Disponible en www.sciencedirect.com
ScienceDirect
ELSEVIER
RIAI
Revista Iberoamericana de Automática e Informática industrial 9 (2012) 24–31
www.elsevier.es/RIAI

# Aplicación de la Arquitectura Orientada a Servicios Universal Plug-and-Play para facilitar la Integración de Robots Industriales en Líneas de Producción

A. Valera[1], D. Juste, A.J. Sánchez, C. Ricolfe, M. Mellado, E. Olmos

*Instituto Universitario de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, España*

**Resumen**

La integración de equipamiento y dispositivos en células robotizadas industriales con tecnologías de interfaz de Ethernet y dispositivos de bajo coste como sistemas de visión, cámaras láser, sensores de fuerza, autómatas programables, PDAs, etc. permite el desarrollo de soluciones potentes e inteligentes. Sin embargo, la programación eficiente de todos estos dispositivos requieren conocimientos muy específicos sobre estos equipos, sus arquitecturas hardware, lenguajes de programación dedicados y detalles sobre protocolos de bajo nivel de comunicación de sistemas.

El artículo describe y utiliza una de las arquitecturas orientadas a servicios más interesante y que mejor se adapta a las células robotizadas: la arquitectura Universal Plug-and-Play. Utilizando esta arquitectura en el artículo se ha desarrollado un banco de pruebas basado en dos robots industriales. Los resultados obtenidos muestran claramente que la utilización de esquemas integrados basados en arquitecturas SOA reduce los tiempos de integración, adaptándose muy bien a la integración de células robotizadas industriales. 

*Palabras Clave:*
programación de células robotizadas industriales, arquitecturas orientadas a servicios, sistemas robotizados, pequeñas y medianas industrias.

## 1. Introducción

Los robots industriales son componentes necesarios en la mayoría de los sistemas de fabricación actuales, para lo que requieren un elevado nivel de integración con el resto de dispositivos y equipos, cada vez más diversos y potentes. Este requisito de capacidad de integración ha generado un reto a la comunidad tecnológica para generar soluciones y sistemas que faciliten la integración y que por tanto, respondan a la demanda creciente en los procesos de producción, principalmente cuando se trata de lotes pequeños de diferentes tipos de productos sin apenas stocks, como suelen trabajar los pequeños fabricantes. En este tipo de producción, la aplicación de los avances tecnológicos recae en las ingenierías e integradores, que tienen una fuerte necesidad de mejorar la eficacia y la flexibilidad mediante la combinación de los tres conceptos siguientes:

- la unificación en la integración de todos los elementos, tanto robots industriales, equipos de fabricación y transporte, dispositivos de entrada y salida, dispositivos de entrada y salida, sensores, etcétera.
- el adecuado interfaz persona-máquina que evite conocer en detalle todos los mecanismos de trabajo a bajo nivel para el control de los distintos elementos.
- la disponibilidad de funciones y métodos de programación que faciliten la integración de todos los elementos enfocándose en la funcionalidad del sistema global, evitando así la dependencia hardware y software de los equipos, como respecto a los lenguajes específicos utilizados, las exigencias particulares de cada uno, etc.

Sin embargo, hoy en día no se ha alcanzado el nivel de flexibilidad deseado en la integración de estos sistemas. Se han encontrado soluciones parciales mediante la especialización en diferentes áreas de aplicación, o dicho de forma más correcta, especialización en los equipos específicos y sus funcionalidades. Pero esta solución está muy limitada, ya que esta forma de trabajar está orientada a resolver las funcionalidades exigidas a cada componente o elemento del sistema, en vez de tratar la funcionalidad global del sistema.

Para las ingenierías e integradores que aplican estas tecnologías en los usuarios finales, incluso a veces para estos últimos, hay varias formas específicas de resolver el problema. La forma más sencilla y común es plantear una arquitectura

[1]Autor en correspondencia
*Correo electrónico:* giuprog@isa.upv.es (A. Valera)


doi:10.1016/j.riai.2011.11.003

basada en comunicaciones cliente-servidor donde, mediante la filosofía de petición y servicio de forma coordinada por un sistema central, se distribuyan las tareas y funcionalidades de cada componente del sistema.

Con los avances recientes en los sistemas de comunicaciones informáticas, han aparecido las arquitecturas orientadas a servicios (SOA) que permiten incrementar la independencia entre los componentes software. Una SOA se basa en el uso de componentes autónomos que están comunicados según una filosofía distribuida. La definición de un servicio en la arquitectura se define en el nivel o contexto superior, lo que significa que se ocultan todos los detalles técnicos y de interconexión de la implementación del servicio.

A nivel de dispositivo, la SOA está emergiendo como la forma de abordar la intercomunicación de dispositivos embebidos presentes en los hogares modernos, las oficinas informatizadas y las industrias automatizadas.

La idea principal que está detrás del paradigma SOA es que las empresas ofrezcan como un producto conjunto un equipo junto a sus servicios, como un software modular, en forma de componentes independientes, que permitan el intercambio de datos a través de HTTP que puedan ser utilizados por otros equipos vía Internet sin requerir la intervención humana.

