Arquitectura para control de robots de servicio teleoperados

Francisco Ortiz, Bárbara Álvarez, Pedro Sánchez, Diego Alonso
División de Sistemas e Ingeniería Electrónica (DSIE)
Universidad Politécnica de Cartagena
E-mail: {francisco.ortiz | balvarez | pedro.sanchez | diego.alonso}@upct.es

Resumen. *La teleoperación de robots es utilizada para realizar tareas que un operador humano no puede llevar a cabo, bien por la complejidad de las mismas o bien por la necesidad de realizarlas en entornos hostiles. Actualmente, se pueden encontrar en la literatura muchas arquitecturas de control para trabajar con este tipo de sistemas; sin embargo, debido a la gran variabilidad en el comportamiento de los mismos, ninguna llega a cubrir todos los objetivos que se plantea inicialmente. El propósito de este artículo es presentar un nuevo marco arquitectónico (ACROSET) que tiene en cuenta las últimas tendencias en arquitecturas para el control de robots adoptando una aproximación orientada a componentes. Tal aproximación provee un marco común para el desarrollo de sistemas con comportamientos muy diferentes y permite la integración de componentes inteligentes.*

1 Introducción

El propósito de este artículo es presentar brevemente los trabajos desarrollados en torno a la obtención de una arquitectura de referencia para aplicaciones de control de robots de servicio (ACROSET). Dichas aplicaciones se emplean para teleoperar manipuladores, vehículos y herramientas que realizan operaciones de inspección y mantenimiento en entornos hostiles. En general, tales actividades son complejas y no es posible trabajar con sistemas completamente autónomos. Por ello, un operador se encarga de monitorizar y operar los diferentes mecanismos. El sistema recibe órdenes del mismo y lleva a cabo las acciones correspondientes para ejecutarlas.

Aunque se han descrito muchas arquitecturas para control de robots [1], es difícil encontrar ejemplos de los procesos de desarrollo que se han seguido para definirlas. Por otro lado, los métodos de desarrollo basados en casos de uso, principalmente RUP (*Racional Unified Process*) [2] y otros similares aunque son apropiados para definir la arquitectura de un determinado sistema, no son adecuados para definir arquitecturas de referencia. Por ello, en este trabajo se ha seguido la metodología ABD *(Architecture Based Design)* [3] propuesta por el SEI *(Software Engineering Institute)* de la Universidad Carnegie Mellon, completándola con las 4 vistas de Hofmeister [4] y su notación inspirada en ROOM basada en el uso de conectores y componentes. El propósito de dicha metodología es diseñar arquitecturas software para un dominio de aplicación o familia de productos y se basa en:

- Una descomposición funcional de acuerdo con los principios de alta cohesión y bajo acoplamiento y el conocimiento de dicho dominio.
- La realización de los requisitos funcionales y de calidad haciendo uso de patrones adecuados.
- La utilización de plantillas software para definir elementos y responsabilidades comunes a un grupo de componentes así como sus interacciones con la infraestructura.

Así, la metodología ABD propone la descomposición del sistema en subsistemas, de forma que las mismas reglas se pueden emplear para descomponer dichos subsistemas en otros más simples. Como modelo final se obtiene una vista conceptual de la arquitectura en la que quedan identificados los principales subsistemas y las relaciones entre ellos, que son descritas utilizando patrones de diseño.

A continuación se comenta brevemente la caracterización realizada para el dominio de los robots de servicio teleoperados y se presenta una breve descripción de la arquitectura ACROSET [5].

2 Caracterización del dominio: Robots de servicio teleoperados.

Los robots de servicio son sistemas mecatrónicos generalmente diseñados para una aplicación concreta. Sin embargo, a pesar de las diferencias en su estructura física, comparten componentes comunes. La caracterización del dominio de aplicación es el punto de partida para definir requisitos funcionales y de calidad. En resumen y en base a nuestra experiencia las principales características que deben ser consideradas son las siguientes:

- El alto grado de especialización y por lo tanto la gran variabilidad en cuanto a funcionalidad y estructura física.
- Las diferentes combinaciones de vehículos, manipuladores y herramientas.
- La gran variedad de infraestructuras de ejecución (procesadores, enlaces de

comunicación e interfaces hombre-máquina).
- La gran variedad de sensores y actuadores.
- Los diferentes tipos de algoritmos de control: desde algoritmos muy simples a otros extremadamente complejos que hacen uso de estrategias de navegación.
- Los diferentes grados de autonomía, desde sistemas dirigidos completamente por el operador a robots semi-autónomos.
- La presencia de requisitos de tiempo real.
- Las implementaciones pueden ser intensivas en hardware o en software con infinidad de posibilidades intermedias.
- Por último, la seguridad de estos sistemas es un tema que no se puede obviar.

