Revisión de la literatura en interoperabilidad entre sistemas heterogéneos de *software*

A state-of-the-art review of interoperability amongst heterogeneous software systems

Carlos Mario Zapata[1] y Guillermo González Calderón[2]

RESUMEN

Los sistemas de información son conjuntos de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Por lo tanto, no pueden coexistir de forma aislada, sino que requieren que sus datos se compartan para incrementar su productividad. La interoperabilidad de dichos sistemas de información se logra, por lo general, por medio de estándares de marcado, lenguajes de consulta y servicios web. En la literatura existen trabajos de interoperabilidad de sistemas de *software*, pero presentan algunas falencias, como la necesidad de utilizar las mismas plataformas y diferentes lenguajes de programación, la utilización de lenguajes de solo consulta y las deficiencias de los formalismos empleados para ello. En este artículo se presenta una revisión crítica de los avances en interoperabilidad de sistemas heterogéneos de *software*.

Palabras clave: sistemas de información, interoperabilidad, lenguajes de programación, sistemas heterogéneos, lenguajes formales.

ABSTRACT

Information systems are sets of interacting elements aimed at supporting entrepreneurial or business activities; they cannot thus coexist in an isolated way but require their data to be shared so as to increase their productivity. Such systems' interoperability is normally accomplished through mark-up standards, query languages and web services. The literature contains work related to software system interoperability; however, it presents some difficulties, such as the need for using the same platforms and different programming languages, the use of read only languages and the deficiencies in the formalism used for achieving it. This paper presents a critical review of the advances made regarding heterogeneous software systems' interoperability.

Keywords: information system, interoperability, programming language, heterogeneous system, formal language.

Recibido: septiembre 8 de 2008
Aceptado: junio 1 de 2009

Introducción

Los sistemas de información necesitan comunicarse e intercambiar información para lograr mayor productividad (Galliers, 2006). A través de los avances de la tecnología en comunicación de computadores se logra interconectar esos sistemas, pero eso no es suficiente para lograr la capacidad de mejora deseada. La completa realización de los beneficios del potencial de interacción sólo se puede lograr si se obtiene interoperabilidad entre los sistemas de información, la cual se logra, por lo general, por medio de estándares de marcado, lenguajes de consulta y servicios web (Parlanti *et ál., 2008*). En la literatura existen trabajos de interoperabilidad de sistemas de información, pero presentan algunas falencias, como la necesidad de utilizar las mismas plataformas y diferentes lenguajes de programación, la utilización de lenguajes de solo consulta y las deficiencias de los formalismos empleados para ello (Zapata *et ál.*, 2007). En este artículo se presenta una revisión crítica de los avances en interoperabilidad de sistemas heterogéneos de *software*, analizando los aspectos que trabajan y las deficiencias que aún existen.

El artículo posee la siguiente estructura: inicialmente se presenta el marco teórico; posteriormente se muestra la revisión de la literatura relativa a la interoperabilidad de sistemas de información y luego se realiza un análisis de los problemas remanentes en dicha área; finalmente se presentan las conclusiones y el trabajo futuro.

Marco teórico

Sistemas de información

Los sistemas de información tratan el desarrollo, uso y administración de la infraestructura de la tecnología de la información organizacional (Galliers, 2006). Actualmente, el enfoque de las compañías pasó de la orientación hacia el producto a la orientación hacia el conocimiento. Así, el mercado compite hoy en día en términos del proceso y la innovación, en lugar del producto.

El mayor de los activos de una compañía hoy en día es su información, representada en su personal, experiencia, conocimiento e innovaciones (Mehdi *et ál.*, 2004). Para poder competir las organizaciones deben poseer una fuerte infraestructura de información, en cuyo corazón se sitúa la infraestructura de la tecnología de información, que se centra en estudiar las formas para mejorar el uso

[1] Ingeniero civil, Especialista, en Gerencia de Sistemas Informáticos, M.Sc., en Ingeniería de Sistemas y Ph.D., en Ingeniería, Universidad Nacional de Colombia, Medellín. Profesor asociado, Escuela de Sistemas, Universidad Nacional de Colombia, Medellín. cmzapata@bt.unal.edu.co
[2] Ingeniero de sistemas e informática y M.Sc., en Ingeniería de Sistemas, Universidad Nacional de Colombia, Medellín. Profesor, Facultad de Ingenierías, Ingeniería de Sistemas, Universidad de Medellín. ggonzalezc@udem.edu.co

de la tecnología que soporta el flujo de información dentro de la organización y que se implementa en variados sistemas de *software*, que son aquellos en los que la funcionalidad ofrecida al usuario se consigue mediante el desarrollo de uno o varios programas ejecutables. En el desarrollo, los recursos de *software* determinan el proceso de todo el sistema y se puede ejecutar sobre una plataforma genérica (Parlanti *et ál.,* 2008).