Para el sector industrial, los procesos de fabricación también pueden ser controlados de modo remoto. Esto implica que los mecanismos de integración que se utilicen sean suficientemente robustos y seguros para garantizar el correcto funcionamiento. Generalmente, los robots industriales que se encuentran en una línea de producción se programan sin considerar posibles fallos o defectos en los equipos. Cualquier situación de error se gestiona desde un sistema centralizado que es el encargado de sincronizar las diferentes tareas y los robots. Sin embargo, si un robot u otro equipo tiene una situación de excepción, puede necesitarse mucho tiempo (y dinero) para reconfigurar la cadena de nuevo y sincronizar todos los equipos para reanudar la producción. Para evitar estas pérdidas, una arquitectura orientada a servicios puede ser una solución muy eficiente ya que facilitaría enormemente la configuración y sincronización de los equipos.

Estas ventajas del paradigma SOA son aún mayores cuando las líneas de producción tienen muchos equipos que además disponen de una alta capacidad de procesamiento. Para potenciar y facilitar aún más la integración, se consideran los llamados dispositivos lógicos SOA, que no son un dispositivo físico, sino el conjunto de un dispositivo físico y un bloque uniforme pero reducido de funcionalidades de este dispositivo. De esta forma un equipo puede implementarse como varios dispositivos lógicos SOA, que cada uno ofrece unas funcionalidades específicas, usando el resto de equipos sólo aquellas funcionalidades que le sean interesantes y necesarias.

Para implementar una SOA se parte de que todos los servicios se hacen públicos en el entorno SOA, especificando las funcionalidades que presenta cada uno de ellos. Así, todos los dispositivos lógicos pueden ser utilizados para constituir un programa de alto nivel.

El presente trabajo aborda el desarrollo de células de fabricación basadas en robots industriales mediante la utilización de arquitecturas orientadas a servicios. Para ello se mostrarán las experiencias realizadas con una de las SOA más prometedoras: la arquitectura Universal Plug-and-Play (UPNP), prestando especial atención a la definición y orquestación de servicios.

## 2. Arquitecturas Orientadas a Servicios

### *2.1. Introducción*

Como se ha comentado anteriormente, en la actualidad podemos encontrar en las células robotizadas industriales actuales una gran variedad de dispositivos y componentes mecatrónicos con una gran capacidad de cálculo y autonomía, como por ejemplo robots, sistemas de visión artificial, autómatas programables, etc. (Gou, Luh y Kyovax, 1997), (El-Kebbe, 2000). Habitualmente se trata de sistemas centralizados que utilizan comunicaciones cliente-servidor punto a punto basado en el método de encuesta. La configuración de estos se suele realizar de forma manual, lo que provoca que esto sea una tarea larga, difícil y tediosa (Pires, Godinho y Araújo, 2006).

Para evitar estos problemas y poder obtener mayores prestaciones, la célula industrial debería poder integrar modernas arquitecturas de redes y sistemas de comunicaciones basados en eventos y publicación-subscripción (Pires, 2007). De esta forma se pueden obtener sistemas inteligentes descentralizados donde la configuración se podrá realizar de una forma casi automática y simple. En la bibliografía se pueden encontrar diversos ejemplos que lo ratifican. En (Rekesh, 1999) y (Bettstetter y Christoph, 2000) se puede encontrar un resumen de las características básicas de las arquitecturas SOA propuestas para la definición del nivel de dispositivo. Por ejemplo, el proyecto europeo SIRENA (SIRENA, 2005) mostró las ventajas que se podrían obtener de la utilización de este tipo de arquitecturas en procesos de automatización industrial. En (Veiga, Pires y Nilsson, 2009) se muestra cómo se pueden programar células robotizadas mediante SOA, y en (Ahn et. al, 2005), (Ahn et. al, 2006) y (Nielsen y Chrysanthakopoulos, 2007) se propone la utilización de arquitecturas SOA como middleware de robots en una plataforma móvil industrial.

De las diferentes arquitecturas SOA que se encuentran disponible actualmente se podrían destacar por su relevancia cuatro: *Jini* (Jini, 2007), *Decentralized Software Services Protocol* (DSSP) (Nielsen y Chrysanthakopoulos, 2007), *Universal Plug-and-Play* (UPnP, 2006) y *Device Profile for Web Services* (DPWS) (Schlimmer et. al., 2004).

*Jini* es una arquitectura propuesta por Sun Microsystems que permite el desarrollo de aplicaciones con el lenguaje Java. Para poder proporcionar una independencia de la plataforma, *Jini* genera una máquina virtual con numerosas librerías, lo que provoca que sean necesarios muchos recursos de memoria. Esto lo hace poco indicada para dispositivos muy pequeños.

DSSP es un protocolo SOA que define un modelo de servicio de estilo *Representational State Transfer* (REST) basado también en tecnología web y que forma, junto con *Concurrency and Coordination Runtime* (CCR), la mayor parte de la plataforma de robots *Microsoft Robotics Studio*.

Tanto UPnP como DPWS de basan en tecnología web y protocolos de red estándar como TCP/IP, UDP, HTTP, SOAP, XML, etc. lo que proporciona una independencia tanto de la plataforma como del lenguaje a utilizar. Además, los formatos XML son aceptados y utilizados ampliamente, proporcionando mecanismos modernos de intercambios de datos y comunicaciones.

La implementación de las SOA se puede basar en RPC (*Remote Procedure Call*), que proporciona un enfoque novedoso ofreciendo muchas posibilidades y mejorar drásticamente el desarrollo de aplicaciones para robots (Al-Jaroodi, Mohamed y Aziz, 2010). El diseño de plataformas middleware para la arquitectura SOA es una ayuda muy importante para alcanzar la interoperabilidad y un acoplamiento flexible y de la alta eficiencia, especialmente para aplicaciones que funcionan en entornos heterogéneos. Así, el trabajo de una SOA es establecer tanto las propiedades funcionales como las no funcionales que incluyen escalabilidad, fiabilidad, flexibilidad y garantía de la calidad del servicio (QoS). La Tabla1 muestra las tres capas del software relacionados con ente tipo de arquitecturas, así como los ejemplos más representativos.