Un análisis más preciso de las diferencias entre sistemas [6] revela que la mayoría de éstas afectan no tanto a los componentes del sistema sino a las interacciones entre ellos. De acuerdo con los puntos anteriores, podríamos considerar que entre las **directrices arquitectónicas**, aquellas que tratan de la variabilidad o modificabilidad, son las más influyentes en el diseño de la arquitectura:

1. Diferentes instanciaciones de la arquitectura deben poder compartir y reutilizar los mismos componentes.
2. Se deben adoptar políticas que permitan una clara separación entre dichos componentes y sus patrones de interacción.
3. La implementación de los componentes de la arquitectura podrá ser hardware o software, pudiéndose tratar de componentes COTS[1]. La implementación podrá realizarse sobre diversas plataformas, incluso distribuidas.
4. Debe ser posible que los sistemas realicen tanto acciones autónomas como teleoperadas o misiones pre-programadas.

3 Un recorrido por la arquitectura

Como indica la primera directriz arquitectónica, debería ser posible que diferentes sistemas compartan los mismos componentes cumpliendo así el requisito de reutilización. Por ello, además de utilizar componentes, puertos[2] y conectores[3], se deben definir las reglas e infraestructura común que permita que dichos componentes sean ensamblados o conectados de diferentes formas, cambiando si es preciso sus esquemas de interacción, como define la segunda directriz.

La tercera directriz arquitectónica identificada establece que los componentes pueden ser implementados en hardware o software, siendo preferible que sean COTS. Para alcanzar este objetivo, será necesario identificar los componentes típicos de este tipo de sistemas, los cuales pueden definirse a **distintos niveles de granularidad.**

Finalmente, la última directriz arquitectónica era la posibilidad de obtener arquitecturas concretas para sistemas con predominio de comportamiento tanto deliberativo como reactivo. Para ello, es necesario separar los dos comportamientos.

La arquitectura propuesta (figura 1) identifica cuatro niveles de diferente granularidad en los que definir los componentes que forman parte de lo que se denomina subsistema *CCAS* (Coordinación, control y abstracción de dispositivos):

- Nivel 1: Características abstractas de componentes elementales, tales como sensores y actuadores.
- Nivel 2: Controlador de Unidad de Dispositivo Simple, **SUC**s *(Simple Unit Controllers)*
- Nivel 3: Controlador de Unidad de Mecanismo, **MUC**s *(Mechanism Controllers)*
- Nivel 4: Controlador de Unidad de Robot, **RUC**s *(Robot controllers)*

Estos niveles se denominan de **abstracción del hardware**, ya que sus componentes podrían ser implementados tanto en hardware como en software. Los componentes más simples modelados por la arquitectura son los sensores y actuadores, que son definidos en el nivel arquitectónico más bajo.

Los SUCs son los componentes definidos en el segundo nivel de la arquitectura, y modelan el control sobre los actuadores a partir de la información sensorial. Pueden definirse por ejemplo para controlar las articulaciones de un mecanismo dado. Un SUC genera las órdenes para un actuador a partir de la orden que recibe de otro componente a través de un puerto, la información sensorial que describe el estado del actuador, y su propia política de control, la cual puede ser modificada. Así, la estrategia de control de una articulación podría ser un tradicional PID y ser cambiada por una estrategia de lógica borrosa.

En el tercer nivel de granularidad están los componentes MUC, que modelan el control sobre un mecanismo global (vehículo, manipulador o herramienta). Un MUC es una entidad lógica responsable de coordinar los SUCs, de acuerdo a la información recibida y su propia estrategia de coordinación. Al igual que en el caso anterior, dicha estrategia puede ser modificada. Por ej, puede ser una solución de su cinemática inversa, una estrategia de navegación, etc.

[1] Componentes comerciales

[2] Un puerto proporciona una manera regular de intercambiar datos entre componentes, e independientemente de su funcionalidad y granularidad, proporcionan una puerta a los servicios que los componentes ofrecen y requieren.

[3] La conexión entre puertos (y en consecuencia, entre componentes) se hace a través de los conectores. Al igual que el comportamiento funcional del sistema se concentra en los componentes, el control lo hace en los conectores.