Interoperabilidad

Una aproximación para unir requisitos de sistemas futuros a través de la integración de sistemas es la formación de un "sistema de sistemas", al interconectar componentes aislados (*stand-alone*) (Young, 2002). La interoperabilidad entre sistemas no incluye solamente la habilidad de los sistemas para intercambiar información, sino también la capacidad de interacción y la ejecución de tareas conjuntas (C4ISR, 1998; Pitoura, 1997, pp. 99-126). Por tanto, el objetivo es crear un "sistema de sistemas" que no provea solamente interconectividad entre sistemas sino que logre una unión de sistemas interoperables (Wileden *et ál.*, 1991).

Una primera dificultad para lograr la interoperabilidad entre componentes heterogéneos de una unión de sistemas es que estos se suelen desarrollar independientemente, sin ningún requisito para interoperar. Así, los sistemas tienen diversas arquitecturas, plataformas de *hardware*, sistemas operativos, lenguajes de máquina y modelos de datos, con la dificultad de volver a desarrollar un nuevo sistema usando los requisitos consolidados de los diferentes componentes de los sistemas; y, además, una arquitectura, plataforma de *hardware*, sistema operativo y lenguaje de equipo común (una aproximación prohibitiva por el costo). Lo anterior obliga la concepción de un medio para lograr la meta de interoperabilidad de componentes con el presupuesto dado. La literatura especializada presenta algunas opciones empleadas para superar estas limitaciones.

XML

El lenguaje de marcado extensible (*Extensible Markup Language*—XML) (W3C, 2008) es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es una simplificación y adaptación del lenguaje de marcado generalizado (*Standard Generalized Markup Language*—SGML) (W3C, 2008) y permite definir la gramática de lenguajes específicos. Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para distintas necesidades. XML no nació sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

XML tiene un papel muy importante en la actualidad, ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. Algunos ejemplos de aplicaciones en este lenguaje son: XML-RPC (UserLand Software, 2008), para hacer llamadas remotas a procedimientos por Internet y FlexXML (Kaplan y Lunn, 2001), que trata de hacer más flexible y adaptable la red.

Lenguajes de consulta

El lenguaje de consulta estructurado (*Structured Query Language*—SQL) (SQL, 1999) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional, permitiendo consultar con el fin de recuperar, de una forma sencilla, información de interés de una base de datos, así como también hacer cambios sobre la misma.

XPath (XML *Path Language*) es un lenguaje que permite construir expresiones que recorren y procesan un documento XML (W3C, 2008). La idea es parecida a las expresiones regulares para seleccionar partes de un texto sin atributos (*plain text*). XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML. XPath se creó para su uso en el estándar de lenguaje extensible de hojas de estilo (*Extensible Stylesheet Language Transformations*—XSLT) (Adler *et ál.*, 2001), en el que se usa para seleccionar y examinar la estructura del documento de entrada de la transformación.

XQuery (XML *Query*) es un lenguaje de consultas diseñado para realizar búsquedas en colecciones de datos XML (W3C, 2008). Es semánticamente similar al SQL, pero incluye algunas capacidades de programación. Proporciona los medios para extraer y manipular información de documentos XML, o de cualquier fuente de datos que se puedan representar mediante XML, por ejemplo, bases de datos relacionales o documentos ofimáticos. XQuery utiliza expresiones XPath para acceder a determinadas partes del documento XML. Añade, además, unas expresiones similares a las usadas en SQL, conocidas como expresiones FLWOR, que toman su nombre de los cinco tipos de sentencias que la pueden componer: *FOR, LET, WHERE, ORDER BY* y *RETURN*.

Servicios web

Un servicio *web* es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones (W3C, 2008). Distintas aplicaciones de software, desarrolladas en diferentes lenguajes de programación y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios *web* para intercambiar datos en redes de ordenadores. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son las responsables de la arquitectura y reglamentación de los servicios *web*.

Revisión de la literatura en interoperabilidad de sistemas de información

Diferencias entre sistemas

Uno de los factores críticos detrás de la heterogeneidad de los sistemas de *software* diseñados y desarrollados aisladamente es la diversidad en los modelos básicos. Batini *et al.* (1986) describen tres causas mayores de representación de heterogeneidad:

-Perspectivas diferentes: las múltiples necesidades de los usuarios, los administradores de los programas y los equipos de diseño, pueden llevar a variadas representaciones de los datos aun cuando se modele la misma información.