Tabla 1: Capas y ejemplos software

| Niveles | Ejemplos |
|---|---|
| SOA | UPnP, DSS |
| Servicios middleware (basados en RCP) | DCOM, CORBA, RMI, SOAP |
| Protocolo de transporte | TCP/IP, HTTP |

La idea que reside detrás del RPC es hacer el desarrollo de los sistemas distribuidos más accesibles a todos los programadores. Los RPC proporcionan al desarrollador una interfaz con el código de comunicación que es lo más cercano posible para simplificar una llamada a procedimiento. Usando este concepto el desarrollador no tiene que escribir código de red, lo que evita que los programadores tengan que ser expertos desarrollando código de red para escribir complejos sistemas distribuidos a través de cualquier número de hosts en una red.

Muchos sistemas han intentado implementar el concepto de RPC de varias maneras. De esta forma podemos encontrar variantes ampliamente extendidas y usadas como por ejemplo CORBA (Common Object Request Broker Architecture), DCOM (Distributed Component Object Model), Java RMI (Remote Method Invocation) o SOAP (Simple Object Access Protocol). Aunque estos sistemas se usan en entornos diferentes, todos comparten el objetivo de hacer más fácil la escritura y el mantenimiento de las aplicaciones distribuidas. La Tabla2 muestra las variantes más utilizadas, el nombre del protocolo y sus características más importantes del RPC.

CORBA es una versión de RPC orientada a objetos. CORBA (más concretamente, GIOP) usa conexiones TCP/IP para transmitir datos. DCOM fue el mayor competidor de CORBA. Hace algunos años los partidarios de estas tecnologías vieron en ellas un modelo de código y reutilización de servicios en internet. Sin embargo, si el cliente dispone de un cortafuegos o el servidor proxy muy restrictivo que sólo admite conexiones HTTP, la comunicación puede ser imposible.

En (Juric, et. al., 2004), se analizó y comparó el funcionamiento de RMI. Aunque éste proporcionó un funcionamiento mucho mejor que los servicios web, las alternativas de ajuste ofrecen unas prestaciones reducidas. Además, los servicios web se tendrían que utilizar si no es posible abrir los puertos de comunicaciones en aplicaciones distribuidas.

Tabla 2: Variantes del *Remote Procedure Call*

| RCP | Protocolo | Características |
|---|---|---|
| CORBA | Inter-ORB GIOP | Cuando se obtiene, la referencia a un objeto CORBA permite la comunicación directa cliente-servidor. Comunicación muy rápida. No es escalable |
| DCOM (Windows) | ORPC | Requiere de varias iteraciones para activar y usar el objeto remoto. Una vez que la referencia del objeto se ha obtenido, el acceso a objetos sin DCOM puede llevarse a cabo directamente desde el cliente. No es escalable. |
| RMI (Java) | JRMP | Buen funcionamiento. Solo funciona con el lenguaje Java. Relativamente escalable. |
| SOAP (sucesor de XML-RPC) | Cualquier protocolo de transporte | Actualmente con limitadas prestaciones. Sobrecarga de extracción sobre el SOAP, análisis de XML, creación de objetos adecuados y conversión de parámetros. Es escalable. |

SOAP es una especificación que permite las comunicaciones entre aplicaciones (Yilmaz, 2006). Los dos objetivos más importantes para SOAP son la simplicidad y la extensibilidad. Para ello trata de alcanzar estos objetivos omitiendo, en el marco de los mensajes, características que habitualmente se encuentran en sistemas distribuidos. Además, ese trata de una especificación estándar abierta que se basa en tecnologías abiertas como XML y HTTP, por lo que es lo suficientemente versátil como para permitir diferentes protocolos de transporte. SOAP es más lento que CORBA, RMI o DCOM. Sin embargo SOAP modela correctamente túneles sobre HTTP en cortafuegos o proxies.

Como se puede comprobar en (Qiang, Danyan y Xiaohong, 2009), el ciclo de integración para el desarrollo de aplicaciones distribuidas de sistemas robotizados basadas en servicios de middleware suele ser largo y tedioso. En este trabajo se desarrolló y analizó un sistema robotizado que utiliza servicios basados en SOA en el marco orientado a servicios distribuidos. Comparando con el trabajo existente, el sistema robotizado soporta mecanismos de publicación/ suscripción, y puede encontrar de forma precisa los servicios y dispositivos.

### *2.2. Arquitectura UPnP*

En este trabajo se plantea el desarrollo de servicios y aplicaciones basados en la arquitectura UPnP. Se ha optado por esta arquitectura porque ha sido probada y validada durante bastante tiempo en entornos de oficina, lo que proporciona una gran variedad de herramientas de desarrollo disponibles. Además, en (Ahn et. al, 2006) o (Veiga, Pires y Nilsson, 2007) se pueden encontrar ejemplos donde se utiliza esta arquitectura para el desarrollo de células robotizadas industriales.

Los elementos básicos de una red UPnP son: servicios, dispositivos y puntos de control. Un servicio es una unidad de funcionalidad que proporciona un conjunto de acciones y que tiene un estado definido por un grupo de variables de estado. Un dispositivo es un contenedor de servicios y/o otros dispositivos. Por último se tienen los puntos de control, que son los que solicitan los servicios invocando una acción del servicio o un determinado evento de una variable.

