Tecnologías *Middleware*: soluciones actuales para grandes empresas y proyectos

Felipe García Sánchez, Antonio Javier García Sánchez, Pablo Pavón Mariño, Joan García Haro
Departamento de Tecnologías de la Información y de las Comunicaciones.
Universidad Politécnica de Cartagena
Campus Muralla de Mar. Edificio Antiguo Hospital de Marina
30202 Cartagena
Teléfono: 968326537 Fax: 968 325973
E-mail: {felipe.garcia, antoniojavier.garcia, pablo.pavon, joang.haro}@upct.es

***Resumen**. Este artículo presenta al lector una visión de la programación distribuida con plataformas middleware, su aplicación en el mundo real y los desarrollos llevados a cabo por grandes empresas. Se pretende con ello divulgar ideas poco conocidas de la tecnología y las ventajas competitivas que los middleware ofrecen para la empresa y profesionales del desarrollo de software. Después de una revisión de los trabajos más destacados, se incluyen brevemente los esfuerzos actuales, como punto de contacto a estas tecnologías.*

1 Introducción

Un *middleware* es un *software* situado entre el nivel de aplicación (interfaz de usuario) y los niveles de transporte e inferiores (que suele presentarse como el interfaz o API de programación de los distintos sistemas operativos). El *middleware* permite la adecuación de la aplicación a un lenguaje general entendible por las aplicaciones que comparten ese lenguaje.

Las plataformas *middleware* son en la actualidad una solución muy utilizada para el desarrollo de proyectos de *software* distribuido, que implican la integración de sistemas heterogéneos. Se trata en general de grandes proyectos desarrollados por empresas de gran capacidad tecnológica. Sin embargo, en muchas ocasiones, la complejidad que popularmente se asociaba a estas tecnologías, hacía que los proyectos en entornos locales apostasen por otras opciones. Esta situación está cambiando. En la actualidad, los *middleware* y los servicios que incorporan, hacen que su uso sea fácil de aprender y rápido de desarrollar, implementando aplicaciones para todo tipo de entornos.

El empleo de plataformas *middleware* se basa en diversos factores: integrabilidad de distintos lenguajes de programación, independencia de las plataformas de comunicaciones, programación basada en patrones y coherencia en la estructura de las aplicaciones. Además, las distribuciones suelen incluir otras interesantes propiedades como seguridad, transmisiones *multicast*, sistemas de transmisión de audio y vídeo, entre otras. La existencia de unas u otras dependen de la plataforma y de la distribución utilizada.

Estas características contribuyen a que los *middleware* sea un software actual, en continua evolución y mejora y que vaya extendiendo su rango de influencia. Actualmente, los entornos militares, médicos, tele-vigilancia, aeronáuticos y aeroespaciales, son los más aplicados para utilizar estas tecnologías. En un futuro no es descartable que éstas lleguen a utilizarse en aplicaciones cotidianas como navegadores o correo electrónico.

Entre las tecnologías middleware más extendidas, podemos destacar tres: CORBA (*Common Object Request Broker Addressing*) [1], JavaRMI (*Remote Method Invocation*) [2] and .NET *Remoting* [3]. Todas ellas tiene en común que están destinadas a la integración de aplicaciones distribuidas, aunque cada una de ellas con peculiaridades. CORBA es por excelencia la plataforma *middleware* más completa, ya que proporciona integrabilidad completa de lenguajes, redes y sistemas operativos. Incluso dentro de CORBA existen numerosas implementaciones que a su vez permiten interoperatibilidad y compatibilidad entre todas ellas.

Por su parte, JavaRMI es también una plataforma completa en la integración “hacia abajo”, es decir, las redes y sistemas operativos son válidos siempre que soporten la máquina virtual de Java, pero ofrece una restricción importante, el lenguaje de programación ha de ser Java. Además, para distintas aplicaciones Java es un lenguaje en ejecución más lento que otros conocidos como C y C++. Esto limita la aplicabilidad de esta plataforma.

Finalmente, .NET Remoting es la tecnología más reciente. Desarrollada por Microsoft, surge como una actualización de las plataformas COM. Es muy versátil, posee (como JavaRMI) una máquina virtual que compila el código intérprete. Sus limitaciones son el sistema operativo (Windows 2000 y posteriores) y que requiere uno de estos tres leguajes, Visual Basic, C++ ó C#, aunque se está trabajando en otros. El lenguaje C# es curiosamente un lenguaje

específico de .NET *Remoting*, sobre el que se desarrollan de manera más adecuada las aplicaciones.

2 Desarrollos relevantes

En esta sección se describen algunos de los proyectos internacionales más destacables en el campo de los entornos distribuidos. En muchos casos, los desarrollos no son publicables debido a seguridad empresarial y/o militar.