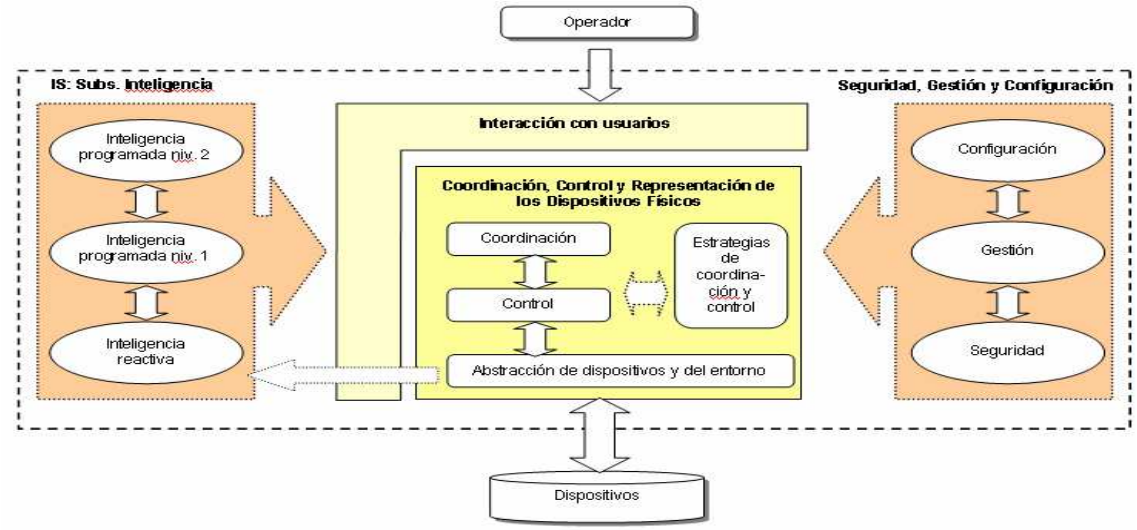

Figura 1. Descripción de alto nivel de la arquitectura ACROSET

Finalmente, la arquitectura define el componente RUC (*Robot Unit Controller*) en el cuarto nivel, para modelar el control de un robot completo, por ejemplo, un robot compuesto por un vehículo, un brazo y varias herramientas. Un RUC es una agregación de MUCs con un coordinador global que genera las órdenes para éstos y coordina sus acciones a partir de la información recibida y de su propia estrategia de coordinación. Dicha estrategia, como en los niveles anteriores es intercambiable.

Esta aproximación es altamente flexible y hace posible integrar inteligencia que no esté directamente relacionada con las misiones de dispositivos robóticos, sino con la gestión de la aplicación como las políticas de tolerancia de fallos, etc.

El subsistema de Interacción con los Usuarios (*UIS*) hace de interfaz entre el *CCAS* y sus usuarios, tanto usuarios externos, como el operador y sistemas de inteligencia externos (sistemas de navegación visión artificial) como el subsistema de Inteligencia (*IS*).

Entre los requisitos del dominio se define la seguridad como un factor importante. En el esquema mostrado en la figura se plantea un subsistema que engloba tres componentes con funcionalidades relativas a seguridad, configuración y gestión. El primero será un componente que monitoriza y diagnostica el correcto funcionamiento del sistema. El segundo será el encargado de gestionar la configuración del resto de los componentes del sistema, y por último un gestor almacenará el diagrama de estado de la aplicación completa y gestionará la viabilidad para realizar ciertas operaciones. Gestionará el arranque del sistema.

4 Conclusiones

La arquitectura descrita recoge los avances más prometedores en el dominio de la teleoperación combinados con la aproximación orientada a componentes, la cual está enfocada a la definición de un marco que permita la reutilización en diferentes sistemas integrando componentes en sistemas inteligentes capaces de conducir el comportamiento del robot. La arquitectura ha sido implementada con éxito para la familia de robots del sistema EFTCoR dedicados a la limpieza de cascos de buques.

Agradecimientos

Este trabajo está parcialmente financiado por la CICYT (TIC2003-07804-C05-02) y la Fundación Séneca (PB/5/FS/02).

Referencias

[1] Coste-Manière, E. et al (2000). Architecture, the Backbone of Robotic System. *Proc. Of the 2000 IEEE Inter. Conf on Robotics & Automation.*

[2] Jacobson I., G. Booch, J. Rumbaugh (1999), The Unified Software Development Process. *Addison-Wesley*, ISBN 0-201-57169-2.

[3] Bachmann F., L. Bass et al. (2001). The Architecture Based Design Method. *Technical Report* CMU/SEI-200-TR-001.

[4] Hofmeister C., R. Nord, D. Soni (2000). Applied Software Architecture. *Addison-Wesley*, ISBN 0-201-3257169-2.

[5] Ortiz F. (2005). Arquitectura de referencia para unidades de control de robots de servicio teleoperados. *PhD Tesis, UPCT.*

[6] Pastor J.A. (2002). Evaluación y desarrollo de una arquitectura software de referencia para sistemas de teleoperación. *PhD Thesis, UPCT.*