Los pasos básicos que componen la configuración del UPnP son:

- *Direccionamiento*: los puntos de control y los dispositivos obtienen una dirección IP válida
- *Descubrimiento*: los puntos de control encuentran los dispositivos de interés
- *Descripción*: los puntos de control obtienen las descripciones de los servicios que ofrecen los dispositivos
- *Control*: los puntos de control invocan las acciones de los dispositivos
- *Notificación*: los servicios anuncian cambios en su estado
- *Presentación*: monitorización del estado de los dispositivos mediante la interfaz HTML

Para obtener la dirección (IP) normalmente se utiliza el protocolo de configuración dinámico DHCP (Protocolo de configuración dinámica de host). Una vez los servicios y los puntos de control tienen las direcciones se puede establecer el descubrimiento. Para ello los dispositivos ofrecen sus servicios a los puntos de control, y los puntos de control encuentran a los dispositivos en la red. El paso de la descripción permite a los puntos de control obtener la información necesaria de los dispositivos. Esto se hace mediante la URL proporcionada por los dispositivos en los mensajes de descubrimiento. En este estado el punto de control ya tiene toda la información sobre las acciones y las variables de estado que los dispositivos ofrecen, lo que permite a los puntos de control ejecutar las acciones en el estado de control. Cuando el estado de un servicio (modelado por sus variables de estado) cambia, los servicios publican estos cambios mediante mensajes en la red, mensajes que se expresan también en XML con el formato utilizado por la arquitectura de notificación de eventos GENA. En el caso de los dispositivos que ofrecen páginas web, el punto de control puede obtener la página a partir de la URL especificada, cargarla mediante un navegador y, dependiendo de la página, permitir incluso ver el estado del dispositivo y establecer su control.

## 3. Desarrollo de Servicios UPnP para Aplicaciones basadas en Robots Industriales

### 3.1. Introducción

En este apartado se va a presentar los métodos y las herramientas utilizadas para el desarrollo de servicios UPnP para el desarrollo de aplicaciones basadas en robots industriales. Para ello se mostrarán primero dos herramientas que permiten la generación automática de los dispositivos de los servicios. Una vez obtenidos los dispositivos únicamente hay que programar las funcionalidades de los servicios ofrecidos por dichos dispositivos. Para ello se utilizan dos herramientas que permiten la programación de los robots industriales ABB y FANUC.

### 3.2. Desarrollo de Servicios UPnP

Aunque es posible encontrar otras soluciones, en este trabajo se han utilizado *Intel® Tools for UPnP™ Technology* para el desarrollo de los distintos servicios [Intel, 2009]. Basadas en Microsoft .NET Framework, estas herramientas de software libre ayudan a los diseñadores y programadores de hardware y software a desarrollar, testear y utilizar servicios UPnP de una forma muy rápida, simple y casi sin necesidad de programar.

Si bien el paquete tiene disponible 11 herramientas, para el desarrollo de este trabajo se ha utilizado la aplicación *Service Author*, que genera de forma automática el formato XML estándar que permite la creación y la ejecución del servicio. Con ella se pueden especificar las variables de estado que permitirán los argumentos de entrada y salida de los servicios. A partir de estas variables de estado se pueden definir las acciones asociadas a los servicios.

La Figura 1 muestra el aspecto que tiene la aplicación *Service Author*. En este caso se trata de un servicio que permite realizar conversiones digital/analógico mediante una tarjeta de adquisición de datos. Para la generación del fichero XML del servicio se ha definido una acción (Escribir_Tarjeta) que tiene 2 parámetros de entrada: el valor de la tensión a convertir y el canal de salida que se desea utilizar.

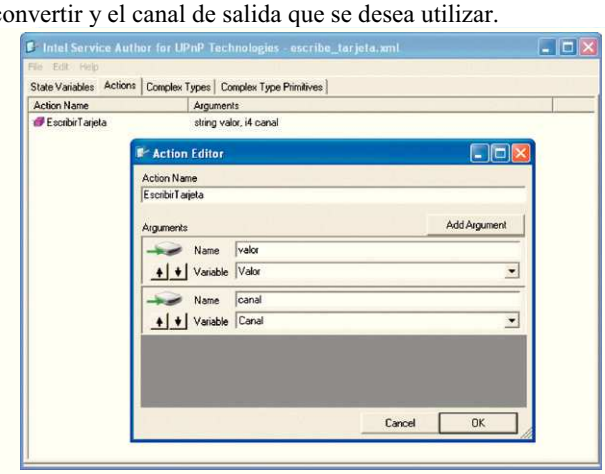


Figura 1: Herramienta *Service Author* utilizada para la generación automática de la descripción XML del servicio.

Una vez se ha generado el fichero XML del servicio sólo se tiene que generar el código para la implementación del dispositivo que va a ofrecer el servicio. Para ello en este trabajo se propone una segunda aplicación de software libre: *Intel Device Builder for UPnP Technologies*, de la librería *Intel® Digital Home Device Code Wizard* (Intel, 2010). Se trata de una aplicación que permite generar dispositivos y puntos de control para diferentes plataformas a partir de las descripciones XML de los servicios generados con la aplicación *Service Author*.

Para exportar el dispositivo deseado con los servicios, *Device Builder* genera de forma automática código C portable, teniéndose que especificar únicamente la plataforma de destino deseada, la ruta donde se generará el código y el espacio de nombres del dispositivo.

