{16}$$

Las diferencias entre la disponibilidad y la capacidad son conceptuales. La disponibilidad puede ser negativa, lo que deberá ser tenido en cuenta por parte del cliente e indicará un tiempo de espera elevado y un retraso en el tiempo de servicio mayor que el tiempo de servicio indicado por el correspondiente parámetro. La capacidad sin embargo, no puede ser negativa.

3.4.2.2.9 Productividad y rendimiento

La productividad es el parámetro más empleado para reflejar la eficiencia, el rendimiento, la utilidad, y en definitiva lo óptimo que resulta un elemento en prácticamente todos los ámbitos en los que existe una evaluación de la calidad. En el caso de FSACtrl la productividad de un elemento de la arquitectura es la tasa de servicio (fórmula 8).

La productividad se asocia a la relación entre los productos y los recursos empleados, sin embargo en estos casos también se habla de rendimiento, por ello en FSACtrl se diferencian ambos parámetros. En el caso de la arquitectura FSACtrl, el rendimiento se define en términos de servicios proporcionados en función de la capacidad máxima de los servicios que puede proporcionar, por lo tanto es un parámetro adimensional que expresa el porcentaje de eficiencia que el elemento está proporcionando (fórmula 17).

$$\eta = \mu / \mu_{\max} \tag{17}$$

3.4.3 Parámetros de QoS de los componentes de control

3.4.3.1 Obtención de los parámetros

3.4.3.1.1 Elementos que intervienen en los cálculos

Los componentes de control, al disponer entre sus elementos sensores lógicos de comunicaciones, tienen una cantidad variable de conexiones entrantes y salientes. Además, los sensores lógicos de control también tienen diversas conexiones entrantes y salientes. Consecuentemente, las conexiones entre los diversos sensores lógicos que componen un componente de control forman un grafo de elementos que poseerán una cantidad de rutas que pueden seguir los mensajes procesados o generados por los sensores lógicos (Figura 65).

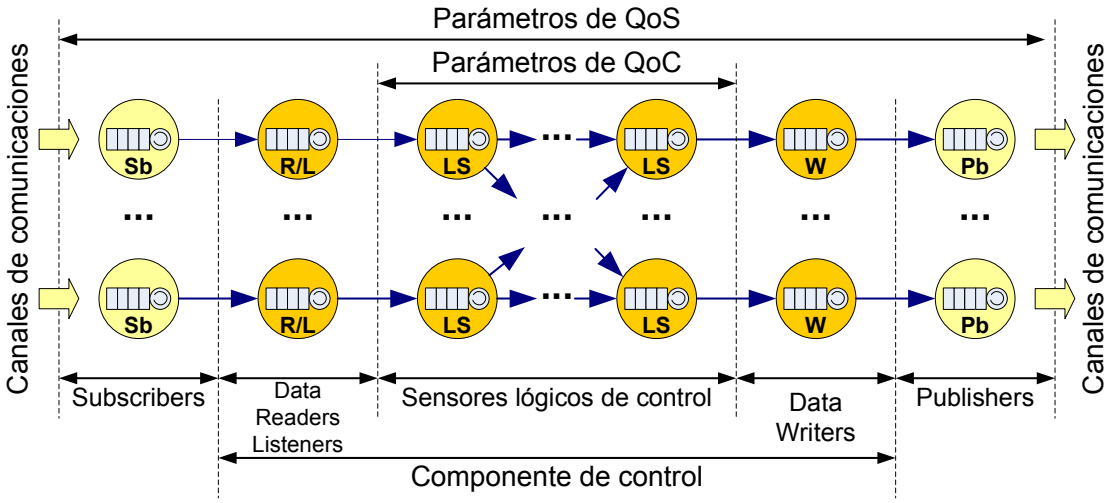


Figura 65. Rutas de mensajes de los componentes de control.

Cuando se configura una política de QoS en un sensor lógico, tanto de comunicaciones como de control, los valores de la política deben cumplirse tanto por los sensores lógicos de los que se recibe mensajes como a los que se envían. Por ello la configuración de los parámetros de QoS está condicionada por las rutas de los mensajes y los canales de comunicaciones.

Los sensores lógicos de control son los que implementan los algoritmos de control, por lo que también son los responsables de proporcionar los parámetros de QoC [Dorf and Bishop, 2008] y los empleados en los sistemas de control basados en eventos [Sánchez et al., 2009].

Los parámetros calculados, ya sea a partir de los sensores lógicos de comunicaciones como de los sensores lógicos de control, son la base de los parámetros de QoS del componente de control y de los parámetros de QoC [Soucek and Sauter, 2004].

3.4.3.1.2 Tipos de parámetros de los componentes de control

En FSACtrl, para calcular los parámetros de QoS de un componente de control, se emplean los parámetros de QoS de los sensores lógicos que lo componen. El nivel de detalle empleado para la obtención de los parámetros de calidad de servicio de los componentes de control varía en función del número y tipo de sensores lógicos contenidos en el componente de control. Dependiendo del ámbito, los parámetros se organizan en los tres niveles de detalle mostrados en la Figura 66.

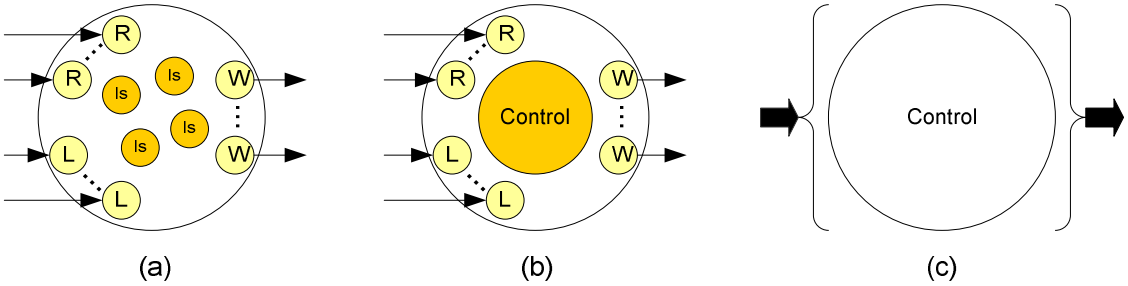


Figura 66. Nivel de detalle de un componente de control.

El primer nivel de detalle, es el que emplea todos los sensores lógicos del componente de control (Figura 66, a), a los parámetros que emplean este nivel de detalle se les ha llamado parámetros intrínsecos. El siguiente nivel de detalle es el que únicamente emplea los sensores lógicos de comunicaciones (Figura 66, b), en este caso se ha llamado a los parámetros, parámetros aparentes, ya que solo dan información de los elementos que actúan de interfaz del componente de control, sin tener en cuenta la composición interna del componente de control. Finalmente, se considera el componente de control globalmente (Figura 66, c), en este caso, al no diferenciar el sensor lógico de comunicaciones que recibe el mensaje, se debe considerar la secuencia de mensajes que produce una secuencia de actuaciones de control. A este último tipo de parámetros se les ha llamado parámetros representativos.

La utilidad de cada tipo de parámetro depende del contexto en el que se use. Los parámetros representativos dan una visión general del componente de control, los parámetros aparentes proporcionan la visión de eficiencia desde el punto de vista de las comunicaciones, mientras que los parámetros intrínsecos informan acerca de la eficiencia en la funcionalidad del componente de control.

3.4.3.2 Parámetros intrínsecos

Los parámetros intrínsecos son aquellos que emplean para su cálculo los parámetros de todos los sensores lógicos que componen los componentes de control, tanto los de comunicaciones como los de control. El subíndice *in*, se emplea para designar la cualidad de intrínseco del parámetro.

Es de especial interés que a través de los parámetros intrínsecos de un sensor lógico se detecta aquellos que no puedan satisfacer la configuración de un parámetro, lo que puede provocar un efecto en cadena en el componente de control. Este aspecto no puede ser detectado por los otros tipos de parámetros.

Los valores marginales de los parámetros intrínsecos indican un probable mal funcionamiento en el componente de control, o un componente de control bien optimizado. Sin embargo, los valores marginales de un sensor lógico pueden estar eclipsados por otros sensores lógicos, es decir, un sensor lógico de control sobrecargado aporta poca carga intrínseca si hay varios sensores lógicos con una carga baja. Este aspecto hace que los parámetros intrínsecos sean útiles para evaluar de una forma muy general el componente de control, mientras que los valores máximos y mínimos de los parámetros intrínsecos permiten localizar sensores lógicos con problemas. A continuación, en la Tabla 4, se presentan los parámetros intrínsecos formulados.

Tabla 4. Fórmulas de los parámetros intrínsecos.

Parámetro	Fórmula	
Promedio del intervalo de tiempo de demanda de servicio.	$E[T]_{in} = \left[\sum_{i=1}^{N}(E[T]_i)\right] \Big/ N$	(18)
Demanda de servicio intrínseca, promedio.	$\lambda_{in} = 1/E[T]_{in}$	(19)
Demanda de servicio intrínseca, máxima.	$\lambda_{in\_\max} = 1/(\max[T_{\max(i)}]_{i=1}^{S})$	(20)
Tiempo de servicio intrínseco.	$E[S]_{in} = \sum_{i=1}^{N} E[S]_i \Big/ N$	(21)
Tiempo de servicio intrínseco máximo.	$S_{in\_\max} = \max[(S_{\max})_i]_{1=1}^{N}$	(22)
Tasa de servicio intrínseca.	$\mu_{in} = 1/E[S]_{in}$	(23)
Tasa de servicio intrínseca máxima.	$\mu_{in\_\max} = 1/S_{in\_\max}$	(24)
Carga intrínseca.	$\rho_{in} = \sum_{i=1}^{N} \rho_i \Big/ N$	(25)
Carga intrínseca máxima	$\rho_{in\_\max} = \max[\rho_i]_{i=1}^{N}$	(26)
Utilización intrínseca	$U_{in} = \sum_{i=1}^{N} U_i \Big/ N$	(27)
Ocupación intrínseca	$\sigma_{in} = \sum_{i=1}^{N} \sigma_i \Big/ N$	(28)
Disponibilidad intrínseca	$\delta_{in} = \sum_{i=1}^{N} \delta_i \Big/ N$	(29)
Capacidad intrínseca	$\kappa_{in} = \sum_{i=1}^{N} \kappa_i \Big/ N$	(30)
Productividad intrínseca	$P_{in} = \mu_{in}$	(31)
Rendimiento intrínseco	$\eta_{in} = \mu_{in} / \mu_{in\_\max}$	(32)

La demanda de servicio de un componente de control se define como la demanda promedio de los elementos que lo componen. La fórmula 18 muestra la fórmula que calcula el intervalo de tiempo promedio entre las solicitudes de cada sensor lógico que compone el componente de control donde N es el total de sensores lógicos del componente de control. A partir de la fórmula anterior se obtiene la demanda de servicio, en solicitudes de servicio por unidad de tiempo. En la Fórmula 19 se muestra la fórmula de la demanda de servicio intrínseca. La cota máxima de la demanda de servicio por medio de la que se conoce la máxima exigencia que se le pide al componente de control se obtiene a partir del máximo intervalo de tiempo de demanda y constituye la demanda máxima de servicio (Fórmula 20).

Al igual que en los sensores lógicos, el tiempo de servicio es el tiempo que tarda un mensaje desde que llega al componente de control hasta que se produce la correspondiente acción de control. El tiempo de servicio de un mensaje concreto servido a un componente de control depende de la ruta de sensores lógicos que procesan el mensaje de solicitud de servicio. La configuración de la ruta está definida por las conexiones entre los sensores lógicos del componente de control.

La fórmula 21, se obtiene el promedio del tiempo de servicio de los mensajes en los sensores lógicos, sin embargo para los cálculos más restrictivos es conveniente emplear los peores casos. En ese caso se tiene que calcular como tiempo de servicio máximo el del sensor lógico con más retardo (fórmula 22).

La tasa de servicio describe la capacidad de suministrar el servicio correspondiente a quienes lo solicitan. En el caso de los componentes de control, el concepto de tasa de servicio debe referirse consecuentemente al concepto de qué se entiende por servicio proporcionado por un componente de control. Las tasas de servicio intrínsecas se obtienen directamente de los tiempos de servicio intrínsecos, como la inversa de los mismos, tanto la tasa promediada (fórmula 23) como la acotación máxima (fórmula 24).

La carga intrínseca se obtiene a partir de las cargas de todos los elementos. En la fórmula 25 se puede ver cómo se calcula como el promedio de las cargas de los sensores lógicos. La carga intrínseca máxima (fórmula 26) se define como la máxima de las cargas de entre todos los elementos. Tanto la carga intrínseca como la carga intrínseca máxima dan una idea de la situación en la que se encuentran los sensores lógicos que forman el componente de control. Sin embargo es la carga intrínseca máxima la que tiene más utilidad ya que avisa de la posibilidad de que se sobrecargue uno de los sensores lógicos que forman el componente de control, con la posible reacción en cadena sobre el resto de los sensores lógicos.

La utilización intrínseca se obtiene como el promedio de las utilizaciones de los sensores lógicos (fórmula 27). Por medio de éste parámetro se determina el promedio de utilización de entre todos los sensores lógicos del componente de control. La utilización intrínseca da una idea de qué uso se está haciendo de los sensores lógicos. La ocupación intrínseca, mostrada en la fórmula 28, especifica el promedio de ocupación del componente de control, en función de las ocupaciones de cada elemento. La ocupación intrínseca permite hacer una estimación del estado interno y así estimar la capacidad, como se verá más adelante, que puede tener el componente de control visto como conjunto.

La disponibilidad intrínseca se calcula como el promedio de las disponibilidades de los sensores lógicos. La fórmula para su cálculo se muestra en la fórmula 29. La disponibilidad intrínseca deja entrever qué margen de fiabilidad interno se puede aplicar al componente de control, ya que un valor muy elevado, indicará que prácticamente todos los sensores lógicos están funcionando correctamente, independientemente de si están siendo más o menos utilizados. Sin embargo, el hecho de estar calculado a partir de todos los sensores lógicos puede hacer que diversos sensores lógicos con un índice de disponibilidad alto, eclipsen a un elemento con un índice de disponibilidad bajo.

La capacidad intrínseca se calcula tal como muestra la fórmula 30. Al igual que la disponibilidad, la capacidad permite percibir el margen que tiene el componente de control de poder admitir más solicitudes de servicio. Un elemento con poca capacidad quedará eclipsado por varios elementos con alta capacidad, por lo es más aconsejable emplear los valores máximos o mínimos para localizar sensores lógicos con mucha capacidad que pueden atender más servicios, o aquellos con poca capacidad que convendrá descargarlo. La productividad intrínseca permite estimar el rendimiento del componente de control de forma global (fórmula 31). El rendimiento se muestra en la fórmula 32. En ocasiones se diferencia productividad y rendimiento en función del ámbito de utilización del mismo: la productividad le interesa al cliente ya que especifica lo que el componente de control es capaz de servir, mientras que el rendimiento interesa

más al diseñador del componente de control ya que le da una estimación de cuánto le está costando producir los servicios.

3.4.3.3 Parámetros aparentes

Los parámetros aparentes son aquellos que se obtienen utilizando los sensores lógicos de comunicaciones del componente de control. En este nivel de detalle, los sensores lógicos de comunicaciones proporcionan los parámetros de calidad de servicio, ya que todos los mensajes pasan por esos elementos. El subíndice *ap*, se emplea para designar la cualidad de aparente del parámetro. En los componentes de control, la interfaz por medio de la que se reciben los mensajes de demanda de servicio son los sensores lógicos de comunicaciones de entrada: *DataReader* y *Listener*. El estado de estos sensores lógicos influye en el comportamiento del componente de control, ya que una saturación de los mismos ralentizará y desaprovechará la potencia de los sensores lógicos de control internos. Asimismo una situación de baja eficiencia en los sensores lógicos de comunicaciones de salida (*DataWriter*) se extenderá a los clientes a los que el componente de control esté suministrando servicio A continuación, en la Tabla 5, se muestran los parámetros de calidad de servicio aparentes de un componente de control, basados en los sensores lógicos de comunicaciones y las rutas de mensajes.

Tabla 5. Fórmulas de los parámetros aparentes.

Parámetro	Fórmula	
Tiempo de demanda de servicio aparente	$E[T]_{ap} = \sum_{i=1}^{DR+L} E[S]_i \Big/ (DR+L)$	(33)
Demanda de servicio aparente	$\lambda_{ap} = 1/E[T]_{ap}$	(34)
Demanda de servicio aparente máxima	$\lambda_{ap\_\max} = 1/\max[T_{\max(i)}]_{i=1}^{DR+L}$	(35)
Tiempo de servicio aparente	$E[S]_{ap} = \sum_{j=1}^{R}\left[\sum_{i=1}^{S(R)} E[S]_i\right]_j \Big/ R$	(36)
Tiempo de servicio aparente mínimo	$S_{ap\_\min} = \min[(S_{\max})_i]_{i=1}^{R}$	(37)
Tiempo de servicio aparente máximo	$S_{ap\_\max} = \max[(S_{\max})_i]_{i=1}^{R}$	(38)
Tasa de servicio aparente	$\mu_{ap} = 1/E[S]_{ap}$	(39)
Tasa de servicio aparente mínima	$\mu_{ap\_\min} = 1/S_{ap\_\min}$	(40)
Tasa de servicio aparente máxima	$\mu_{ap\_\max} = 1/S_{ap\_\max}$	(41)
Carga aparente	$\rho_{ap} = \lambda_{ap}/\mu_{ap}$	(42)
Carga aparente mínima	$\rho_{ap\_\min} = \min[\rho_i]_{i=1}^{R}$	(43)
Carga aparente máxima	$\rho_{ap\_\max} = \max[\rho_i]_{i=1}^{R}$	(44)
Utilización aparente	$U_{ap} = \rho_{ap}/\rho_{ap\_máx}$	(45)
Ocupación aparente	$\sigma_{ap} = \lambda_{ap}/\mu_{ap\_\lim}$	(46)
Disponibilidad aparente	$\delta_{ap} = (1-\rho_{ap})$	(47)
Capacidad aparente	$\kappa_{ap} = (1-\sigma_{ap})$	(48)
Productividad aparente	$P_{ap} = \mu_{ap}$	(49)
Rendimiento aparente	$\eta_{ap} = \mu_{ap}/\mu_{ap\_\max}$	(50)

La demanda de servicio mide las solicitudes de servicio que se realizan al componente de control, en el caso de la demanda de servicio intrínseca, la demanda del componente de control, se mide a partir de las demandas de servicio de los *DataReader* y los *Listener*. El intervalo de tiempo de demanda se muestra en la fórmula 33, a partir de del cual se calcula la demanda de servicio aparente (fórmula 34). Uno de los parámetros aparentes más relevantes es la demanda de servicio máxima (fórmula 35) a partir de la cual se puede obtener la cota superior de la demanda que se hace del componente de control.

Los componentes de control sólo suministran mensajes a través de los sensores lógicos de comunicaciones que cierran la ruta de los mensajes (*DataWriter*), por lo que es posible emplear los tiempos en los que sirven los mensajes para obtener una estimación del tiempo que se tarda en atenderlos. Esta estimación depende en gran medida de la relación que haya entre la secuencia de entradas de solicitudes de servicio y las secuencias de acciones de control que se correspondan a esa secuencia de mensajes (fórmula 36).

Al igual que ocurre con la demanda, las cotas de los tiempos de servicio son indicadores de la eficiencia con la que se están sirviendo los servicios demandados. En el caso del tiempo de servicio, el mínimo (fórmula 37) proporciona una cota superior de la eficiencia en servir, mientras que el máximo (fórmula 38) proporciona la cota inferior.

A partir de los tiempos de servicio se obtienen las tasas que proporcionan la información en función de los servicios por unidad de tiempo que el componente de control es capaz de proporcionar. La tasa de servicio aparente se muestra en la fórmula 39 y proporciona el promedio de los servicios ofrecidos por las rutas. Las acotaciones de la tasa de servicio se muestran en la fórmula 40 y la fórmula 41 y ofrecen la visión de la efectividad mínima y máxima alcanzable por el componente de control. Además es posible determinar la diferencia entre rutas

Para el cálculo de la carga aparente del componente de control se emplean las cargas de los sensores lógicos de comunicaciones (fórmula 42). El valor mínimo y máximo de la carga aparente se muestra en la fórmula 43 y en la fórmula 44. Por medio de la carga mínima se detectan los sensores lógicos de comunicaciones que están infrautilizados, mientras que por medio de la carga máxima se detectan los sensores lógicos de comunicaciones cercanos a la situación de desbordamiento.

Tal como se muestra en la fórmula 45, la utilización aparente se obtiene a partir de las cargas aparentes prorrateando la carga promedio entre la carga máxima. A partir de la utilización aparente se obtiene una estimación de cuánto se está empleando el componente de control en función de las carga de las solicitudes de servicios y de los servicios proporcionados. La fórmula para su obtención se muestra en la fórmula 46.

La fórmula 47 nuestra la fórmula de la disponibilidad aparente. Por medio de la disponibilidad aparente se obtiene la probabilidad de ver ofrecido un servicio, tanto su demanda, dado que se tienen en cuenta los *DataReader* como los *Listener* para la disponibilidad de la demanda, como los *DataWriter* para la disponibilidad de los servicios ofrecidos por la tasa de servicio. La capacidad aparente se calcula por medio de la fórmula 48.

La productividad aparente se calcula a partir de la tasas de servicio aparente, promedio tal como se muestra en la fórmula 49. El rendimiento se calcula, de forma similar, a partir de los rendimientos de los sensores lógicos de comunicaciones (fórmula 50). Al estar calculada por medio de las tasas de servicio aparente y máxima de cada sensor

lógico de comunicaciones, el rendimiento proporciona la eficiencia del componente de control, basada en menos sensores lógicos que los equivalentes intrínsecos.

3.4.3.4 Parámetros representativos

Se consideran parámetros representativos aquellos que se obtienen considerando globalmente el componente de control. Los componentes de control proporcionan un servicio o acción de control (en adelante se empleará la nomenclatura de servicio ya que se trata de obtener los parámetros de calidad de servicio) que se solicita por medio de la llegada de un conjunto de mensajes a los sensores lógicos de comunicaciones de entrada: *DataReader* y *Listener*. El servicio es proporcionado a través de otro conjunto de mensajes emitidos desde los sensores lógicos de comunicaciones de salida: *DataWriter*.

Al conjunto de mensajes entrantes que generan como resultado un servicio se le denomina "patrón de mensajes". Un componente de control puede ofrecer diferentes patrones de mensajes, aunque es deseable que un componente de control actúe sólo con un patrón de mensajes. Por tanto, los parámetros de calidad de servicio de los componentes de control están referidos a patrones de mensajes y no sólo a mensajes individuales.

Una vez se ha dado un patrón de mensajes, el componente de control procesa la información y proporciona el servicio. Algunos de los mensajes entrantes pueden adelantar el proceso de obtención del servicio al ser preprocesados, y depender el resultado de la llegada de los mensajes restantes. Asimismo, algunos de los mensajes resultantes del servicio pueden también enviarse antes de finalizar todos los cálculos que el componente de control debe realizar. En la Figura 67 se muestran los intervalos de tiempo implicados en solicitad, procesar y proporcionar un servicio teniendo en cuenta el posible solapamiento de mensajes a través del tiempo.

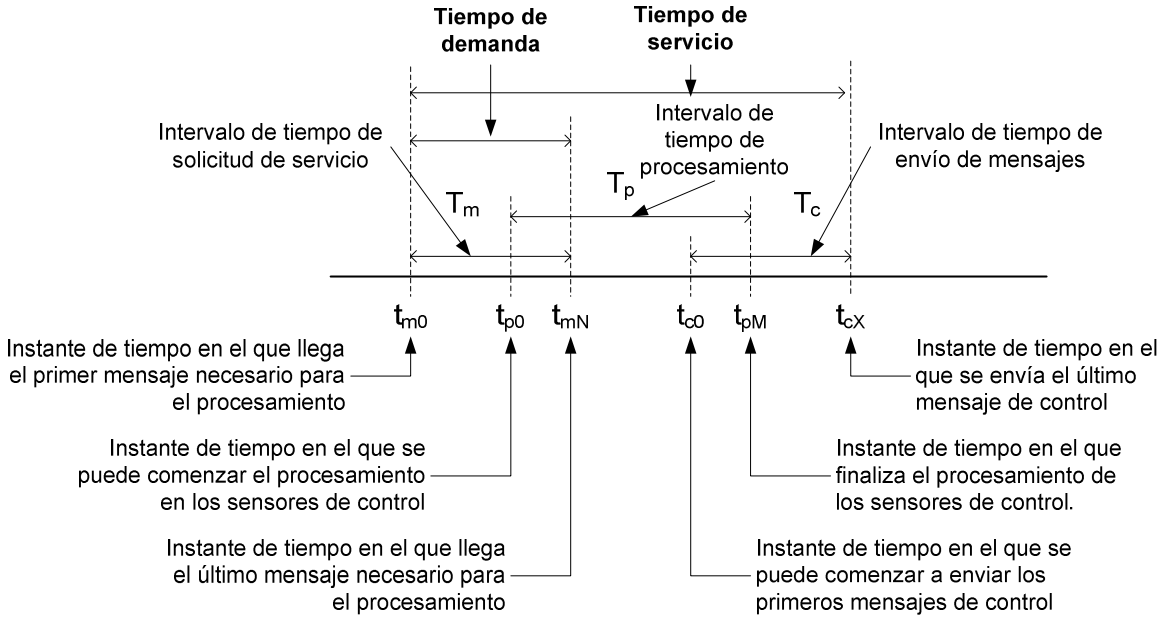


Figura 67. Intervalos de tiempos en la gestión del servicio en un componente de control.

El tiempo de llegada de los mensajes de solicitud de servicio (T_m), o el tiempo de envío de los mensajes de la acción de control (T_c), de los patrones de mensajes puede solaparse tanto con el tiempo de procesamiento como entre ellos. En el cálculo de los siguientes parámetros se considera el tiempo de demanda máximo (T_m) como el mayor de los tiempos de demanda de servicio de los patrones de mensajes válidos en las diversas secuencias de llegada de mensajes al componente de control. Lo mismo debe entenderse para el tiempo de servicio calculado como la diferencia del máximo tiempo entre que se envía el último mensaje de control válido y la llegada del primer mensaje que permite el procesamiento (T_{cx}- T_{m0}).

Los máximos tiempos de los patrones de mensajes se obtienen a partir de los casos más estrictos de los tiempos anteriores, dando lugar a los tiempos mostrados en la Figura 66. El caso más estricto de la demanda de servicio se da cuando sólo con la llegada de todos los mensajes necesarios para solicitar el servicio se puede comenzar a procesar el mismo. De forma similar, el máximo tiempo de servicio se da, aparte de por el caso anterior, cuando los mensajes de respuesta del servicio sólo pueden comenzarse a dar cuando todos los sensores lógicos de control finalizan su procesamiento (Figura 68).

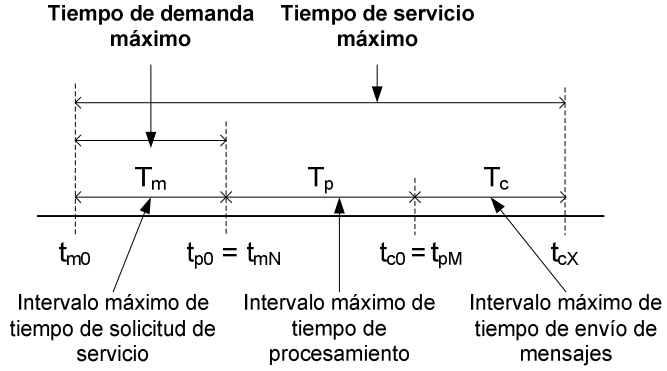


Figura 68. Intervalos máximos de tiempos en la gestión del servicio en un componente de control.

Para analizar los patrones de mensajes es necesario analizar los tiempos significativos en un mensaje individualmente para extender el análisis al conjunto de los mismos. Cuando un mensaje llega al componente de control, el mensaje tiene un tiempo de validez durante el cual su información puede usarse para procesar la solicitud del servicio, una vez transcurrido ese tiempo la información del mensaje no es válida y por tanto se debe esperar al siguiente mensaje. Un mensaje caduca, bien por la llegada de otro más reciente que actualiza la información o bien porque se considera que la antigüedad de la información del mensaje ya no es relevante. Estos aspectos generan unas restricciones que se muestran en la Figura 69.

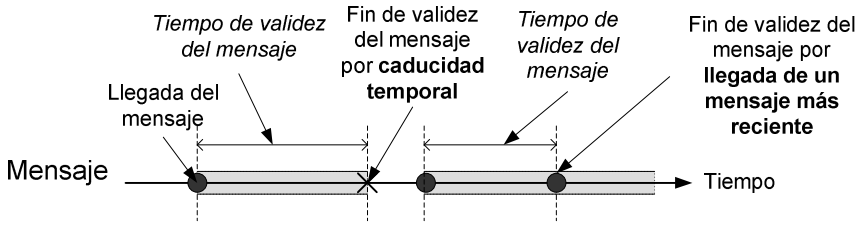


Figura 69. Instantes temporales de la validez de un mensaje.

Los componentes de control pueden tener diversos sensores de comunicaciones, tanto de entrada como de salida, lo que implica que el inicio de procesamiento de los mensajes de solicitud de un servicio está condicionado a la validez de los mensajes en el patrón de mensajes. En la Figura 70 se presenta el concepto de "ventana válida de solicitud de servicio" por medio del cual el componente de control puede comenzar a procesar los mensajes de todos los sensores lógicos de comunicaciones de entrada y por tanto proporcionar el servicio.

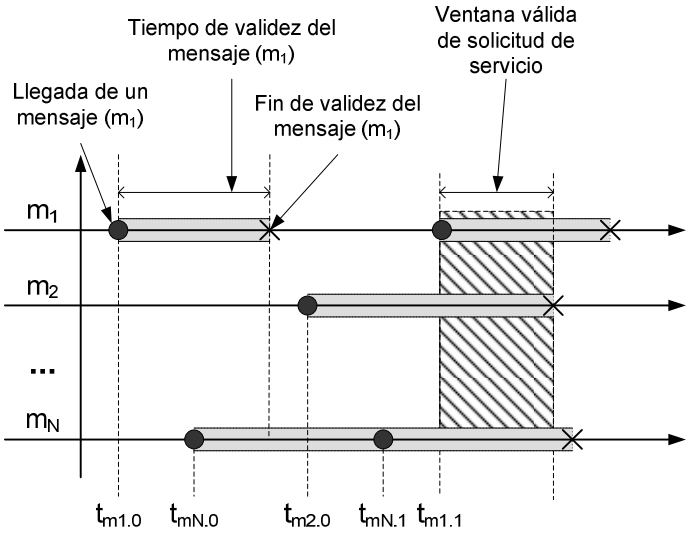


Figura 70. Ventana válida de solicitud del servicio.

Los tiempos anteriores son la base de los cálculos de los parámetros intrínsecos de calidad de servicio y tienen la ventaja de gestionarse por medio de las políticas de calidad de servicio de cada sensor lógico. En la Figura 71 se muestran las tres políticas de QoS, empleadas en detectar la ventana de recepción de mensajes para un componente de control.

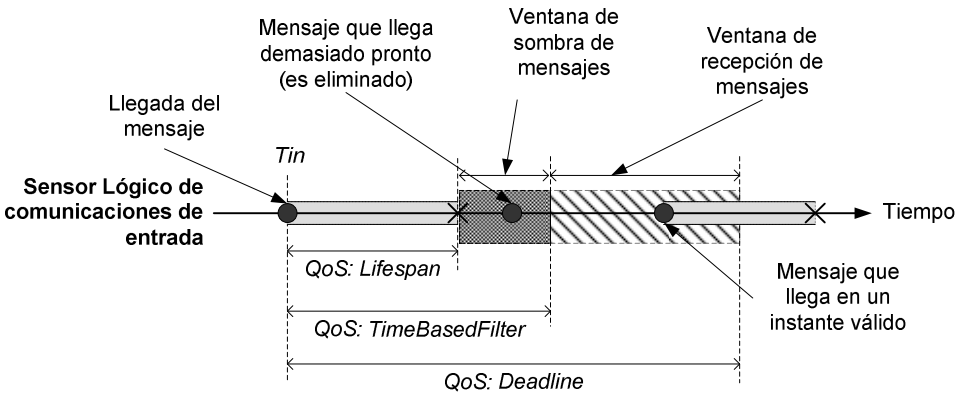


Figura 71. Ventana de sombra y de recepción de mensajes

Por medio de la política de calidad de servicio *Lifespan* se determina el tiempo durante el que un mensaje puede considerarse válido, por ejemplo la antigüedad de una medición de la distancia a un obstáculo en un robot móvil puede ser útil durante los primeros instantes, pero a medida que el robot se mueve, la información de la distancia obtenida pierde validez y justifica la adquisición de una nueva distancia; lifespan se refiere al tiempo durante el cual la última distancia recibida puede ser utilizada para realizar los cálculos del componente de control.

La política *TimeBasedFilter* especifica el intervalo mínimo de tiempo que debe transcurrir para que se admita como válido el siguiente mensaje. Esta política requiere una justificación: su objetivo principal no es ralentizar la atención de mensajes, sino permitir dar un tiempo al sensor lógico en poder procesar el mensaje anterior, lo que permite sincronizar las diferentes fuentes de mensajes, en lugar de tener que estar esperando una coincidencia de llegada de mensajes, por tanto define el inicio de la ventana de recepción de mensajes. Finalmente el tiempo especificado en la política *Deadline*, permite acotar el final de la ventana de recepción de mensajes, especificando cuánto tiempo se puede esperar a que llegue un mensaje. Los tiempos que delimitan la ventana de recepción se muestran en la fórmula 51 y la fórmula 52.

$$T_{VRM(start)} = T_{in} + timebasedfilter \tag{51}$$

$$T_{VRM(end)} = T_{in} + deadline \tag{52}$$

En la fórmula 53 se muestra el cálculo de la duración de la ventana de recepción de un mensaje por parte de un sensor lógico de comunicaciones de entrada.

$$VRM = T_{in} + \left(deadline - \max\left(lifespan, timebasedfilter\right)\right) \tag{53}$$

De la misma forma, la ventana de sombra de recepción de mensajes especifica un intervalo de tiempo durante el cual los mensajes entrantes se reciben fuera del tiempo establecido (fórmula 54) para procesarlos.

$$VSM = T_{in} + \left(timebasedfilter - lifespan\right) \tag{54}$$

De las combinaciones de la VRM y VSM, se pueden dar tres casos en lo que respecta a los valores de los parámetros de las políticas de QoS (Figura 72). La especificación DCPS establece que la separación mínima entre mensajes no debe ser mayor que el periodo de espera de mensaje. Esta relación es obvia y la consecuencia se da cuando se negocian las políticas de QoS entre elementos ya que el valor *TimeBasedFilter* de un componente puede ser mayor que el *Deadline* de otro elemento.

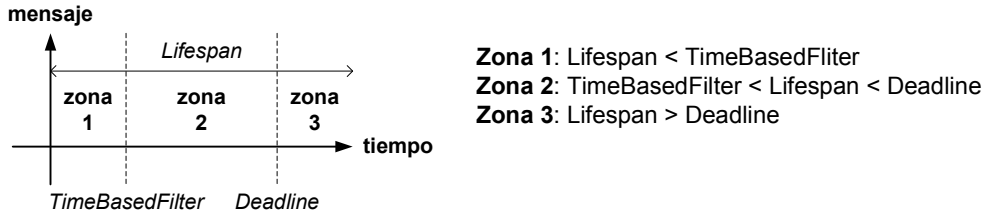


Figura 72. Escenarios en función de las políticas de QoS.

La duración del *Lifespan* puede ubicarse en tres áreas diferentes con respecto a las otras políticas de QoS.

1. *Lifespan* < *TimeBasedFilter*: Esta situación implica que la validez del mensaje es menor que el tiempo mínimo de llegada de los mismos, lo que directamente quiere decir que siempre habrá zona de sombra dado que no llegarán mensajes y el último mensaje anteriormente ya no se considera válido.
2. *TimeBasedFilter* < *Lifespan* < *Deadline*: En esta situación el mensaje tiene una validez temporal mayor que el tiempo mínimo de admisión del mismo y menor que el plazo temporal. Esta situación implica tener zona de sombra, pero también que el control cumpla siempre el deadline.

3. *Lifespan* > *Deadline*: Esta situación es más estable que la anterior, ya que implica que la información de un mensaje puede suplir la ausencia de la llegada de mensajes retrasados. Sin embargo puede considerarse incompatible, ya que el plazo temporal indica una preferencia en el máximo retraso admisible del mensaje, lo que a su vez puede ser considerado como cota superior de la validez de un mensaje.

La propagación de los valores de las políticas de QoS implica una relación muy estricta y dependiente entre los elementos. Por ejemplo, un valor de *TimeBasedFilter* en un sensor lógico afectará directamente al instante de tiempo en que el sensor lógico que envía mensajes al siguiente sensor lógico deberá enviar un mensaje. Lo mismo con el *Deadline*, ya que fuerza a enviar un mensaje desde el sensor lógico anterior.

La dependencia de los valores de configuración entre sensores lógicos implica una difusión de las restricciones entre ellos que finalmente llega a los orígenes de los datos, incluyendo los canales de comunicaciones, lo que permite configurar los parámetros de calidad de servicio de los productores de información tales como frecuencias de muestreo o el *deadline*.

Extendiendo los conceptos de ventana de sombra del mensaje y de ventana de recepción del mensaje, al conjunto de sensores lógicos de comunicaciones de entrada que pueda tener un componente de control, se define la ventana válida de solicitud del servicio (VVSS) como el intervalo de tiempo en el que la llegada de un patrón de mensajes asegura el servicio ofrecido por el componente de control. Asimismo, las ventanas de sombra de los mensajes individuales de los *DataReader* y *Listener* pueden generar una ventana de sombra de solicitud de servicio (VSSS), donde la llegada de un mensaje no generará un servicio. Ambas ventanas se muestran en la Figura 73.

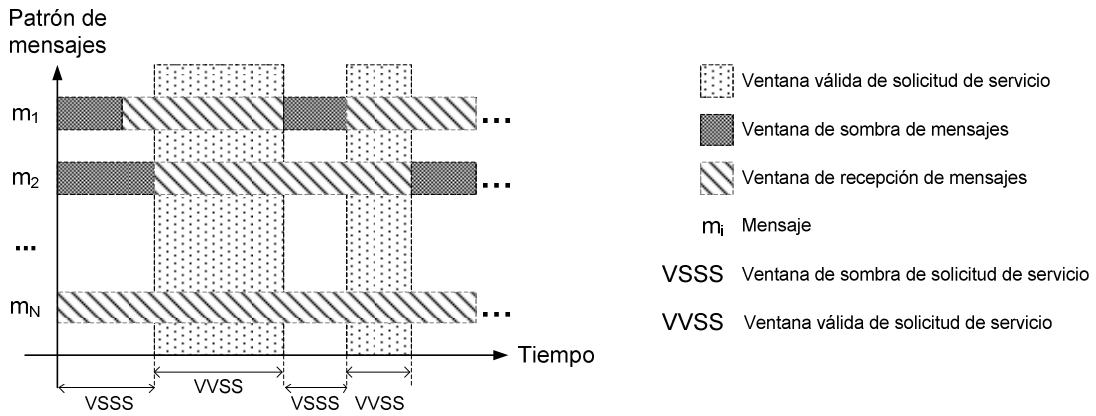


Figura 73. Ventana de validez de los mensajes en un componente de control.

Tanto la VVSS como la VSSS pueden variar en su frecuencia, pudiendo hacer que el componente de control no pueda responder a solicitudes de servicio, dado que nunca se dan las coincidencias de las ventanas de recepción de mensajes. El hecho de poder configurar los parámetros de QoS de un componente de control hace que sea posible configurar los tiempos de demanda de servicio ajustando las VVSS. Estos ajustes permiten un control sobre los parámetros de QoS representativos.

La duración de la VVSS debe ser compatible con el tiempo máximo de solicitud de servicio (Figura 68) de manera que si la duración de ninguna de las VVSS es mayor que el tiempo máximo de solicitud de servicio, el componente de control no es funcional debiendo dar lugar a los correspondientes ajustes.

3.4.3.4.1 Demanda del servicio

La demanda de servicio representativa describe lo solicitado que está el componente de control. Para que el componente de control entienda que se le ha solicitado un servicio no basta con la llegada un mensaje, sino que tiene que llegar una secuencia válida de mensajes (SVM) dentro de una ventana válida de mensajes. El cálculo de la demanda se realiza a partir de los intervalos de tiempo entre dos SVM consecutivas (fórmula 55), empleados para calcular el promedio del intervalo de demanda (fórmula 56).

$$T_i = t_e(SVM_i) - t_e(SVM_{i-1}) \tag{55}$$

$$E[T]_{rep} = \left[\sum_{i=1}^{N}(T_i)\right] \Big/ N \tag{56}$$

Para expresar los tiempos de demandas en forma de tasa de SVM por unidad temporal se define la demanda de servicio representativa. En la fórmula 57se muestra la fórmula correspondiente.

$$\lambda_{rep} = 1/E[T]_{rep} \tag{57}$$

De forma similar, el mínimo intervalo de tiempo entre dos secuencias válidas de mensajes que se puede dar definirá la máxima demanda de servicio representativa (fórmula 58).

$$\lambda_{rep\_\max} = \max[T_i]_{i=1}^{N} \tag{58}$$

3.4.3.4.2 Tiempo de servicio

El tiempo de servicio representativo es el tiempo que tarda una solicitud de servicio en producir el resultado. En un componente de control que genere una acción de control, el resultado será la emisión de la correspondiente secuencia de mensajes de control; mientras que en un componente de control dedicado, por ejemplo al almacenar el estado de un sistema, el resultado será el final del proceso de almacenamiento. Para calcular el tiempo de servicio representativo se definen dos instantes temporales: t_{in}, cuando se termina de recibir el último de los mensajes representativos de la SVM, y t_{out}, cuando se termina la última de las acciones consecuentes a la atención de la solicitud del servicio de la SVM. La fórmula correspondiente se muestra en la fórmula 59.

$$S_i = t_{in}(SVM_i) - t_{out}(SVM_i) \tag{59}$$

El tiempo de servicio representativo máximo, representado en la fórmula 60, indica el máximo de los tiempos de servicio representativos que se han dado a lo largo del intervalo de medición.

$$S_{rep\_\max} = \max[S_i]_{i=1}^{N} \tag{60}$$

A partir del tiempo de servicio representativo de una SVM, se puede obtener, a lo largo del intervalo de medición, el promedio del tiempo de servicio de todas las SVM. De esta forma se hace una estimación representativa para el componente de control. La función correspondiente se puede ver en la fórmula 61.

$$E[S]_{rep} = \left[\sum_{i=o}^{n} S_i\right] \Big/ N \tag{61}$$

3.4.3.4.3 Tasa de servicio

La tasa de servicio representativa de un componente de control se obtiene como la inversa del tiempo de servicio representativo (fórmula 62) y describe la cantidad de servicios cumplidos en función de la cantidad de tiempo invertida en cumplir los SVM que han llegado al componente de control.

$$\mu_{rep} = 1/E[S]_{rep} \tag{62}$$

En la fórmula 63 se muestra la fórmula de la tasa de servicio representativa máxima que indica el máximo número de SVM que es capaz de atender el componente de control por unidad de tiempo medido.

$$\mu_{rep\_\max} = 1/S_{rep\_\max} \tag{63}$$

3.4.3.4.4 Carga

La carga representativa de un componente de control (fórmula 64) indica la relación entre las solicitudes de servicio y los servicios atendidos. En el caso de los componentes de control es de gran utilidad, pues indica aspectos como capacidad del mismo en asumir más solicitudes válidas de servicio.

$$\rho_{rep} = \lambda_{rep} / \mu_{rep} \tag{64}$$

La carga representativa máxima se obtiene por medio de la fórmula 65, para contemplar la situación más estricta y se obtiene prorrateando la demanda máxima de servicio con la menor tasa de servicio.

$$\rho_{rep\_\max} = \lambda_{rep\_\max} / \mu_{rep\_\min} \tag{65}$$

La carga representativa máxima en un componente de control indica la cota superior de los servicios que el componente de control es capaz de admitir. Como cota superior indica un valor ideal difícilmente alcanzable, pero especialmente útil para comprobar el nivel de optimización que es posible obtener, por medio del descenso de la carga máxima representativa.

3.4.3.4.5 Utilización

La utilización de un componente de control se calcula comparando la carga representativa con el máximo valor de la carga, de esta forma se tiene la cota superior del uso que puede hacer del componente de control (fórmula 66).

$$U_{rep} = \rho_{rep} / \rho_{rep\_máx} \tag{66}$$

En el caso de los componentes de control, la utilización indica el margen real que se obtiene a partir de las optimizaciones que se hayan hecho en el componente de control.