2.1 Proyecto CLIPS

El primer gran proyecto de la NASA en la utilización de CORBA data de 1986 y fue el proyecto CLIPS (*C Language integrated Production System*) [4]. Fue realizado sobre C, el lenguaje más extendido y fiable de aquel momento. Se trataba de aprovechar la capacidad de comunicación de CORBA para establecer el procesado de distintas máquinas distribuidas. El problema a enfrentar consistía en que las aplicaciones CORBA eran casi exclusivamente programadas en lenguaje original y bajo funciones nativas del propio CORBA. El objetivo del proyecto, era ampliar la utilización a funciones externas a CORBA, que se pudieran programar de manera más cómoda y reutilizar el código ya hecho. En la actualidad, todas las implementaciones de CORBA proporcionan este funcionamiento.

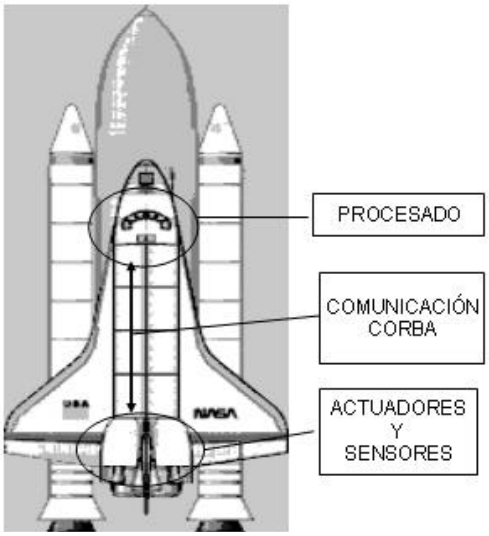

Fig. 1. Los transbordadores espaciales utilizan CORBA para sus procesados distribuidos.

2.2 Proyecto MOOPS

También desarrollado para la NASA, MOOPS (*Mission Operations Planning and Scheduling System*) proporciona un interfaz integrado para la programación distribuida. En base a la toma las decisiones de vuelo, los equipos manejaban gran cantidad de datos, valores, estadísticas, etc. Todo ello debía ser procesado de manera rápida y fiable, dentro de un interfaz gráfico, lo que los equipos de forma individual eran incapaces de realizar. Así, se desarrollo una amplia plataforma compuesta por códigos C/C++, X-Motif, sobre la base de Orbix como distribución CORBA, proporciona al servidor central una capacidad de cálculo casi infinita, pudiendo utilizar todos los ordenadores conectados en la Intranet de la NASA, así como todos aquellos que lo deseen a través de Internet.

2.3 Proyecto CORBAsec

Este proyecto desarrollado también para la NASA desde Julio de 2003, es una importante innovación del sistema CORBA ya que incluye la interacción de las distribuciones Visibroker (válida para el lenguaje Delphi) y Orbix2000. Añade funciones de seguridad de su plataforma HSS (*Hitachi Security Service*), y proporciona sistemas de escuchas y de *masquerading* CORBA entre otras. El nivel de seguridad es variable y se implementa sobre la base de *firewalls* distribuidos. Su objetivo es proponer un sistema de control de la propulsión de forma distribuida, de forma que los cálculos se desarrollen en unas máquinas alejadas de los motores y en ellos sólo se encuentres los actuadores y sensores de datos y registros (Figura 1).

2.4 CORBA como simulador

Otra importante cualidad de CORBA y su programación distribuida es la capacidad de comportarse como un simulador distribuido. Así tanto las empresas Boeing and LockHeed Martin, conocidas en USA por sus capacidades tecnológicas y por sus contratos con la Defensa norteamericana, desarrollan simuladores de sus productos en distintas distribuciones de CORBA. Como ejemplo, el grupo de la Universidad de Arizona e IBM que desarrollan el software para distintas implementaciones como los preyectos SIMPROCCESS (integración de componentes CACI), SCA (simulador de IBM), utilizan simuladores DEVS/CORBA (simuladores discretos) [6] para prever el comportamiento de distintos sistemas.

2.5 CORBA para sistemas empotrados

Diferentes distribuciones de CORBA se están esforzando en mejorar los servicios para sistemas empotrados (*embedded*). Estos sistemas son aquellos que una vez instalados raramente se vuelven a modificar, o requieren una gran inversión para ello. Por ejemplo, los sistemas de los barcos y submarinos, como los rádares, sónares, localización, etc., se suelen instalar durante la construcción del buque, y en algunos casos, no se vuelven a modificar salvo en caso de rotura. Sin embargo, los interfaces de usuarios, suelen modificarse y actualizarse. Por ello, es necesario que exista algún tipo de comunicación estándar para que cualquier nueva aplicación pueda utilizar los sistemas ya instalados. Distribuciones como ACE/TAO y ORBacus, tratan de definir interfaces estándar, de manera que siendo lo más abiertos y generales posibles, faciliten la entrada y salida de datos de forma rápida, segura y eficiente.