Aunque se puede generar código para distintas plataformas (como Linux, Windows, PocketPC, etc.), en este trabajo se ha escogido la solución de .NET Framework y C# porque es el entorno de programación con el que se trabaja en el grupo de investigación para el desarrollo de las distintas aplicación de programación y control de robots industriales. La Figura 2 muestra el aspecto que tiene la herramienta *Device Builder*.

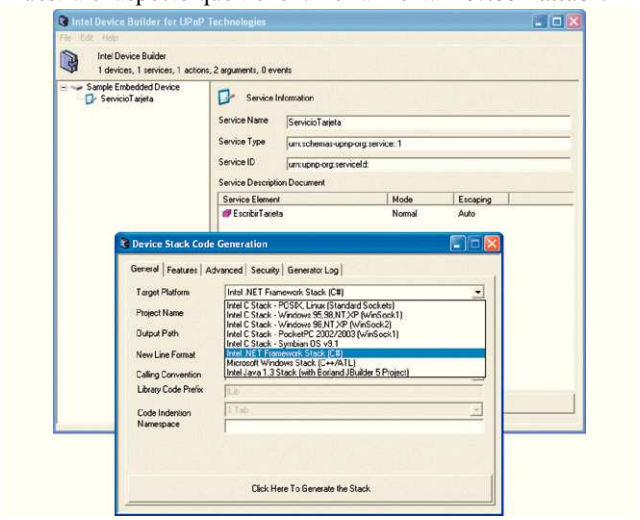


Figura 2: Herramienta *Device Builder* utilizada para la generación automática de dispositivos UPnP.

### *3.3. Entornos de Programación de Robots Industriales*

Por la selección de plataforma realizada, *Device Builder* generará un proyecto .NET con el código necesario para poner en marcha y parar el servicio, detectar cuándo se hace una llamada a dicho servicio, etc. Lo único que resta por hacer es programar el código que implemente la funcionalidad requerida del dispositivo. Para ello básicamente solo hay que modificar uno de los ficheros del proyecto .NET que se genera: el fichero *DVImportedService.cs*.

Para el desarrollo de las funcionalidades de los servicios basados en los robots industriales se han utilizado 2 entornos: *Robot Application Builder* y *PCDK*.

*Robot Application Builder* (RAB) es un kit de desarrollo de software (SDK) para robots de ABB. RAB utiliza la herramienta Microsoft® VisualStudio .NET para crear interfaces de operador personalizadas tanto para PC como para FlexPendant y permiten al programador desarrollar aplicaciones de control de robots industriales, acceder a las señales de los mismos, modificar el valor de las variables de un programa en código RAPID, cargar programas, mover el robot a través de programas RAPID, obtener las características del controlador, acceder al sistema de ficheros del IRC5 y todas las acciones que se puedan realizar sobre un robot industrial.

RAB está basado en una jerarquía de clases .Net, permitiendo una programación (C#.Net o VB.Net) más cómoda y sencilla donde cada una de estas clases representa una entidad del robot. De esta forma, por ejemplo, para acciones que requieren un cierto grado de seguridad como el acceso a las variables de un programa RAPID cargado en el robot industrial hay una clase (*Mastership*) que controla los accesos de los usuarios y comprueba sus derechos de acceso y control sobre el robot y el controlador IRC5, evitando así que dos usuarios puedan acceder a la vez a una variable.

PCDK es el segundo entorno de desarrollo utilizado. Se trata de una herramienta muy potente que permite una gran eficiencia de comunicación de información e instrucciones entre el PC y los controladores de los robots FANUC, permitiendo el desarrollo de aplicaciones en lenguajes .Net, como C# o VB.

El PCDK proporciona servidores de robots, de manera que se permite el acceso a los controladores de los robots FANUC, lo que permite escribir y leer variables KAREL, escribir y leer registros numéricos de TPE, acceder y configurar las señales de entrada y salida del robot, cargar, guardar y lanzar la ejecución de programas del robot y monitorizar su estado, acceder a posiciones del robot y generar trayectorias de movimiento, monitorización de alarmas, manejo de eventos etc.

## 4. Desarrollo de un Banco de Pruebas Experimental para Células Robotizadas

En este apartado se va a presentar un banco experimental que se ha desarrollado que permite validar a las arquitecturas SOA como una solución general para la programación de células robotizadas industriales.

El demostrador propuesto es una estación de inspección y envasado automático basado en robots industriales. El demostrador está compuesto básicamente por los siguientes elementos:

- Robot ABB IRB 140 equipado con la unidad de control IRC5
- Robot FANUC LR Mate 200Ib equipado con la unidad de control R-J3iB.
- 2 cintas transportadoras controladas por variadores de frecuencia y equipadas con sensores de presencia y encoders.
- 2 webcam ubicadas encima de las cintas transportadoras.

La funcionalidad de la estación se puede resumir de la forma siguiente: la primera cinta transporta piezas a inspeccionar. Se trata de piezas cilíndricas de dimensiones iguales pero de dos colores distintos. Cuando éstas llegan al primer sensor de presencia la webcam toma una imagen y se analiza. En el caso de que se detecte la existencia de objetos que no pasan el test de calidad (en función de su color) se envía al primer robot (IRB140 equipado con una garra de dedos) un vector con las posiciones de los centroides de las piezas que tienen que ser retiradas de la cinta.