3.4.3.4.6 Ocupación

La ocupación de un componente de control (fórmula 67) se calcula dividiendo la demanda de servicio entre la tasa de servicio máxima, de esta forma se puede estimar la capacidad de admitir más servicios del componente de control hasta alcanzar la máxima carga admisible.

$$\sigma_{rep} = \lambda_{rep} / \mu_{rep\_\max} \tag{67}$$

La obtención de la ocupación representativa es muy similar a la carga representativa máxima. Sin embargo, la ocupación es una cota superior real a diferencia de la carga representativa máxima, que muestra un límite.

3.4.3.4.7 Disponibilidad

Para el cálculo de la disponibilidad representativa se emplea la carga representativa de los sensores lógicos, adaptada a la disponibilidad de los componentes de control (fórmula 68)

$$\delta_{rep} = \left(1 - \rho_{rep}\right) \tag{68}$$

La disponibilidad representativa presupone una carga máxima del 100%. Sin embargo un componente de control puede estar sobrecargado, por lo que la disponibilidad es un parámetro orientativo de interés para el usuario del componente de control, a diferencia de la capacidad representativa que se verá a continuación.

3.4.3.4.8 Capacidad

De manera análoga a la disponibilidad, la capacidad representativa de un componente de control se calcula como la diferencia entre el punto de sobrecarga, carga igual a uno, y la carga representativa (fórmula 69)

$$\kappa_{rep} = \left(1 - \sigma_{rep}\right) \tag{69}$$

Tal como se comentó en el punto anterior, la capacidad es un parámetro que se obtiene a partir de la ocupación por lo que indica una característica interna, lo que recomienda su uso para la gestión de los sensores lógicos que lo componen.

3.4.3.4.9 Productividad y rendimiento

La productividad representativa es la tasa de servicios (fórmula 62) mientras que el rendimiento relaciona los servicios ofrecidos con la capacidad máxima de ofrecerlos (fórmula 70), de forma similar a como se ha visto

$$\eta_{rep} = \mu / \mu_{rep\_\max} \tag{70}$$

En el caso de los componentes de control, tanto la productividad como el rendimiento son de especial interés ya que están calculadas en función de los servicios ofrecidos, y no solo en función de los mensajes.

3.5 Conclusiones

En el este capítulo se ha descrito la propuesta del modelo FSACtrl que permite diseñar de forma organizada arquitecturas de control inteligente distribuido con soporte para la calidad de servicio. Al emplear elementos de comunicaciones basados en el paradigma de publicación-suscripción el sistema constituye una base excelente para implementar sistemas de control basados en eventos.

Las comunicaciones se fundamentan en el modelo DCPS por lo que implementan un modelo basado en el paradigma de publicación-subscripción. El control toma el modelo SWE como base para los sensores de control interconectados en una estructura llamada SLG. El modelo SWE se amplía organizando jerárquicamente los componentes de control en una estructura de árbol llamada CCT.

La conexión de los elementos de comunicaciones y los componentes de control se realiza mediante una estructura de datos denominada LNT, consistente en una estructura jerárquica que aísla los detalles específicos del control y de las comunicaciones, pero que a su vez proporciona una visión del sistema adaptada a la funcionalidad de los componentes.

La arquitectura presentada parte de la sinergia de los estándares en los que está inspirada. Poder emplear los interfaces estándares de DDS o SWE ofrece una visión de FSACtrl como sistema de comunicaciones y un de sistema de control.

En el siguiente capítulo se presenta el proceso de diseño basado en el modelo propuesto, es de especial interés puesto que permite comprobar la validez del modelo mediante un entorno de diseño de sistema basado en FSACtrl.

4 Diseño de sistemas basados en FSACtrl

4.1 Introducción

4.1.1 Motivación

En el capítulo anterior se han presentado los componentes y operaciones de la arquitectura FSACtrl. Además, se han presentado las fórmulas que permiten asignar parámetros a los elementos de la arquitectura. A partir de estos elementos se potencia el desarrollo de sistemas más complejos, como sistemas distribuidos de control o sistemas de control basados en agentes.

En el presente capítulo se describe cómo aplicar la arquitectura FSACtrl como middleware de comunicaciones basado en el estándar DDS. Además, al emplear el estándar DDS, el soporte a la QoS que ofrece el sistema permite emplear los parámetros y las políticas para monitorizar el funcionamiento y gestionar la configuración del sistema.

Para dar soporte al diseño del sistema se ha desarrollado una plataforma para el desarrollo de sistemas a partir de la cual insertar y gestionar todos los elementos de la arquitectura, especialmente los componentes de control. Por medio del entorno es posible comprobar la validez en el funcionamiento de un sistema basado en la arquitectura FSACtrl así como la utilidad en la aplicación a entornos reales de los parámetros y las políticas de calidad de servicio.

4.1.2 Entorno de diseño

Para facilitar el diseño y permitir comprender y validar la arquitectura propuesta, se ha desarrollado un entorno de diseño visual. El entorno de diseño se ha desarrollado en C++, empleando la plataforma Microsoft Visual Studio sobre Windows. La aplicación genera los correspondientes archivos en C++ estándar y los correspondientes archivos de trabajo en el estándar XML.

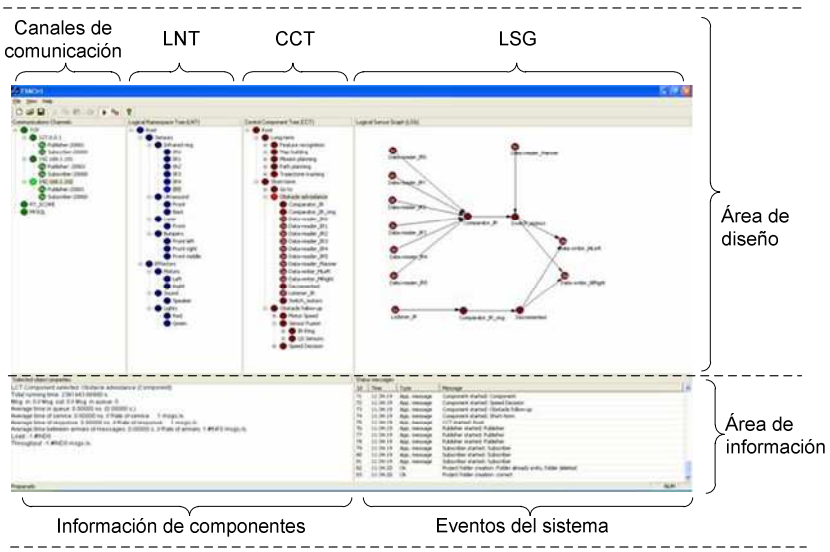


Figura 74. Aplicación de diseño de sistemas basados en FSACtrl.

La pantalla principal, así como los marcos que organizan los elementos de la arquitectura se muestran en la Figura 74. La aplicación proporciona una única interfaz en la que se accede a todos los elementos del *Frame*: desde los canales de comunicaciones y sus correspondientes adaptadores a los componentes de control con los sensores lógicos que los componen, enlazados por los datos lógicos.

La parte superior de la ventana contiene los marcos de diseño de la aplicación, con las estructuras presentadas en el capítulo anterior. La parte inferior izquierda muestra la información concerniente a los elementos y los eventos del sistema. La parte superior tiene cuatro áreas, a continuación se enumeran de izquierda a derecha. Inicialmente se tiene acceso a los canales de comunicaciones junto a los adaptadores (*Publisher* y *Subscriber*) asociados a cada uno. A continuación se presenta el LNT mostrando la jerarquía en árbol de los datos lógicos. Seguidamente se exponen los componentes de control jerarquizados en forma de árbol para formar el CCT. Finalmente se tiene acceso al contenido del componente de control seleccionado en el CCT, es decir a los sensores lógicos de comunicaciones y de control así como las conexiones entre ellos que forman la ruta de los mensajes.

El proceso de diseño del sistema no es secuencial, durante el mismo puede modificarse el LNT, cambiar canales de comunicaciones o variar la composición de los sensores lógicos. Por lo tanto, los pasos en los que está organizado el capítulo no necesariamente siguen la secuencia que se debe realizar en el diseño de un sistema basado en la arquitectura FSACtrl.

4.1.3 Descripción del capítulo

A continuación, en el apartado 4.2, se presentan las operaciones por medio de las cuales se generan y organizan los elementos de la arquitectura. La aplicación permite insertar los elementos y realizar las conexiones entre ellos para así generar prototipos de sistemas más complejos. A través del entorno se accede a los elementos que componen las estructuras principales de la arquitectura (LSG, CCT y LNT) además de la asociación de las políticas de QoS y los parámetros de QoC a los elementos.

El apartado 4.3 está dedicado a la gestión de los eventos a través del entorno de desarrollo. Se detalla cómo crear condiciones y cómo asociar eventos y acciones a las mismas y las implicaciones en el funcionamiento del sistema. A partir de la gestión de los eventos, se introduce el "ciclo integral de la calidad" que permite la gestión dinámica y automática de los elementos del sistema.

Finalmente, se presentan las bases de la implementación de un sistema de agentes basándose en la arquitectura FSACtrl. Se presentan los elementos de la arquitectura FSACtrl y cómo combinarlos para crear agentes con diferentes responsabilidades. También se presentan las operaciones fundamentales para el movimiento de agentes a través del sistema.

4.2 Proceso de diseño de sistemas basados en FSACtrl

4.2.1 Fuentes de los datos: diseño de las comunicaciones

Cuando se comienza a diseñar un sistema, la aplicación muestra por defecto los canales de comunicación que se hayan definido en el sistema. Cabe destacar que cada canal de comunicaciones tiene sus particularidades, por lo que en los canales no es habitual jerarquizar la estructura y por tanto cada uno de ellos se ha decidido implementar

atendiendo a un criterio. Inicialmente se dispone de tres por defecto: TCP-IP, RTSCore y MySQL [Reese et al., 2002].

El canal de comunicación TCP implementa las comunicaciones basadas en el modelo TCP-IP, añadiendo los elementos *Publisher* y *Subscriber* a direcciones y puertos para leer o escribir la información.

RTSCore [Posadas, 2003] es un sistema basado en una pizarra distribuida [Nii, 1989], desarrollado por el grupo de investigación y ampliamente probado en proyectos de domótica [Camacho et al., 2007], robótica [Posadas et al., 2008] y entornos industriales amplios [Poza et al., 2003].

MySQL, aún siendo un sistema gestor de bases de datos, se ha incluido como ejemplo de abstracción de una base de datos como modelo de comunicaciones, dado que en diversos proyectos de automatización las bases de datos son importantes y van tomando cada vez más un papel activo debido a la capacidad de ofrecer un entorno de almacenamiento y consulta de datos universalizado y a la potencia añadida de las funciones implementadas y de los procesos de minería de datos que se pueden realizar sobre los datos almacenados.

4.2.1.1 Comunicaciones TCP-IP

En FSACtrl, la unidad mínima de los canales de comunicación son los adaptadores, en el caso de TCP-IPlos adaptadores de TCP-IP representan direcciones IP, por tanto el primer paso consiste en agregar la dirección IP de la que se desea leer o escribir información. Una vez establecida la dirección IP, las conexiones a los puertos se establecen por medio de los elementos *Publisher* y *Subscribers* (Figura 75). Esto se realiza así para facilitar la programación de la calidad de servicio a nivel de componentes DCPS y depender lo menos posible de otras colas de mensajes.

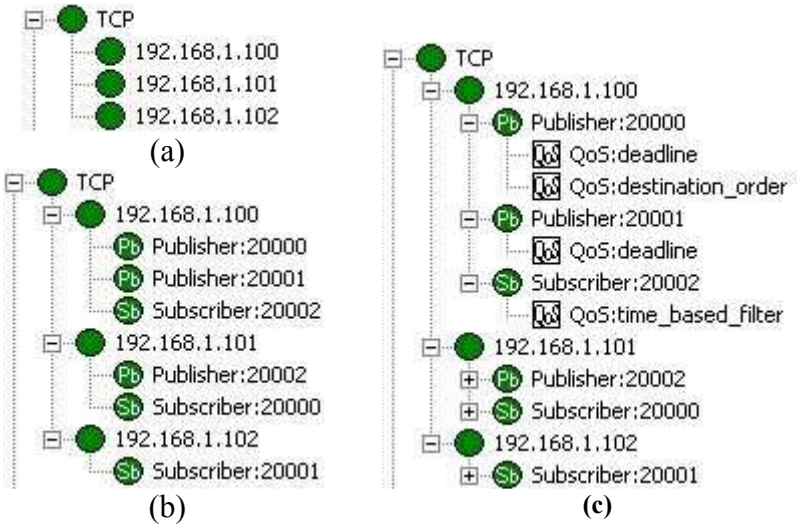


Figura 75. Pasos a seguir en la creación de las comunicaciones TCP-IP.

A cada *Publisher* o a cada *Subscriber* se le asocian las políticas de calidad de servicio y se configuran los valores particulares de cada una de ellas. Tal como establece el estándar DCPS.

4.2.1.2 Comunicaciones RTSCore

El canal de comunicaciones RTSCore organiza la información en objetos llamados *datos* que abstraen el origen y destino de los mismos. El conjunto de todos los datos es

el espacio de datos de la pizarra distribuida. Las aplicaciones escriben y leen de los datos, o se conectan a los mismos para ser avisadas de cualquier modificación.

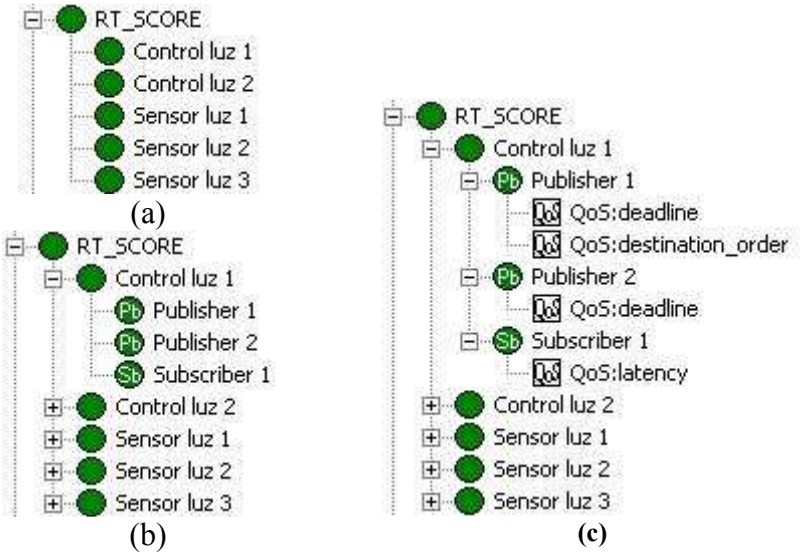


Figura 76. Pasos a seguir en la creación de las comunicaciones RTSCORE.

El primero de los pasos será decidir cuáles de los datos son los que interesan para la aplicación de control que se esté diseñando (Figura 76). Una vez seleccionados los datos, y de forma similar a como se hacía con las conexiones TCP, se deberá conectar los elementos *Publisher* o *Subscriber* a los mismos. Finalmente se les asociarán las políticas de calidad de servicio correspondientes.

4.2.1.3 Comunicaciones MySQL

Las conexiones con MySQL son más complejas, ya que la abstracción que se puede hacer de una base de datos puede variar desde una visión muy detallada, a nivel de campos de las tablas, hasta una visión general en la que sólo se tenga en cuenta la conexión y la base de datos con las que se desee trabajar. En el caso del entorno de diseño FSACtrl se ofrece la visión general ya que ofrece menos restricciones y es la visión que habitualmente se emplean en los lenguajes de programación de acceso a bases de datos.

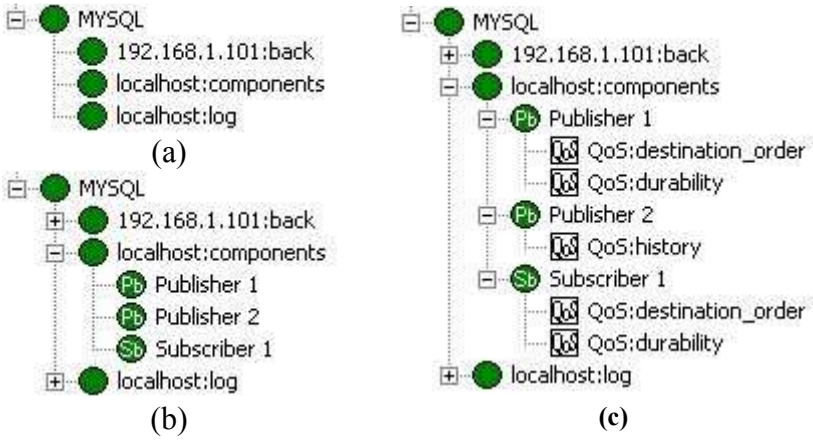


Figura 77. Pasos a seguir en la creación de las comunicaciones mySQL.

El proceso de diseño comienza definiendo la conexión con el servidor de bases de datos y la base de datos con la que se establecerá la comunicación. Seguidamente se definen

los *Publisher* y *Subscriber* correspondientes dependiendo de cuántas conexiones se deban realizar, junto a las políticas de calidad de servicio correspondientes a cada uno (Figura 77). Los accesos a las bases de datos se realizan mediante consultas y, en los casos que se requieran, los correspondientes resultados. Para implementar las consultas se requiere un *Publisher* y un *Subscriber*, ambos deben proporcionar el correspondiente control de mensajes, implementado a través de la política *DestinationOrder*, para poder asociar las respuestas del SGBD con las correspondientes consultas.

4.2.2 Gestión del control: CCT y LSG

4.2.2.1 CCT

Una vez establecidos los orígenes y los destinos de los datos se puede pasar a diseñar el sistema de control. En este caso suele ser habitual ir formando los componentes de control de los más genéricos a los más específicos. Inicialmente sólo se tiene un componente de control, denominado *root*, a partir de ahí se irá desarrollando la jerarquía de componentes de control. Para establecer un árbol de componentes de control hay que ir añadiendo los componentes de control al componente de control de jerarquía superior (más cercana al *root*). Por ejemplo, en la Figura 78-a se tienen tres componentes de control dependientes del *root*, cada componente de control es el responsable de una de las áreas de control de un sistema domótico. Dentro de cada componente de control se definen otros componentes responsables de las subáreas (Figura 78-b). Esta inclusión jerárquica puede realizarse con tanto detalle como lo requiera el control.

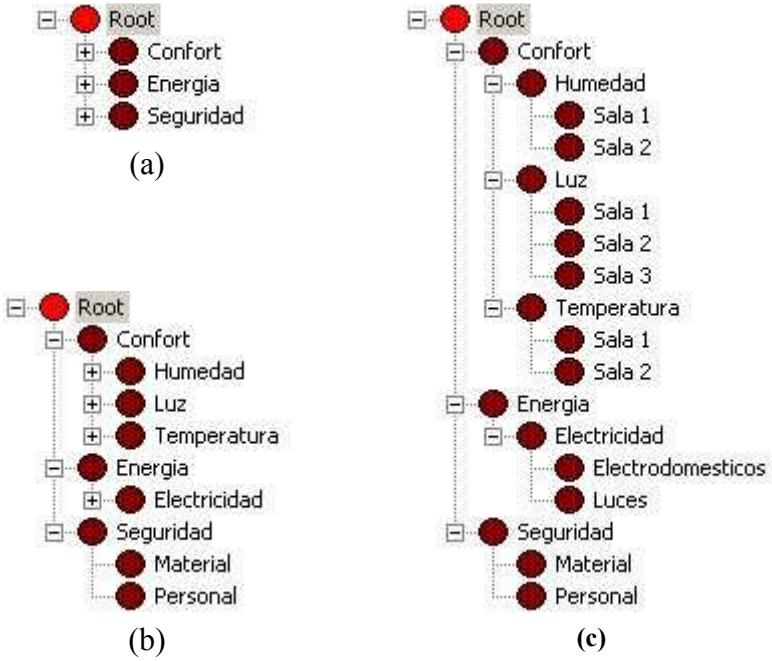
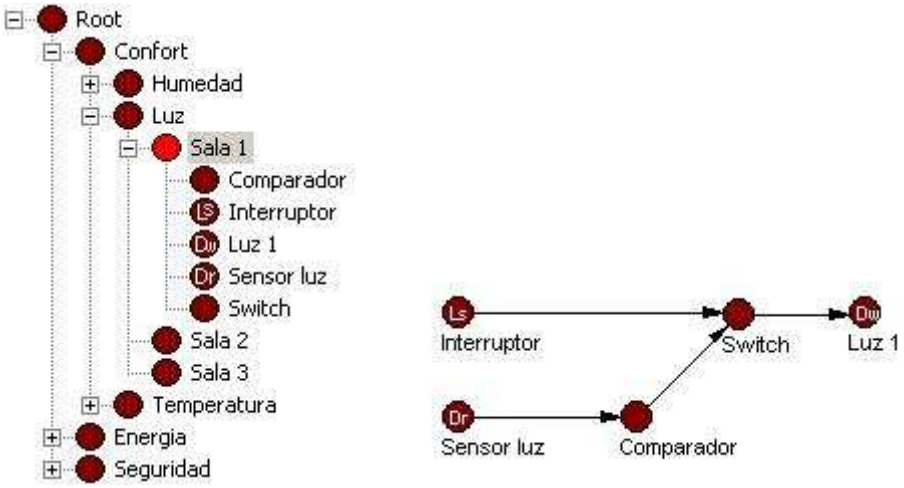


Figura 78. Evolución en la profundidad de los componentes de control.

Cuando ya se ha definido un componente de control, es posible guardarlo de tal forma que es reutilizable. Eso es interesante especialmente en lo que respecta a componentes de control de utilidad genérica. Por ejemplo, en la Figura 78-c si el control de Luz del área de Confort es idéntico para las tres salas, pero variarán los orígenes (sensores) y destinos (actuadores) de los datos, que están en el LNT, pero no en los algoritmos de control, por lo que el hecho de poder reutilizar un componente de control proporciona un gran potencia en el diseño de los sistemas.

4.2.2.2 LSG

Para crear un LSG se deben agregar sensores lógicos a un componente de control. Normalmente se definen primero los sensores de comunicaciones, que determinarán las entradas y las salidas de datos de los componentes de control. Seguidamente se van incluyendo sensores lógicos o componentes de control y las correspondientes conexiones para ir formando así el componente de control final y las rutas que seguirán los mensajes que entre ellos se envíen.



(a) **(b)**

Figura 79. Sensores lógicos de un componente de control y detalle del LSG correspondiente.

En el modelo SWE se establecía un componente, llamado *Process Method*, que actuaba a modo de plantilla de componentes de control preestablecido. En la aplicación de diseño también se dispone de archivos en los que definir componentes de control reutilizables a los que sólo habrá que variar sus parámetros internos. Por ejemplo, en la Figura 79-a está seleccionado el componente de control correspondiente a la gestión de la luz de la "Sala 1" mostrándose su contenido en la Figura 79-b. Los sensores lógicos y las conexiones que forman el componente de control son similares para los componentes de control "Sala 2" y "Sala 3" variando únicamente en los datos lógicos y las configuraciones internas correspondientes.

Algunos sensores lógicos de control también se corresponden con algoritmos de control básico, en la Figura 79-b son el "Comparador" que compara la entrada con una referencia preestablecida y el "Switch" que acepta o no, dependiendo de su estado, los valores del "Comparador".

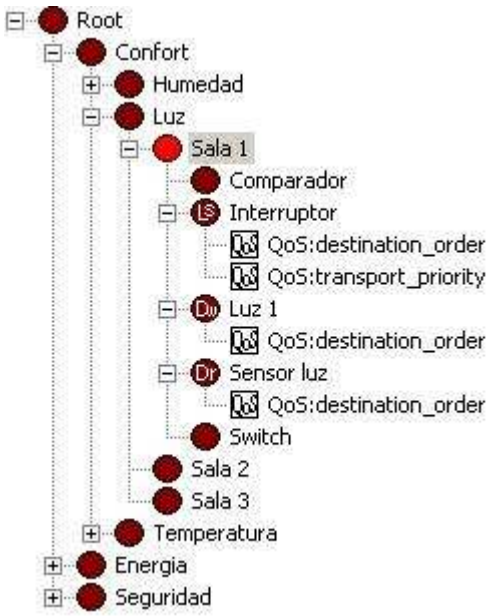


Figura 80. Sensores de comunicaciones con las correspondientes políticas de QoS.

A los sensores lógicos, tanto los de comunicaciones como los de control, se les asocian las políticas de calidad de servicio (Figura 80). Lo habitual es asociarlas a los sensores lógicos de comunicaciones, ya que son éstos los que se conectarán a través del LNT con los elementos de comunicaciones, lo que no excluye el emplear las políticas de QoS en los sensores lógicos de control. Por ejemplo, la política *QoSDeadline* es fundamental para un sensor lógico de control que requiera comprobar por sí mismo el cumplimiento del plazo temporal de los mensajes que emplea para los algoritmos de control.

4.2.3 Conexión del control y de las comunicaciones: LNT

Una vez se tienen definidas tanto las conexiones que proporcionan los datos al sistema como los componentes de control, es el momento de conectarlos. Para ello, se deben crear, caso de no existir, los datos lógicos que servirán de enlace entre las comunicaciones y el control. Los datos lógicos necesarios surgen desde el momento en que se comienza a definir los canales de comunicaciones o los componentes de control, ya que éstos representan los orígenes y destinos de los datos.

Para crear el LNT se comienza por insertar nodos lógicos que se conectan unos a otros para ir formando datos lógicos del árbol de datos lógicos. Por medio de los nodos lógicos se construye el LNT proporcionando una abstracción del sistema. Los datos lógicos más cercanos al *root*, representan información general o un conjunto de información agrupada según un criterio. Por ejemplo, en la Figura 81-a, el dato "root/Domótica" representa todas las áreas de gestión domótica, mientras que el dato "root/Sistema" representa el hardware empleado en el sistema.

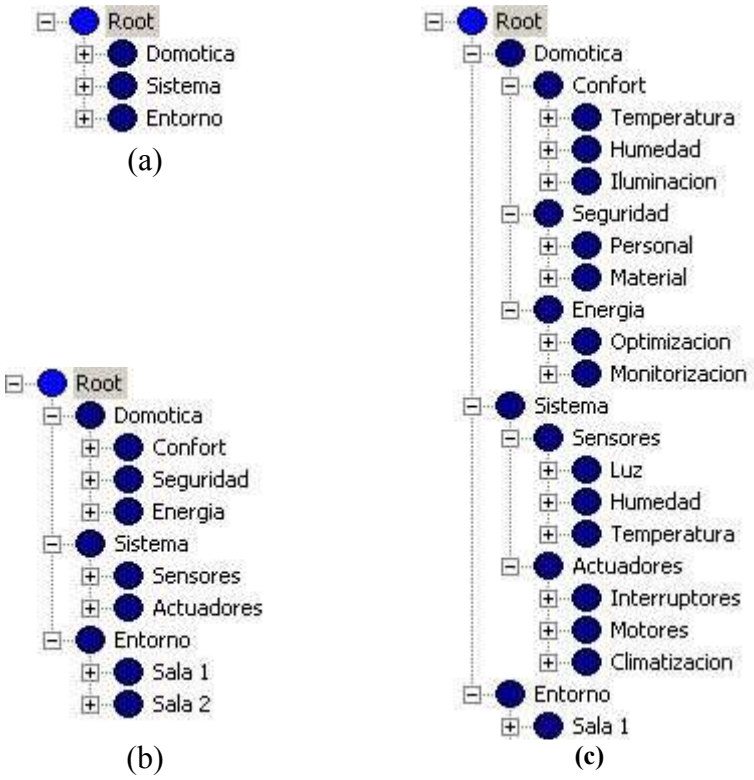


Figura 81. Evolución en la profundidad del LNT.

A medida que se profundiza en los datos lógicos, alejándose del *root*, se accede a un mayor nivel de detalle de la abstracción. En la Figura 81-b, se aprecian las tres áreas de la domótica, o la división del hardware en sensores y actuadores. En la Figura 81-c el nivel aumenta hasta presentar las subáreas de la domótica o la organización correspondiente de los sensores. Se puede crear un LNT con diferentes abstracciones de un mismo sistema. Dependiendo del tipo de control que se desee realizar se puede dar diversas visiones del mismo entorno.

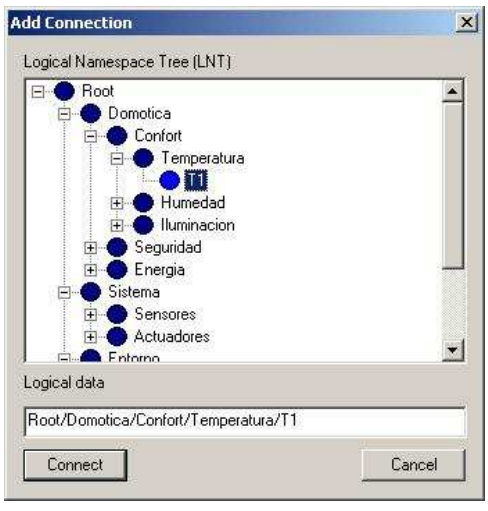


Figura 82. Asignación de un dato lógico a un elemento.

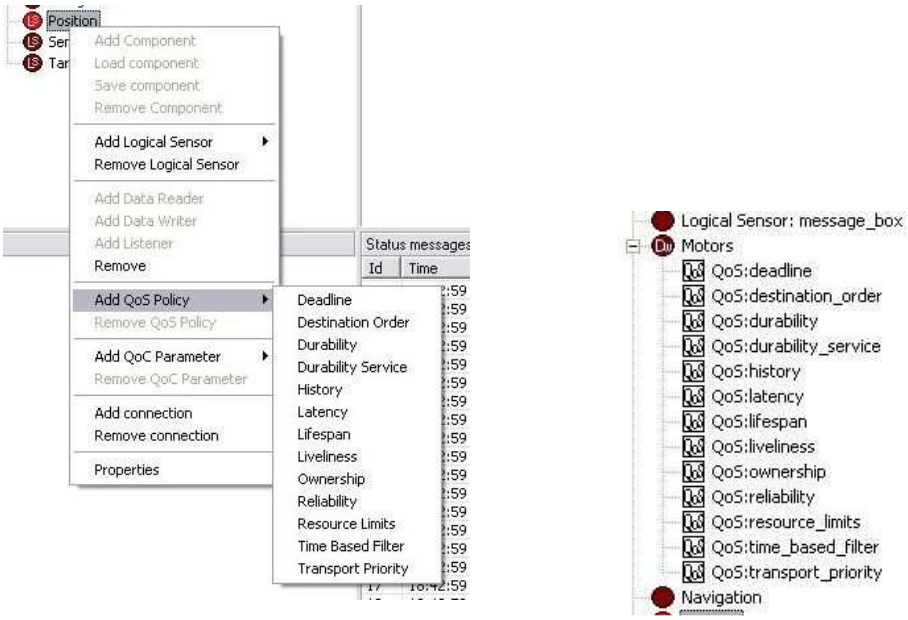
Cuando se selecciona un elemento de comunicación, *Publisher* o *Subscriber*, o un sensor lógico de comunicación, *DataWriter*, *DataReader* o *Listener*, se selecciona a qué dato lógico se conectará para enviar o recibir la información (Figura 82).

A partir del instante en que se han conectado dos elementos, el nodo de control pasa a revisar la coherencia entre las políticas de calidad de servicio de los elementos conectados. Posteriormente se revisará la coherencia de las políticas de calidad de servicio en todo el componente de control.

4.2.4 Gestión de la calidad de servicio

4.2.4.1 Asociación

La calidad de servicio se asocia a cualquier elemento de la arquitectura FSACtrl, aunque principalmente a los elementos de comunicaciones, tanto los comunes al nodo de control (*Publisher* y *Subscriber*) como los elementos de comunicaciones específicos de cada componente de control (*Listener*, *DataReader* y *DataWriter*). Para cada elemento de comunicaciones es posible asignar una calidad de servicio de cada tipo (Figura 83).



(a) **(b)**

Figura 83. Asignación de políticas de QoS (a) y políticas asignadas (b).

Cuando se asigna una política, no se asignan valores de configuración por defecto, ya que éstos podrían entrar en conflicto con las políticas ya existentes, como se verá en el siguiente apartado de configuración, la relación completa de las incompatibilidades pueden consultarse en [OMG, 2007].

4.2.4.2 Configuración

Una vez se ha insertado la política de QoS se pasa a configurar sus parámetros. Para ello, cada política dispone de la ventana propia de configuración (Figura 84) en la que poder cambiar los parámetros específicos.

Figura 84. Configuración de política de QoS (caso concreto: *DurabilityService*).

Una vez se ha configurado la política, se pasa a evaluar el impacto de la nueva configuración con el resto de políticas. La evaluación del impacto es progresiva, ya que implica comprobar las incompatibilidades internas en el elemento y la posible incompatibilidad con el resto de elementos. La comprobación de coherencia se realiza progresivamente por fases en diversos ámbitos (Figura 85).

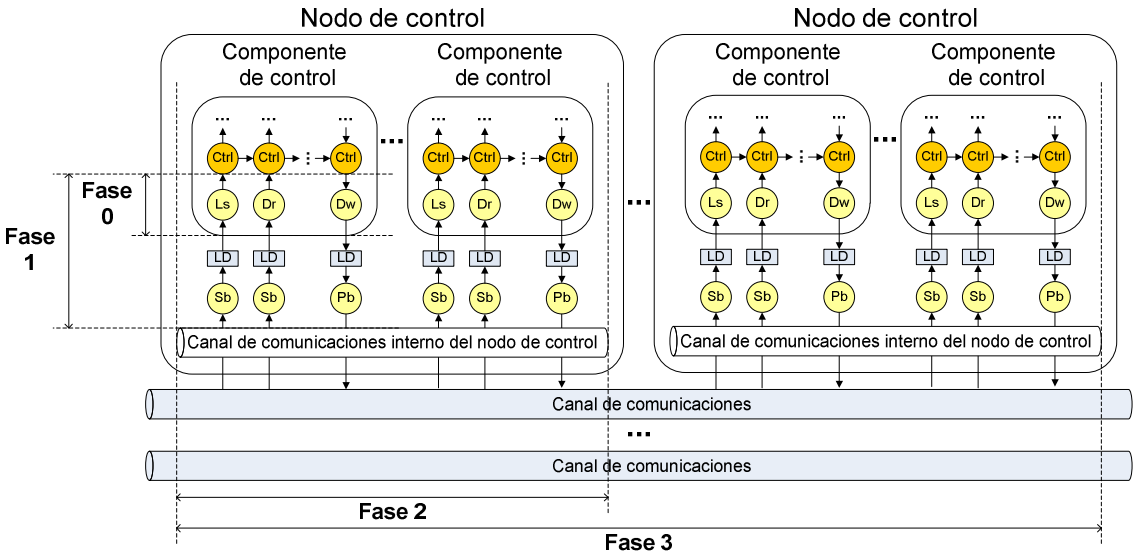


Figura 85. Fases en la comprobación de la integridad.

La fase inicial (fase 0) de comprobación de la coherencia en la QoS es la interna del propio elemento al que se le ha cambiado alguna de las políticas. Esta comprobación es fundamental y consiste en aplicar las fórmulas de compatibilidades, como por ejemplo que el tiempo de *Deadline* (plazo máximo de espera para el siguiente mensaje) sea mayor que un tiempo de *Lifespan* (plazo mínimo que debe transcurrir entre mensajes).

La fase 1 comprueba la coherencia de las políticas de QoS dentro del componente de control al que pertenezca el elemento cuya configuración de QoS se ha variado. Es especialmente importante que los sensores lógicos de comunicaciones puedan proporcionar los requisitos de los sensores lógicos de control en la ruta de mensajes.

La fase 2 se realiza en el ámbito de las comunicaciones entre el sensor lógico de comunicaciones y el adaptador de comunicaciones. En esta fase, si los *Publisher* o el *Subscriber* no pueden proporcionar el parámetro solicitado por el sensor lógico de comunicaciones se pasa a la negociación de los mismos, pero dentro del ámbito del componente de control, pudiéndose variar los parámetros del *Publisher* o el *Subscriber* para que sea compatible. La variación de la configuración de un *Publisher* o un *Subscriber* da lugar a la comprobación de la posibilidad de mantener los parámetros por parte del nodo de control.

Finalmente la última fase (fase 4) consiste en comprobar la viabilidad de todo el sistema ante el impacto que supone una alteración de un requisito de QoS en uno de sus elementos. Para ello se comprueba el cumplimiento de los requisitos en todos los nodos de control a los que estén conectados los *Publisher* y *Subscriber* del componente de control al que pertenece el elemento cuya QoS se ha variado. Caso de no cumplirse se generarán los correspondientes eventos de incumplimiento de QoS.

4.2.5 Gestión de la calidad de control

La calidad de control se asocia únicamente a los sensores lógicos de control. Aunque depende de cada tipo de parámetro de QoC, la mayor parte de ellos se basan en la evaluación del error en función de un valor de referencia.

Figura 86. Configuración de política de QoC (caso concreto: IAE).

La procedencia del valor con el que calcular el error se hace a partir de los mensajes de uno de los sensores lógicos de entrada, para ello se debe seleccionar la procedencia ("Sensor value" en la Figura 86). Además, se puede asignar un sensor al que se le envía el resultado del cálculo del parámetro de QoC. Una vez asignados los sensores, se pasa a asignar los parámetros correspondientes de la QoC.

A diferencia de las políticas de QoS, los parámetros de QoC no requieren de una comprobación adicional del impacto en la arquitectura. Esto se debe a que los parámetros de QoC se emplean para comprobar la eficiencia en el control del sensor lógico de control, por lo que fuera de este ámbito no afecta a los demás elementos.

4.2.6 Gestión de los eventos del sistema

Como se explicó en el apartado anterior, la mayor parte de acciones que se pueden realizar en un componente de control generan una secuencia de eventos dentro del sistema. El caso más concreto en el que se debe ofrecer una gestión de los eventos es el del ciclo integral de la calidad.

En la arquitectura FSACtrl, los eventos se consideran atómicos y están agrupados en "*Condition*", de manera que todo evento está asociado a una condición. A continuación se detalla el proceso de la gestión de los eventos a través de condiciones.

4.2.6.1 Inserción de condiciones

Las condiciones se pueden asociar a cualquier elemento del sistema, lo único que varía son los eventos que se pueden asociar, ya que estos eventos pueden ser específicos del tipo de elemento. Por ejemplo, los eventos relativos a la calidad de servicio se asocian a condiciones pertenecientes a los elementos de comunicaciones, tanto adaptadores como sensores lógicos de comunicaciones.

Figura 87. Asociación de condiciones y eventos a un elemento de la arquitectura.

Un elemento puede tener diversas condiciones y cada condición diversos eventos (Figura 87), por lo que se puede realizar una gran variedad de combinaciones de gestión de eventos por elemento.

4.2.6.2 Inserción y configuración de eventos

4.2.6.2.1 Eventos relativos a elementos

Las circunstancias a lo largo del funcionamiento del sistema pueden generar una serie de eventos, estos eventos son de especial interés, ya que son los que determinan la ejecución de acciones importantes para el funcionamiento, como iniciar o detener un sensor lógico.

Como ya se expuso anteriormente, los eventos que se pueden asociar a un elemento dependen del tipo (Figura 88). Los eventos comunes que se pueden detectar son el inicio o detención de la actividad del elemento, la llegada de un mensaje o el final del procesamiento del mismo, para el caso de los sensores lógicos de control. En el caso de los elementos de comunicaciones, el evento detectable es la asociación del elemento a un dato lógico inexistente. En el caso de los componentes de control la variedad de eventos detectables se corresponden con los aspectos estructurales asociados a los elementos que forman el componente de control, como la adición o eliminación de un sensor lógico, la creación o eliminación de conexiones entre sensores lógicos o los cambios en políticas de QoS o parámetros de QoC. En este último caso cabe destacar que tanto la inserción como la eliminación siempre se consideran un cambio, ya que la medición de la viabilidad a consecuencia de estas acciones podrá forzar la generación de un evento.

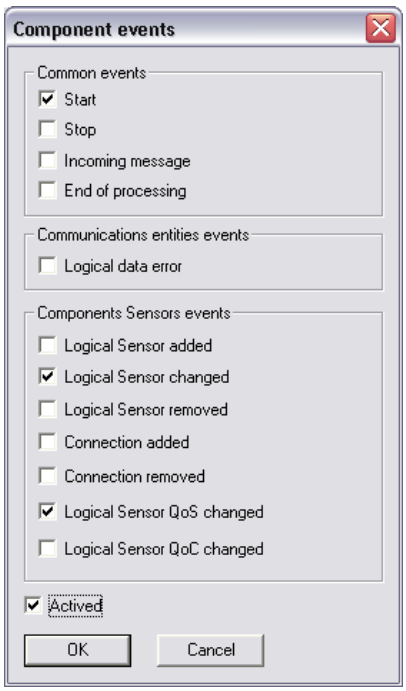
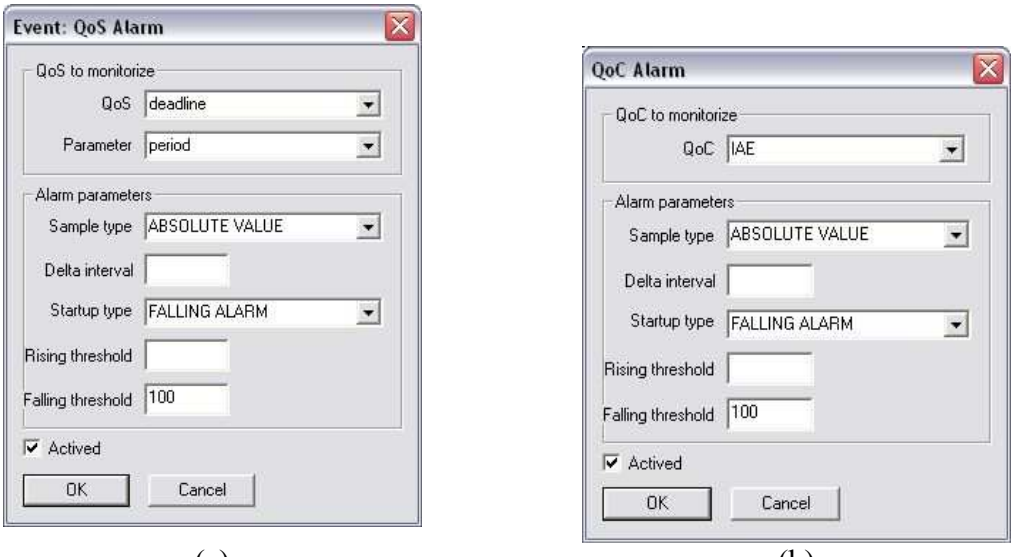


Figura 88. Ejemplo de configuración de evento de elementos.

4.2.6.2.2 Alarmas de QoS y QoC

La gestión de las alarmas de la calidad de servicio y de control es muy similar, ya que en ambos casos se trata de monitorizar un parámetro. Todos los elementos pueden gestionar los eventos de calidad de servicio, ya que todos ellos tienen colas de mensajes. Sin embargo, la calidad de control sólo puede producir eventos en los sensores lógicos de control. En el caso de la calidad de servicio se trata de alguno de los parámetros numéricos de las políticas de la QoS, mientras que en el caso de la QoC se trata del valor de la calidad en sí misma (Figura 89).



(a) (b)

Figura 89. Ejemplo de configuración de alarmas de calidad de servicio (a) y calidad de control (b).

Los valores necesarios para especificar una alarma son los expuestos en el apartado correspondiente del capítulo anterior. Por medio de los valores máximos y mínimos de las alarmas asociadas a políticas de QoS se configuran los márgenes en los que la política de QoS deben negociar en las fases de evaluación del impacto en las variaciones de la configuración en la gestión de la calidad de servicio (apartado 4.2.4.2).

4.2.6.2.3 Filtros de mensajes

Los eventos asociados a filtros pueden asociarse a cualquier elemento, ya que todos los elementos de la arquitectura FSACtrl disponen de colas de mensajes. Los parámetros necesarios para delimitar las secuencias a detectar y las máscaras a aplicar son los expuestos en el apartado 3.2.7.1.1.3 del capítulo anterior: desplazamiento de la máscara en el interior del mensaje, patrón de comparación y máscaras de coincidencia y no coincidencia (Figura 90).