2.6 Desarrollos en .NET

La plataforma .NET abarca un campo más específico que las distribuciones CORBA, más generalistas. Así, se suelen emplear para el acceso a páginas *web*, a bases de datos, servidores de correo o simplemente implementar objetos distribuidos que proporcionen servicios a clientes remotos (*.NET Remoting*). Bajo la denominación de *.NET Visual Studio*, se proporcionan una serie de herramientas y librerías que permiten acceso de forma sencilla a entornos ASP.NET, PHP, bases de datos basadas en Oracle, y en general todas aquellas aplicaciones preparadas para comunicación a través del protocolo SOAP (*Simple Object Access Protocol*).

Para mejorar la accesibilidad a las aplicaciones mediante .NET, se están desarrollando interfaces entre los protocolos SOAP y el protocolo de comunicaciones de CORBA, IIOP (*Internet InterORB Protocol)*. De esta forma se consigue acceder a servicios más complejos o ya diseñados a través de aplicaciones .NET mucho más sencillas de programar. Además, los últimos esfuerzos indican que .NET tendrá sus réplicas en otros sistemas operativos, con lo que las limitaciones naturales de esta tecnología quedarán eliminadas. Es reseñable el rápido desarrollo de esta plataforma que, en muy poco tiempo, se ha convertido en un sistema generalizado para la mayoría de empresas que desean distribuir sus bases de datos, información, servicios de correo, etc.

2.7 Desarrollos en JavaRMI

JavaRMI surgió como competencia directa de CORBA, ya que hasta ese momento se desarrollaban las aplicaciones en C ó C++ con todas sus librerías, y no existía interfaz para las aplicaciones Java. De esta forma gracias a su mayor simplicidad de programación, se desarrollaron distintas aplicaciones del tipo de CORBA, generales y comerciales. Con el tiempo, el retardo que imponía la ejecución sobre la máquina virtual de Java y su suscripción exclusiva al lenguaje Java hizo que perdiera fuerza.

Sin embargo, dos hechos han ayudado a que esta tecnología vuelva a estar de máxima actualidad: el aumento de la velocidad de ejecución de los equipos y de las máquinas virtuales, y la adaptación a entornos móviles. Actualmente, JavaRMI es compatible con los objetos Jini que pueden integrarse en equipos móviles como PDA's.

3 Conclusiones y líneas actuales

Este artículo ha repasado distintos proyectos de desarrollo de tecnologías *middleware* por parte de grandes empresas. Las líneas actuales de investigación tienen como campo de actuación la implementación de aplicaciones enfocadas, no solo a grandes proyectos, si no también pensadas a pequeños usuarios. Estas implementaciones están englobadas en dos grandes grupos. El primero es el dedicado a las aplicaciones multimedia, adaptando los modelos existentes para la integración de servicios multimedia a entornos inalámbricos. En un futuro, se espera que CORBA pueda ser la herramienta que integre la movilidad de aplicaciones desde entornos fijos a otros inalámbricos e inalámbricos entre sí.

El desarrollo de aplicaciones multimedia sobre los *middleware* .NET *Remoting* y JavaRMI, permiten la transmisión de flujos de vídeo y amplían los actuales servicios de estas plataformas. Conjuntamente, se han implementado controles multimedia que facilitan la operación de controles remotos, utilizando ambas plataformas y entorno Windows.

El segundo grupo consiste en el diseño de un *middleware* CORBA específico para el desarrollo de aplicaciones *peer-to-peer*. Dichas aplicaciones permiten una mayor interoperatibilidad de usuarios, independientemente del programador o fabricante de dichas aplicaciones.

Referencias

[1] Object Management Group. "CORBA, Basics", http://www.omg.org/gettingstarted/corbafaq.htm

[2] Sun Developer Network, "Java Remote Method Invocation"http://java.sun.com/products/jdk/rmi

[3] David Conger, "Remoting with C# and .NET", Wiley Publishing, Inc 2003.

[4] Software Technology Branch, Lyndon B. Johnson Space Center, "CLIPS Reference Manual: Volume I Basic Programming Guide, CLIPS Version 6.0", 1993.

[5] NASA Glenn Research Center "NPSS Corbasec Test Bed", DOC Sec 2001.

[6] P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G. W. Evans, "Distributed Supply Chain in a DEVS/CORBA Execution Environment", Proc. 1999 Winter Simulation Conference, Phoenix, Diciembre 2001.