Las piezas correctas pasan automáticamente a una segunda cinta transportadora en cuyo extremo se encuentra una segunda webcam. Ésta se encarga de tomar otra imagen para detectar las posiciones de las piezas. Esta información se envía al robot FANUC LR Mate que es el encargado ir cogiendo las piezas una a una para realizar el envasado de las mismas.

La Figura 3 muestra los dispositivos utilizados para el desarrollo del banco experimental.

Para el desarrollo de la aplicación basada en la arquitectura SOA se han desarrollado 5 aplicaciones software correspondiéndose con los 5 dispositivos UPnP de la red (ABB_IRB_140, FANUC, Conveyor, Camara y Sistema). Los distintos computadores encargados de la ejecución de las aplicaciones software están conectados a una red de área local Ethernet. Se trata de una red de alta

velocidad con switches CISCO serie 2960 con puertos Fast-Ethernet con velocidad de 1Gbps. La Figura 4 muestra las dependencias tecnológicas que se tienen entre los distintos componentes del banco de pruebas experimental.

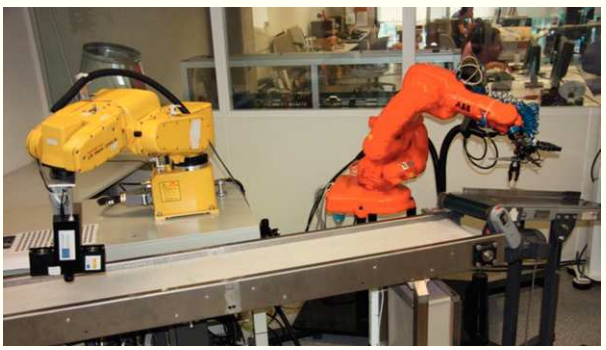

Figura 3: Banco de pruebas experimental.


Figura 4: Dependencias tecnológicas del banco de pruebas.

La Figura 5 muestra parte de los dispositivos UPnP diseñados y los servicios que éstos proporcionan.


Figura 5: Servicios UPnP diseñados.

Puesto que los robots industriales del sistema no tienen soporte nativo UPnP, es necesario desarrollar una capa software adicional para integrarlos como dispositivos y servicios estándar UPnP en la red. De esta forma, el dispositivo *ABB_IRB_140* se implementó con una aplicación software que comunica con el controlador del robot ABB mediante una red TCP/IP. El controlador del robot ejecuta una aplicación servidora desarrollada en RAPID (ABB, 2005). El dispositivo UPnP proporciona diversos servicios que aceptan una posición, una configuración (posición y orientación) o un vector de posiciones a las que deberá ir el robot, leer o escribir el contenido de una variable RAPID o leer una señal digital.

El dispositivo *FANUC* es un dispositivo análogo al anterior, desarrollándose en este caso para la programación y el control del robot FANUC. Para este dispositivo se han programado los servicios para enviarle al robot una configuración o un vector de posiciones a las que debe ir, y el servicio de leer el contenido de una variable. Además, se ha tenido que desarrollar una aplicación KAREL que se ejecuta en el controlador del robot. La aplicación se comporta como un servidor, de forma que una vez conectado el cliente (el dispositivo UPnP), espera recibir por el socket la información relativa a, por ejemplo, las posiciones a donde éste debe desplazarse.

El dispositivo *Conveyor* se implementó también como una aplicación software para el control de las cintas transportadoras. Cada una de las cintas está accionada mediante un motor de corriente alterna accionado con un variador de frecuencia (Altivar-31, de Telemecanique). El control de las cintas se realiza mediante un PC equipado con una tarjeta de adquisición de datos PCI-1720 de Advantech. Gracias a los canales de las salidas digitales que tiene esta tarjeta se puede poner en marcha y parar las cintas, así como especificar el sentido y la velocidad de avance de éstas. La programación para el acceso a la tarjeta se ha realizado mediante el lenguaje C#.

El dispositivo *CamaraIns* se programó en C# para controlar el acceso a las cámaras comerciales Logitech Webcam ubicadas en las cintas de inspección y envasado. Éste devuelve el número y la posición de las piezas en dichas cintas.

El último dispositivo es un servicio que se utiliza para realizar la inicialización y la finalización de la célula robotizada, y se encarga por ejemplo de cargar y arrancar los programas que se ejecutan en los robots industriales. Además, también tiene un servicio que permite la finalización del sistema.

Una vez generados los servicios de los distintos dispositivos que forman parte de la célula robotizada, para el desarrollo de aplicaciones industriales sólo hace falta realizar una planificación y sincronización de las distintas tareas de alto nivel programadas mediante los servicios. Para este trabajo se ha programado una aplicación software escrita en Java que permite la orquestación de los servicios de la célula. Dicha aplicación básicamente es un punto de control UPnP con algunas herramientas adecuadas para la construcción de pilas, donde la pila representa el flujo de

control de las tareas que se van utilizar para el desarrollo de la aplicación industrial.

Cuando se arranca el orquestador aparecen 2 ventanas. En la ventana de *Orquestación de servicios* (Figura 6) se pueden buscar los servicios disponibles en ese momento en la red pulsando el botón "*Buscar*…". En la misma ventana se mostrarán los distintos servicios descubiertos. Para seleccionarlos y añadirlos a la lista de servicios que se desean ejecutar sólo hay que pulsar el botón "*Añadir*".