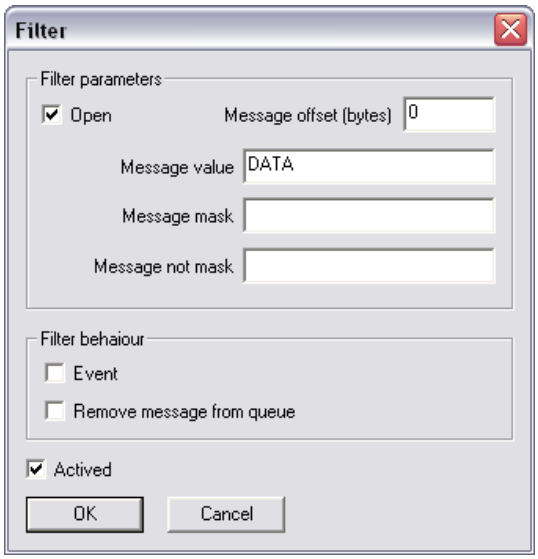


Figura 90. Ejemplo configuración de filtro.

Se debe tener en cuenta que la inserción de un filtro implica un gasto de proceso importante, por lo que es recomendable realizarla en el origen del mensaje, o lo más pronto que éstos llegan al sistema de control, es decir en los adaptadores. Para optimizar el funcionamiento de los filtros, pueden realizar acciones directamente en la cola de mensajes como es la eliminación del mensaje.

4.2.6.3 Inserción y configuración de acciones

Una vez que se han definido los eventos a monitorizar se debe decidir qué acciones realizar cuando estos eventos sucedan. En la arquitectura FSACtrl son las condiciones las que llevan asociadas acciones (Figura 91) que se realizan cuando la condición sea cierta.

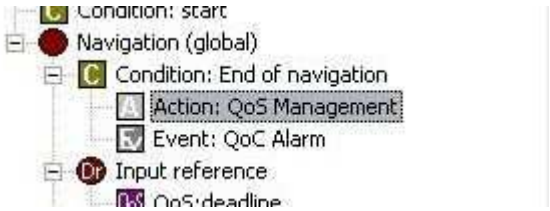


Figura 91. Asociación de acciones a condiciones.

Algunas de las acciones sólo se pueden realizar en cierto tipo de elementos, por ejemplo la variación de una QoC sólo puede aplicarse en sensores lógicos de control, ya que son éstos elementos los únicos que tienen asociados parámetros de QoC.

4.2.6.3.1 Acciones de los elementos y de sus conexiones

Las acciones que se pueden realizar sobre los elementos son muy diversas. Son muchas las características de un elemento que pueden variar, a las que hay que añadir la variación de la estructura del componente de control. El ámbito de las acciones de elementos depende del elemento que la realiza: un componente de control puede realizar acciones sobre los elementos que contiene, pero no componentes externos que no pertenecen a su ámbito. La Figura 92 muestra la ventana de asociación de acciones a un elemento.

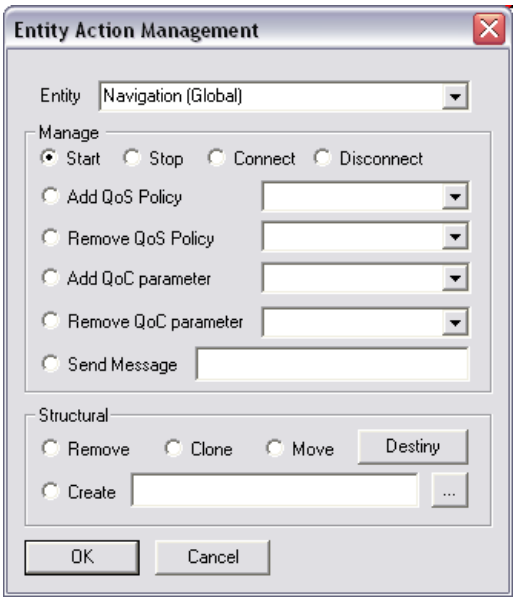


Figura 92. Acciones asociadas a las características de un sensor lógico.

Por defecto el elemento sobre el que se realiza la acción es el que contiene la condición, pero puede cambiarse dependiendo del ámbito al que se tenga acceso. Sobre el elemento seleccionado se puede seleccionar entre conectarlo, desconectarlo, detenerlo o ponerlo en marcha, añadir políticas de QoS o parámetros de QoC.

Otra de las acciones es el envío de un mensaje al sensor lógico seleccionado, en este caso el responsable del envío es el componente de control, ya que es quien tienen acceso a todos los elementos que contiene.

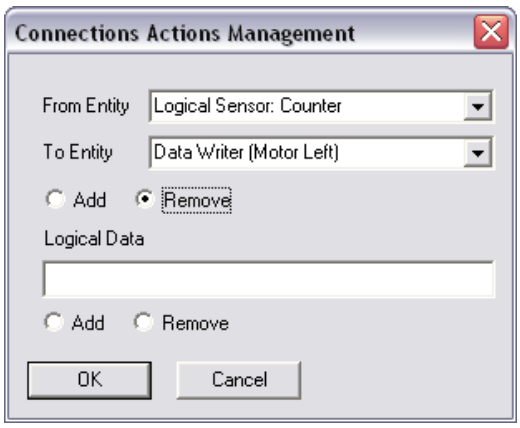


Figura 93. Acción de inserción o eliminación de conexiones entre sensores lógicos.

Además de la gestión de los elementos, también e dispone de las acciones de gestión de conexiones (Figura 93). La gestión de las conexiones permite conectar y desconectar sensores lógicos pertenecientes a un componente de control o bien elementos de comunicaciones entre ellos. Para este último caso, en el caso de crear una conexión se debe especificar el dato lógico a emplear. También es posible crear o eliminar datos lógicos sin necesidad de crear conexiones entre elementos.

4.2.6.3.2 Acciones de las políticas de QoS y de los parámetros de QoC

La acción de variación de una política de QoS actúa sobre las políticas de QoS de los sensores lógicos que pertenecen al mismo componente de control. Las variaciones de una política de QoS se realizan sobre un parámetro en particular (Figura 94).

Figura 94. Acción de variación de una política de QoS.

En el caso de que el parámetro sea numérico, el valor a variar puede ser incremental, sumando o restando una cantidad al valor existente, o bien imponer un valor concreto. El primer caso se emplea para ajustar los parámetros de QoS a las condiciones, el segundo en el caso de que la acción consista en establecer una determinada configuración de QoS. En el caso de que el parámetro consista en la selección de una lista de configuración, esta opción se activará desactivándose la del valor numérico.

Las acciones relacionadas con la QoC son similares a las de la QoS, aunque en este caso únicamente se trata de configurar el parámetro de QoC (Figura 95).

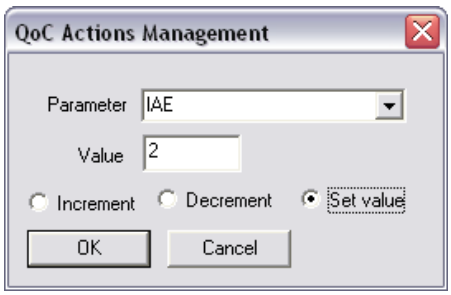


Figura 95. Acción de variación de una política de QoC.

Al igual que sucede con la QoS, la acción puede variar incrementalmente sumando o restando una cantidad al valor actual, o bien establecer un valor determinado. De la primera forma se permite adaptar las restricciones de la QoC, mientras que de la segunda manera se configura el parámetro de QoC.

4.3 Ciclo integral de la calidad

4.3.1 Fundamentos

La optimización del componente de control se realiza a partir de las políticas de calidad de servicio tanto las locales a los sensores lógicos como las comunes a los sensores componentes de control. Por tanto, las políticas de calidad de servicio permiten ajustar la configuración de los parámetros de los sensores lógicos. El ajuste puede ser tanto interno en el propio sensor lógico como externo en las conexiones, tal como se mostraba en la Figura 31. En el caso de ser un ajuste interno, los parámetros se utilizan para ajustar el funcionamiento de la cola de mensajes o del hilo de control.

El uso principal de la calidad de servicio es la caracterización de un sistema para la optimización del mismo. En la Figura 96 se muestra cómo la calidad de servicio genera un ciclo entre el cliente y el proveedor [González, 2007].

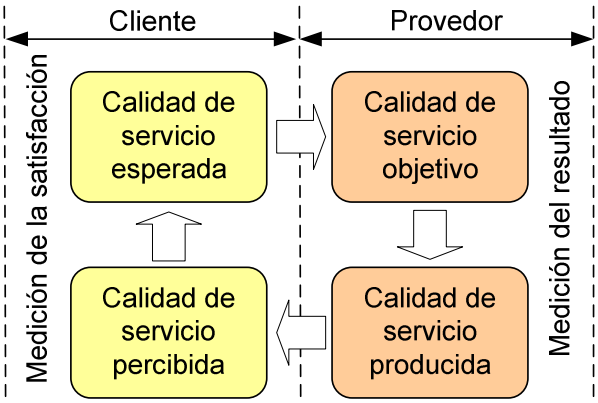


Figura 96. Ciclo de la calidad de servicio.

El ciclo de la calidad de servicio mostrado en la Figura 96 se basa en mediciones subjetivas. Obtener los parámetros de todos los elementos, para poder obtener una percepción cuantitativa de la calidad de servicio, permite comparar cada una de las fases del ciclo de la calidad de servicio, lo que a su vez permite automatizar el proceso completo de monitorización de los parámetros para obtener una estimación del funcionamiento de las comunicaciones y del control.

4.3.2 Ciclo completo

La arquitectura FSACtrl permite emplear los parámetros de las colas de mensajes de los elementos para componer índices de calidad de servicio más complejos a partir de los cuales poder ajustar el control del sistema. Asimismo, también se puede incluir la calidad de control dentro del mismo proceso, permitiendo hacer un seguimiento integral de la calidad de los elementos del sistema. En la Figura 97 se muestra el ciclo completo de gestión integral de la calidad de servicio y la calidad de control, donde se combinan los parámetros de calidad de servicio y los parámetros de calidad de control.

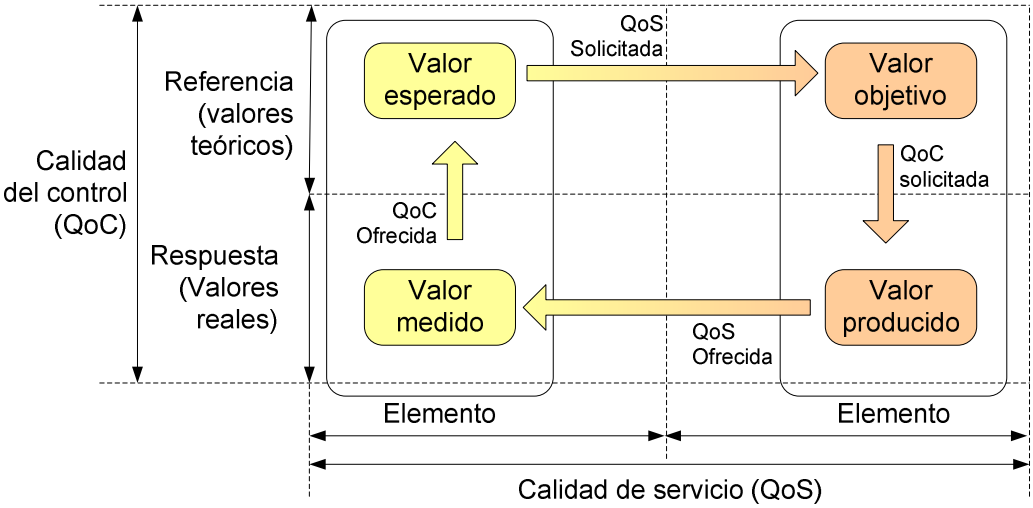


Figura 97. Ubicación de la QoS y la QoC en el ciclo integral de la calidad.

El cumplimiento de la QoS se obtiene de la comparación entre la QoS solicitada y la QoS ofrecida por parte de los elementos que intervienen en la comunicación. El cumplimiento de la QoC se obtiene comparando los valores internos teóricos y reales para obtener los índices de calidad de control correspondientes a los sensores lógicos de control.

4.3.3 Eventos del ciclo integral

La utilidad del ciclo integral de gestión de la calidad es muy amplia (monitorizar la QoS y QoC de los componentes de control, evaluar el impacto de cambios en el entorno, emplearlo en la negociación de los parámetros de QoS). Para que se inicie el ciclo deben darse, como mínimo, uno de los eventos que se muestran en la Figura 98.

Los eventos pueden ser externos o internos al sistema. Los eventos externos que inician el ciclo integral son dos: la acción del usuario o los cambios del entorno, eventos 1 y 4 de la Figura 98.

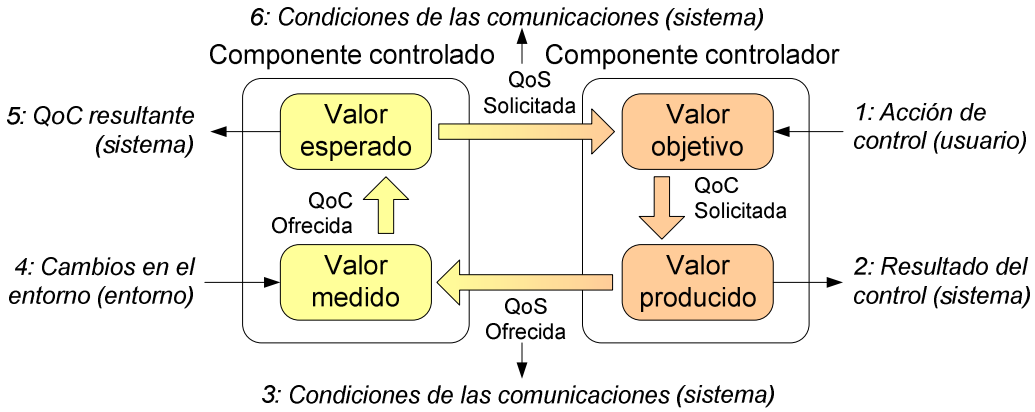


Figura 98. Eventos del ciclo integral de la calidad.

Los eventos restantes se consideran internos y los produce el sistema como consecuencia de los errores en la acción del control, localizables por los parámetros de QoC o el no cumplimiento de las políticas de calidad de servicio. En el caso de la QoC, eventos 2 y 5 de la Figura 98, son eventos que se producen en los sensores lógicos de control. Por lo que respecta a la QoS, eventos 3 y 6 de la Figura 98, se producen en cualquiera de los elementos del sistema.

4.3.4 Soporte al ciclo integral en los componentes de control

Para implementar el ciclo integral de la calidad, se emplean algunas de las funciones de los componentes de control (Figura 99). Los sensores lógicos de control son los responsables de proporcionar a los parámetros de QoC los valores medido y objetivo.

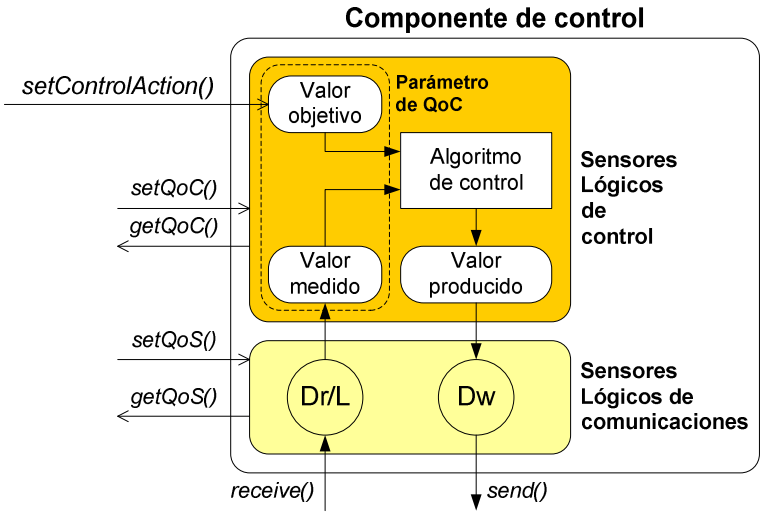


Figura 99. Funciones de los agentes de control para el soporte del ciclo integral de la calidad.

A continuación se muestra las secuencias de eventos así como los elementos y las funciones relacionadas en la gestión de cada una de las secuencias en función del escenario concreto.

4.3.4.1 Cambio en la acción de control

La acción de control es un evento externo al componente de control y se produce cuando varía el valor objetivo. Por ejemplo, cuando el usuario cambia el objetivo de control, la secuencia de eventos es: 1, 2, 4, 2.

Cuando se varía el objetivo del control, el valor objetivo cambia, por lo que la QoC del sensor lógico de control al que se le ha variado el objetivo variará, ya que se producirá un error entre el valor objetivo y el valor medido, lo que pondrá en marcha el algoritmo de control y se generará la correspondiente acción de control.

A consecuencia de la acción de control, se puede enviar un mensaje a los actuadores por lo que los valores de los sensores pueden variar. En el caso de que varíen los valores de los sensores de forma significativa, se enviará al sensor lógico de control el nuevo valor de forma que la QoC puede variar. En el caso de que la variación produzca un incumplimiento de la QoC, el sensor lógico de control generará una nueva acción de control. Mientras no se cumpla la QoC los eventos mantendrán activo el bucle de control, que finalizará cuando la QoC vuelva a cumplirse.

4.3.4.2 Cambio de condiciones en el entorno

El cambio de las condiciones del entorno es un evento externo al sistema y al sensor lógico de control. Lo habitual es que el cambio en las condiciones se produce por el cambio en la medición en un sensor. La variación en la medición del sensor puede provenir del propio cambio del entorno, por ejemplo si un objeto se acerca a un robot móvil, o bien de la propia acción de control, por ejemplo si se ordena mover el motor de un robot móvil y a consecuencia del movimiento el robot se acerca a un obstáculo. La secuencia de eventos en el cambio de las condiciones del entorno es: 4, 2, 4.

Cuando se da un cambio en las condiciones, el valor medido es enviado al sensor lógico de control, al ser distinto al valor objetivo se lanza el algoritmo de control, lo que produce un cambio en el valor escrito en el actuador y se entra en el bucle de control ya descrito en el apartado anterior.

4.3.4.3 Cambio en las condiciones de comunicaciones

El cambio en las condiciones en las comunicaciones es un evento externo al sensor lógico de control, y consecuentemente al componente de control. Este cambio de condiciones puede darse en ambos sentidos de la comunicación: QoS solicitada y QoS ofrecida. La secuencia de eventos en el cambio de las condiciones de comunicaciones es: 3, (5), 6, (2), 3.

Cuando la QoS varía, en alguno de los dos sentidos, el cálculo de la acción de control puede verse afectado, ya que un incumplimiento en una política de QoS puede afectar al cálculo de la acción de control y que el resultado no tenga la suficiente calidad para ser efectivo. En este caso, cuando se dan los eventos de QoS, se debe replantear el cálculo de la acción de control, ya sea variando la QoS en el sentido contrario, bien el valor objetivo o bien la QoC para el valor calculado.

4.3.4.4 Cambio de la estructura del sistema

El cambio en la estructura es la variación interna de los componentes de control, la variación en el nodo de control o bien la aparición de un nuevo elemento en el sistema, lo que puede dar lugar al replanteamiento de la viabilidad del sistema a consecuencia del impacto que produce el nuevo elemento. En este caso la secuencia es: 2, 3, 5, 6, 2.

Debido al cambio, es necesario comprobar el cumplimiento de los parámetros de QoC y la QoS. En el caso de que se pierda calidad, los nuevos valores deben estar dentro de los márgenes de trabajo especificado en los eventos, caso de ser necesario.

4.4 Implementación de un sistema de agentes basado en FSACtrl

4.4.1 Componentes

Los sistemas multi-agente son una de las tecnologías más adecuadas para la implementación de sistemas distribuidos inteligentes. El soporte que el sistema distribuido proporciona a los agentes influye en la funcionalidad que los agentes pueden desarrollar en el sistema. En la arquitectura FSACtrl se dispone de diversas semejanzas entre elementos, lo cual permite implementar un sistema de agentes. A partir de los parámetros de QoS y de QoC se determina la calidad en el funcionamiento del agente en el nodo de control, término que actualmente se conoce como Calidad del Agente (QoA). En cuanto a operatividad, en la arquitectura FSACtrl se puede embeber parte del código de control del sistema distribuido en el middleware, lo que permite realizar las operaciones de movimiento de datos y agentes de una forma suave entre los nodos de control.

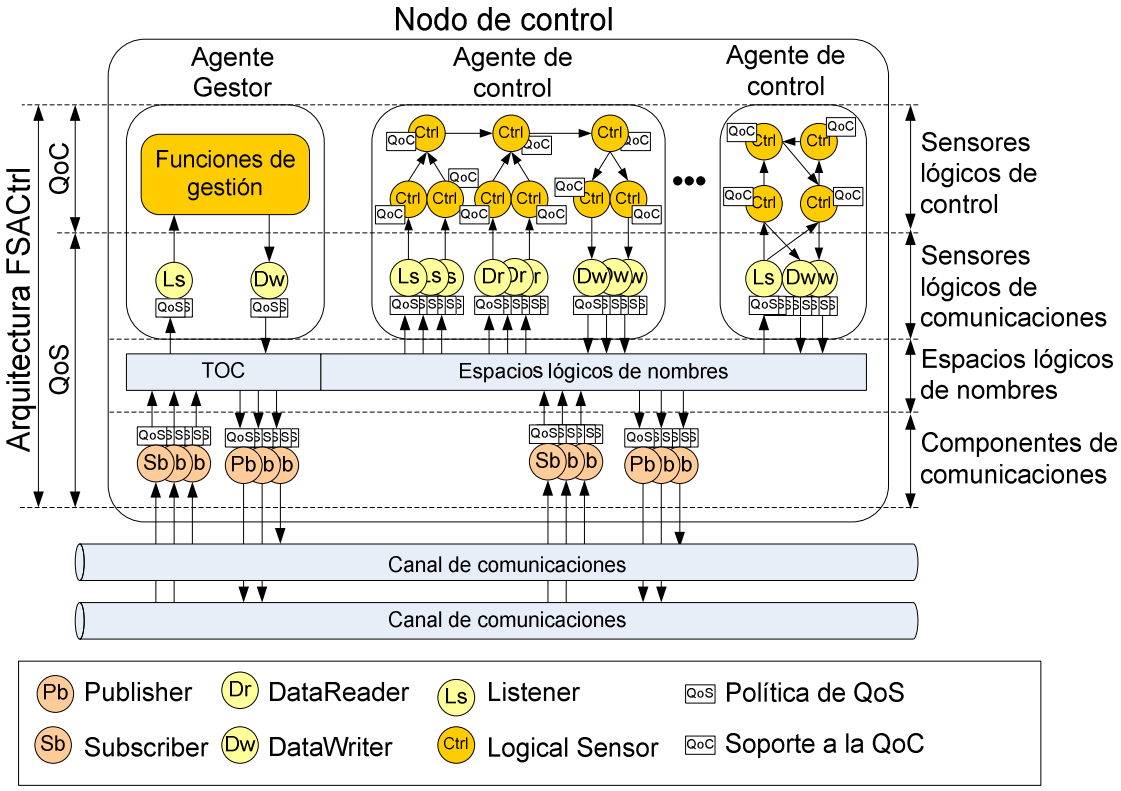


Figura 100. Organización de los elementos FSACtrl como sistemas de agentes.

Para adaptar FSACtrl a un sistema de agentes de control, son necesarios diversos componentes que se implementan a partir de la organización de los elementos FSACtrl (Figura 100).

Los agentes de control residen en un nodo de control responsable de suministrar la infraestructura de computación necesaria para las comunicaciones y el procesamiento.

El nodo de control lo forma el hardware y el software necesario para que los agentes y los componentes de la conexión, todos ellos elementos de la arquitectura FSACtrl, puedan realizar todas sus funcionalidades.

Cada nodo de control tiene un Agente Gestor que da soporte a las operaciones del sistema multi-agente. Tramita las solicitudes de los agentes de control, tanto del propio nodo de control como de otros nodos de control, así como la gestión del espacio lógico de nombres y la intermediación con las políticas de QoS y de QoC. El agente gestor funciona como un servicio al que se accede a través del nodo de control.

Los agentes de control son los agentes que reciben, procesan y envían los datos correspondientes, en FSACtrl se implementan a partir de los componentes de control ya que éstos son los que agrupan a los sensores lógicos de control para realizar las funciones más complejas. El uso de los componentes de control de la arquitectura FSACtrl tiene un valor añadido, ya que tienen una visión conjunta, tanto de la QoS como de la QoC, lo que define la QoA como la combinación de la QoS y la QoC. Teniendo en cuenta que la QoS y la QoC, de la arquitectura FSACtrl, están definidas en función de políticas que se basan en parámetros empíricos, es posible definir la calidad del funcionamiento del agente de forma empírica.

En FSACtrl los espacios de nombres lógicos tienen un papel importante al ser el medio por el que se ocultan los detalles de las comunicaciones, incluidos los detalles de la localización física de los agentes. Cada nodo de control contiene una jerarquía llamada *Table of Contents* (TOC), que describe los agentes y las funciones disponibles en el nodo de control al resto de nodos de control del sistema. La información que proporciona el TOC es la mínima que se requiere para conocer la composición y las características de los agentes en un nodo de control (Figura 101).

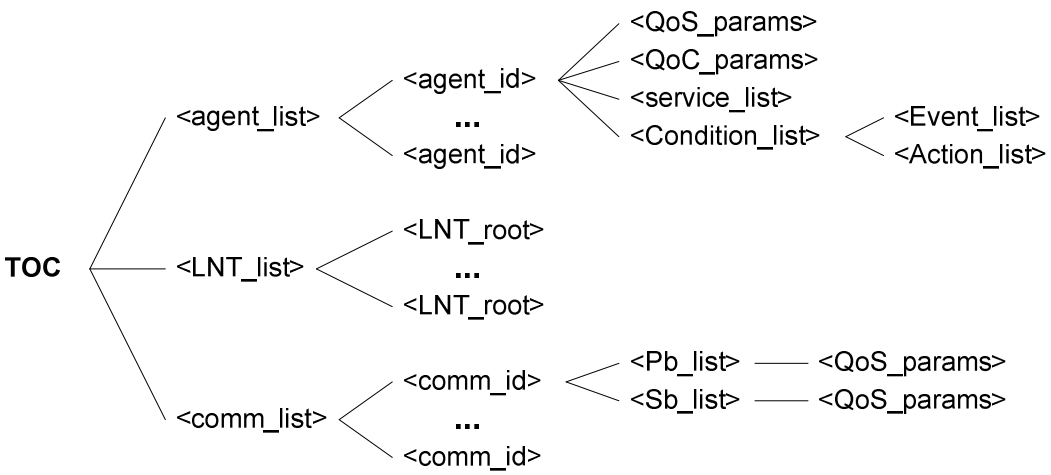


Figura 101. Estructura del TOC.

La tabla de contenidos proporciona información acerca de los agentes que se encuentran en el nodo de control, los árboles de espacios lógicos que emplean los agentes y los canales de comunicaciones que están disponibles. Para cada agente es posible obtener los valores de los parámetros de QoS y QoC, además se proporciona la lista de servicios disponible por agente. De cada LNT se proporciona el nodo raíz a partir del cual se obtiene la lista de datos lógicos con los que poder comunicarse. Finalmente, de cada canal de comunicaciones se obtiene la lista de *Publishers* y *Subscribers* con los que poder comunicarse, así como la lista de parámetros de QoS.

Los elementos de comunicaciones son los propuestos por el modelo DDS y usados en la arquitectura FSACtrl. Están repartidos entre los agentes de control y el nodo de control. Los *Publishers* y *Subscribers* son comunes a todos los agentes de control. Mientras que los *DataWriters*, los *DataReaders* y los *Listeners* son específicos de cada agente de control.

4.4.2 Operaciones

De entre los componentes de un sistema de agentes que se han expuesto anteriormente, las operaciones con el sistema se realizan por medio del agente gestor, quien intermedia entre los agentes externos y los usuarios con el sistema albergado en el nodo de control.

El agente gestor es el intermediario entre el resto del sistema, tanto agentes como usuarios, con los agentes albergados en el nodo de control. Para ello el agente gestor proporciona una serie de operaciones. Estas operaciones se sirven a través de los *Publishers* y *Subscribers* de cada canal de comunicaciones proporcionado por el nodo de control.

Las principales operaciones que los agentes pueden llevar a cabo en el sistema son similares a las de los sistemas multi-agente distribuidos [FIPA, 2000]. Cuando se consideran las agrupaciones de elementos como agentes, las operaciones que se pueden realizar se basan especialmente en el movimiento de los agentes entre los nodos de control. A continuación se describirán las operaciones necesarias que dan soporte al movimiento de los agentes:

- Fragmentación o fusión de sensores lógicos en los agentes de control.
- Inserción, modificación y eliminación de agentes de control en los nodos de control.
- Inserción, modificación y eliminación de datos lógicos del LNT.
- Consulta de la tabla de contenidos (descubrimiento del sistema)

Mover o clonar agentes, implica la inserción y/o eliminación de un agente en un nodo de control. Insertar un agente en un nodo de control consiste en la inserción de sus sensores lógicos de control y sus sensores de comunicaciones. Además, la propia composición de los agentes, basada en sensores lógicos de control, hace que un agente se pueda dividir en otros agentes. También se puede realizar el paso complementario y fusionar agentes a partir de la fusión de los elementos que lo forman en un mismo agente. A continuación se describe con detalle las operaciones, se comenzará con las más sencillas para finalizar con las más complejas.

Cabe destacar que las operaciones de fragmentación, fusión, clonado y movimiento de agentes pueden realizarse deteniendo los agentes implicados o sin detenerlos. En el primer caso, la operación no es complicada, ya que el agente gestor sólo tiene que crear o eliminar los elementos del agente de control. Sin embargo, una de las ventajas de la jerarquización de la arquitectura FSACtrl es que se pueden detener sólo aquellos sensores de control implicados en la operación o en una fase concreta de la operación.

4.4.2.1 Operaciones principales

4.4.2.1.1 Consulta a la tabla de contenidos

La consulta de la tabla de contenidos es la operación por medio de la que se proporciona la información del nodo de control al solicitante. Esta información se proporciona en

formato XML, para que sea autocontenido. A partir de esta información, FSACtrl en su orientación como arquitectura de agentes proporciona la capacidad de descubrimiento por la que un nuevo usuario o agente puede conocer el entorno en el que debe desenvolverse.

4.4.2.1.2 Inserción, modificación y eliminación de datos lógicos

La inserción de datos lógicos está justificada en diversos casos, generalmente cuando un agente se inserta en un nodo de control puede necesitar datos lógicos que no se encuentran en dicho nodo de control. La inserción de un dato lógico se realiza comprobando primero si existe, y en el caso de no existir, el agente gestor lo crea.

La modificación de un dato lógico es más compleja dado que el dato lógico puede estar en uso por diversos agentes. La modificación puede ser sintáctica, por ejemplo si se cambia la nomenclatura de un nodo lógico; o semántica, por ejemplo si se mueve un nodo lógico, con las implicaciones que implica el movimiento de todo el subárbol que dependa del nodo lógico.

La eliminación de un dato lógico tiene un mayor impacto, ya que implica que los agentes que lo estén usando no podrán enviar ni recibir mensajes por ese canal.

4.4.2.1.3 Inserción, modificación y eliminación de agentes de control

La inserción de un agente en un nodo de control puede ser solicitada por el usuario del sistema o por el propio agente que desea moverse o clonarse desde otro nodo de control. La información de la composición del agente de control, como los datos lógicos necesarios, los algoritmos de control a implementar o los parámetros de QoS y QoC, está contenida en la especificación del agente. Para insertar un agente de control en un nodo de control, el agente gestor debe comprobar que existen los datos lógicos necesarios para que el agente pueda comunicarse. Es responsabilidad del agente gestor realizar las correspondientes comprobaciones y creación de los datos lógicos, si procede.

La modificación de un agente de control implica cambiar algunas de sus características. Al igual que sucede con los componentes de control, en los que están basados, la modificación consiste en detener las comunicaciones del agente, realizar los cambios en la configuración y reiniciar las comunicaciones del agente.

La eliminación de un agente de control supone eliminar los sensores lógicos de control que lo componen. En cuanto a los datos lógicos, se eliminarán los datos lógicos del agente de control que no estén en uso por parte de otros agentes de control.

4.4.2.2 Operaciones avanzadas

4.4.2.2.1 Consideraciones iniciales

Las operaciones de fragmentación, fusión, clonado y movimiento de agentes afectan directamente a la composición interna de los agentes de control. Todas las operaciones tienen algunos aspectos en común que se tratarán a continuación. En todas las operaciones avanzadas hay una copia o movimiento de sensores lógicos, tanto de control como de comunicaciones, por ello se debe prestar especial atención a cómo se tratan las conexiones internas de los sensores.

En la Figura 102 se muestran los elementos de los agentes de control, y la nomenclatura que se va a emplear para describir las operaciones según su ubicación en el agente. Se debe prestar especial atención a las conexiones entrantes y salientes de todo sensor lógico de control que se deba copiar o mover. En el momento en que se copie o mueva un sensor lógico deberá poder recibir y enviar mensajes de la misma forma que el sensor lógico original. Para ello se deberá dotar de la infraestructura para enviar o recibir los mismos mensajes que llegarían o enviaría el sensor lógico de control en el caso de no disponer de los elementos adecuados en otro agente de control.

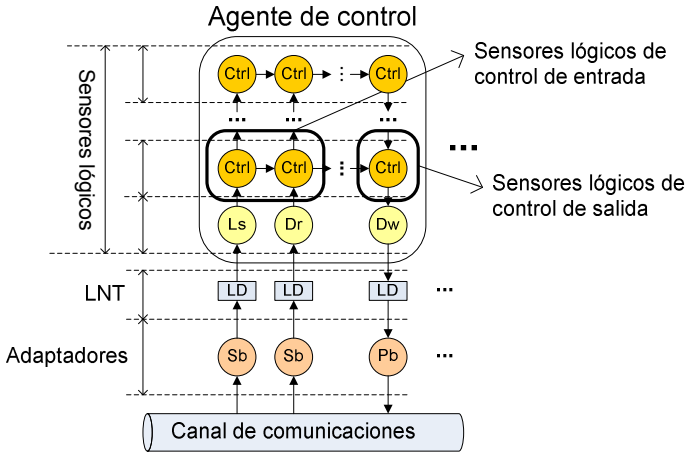


Figura 102. Estructura del agente de control.

El hecho de que entre el agente de control origen y el agente de control destino deba existir una comunicación orientada específicamente a los nuevos sensores lógicos que residen en el agente de control destino, hace que se deban crear diversos elementos auxiliares con la misión exclusiva de dar soporte al sensor lógico del agente de control destino.

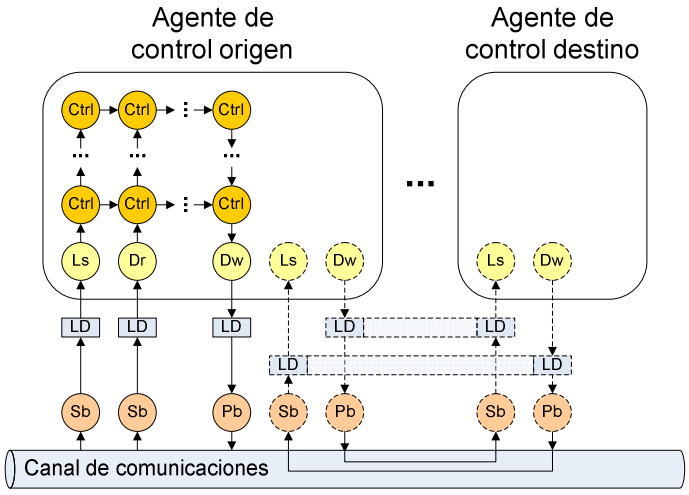


Figura 103. Elementos auxiliares a las operaciones de los agentes: bidireccional.

Si las conexiones internas del sensor lógico a mover son tanto entrantes como salientes a otros sensores lógicos de nivel 1 o superior (sensores lógicos de control), entonces es necesario un canal bidireccional simultáneo (Figura 103). Para disponer de un canal bidireccional hacen falta los correspondientes sensores lógicos de comunicaciones, un *Listener* y un *DataWriter* en cada agente de control.

En el nodo de control de cada agente de control hace falta crear un dato lógico para conectar el *Listener* del agente de control origen con el *DataWriter* del agente de control destino, e idénticamente en el sentido del agente de control destino al agente de control origen. Para dar servicio a los datos lógicos hacen falta los adaptadores correspondientes al canal de comunicaciones, un *Publisher* y un *Subscriber* por cada dato lógico.

El mantenimiento de los nuevos elementos en un mismo nodo de control en el caso de la fragmentación y la fusión, o en distintos nodos de control en el caso de la clonación y el movimiento tiene un coste que es posible optimizar (Figura 104).

La optimización consiste en transferir las conexiones desde los sensores lógicos de control de salida del agente de control origen. De esta forma, trasladando, o copiando si es el caso, primero los *DataWriter*, con los correspondientes datos lógicos y *Publisher* al agente de control destino, no es necesario retornar la conexión del sensor lógico de control al agente de control origen.

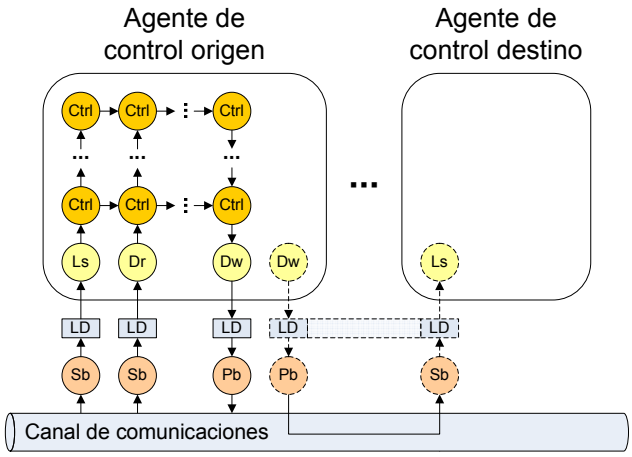


Figura 104. Elementos auxiliares a las operaciones de los agentes: mínimo.

Este proceso no se podrá llevar a cabo en el caso de que las conexiones entre los sensores lógicos de control tengan ciclos internos. A continuación se expondrán los detalles de cada una de las cuatro operaciones principales de los agentes.

4.4.2.2.2 Fragmentación y fusión

La fragmentación y la fusión de un agente de control son dos operaciones que no siempre están soportadas por los sistemas multi-agente. La fragmentación de agentes consiste en obtener dos agentes a partir de uno, de manera que entre ambos cumplan la misma funcionalidad que el agente de control original. La fusión de agentes consiste en la creación de un agente de control único a partir de dos agentes de control, de forma que la funcionalidad del nuevo agente es la misma que la funcionalidad conjunta de los dos anteriores.

Los motivos por los que fragmentar o fusionar agentes pueden ser muy diversos. Es posible repartir o concentrar el procesamiento entre diversos nodos de control y de esta forma adaptarse a una topología específica de sistema. También es posible optimizar los parámetros de calidad del agente en función de la composición interna del mismo. La iniciativa para la fragmentación o fusión de agentes puede provenir de distintas fuentes.

El agente gestor puede configurarse para que decida fragmentar o fusionar agentes en función de los parámetros de calidad. También es posible programar al propio agente para que se divida a iniciativa propia o solicite la fusión con otro agente. El usuario también puede realizar las operaciones para ensayar o conocer el comportamiento del sistema o de la composición de un agente específico.

En la arquitectura FSACtrl estas operaciones son sencillas de implementar. La propia composición de los agentes, basada en los sensores lógicos facilita las operaciones, ya que los sensores lógicos asumen el papel de módulos de procesamiento interno del agente. La fragmentación se realiza en el nodo de control por medio de la supervisión y dirección del agente gestor.

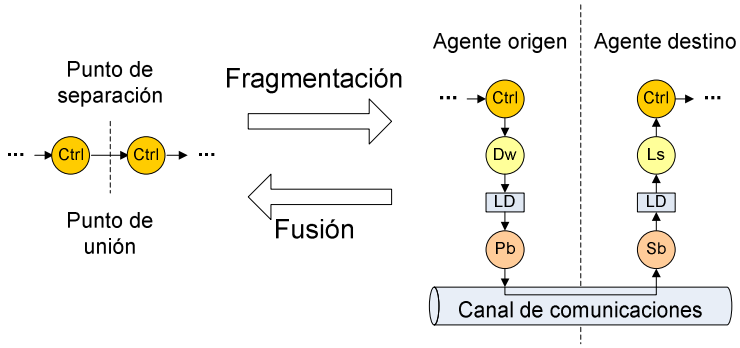


Figura 105. Fragmentación y fusión: elementos que intervienen en las operaciones.

En las operaciones de fusión, el punto de separación, por el que se decide dividir el agente, pasa a ser una secuencia de elementos de comunicaciones (Figura 105) que enviarán los mensajes que, anteriormente a la fragmentación, iban directamente entre sensores. Las operaciones empleadas en la fusión de dos agentes, son complementarias a las empleadas en la fragmentación. Consecuentemente la conexión que empleaba elementos de comunicaciones pasa a ser una conexión entre dos sensores lógicos.

La fragmentación de un agente comienza por la selección de los sensores lógicos que se separarán del agente origen para formar otro agente, considerado el agente destino. Una vez seleccionado los sensores lógicos a trasladar, el agente gestor debe crear la infraestructura necesaria para realizar la operación de fragmentación. La infraestructura parte de la creación de un agente de control vacío, que será el agente destino y albergará los sensores lógicos que se trasladarán desde el agente origen.

Una vez creado el agente de control duplicado, se crean dentro de él, los sensores lógicos que se deben trasladar. Los sensores lógicos están basados en los patrones de sensores cuya configuración se copia del sensor lógico origen. A continuación se crean los sensores lógicos de comunicaciones. Estos sensores lógicos de comunicaciones tienen su equivalente en el agente de control original. La creación por defecto del sensor lógico de comunicaciones de entrada es un *Listener* ya es necesaria una sincronización para realizar el cambio de mensajes.

Para comunicarse a través del canal interno, es necesario crear los adaptadores de comunicaciones necesarios, en este caso un *Publisher* para cada *DataWriter* y un *Subscriber* para cada *Listener*. Finalmente se crean los datos lógicos necesarios para la transferencia, en concreto uno por cada conexión entrante o saliente del sensor lógico de control que se debe mover. Con todos los elementos creados se tiene la infraestructura necesaria lista para comenzar a hacer el cambio de ruta de los mensajes (Figura 106)

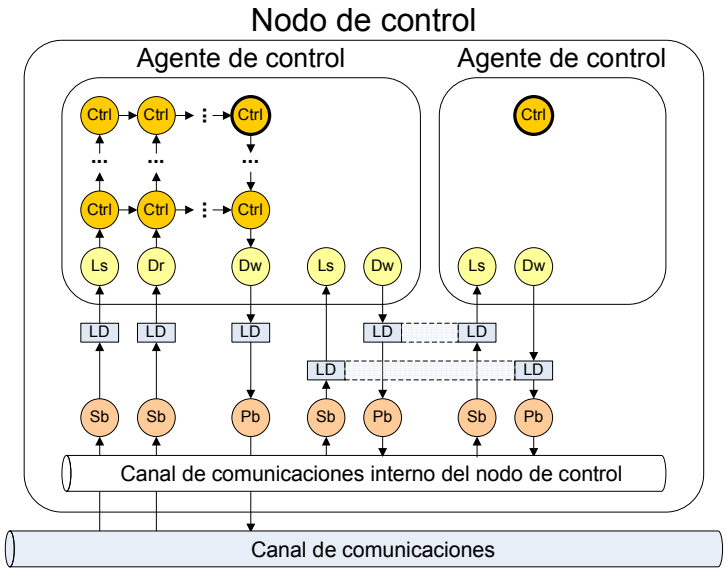


Figura 106. Fragmentación: infraestructura lista para la operación.

El cambio de procesamiento del agente de control original y el duplicado debe ser sincronizado para no perder mensajes. Durante los tiempos en que están cerrados los dos sensores lógicos, son las colas de mensajes de los sensores lógicos de comunicaciones y de los adaptadores las que albergarán los mensajes para ser enviados a los respectivos sensores lógicos de control. Estos tiempos son fácilmente modificables por medio de las políticas de QoS *Durability* y *Lifespan*.