-Construcciones equivalentes: modelos equivalentes del mismo dominio se pueden crear usando diversas combinaciones de las mismas construcciones básicas de modelado.

-Especificaciones incompatibles de diseño: los diseños de especificaciones de aplicaciones pueden generar varios esquemas de bases de datos para el mismo dominio.

Aunque estas causas se citaron, originalmente, en el contexto de la integración de esquemas de bases de datos (Batini *et ál.*, 1986); McCarthy y Dayal, 1989; Widom, 1996; Min, 2000), esos factores también aplican directamente a los tipos de modelos heterogéneos encontrados en sistemas desarrollados autónomamente.

Trabajos anteriores sobre arquitecturas de bases de datos múltiples (Gatziu *et ál.*, 1992; Gatziu y Dittrich, 1993; Paton y Díaz, 1999;

Türker y Gertz, 2001; Díaz *et al.*, 2001; Li *et ál.*, 2005) categorizaron diferencias de modelado en sistemas de bases de datos heterogéneos. En esta área, Wiederhold (1993) definió siete clases de heterogeneidad encontradas en sistemas de bases de datos desarrollados autónomamente, las cuales se relacionan estrechamente con las clases de heterogeneidad halladas en el contexto de interoperabilidad. Usando la clasificación de Wiederhold como línea base y reflejando las vistas de (Hammer y McLeod, 1999; Holowczak y Li, 1996; Kim y Seo, 1991; Kahng y McLeod, 1998; Young *et ál.*, 2003), se definen ocho clases de heterogeneidad cuando se trata de lograr interoperabilidad entre una unión de sistemas desarrollados independientemente. Estas, a su vez, se clasifican en dos categorías: "diferencias en vista", que muestran qué características del mundo real modelan diversos sistemas, y "diferencias en representación", que señalan cómo distintos sistemas modelan esas características del mundo real. En las "diferencias en vista" se encuentran las heterogeneidades de ámbito, nivel de abstracción y validez temporal, mientras que en las de "representación" se hallan las heterogeneidades de *hardware* y sistemas operativos, modelos organizacionales, estructura, presentación y significado.

Criterios para evaluar interoperabilidad

Con base en los trabajos de Young *et ál.* (2002, 2003), Wiederhold (1993), Kahng y McLeod (1998), Holowczak y Li (1996), Hammer y McLeod (1999) y Pressman (2001), se compendian los criterios comunes para evaluar las aproximaciones hacia la interoperabilidad, que son: los tipos de aproximación a la heterogeneidad, la capacidad para aplicación de ayuda computarizada en la correlación de modelos, el desarrollo de traducción y el intercambio de información contra ejecución de tareas conjuntas, el conocimiento requerido sobre operaciones remotas, la modificación necesaria para sistemas existentes y la metodología de traducción.

Aproximaciones hacia la interoperabilidad

Algunas arquitecturas, tecnologías, aplicaciones y lenguajes afirman proveer soporte para la interoperabilidad de componentes o sistemas. A continuación se presenta una evaluación de cada propuesta utilizando los criterios definidos en la sección anterior.

3.3.1 Arquitectura común de intermediarios en peticiones a objetos (Corba)

El Grupo de Gestión de Objetos (*Object Management Group*—OMG) tiene, como una de sus metas, «proveer una estructura común de arquitecturas para aplicaciones orientadas a objetos basada en las especificaciones de interfaz disponibles ampliamente» (OMG, 2008).

Corba provee capacidades en tres áreas para soportar interoperabilidad: 1) provee un mecanismo estándar para definir las interfaces entre componentes; 2) especifica un número de servicios estándar como directorio y servicios de nombre, de objetos persistentes y de transacciones disponibles para todas las aplicaciones compatibles con Corba; y 3) provee los mecanismos para permitir a los componentes de aplicación o aplicaciones separadas comunicarse entre ellas. Una plataforma y un lenguaje independiente proveen estas capacidades (Rosenberger, 1998).

Respecto a los criterios definidos anteriormente, Corba provee la capacidad para dirigir heterogeneidad de *hardware* y sistemas operativos, modelos organizacionales y, en parte, para resolver la heterogeneidad de representación (Pope, 1998). También, provee la capacidad para el intercambio de información y escenarios de ejecución de tareas conjuntas. Sin embargo, tiene como desventajas el fallo para dirigir el espectro completo de heterogeneidad, la falta de asistencia para correlacionar diferentes modelos de la misma entidad del mundo real en componentes de sistemas y la falta de asistencia en definición de la traducción requerida para resolver diferencias pequeñas de *hardware* y sistemas operativos. En Corba se requiere, para habilitar la interoperabilidad de sistemas, conocimiento previo sobre el nombre de un método del servidor y el tipo y modelo de los parámetros del método para utilizar su funcionalidad y exige la modificación a sistemas existentes.