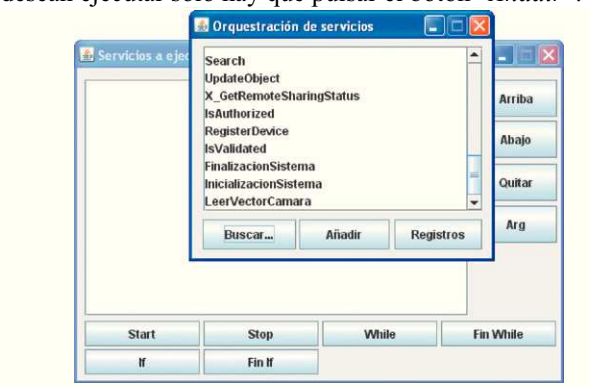


Figura 6: Aplicación de orquestación desarrollada.

Existe un tercer botón (*Registros*) que cuando se pulsa se abre una nueva ventana en la que pueden modificar el valor de los registros que proporciona la aplicación para el paso de parámetros entre servicios, proporcionando variables de tipo booleano, entero, real, posición, configuración y vectores.

Conforme se van seleccionando los servicios, éstos aparecerán en la ventana principal Servicios a ejecutar. Mediante los botones "*Arriba*" y "*Abajo*" se puede cambiar el orden de la lista de ejecución de los servicios. De la misma forma, si se desea eliminar un servicio de dicha lista, se debe seleccionar y pulsar a continuación el botón "*Quitar*".

Con el botón "*Arg*" se pueden asignar los argumentos de entrada y salida del servicio seleccionado, especificando el tipo y el registro a utilizar. La Figura 7 muestra la ventana que se abre cuando se seleccionar el botón de los argumentos. En ésta se puede apreciar cómo se pueden especificar los parámetros de entrada y salida del servicio, lo que permite comunicar un servicio con otro mediante los valores de los registros.

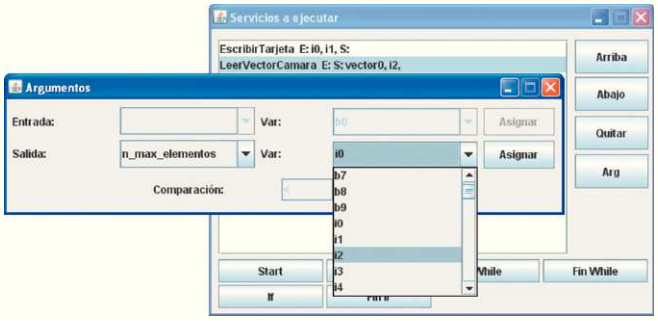


Figura 7: Ventana de asignación de parámetros de los servicios.

Los botones de la parte inferior de la ventana de los servicios a ejecutar permiten arrancar y parar la ejecución de los servicios. Además, el orquestador proporciona también la posibilidad de establecer bucles y verificar condiciones mediante las típicas estructuras de control de flujo *while* e *if*.

La Figura 8 muestra un ejemplo simple de orquestación de servicios para la estación de inspección de la célula robotizada. En ella se puede observar cómo se llama a la inicialización del sistema y pone en marcha la primera cinta transportadora. Después se entra en un bucle *while*, cuya finalización se controla mediante una señal digital del robot. Esta señal estará controlada por el usuario que es el que indica cuándo finaliza la ejecución de la célula robotizada.

En del bucle se toma una imagen y se analiza, proporcionando el vector de las posiciones y número de los objetos que son necesarios retirar. Posteriormente esta información es suministrada al robot para que realice los movimientos necesarios para eliminar dichos objetos, para evitar que lleguen a la estación de envasado.

Por último se hace una llamada para la finalización del sistema, encargándose por ejemplo de parar la ejecución del programa RAPID del robot y parar la cinta transportadora.


Figura 8: Ejemplo de orquestación de servicios.

## 5. Conclusión

El objetivo principal de este artículo ha sido describir una de las arquitecturas SOA propuestas recientemente y verificar su utilidad en aplicaciones industriales reales como es la programación de células robotizadas. Para ello se ha diseñado un banco de pruebas experimental para implementar una de las tecnologías SOA más prometedora: UPnP.

Centrándose en la automatización industrial y especialmente en la programación de células robotizadas industriales, UPnP puede permitir a los ingenieros de control concentrarse en sus habilidades y experiencias (visión por computador, programación de robots, control de fuerza, etc.) utilizando el lenguaje/plataforma preferido. Además, como se basa en tecnologías estándar, les permite dedicar menos tiempo y atención a tareas complejas de interconexión de los distintos dispositivos que conforman la célula robotizada.

Al poderse basar en servicios previamente implementados, se ha comprobado que la programación de este tipo de células industriales utilizando arquitecturas

SOA para especificar la lógica del sistema no sólo permite reducir el tiempo necesario para realizar la configuración y programación de aplicaciones industriales, sino que además permite que esta programación se pueda realizar de una forma muy fácil e intuitiva por incluso usuarios no expertos en la materia. Además, se pueden encontrar y utilizar soluciones desarrolladas para los componentes más importantes de las células, como robots, cámaras, autómatas, sensores inteligentes, etc.

La solución propuesta permite también extender el concepto de plug-and-play de las arquitecturas orientadas a servicios al concepto plug-and-produce sin más que añadir los dispositivos al conjunto de servicios que corresponde a las funcionalidades de la célula en particular. Así, los sistemas de fabricación deben prepararse para usar los nuevos servicios para empezar o mantener la productividad. De esta forma, el trabajo presentado ha mostrado que la tecnología UPnP tiene características interesantes que casan bien con el entorno industrial y las necesidades de soluciones de programación de alto nivel.