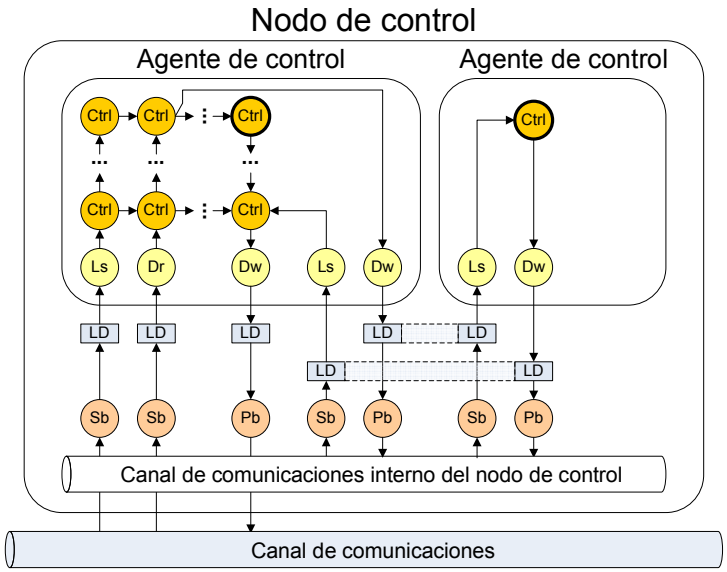


Figura 107. Fragmentación: cambio del flujo de mensajes.

El proceso de cambio (Figura 107) se realiza conectando los sensores lógicos de comunicaciones del agente de control original a los sensores lógicos que enviaban o recibían mensajes del sensor lógico de control duplicado. Asimismo, se conectan los sensores lógicos de comunicaciones con el sensor lógico creado en el agente de control receptor. Una vez creadas las conexiones, se van cerrando cada una de las conexiones entrantes del sensor lógico original en el agente de control origen, para abrirlas en el sensor lógico destino del agente de control receptor.

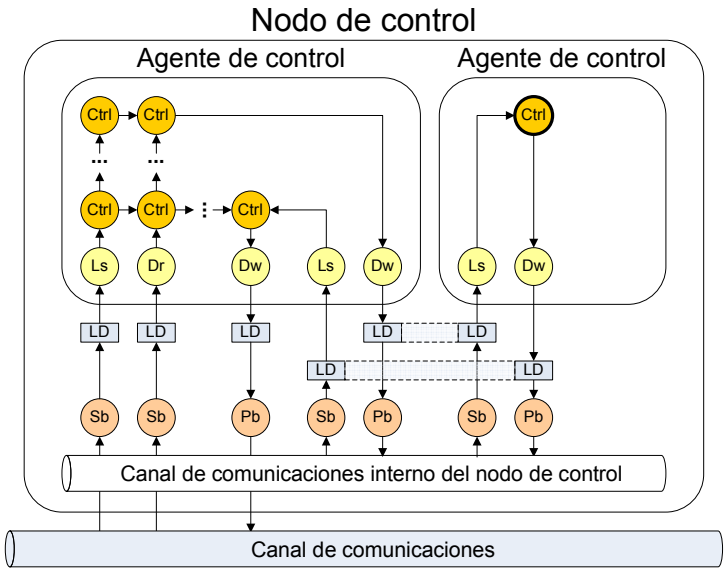


Figura 108. Fragmentación: final de la translación del sensor lógico de control.

Una vez cerradas las conexiones del sensor lógico de control original se procede a eliminarlo del agente de control origen (Figura 108). El proceso se repite por cada sensor lógico de control que se haya escogido para trasladarse, hasta que se hayan trasladado todos los sensores lógicos de control seleccionados.

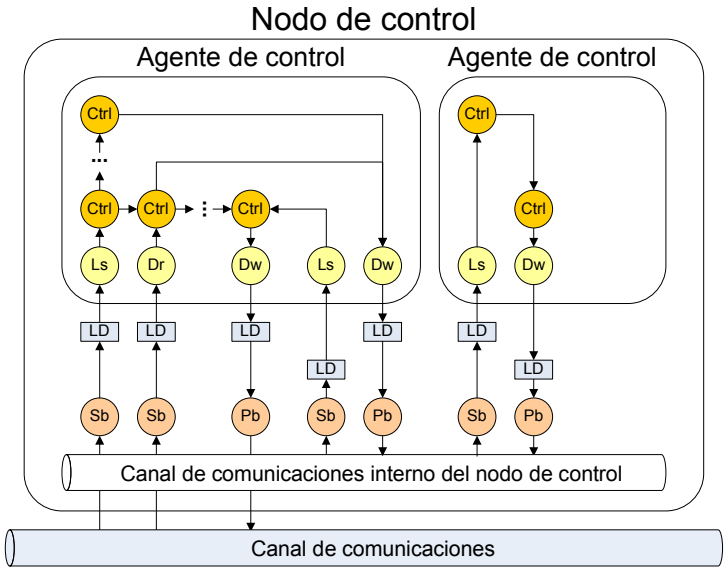


Figura 109. Fragmentación: situación final.

Una vez que se haya llegado al final del proceso de fragmentación, el nodo de control pasará a tener dos agentes de control, además de los nuevos datos lógicos y de los adaptadores de comunicaciones necesarios para el nuevo agente de control (Figura 109).

La fusión implica una operación muy similar, y en parte complementaria a la fragmentación descrita anteriormente. Aunque la desaparición del agente de control a fusionar implica tener que mantener los datos lógicos que esté empleando el agente de control en el agente de control receptor.

Si los dos agentes de control a fusionar no comparten ningún dato lógico, la fusión se reduce a crear los sensores lógicos del agente de control a fusionar en el agente de control receptor haciendo una imagen similar de la composición de uno en otro. Una vez no queden sensores lógicos, el agente gestor elimina el agente de control vacío. La transición de las comunicaciones se realiza de la misma forma que en la fragmentación, aunque de forma inversa.

Si los dos agentes de control comparten datos lógicos, la fusión implica la realización inversa de los pasos descritos en la fragmentación, con la salvedad de que si el dato lógico sólo lo empleaban esos dos agentes de control, pasa a desaparecer ya que la conexión entre los dos sensores lógicos se hace a nivel interno y no por medio de los sensores de comunicaciones.

4.4.2.2.3 Clonación y movimiento de agentes de control

La clonación y el movimiento de agentes son dos operaciones en las que intervienen como mínimo dos nodos de control. Esto hace que los agentes gestores deban sincronizarse. Clonar consiste en copiar el mismo agente con la misma funcionalidad, lo que implicará reproducir los mismos elementos del agente de control en el nodo de control receptor. El movimiento de agentes de control consiste en trasladar los elementos del agente de control del nodo de control origen al nodo de control destino.

Para la clonación, el primero de los pasos consiste en crear la infraestructura necesaria para el clon. Para la transición en la creación del clon es necesario que el nodo de control destino cree el agente de control destinatario. Además debe crear un dato común por medio del cual el agente de control origen enviará los mensajes al agente de control destino para que el clon pueda funcionar. Para dar servicio al dato lógico de soporte al clonado (Figura 110), serán necesarios algunos elementos auxiliares: dos elementos de comunicaciones en el agente de control origen: un *DataWriter* y un *Publisher* y otros dos elementos de comunicaciones en el agente de control destino: un *Subscriber* y un *Listener*.

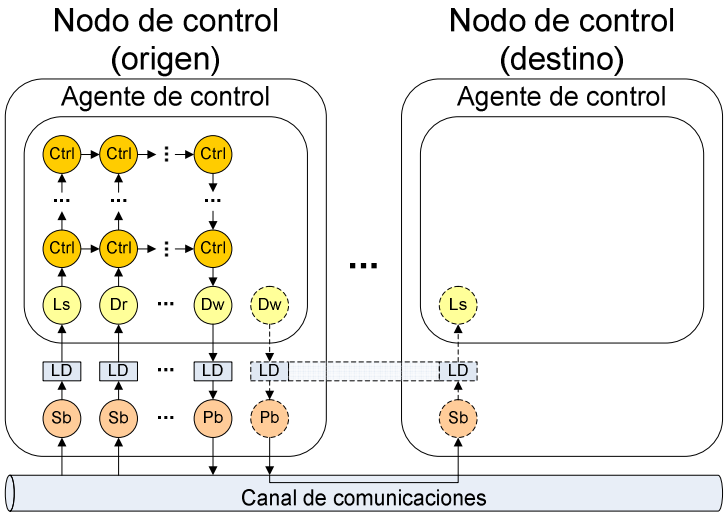


Figura 110. Clonación de agentes: creación de la infraestructura.

El primero de los pasos consiste en crear en el agente de control destino los *Publisher* para conectar los sensores lógicos de control (Figura 111) para mantener la comunicación con el clon desde el primer instante.

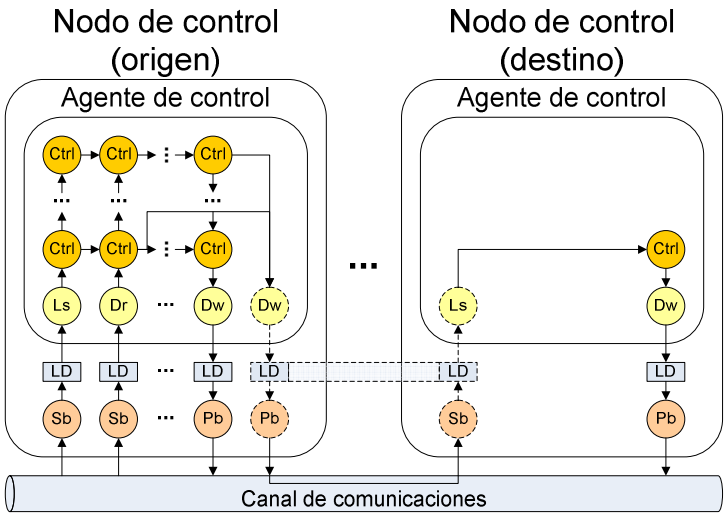


Figura 111. Clonación de agentes: creación de los sensores lógicos de control.

El hecho de que desde el primer instante el clon esté enviando mensajes producirá una redundancia de información. Este proceso se va repitiendo para cada uno de los sensores lógicos de control, cambiando las conexiones internas para que los sensores lógicos auxiliares transmitan los datos. A medida que los sensores lógicos de control lo requieren, se van creando los elementos de comunicaciones necesarios hasta que se crea el último elemento de comunicaciones (Figura 111).

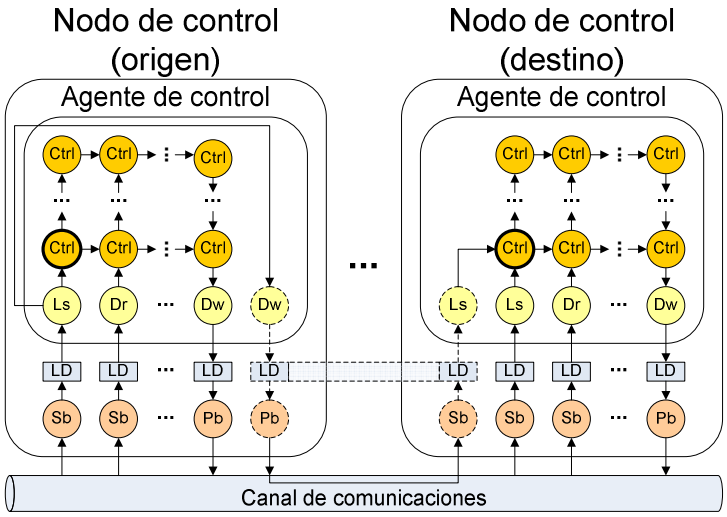


Figura 112. Clonación de agentes: finalización de la clonación.

Una vez se tienen todos los sensores lógicos y los adaptadores de comunicaciones en el nodo destino, se desconecta la conexión del *Listener* auxiliar eliminando los elementos que ya no son necesarios. Al finalizar el clon ya es funcional y puede operar de forma autónoma (Figura 112).

La infraestructura necesaria para el movimiento de agentes es la misma que en el caso de la clonación de agentes de control (Figura 110). Si es posible, el movimiento comienza por los sensores lógicos de comunicaciones de escritura (*DataWriter*) y sus correspondientes datos lógicos y *Publisher*, en el caso de que estos no estén ya en el nodo de control destino.

El movimiento de agentes sigue un proceso similar al clonado (Figura 113), con la salvedad de que el sensor lógico que, en este caso, se mueve debe eliminarse del nodo de control fuente.

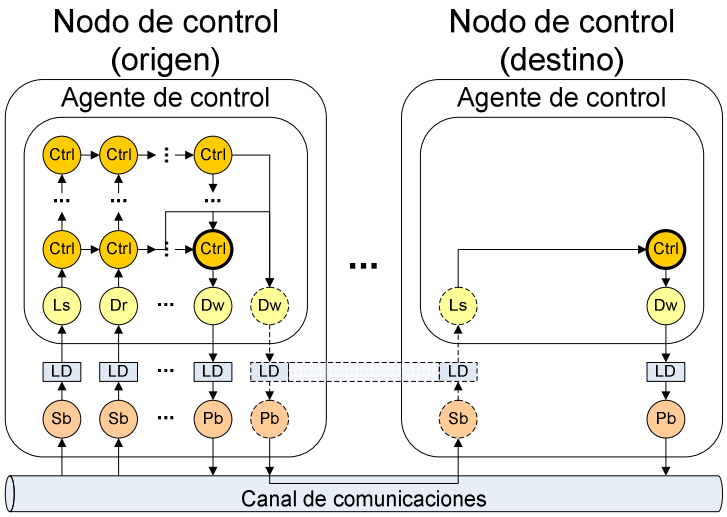


Figura 113. Movimiento de agentes: reconexión de las salidas de los mensajes.

En este caso, debido a la desaparición del sensor lógico en el nodo de control origen, se pierde su funcionalidad. Para evitar esta pérdida de funcionalidad, el nodo de control destino pone en marcha el sensor lógico de control copiado cuando el nodo de control origen detiene el sensor lógico de control. La ventaja en el uso de colas internas en los elementos hace que los elementos de comunicaciones auxiliares serán los que mantengan los mensajes que por cualquier motivo no sean procesados en el instante de transición.

Una vez el sensor lógico ha sido creado en el nodo de control destino, se pasa a eliminar del nodo de control origen. La eliminación comienza por la reconexión de las conexiones entrantes del sensor lógico que se mueve al *DataWriter* auxiliar, de una forma similar a como se hacía en la fragmentación de agentes de control (Figura 114).

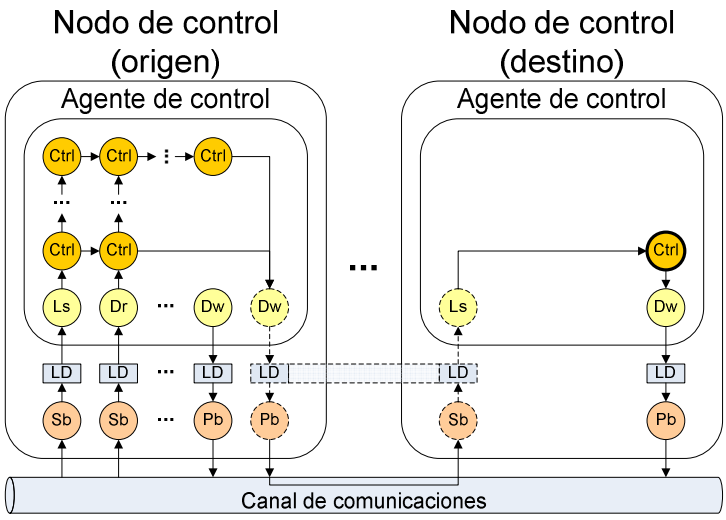


Figura 114. Movimiento de agentes: eliminación del sensor lógico original.

4.5 Conclusiones

En este capítulo se ha presentado el diseño de sistemas basados en la arquitectura FSACtrl. Inicialmente se ha mostrado la gestión de los elementos y las relaciones entre ellos. A continuación se ha expuesto cómo se gestionan los eventos en el sistema. A partir de los elementos y de los eventos, se ha enseñado cómo es posible implementar un sistema completo de agentes inteligentes distribuidos a partir de los elementos de la arquitectura. Como parte de la funcionalidad de la arquitectura FSACtrl se muestra la implementación de las operaciones necesarias para poder clonar y mover agentes en el sistema de forma gradual y adaptativa a las necesidades del sistema.

Cabe destacar cómo las estructuras organizativas de la arquitectura FSACtrl constituyen un medio muy adecuado para implementar los sistemas de agentes de forma eficiente. El CCT determina la jerarquía entre componentes de control y por tanto entre agentes, permitiendo determinar qué agentes están incluidos dentro de otros, o cómo establecer la descomposición de una tarea de control compleja en sus tareas más sencillas.

La gestión de los eventos por medio de combinaciones basadas en condiciones y la asociación de las acciones, permite a los agentes tomar decisiones propias, y por tanto tomar la iniciativa a partir de las circunstancias que se estén dando durante la acción de control. Las acciones que los agentes pueden tomar van desde las variación de características internas, como los umbrales de QoS o QoC, hasta llegar a realizar cambios estructurales en el sistema, como fragmentar o mover agentes, permitiendo ser los propios agentes los responsables de la gestión automática y de la evolución del sistema.

Un sistema de agentes implementado por medio de la arquitectura FSACtrl, tiene diversas ventajas:

- Permite estructurar detalladamente un sistema de control jerárquico basado en agentes.
- Permite implementar un sistema de agentes supervisado por un agente gestor, o autogestionado.

- Facilita la medición de la eficiencia del sistema a partir de los parámetros de QoS y QoC, incluyéndolos en los agentes, aun siendo parámetros asociados a los sistemas de comunicaciones o al control reactivo.
- Facilita la gestión de los eventos asociándolos a acciones que se activan a partir de eventos significativos del sistema.

Sin embargo, también se debe tener en cuenta que, al necesitarse un nodo de control donde ejecutar el sistema y que cada elemento dispone de una cola de mensajes, se ralentiza la ejecución en el caso de emplear el sistema de agentes directamente para el control.

Para situar al sistema dentro del entorno de sistemas similares, en el siguiente capítulo se realiza una comparación de las características de la arquitectura FSACtrl contextualizada a los sistemas distribuidos con los que es posible compartir escenarios de funcionamiento: los sistemas de redes de sensores, sistemas de control en red y sistemas basados en DDS.

5 Contextualización de la arquitectura FSACtrl

5.1 Introducción

5.1.1 Motivación

La arquitectura FSACtrl, presentada en esta tesis, se ubica en diversas áreas de investigación. Las áreas abarcan desde las arquitecturas middleware genéricas hasta los middleware específicos que dan soporte al control de sistemas distribuidos. Por ello, las áreas en las que se contextualiza la arquitectura FSACtrl son: los middleware de redes de sensores, como ejemplo de sistemas altamente distribuidos con muchos proveedores de información, los middleware de sistemas de control en red, como ejemplo de sistemas en los que la comunicación en ambos sentidos está equilibrada, y los middleware basados en el estándar DDS, como ejemplo de sistemas que emplean el mismo modelo que la arquitectura FSACtrl.

La comparación con modelos o middleware genéricos como CORBA, JMS, FIPA o DDS se realiza en [Poza et al., 2011] donde se concluye la conveniencia de seleccionar DDS como modelo de middleware con un soporte completo a la QoS.

5.1.2 Contexto

En la Figura 115 se muestran los ámbitos que cubren los middleware. Hay dos ámbitos claramente diferenciados: las redes de sensores inalámbricos y los sistemas de control en red. Desde un punto de vista de funcionalidad del middleware, los middleware basados en eventos, y concretamente los que se inspiran en el modelo de publicación y subscripción, son los más empleados en los sistemas anteriores. Si se considera importante que el middleware incluya el soporte a la calidad de servicio, los middleware basados en el estándar DDS deben incluirse como ejemplo de middleware transversal que cubre tanto las redes de sensores como los sistemas de control en red.

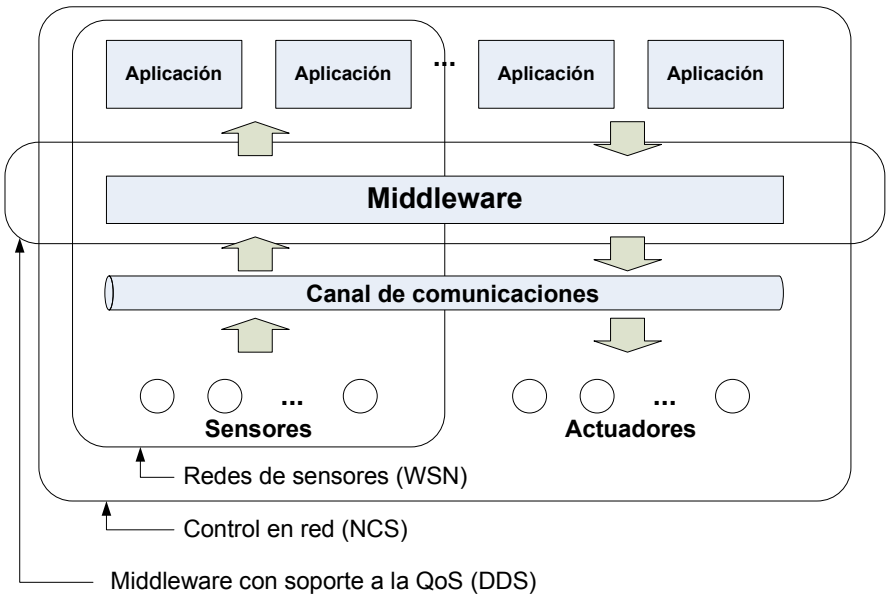


Figura 115. Contextualización de los diferentes sistemas distribuidos que emplean middleware.

Dentro de las redes de sensores, las redes de sensores inalámbricos (WSN) se consideran los sistemas más complejos, ya que la localización espacial inherente a los sensores, así como las características propias de la transmisión inalámbrica, hace que la gestión de la información sea ideal realizarla por el mediador entre las fuentes de datos y los consumidores, o lo que es lo mismo, el middleware.

Los sistemas de control en red (NCS) son otro tipo de sistemas donde la red da soporte tanto a la información proveniente de los sensores como a la información dirigida a los actuadores, después de ser procesada por los componentes de control. A diferencia de las redes de sensores, la cantidad de información que gestiona en ambos sentidos el middleware no es habitual que sea tan elevada. Sin embargo, en los NCS los requisitos temporales son más estrictos y por lo general el middleware debe cubrir estas restricciones.

Tanto en los WSN como en los NCS, el middleware debe cubrir todas las necesidades de comunicaciones. La adopción de la arquitectura de comunicaciones y del modelo de calidad de servicio de la capa DCPS hace que la arquitectura presentada pueda englobarse dentro del conjunto de middleware compatibles con DDS.

5.1.3 Descripción del capítulo

El apartado 5.2 analiza cómo FSACtrl proporciona las características requeridas en las arquitecturas de control distribuido y cumple con los requisitos de los sistemas de comunicaciones que se concluyeron en el Capítulo 2.

Seguidamente, en el apartado 5.3 se realiza la comparación de la arquitectura FSACtrl con los sistemas similares. La comparación permite contextualizar FSACtrl en las diversas áreas de aplicación: middleware de redes de sensores, middleware que emplean el estándar DDS y middleware de los sistemas de control en red.

5.2 Características de FSACtrl

En el capítulo 2 de la tesis se concluyeron las principales características deseables en los sistemas distribuidos de control en red. A continuación se contrastará la arquitectura FSACtrl con dichas características analizando el soporte que se da a las mismas.

5.2.1 Arquitectura como sistema de control distribuido

Tal como se concluyó en el capítulo 2, hay ciertas propiedades de un sistema distribuido, especialmente en los orientados al control inteligente, que permiten o facilitan la optimización del mismo. Estas propiedades son cubiertas por FSACtrl. En la Tabla 6 se muestra los componentes de la arquitectura que dan soporte a dichas características.

Tabla 6. Relación entre elementos FSACtrl y los requisitos para la optimización de un sistema.

Características de las arquitecturas de control distribuido / Características de FSACtrl		Autonomía Temporal	Autonomía Espacial	Confiabilidad	Recuperabilidad	Reutilización	Adaptabilidad	Estabilidad	Movilidad
Políticas de QoS		X	X	X				X	
Parámetros de QoC								X	
Ciclo de la Calidad					X		X	X	X
Espacio lógico de nombres			X		X		X		X
Componentes de control						X			
Agentes de control	Movilidad			X	X		X		X
	Clonación			X			X		X
Componentes DDS			X			X			

La autonomía temporal se logra a partir de las políticas de QoS. Para poder dar soporte a estas políticas es necesario marcar temporalmente los mensajes, lo cual se realiza en la arquitectura FSACtrl por medio del protocolo TTP (Time-Triggered Protocol) [Kopetz and Grunsteidl, 1993]. Además del uso del protocolo TTP, las políticas de QoS que dan soporte a la autonomía temporal son:

- Durability: permite mantener un historial de los datos que los agentes de control pueden emplear para extrapolarlos mientras no se tengan datos más recientes. A partir del margen determinado en el historial es posible conocer la precisión de los cálculos extrapolados.
- Lifespan: determina el tiempo que un dato es válido, por lo que proporciona un mecanismo interesante para dotar de cierta autonomía temporal. A partir del valor configurado se puede saber cuánto tiempo puede estar el elemento que dependa de este dato sin necesidad de actualizarlo. Además, aumentando su valor, se dota de más autonomía a costa de las prestaciones.

La autonomía espacial, desde el punto de vista de la capacidad de los elementos de continuar con parte de la información, la proporcionan las políticas de QoS, al igual que la autonomía temporal. La autonomía espacial, entendida como la no necesidad de conocimiento de la ubicación de los elementos, está soportada por la propia arquitectura, ya que el modelo de publicación-subscripción a través de un espacio de nombres lógicos aísla al elemento, el cual se limita a publicar en datos lógicos o a subscribirse a los mismos sin saber qué localización tienen los *Subscriber* o los *Publisher* de la información con la que trabaja.

La confiabilidad, dado que es una síntesis de las dos propiedades anteriores, está garantizada. Si se considera la fiabilidad como el valor añadido de la confiabilidad, la política de QoS *Reliability* proporciona el soporte para conocer cuánto de confiable es un sistema. Finalmente, la confiabilidad entendida como la capacidad, no solo de conocer los fallos, sino de poder afrontarlos, está garantizada en FSACtrl, ya que los componentes de control, desde el punto de vista reactivo, pueden amortiguar el efecto. La movilidad de los agentes de control permite trasladar el agente a nodos más confiables, y la clonación proporciona la redundancia en el procesamiento, necesaria en los sistemas tolerantes a fallos.

La recuperabilidad se considera la capacidad para reorganizar el sistema. La reorganización está soportada por diversas partes de la arquitectura FSACtrl. Para

reorganizarse es necesario un mecanismo que permita amortiguar el cambio para mantener el sistema dentro de unos márgenes de funcionamiento, para ello el ciclo de la calidad resulta idóneo, ya que permite ajustar dinámicamente el sistema. La reorganización de elementos implica que la arquitectura pueda soportar ciertas operaciones como dividir los componentes de control, moverlos o clonarlos. Estas operaciones están soportadas por la arquitectura.

En lo que a reutilización se refiere, la arquitectura debe ofrecer un mecanismo para emplear, en otro sistema, un elemento sin apenas cambios. En FSACtrl, los componentes de control están inspirados en los *ProcessModel* que tienen su modelo de patrón en XML, conocido como *ProcessMethod* en el estándar SWE. Estos patrones los tienen todos los elementos de la arquitectura FSACtrl lo que permite reutilizar cualquier elemento de la arquitectura en diferentes escenarios.

La adaptabilidad, considerada como la respuesta robusta ante el cambio de condiciones, es muy similar a la recuperabilidad. Sin embargo, esta última está orientada a la información mientras que la adaptabilidad está orientada a los componentes. En la arquitectura FSACtrl, al igual que en la recuperabilidad, el ciclo de la calidad y las operaciones de movimiento y clonación de agentes son los elementos que soportan la adaptabilidad.

Un sistema estable es un sistema que se mantiene en unos márgenes de funcionamiento. En FSACtrl, la estabilidad comienza en la definición de los márgenes de funcionamiento por medio de los parámetros de QoS y los parámetros de QoC. Con el ciclo de la calidad se proporciona un método para comprobar que el sistema está dentro de los márgenes determinados por los parámetros definidos cuantitativamente.

Finalmente, la movilidad o la clonación de componentes la soporta FSACtrl por medio de la implementación de agentes de control a partir de los componentes de control. El ciclo de la calidad proporciona el método por medio del cual decidir los criterios a partir de los cuales cambiar la morfología de los agentes o la distribución de los mismos en el sistema.

5.2.2 Requisitos como middleware de control en red

Una de las aportaciones del capítulo 2 consistía en el análisis de las características de los sistemas distribuidos y los sistemas de comunicaciones que les dan soporte para determinar una serie de requisitos deseables de todo sistema de control inteligente distribuido. En este subapartado se expondrá cómo la arquitectura FSACtrl cubre estos aspectos.

- Soporte al uso de parámetros. En FSACtrl se ha estudiado e implementado la parametrización tanto interna, en el caso de los elementos, como externa, en el caso de las comunicaciones. Los parámetros propuestos constituyen una de las principales aportaciones de la tesis. En las tablas comparativas de los siguientes apartados se esta característica se especifica por medio de la etiqueta “QoS”.
- Homogeneidad. El uso del modelo DDS, que define todos los componentes a partir de un elemento único, y la inserción en el mismo del modelo FSA, que simplifica y clasifica funcionalmente los componentes DDS, hace que la homogenización, especialmente en la estructura interna, de los elementos y consiguientemente de la arquitectura sea muy elevada. En las tablas comparativas de los siguientes apartados esta característica se especifica por medio de la etiqueta “Estándares”.

- Control de los aspectos temporales del sistema. La parametrización del sistema, por medio de los parámetros propuestos, permite controlar las características temporales internas de los elementos como las de los mensajes que se intercambian entre ellos. En las tablas comparativas posteriores, este aspecto está incluido en la etiqueta "QoS".
- Control de los aspectos espaciales del sistema. Este aspecto está cubierto por medio del espacio lógico de nombres. Tal como se ha comentado anteriormente, la jerarquización en una estructura de árbol permite abstraer al usuario o a los elementos del sistema de las ubicaciones físicas de los componentes. En las tablas comparativas posteriores este aspecto está incluido en la etiqueta "Jerarquía".
- La programación, en cuanto al acceso en algún lenguaje a las funciones del middleware, es importante en cuanto que permite adaptar el middleware a las especificidades del sistema ofreciendo un interfaz programable. En las tablas comparativas, esta característica se etiqueta como "Programación"
- Soporte a sistemas reactivos y deliberativos. La creación de los componentes de control y la posibilidad de incluir indefinidamente unos componentes de control dentro de otros, hace que el margen del nivel de detalle con el que es posible ver o programar el sistema sea tan amplio como lo desee el usuario. El nivel de detalle puede ir desde el sensor lógico de control hasta el de un agente que realice todo el control. Este aspecto está etiquetado en las tablas comparativas como "Agentes", para el caso deliberativo y como "Control" en el caso reactivo.
- Además de los anteriores criterios se ha considerado destacar el filtrado de mensajes. El filtrado es importante dado que una de las labores principales del middleware es seleccionar los mensajes que se cursarán a las capas superiores en función de criterios elaborados. En las tablas comparativas, esta característica se etiqueta como "Filtrado".
- Soporte a fallos. El soporte a fallos está contemplado en FSACtrl por medio de su implementación como sistema dinámico. La estructura de condiciones permite reaccionar, por medio de las acciones, a los eventos de errores que se produzcan.
- Gestión de la seguridad. La seguridad se contempla con el protocolo presentado en [Poza et al., 2005] por medio del cual se permite variar el nivel de encriptación de los mensajes y la identificación correcta de los componentes.

Los elementos de la arquitectura FSACtrl que cubren estos requisitos se muestran en la Tabla 7.

Tabla 7. Relación entre elementos FSACtrl y los requisitos como middleware de control en red.

Características de los middleware de control en red / Características de FSACtrl		QoS	Estándares	Jerarquía	Programación	Control	Agentes	Filtrado	Soporte a fallos	Gestión de la seguridad
Políticas de QoS		X						X		
Parámetros de QoC						X				
Ciclo de la Calidad									X	
Espacio lógico de nombres				X						X
Componentes de control				X		X				
Agentes de control	Movilidad						X			
	Clonación						X			
Componentes DDS			X		X					

5.3 Comparación de FSACtrl

5.3.1 Comparación con los middleware de sistemas de redes de sensores

5.3.1.1 Revisión de los middleware de sistemas de redes de sensores

Las redes de sensores son conjuntos de sensores ubicados en el espacio, generalmente sin conexión física, y sin una topología específica. Las redes de sensores son redes donde tienen especial relevancia los proveedores de información. La información que circula en las redes de sensores generalmente es muy heterogénea, tanto en las características como en los requisitos temporales o la semántica de los formatos de los datos.

En las WSN, los middleware deben poder dar soporte a la heterogeneidad inherente en el sistema. Además, las redes de sensores suelen caracterizarse por la gran cantidad de sensores que a medida que evolucionan los sistemas va en aumento, por lo que los middleware empleados en las WSN deben estar preparados para poder dar soporte a los cambios que el sistema tendrá a lo largo del tiempo.

La cantidad de middleware de soporte a redes de sensores es elevada, prueba de ello son el gran número de estudios relacionados con este tipo de middleware, entre los que destacan [Hadim and Mohamed, 2006], [Molla and Ahamed, 2006] o [Wang et al., 2008]. De los estudios anteriores, se han seleccionado las arquitecturas más citadas, ordenadas temporalmente por más antigüedad en la citación son: SINA [Srisathapornphat et al., 2000], Cougar [Bonnet et al., 2001], Maté [Levis and Culler, 2002], DSWare [Li et al., 2003], Impala [Liu and Martanosi, 2003], MiLAN [Heinzelman et al., 2004], EnviroTrack [Abdelzaher et al., 2004], Mires [Souto et al., 2004] y TinyDB [Madden et al., 2005].

5.3.1.2 Comparación con las características de los middleware de sistemas de redes de sensores

Es habitual que los entornos de trabajo sobre los que se despliegan las WSN sean dinámicos, por lo que hace que sea especialmente de valor la capacidad de adaptación a los cambios, así como a la cantidad de información que circula. Esta capacidad puede medirse a través de diversas características mostradas en la Tabla 8. A continuación, se muestra una descripción de consenso a partir de las referencias anteriores:

- Apertura: se entiende como apertura la capacidad de modificar el sistema fácilmente debido a los cambios de requisitos de funcionalidad sin que haya una variación del entorno.
- Escalabilidad: la definición más habitual que se asocia a la escalabilidad es la capacidad de adaptación a los cambios dinámicos de la topología de la red. En algunos casos [Hadim and Mohamed, 2006] la tolerancia a fallos se incluye en la escalabilidad ya que la capacidad de cambio de la red es parte de la reacción de la misma ante un error.
- Heterogeneidad: con este término se engloban diversos conceptos, pero principalmente la capacidad de funcionar en diferentes tipos de sistemas operativos, de redes o de dispositivos y equipos.
- Movilidad: la movilidad tiene diversas interpretaciones, aunque la más empleada en el área de las redes de sensores es la capacidad de adaptación a los cambios del entorno ofreciendo la misma funcionalidad.
- Facilidad de uso: este término se refiere a la forma en la que se interacciona con el middleware. Facilidades como el empleo de librerías o la existencia de interfaces gráficas sencillas de programación o de diseño del sistema son las que favorecen la valoración en este aspecto.
- Propósito general. Este término trata de valorar la capacidad del middleware de ser empleado para diversos tipos de sistemas.

Tabla 8. Middleware de redes de sensores: características de los sistemas de redes de sensores.

Middleware	Apertura	Escalabilidad	Heterogeneidad	Movilidad	Facilidad de uso	Propósito General
SINA	No	Parcial	No	Sí	Sí	Parcial
Cougar	No	No	No	No	Sí	Parcial
Mate	Sí	Sí	Sí	Sí	No	Sí
DSWare	Parcial	Parcial	No	No	Sí	Parcial
Impala	Sí	Sí	No	Sí	Parcial	No
MiLAN	Sí	Parcial	No	No	Sí	Parcial
EnviroTrack	---	Sí	Sí	Sí	No	No
Mires	Sí	Parcial	Parcial	Parcial	Sí	---
TinyDB	Si	No	Sí	---	---	---
FSACtrl	*Sí*	*Sí*	*Parcial*	*Parcial*	*Sí*	*Parcial*

De la Tabla 8 puede deducirse que la mayor parte de los middleware empleados en redes de sensores, especialmente a medida que evolucionan, cubren las características de los sistemas distribuidos. FSACtrl es un sistema abierto y escalable, ya que permite la creación modular y dinámica de elementos y la inclusión de los mismos conociendo el impacto que éstos tendrán en el funcionamiento del sistema. La heterogeneidad está cubierta en parte, ya que el prototipo desarrollado funciona sólo en un sistema Windows ™.

FSACtrl proporciona movilidad al poder programarse acciones asociadas a eventos relacionados con el cambio de condiciones. En cuanto a facilidad de uso y propósito general, FSACtrl proporciona un bajo número de elementos con una funcionalidad específica y elementos programables en función de las necesidades. Consecuentemente, la arquitectura FSACtrl es adecuada para la adquisición de datos en los sistemas complejos de sensores.

Uno de los aspectos destacables que se puede observar en la Tabla 8 es que la mayoría de los middleware cubren solo algunas de las características, debido a su especificidad en el ámbito de aplicación, por lo que es complicado encontrar una arquitectura completa. Por lo general, las arquitecturas de middleware de redes de sensores dan soporte a los sistemas abiertos y escalables, tal como es de esperar ya que son las principales características de los WNS. Evidentemente, un middleware que proporcione estas características es potencialmente un buen middleware. Sin embargo, las tareas asociadas a estas características, como el descubrimiento de los nuevos componentes o el mantenimiento de grandes bases de datos con información topológica de la ubicación de las fuentes de datos, hacen que el middleware tenga un alto consumo de procesamiento sin ofrecer una optimización en la ejecución.

Es en el soporte a la heterogeneidad o a la movilidad, entendida como la posibilidad de funcionar en un entorno diferente sin apenas cambios, donde algunos de los sistemas no dan soporte. Esto provoca que no sean sistemas de propósito general, ya que están fuertemente ligados a los entornos sobre los que se despliegan.

5.3.2 Comparación con los middleware basados en DDS

5.3.2.1 Revisión de los middleware basados en DDS

Hay un gran número de middleware basados en DDS, entre ellos se han seleccionado aquellos que se mencionan con más frecuencia en la bibliografía relativa. Los middleware revisados, por orden de aparición de la citación, son: CDDS [Dursun, 2006], NDDS [RTI, 2006], TAO DDS [Busch, 2006], JacORB [JacORB, 2007], RTjDDS [Caruso et al., 2007], MilSoft-DDS [Kutluca et al., 2007], DDSS [Kwon et al., 2008], InterCOM DDS [Gallium, 2009], OpenSplice [PrismTech, 2009], CoreDX-DDS [TwinOaks, 2009], JAVA DDS [SSI, 2010], OpenDDS [OCI, 2010], MicroDDS [Sebastian, 2010], ServiceDDS [Dianes et al., 2010] y TinyDDS [Boonma and Suzuki, 2010].

5.3.2.2 Comparación con la funcionalidad de los middleware basados en DDS

Los middleware revisados se muestran en la Tabla 9. La interpretación de cada término (columna) de la tabla descriptiva es la siguiente:

- Desarrolladores: para cada middleware evaluado, se ha considerado la fuente de las primeras versiones. En algunos casos como JacORB el desarrollo es compartido por empresas y desarrolladores particulares.
- Implementación: la implementación se ha clasificado en tres ámbitos, “Experimental”, cuando el middleware es un proyecto o no ha sido posible localizar alguna implementación aunque fuese un prototipo; “Abierta”, cuando para el middleware es posible localizar alguna implementación, incluido un prototipo, y no son productos comercializados; y “Comercial”, para aquellas implementaciones desarrolladas en empresas que proporcionan soporte a las mismas.

- Funcionalidad: la funcionalidad del middleware se entiende como la orientación principal u objetivo principal definido.

Tabla 9. Middleware basados en DDS: implementación y funcionalidad.

Middleware	Desarrolladores	Implementación	Funcionalidad
CDDS	Middle East Technical University (Turkey)	Experimental	Soporte DDS para CORBA
NDDS	RTI	Comercial	Bus de tiempo real
TAO DDS	Object Computing INC	Abierta	Soporte DDS para CORBA
JacORB	OCI, PrismTech. Programadores	Experimental	Soporte DDS para CORBA
RTjDDS	Kydosoft	Experimental	Soporte DDS para JAVA
MilSoft-DDS	MilSoft	Comercial	Bus de tiempo real
DDSS	Chungnam National University (Korea)	Experimental	Soporte DDS para SQL
InterCOM DDS	Gallium	Comercial	Integración de datos geoespaciales
OpenSplice	PrismTech	Comercial	Bus de tiempo real
CoreDX-DDS	TwinOaks	Comercial	Bus de tiempo real para sistemas empotrados
JAVA DDS	Space Software Italia	Experimental	Soporte DDS para JAVA
OpenDDS	Object Computing INC	Abierta	Bus de tiempo real
MicroDDS	Icoup Consulting	Experimental	Sistemas empotrados.
ServiceDDS	Universidad de Málaga (Spain)	Experimental	Soporte al acceso a servicios vía Web
TinyDDS	University of Massachusetts, Boston	Abierta	Soporte DDS para redes de sensores
FSACtrl	*Universidad Politécnica de Valencia (Spain)*	*Abierta*	*Sistemas distribuidos de control. Agentes*

Como se aprecia en la Tabla 9, las características de los Middleware basados en DDS son muy similares centrándose la gran mayoría en proveer servicios de comunicaciones de tiempo real crítico. DDS tiene implementaciones especialmente como producto experimental y comercial, con una implantación muy grande en sistemas militares críticos.

La funcionalidad de los middleware con soporte a DDS es, en la mayor parte de los casos, muy específica. Los productos comerciales están orientados a ofrecer un middleware que actúe como bus, en la línea de los buses empresariales, mientras que los productos experimentales están orientados a dar soporte a una tecnología o entorno concreto.

FSACtrl tiene una implantación abierta dado su carácter experimental. Las principales áreas de uso son los sistemas distribuidos de control, ya que la inclusión de los sensores lógicos de control, y de los componentes de control, permiten ofrecer esta funcionalidad desde el middleware. La implementación de agentes a partir de los elementos de FSACtrl es la otra principal funcionalidad ya que se dispone de contenedores de elementos, los sensores lógicos de control, que pueden representar comportamientos y pueden reaccionar autónomamente.

5.3.3 Comparación con los middleware de sistemas de control en red

5.3.3.1 Revisión de los middleware de sistemas de control en red

Como ya se expuso, las redes de sensores son principalmente proveedores de datos en los que la mayor parte de los mismos son lecturas de sensores, quedando la escritura relegada a los aspectos de configuración. Sin embargo, en los sistemas de control sobre redes o sobre sistemas distribuidos el sentido de la información es doble, no solo proviene de los sensores, sino que también se debe tener en cuenta la información que se transmite a los actuadores.

Con la información circulando en dos sentidos en el middleware, se cierra el bucle de control pero con las correspondientes particularidades de estar actuando en un sistema distribuido. Los sistemas que permiten realizar una acción de control sobre una red, con un control de tiempo real, se conocen como Sistemas de Control en Red (NCS, de Networked Control Systems).

El área de los NCS está en crecimiento, centrándose la investigación en la formalización de los sistemas y estrategias [Huo et al., 2004], especialmente el control temporal [Tang and Yu, 2007] o el sistema de comunicaciones [Li and Wang, 2008]. Actualmente la tendencia está en el uso de los NCS en todos los ámbitos del control [Gupta and Chow, 2010]. Entre ellos, es especialmente interesante e innovador el uso de los mismos para el co-diseño de sistemas de control complejos [Yan et al., 2011]. Sin embargo, las arquitecturas de middleware explícitamente orientadas a NCS son escasas: [Bergmans et al., 2000] ControlWare [Zhang et al., 2002] y EtherWare [Baliga and Kumar, 2004].

5.3.3.2 Comparación con las características de los middleware de sistemas de control en red

Dado que FSACtrl es un middleware con soporte al control en red, la primera comparación se realiza en este ámbito (Tabla 10).

Tabla 10. Middleware de sistemas de control en red: características.

Nombre	QoS	Estándares	Jerarquía	Programación	Control	Agentes	Filtrado
Bergmans et al., 2000	Parcial	No	No	CORBA	Sí	No	No
Controlware	Parcial	No	No	C++	Sí	Parcial	No
Etherware	No	No	Sí	JAVA	Sí	No	Sí
FSACtrl	*Sí*	*DDS, SWE*	*Sí*	*C++, Agentes*	*Sí*	*Sí*	*Sí*

La arquitectura presentada en [Bergmans et al., 2000] no tiene nombre, debido a que se trata de un proyecto teórico.

En la Tabla 10 se aprecia cómo las cuatro arquitecturas tienen un lenguaje de programación para poder dar soporte a la programación de los algoritmos de control (C++ o JAVA) o CORBA como modelo para el soporte a objetos distribidos.

El uso de estándares no es común, ya que DDS, como primer estándar de publicación-subscripción con soporte implícito a la QoS, aparece después de las primeras arquitecturas. Por ello, la QoS está considerada como el control temporal de los mensajes, lo que concuerda con la orientación del control planificado. Sin embargo, las tendencias actuales de control basado en eventos requieren una consideración más amplia de la QoS que abarque aspectos temporales, de flujo de mensajes o de jerarquización del sistema.

El filtrado de mensajes y la posibilidad de jerarquizar los componentes de control son características que sólo ofrecen Controlware y FSACtrl. Estos dos últimos aspectos son característicos de los middleware y recientemente han despertado un gran interés para su aplicación en los sistemas de control. Por ejemplo, en DDS hay posibilidad de agrupar zonas de control, aunque no jerarquizarlas.

Considerando los sistemas de redes de sensores como sistemas de control distribuido, a continuación (Tabla 11) se presenta la comparativa en función de estas características.

Tabla 11. Middleware de redes de sensores: características de los sistemas de control en red.

Middleware	QoS	Estándares	Jerarquía	Programación	Control	Agentes	Filtrado
SINA	No	No	Sí	SQL	No	No	Sí
Cougar	No	No	No	No	No	No	No
Mate	No	TinyOS	No	C	No	No	No
DSWare	Parcial	No	Sí	SQL	No	No	Sí
Impala	No	No	Sí	No	No	Sí	Sí
MiLAN	Sí	No	No	No	No	No	Sí
EnviroTrack	Sí	TinyOS	No	C	No	No	No
Mires	No	TinyOS	Sí	C	No	No	No
TinyDB	No	PostGIS	No	SQL, C++	No	No	Sí
FSACtrl	*Sí*	*DDS, SWE*	*Sí*	*C++, Agentes*	*Sí*	*Sí*	*Sí*

Cabe destacar que algunos de los sistemas anteriores (Mate, EnviroTrack) funcionan bajo el sistema operativo TinyOS [Hill et al., 2000]. Este aspecto es habitual en muchos de los Middleware para redes de sensores, lo que limita algunas de las características de los mismos, como la extensión a otros sistemas.

Finalmente, se evalúa la capacidad de los middleware basados en DDS para el soporte a las características del control distribuido (Tabla 12). La interpretación de la columna QoS incluye entre paréntesis la cantidad de políticas de QoS a las que da soporte el sistema.

Tabla 12. Middleware basados en DDS: características de los sistemas de control en red

Middleware	QoS	Estándares	Jerarquía	Programación	Control	Agentes	Filtrado
CDDS	Propuesta (1)	DDS	No	CORBA	No	No	No
JacORB	No	DDS	No	CORBA	No	No	No
NDDS	Parcial	DDS	No	C, C++, Java	No	Parcial	Sí
TAO DDS	Parcial (4)	DDS	No	CORBA	No	No	No
RTjDDS	No	DDS	No	JAVA	No	No	No
MilSoft-DDS	Parcial (8)	DDS	No	C++	No	No	Sí
DDSS	Parcial (2)	DDS	No	SQL	No	No	No
InterCOM DDS	Parcial (8)	DDS	No	C++, Java	Parcial	No	No
OpenSplice	Completa	DDS	No	C, C++, C#, Java	No	No	Sí
CoreDX-DDS	Parcial (18)	DDS	No	C, C++, Java	No	No	Parcial
JAVA DDS	No	DDS	No	Java	No	No	No
OpenDDS	Completa	DDS	No	C++, Java	No	No	Sí
MicroDDS	No	DDS	No	C	No	No	No
ServiceDDS	No	DDS	No	XMPP	No	No	No
TinyDDS	Parcial (2)	DDS	No	nesC, Java	No	Parcial	Parcial
FSACtrl	*Parcial (12)*	*DDS, SWE*	*Sí*	*C++, Agentes*	*Sí*	*Sí*	*Sí*

La QoS propuesta por el middleware CDDS, es la llamada BLOCK, orientada específicamente a la optimización de CORBA sobre un modelo de publicación-subscripción. Este aspecto no es único ya que DDS se emplea más como modelo de publicación-subscripción que como modelo de middleware con soporte a la QoS. El lenguaje XMPP empleado por ServiceDDS es un protocolo por el que acceder al middleware, en lugar de ser integrado como librería en el sistema que lo emplea.

5.3.4 Discusión

De las comparaciones anteriores pueden deducirse algunos aspectos interesantes. El lenguaje más empleado como implementación o como interfaz es C y C++, lo que coincide en gran medida con los lenguajes empleados en los middleware de sistemas de control en red y de los sistemas de redes de sensores, por lo que los middleware basados en DDS resultan adecuados para adaptarse a estos entornos.

El soporte a agentes no aparece en prácticamente ningún middleware. Esto es normal, ya que los middleware comerciales están orientados a la optimización en tiempo real y en fiabilidad de la comunicación, especialmente si se tiene en cuenta que son middleware que trabajan en sistemas críticos. Los middleware experimentales están muy especializados sin que se tenga constancia de uno que busque aportar estandarización en las comunicaciones a los sistemas de agentes como hace FSACtrl.

El soporte a control es otro de los aspectos donde los middleware inspirados en el estándar DDS analizados no se centran. El incluir parte del control en las comunicaciones, como ya se ha comentado, es uno de los campos más recientes de investigación, y no es uno de los requisitos más importantes de los buses comerciales.

En lo que respecta al soporte a la QoS hay unas cuestiones muy interesantes. Como es de esperar, cada middleware ofrece las políticas de QoS más adecuadas al entorno en el que deben trabajar. Los sistemas comerciales o abiertos (pero completos) ofrecen más políticas de QoS que los sistemas experimentales. Esto se debe, además de una cuestión comercial, a que los sistemas experimentales usan el modelo de publicación-subscripción del DCPS de DDS como método de comunicación, pero no están tan interesados en las políticas de QoS.

Otro de los aspectos interesantes es el soporte al filtrado de mensajes, no solo en sus características temporales o de flujo, sino también en cuanto al contenido. En este caso la mayor parte de los middleware no ofrecen la posibilidad de dicho filtrado, excepto en los sistemas comerciales o abiertos, donde se ofrece dada la potencia que proporciona al sistema.

Finalmente, la posibilidad de jerarquizar los componentes del sistema es el aspecto donde FSACtrl ofrece otra ventaja con respecto al resto de middleware. Para los sistemas en los que la mayoría de los middleware analizados se desenvuelven, la organización de los componentes en dominios aislados es suficiente. En cambio, en FSACtrl, especialmente cuando se trata del soporte a agentes, la posibilidad de generar dependencias conceptuales, incluso ontológicas, es uno de los aspectos de más interés, que añade una cierta potencia al sistema que emplee FSACtrl.

5.4 Conclusiones

5.4.1 Tendencias

Los middleware para soporte a SWN y NCS aparecen cerca del año 2000 y, aunque algunos se mantienen activos, los últimos en aparecer lo hacen hacia el 2005. A partir del 2006 es cuando los middleware comienzan a tomar más responsabilidades dentro del sistema, lo que implica una convergencia en las necesidades de los sistemas de control distribuido y los sistemas de sensores distribuidos (Figura 116).

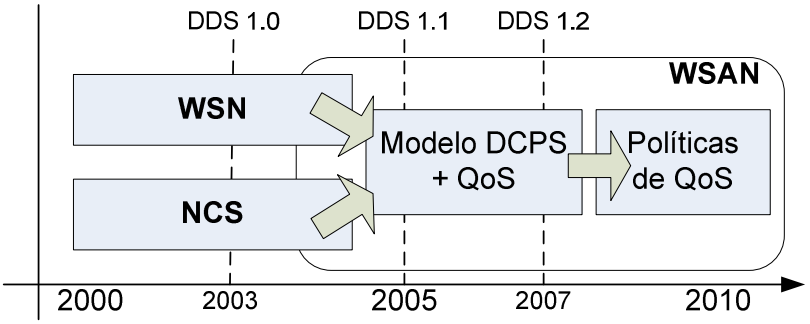


Figura 116. Evolución temporal del ámbito de los middleware analizados.

La convergencia se da entre los sistemas que manejan grandes cantidades de información con los sistemas que, manejando cantidades menos voluminosas de información, tienen requisitos temporales críticos. Esta convergencia se produce aproximadamente cuando aparece la versión más conocida de DDS [OMG, 2005]. Para gestionar tanto los aspectos de información propios de los WSN como las restricciones temporales propias de los NCS, se hace necesario emplear políticas que gestionen y que procesan estos aspectos. Parece ser que la evolución natural de los sistemas distribuidos en red apunta a que los middleware basados en estándares y con soporte a la QoS se hacen cada vez más necesarios. Actualmente la fusión de los dos tipos de sistemas se plasma en el concepto de Redes Inalámbricas de Sensores y Actuadores (WSAN, por las siglas *Wireless Sensors and Actuator Networks*) como ejemplo de sistemas complejos donde las comunicaciones deben proporcionar soporte al control incluso de tiempo real estricto [Xia, 2008].

5.4.2 Contextualización

En el presente capítulo se han contextualizado las características de FSACtrl en comparación con las características de las arquitecturas o sistemas empleados en WSN (como ejemplo de sistemas con amplios requisitos de información), NCS (como ejemplo de sistemas con amplios requisitos temporales) y sistemas basados en DDS (como ejemplo de sistemas que emplean una arquitectura similar).

Como arquitectura para el tratamiento de la información en sistemas distribuidos amplios FSACtrl cumple con los requisitos. En lo que respecta a las características propias del control distribuido, las arquitecturas de soporte a los WSN tienen carencias en el soporte a sistemas de agentes y en el control incluido en el middleware. Las carencias son comprensibles ya que, inicialmente, los WSN no tienen como objetivo dar soporte a un bucle de control. La propia configuración a partir de componentes jerárquicos y de autodescubrimiento, como sistema de agentes, ofrece una adaptabilidad muy alta a la heterogeneidad implícita en los diferentes sistemas WSN.

Como middleware de soporte a sistemas de control en red, FSACtrl cubre las características deseables. Los aspectos de soporte a estándares, QoS y filtrado son implícitas a la capa DCPS del modelo DDS. FSACtrl aporta a este tipo de sistemas el soporte a agentes y a la jerarquización de los componentes de control.

FSACtrl también cumple con los requisitos de la capa DCPS del modelo DDS, por lo que se adquieren las ventajas del modelo, especialmente en lo que se refiere al uso de las políticas de QoS, como medio para administrar los servicios, medir el sistema y ajustarlo en función de los requisitos del mismo.

6 Trabajo experimental

6.1 Introducción

6.1.1 Motivación

En los anteriores capítulos se ha descrito los elementos de la arquitectura FSACtrl y se ha contextualizado la arquitectura entre los diferentes ámbitos de trabajo. Sin embargo, es necesario comprobar la viabilidad de la arquitectura corroborando las propuestas que se han expuesto anteriormente.

Para ello, se ha desarrollado un entorno de pruebas consistente en un simulador de robots móviles y una aplicación con la doble funcionalidad de diseño de los elementos de la arquitectura FSACtrl que a la vez implementa el control.

6.1.2 Contextualización

Para validar las propuestas expuestas en los anteriores capítulos se han implementado y controlado los primeros vehículos de Braitenberg [Braitenberg, 1984]. Estos vehículos son una plataforma muy adecuada para probar la arquitectura ya que implementan comportamientos muy básicos de navegación conectando sensores y motores con un control mínimo.

El hecho de que el control necesario para la navegación de los primeros vehículos de Braitenberg no realice operaciones complejas lo convierte en un buen candidato a ser embebido en el middleware. De esta forma, es posible descargar del comportamiento reactivo a otros componentes de control más complejo.

Los primeros tres vehículos de Braitenberg se basan en las conexiones, sin intermediación de operaciones, que vayan más allá de la adecuación de los mensajes de entrada a los mensajes de salida, entre sensores y motores. El vehículo introduce las funciones entre los sensores y motores, lo que añade un mínimo procesamiento de control. Los primeros cuatro vehículos son entornos muy adecuados para probar la viabilidad de la arquitectura como middleware y cómo es posible optimizar el sistema incluyendo las operaciones reactivas de control en las comunicaciones.

El quinto vehículo de Braitenberg introduce la lógica entre los sensores y actuadores lo que implica la selección entre diferentes comportamientos en función de las condiciones del entorno En este vehículo se puede experimentar cómo la arquitectura FSACtrl permite optimizar la navegación al conocer en el middleware parte de la información de control.

6.1.3 Descripción del capítulo

En el apartado 6.2 se comprueba la viabilidad de la arquitectura FSACtrl como sistema de control reactivo y la optimización del control por medio de las políticas de QoS y del uso de los sensores lógicos de control. En el apartado 6.3 se emplean las políticas de QoS para optimizar dinámicamente un control reactivo basado en comportamientos, donde es necesaria una jerarquización de los mismos y una priorización de la información.

6.2 Aplicación de políticas de QoS y de sensores lógicos de control

6.2.1 Escenarios

Los primeros cuatro vehículos de Braitenberg se probarán para tres escenarios diferentes (Figura 117). El primer escenario (Figura 117.a) es el más básico ya que implementa el control y las comunicaciones sin emplear las políticas de QoS y sin incluir sensores lógicos de control que permitan optimizar el funcionamiento.

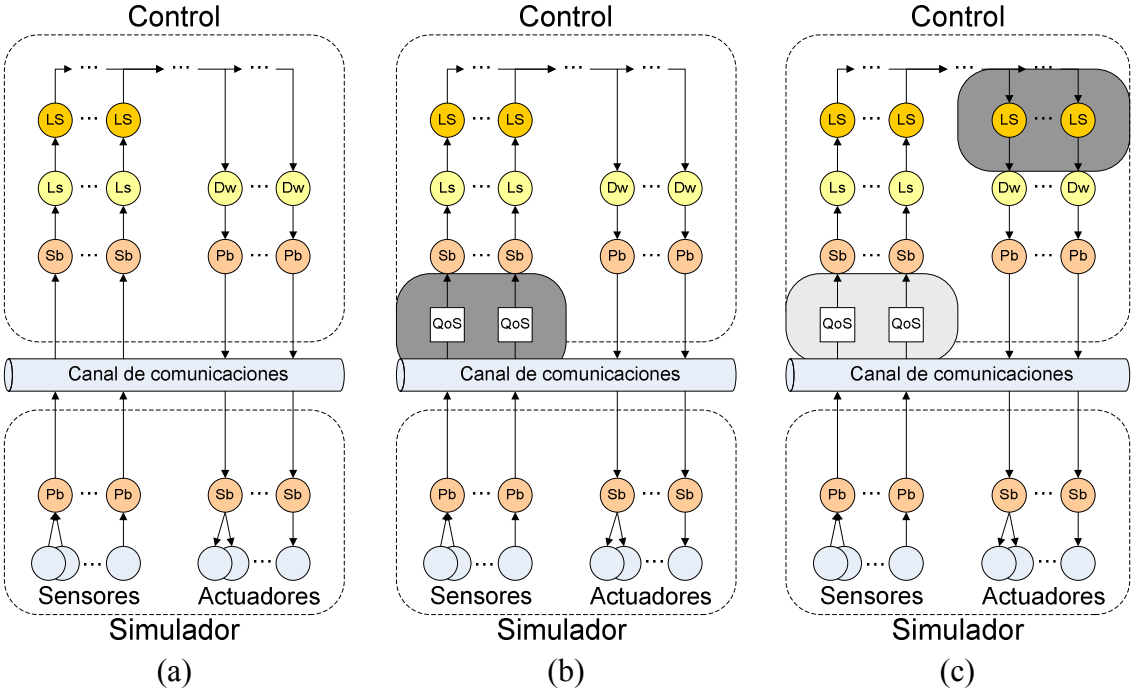


Figura 117. Escenarios de experimentación para la optimización basada en la aplicación de políticas de QoS y de sensores lógicos de control.

El segundo de los escenarios (Figura 117.b) incluye la gestión de las comunicaciones por medio de políticas de QoS, en el caso implementado se han empleado las políticas de QoS para disminuir la frecuencia de muestreo de los sensores y descargar al control con una menor frecuencia de mensajes dado que la velocidad máxima del vehículo no precisa de tanta información. En el tercer escenario se seleccionan internamente aquellos mensajes que sean significativos para el control, permitiendo descargar al componente de control con mensajes cuyo procesamiento no produciría una acción de control.

6.2.2 Vehículo 1. Desplazamiento

6.2.2.1 Descripción

El primer vehículo de Braitenberg dispone de tan sólo un sensor, habitualmente de luz, y un actuador, habitualmente un motor. El sensor y el actuador están conectados directamente de manera que la velocidad del motor es directamente proporcional a la señal proporcionada por el sensor.

El comportamiento del vehículo es igualmente básico, consistente en avanzar en línea recta en la medida en que haya luz. Con este primer vehículo es la cota inferior, en lo que a control se refiere, ya que es el vehículo más básico que es posible realizar.

En cuanto a lo que respecta a la implementación mediante los elementos de la arquitectura FSACtrl, el vehículo sirve para mostrar cómo recibir señales de los sensores y enviarlas a los actuadores sin mediar apenas procesamiento.

6.2.2.2 Implementación

6.2.2.2.1 Simulación

El vehículo se compone de un único sensor de luz y de dos motores, para proporcionar la correspondiente estabilidad. En el entorno basta con disponer de una fuente de luz que proporcione la información al sensor. En la Figura 118 se muestra el modelo del vehículo 1 y su implementación en el simulador.

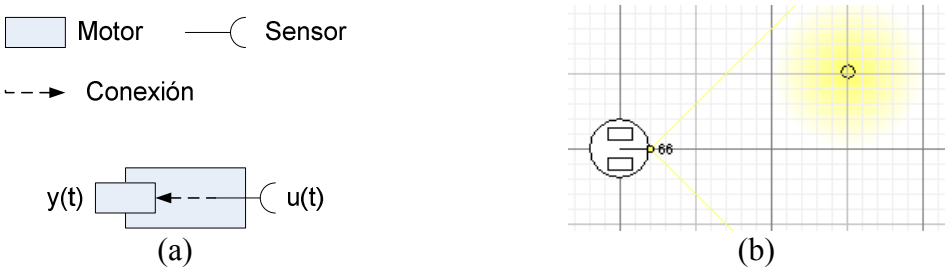


Figura 118. Vehículo 1 de Braitenberg: conceptual (a) y simulado (b).

Inicialmente se considera que no existe error en la medición de los sensores ni fricción en la actuación de los motores, en tal caso el vehículo se mueve en línea recta en función del nivel de luz detectado. En el caso de haber perturbaciones en la adquisición de datos o en el movimiento, estas resultan de especial interés para añadir comportamientos a un vehículo tan sencillo. Por ejemplo, el rozamiento determinará el umbral de error que se producirá en la velocidad o incluso el umbral en el que el vehículo se detendría pese a estar recibiendo luz en el sensor. Además, el error en los motores ocasionará que no necesariamente se siga la dirección en una línea recta exacta, por lo que el error de dirección generará un movimiento no rectilíneo lo que da un margen de movimiento variable.

6.2.2.2.2 Control

La implementación del vehículo 1 por medio de los elementos de la arquitectura FSACtrl es muy sencilla (Figura 119). Las comunicaciones, aunque menos relevantes, se realizan por medio de conexiones TCP con el vehículo simulado donde cada sensor o actuador se corresponden a un puerto TCP concreto. Los puertos TCP se asocian a los datos lógicos del espacio de nombres. El espacio lógico de nombres (LNT) se divide en dos áreas: la de los datos de los sensores y el área de los datos de los motores. Los sensores se organizan según el tipo de sensor, en el caso del vehículo 1 solo es necesario un sensor de luz. Los motores no precisan organizarse salvo para diferenciar el izquierdo del derecho.

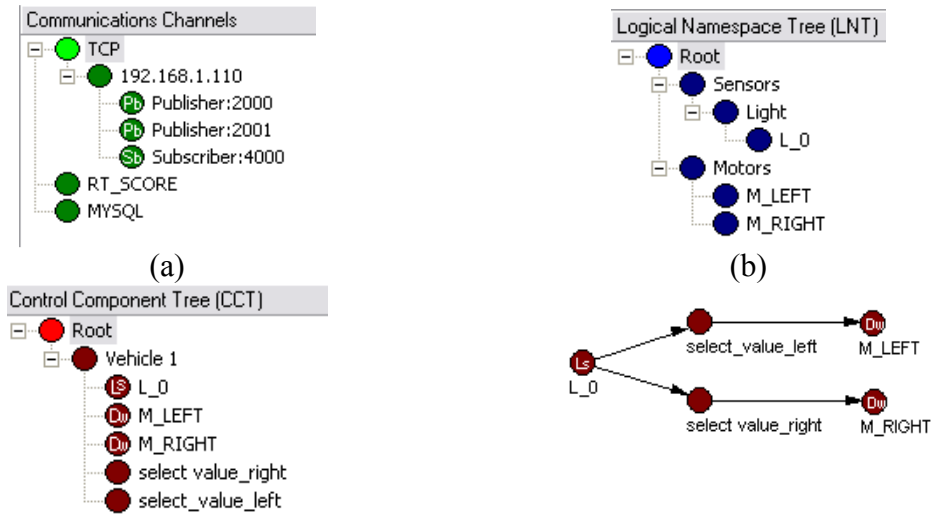


Figura 119. Vehículo 1 de Braitenberg: Comunicaciones (a), LNT (b), CCT (c) y LSG correspondiente al escenario 1 (d).

El árbol de componentes de control (CCT) es muy básico ya que sólo se deben conectar la entrada del sensor, adaptar la velocidad del motor a la medición del sensor y enviar dicha información, por lo que solo es necesario disponer de un componente de control, *Vehicle 1* en la Figura 119. El Grafo de sensores lógicos (LSG) correspondiente al componente de control está formado por un único *Listener* que recibe los datos del sensor y envía el mensaje a dos sensores lógicos que recogen el valor del sensor y lo adaptan al formato del mensaje de los motores. Los sensores lógicos adaptadores de formato se conectan respectivamente a un sensor lógico de comunicaciones *DataWriter* que se corresponderá con cada motor del vehículo.

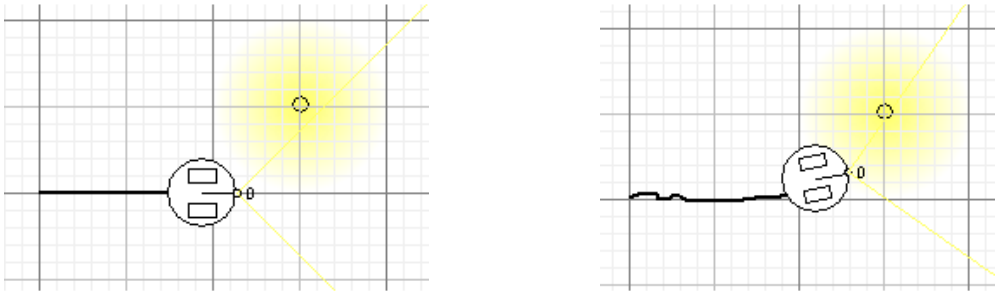


Figura 120. Vehículo 1 de Braitenberg: sin distorsiones (a) y con distorsiones (b).

El comportamiento del vehículo 1 de Braitenberg se muestra en la Figura 120. Sin rozamiento en motores ni distorsión en las mediciones (a) y con una mínima distorsión para acercar el control a la realidad.

6.2.2.3 Experimentación

6.2.2.3.1 Escenarios

Los primeros vehículos de Braitenberg son el escenario mínimo para probar la arquitectura FSACtrl. En los primeros vehículos, el control cede el protagonismo a las comunicaciones, por lo que es posible comprobar cómo funcionan y qué pueden aportar al control los elementos de comunicaciones de la arquitectura FSACtrl.

Ante la misma posición de partida y la misma posición de las fuentes de luz, si no se consideran distorsiones, los primeros vehículos de Braitenberg siempre tienen la misma trayectoria. Además, la arquitectura FSACtrl permite que parte del control esté delegado en el middleware, bien por medio de las políticas de QoS o bien por medio de los sensores lógicos de control, por lo que el escaso control que se necesita en los vehículos de Braitenberg es posible embeberlo y evaluarlo junto al middleware. Esto nos permite comparar, entre otros aspectos, cómo diferentes configuraciones de las comunicaciones afectan al control.

El parámetro de simulación que determina lo óptima que ha sido una navegación simple de un robot (sin misión determinada) es la velocidad promedio para un mismo punto de partida y punto de destino. En este caso es posible evaluar cómo afecta la frecuencia de muestreo a la carga de las comunicaciones, a través del middleware. Para asegurar una homogeneidad en los datos, todos los casos se han realizado para una distancia de navegación constante dentro de cada uno de los robots.

El escenario base (señalado con 1 en las series de las gráficas) consiste en hacer navegar el vehículo 1 sin programar políticas de QoS en el middleware y sin incluir sensores lógicos de control que realicen una optimización de las comunicaciones, como enviar una actualización de velocidad sólo en el caso de haber cambiado la entrada del sensor. En este caso se mide para diferentes frecuencias de muestreo.

La primera optimización consiste en filtrar la frecuencia de muestreo que el middleware ofrece al control. Esto se hace por medio de la política de calidad de servicio *Lifespan* asociada al *Listener* de manera que sólo se ofrecen al control aquellos mensajes que superen una diferencia temporal con respecto al tiempo que se considera válido un mensaje. Este escenario se ha señalado con el número 2 en las series de las gráficas.

A la optimización anterior, se le añade un sensor lógico de control que filtra los mensajes en función del cambio de contenido. En este caso, solamente se envían al control aquellos mensajes que hayan variado en un tanto por cien del valor del anterior mensaje. De esta forma el middleware está filtrando aquellos mensajes del sensor que, dado que la velocidad del motor es directamente proporcional al valor del sensor, se sabe que no producirán cambios. Este escenario se ha señalado con el número 3 en las series de las gráficas.

6.2.2.3.2 Resultados y discusión

En las pruebas realizadas se mide la velocidad del vehículo y la carga que tiene el middleware a través de la carga representativa del componente de control, teniendo en cuenta que en el segundo escenario se añade una política de QoS y en el tercero, además de la política de QoS se añade un sensor lógico de control adicional. El parámetro que se varía es la frecuencia de muestreo del sensor. Para observar y comparar la carga del middleware, se ha aumentado deliberadamente el tiempo de respuesta de los elementos de la arquitectura. Los resultados se muestran en la Figura 121

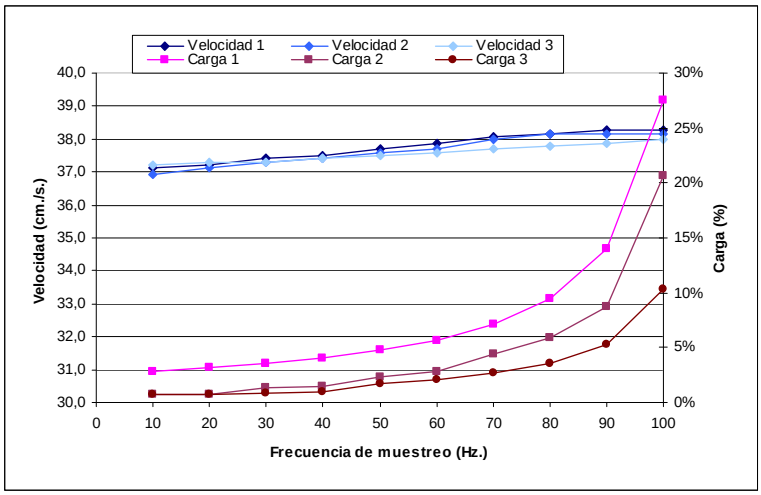


Figura 121. Vehículo 1 de Braitenberg: frecuencia de muestreo y carga del middleware.

Como se puede apreciar, la velocidad sufre una muy ligera disminución a medida que se aplican las optimizaciones. Esto se debe a que en el primer escenario la actualización de la velocidad es mucho más frecuente al darse en todos los casos en que el sensor envía un dato. Sin embargo es una disminución muy baja, menor del 1%, y dependiendo de los requisitos puede asumirse teniendo en cuenta la optimización que se produce en la carga del middleware.

6.2.3 Vehículo 2. Miedo y agresión

6.2.3.1 Descripción

El primer vehículo de Braitenberg muestra la filosofía reactiva de los mismos, pero no es operativo ya que sólo permite la navegación en línea recta. El segundo vehículo de Braitenberg dispone de dos sensores y dos motores alimentados directamente de forma positiva, es decir, el aumento en el valor del sensor implica el aumento en la velocidad del motor. Las posibles combinaciones de alimentación entre los dos sensores y los dos motores forman las dos diferentes versiones de los vehículos (Figura 122.a y Figura 122.b) y su implementación en el simulador.

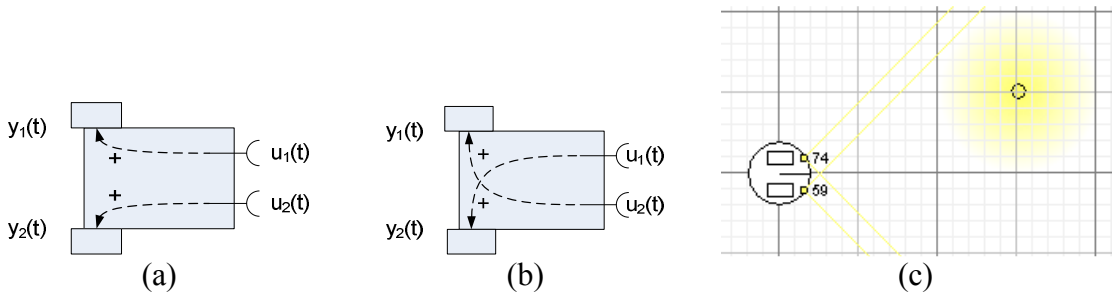


Figura 122. Vehículo 2 de Braitenberg conceptual: Miedo (a), agresión (b) y simulado (c).

Braitenberg descarta la conexión de los dos sensores con cada uno de los motores ya que eso implicaría un vehículo similar al vehículo 1, o bien la necesidad de componer las dos señales de los dos sensores y eso implica una fusión sensorial reservada para vehículos posteriores.

Con el vehículo 2 de Braitenberg se comprueba cómo es posible que un sistema muy simple reaccione alejándose o acercándose a un estímulo externo según sea la conexión interna que se realice con los motores.

En cuanto a la implementación a través de elementos de la arquitectura FSACtrl, el vehículo 2 de Braitenberg permite mostrar cómo emplear los espacios de nombres para ofrecer diferentes visiones de los mismos componentes.

6.2.3.2 Implementación

6.2.3.2.1 Simulación

En las dos versiones del vehículo 2 los elementos necesarios para la simulación son los mismos: un robot con dos sensores de luz y dos motores junto a una fuente de luz que alimente los sensores. El entorno puede observarse en la Figura 122.c.

6.2.3.2.2 Control

En los vehículos 2 de Braitenberg, las comunicaciones son similares al vehículo 1, con el nuevo sensor añadido como nuevo puerto al que conectarse para recibir los correspondientes datos (Figura 123.a). En cuanto al espacio de nombres lógicos es posible hacer diferentes organizaciones, como por ejemplo por lados (derecho e izquierdo) del vehículo. Por coherencia con el vehículo anterior se organizan por sensores y motores (Figura 123.b). El árbol de componentes de control es similar al del vehículo 1, aunque en este caso hay dos *Listener*, uno por cada sensor de luz (Figura 123.b).

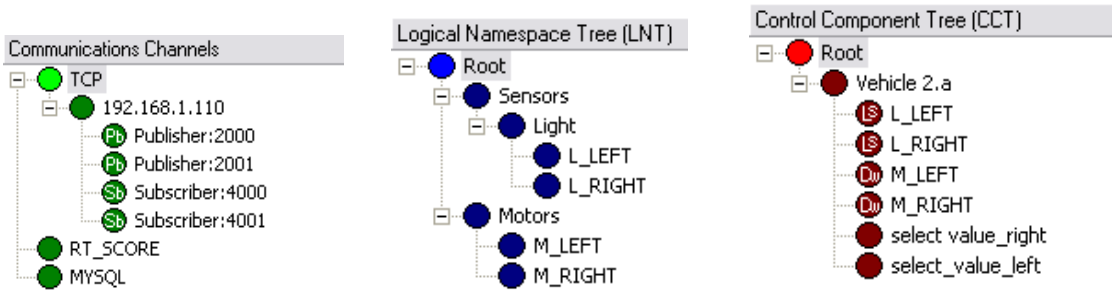


Figura 123. Vehículo 2 de Braitenberg: comunicaciones (a), LNT (b) y LSG (c).

La diferencia entre las dos versiones del vehículo 2 de Braitenberg está en cómo se conectan los datos de los sensores, representados por los *Listener*, a los motores, representados por los *DataWriter*.

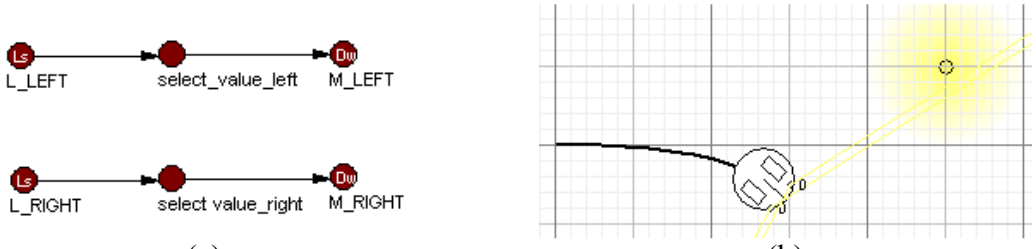
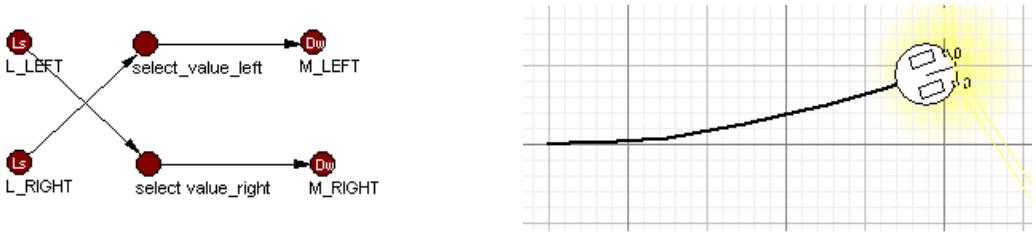


Figura 124. Vehículo 2, versión a, de Braitenberg: LSG (a) y comportamiento (b).

En la primera versión las conexiones son directas (Figura 124.a) produciendo el correspondiente comportamiento de alejamiento de la fuente de luz (Figura 124.b).



(a) (b)

Figura 125. Vehículo 2, versión b, de Braitenberg: LSG (a) y comportamiento (b).

En la segunda versión las conexiones son inversas (Figura 125.a) produciendo el correspondiente comportamiento de alejamiento de la fuente de luz (Figura 125.b).

6.2.3.3 Experimentación

6.2.3.3.1 Escenarios

En el segundo vehículo de Braitenberg se incrementa una fuente de información correspondiente al segundo sensor de luz. En ambos tipos de vehículos se plantean los mismos escenarios que en el caso anterior: sin optimización, optimización a partir de las políticas de QoS y optimización con sensores lógicos de control. En este caso la optimización por medio de sensores lógicos consiste en un sensor que solamente deja pasar aquellos mensajes que hayan variado en contenido con el anterior, y además otro sensor lógico de control que comparar ambos valores y únicamente considera el paso de mensajes en los casos en que la diferencia de valores se considere significativa superando el porcentaje suficiente para que se provoque el giro del vehículo.

6.2.3.3.2 Resultados y discusión

Los resultados del vehículo 2.a para los diferentes escenarios planteados se muestran en la Figura 126.

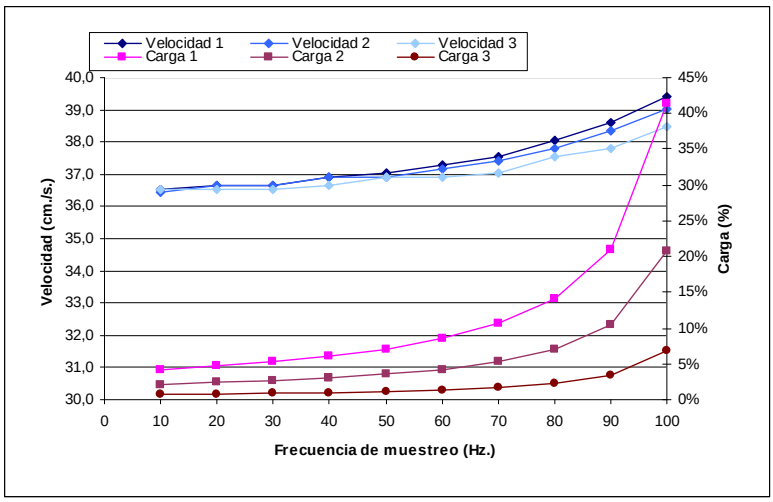


Figura 126. Vehículo 2.a de Braitenberg: frecuencia de muestreo y carga del middleware.

Como se puede apreciar, hay un ligero aumento de la carga con respecto al vehículo 1. Esto se debe a que hay un *Listener* más que recibe información, lo que añade un procesamiento adicional que aumenta la carga.

Los resultados del vehículo 2.b para los diferentes escenarios planteados se muestran en la Figura 127.

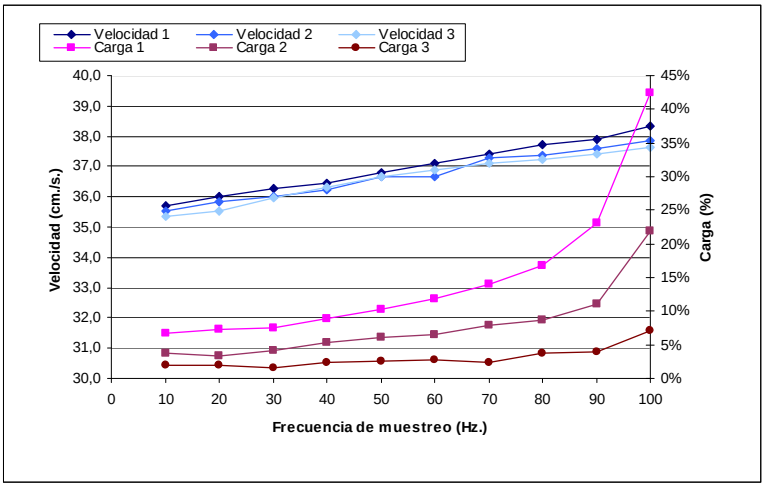


Figura 127. Vehículo 2.b de Braitenberg: frecuencia de muestreo y carga del middleware.

Como se puede apreciar, también aumenta la carga con respecto al escenario del vehículo 1, y ligeramente con respecto al vehículo 2.a. El vehículo 2.b recorre una distancia mayor que el 2.a, por lo que está intercambiando datos durante más tiempo.

El comportamiento general de ambos vehículos es similar. Las variaciones de velocidad en función del escenario son mínimas, lo que hace que la mejora en la carga de control supone una optimización del sistema.

6.2.4 Vehículo 3. Amor

6.2.4.1 Descripción

En los dos vehículos anteriores la señal suministrada al motor era directamente proporcional a la señal obtenida por el sensor. A partir del tercer vehículo la señal puede ser inversamente proporcional a la señal servida por el sensor. El tercer vehículo de Braitenberg tiene tres versiones. Las dos primeras versiones son similares a las del segundo vehículo con la salvedad de que la señal del sensor debe ser invertida antes de ser enviada al motor.

A partir de las versiones de los vehículos de Braitenberg presentadas hasta el momento es posible crear un vehículo donde se disponga de diversos tipos de sensores con la posibilidad de que alimenten a los motores con una señal directa o inversamente proporcional. La tercera versión de los vehículos de Braitenberg combina todo lo visto hasta el momento. En la tercera versión se aumenta el tipo de sensores de los que el robot debe disponer a cuatro: luz, temperatura, oxígeno y materia orgánica. La aportación de cada sensor al motor correspondiente puede ser directa o inversa. Braitenberg no detalla cómo debe ser la combinación de las cuatro señales de cada uno de los tipos de sensores para que produzcan una única señal, este aspecto se deja a la elección del diseñador del control del vehículo.

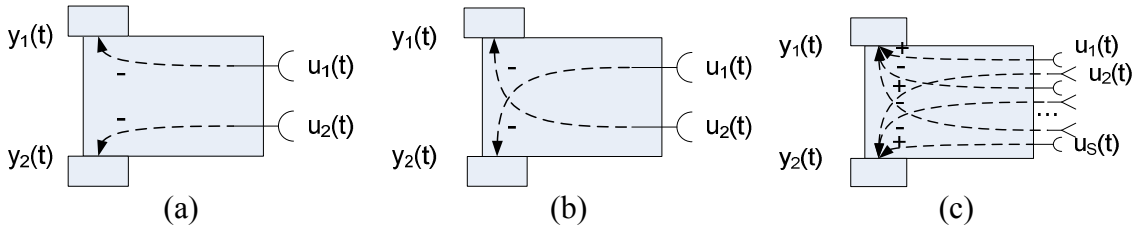


Figura 128. Vehículos 3 de Braitenberg: conexión directa (a), conexión inversa (b) y multisensor (c).

En lo que al control se refiere, la tercera versión de los vehículos de Braitenberg implica tener que añadir dos elementos. El primero debe permitir adaptar la señal del sensor a la del actuador de forma directa o inversa, según se decida. Esta operación se puede realizar por medio de una interpolación lineal sencilla por lo que al sensor lógico de control se le ha llamado "interpolador lineal". El segundo de los elementos debe permitir combinar tantas señales de entrada como sensores deban tenerse en cuenta para el cálculo de la velocidad del motor. La forma más sencilla de realizar esta operación es por medio una función polinomial de grado uno con tantas variables como sensores se tengan. Para realizar la operación se ha implementado un sensor lógico de control llamado "compositor".

6.2.4.2 Implementación

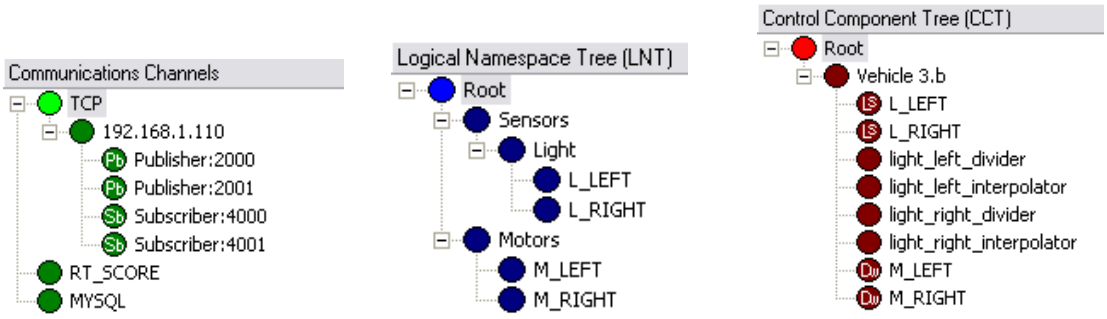
6.2.4.2.1 Simulación

La simulación del tercer vehículo es similar a la del segundo en sus dos primeras versiones, sin embargo la tercera versión hace que el simulador deba proporcionar cuatro tipos de sensores y los correspondientes cuatro tipos de fuentes.

Aunque Braitenberg no entra en los detalles de las fuentes de magnitudes que alimentan los distintos tipos de sensores del vehículo tres en su tercera versión, cabe destacar que la simulación real implica determinar algunos aspectos como la intensidad de la emisión, o el comportamiento de la magnitud frente a los obstáculos. Estos dos aspectos están contemplados en el simulador.

6.2.4.2.2 Control

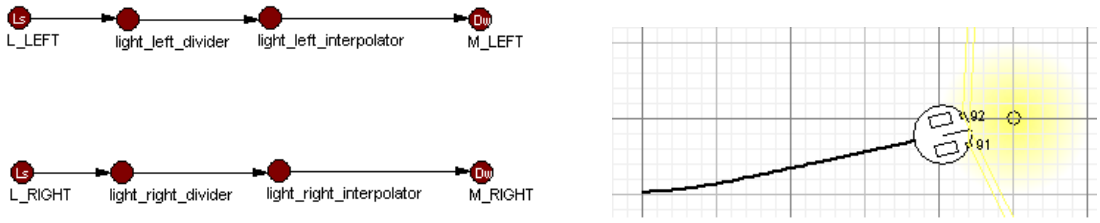
El control del tercer vehículo de Braitenberg en sus dos primeras versiones es muy similar al control de las dos primeras versiones del segundo vehículo. Las comunicaciones y el LNT son idénticos en el segundo y tercer vehículo. El CCT es similar en las dos primeras versiones del tercer vehículo y similar al CCT del segundo con la ampliación del interpolador que permite invertir la relación entre la señal del sensor y el valor enviado al motor.