## English Summary

### Development of Industrial Robots Based Applications by SOA Universal Plug-and-play Architecture.


### Abstract

The equipment integration and devices in industrial robotic cells with Ethernet interface technologies and low cost devices such as vision systems, laser cameras, force sensors, programmable controllers, PDAs, etc. allows the development of powerful and intelligent solutions. However, the efficient programming of these devices require specific knowledge about them, their hardware architecture, specific programming languages and details of specific low-level protocol communication systems. The article describes and uses a service-oriented architecture: Universal Plug-and-Play. It is one of the most interesting and best adapted to robotic cells. Using this architecture, in this article it has been developed a test based on two industrial robots. The results show clearly that the use of schemes based on SOA reduces integration times, adapting very well to the integration of industrial robotic cells.



*Keywords:*

industrial robotic cells programming, service-oriented architecture, robotic systems, small and medium companies.


## Agradecimientos


Los autores desean expresar su agradecimiento al Plan Nacional de I+D, Comisión Interministerial de Ciencia y Tecnología (FEDER-CICYT) por la financiación parcial de este trabajo bajo los proyectos de investigación DPI2009-13830-C02-01 y DPI2010-20814-C02-02.


## Referencias


ABB, 2005. ABB IRC5 Documentation, ABB Flexible Automation, Merrit.

Ahn, S. C., Kim, J.H., Lim, K.W., Ko, H., Kwon, Y.M., Kim, H.G., 2005. UPnP Approach for Robot Middleware, Proc. of the 2005 IEEE International Conference on Robotics and Automation, Barcelona (Spain), p. 1959-1963.

Ahn, S. C., Lee, J.W., Lim, K.W., Ko, H., Kwon, Y.M., Kim, H.G., 2006. UPnP SDK for Robot Development. Proc. of the SICE-ICASE Int. Joint Conference, Busan (Corea), pp. 363-368.

Ahn, S. C., Lee, J.W., Lim, K.W., Ko, H., Kwon, Y.M., Kim, H.G., 2006. Requirements to UPnP for Robot Middleware. Proc. of the 2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Beijing (China), pp. 4716-4721.

Al-Jaroodi, J., Mohamed, N., Aziz, J., 2010. Service Oriented Middleware: Trends and Challenges. Information Technology: New Generations, Third Int. Conf. on, 2010 Seventh International Conference on Information Technology, pp. 974-979.

Bettstetter, C., Christoph, R., 2000. A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol. Sixth EUNICE Open European Summer School, Twente (Netherlands).

El-Kebbe Salaheddine, D. A., 2000. Towards a Manufacturing System under Hard Real-Time Constraints. In Informatik 2000: 30. Jahrestagung der Gesellschaftf˙ur Informatik, Berlin.

Gou, L., Luh, P., Kyoyax, Y., 1997. Holonic Manufacturing Scheduling: Architecture, Cooperation Mechanism, and Implementation, IEEE/ASME International Conf. on Advanced Intelligent Mechatronics Vol. 37, p. 213-231.

Intel. http://software.intel.com/en-us/articles/intel-software-for-upnp-technology-download-tools, 2009.

Intel. http://intel-r-digital-home-device-code-wizard.software.informer.com, 2010.

Jini, The Community Resource for Jini technology: http:www.jini.org. 2007.

Juric, M.B., Kezmah, B., Hericko, M., Rozman, I., Vezocnik, I., 2004. Java RMI, RMI tunneling and Web services comparison and performance analysis, ACM Sigplan Notices, 39 (5), 58-65.

Nielsen, H., Chrysanthakopoulos, G., 2007. Decentralized Software Services protocol – DSSP/1.0 July.

Pires, J. N., Godinho, T., Araújo, R., 2006. Controlo e Monitorização de Células Robotizadas Industriais, Robótica. Abril.

Pires, J. N., 2007. Industrial Robots Programming Building Applications for the Factories of the Future. Springer.

Qiang, Z., Danyan, C., Xiaohong, C., 2009. An Approach to Constructing Event-Driven Virtual Enterprise Based on SOA. Int. Forum on Computer Science-Technology and Applications, pp. 443-446.

Rekesh, J., 1999. UPnP, Jini and Salutation A look at some popular coordination frameworks for future networked devices: California Software Labs.

Schlimmer, J., Chan, S., Kaler, C., Kuehnel, T., Regnier, R., Roe, B., Sather, D., Sekine, H., Walter, D., Weast, J., Whitehead, D., Wright, D., 2004. Devices Profile for Web Services: A Proposal for UPnP 2.0 Device Architecture. Available: http://xml.coverpages.org/ni2004-05-04-a.html.

SIRENA Project (2005), Service Infrastructure for Real-time Networked applications, Eureka Initiative ITEA. www.sirena-itea.org.sadasd.

UPnP forum Available: http://www.upnp.org.

Veiga, G., Pires, J.N., Nilsson, K., 2009. Experiments with Service-Oriented Architectures for Industrial Robotics Cells Programming. Robotics and Computer-Integrated Manufacturing, 25 (4-5), 746-755.

Veiga, G., Pires, J.N., Nilsson, K., 2007. On the use of Service Oriented Software Platfoms for Industrial Robotic Cells. IFAC Int. Workshop Intelligent Manufacturing Systems, Alicante (Spain).

Yilmaz, G., 2006. Comparison of Soap based Technologies: .Net Remoting and Asp.Net Web Services, Journal of Aeronautics and Space Technologies, 2 (4), 23-28.