(a) (b) (c)

Figura 129. Vehículo 3 de Braitenberg: comunicaciones (a), LNT (b) y LSG (c).

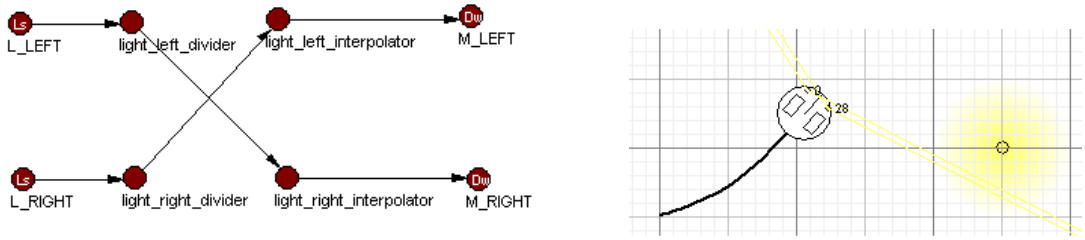
La primera versión del tercer vehículo conecta directamente el *Listener* con el *DataWriter*, con los pasos intermedios correspondientes a la adaptación del mensaje y a la inversión del valor (Figura 130.a). Aunque el comportamiento pueda parecer similar al de la primera versión del segundo vehículo (Figura 125.a), se diferencia en la velocidad de acercamiento a la fuente de luz. En el vehículo 2 la velocidad decrece a medida que se acerca a la fuente, en el vehículo 3 la velocidad aumenta a medida que se acerca a la fuente.



(a) (b)

Figura 130. Vehículo 3, versión a, de Braitenberg: LSG (a) y comportamiento (b).

La segunda versión del tercer vehículo cruza las conexiones entre el *Listener* y el *DataWriter*, con los pasos intermedios correspondientes a la adaptación del mensaje y a la inversión del valor (Figura 131.a). Al igual que en la versión anterior, el comportamiento puede parecer similar al segundo vehículo equivalente, aunque el hecho de que la velocidad del motor aumente de forma inversa a la aportación del sensor, provoca que el vehículo se aleje con mayor velocidad angular.



(a) (b)

Figura 131. Vehículo 3, versión b, de Braitenberg: LSG (a) y comportamiento (b).

En los dos vehículos anteriores se precisa añadir un sensor lógico que transforme la señal de entrada a los valores correspondientes del rango de actuación de los motores. Para ello se emplea un sensor lógico que realiza una interpolación lineal (fórmula 71) entre los valores mínimo y máximo (E_{min} y E_{max} respectivamente) que delimitan el rango de la señal del sensor, y los puntos máximo y mínimo que se proporcionan al

motor (S_{min} y S_{max} respectivamente) y que delimitan el rango de velocidades a las que se puede programar el motor del vehículo. La salida del sensor lógico es $S_{lineal\_interpolator}$ y es la velocidad que se corresponde al valor leído por el sensor correspondiente a la entrada E_i.

$$S_{lineal\_interpolator} = (S_{max} - S_{min}) \cdot (E_i - E_{min}) / (E_{max} - E_{min}) + S_{min} \quad (71)$$

La tercera versión del tercer vehículo de Braitenberg aumenta el número de sensores lógicos necesarios aunque conceptualmente el control es similar a los dos anteriores ya que la conexión y el tratamiento de la información de los sensores sigue la misma filosofía. La tercera versión del tercer vehículo combina todas las versiones de los vehículos vistos hasta el momento (Tabla 13).

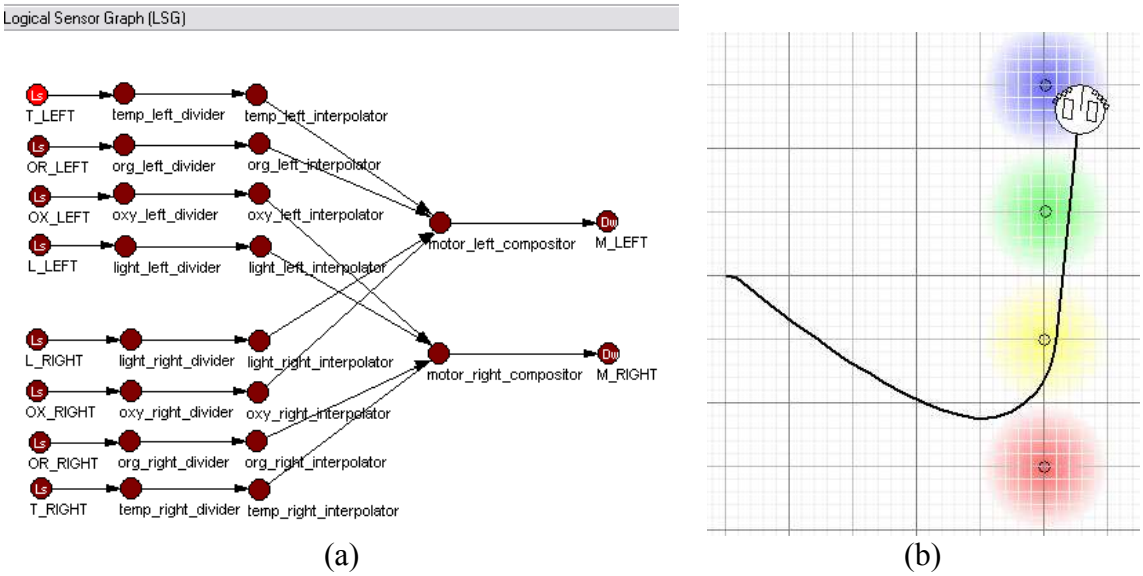
Tabla 13. Combinaciones de las versiones de los vehículos 2 y 3 de Braitenberg

	Conexiones directas entre sensor y motor (versiones 'a')	Conexiones cruzadas entre sensor y motor (versiones 'b')
Aportación positiva del sensor al motor (vehículos 2)	Sensores de temperatura: comportamiento de huida	Sensores de luz: comportamiento de atracción.
Aportación negativa del sensor al motor (vehículos 3)	Sensores de materia orgánica: comportamiento de atracción	Sensores de oxígeno: comportamiento de huida

Debido a que sólo se debe enviar una señal a los motores pero se tienen cuatro fuentes diferentes por motor, las señales de los sensores deben combinarse, lo que da lugar a la otra diferencia con los vehículos anteriores, la aparición del sensor lógico compositor, responsable de ponderar cada uno de los valores de cada sensor físico del vehículo. El comportamiento del sensor lógico compositor responde a la ecuación mostrada en la fórmula 72.

$$S_{compositor} = \sum_{i=1}^{N} K_i \cdot E_{compositor_i} \quad (72)$$

El compositor obtiene la salida S, a partir de la suma de las N entradas E, ponderadas por un factor K específico para cada entrada. El resultado es que la aportación de cada sensor tiene un peso específico.



(a) (b)

Figura 132. Vehículo 3, versión c, de Braitenberg: LSG (a) y comportamiento (b).

El grafo de sensores lógicos que corresponde al control del vehículo 3.c aumenta en complejidad (Figura 132.a). El comportamiento depende de la configuración de la simulación, ya que tanto la disposición de las fuentes (Figura 132.b) como el peso que se le da a cada tipo de estímulo, determinarán la trayectoria de la navegación del vehículo.

6.2.4.3 Experimentación

6.2.4.3.1 Escenarios

El comportamiento de los primeros dos vehículos 3 de Braitenberg son muy similares a los dos primeros vehículos 2. En cambio, el aumento en el número y tipo de sensores del vehículo 3.c de Braitenberg aumenta la complejidad del control. Los dos primeros vehículos comparten escenarios de casos de uso con los vehículos anteriores.

En el vehículo 3.c de Braitenberg, el escenario básico es similar a los anteriores, pero con la salvedad de que la cantidad de *Listener* ha crecido debido a la gran cantidad de sensores de los que se obtiene información. La variación de las políticas de QoS que permite el vehículo es mayor, ya que se puede considerar filtrar con distintos umbrales temporales a diferentes fuentes de datos. En este vehículo cuando el filtrado se realiza con el mismo umbral para todos los *Listener* se considera escenario 2.

El escenario 3 es en el que se agregan los sensores lógicos de control que comparan los pares de valores de los sensores para reenviar únicamente aquellos que tengan una variación significativa y así descargar el control proporcionándole únicamente los valores significativos.

6.2.4.3.2 Resultados y discusión

Para la obtención de datos se han realizado los mismos ensayos que con los vehículos anteriores: obtención de la carga para diferentes frecuencias de muestreo y comprobación de que la velocidad del vehículo no se ve afectada de manera significativa. Los resultados del vehículo 3.a se muestran en la Figura 133.

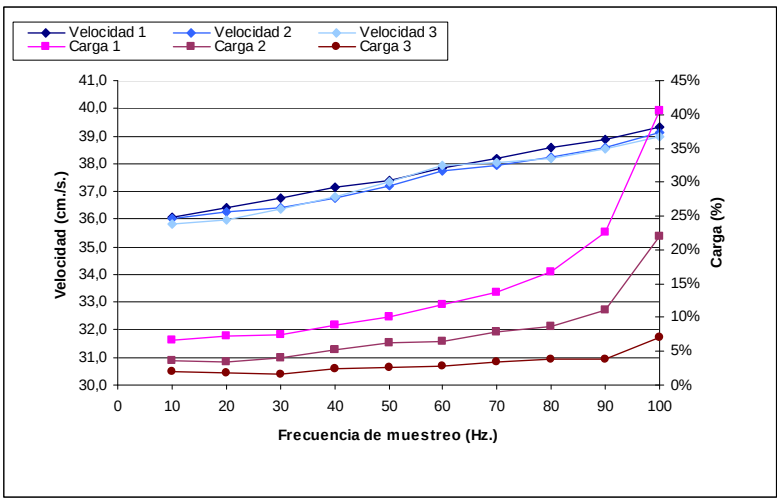


Figura 133. Vehículo 3.a de Braitenberg: frecuencia de muestreo y carga del middleware.

Los resultados del vehículo 3.b se muestran en la Figura 134.

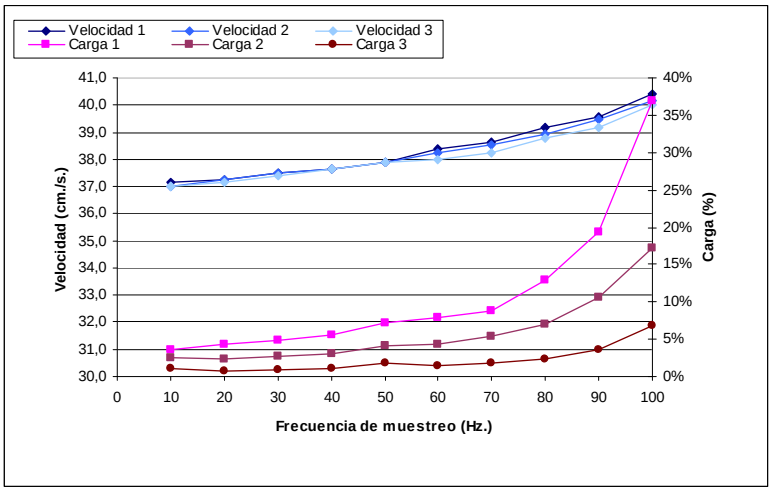


Figura 134. Vehículo 3.b de Braitenberg: frecuencia de muestreo y carga del middleware.

Los resultados del vehículo 3.c se muestran en la Figura 135.

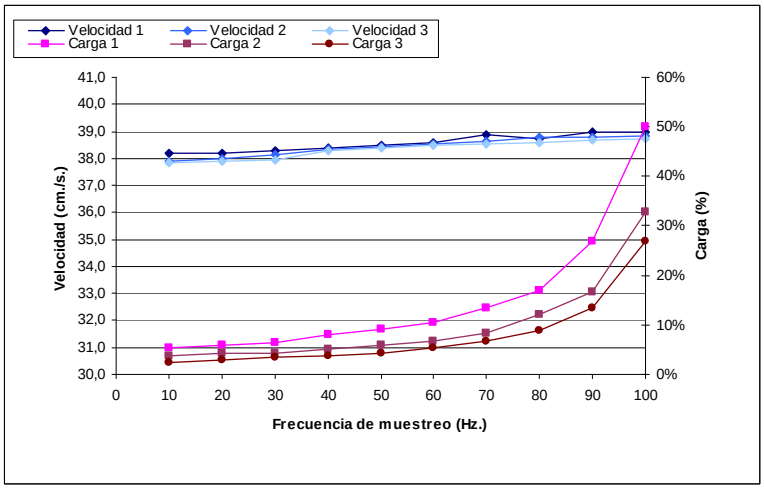


Figura 135. Vehículo 3.c de Braitenberg: frecuencia de muestreo y carga del middleware.

Como se puede observar en las anteriores figuras, los vehículos 3.a y 3.b tienen una optimización similar a los correspondientes vehículos de comportamiento similar: 2.b y 2.a respectivamente.

El vehículo 3.c tiene una carga mayor dado que el control resulta más complejo que los anteriores vehículos. Además se puede comprobar cómo la optimización que aporta la inclusión de sensores lógicos comparadores no mejora sustancialmente la carga. Esto último se debe a que el aumento de la carga que conlleva la inclusión de operaciones más complejas de control diluye el efecto de la mejora introducida en la selección, en función del contenido, de los mensajes que se envían al control.

6.2.5 Vehículo 4. Valores y gustos

6.2.5.1 Descripción

Los vehículos descritos hasta el momento tienen una relación lineal, directa o inversa, entre las entradas, sensores, y las salidas, motores, del vehículo. Sin embargo, cabe la posibilidad de relacionar los sensores y los motores por medio de funciones más complejas. El cuarto vehículo de Braitenberg añade funciones no lineales.

En un primer caso (vehículo 4.a) la función que se considera es una distribución normal que permita centrar el interés del vehículo en un estímulo. En el segundo caso (vehículo 4.b) las funciones que modifican el valor de los sensores son discontinuas. Se pueden crear por medio de combinaciones de sensores lógicos que escogen una u otra fuente en función de si se supera o no un umbral.

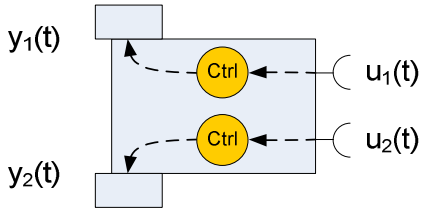


Figura 136. Vehículo 4 de Braitenberg.

6.2.5.2 Implementación

6.2.5.2.1 Simulación

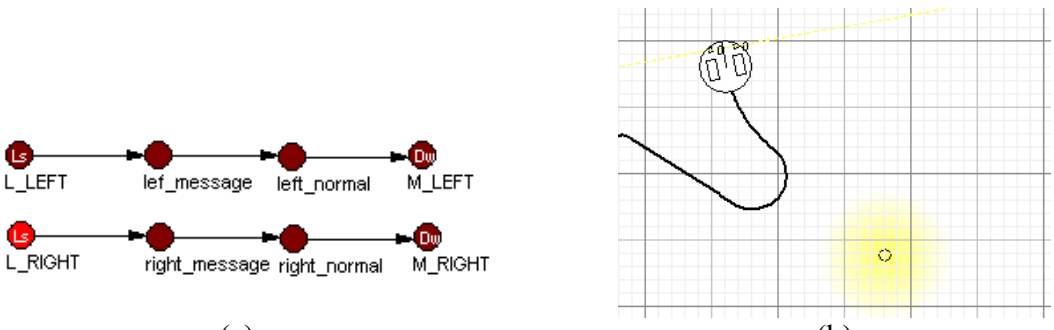
Para la simulación del cuarto vehículo, el entorno puede ser más variable que en los casos anteriores, a excepción del vehículo 3.c. Al igual que este último vehículo, la disposición de más de una fuente de estímulos para los sensores determinará la trayectoria y el correspondiente comportamiento del robot.

6.2.5.2.2 Control

El control del vehículo 4.a es similar a los vistos hasta ahora, con la salvedad de que el sensor lógico que adapta el valor del sensor al motor implementa una función similar a la distribución normal pero optimizada para la implementación en código (fórmula 73).

$$S_{normal\_interpolator} = S_{\max} \cdot \sqrt{(2 \cdot \pi)} \cdot e^{-\frac{1}{2}Z^2}, Z = \frac{\mu - E_i}{\sigma} \quad (73)$$

La implementación del control por medio de los sensores lógicos de interpolación normal se muestra en la Figura 137.a. El hecho de que la función normal pueda ser configurada en función del punto medio de la distribución (μ) o de la varianza de la misma (σ) hace que el movimiento resultante del vehículo sea completamente diferente para un mismo entorno y situación inicial, un ejemplo de trayectoria para este tipo de sensores se muestra en la Figura 137.b.



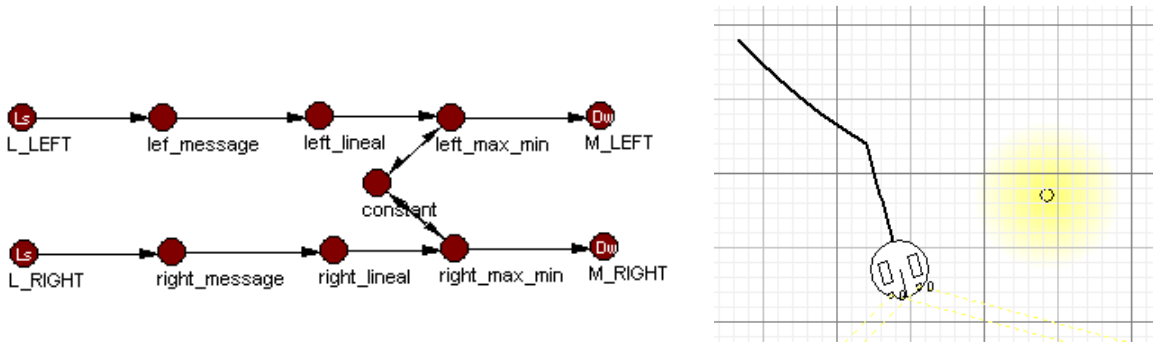
(a) (b)

Figura 137. Vehículo 4, versión a, de Braitenberg: LSG (a) y comportamiento (b).

El vehículo 4.a no añade grandes diferencias con respecto a los vehículos 2 y 3 de Braitenberg. Sin embargo, el vehículo 4.b implementa diferentes funciones de forma

discontinua, lo que implica comportamientos diferentes ante diferentes rangos de valores de los sensores.

La implementación de diversas funciones discontinuas se puede realizar bien por medio de sensores lógicos de control programados al efecto o bien por combinaciones de sensores lógicos de funciones básicas vistas hasta el momento. En este segundo caso, en función de un umbral, se decide si se transmite una u otra función a los motores (Figura 138.a).



(a) (b)

Figura 138. Vehículo 4, versión b, de Braitenberg: LSG (a) y comportamiento (b).

El resultado del movimiento del vehículo 4.b es equivalente a las funciones que lo implementan. Dependiendo del estímulo actúa una u otra función, por lo que parte la trayectoria está determinada por diversas funciones. En el caso del ejemplo se produce un acercamiento a la fuente de luz hasta que, a cierta distancia, deja de actuar el acercamiento y se produce una trayectoria paralela al estímulo (Figura 138.b).

6.2.5.3 Experimentación

6.2.5.3.1 Escenarios

Los vehículos 4.a y 4.b de Braitenberg guardan cierta similitud, en cuanto a la configuración del control, con los equivalentes vehículos 2.a, 2.b, 3.a y 3.b. Esta similitud implica que, para experimentar la relación de la inclusión de las políticas de QoS y de los sensores lógicos de control con la carga de los vehículos, sea conveniente volver a emplear los escenarios sencillos en los que sólo se emplea una luz como fuente de estímulos.

6.2.5.3.2 Resultados y discusión

Los resultados obtenidos para el vehículo 4.a se muestran en la Figura 139.

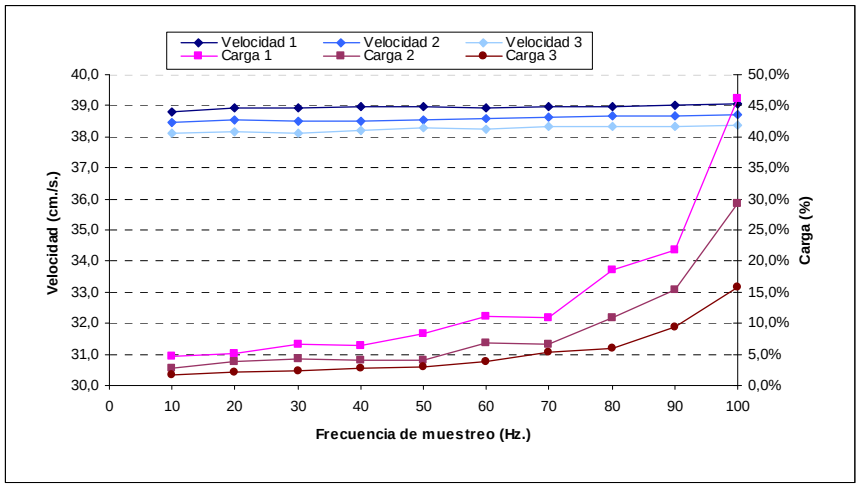


Figura 139. Vehículo 4.a de Braitenberg: frecuencia de muestreo y carga del middleware

Los resultados obtenidos para el vehículo 4.b con las dos funciones descritas anteriormente se muestran en la Figura 140.

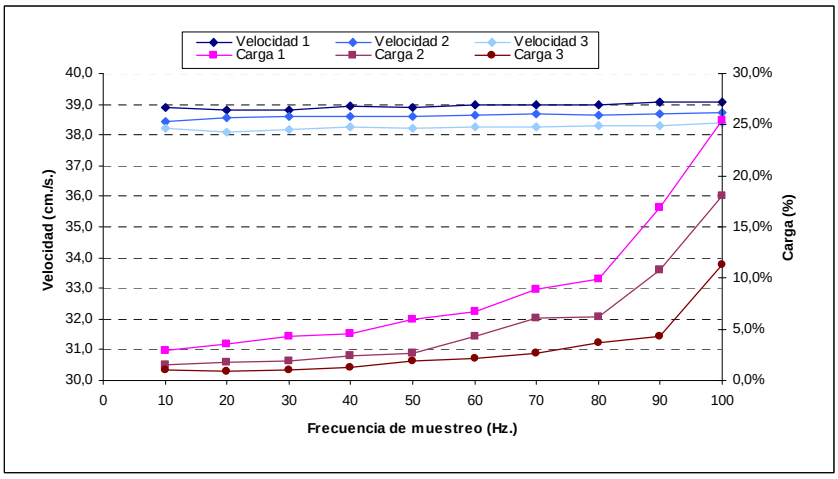


Figura 140. Vehículo 4.b de Braitenberg: frecuencia de muestreo y carga del middleware.

Como se puede observar en la Figura 139, el aumento de complejidad de la función de control aumenta la carga del componente de navegación. Esto implica que la aportación a la optimización del sensor lógico de control, que selecciona las diferencias entre los valores de entrada de los diferentes sensores, es escasa.

En el segundo de los casos en los que se emplean vehículos con diferentes funciones de control los resultados indican que la aportación del control no es tan significativa como el uso de las políticas de QoS. Sin embargo, la cantidad de vehículos que se pueden implementar bajo la denominación de vehículo 4.b de Braitenberg es tan elevada que no es conveniente comparar tan solo uno de ellos, ya que los resultados dependen completamente de la implementación que se haya realizado.

Los vehículos 4 de Braitenberg son vehículos donde aparece un aumento en el procesamiento del control debido al aumento en la complejidad de los algoritmos a aplicar. El aumento del protagonismo en el control hace que la inclusión de sensores lógicos de control influya en menor medida que en los vehículos donde el procesamiento era más sencillo.

6.3 Gestión dinámica basada en políticas de QoS

6.3.1 Escenario

En las pruebas anteriores las modificaciones estaban orientadas a optimizar el control por medio de modificaciones de sensores lógicos de control. Las modificaciones han consistido en la incorporación, en el middleware, de políticas de QoS y de elementos de control que optimicen el uso de los recursos del nodo de control. Sin embargo, el control también puede optimizarse si se actúa sobre el emisor de la información, lo que optimizará el uso del canal de comunicaciones. En este sentido se plantea el cuarto escenario donde se experimentará cómo la arquitectura FSACtrl, basándose en el modelo de publicación-suscripción, permite configurar las comunicaciones en función de los parámetros de eficiencia en el control. Este escenario se muestra en la Figura 141.

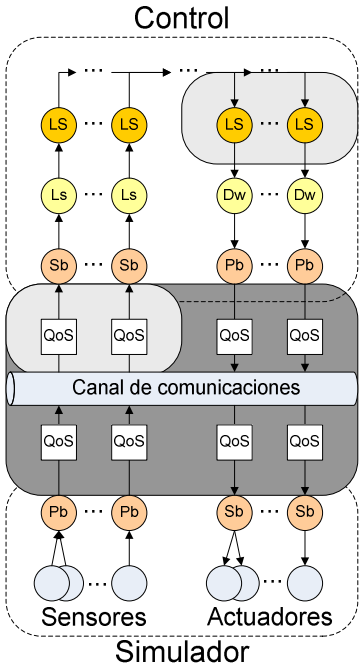


Figura 141. Escenario de la experimentación del vehículo 5.

El escenario incorpora la variación de las condiciones de las comunicaciones en función de las políticas de QoS tanto del emisor como del receptor. Este escenario permite, entre otras cosas, comprobar cómo el ajuste dinámico de las comunicaciones optimiza el control. También permite la operación complementaria, es decir, optimizar el uso de las comunicaciones en función de los requisitos del control.

Para comprobar si el ajuste dinámico descrito anteriormente optimiza el control o las comunicaciones, resulta adecuado el empleo de dos comportamientos complementarios. La complementariedad de los comportamientos debe centrarse en la necesidad de recursos de control y de comunicaciones. Esto último se logra por medio del vehículo 5 de Braitenberg en el que se agrega el concepto de lógica que permite seleccionar unas operaciones u otras en función de criterios básicos.

6.3.2 Vehículo 5. Lógica

6.3.2.1 Descripción

El quinto vehículo de Braitenberg añade una secuenciación lógica de acciones por medio de unos elementos que, a modo de puertas lógicas, se accionan a partir de umbrales de señales. Las señales pueden ser inhibitorias cuando evitan que la señal se propague o excitadoras en caso contrario. Por medio de la secuenciación y la retroalimentación de estos elementos se implementa una memoria básica en el vehículo que es capaz de contar eventos para reaccionar en función de la cantidad o del peso que se le puede otorgar a los eventos. Teóricamente a partir de la combinación de este tipo de elementos es posible desarrollar sistemas con un comportamiento complejo a partir de elementos muy básicos.

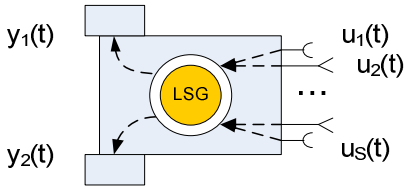


Figura 142. Vehículo 5 de Braitenberg.

En el caso del escenario planteado, se propone un vehículo 5 de Braitenberg con dos comportamientos reactivos básicos: seguimiento de rumbo y evitación de obstáculos. Estos dos comportamientos están implementados por medio de sendos componentes de control. La combinación de los dos comportamientos la implementa un tercer componente que pondera ambas señales.

La gestión dinámica del control se realiza mediante el ciclo integral de la calidad descrito en el apartado 4.3, adaptado al contexto del vehículo 5 (Figura 143).

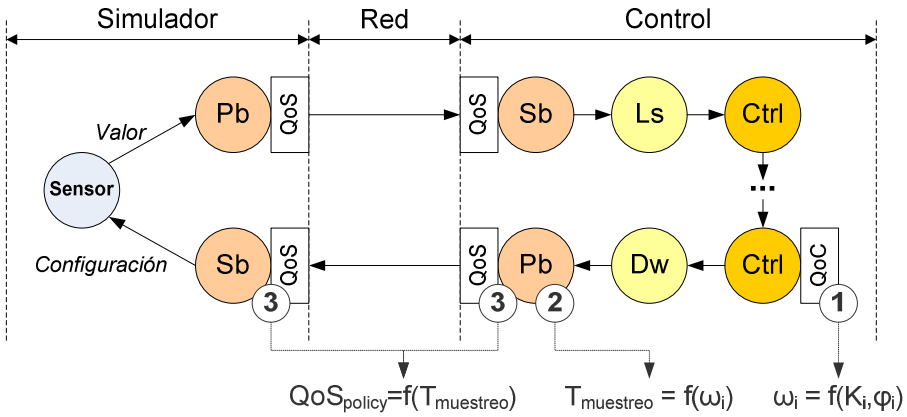


Figura 143. Ciclo integral de la calidad implementado.

En la Figura 143 se numeran los pasos correspondientes a la gestión dinámica de las comunicaciones y del control basada en las políticas de QoS. La gestión comienza por la determinación de la velocidad angular máxima (punto 1) del vehículo ya que la figura se corresponde con el comportamiento de seguimiento de rumbo. En función de la velocidad angular máxima se determina la frecuencia de muestreo necesaria para que el correspondiente cambio de velocidad sea seguro. La seguridad se entiende como que la siguiente medición se obtendrá a tiempo.

En función de los valores de la velocidad obtenidos en los comportamientos se determina cómo debe ser la frecuencia de muestreo (punto 2). Finalmente, a partir de las políticas de QoS y de las condiciones de la red se pasa a negociar la frecuencia de muestreo del sensor (punto 3).

6.3.2.2 Implementación

6.3.2.2.1 Simulación

En la simulación, la cantidad de sensores que se pueda dotar a un vehículo dependerá de las necesidades de procesamiento que el robot tenga. Dado que el vehículo 5 de Braitenberg puede ser tan complejo como elementos de procesamiento se tengan el simulador debe proveer de la mayor cantidad de sensores posibles.

Braitenberg no define una funcionalidad concreta para el vehículo 5 por lo que la implementación de los comportamientos dependerá de la funcionalidad que se desee dar al vehículo.

Para el vehículo simulado, se implementan dos comportamientos: seguimiento de rumbo y evitación de obstáculos. Cada comportamiento precisa de un tipo de sensor especializado. El seguimiento de rumbo precisa de un sensor de orientación que, a modo de brújula y a partir de una baliza, proporciona el ángulo relativo del vehículo con la localización de la baliza. El comportamiento de evitación de obstáculos precisa de un anillo de sensores que proporcionen la distancia con los obstáculos del entorno que el vehículo pueda encontrarse a lo largo de la navegación.

El entorno de simulación en este robot, y los siguientes, debe ser tan complejo como complejo sea el comportamiento que se requiera del vehículo. En el caso del simulador es posible dotar al entorno de una gran cantidad de estímulos así como de obstáculos que deberían ser detectados y evitados por los vehículos.

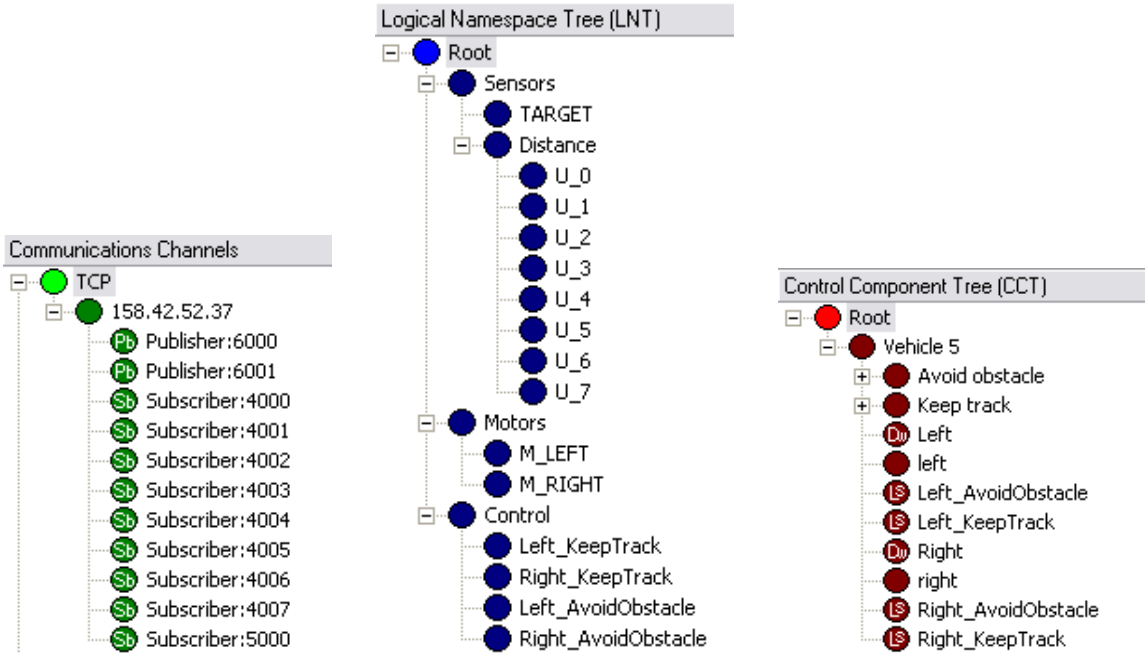
En el caso del vehículo 5, el entorno de experimentación se compondrá de un recinto cerrado y una baliza. El vehículo deberá acudir a la baliza evitando los obstáculos que pueda encontrar en su camino.

6.3.2.2.2 Control

Los sensores lógicos de control de la arquitectura FSACtrl son perfectos para la implementación de los elementos que describe Braitenberg para el vehículo 5. Las posibilidades en cuanto a comportamientos que es posible implementar a partir de estos elementos son muy amplias. Por ejemplo, es posible implementar contadores sencillos que activan la recarga de la batería de un vehículo o la inhibición de un comportamiento de ir a un objetivo para poder evitar un obstáculo. Además, la posibilidad de incluir sensores lógicos de control dentro de componentes de control, y a su vez, estos componentes de control dentro de otros componentes, permite jerarquizar los comportamientos de los vehículos para priorizar unos sobre otros o combinarlos entre ellos.

Para las simulaciones se ha empleado un canal TCP/IP al que se ha asociado un *Publisher* por cada sensor y un *Subscriber* por cada motor. Los datos necesarios para el control del vehículo 5 de Braitenberg se muestran en el LNT de la Figura 144.a. Se han organizado en tres áreas: sensores (nodo *Sensors*), actuadores (nodo *Motors*) y datos internos de intercambio entre componentes (nodo *Control*).

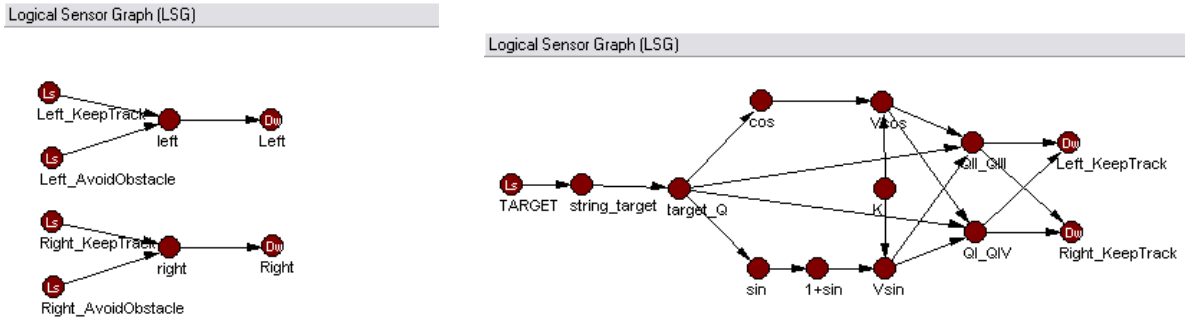
Los sensores se han organizado en dos grupos, uno con el sensor que define el objetivo para el comportamiento de seguimiento de rumbo y otro para el anillo de sensores que proporcionan las distancias al comportamiento de evitación de obstáculos (Figura 144.b).



(a) (b) (c)

Figura 144. Vehículo 5 de Braitenberg: LNT (a), comunicaciones (b) y CCT (c).

El esquema de la jerarquía de componentes de control, o CCT, se muestra en la Figura 144.c. El componente principal es el que representa el control de todo el vehículo, etiquetado como "*Vehicle 5*". Este componente de control contiene otros dos componentes que representan cada uno de los comportamientos: el comportamiento de evitación de obstáculos o *Avoid obstacles* (AO) y el comportamiento de seguimiento de rumbo *Keep track* (KT). Además de estos componentes de control, el componente principal (*Vehicle 5*) también tiene sensores lógicos de control responsables de hacer la composición de los dos comportamientos principales (Figura 145.a).



(a) (b)

Figura 145. Vehículo 5 de Braitenberg: LSG del composición de comportamientos (a) y del comportamiento de seguimiento de rumbo (b).

En la Figura 145.b se muestra el esquema del grafo lógico de sensores de control correspondientes al comportamiento de seguimiento de rumbo. Este sensor lógico de

control calcula el ángulo que debe tomar el vehículo por medio de las fórmulas para determinar la velocidad de cada rueda en función del ángulo relativo que forma la trayectoria actual del robot con la línea que lo une al objetivo (Figura 146).

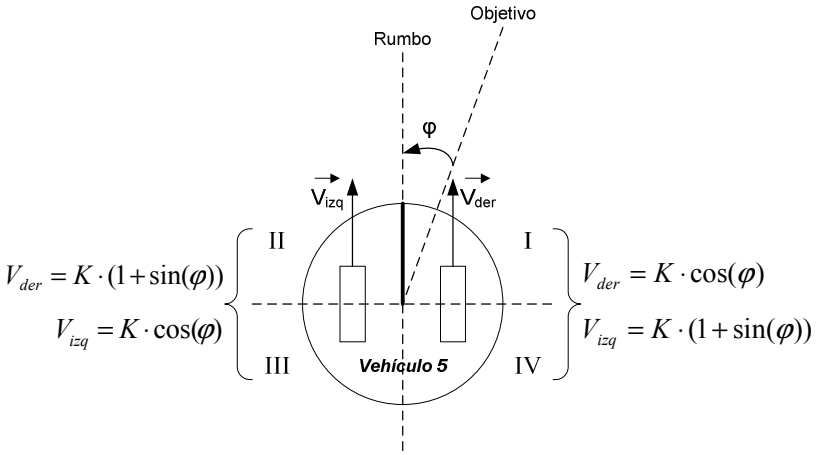


Figura 146. Comportamiento de seguimiento de rumbo del vehículo 5.

El gráfico lógico de sensores correspondiente al comportamiento de evitación de obstáculos se muestra en la Figura 145. Este comportamiento se ha implementado mediante sensores lógicos de control de operaciones básicas y se corresponde con las fórmulas que ponderan cada uno de los rayos del anillo de sensores de distancia.

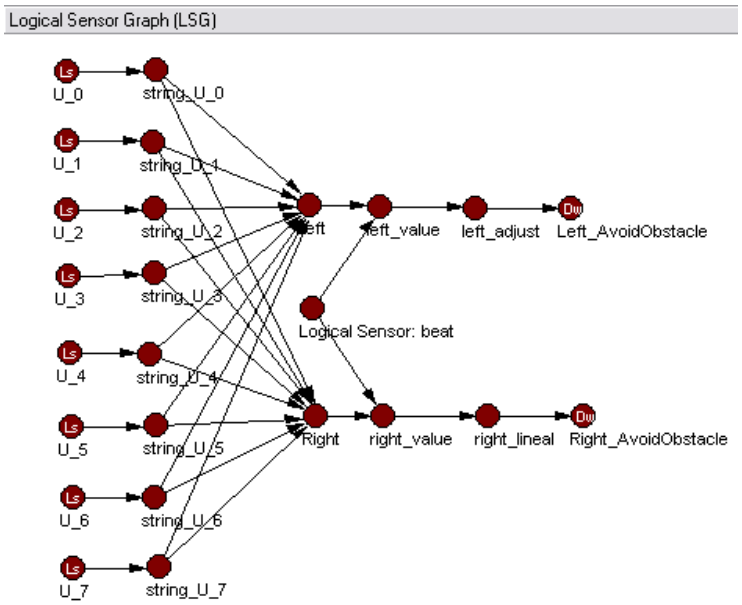


Figura 147. Vehículo 5 de Braitenberg: LSG del comportamiento de evitación de obstáculos.

Las fórmulas se corresponden con las mostradas en la Figura 148 en las que la medición de cada sensor es ponderada por un factor, que depende de cada implementación. La velocidad del motor se obtiene por medio de la suma de todas las aportaciones de los sensores.

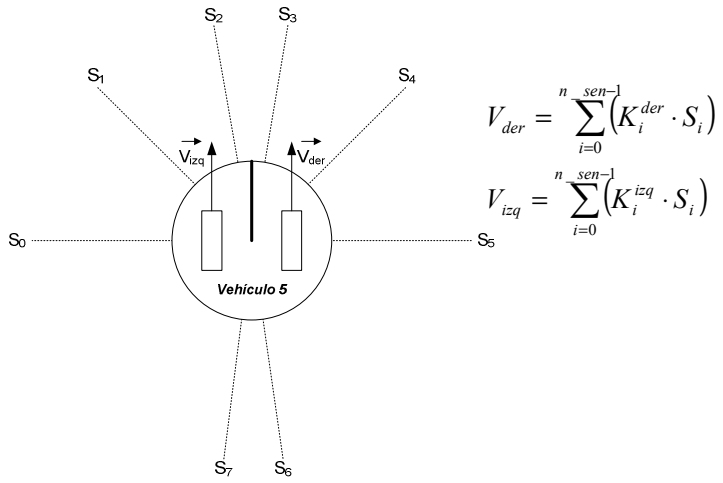


Figura 148. Comportamiento de evitación de obstáculos del vehículo 5.

Cabe destacar la diferencia de implementación de los dos comportamientos que configuran el vehículo 5. El comportamiento de seguimiento de rumbo trabaja con un solo dato de origen, por lo que no se precisa tener en cuenta para las operaciones la sincronización temporal de los datos. Sin embargo, el comportamiento de evitación de obstáculos precisa de una fuente de datos por sensor, por lo que se debe tener en cuenta la antigüedad de cada una de las fuentes.

6.3.2.2.3 Obtención de los parámetros de calidad de control

Los parámetros de calidad de control son dependientes del error de control que, a su vez, dependen del comportamiento controlado. En el caso de los robots móviles experimentados se obtendrán los parámetros de calidad de control para cada comportamiento. Para la obtención de los parámetros de calidad de control de la navegación, resultan convenientes los parámetros IAE y ITAE, siendo el primero el que proporcionará la calidad del control general de toda la trayectoria seguida, mientras que el segundo proporcionará la calidad de control inmediata.

En el caso del comportamiento de seguimiento de rumbo, el error de control depende de la diferencia entre el ángulo del rumbo y el ángulo en el que se encuentra el objetivo. Los parámetros correspondientes se muestran en las ecuaciones 74 y 75.

$$IAE = \int_{t_{inicial}}^{t_{final}} \left[\varphi_{sensor}(t) - \varphi_{referencia}(t) \right] \cdot dt \tag{74}$$

$$ITAE = \int_{t_{inicial}}^{t_{final}} t \cdot \left[\varphi_{sensor}(t) - \varphi_{referencia}(t) \right] \cdot dt \tag{75}$$

En el caso del comportamiento de evitación de obstáculos, el error se obtiene por la combinación de las distancias obtenidas por cada sensor. De la misma forma que los valores de los motores dependen de una ponderación del valor de cada uno de los sensores, la calidad del control dependerá de la combinación de cada una de las calidades de control de cada sensor. En este caso los parámetros de calidad de control IAE y ITAE se obtienen por medio de las ecuaciones 76 y 77-

$$IAE_{sensor(i)} = \int_{t_{inicial}}^{t_{final}} \left[d_{sensor}(t) - d_{seguridad} \right] \cdot dt \quad (76)$$

$$ITAE_{sensor(i)} = \int_{t_{inicial}}^{t_{final}} t \cdot \left[d_{sensor}(t) - d_{seguridad} \right] \cdot dt \quad (77)$$

En el caso concreto del comportamiento de seguimiento de rumbo, la ponderación empleada es la misma que se emplea para la toma de decisiones de la velocidad (Figura 148)

6.3.2.3 Experimentación

En los escenarios de las pruebas realizadas sobre los vehículos del 1 al 4, se podía comprobar cómo las políticas de QoS y los sensores lógicos de control de la arquitectura FSACtrl se podían emplear para optimizar el funcionamiento del vehículo. El escenario en el que se desenvuelve el vehículo 5, demuestra cómo, además de las optimizaciones anteriores, es posible adecuar el rendimiento de las comunicaciones a los requisitos de funcionamiento del vehículo.

6.3.2.3.1 Variación de la carga de las comunicaciones

La carga de las comunicaciones depende del tráfico de red, y este tráfico depende de la frecuencia de muestreo a la que se leen los sensores de manera que a mayor frecuencia de muestreo más tráfico de red. La velocidad máxima a la que el vehículo puede navegar depende también de la frecuencia de muestreo, de manera que cuanta más frecuencia de muestreo se tenga, más pronto es posible detectar el obstáculo y por tanto adaptarse antes a la situación (fórmula 78).

$$V_{\max} = d_{sensor} / T_{muestreo} \quad (78)$$

La relación mostrada anteriormente da lugar a la representación gráfica mostrada en la Figura 149.

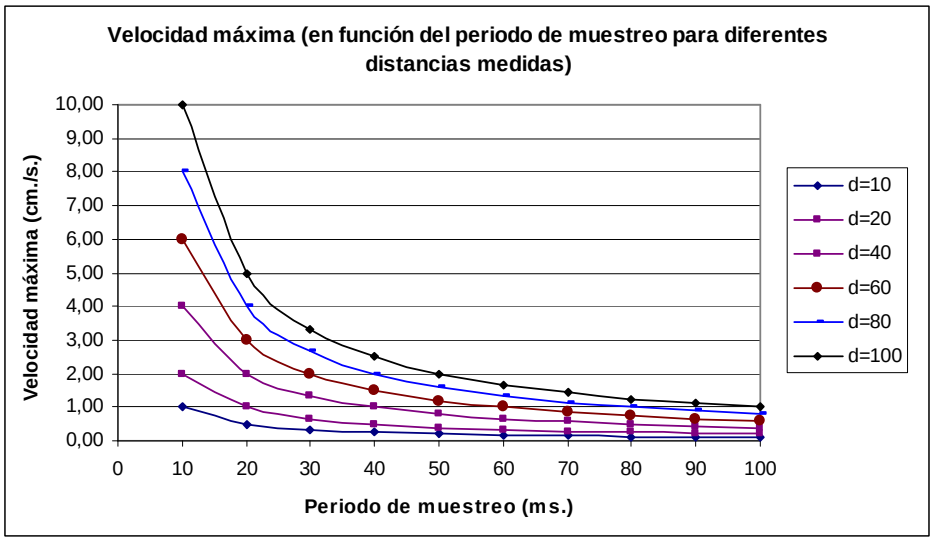


Figura 149. Relación entre la velocidad máxima y el periodo de muestreo para diferentes mediciones de los sensores de distancia.

El error cometido en el control del comportamiento de evitación de obstáculos se basa en la maximización de las distancias a los obstáculos. Sin embargo estas distancias dependen de la localización de los sensores en el vehículo por lo que el cálculo del error en el control depende de las características del entorno.

Para el comportamiento de seguimiento de rumbo, la velocidad máxima no está relacionada con el ángulo que proporciona el parámetro del sensor. Sin embargo, sí que hay una relación entre el error (empleado como parámetro de QoC) del control que realiza el comportamiento de seguimiento de rumbo y el periodo de muestreo del sensor (Figura 150).

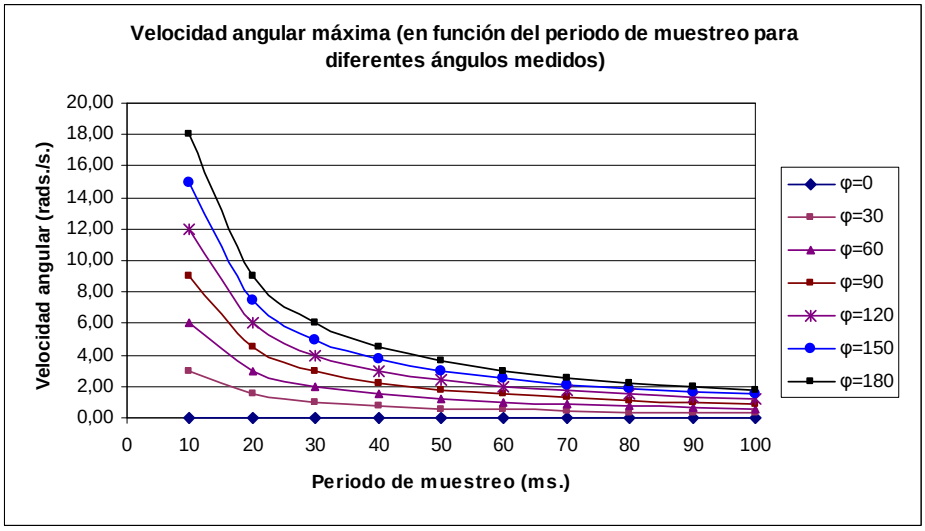


Figura 150. Relación entre la velocidad máxima y el periodo de muestreo para diferentes mediciones de los sensores de distancia.

Esta relación se debe a que cuando el vehículo lleva el rumbo adecuado, apenas se debe variar la velocidad angular mientras que cuando el vehículo está alejado del rumbo se debe aumentar la velocidad angular para acercarse al rumbo adecuado. En este caso, un periodo de muestro alto permite una mayor velocidad angular lo que implica acercarse

al rumbo adecuado antes que con un periodo de muestreo bajo. La relación entre la velocidad angular y el periodo de muestreo, para un ángulo relativo concreto, se muestra en la fórmula 79.

$$\omega_{\max} = \varphi_{sensor} / T_{muestreo} \tag{79}$$

A partir de las fórmulas 78 y 79 se puede determinar el periodo de muestreo de los sensores de los que obtienen la información para sus cálculos.

El primer paso en el ajuste dinámico del control del vehículo a las condiciones de comunicaciones consiste en determinar el error cometido y empleado para la obtención de los parámetros de QoC. Se considera una optimización del control el decremento del ángulo en el comportamiento de seguimiento de rumbo y el aumento de las distancias detectadas, ponderado por la posición del sensor, en el comportamiento de evitación de obstáculos.

A partir del valor del error se debe determinar la acción de control. Esta acción de control proporciona la velocidad máxima a aplicar en los motores del vehículo. Sin embargo, como se ha mostrado en la Figura 149 y en la Figura 150, esta velocidad máxima implica, a su vez, un periodo de muestreo determinado para poder obtener el siguiente mensaje a tiempo suficiente para poder reaccionar de forma segura.

El periodo de muestreo tiene una cota inferior consistente en el tiempo de servicio, no se debe proporcionar más mensajes de los que el elemento de control puede procesar (80).

$$T_{servicio} \leq T_{muestreo} \tag{80}$$

A partir del ciclo integral de la calidad (uso de la QoC y de la QoS conjuntamente) se optimiza el periodo de muestreo del sensor en función de la velocidad requerida de los motores.

En el caso de que el tráfico de red sea una restricción, se realiza el proceso inverso, es decir, se adecua la acción de control (velocidad de los motores) en función de los requisitos de tráfico de la red.

Para configurar el periodo de muestreo del sensor se emplean las políticas de QoS *Deadline* y *TimeBasedFilter*, de manera que delimitan el máximo y mínimo periodo de muestreo según la fórmula 81.

$$TimeBasedFilter_{minimum\_separation} = T_{servicio} \leq T_{muestreo} = Deadline_{duration}\ T_{muestreo} \tag{81}$$

El ajuste de las políticas de QoS lo realizan los correspondientes *publisher* y *subscriber* correspondientes del simulador del vehículo y de la aplicación de control. Además de las relaciones anteriores hay más restricciones que intervienen en los comportamientos del vehículo, ya que el aumento de la frecuencia de muestreo implicará un aumento de la demanda de servicio y, consecuentemente, de la carga del componente de control. Por tanto, la carga de las comunicaciones o la carga del componente de control serán la restricción que determinará la cota inferior de la frecuencia de muestreo del sensor.

6.3.2.3.2 Análisis de la carga

Para comprobar cómo afecta la variación dinámica de la configuración de las políticas de QoS se ha probado el vehículo 5 en un mismo entorno con diferentes escenarios. En el primero de los casos se realiza la navegación de un punto a otro a periodos de muestreo fijos. En el segundo de los casos se realiza la navegación entre los mismos

puntos a un periodo de muestreo variable. Los parámetros que se comparan para cada escenario son la velocidad de la navegación, la carga del componente de control y la carga de la red.

Las diferencias de la navegación del vehículo entre distintos periodos de muestreo fijos son apreciables a simple vista. En la Figura 151 se muestra la trayectoria del vehículo con un periodo de muestreo alto que implica una baja actualización de la acción de control y por tanto una reacción más tardía, tanto en la orientación hacia el rumbo como en la evitación del obstáculo.

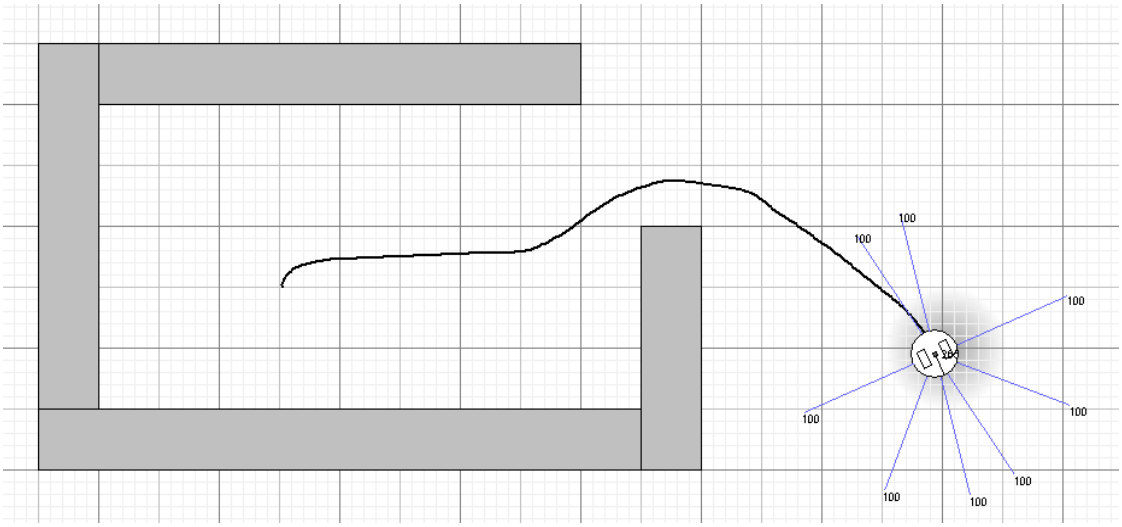


Figura 151. Navegación del vehículo 5 con un periodo de muestreo alto.

A consecuencia de la baja actualización el vehículo va más lento de lo que podría ir teniendo el mismo margen de seguridad. En la Figura 152 se muestra la trayectoria seguida por el mismo vehículo, con los mismos componentes de control, cuando el periodo de muestreo es bajo y consecuentemente el vehículo reacciona antes para orientar el rumbo o para evitar el obstáculo.

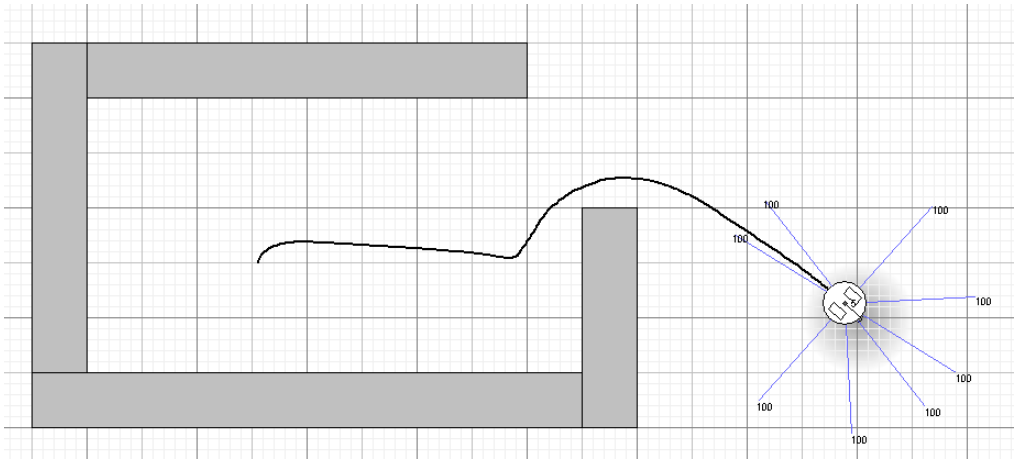


Figura 152. Navegación del vehículo 5 con un periodo de muestreo bajo.

A continuación, en la Figura 153, se muestran la carga de red y la carga del control en función del periodo de muestreo que se aplique. La evolución de la carga de la red es lineal y está escalada para hacerla comparable con la del control de manera que un valor de 1 se corresponde con 100 mensajes por segundo. La carga del control es la definida

en las anteriores pruebas, de forma que un valor de 1 implica un componente de control cargado al 100%. Al igual que en las anteriores pruebas, el tiempo de servicio de los sensores lógicos de control ha sido escalado para poder apreciar gráficamente las variaciones de la carga. En la Figura 153 también se muestra cómo evoluciona la velocidad que el vehículo emplea para alcanzar el objetivo.

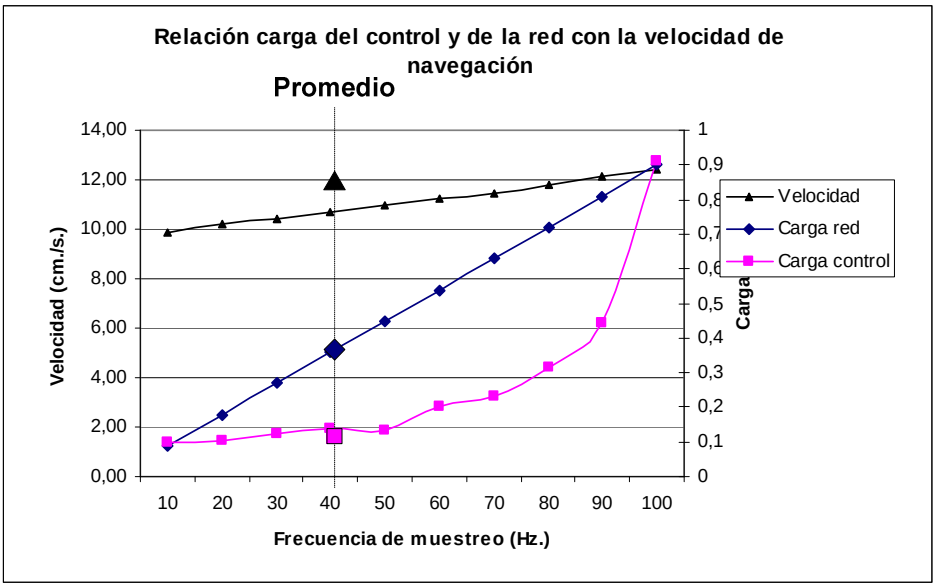


Figura 153. Relación de la carga y eficiencia en la navegación del vehículo 5.

Como se puede observar, la carga de la red evoluciona de manera lineal a medida que aumenta la frecuencia de muestreo de los sensores. Esta evolución es la que cabe esperar de un sistema de comunicaciones basado en el protocolo TCP/IP. La carga del control también sigue la evolución esperada y es similar a los componentes de control de los vehículos anteriores. La velocidad aumenta a medida que aumenta la frecuencia de muestreo, ya que el vehículo puede reaccionar antes y al minimizar el error de control orienta el rumbo o evita los obstáculos más rápidamente a medida que tiene disponibles nuevas mediciones en un intervalo de tiempo menor. Este aspecto se mostraba gráficamente en la Figura 149 y la Figura 150.

El resultado de la navegación con frecuencia de muestreo variable se muestra como puntos resaltados. Es posible observar cómo se optimiza la velocidad de navegación para el valor que cabría esperar a la frecuencia de muestreo promedio. Esto se debe a que las frecuencias altas de muestreo se dan en las maniobras en las que el vehículo necesita datos con mayor precisión para realizar las maniobras de forma segura. En el caso del comportamiento de seguimiento de rumbo esto se da cuando el vehículo no está orientado hacia el objetivo.

6.3.2.3.3 Evolución temporal de la frecuencia de muestreo

Para comprobar el efecto de la variación de la frecuencia de muestreo, se ha capturado el valor autoconfigurado a largo del tiempo de navegación del vehículo (Figura 154).

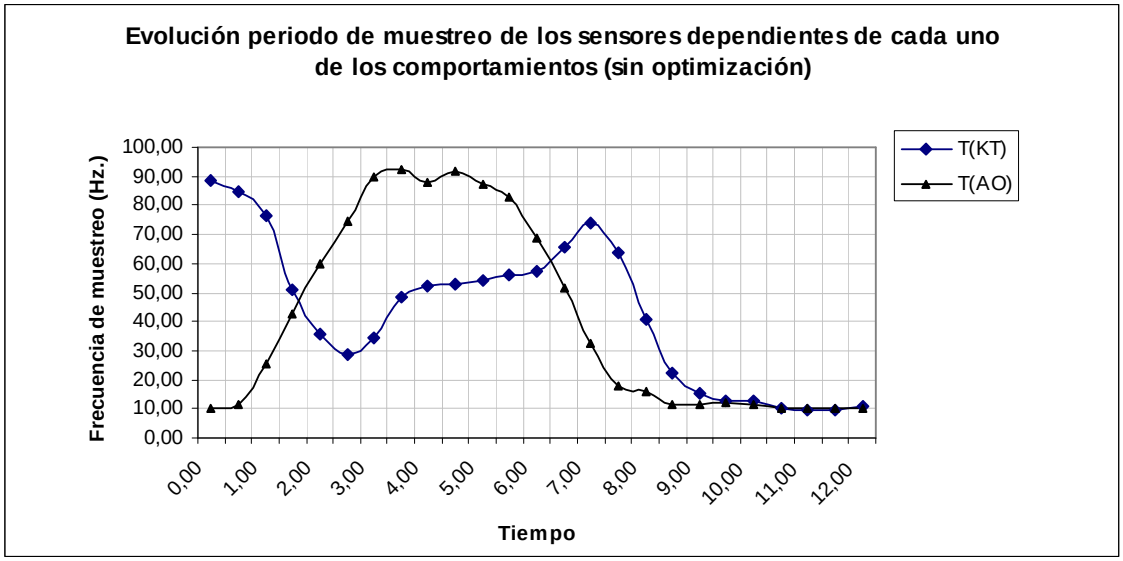


Figura 154. Evolución de la frecuencia de muestreo a lo largo de la navegación del vehículo 5.

Como se aprecia en la Figura 154, la frecuencia de muestreo inicial del comportamiento de evitación de obstáculos (AO) es baja ya que las distancias detectadas por los sensores son lejanas. A medida que se detecta la primera pared la frecuencia aumenta para permitir el giro más rápido. Los momentos de mayor frecuencia de muestreo son los giros que se producen a la salida del recinto cerrado: el giro de la primera esquina, que se produce entre los instantes 3 y 4 de la navegación, y el giro de la segunda entre esquina, que se produce entre los segundos 4 y 5 de la navegación. A partir de ese instante la frecuencia va disminuyendo gradualmente al no detectarse obstáculos.

En el caso del comportamiento de seguimiento de rumbo (KT), sucede algo similar, ya que inicialmente el vehículo tiene una gran deriva con respecto al rumbo. Una vez se ha orientado la frecuencia de muestreo necesaria para corregir los pequeños errores es baja. Sin embargo, al detectar un obstáculo, el vehículo gira para evitarlo, lo que aumenta el error.

El aumento de la frecuencia de muestreo en el comportamiento de evitación de obstáculos se debe a la detección de un obstáculo. A consecuencia de la detección del obstáculo, se produce un cambio de rumbo debido a las acciones correspondientes a la evitación del mismo. Este cambio de rumbo produce un aumento del error de control del comportamiento de seguimiento de rumbo que, a su vez, produce un aumento de la frecuencia de muestreo del sensor de rumbo. Sn embargo, ese aumento de frecuencia de muestreo no se traduce en una mejora de la orientación ya que la evitación del obstáculo es prioritaria a la orientación de rumbo.

Para optimizar el caso anterior, se programa la frecuencia de muestreo de manera prioritaria, de forma que aunque el error de control en el rumbo sea grande, si se esta detectando un obstáculo la frecuencia de muestreo del sensor de rumbo permanece en el valor mínimo. Los resultados se muestran en la Figura 155.

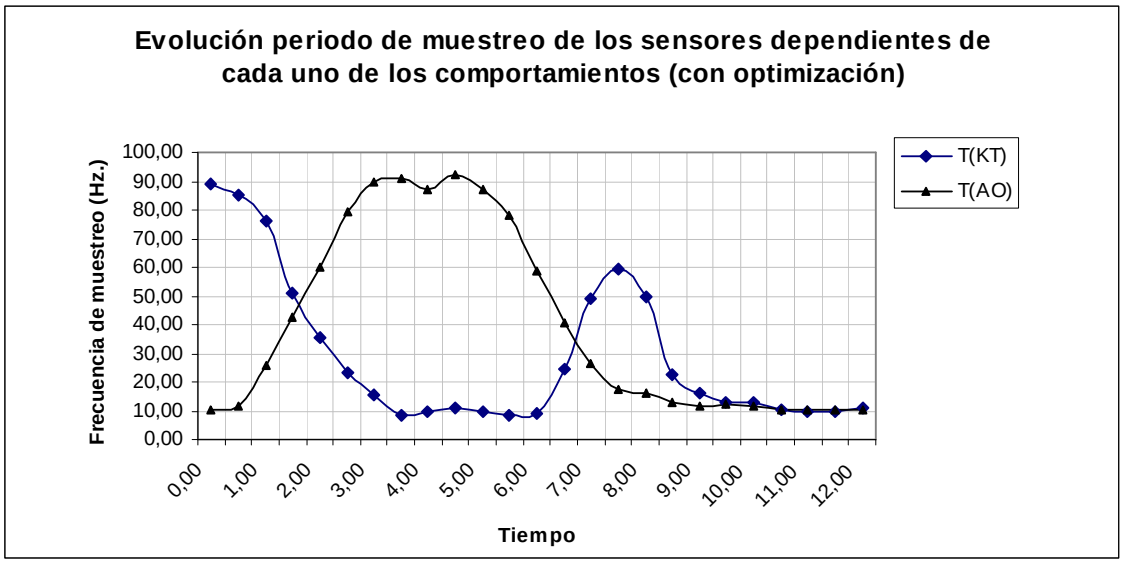


Figura 155. Evolución de la frecuencia de muestreo a lo largo de la navegación del vehículo 5.

En este caso, la posibilidad de priorizar comportamientos hace que mejore ostensiblemente la frecuencia de muestreo del comportamiento de seguimiento de rumbo. Una baja frecuencia de muestreo en el comportamiento de seguimiento de rumbo no es problemática, mientras el vehículo está evitando un obstáculo, ya que actúa sólo de recordatorio del rumbo a seguir cuando el obstáculo se haya superado. A partir de ese momento, la frecuencia de muestreo del seguimiento de rumbo aumenta para orientar el vehículo a su destino lo antes posible, y una vez orientado disminuye para ahorrar ancho de banda.

6.3.2.3.4 Análisis final de la optimización obtenida

En los apartados anteriores se ha mostrado cómo la gestión dinámica de las comunicaciones y del control permite reducir la carga de red y de control manteniendo las prestaciones de la navegación (Figura 156). A continuación se comprara la optimización de los dos métodos empleados.

Para la comparación de los métodos, se ha escogido como muestra básica la mayor frecuencia de muestreo experimentada (100 Hz.) ya que es la que obtiene una mayor velocidad de navegación del escenario planteado y permite comparar las optimizaciones como porcentajes. Esta velocidad se logra igualar por medio de la gestión dinámica de la frecuencia de muestreo para los dos métodos de optimización empleados.

Cuando se emplea el método de optimización basado en el ciclo integral de la calidad, la frecuencia de muestreo promedio de la navegación es de 41.98 Hz. Si al método de optimización anterior se le agrega la inclusión de los sensores lógicos de control, que detectan si el comportamiento de seguimiento de rumbo debe actuar, el promedio de frecuencia de muestreo empleado en la navegación es de 34.46 Hz. En ambos casos la velocidad promedio de navegación es similar.

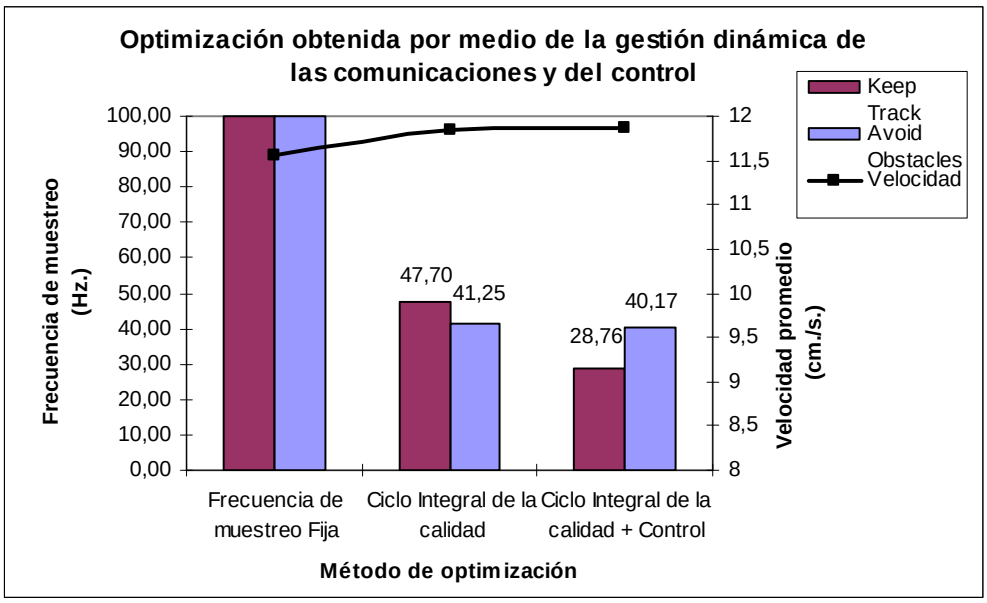


Figura 156. Comparación de la optimización obtenida a partir del ciclo integral de la calidad.

A partir de los resultados obtenidos, se puede comprobar cómo la gestión integral de la calidad, basada en el error de la medición empleado en la QoC y en las políticas de QoS, optimiza la gestión de las comunicaciones y disminuye la carga de control.

6.4 Conclusiones

Los vehículos de Braitenberg llegan hasta el nivel 14 en el que se alcanza la complejidad inherente al comportamiento de sistemas de gran complejidad llegando a basarse en sensaciones futuras como el optimismo. Con el vehículo 5, al añadir elementos intermedios que cuentan, secuencian o priorizan señales aparece un control reactivo que da lugar a comportamientos básicos. En la Tabla 14, se muestra el resumen de las pruebas realizadas y las optimizaciones correspondientes obtenidas.

Tabla 14. Optimizaciones obtenidas en el trabajo experimental.

Vehículo / Escenario	Trabajo experimental
1, 2, 3 y 4 / 1,2 y 3	Optimización por medio de políticas de QoS de la gestión de los mensajes entrantes a los elementos de control. Optimización por medio de sensores lógicos de control que implementan un filtrado previo de mensajes no significativos.
5 / 4	Optimización por medio del ciclo integral de la calidad (QoS y QoC) para variar dinámicamente las características del control en función de las circunstancias de la navegación.

Los primeros vehículos se han empleado para comprobar cómo afecta el tráfico al control y cómo es posible optimizar la carga del middleware por medio de la detección de eventos de políticas de QoS y la realización de acciones asociadas a los elementos de comunicaciones de la arquitectura FSACtrl.

La siguiente optimización ha consistido en emplear sensores lógicos de control que permitan discriminar en función del contenido qué mensajes se envían a los actuadores y descargar aún más el canal de comunicaciones.

Una vez ensayados los primeros cuatro vehículos se puede concluir que la inclusión de políticas de QoS para filtrar temporalmente los mensajes optimiza la carga del componente de control sin que este aspecto afecte de forma relevante a la navegación

del robot ya que la velocidad de la navegación apenas disminuye. Sucede algo similar con la optimización del control mediante la selección de aquellos mensajes que, por su contenido, son relevantes para el control, se obtiene una mejora de la carga del componente de control sin que apenas se vea afectada la navegación del vehículo.

Si se compara los resultados entre los primeros cuatro vehículos se aprecia que a medida que aumenta la complejidad del control, la diferencia entre optimizar por políticas de QoS y optimizar por sensores lógicos de control (control embebido en el middleware) es menor.

De estos cuatro primeros vehículos podemos concluir que la inclusión de políticas de QoS en el middleware permite optimizar el control. Además, la inclusión de elementos de control capaces de realizar sencillas operaciones permite disminuir la carga a la que se somete los elementos de control que puedan estar conectados al middleware.

Con el vehículo 5 se comprueba cómo se emplean las políticas de QoS para variar la frecuencia de muestreo en el origen y, de esta forma, trasladar la optimización del middleware a la fuente de datos. Además, la variación dinámica de las comunicaciones en función de parámetros de calidad de control embebidos en el middleware optimiza el tráfico de red, lo que se traduce en una optimización de la carga que debe soportar el nodo de control.

7 Conclusiones

7.1 Trabajo desarrollado

7.1.1 Investigación

Para desarrollar esta tesis, se ha realizado una labor inicial de investigación sobre los sistemas distribuidos de control, tanto los que tienen escasas restricciones de eficiencia, especialmente temporal, como los sistemas domóticos, y aquellos que tienen más restricciones, como los sistemas de navegación de robots. A partir de las características de los mismos, y teniendo en cuenta las tendencias en comunicaciones, se ha realizado una propuesta de las características que un sistema distribuido de control inteligente debe proporcionar. Además, se ha estudiado el papel de la calidad de servicio en la gestión dinámica de un sistema y la aportación, en términos de calidad, de los estándares de comunicaciones y middleware.

7.1.2 Diseño

A partir de la propuesta de las características deseables concluidas en la investigación inicial, se ha diseñado la arquitectura FSACtrl con el objetivo de cubrir los requisitos de un sistema distribuido de control inteligente. La arquitectura se ha diseñado basándose en el modelo DCPS del estándar DDS propuesto por la OMG y en el modelo SWE propuesto por la OGC. Dada la importancia de la monitorización y la gestión de los eventos en los sistemas distribuidos de control, se ha ampliado el modelo DCPS por medio de la gestión de eventos del estándar RMON propuesto por el *Internet Engineering Task Force* (IETF).

El modelo FSACtrl se ha especificado formalmente mediante el lenguaje UML para facilitar la comprensión de cómo se han unido los diversos estándares. Unir estándares provoca unos problemas de diseño y comportamiento que se han tratado y que dan como resultado la especificación conceptual de la arquitectura. Dado que la arquitectura FSACtrl es una arquitectura basada en un estándar de comunicaciones, la gestión de los mensajes es fundamental. La implementación de dicha gestión se realiza por medio de colas parametrizadas que proporcionan unos parámetros cuantitativos a partir de los cuales se analiza la eficiencia de un sistema.

7.2 Aportaciones

Se resumen a continuación las principales aportaciones de la tesis, centrándose éstas en la arquitectura desarrollada y en el ámbito de aplicación de los sistemas distribuidos de control inteligente.

7.2.1 Investigación

7.2.1.1 Características de las arquitecturas de control distribuido

En la tesis se realiza un estudio detallado de las arquitecturas de control inteligente. Las características se han obtenido a partir de la investigación realizada en el estudio de campo de los sistemas domóticos y los sistemas de navegación de robots, como ejemplo de diferentes tipos de sistemas distribuidos con diferentes necesidades de prestaciones.

La revisión se realizó a partir tanto de las arquitecturas principales empleadas en los sistemas domóticos y de navegación de robots (como ejemplos de plataformas con bajas y altas restricciones respectivamente) como de las sugerencias de características realizadas por los principales autores en cada área.

Como aportación en este campo, en la tesis se propone una organización de las características deseables en un sistema de control inteligente distribuido y las relaciones entre las mismas, estas relaciones determinan de forma teórica las líneas principales que una arquitectura debe cubrir para dar soporte a los sistemas ciber-físicos. Las características sientan las bases para la optimización de un sistema. La optimización justifica la monitorización y la gestión del sistema. Para la gestión del funcionamiento se demuestra como acertada la elección de los parámetros de QoS y de QoC.

7.2.1.2 Sistemas de comunicaciones

De forma análoga a la revisión de las arquitecturas, en la tesis se revisan los paradigmas de comunicaciones y de los sistemas de comunicaciones empleados en las arquitecturas de control, realizando un análisis evolutivo y funcional de los sistemas, especialmente en su aplicación en los middleware actuales.

Como aportación se determinan las características deseables de los sistemas de comunicaciones que, combinadas con las características de las arquitecturas de control, determinan las características que den soporte a los sistemas ciber-físicos.

7.2.1.3 Calidad de servicio

Como consecuencia del análisis de los sistemas de comunicaciones, se ha realizado un estudio de los parámetros de calidad de servicio más habituales en los entornos de comunicaciones. Para ello, en el análisis se ha cubierto desde los paradigmas de comunicaciones hasta los middleware más empleados, demostrándose cómo la medición de la eficiencia de un canal de comunicaciones evoluciona desde el uso de parámetros sencillos hasta la actual tendencia en el que la QoS toma un papel activo ya que se emplea no sólo para la medición sino también para la gestión de las comunicaciones y del control.

Como aportación se han establecido las relaciones existentes entre los parámetros de QoS más empleados y se ha establecido un método para la obtención de los parámetros cualitativos a partir de los parámetros cuantitativos según las tendencias actuales, se ha demostrado cómo los parámetros de las colas de mensajes pueden ser empleados como base para el cálculo de los parámetros de QoS.

7.2.2 Diseño

7.2.2.1 Modelo de arquitectura

Se ha propuesto un modelo de arquitectura de control inteligente distribuido basado en la sinergia que se obtiene de la composición del modelo estándar DCPS, propuesto por la OMG, de comunicaciones y el modelo estándar SWE, propuesto por la OGC, de servicios de control. Las mejoras que se han propuesto están orientadas a emplear la arquitectura propuesta para el modelado y la optimización de componentes de control mediante la gestión dinámica basada en la QoS. Las ampliaciones más relevantes con respecto a los estándares del modelo DDS son las siguientes:

7.2.2.1.1 Espacio de nombres lógicos

Se proporciona una conexión de los elementos de comunicaciones y de control por medio de un espacio de nombres lógicos. A esta estructura se la ha llamado LNT (*Logical Namespace Tree*). El LNT permite abstraer al control de las particularidades de los canales de comunicación, posibilitando diversas representaciones de un mismo sistema desde diversos puntos de vista. Esta abstracción del sistema en función de las necesidades de comunicación y procesamiento permite una adaptación de la información a los elementos que la procesan.

7.2.2.1.2 Árbol de componentes de control

Se suministra una definición de la jerarquía entre los componentes de control por medio de la inclusión de unos componentes de control en otros. A esta estructura se le ha llamado CCT (*Component Control Tree*).

A partir del CCT es posible determinar la jerarquía de los componentes de control del sistema, desde una visión general de todo el sistema de control hasta los detalles de los núcleos de control más específicos. El CCT permite analizar el control jerárquicamente.

7.2.2.1.3 Grafo de sensores lógicos

Se proporciona una especificación de los componentes de control por medio de un grafo dirigido llamado LSG (*Logical Sensor Graph*). Por medio del LSG es posible conocer los detalles del procesamiento y extender hacia el control los parámetros de calidad de servicio, tradicionalmente asociados a las comunicaciones.

Al dividir el componente de control en sensores lógicos, responsables de comunicaciones y de acciones de control independientes, se tiene una visión detallada de la composición del componente de control.

7.2.2.2 Gestión de eventos

La inclusión de los eventos y de las acciones entre los elementos de la arquitectura permite la toma de decisiones dinámicamente. De esta forma se puede automatizar la composición, conexión y ubicación de los componentes de control mientras el sistema está en funcionamiento.

El mecanismo de control de eventos a partir de condiciones que relacionan los sucesos relacionados con la QoS, la QoC y los elementos, junto al filtrado de los mensajes, permite uniformizar el mecanismo desencadenante de acciones sobre el sistema.

Las acciones asociadas a las condiciones completan el ciclo que permite la autogestión de los elementos y del comportamiento del sistema de forma que la intervención del usuario durante la ejecución de la acción de control se reduce a diseñar los eventos, las condiciones y asociar correctamente las acciones correspondientes.

7.2.2.3 Índices de QoS de un sistema

Como consecuencia de las mejoras al modelo DCPS se ha propuesto una serie de índices de QoS con un ámbito mayor al del elemento y que pueden ser empleados para optimizar el funcionamiento de un sistema distribuido de control inteligente. Para la obtención de los índices se ha seguido un modelo constructivo en el que se parte de los parámetros de calidad de servicio cuantitativos. A partir de estos parámetros se obtienen los parámetros de calidad de servicio cualitativos en los que basarse la optimización del funcionamiento del sistema.

7.2.2.4 Ciclo integral de la calidad

A partir de las políticas de QoS, de los parámetros de QoC y con el soporte de la gestión del sistema basada en eventos se puede integrar el ajuste dinámico del sistema. De esta forma se demuestra que la toma de decisiones acerca de diversos aspectos sobre los componentes en los sistemas distribuidos puede realizarse a partir de los parámetros de QoS y los parámetros de QoC. Entre los aspectos que pueden ser gestionados se encuentran la gestión distribuida y dinámica de la información redundante, los criterios de movilidad de elementos o la gestión de la topología correcta de elementos para la realización de acciones de control específicas.

7.2.2.5 Soporte a la implementación de sistemas de agentes

El trabajo desarrollado proporciona el soporte adecuado para la implementación de sistemas de agentes de control a partir de los componentes de control. La gestión de los eventos, asociándolos a acciones, hace que los componentes de control proporcionen la infraestructura necesaria para que pasen de ser componentes de control pasivos programados a componentes de control activos que pueden tomar la iniciativa durante el funcionamiento y variar las características del sistema.

La aportación principal como sistema de agentes la forma el conjunto de protocolos para las operaciones de división y fusión, como operaciones estructurales de agentes no soportadas en todos los sistemas, y las operaciones más generales de clonación y movimiento de agentes.

7.2.3 Trabajo experimental

7.2.3.1 Desarrollos

Para la experimentación con la arquitectura propuesta en esta tesis doctoral se han desarrollado dos aplicaciones: una aplicación de diseño y ejecución de los elementos del sistema de control y otra aplicación de simulación de entornos para robots móviles que permite obtener datos comparativos que demuestran la viabilidad de la arquitectura.

7.2.3.2 Experimentos

Los experimentos desarrollados demuestran la viabilidad de la arquitectura como sistema de control reactivo. Se ha comprobado cómo evaluar el rendimiento de las comunicaciones y del control por medio de los parámetros de QoS. Además, se ha mostrado cómo optimizar un sistema por medio de la gestión de las comunicaciones y de la inclusión de parte del control en el middleware. Finalmente, se comprueba cómo optimizar el rendimiento de las comunicaciones y del control a través de la gestión de las comunicaciones por medio de la configuración de las políticas de QoS.

7.3 Trabajo futuro

A partir de la tesis doctoral es posible desarrollar diversos trabajos en diferentes áreas de investigación de los campos cubiertos por la arquitectura FSACtrl: comunicaciones y control. A continuación se exponen las líneas organizadas en estos campos.

7.3.1 Arquitecturas de control distribuido inteligente

7.3.1.1 Comunicaciones

Es especialmente atractiva la investigación en el papel, que cada vez más relevante, va tomando el middleware. En la mayor parte de los sistemas distribuidos el middleware ha pasado de ser un intermediario pasivo con labores de adaptación a un elemento activo que puede influir en gran medida en el rendimiento del sistema.

Se propone emplear la arquitectura FSACtrl como middleware para el descubrimiento automático de las topologías de los canales de comunicaciones de los sistemas sobre los que funciona. Además, la inclusión de parte del control como elementos del middleware puede emplearse para comprobar la redundancia en la información desde las propias fuentes de datos.

7.3.1.2 Control basado en eventos

En cuanto a los sistemas de control, el creciente interés en el control basado en eventos hace que la arquitectura FSACtrl sea un banco de pruebas idóneo para la determinación de los eventos necesarios o relevantes en el procesamiento de un determinado comportamiento. Esto se realiza empleando el ciclo integral de la calidad para realizar la correlación de los eventos con los parámetros, especialmente los de QoC, para comprobar la incidencia que tiene cada evento en la calidad global del sistema.

7.3.1.3 Medición del impacto de las operaciones

El ciclo integral de la calidad que, unido a las condiciones y las acciones, permite configurar márgenes de funcionamiento del sistema hace que la arquitectura FSACtrl permita evaluar qué impacto tienen las operaciones de los agentes de control en el sistema cuando se añaden o eliminan componentes de control, o cuando se realizan operaciones estructurales con componentes de control.

Este mecanismo de medición del impacto de las operaciones en el sistema permite la evaluación de los recursos computacionales consumidos debido a una configuración de elementos determinada, o la ejecución de la acción de control en un escenario concreto.

7.3.1.4 Simulación para la composición topológica óptima y la auto-distribución

Una de las líneas más interesantes que se abren a partir del trabajo desarrollado es la simulación de diversas configuraciones de elementos para conocer la conveniencia de qué estructura topológica de los mismos es más adecuada a un escenario concreto.

Por medio de las acciones estructurales, que permiten tanto modificar los elementos como añadirlos o eliminarlos, es posible reubicar el sistema distribuido. Además, la medición en términos de QoS de los elementos proporciona un medio para generar propuestas de sistemas de forma automática.

7.3.2 Gestión integral basada en la QoS y la QoC

Donde más líneas de investigación parten de la tesis es en el empleo de la QoS y la QoC en los sistemas distribuidos de control inteligente, especialmente en el ámbito de los sistemas de agentes móviles. Es determinante el papel de la QoS y QoC en la creación de índices de calidad de un agente (QoA) que permitan conocer la adecuación de un agente a un entorno concreto y a unas circunstancias concretas.

También es posible emplear los parámetros de QoS y de QoC para ajustar dinámicamente un sistema, y de emplear estos ajustes para medir cómo de optimo es el funcionamiento de un sistema.

7.3.2.1 Ventana óptima de trabajo

Con la arquitectura FSACtrl es posible determinar valores de márgenes de funcionamiento basados en los parámetros de QoS y de QoC que determinan una ventana de trabajo.

A partir de una situación inicial del índice combinado de los dos parámetros (i_0), se van tomando decisiones en el sistema como el movimiento de agentes, la variación de otros parámetros de QoS o acciones similares. En función del acercamiento o no a la zona óptima es posible comprobar la secuencia de acciones que optimiza un índice concreto.

A partir de esos márgenes, monitorizando los parámetros de QoS y de QoC, es posible evaluar la evolución de los parámetros y cómo se acercan a una zona objetivo de la ventana de trabajo, lo que implica el acercamiento a la optimización del sistema.

Esta evolución de los parámetros hace que el sistema pueda realizar una adaptación dinámica a las condiciones del entorno sobre el que funciona, pero basándose en el empleo de parámetros estandarizados (QoS y QoC) y no de variables propias del sistema (tiempo de navegación, distancia recorrida).

Las variables basadas en parámetros de QoS y QoC son conocidas en todo momento, por lo que es posible inferir qué tendencia tiene el sistema en cuanto a optimización, mientras que las variables inherentes al sistema, como el tiempo o la distancia recorrida son conocidas a la finalización de la misión.

Una línea interesante consiste en averiguar la relación entre las variables de calidad y las variables inherentes del sistema. De esta forma es posible determinar qué parámetros de QoS deben optimizarse, y con qué valores, en las ventanas óptimas de trabajo.

7.3.2.2 Índices de optimización de un sistema

A partir de la arquitectura FSACtrl es posible extender los resultados de calidad a fórmulas más complejas generando índices de calidad abstractos pero con una base empírica sólida ya que los parámetros de QoS están basados en parámetros empíricos comunes a todos los elementos del sistema. Una de las líneas actualmente en desarrollo es la generación de los índices propuestos en el proyecto SIDIRELI [SIDIRELI, 2008].

Los índices de calidad de un sistema, o de algunos de sus componentes, son tan variados como lo son los sistemas sobre los que se aplican. Es habitual concentrar en un solo índice el valor de calidad del sistema de forma que se pueda cuantificar la eficiencia del mismo en un aspecto [Posadas, 2003]. En el caso de los sistemas de control también suele hablarse de un único parámetro que especifica la calidad del control [Gabel and Litz, 2003].

Existen diversas aproximaciones sobre parámetros que proporcionan la calidad de un sistema de control. Por ejemplo, en [Gabel and Litz, 2003] se propone el cálculo de la QoC a partir de una fórmula en la que aparecen constantes de ajuste y funciones generales sobre el comportamiento final de los componentes. Sin embargo, una visión más detallada aparece en [Soucek and Sauter, 2004] donde se muestra una aproximación para obtener la QoC relacionada con la QoS.

Si se considera el control como un servicio, y se tiene en cuenta la definición de calidad de servicio donde se habla de efecto colectivo, parece conveniente averiguar la posibilidad de obtener la calidad del control a partir de los parámetros de los elementos. Para ello es conveniente conocer los índices que permitan cuantificar las cualidades definidas por las características: cabe destacar que hay características fácilmente cuantificables, como la estabilidad y otros más difícilmente cuantificables, como el direccionamiento semántico. A partir de estos índices se deben desarrollar las fórmulas que los implementan y la relevancia que tienen los índices en la optimización de un sistema distribuido de control inteligente.

7.4 Proyectos y publicaciones

Las publicaciones realizadas, según las características de la arquitectura tratadas en cada caso, así como la relación de las mismas con los proyectos de investigación en los que se ha enmarcado esta tesis se muestran en la Figura 157. A continuación se detallan estos dos aspectos.

7.4.1 Proyectos

Los aspectos desarrollados en la tesis se enmarcan en la línea de investigación de las Arquitecturas para Sistemas de Control Inteligente, perteneciente al grupo de investigación de Informática Industrial del cual forma parte el autor de esta tesis. Esta línea de investigación trata principalmente los temas de interfaces de comunicaciones y control en los sistemas distribuidos, protocolos y distribución del procesamiento y de la información. El trabajo desarrollado en esta tesis se enmarca en dos proyectos de investigación: KERTROL [KERTROL, 2005] y SIDIRELI [SIDIRELI, 2008].

Las primeras ideas acerca de la ampliación del modelo FSA, empleado para la navegación de robots en el grupo de investigación, a modelos distribuidos aparecen en el año 2004 y sirven de base para la investigación desarrollada en el proyecto KERTROL. La inclusión de los modelos de agentes y del ciclo integral de la calidad se incluye dentro del proyecto SIDIRELI.

Las aportaciones que la tesis presentada ha realizado a los proyectos se han centrado fundamentalmente en los aspectos de las comunicaciones, así como la formalización de las interfaces de programación del control inteligente, la generación de los índices de calidad de un sistema de control distribuido y la extrapolación del control a sistemas de agentes distribuidos.

7.4.2 Publicaciones

La divulgación de los resultados de la tesis se ha realizado en diversos congresos internacionales, la mayor parte de ellos indexados en los índices de prestigio. El ámbito de los congresos indexados es el de sistemas de agentes (IWPAAMS) y el de sistemas distribuidos inteligentes (DCAI).

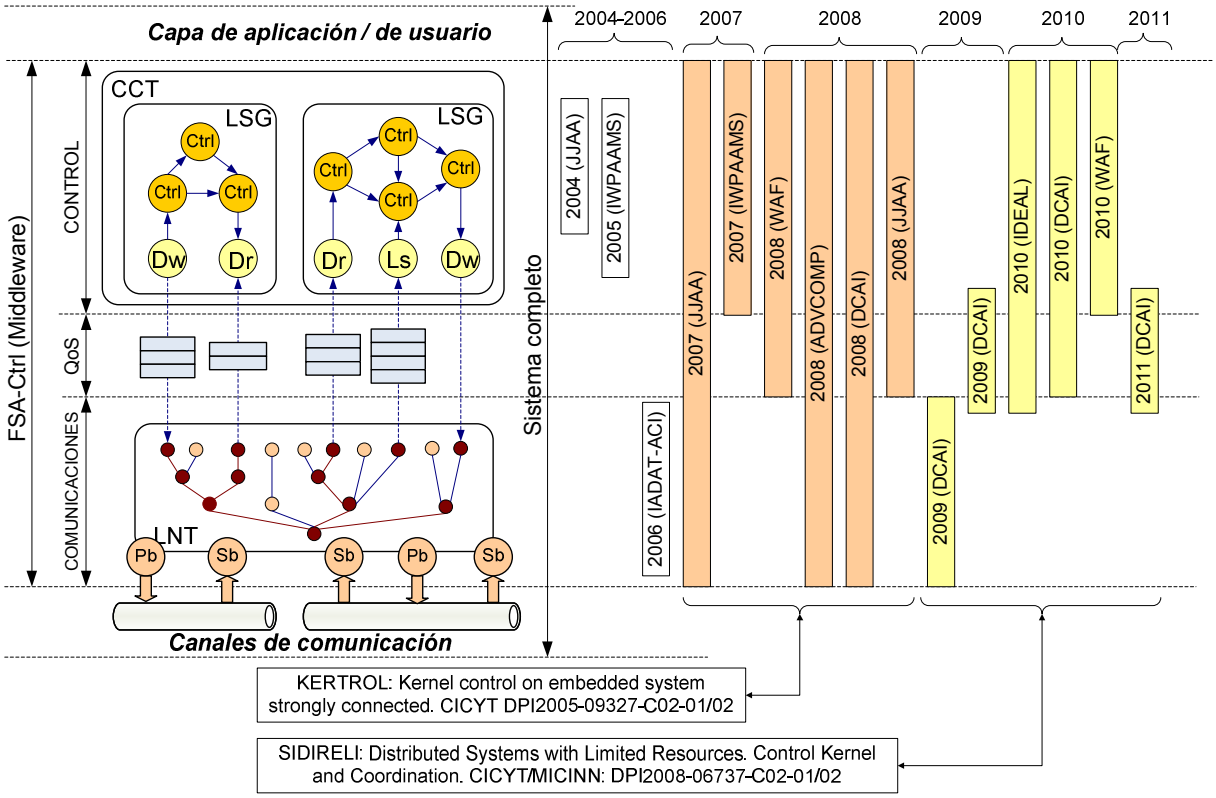


Figura 157. Áreas de la arquitectura FSACtrl cubiertas por las publicaciones.

En cuanto a las primeras publicaciones, la temática se centró en los sistemas industriales, tanto en el control como en las comunicaciones y la incidencia de las mismas en la obtención de información relevante para el control, ya que eran el punto de partida en el diseño de la arquitectura. En concreto, las áreas de la tesis cubierta por las publicaciones fueron las siguientes:

- [Poza et al., 2004]. *Modelo de Arquitectura de Comunicaciones Frame-Sensor-Adapter (FSA)*. En esta publicación se muestra la necesidad de incluir parte del procesamiento de control en los elementos de comunicaciones en forma de agentes mínimos de control.
- [Poza et al., 2005]. *Sistema de seguridad variable en una arquitectura de agentes móviles*. En esta publicación se muestra el mecanismo de seguridad adaptativo para la comunicación entre agentes de control. Además se destaca la necesidad de organizar jerárquicamente los agentes de control.
- [Poza et al., 2006] *Hierarchical Communications System to Manage maps in Mobile robot navigation*. En esta publicación se muestra el LNT cuya necesidad se había expuesto en la anterior publicación, aunque sin contextualizarlo todavía a la arquitectura FSACtrl.

Seguidamente, se traslada el conocimiento a entornos más amigables para el ensayo de nuevos sistemas, tanto la robótica como la domótica. Este último como ejemplo de línea de sistemas automáticos, generalmente centralizados, cuya tendencia es pasar a entornos inteligentes distribuidos. Las áreas de la tesis cubierta por las publicaciones son:

- [Poza et al, 2007a]. *Arquitectura Distribuida para el control de sistemas*. En este artículo se plantea la combinación de los elementos expuestos en las publicaciones anteriores, apareciendo la necesidad de disponer de elementos que agrupen y jerarquicen el control.
- [Poza et al, 2007b]. *Especificación de agentes distribuidos para una arquitectura de control inteligente*. En esta publicación se aplica el concepto de componente de control para la implementación de agentes móviles.
- [Poza et al, 2008a]. *Arquitectura de agentes de control con soporte a la calidad de servicio*. En esta publicación aparece la necesidad de emplear la QoS como medio para medir la eficiencia del sistema. Se muestra el modelo DCPS del estándar DDS en el que se basa la arquitectura.
- [Poza et al, 2008b]. *Middleware with QoS support to control intelligent systems*. En esta publicación se ofrece la primera visión integral de la arquitectura FSACtrl como arquitectura de middleware con soporte al control.
- [Poza et al, 2008c]. *QoS-bassed middleware architecture for distributed control systems*. En esta publicación se ofrece la especificación formal de la arquitectura FSACtrl presentada en el anterior artículo.
- [Poza et al, 2008d]. *Arquitectura de control jerárquico inteligente con soporte a la calidad de servicio*. En esta publicación se contextualiza a los sistemas de control automático con los elementos de la arquitectura FSACtrl.

Finalmente, la tesis se centra en la QoS y la QoC como medios para medir el rendimiento del sistema por medio del ciclo integral de la calidad. Además, se implementan estos aspectos a partir de las colas y la elaboración de los índices y se presenta la implementación de los sistemas de agentes con operaciones complejas como el clonado y el movimiento de agentes. Las publicaciones relacionadas con estos aspectos son las siguientes:

- [Poza et al, 2009a]. *From the queue to the quality of service*. En esta publicación se demuestra cómo obtener los parámetros de las políticas de QoS del modelo DCPS a partir de los parámetros de las colas de mensajes empleadas en los elementos de la arquitectura FSACtrl.
- [Poza et al, 2009b]. *Adding an ontology to a standardized QoS-based MAS middleware*. En esta publicación se expone cómo el LNT puede usarse como método de abstracción del sistema y cómo los parámetros de QoS describen el funcionamiento instantáneo del mismo.
- [Poza et al., 2010a]. *Multi-Agent architecture with support to Quality of Service and Quality of Control*. En este artículo se presenta el soporte conjunto a la QoS y QoC de la arquitectura FSACtrl así como el ciclo integral de la calidad.
- [Poza et al., 2010b]. *Quality of Service and Quality of Control Based Protocol to Distribute Agents*. En este artículo se presenta el protocolo para distribuir agentes a partir del ciclo integral de la calidad presentado en la publicación anterior.
- [Poza et al., 2010c]. *Arquitectura multi-agente distribuida con soporte a la calidad de servicio y a la calidad de control*. En este artículo se presenta la gestión de los eventos empleada para el ciclo integral de la calidad y su utilidad para el control basado en eventos.

- [Poza et al., 2011]. A Survey on Quality of Service Support on Middleware-Based Distributed Messaging Systems Used in Multi Agent Systems. Por medio de este artículo se publica la revisión de la QoS en relación a los middleware empleados en los sistemas multi-agente, así como las consecuencias del uso de las mismas.

Actualmente están enviadas y en desarrollo diversas publicaciones centradas en las siguientes áreas de la arquitectura:

- Experimentación con los vehículos de Braitenberg como ejemplo del uso de FSACtrl para evaluar, por medio de los parámetros de QoS de los componentes de control, la idoneidad del sistema de control.
- Explotación de las condiciones, basándose en la combinación de la detección de eventos y la asociación de acciones.

"Con un hisopo entintado marcó cada cosa con su nombre: mesa, silla, reloj, puerta, pared, cama, cacerola. Fue al corral y marcó los animales y las plantas: vaca, chivo, puerca, gallina, yuca, malanga, guineo. Paco a poco, estudiando las infinitas posibilidades del olvido, se dio cuenta de que podía llegar un día en que se reconocieran las cosas por sus inscripciones, pero no se recordara su utilidad. Entonces fue más explícito. El letrero que colgó en la cerviz de la vaca era una muestra ejemplar de la forma en que los habitantes de Macondo estaban dispuestos a luchar contra el olvido: Ésta es la vaca, hay que ordeñarla todas las mañanas para que produzca leche y a la leche hay que hervirla para mezclarla con el café y hacer café con leche. Así continuaron viviendo en una realidad escurridiza, momentáneamente capturada por las palabras, pero que había de fugarse sin remedio cuando olvidaran los valores de la letra escrita."

Cien años de soledad. Gabriel García Márquez.

Referencias

Abdelzaher et al., 2004 — T. Abdelzaher, B. Blum, Q. Cao, D. Evans, J. George, S. George, T. He, L. Luo, S. Son, R. Stoleru, J. Stankovic, A. Wood. EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks. In Proc. of the 24th Int'l Conf. on Distributed Computing Systems (ICDCS 04) March 23-26, Tokyo, Japan, 2004.

Aiello, 2005 — Aiello, Marco (2005) The Role of Web Services at Home. Technical Report DIT-05-065, Informatica e Telecomunicazioni, University of Trento.

Alami et al., 1998 — R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand (1998). An architecture for Autonomy. International Journal of Robotics Research, Vol. 17, No. 4, pp. 315-337.

Alami et al., 2000 — R. Alami, R. Chatila, S. Fleury, M. Herrb, F. Ingrand, M. Khatib, B. Morisset, P. Montarlier, and T. Simeon, (2000). Around the lab in 40 labs... In IEEE International Conference on Robotics and Automation, ICRA'O0, San Francisco.

Albus and Barbera, 2005 — J.S. Albus and A.J. Barbera, "RCS: A cognitive architecture for intelligent multi-agent systems," In Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles (2004).

Antsaklis, 1999 — Antsaklis, P.J. "Intelligent Control," Encyclopedia of Electrical and Electronics Engineering, Vol. 10, pp. 493-503, John Wiley & Sons, Inc., 1999.

Antsaklis and Baillieul, 2007 — Antsaklis, P. J. and Baillieul, J. Special Issue on Technology of Networked Control Systems. Proceedings of the IEEE, 95(1), 5-8

Arkin, 1990 — Arkin, R.C., (1990). Integrating Behavioural, Perceptual and World Knowledge in Reactive Navigation. Robots and Autonomous Systems, 6, 105-122.

Arkin, 1998 — Arkin, R.C., (1998). Behavior-Based Robotics. MIT Press, Cambridge.

Astigarraga, 2001 — A. Astigarraga Pagoaga, Agent-based control architecture for a mobile robot, M.Sc. thesis for Dept. of Informatics, Universidad del País Vasco, 2001.

Averill and Kelton, 2000 — Averill M. Law y W. David Kelton. Simulation Modeling and Analysis. Tercera edición. McGraw-Hill, 2000.

Baliga and Kumar, 2004 — Baliga, G., Kumar, P. R. A Middleware for Control over Networks. In proceedings of the 44th IEEE Conference on Decision and Control. Sevilla, Spain. 482-487 (2005)

Bass et al., 2003 — Bass, L., Clements, P., Kazman, R. Software Architecture in Practice. Second Edition. Addison Wesley. 2003.

Bergmans et al., 2000 — L. Bergmans, A. van Halteren, L. Ferreira Pires, M. van Sinderen, M. Aksit. A Qos control architecture for object middleware. 7th Intl. Conf. on Interactive Distributed Multimedia Systems and Telecommunication Services, 2000.

Bizzarri, 1999 — Bizzarri, Maurice (Nov. 1999). Home API: A network-independent home control architecture. The Universal Plug and Play Forum.

Bolch et al., 2006 — Bolch G, Greiner S, de Meer H, Trivedi KS: Queueing Networks and Markov Chains. John Wiley & Sons, Inc; 2on Ed. 2006

Bonasso, 1997 — Bonasso, R.P., Firby, J., Gat, E., Kortenkamp, D., Miller, D., Slack, M., (1997). A Proven Three-tiered Architecture for Programming Autonomous Robots. Journal of Experimental and Theoretical Artificial Intelligence, vol. 9, no. 2.

Bonnet et al., 2001 — P. Bonnet, J. Gehrke, P. Seshadri. Towards Sensor Database Systems. Proc. of the 2nd Int'l Conf. Mobile Data Management (MDM 01), Hong Kong, China, 2001, pp 314–810.

Boonma and Suzuki, 2010 — P. Boonma and J. Suzuki, "TinyDDS: An Interoperable and Configurable Publish/Subscribe Middleware for Wireless Sensor Networks," *In A. Hinze and A. Buchmann (eds.) Principles and Applications of Distributed Event-Based Systems*, Chapter 9, pp. 206 - 231, IGI Global, ISBN: 978-160-566-698-3, May 2010.

Botts et al., 2006 — Botts M, Percivall G, Reed C, Davidson J, (2006) OGC®. Sensor Web Enablement: Overview And High Level Architecture, OpenGIS Consortium Inc

Braitenberg, 1984 — Braitenberg, V., 1984. Vehicles: Experiments on Synthetic Psychology. MIT Press, Cambridge, Massachusetts.

Brooks, 1986 — Brooks, R.A., (1986). A robust layered control architecture for a mobile robot. IEEE Journal of Robotics and Automation, 2, (1), 14-23.

Brooks, 1991a — Brooks, R.A., (1991) "Integrated Systems Based on Behaviors", SIGART Bulletin (2:4), August 1991, pp. 46–50.

Brooks, 1991b — Brooks, R.A., (1991). Intelligence without representation. Artificial Intelligence, 47, pp. 139-159.

Busch, 2006 — Busch, D. TAO's Data Distribution Service. Object Computing, Inc. DDS Information Day. 2006

Camacho et al., 2007 — Jesús Camacho Villanueva, Juan Luis Posadas Yagüe, José Luis Poza Luján, Jesús Friginal López. Arquitectura SCHome: interconexión entre distintos buses domóticos mediante mensajes XML y pasarelas software. XXVIII Jornadas de Automática. Huelva. 2007.

Caruso et al., 2007 — KydoSoft: High Integrity RT-Java DDS. Kydo Soft 2007 (www.kydosoft.com)

Chao-Lin et al., 2004 — Chao-Lin Wu, Wei-Chen Wang, Li-Chen Fu. Mobile agent based integrated control architecture for home automation system. Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, Vol.4, Iss., 28 Sept.-2 Oct. 2004. Pages: 3668- 3673 vol.4

Cheng-Fa and Hang-Chang, 2002 — Cheng-Fa Tsai and Hang-Chang Wu, "Massihn: A Multi-Agent Architecture for Intelligent Home Network System," IEEE Transactions on Consumer Electronics Vol.48, Issue 3, pp.505-514, August 2002.

Cook and Das, 2007 — Diane J. Cook, Sajal K. Das, How smart are our environments? An updated look at the state of the art, Pervasive and Mobile Computing, v.3 n.2, p.53-73, March, 2007.

Cook et al., 2003 D. J. Cook, M. Youngblood, E. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, MavHome: An Agent-Based Smart Home. In proceeding of the Conference on Pervasive Computing, 2003.

Coste-Manière and Simmons, 2000 Coste-Manière, E. & Simmons, R., (2000), "Architecture, the Backbone of Robotic Systems", Proceedings of the 2000 IEEE International Conference on Robotics & Automation, San Francisco, CA.

Coulouris et al., 2001 Coulouris, G., Dollimore, J., Kindberg, T. Sistemas Distribuidos, Conceptos y diseño. Tercera Edición. Addison Wesley. Madrid. 2001.

Crawley et al., 1998 Crawley, E.; Nair, R; Rajagopalan, B. "RFC 2386: A Framework for QoS-based Routing in the Internet". August. 1998, pp. 1-37, XP002219363

De Carolis and Cozzolongo, 2004 Berardina De Carolis, Giovanni Cozzolongo: C@sa: Intelligent Home Control and Simulation. International Conference on Computational Intelligence 2004: 462-465

Demichelis and Chimento, 2002 Demichelis, C.; Chimento, P. "RFC 3393: IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)". The Internet Society. November 2002.

Dianes, J.A. et al., 2010 Dianes, J.A. Díaz, M. Rubio, B. ServiceDDS: A Framework for Real-Time P2P Systems Integration. 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC). 2010

Dorf and Bishop, 2008 R.C. Dorf and R.H. Bishop, Modern Control Systems, 11th Edition, Prentice Hall (2008)

Dormido et al., 2008 S. Dormido, J. Sánchez, E. Kofman. Muestreo, Control, y Comunicación basados en eventos. RIAI. Revista Iberoamericana de Automática e Informática Industrial. Vol. 5, núm. 1, Enero 2008, pp. 5-26.

Dursun, 2006 Mustafa Dursun. Data Sharing and Access with a CORBA Data Distribution Service Implementation. Msc Thesis. Middle East Technical University (Turkey) 2006

FIPA, 1997 Foundation for Intelligent Physical Agents: FIPA 97 Specfication. Part 2, Agent Communication Language. (1997).

FIPA, 2000 Foundation for Intelligent Physical Agents. FIPA Agent Management Specification, Doc: FIPA00023. (2000)

FIPA, 2002 Foundation for Intelligent Physical Agents. FIPA Quality of Service Ontology Specification. Doc: SC00094A. (2002)

Gabel and Litz, 2004 Gabel, O. Litz, L. QoS-adaptive Control in NCS with Variable Delays and Packet Losses – A Heuristic Approach. 43rd IEEE Conference on Decision and Control. 2004

Gaddah and Kunz, 2003 Gaddah A., and Kunz, T. A survey of middleware paradigms for mobile computing. Technical Report SCE-03-16. Carleton University Systems and Computing Engineering. (2003)

Gallium, 2009 Gallium InterCOM DDS . www.gallium.com/whitepapers/intercom_ebook.pdf

Gang et al., 2000 Gang S, Weinraub Y, Giladi R. SNMP interface for building automation components, Proceedings of EIB Scientific Conference 2000, Munich, Germany.

Giladi, 2004 Giladi, R. SNMP for home automation, Intern. Journal of Network Management, Wiley, Vol.14, Iss.4, PP231-239, May 4th, 2004.

González, 2007 González, E. Calidad en servicios de transporte público de personas: La UNE-EN 13816:2003. Dyna, ISSN 0012-7361, Vol. 82, Nº 1, 2007, Págs. 44-46

Gowdy, 2000 Gowdy, J. A Qualitative Comparison of Interprocess Communications Toolkits for Robotics. Tech. report CMU-RI-TR-00-16, Robotics Institute, Carnegie Mellon University, June, 2000

Gupta and Chow, 2010 R. A. Gupta, M.-Y. Chow. Networked Control System: Overview and Research Trends. Transaction of Industrial Electronics, vol.57, no.7, pp.2527-2535, July 2010.

Hadim and Mohamed, 2006 S. Hadim and N. Mohamed. Middleware: Middleware challenges and approaches for wireless sensor networks. In IEEE distributed systems online, volume 7. IEEE Computer Society, March 2006.

Heinzelman et al., 2004 W.B. Heinzelman et al. Middleware to Support Sensor Network Applications. IEEE Network, vol. 18, no. 1, 2004, pp. 6–14.

Hill et al., 2000 J. Hill, R. Szewczyk, A. Woo, S.Hollar, D. Culler and K. Pister, "System Architecture Directions for Networked Sensors". ASPLOS-IX Cambridge, MA, USA. ACM 2000

Houston, 1998 P. Houston. Building distributed applications with message queuing middleware – white paper. Technical report, Microsoft Corporation, 1998.

Hristu-Varsakelis and Levine, 2005 Hristu-Varsakelis D. Y Levine W.S. (Ed.), —Handbook of Networked and Embedded. Control Systems", Springer, 2005

Huo et al., 2004 Zhihong Huo, Huajing Fang, Chaglin Ma, Networked control system: state of the art, in: Proceedings of WCICA'2004, pp. 1319–1322

IEEE, 2000 IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, Sponsor: Software Engineering Standards Committee of the IEEE Computer Society, Approved 21 September 2000. (ISO/IEC 42010:2007)

ITU, 1994 ITU-T Recommendation E.800 (0894). Terms and Definitions Related to Quality of Service and Network Performance Including Dependability, 1994.

JacORB, 2007 JacORB DDS. http://www.java2s.com 2007

Jain, 1991 Raj Jain. The art of Computer Systems Performance Analysis. John Wiley & Sons Inc. New york. 1991

KERTROL, 2005 Kernel control on embedded system strongly connected. CICYT DPI2005-09327-C02-01/02. Ministerio de Educación y Ciencia, Gobierno de España. (http://guada.disca.upv.es/kertrol/)

Kopetz and Grunsteidl, 1993 — Kopetz, H., Grunsteidl, G. "TTP - A time-triggered protocol for fault-tolerant real-timesystems", Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers., The Twenty-Third International Symposium on, Toulouse, France: IEEE, pp. 524–533.

Kruchten, 1995 — Kruchten, P. "*Architectural Blueprints--The 4+1 View Model of Software Architecture*". IEEE Software, Institute of Electrical and Electronics Engineers. November 1995, pp. 42-50

Kumar, 2001 — Kumar, P.R. New technological vistas for systems and control: The example of wireless networks. Control Systems Magazine, 21(1):24–37, 2001. [24]

Kutluca et al., 2007 — Kutluca, H. Emre, I, Deniz, E., Bal, B., Kilic, M., Cakir. U. Developing MilSOFT DDS Middleware. OMG Real-time and Embedded Systems Workshop Arlington-VA USA-July-9-12, 2007

Kwon et al., 2008 — Ki-Jeong Kwon, Choong-Bum Park, Hoon Choi, "DDSS: A Communication Middleware based on the DDS for Mobile and Pervasive Systems," The 10th International Conference on Advanced Communication Technology, Feb. 17-20, 2008.

Lee, 2008 — Edward A. Lee, "Cyber Physical Systems: Design Challenges" isorc, pp.363-369, 2008 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing.

Levis and Culler, 2002 — P. Levis and D. Culler , "Mate: A Tiny Virtual Machine for Sensor Networks,"Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOSX), ACM Press, 2002, pp. 85-95.

Li and Wang, 2008 — Li Li, Fei-Yue Wang. Control and Communication Synthesis in Networked Control Systems. International Journal Of Intelligent Control And Systems. Vol. 13, No. 2, June 2008, 81-88

Li et al., 2003 — S. Li, S. Son, J. Stankovic. Event Detection Services Using Data Service Middleware in Distributed Sensor Networks. In Proc. of the 2nd Int'l Workshop Information Processing in Sensor Networks (IPSN 03), Palo Alto, California, USA, April 22-23, 2003, pp. 502–517

Liu and Martanosi, 2003 — T.Liu, M. Martonosi, Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. PPoPP'03, San Diego, California, USA, June 2003.

Liu, 2004 — Liu, M.L., Distributed Computing: Principles and Applications. Addison-Wesley, 2004.

Madden et al., 2005 — S.R. Madden , M.J. Franklin and J.M. Hellerstein , "TinyDB: An Acquisitional Query Processing System for Sensor Networks,"ACM Trans. Database Systems, vol. 30, no. 1, 2005, pp. 122-173.

Miori et al., 2006 — Miori, V.; Tarrini, L.; Manca, M.; Tolomei, G. An open standard solution for domotic interoperability. Consumer Electronics, IEEE Transactions on, Vol.52, Iss.1, Feb. 2006. Pages: 97- 103.

Molla and Ahamed, 2006 — Molla, M.M., Ahamed, S.I.: A Survey of Middleware for Sensor Network and Challenges. In ICPP Workshops(2006) 223-228

Moravec, 2001 Moravec, H. Entrevista de la serie documental “Beyond Human”, Capítulo: “Living Machines”. Producido y dirigido por Thomas Lucas. Lise Zumwalt PBS Video, 2001.

Mozer, 1998 M. C. Mozer. The neural network house: An environment that adapts to its inhabitants. In Proceedings of the AAAI 1998.

Murphy, 2000 Murphy, R.R., (2000). Introduction to AI Robotics. MIT Press.

Nii, 1989 Nii, H. P., (1989). Introduction, in: Blackboard architectures and applications. Edited by V. Jagannathan, Rajendra Dodhiawala and Lawrence S. Baum, (Perspectives in artificial intelligence, volume 3). Academic Press, Boston.

OCI, 2010 OpenDDS Developer’s Guide. Object Computing, Inc. (www.opendds.org) 2010

OMG, 2001 Object Management Group (OMG). The Common Object Request Broker Architecture and Specification. CORBA 2.4.2. (2001)

OMG, 2003 Object Management Group (OMG). Unified Modelling Language Specification, v1.5. Formal/03-03-01 (ISO/IEC 19501), March 2003.

OMG, 2005 Object Management Group (OMG). “Data Distribution Service for Real-Time Systems, v1.1.” Document formal/2005-12-04. December 2005.

OMG, 2007 Object Management Group (OMG). “Data Distribution Service for Real-Time Systems, v1.2.” Document formal / 2007-01-01. January 2007.

Orebäck and Christensen, 2003 Orebäck, A. y Christensen, H.I., (2003). Evaluation of Architectures for Mobile Robotics. Autonomous Robots 14, pp. 33-49.Kluwer Academic Publishers.

Pardo-Castellote, 2003 Pardo-Castellote, G. OMG Data-Distribution Service: architectural overview. Proceedings of 23rd International Conference on Distributed Computing Systems Workshops. Providence, USA. Vol. 19-22, pp. 200-206. (2003)

Paxson et al., 1998 Paxson, V., Almes, G., Mahdavi, J. and M. Mathis, "Framework for IP Performance Metrics", RFC 2330, Mayo 1998.

Pérez et al., 2001 P.Pérez, J.L.Posadas, J.E.Simó, G.Benet, F.Blanes. Communication Architecture for Mobile Robots Teleoperation. Proceedings of the 1st IFAC Conference on Telematics Applications in Automation and Robotics. TA2001.

Petriu et al., 2000 E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis, V.Z. Groza, Sensor-based information appliances, IEEE Instrumentation and Measurement Magazine 3 (4) (2000) 31–35.

Posadas et al., 2002 Posadas, J. L. Pérez, P. Simó, J. E. Benet, Blanes, G. F. Communications structure for sensory data in mobile robots. Engineering Applications of Artificial Intelligence. Volume 15, Issues 3-4, June-August 2002, Pages 341-350

Posadas et al., 2008 Posadas, J.L., Poza J.L., Simó J.E., Benet G., Blanes, F. Agent Based Distributed Architecture for Mobile Robot Control. Engineering Applications of Artificial Intelligence. Pergamon Press Ltd. 2008. Vol.: 21 N. 6. p.p. 805-823

Posadas, 2003 Posadas, J. L. Arquitectura para el Control de Robots Móviles Mediante Delegación de Código y Agentes. Tesis Doctoral. Universidad Politécnica de Valencia. 2003.

Poza et al., 2003 Poza, J.L., Posadas, J.L., Simó, J.E., Crespo, A. Communication System Architecture to Manage a Containers Terminal. Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA2003), Lisbon, 2003

Poza et al., 2004 Poza, J.L., Posadas, J.L., Simó, J.E. Pérez, P., Simarro, R. Modelo de Arquitectura de Comunicaciones Frame-Sensor-Adapter (FSA). XXV Jornadas de Automática.

Poza et al., 2005 Poza, J.L., Posadas, J.l. and Simó, J.E. Sistema de seguridad variable en una arquitectura de agentes móviles. 4th International Workshop on Practical Applications of Agents and Multiagent Sistems. León (Spain). IWPAAMS 2005.

Poza et al., 2006 Poza, J. L., Posadas, J.L., Simó, J.E. and Benet, G. (2006) Hierarchical communication system to manage maps in mobile robot navigation. Proceedings of International Conference on Automation, Control and Instrumentation.

Poza et al., 2007a Poza, J.L. Posadas, J.L., Simó, J.E. Ginés, B. (2007) Arquitectura Distribuida para el Control de Sistemas. XXVIII Jornadas de Automática.

Poza et al., 2007b Poza, J.L., Posadas, J.L. and Simó, J.E. Distributed agent specification to an Intelligent Control Architecture. 6th International Workshop on Practical Applications of Agents and Multiagent Systems. Salamanca, Spain. pp. In Press. (2007)

Poza et al., 2008a Poza, J.L., Posadas, J.L. and Simó, J.E. Arquitectura de Agentes de Control con Soporte a la Calidad de Servicio. IX Workshop De Agentes Físicos, 2008.

Poza et al., 2008b Poza, J.L., Posadas, J.l. and Simó, J.E. Middleware with QoS support to control intelligent systems. Second International Conference on Advanced Engineering Computing and Applications in Sciences. 2008.

Poza et al., 2008c Poza, J.L., Posadas, J.L. and Simó, J.E. Benet, G. Arquitectura de Control Jerárquico Inteligente con Soporte a la Calidad de Servicio. XXIX Jornadas de Automática. 2008.

Poza et al., 2008d Poza, J.L., Posadas, J.L. and Simó, J.E. QoS-based middleware architecture for distributed control systems. International Symposium on Distributed Computing and Artificial Intelligence. 2008

Poza et al., 2009a Poza, J.L., Posadas, J.L. and Simó, J.E. From the Queue to the Quality of Service Policy: a Middleware Implementation. International Symposium on Distributed Computing and Artificial Intelligence. 2009

Poza et al., 2009b Poza, J.L., Posadas, J.L. and Simó, J.E. Adding an ontology to a standardized QoS-based MAS middleware. International Symposium on Distributed Computing and Artificial Intelligence. 2009

Poza et al., 2009c Poza, J.L. Posadas, J.L. Revisión de las arquitecturas de control distribuido. Informe Técnico. Universidad Politécnica de Valencia. 2009. HDL® Identifier: 10251/6407

Poza et al., 2010a Poza-Luján, Jose-Luis and Posadas-Yagüe, Juan-Luis and Simó-Ten, José-Enrique, Multi-agent architecture with support to quality of service and quality of control. Proceedings of the 11th international conference on Intelligent data engineering and automated learning. September 2010 Paisley, Scotland. Springer-Verlag.

Poza et al., 2010b Poza-Luján, Jose-Luis and Posadas-Yagüe, Juan-Luis and Simó-Ten, José-Enrique, Quality of Service and Quality of Control Based Protocol to Distribute Agents. International Symposium on Distributed Computing and Artificial Intelligence (Valencia sep.2010). Advances in Soft Computing. Springer Berlin.

Poza et al., 2010c Poza-Luján, Jose-Luis and Posadas-Yagüe, Juan-Luis and Simó-Ten, José-Enrique, Arquitectura multi-agente distribuida con soporte a la calidad de servicio y a la calidad del control. CONGRESO ESPAÑOL DE INFORMÁTICA (CEDI) - XI Workshop of Physical Agents. Valencia, Spain. Setember 07-10, 2010.

Poza et al., 2011 Poza, J.L., Posadas, J.L. and Simó, J.E. A Survey on Quality of Service Support on Middleware-Based Distributed Messaging Systems Used in Multi Agent Systems. International Symposium on Distributed Computing and Artificial Intelligence. DCAI 2011.

Poza, 2009a Poza, J. L. Comunicaciones en los sistemas distribuidos. Informe Técnico. Universidad Politécnica de Valencia. 2009. HDL® Identifier: 10251/6408

Poza, 2009b Poza, J. L. El modelo de arquitectura SWE. Informe Técnico. Universidad Politécnica de Valencia. 2009. HDL® Identifier: 10251/6405

Poza, 2009c Poza, J. L. El modelo de comunicaciones DCPS. Informe Técnico. Universidad Politécnica de Valencia. 2009. HDL® Identifier: 10251/6409

Poza, 2009d Poza, J. L. El papel de la calidad de servicio en las comunicaciones. Informe Técnico. Universidad Politécnica de Valencia. 2009. HDL® Identifier: 10251/6406

PrismTech, 2009 OpenSplice DDS, C Tutorial Guide. PrismTech Limited. 2009

Reese et al., 2002 Reese, G., Yarger, R.J., King, T. Managing & using mysql. ed.: O'Reilly Vlg. GmbH & Co. 2002

Robin and Botts, 2006 Alexandre Robin and Michael E. Botts. Creation of Specific SensorML Process Models. White Paper Earth System Science Center (NSSTC). University of Alabama in Huntsville (UAH). 2006.

Ronda, 2008 Ronda León, Rodrigo; (2008). Arquitectura de Información: análisis histórico-conceptual. No Solo Usabilidad, nº 7, 2008. ISSN 1886-8592

RTI, 2006 RTI Data Distribution Service. Real-Time Innovations, Inc (www. rti.com) 2006

Ruyter et al., 2005 Ruyter, B.E.R. de, Aarts, E.H.L., Markopoulos, P., IJsselsteijn, W.A. (2005). Ambient intelligence research in homelab: Engineering the user experience In W. Weber, J.M. Rabaey, E. Aarts, Ambient Intelligence (pp. 49-62) Berlin: Springer Verlag.

Sánchez et al., 2009 Sánchez, J., Guarnes, M.Á., Dormido, S.: On the Application of Different Event-Based Sampling Strategies to the Control of a Simple Industrial Process. Sensors. 9, 6795-6818. (2009)

Sebastian, 2010 Sebastian, J., http://www.icoup-consulting.com/microdds.html 2010

Shoham, 1997 Shoham, Y. An overview of agent-oriented programming. Software agents, pp. 271 – 290. MIT Press. 1997.

SIDIRELI, 2008 SIDIRELI: Distributed Systems with Limited Resources. Control Kernel and Coordination. Ministerio de Ciencia e Innovación, Gobierno de España. CICYT/MICINN: DPI2008-06737-C02-01/0 (http://wks.gii.upv.es/sidireli/)

Soucek and Sauter, 2004 Soucek, S. Sauter, T. Quality of Service Concerns in IP-Based Control Systems. IEEE Transactions on Industrial Electronics, Vol. 51, No. 6, December 2004

Souto et al., 2004 E. Souto et al. A Message-Oriented Middleware for Sensor Networks. In Proc. of the 2nd Int'l Workshop Middleware for Pervasive and Ad-Hoc Computing (MPAC 04), Toronto, Canada, Oct 2004, ACM Press, pp. 127–134.

Srisathapornphat et al., 2000 C. Srisathapornphat, C. Jaikaeo, C. Shen. Sensor Information Networking Architecture. In Proc. of the Int'l Workshop Parallel, IEEE CS Press, 2000, pp. 23–30.

SSI, 2010 JAVA DDS. Space Software Italia. http://www.ssi.it/ 2010

Stallings, 1999 Stallings, W., (1999). SNMP, SNMPv2, SNMPv3 and RMON 1 and 2. Part I. Network Management Fundamentals. Addison-Wesley.

Stuck and Arthus, 1984 B.W. Stuck and E. Arthurs. A Computer & Communications Network Performance Analysis Primer. Prentice Hall. 1984.

Sun. 2002 Sun Microsystems, Inc. Java(TM) Message Service Specification Final Release 1.1. (2002)

Tang and Yu, 2007 X. Tang and J. Yu, "*Networked control system: survey and directions*," Lecture Notes in Computer. Science, pp. 473–481, 2007

TwinOaks, 2009 CoreDX DDS. Version 2 Information Brief. Twin Oaks Computing, Inc. 2009

Vogel et al., 1995 Andreas Vogel, Brigitte Kerherve, Gregor von Bochmann, Jan Gecsei. Distributed Multimedia and QoS: A Survey. Vol.2,. No. 2, 1995, pp.10-19.

Waldbusser, 2000 Waldbusser, S. RFC 2819 - Remote Network Monitoring Management Information Base. Network Working Group. Lucent Technologies. 2000

Wang et al., 2000 Yi-Min Wang, Wilf Russell, Anish Arora, Rajesh Jagannathan, Jun Xu. Towards Dependable Home Networking: An Experience Report. Dependable Systems and Networks. 2000.

Wang et al., 2008 Wang, M., Cao, J., Li, J., and Das, S.K. Middleware for Wireless Sensor Networks: A Survey. In Proceedings of J. Comput. Sci. Technol.. 2008, 305-326.

Wooldridge and Jennings, 1995 Wooldridge, M., Jennings, N. Agent Theories, Architectures, and Languages: A Survey, in Intelligent. Agents: Theories, Architectures and Languages, LNAI Volume 890., pp 1-39, Springer-Verlag, Heidelburg, Germany.

Xia, F, 2008. Xia, Feng (2008) QoS Challenges and Opportunities in Wireless Sensor/Actuator Networks. Sensors, 8(2), pp. 1099-1110.

Yan et al., 2011 Hehua Yan, Jiafu Wan, Di Li, Yuqing Tu, Ping Zhang. Codesign of Networked Control Systems: A Review from Different Perspectives. In Proceedings of IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems. Kunming, China, March, 2011

Yang, 2006 Yang, T.C. (2006) Networked control system: a brief survey, IEEE Proceedings – Control Theory and Applications, 153(4), 403-4112.

Zhang et al., 2002 Zhang. R., Chenyang, L., Abdelzaher, T.F., Stankovic, J.A. ControlWare: middleware architecture for feedback control of software performance. Proceedings. 22nd International Conference on Distributed Computing Systems, 2002.

Símbolos, siglas y acrónimos

μ	Tasa de servicio
3T	Arquitectura de navegación de robots
ACHE	Arquitectura domótica
Aladdin	Arquitectura domótica
ATLANTIS	Arquitectura de navegación de robots
AuRA	Arquitectura de navegación de robots
BPS	Bits Per Second
C@sa	Arquitectura domótica.
CCT	Control Component Tree
COM	Component Object Model
CORBA	Common Object Resource Broker Architecture
DCE	Distributed Computing Environment
DCOM	Distributed COM
DCPS	Data Centric Publish Subscribe.
DDE	Dynamic Data Exchange
DDS	Data Distribution Service for Real-time Systems
DLRL	Data Local Reconstruction Layer
DomoNet	Arquitectura domótica
Dr	Data Reader
Dw	Data Writer
E[]	Promedio
EBC	Event Based Control
ERSP	Evolution Robotics Software Platform
FIFO	First In First Out
FIPA	Foundation for Intelligent Physical Agents
FSA	Frame Sensor Adapter
FSACtrl	FSA with Control
HAS	Arquitectura domótica
HomeAPI	Arquitectura domótica
IAE	Integrated Absolute Error
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IPC	Inter-Process Toolkit
ITAE	Integrated in Time Absolute Error
ITU	International Telecommunication Union
JMS	Java Message Service
KERTROL	Núcleo De Control En Los Sistemas Empotrados Fuertemente Interconectados
Ls	Listener
LAAS	Arquitectura de navegación de robots
LISP	LISt Processing
LNT	Logical Namespace Tree
Ctrl	Logical Sensor
LSG	Logical Sensor Graph
MASSIHN	Arquitectura domótica
MavHome	Arquitectura domótica
MFLOPS	FLoating point OPerations per Second
MIPS	Million Instructions Per Second

MOM	Message Oriented Middleware
MPI	Message Passing Interface
msg	Mensaje
MTTF	Mean Time to Failure
NASA	National Aeronautics and Space Administration
NASREM	Arquitectura de navegación de robots
NCS	Networked Control Systems
NML	Neutral Messaging Language
O&M	Observations and Measurements Schema
OGC	Open Geospatial Consortium
OLE	Object Linking and Embedding
OMG	Object Management Group
η	Productividad
Pb	Publisher
PCh	Process Chain
PDV	Packet Delay Variation
PMd	Process Model
PMt	Process Method
PPS	Packets Per Second
QoA	Quality of Agent
QoC	Quality of Control
QoS	Quality of Service
R	Reader
RFC	Request For Comments
RMI	Java Remote Method Invocation
RMON	Remote Network MONitoring
RPC	Remote Procedure Call
RTC	Real-Time Communications
S	Tiempo de Servicio
SAS	Sensor Alert Service
Sb	Subscriber
SCS	Sensor Collection Service
SensorML	Sensor Model Language
SFX	Arquitectura de navegación de robots
SIDIRELI	Sistemas Distribuidos Con Recursos Limitados. Núcleo De Control Y Coordinación
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SOS	Sensor Observation Service
SPS	Sensor Planning Service
SVM	Secuencia Válida de Mensajes
SWE	Sensor Web Enablement
T	Tiempo
TCX	TCA eXchange
TML	Trasducer Markup Language
TOC	Table Of Contents
TPS	Transactions Per Second
TTP	Time-Triggered Protocol
U	Utilización
UML	Unified Modelling Language

VRM	Ventana de Recepción de un Mensaje
VSM	Ventana de Sombra de un Mensaje
VSSS	Ventana de Sombra de Solicitud de Servicio
VVSS	Ventana Válida de Solicitud de Servicio
W	Writer
WNS	Web Notification Service
WSAN	Wireless Sensor/Actuator Networks
WSN	Wireless Sensor Networks
XML	Extensible Markup Language
δ	Disponibilidad
κ	Capacidad
λ	Demanda de servicio
ρ	Carga
σ	Ocupación