3.3.2 COM, DCOM, COM+

El modelo de componentes objetual (COM), introducido por Microsoft® en 1993 (Microsoft, 2008), es una arquitectura de *software* que permite construir aplicaciones y sistemas a partir de componentes binarios suministrados por variados proveedores de *software*. COM y sus arquitecturas sucesoras distribuidas (DCOM y COM+) compiten con Corba. La función original de COM es proveer un mecanismo de propósito general para la integración de componentes en plataformas Windows.

Al evaluar COM, DCOM y COM+ contra los criterios usados para comparar las aproximaciones de interoperabilidad, las tres tecnologías se entienden como una sola, ya que representan la evolución del mismo concepto. La familia de arquitecturas COM+ provee una capacidad similar a la provista por Corba. Adicionalmente, habilita la interoperabilidad de componentes binarios de *software,* mientras que Corba dirige interoperabilidad a nivel de código fuente. La familia COM+ comparte la falencia de Corba para dirigir el espectro completo de heterogeneidad y la falta de asistencia en las diferencias de correlación y resolución de entidades de modelos del mundo real. Finalmente, la familia COM+ requiere conocimiento previo de los métodos con el fin de utilizar su funcionalidad y exige modificación a los sistemas existentes no desarrollados con compatibilidad con los estándares COM+ para habilitar su interoperabilidad.

3.3.3 Microsoft® .NET

Microsoft .NET es una plataforma construida bajo el sistema operativo Windows®, que provee la infraestructura necesaria para resolver problemas comunes en aplicaciones de Internet (Microsoft, 2008). Microsoft .Net provee algunas capacidades que soportan interoperabilidad de sistemas y componentes, incluyendo un lenguaje común en tiempo de ejecución *(Common Language Runtime*—CLR*),* un lenguaje intermedio que remueve diferencias en implementación de componentes de sistema, y un mecanismo de interoperabilidad que permite a los programas .NET acceder a código previo. El corazón de .NET es el CLR, el cual es el sucesor a la tecnología de componentes COM de Microsoft®. Microsoft® .NET provee mecanismos de interoperabilidad que permiten la integración entre programas .NET y código previo. El CRL entrega a .NET un ambiente de programación que soporta interoperabilidad, donde las clases y los objetos en un lenguaje se pueden usar en otro lenguaje sin necesidad de usar una interfaz especializada de definición de lenguaje (*Interface Definition Language*—IDL*)*. Esto contrasta con la aproximación usada por COM y Corba, la cual permite llamar a los métodos de una clase remota como métodos externos solo si la llamada se adhiere a un modelo estándar. Microsoft .NET provee una capacidad superior para interoperabilidad a la que ofrece la familiar COM+, ya que algunas de las heterogeneidades entre sistemas se eliminan a través del uso de un lenguaje común intermedio (*Microsoft Intermediate Language*—MSIL). Debido a que .NET es una plataforma Windows® y su capacidad de lenguajes se limita a

cuatro (Visual Basic, C++, C# y JScript), no se debería considerar una solución de interoperabilidad.

3.3.4 Java 2 Enterprise Edition (J2EE)

La especificación de J2EE define una plataforma JAVA con características que apuntan a los ambientes computacionales de nivel empresarial (Berg, 1999). Extiende la definición de la edición estándar principalmente en las áreas de seguridad, despliegue e interoperabilidad. J2EE presenta una aproximación competitiva a la provista por Corba y la familia COM+. Entre sus fortalezas se incluyen el soporte para intercambio de información y la ejecución de tareas en conjunto entre sistemas unidos. Entre las desventajas se encuentran la falla para dirigir el espectro completo de la heterogeneidad de sistemas, la falta de capacidad para asistencia en establecer correspondencia entre diferentes modelos de la misma entidad del mundo real, la necesidad de conocer el nombre del servidor o identificar atributos para invocar los métodos del servidor desde un cliente y la restricción de que las aplicaciones del cliente y el servidor se escriban en JAVA.

3.3.5 Lenguaje de marcado extensible–XML

El XML, descrito anteriormente, se publicita ampliamente como un medio para lograr la interoperabilidad de sistemas, pues provee medios de autodescripción para representar los datos usados y compartidos entre aplicaciones (Bray *et ál.*, 2000). Aunque no ofrece una facilidad de computación distribuida como las ofrecidas por Corba, COM+ o J2EE, XML provee soporte para lograr interoperabilidad entre sistemas desarrollados independientemente.

De las aproximaciones presentadas, XML ofrece el mayor soporte para los criterios de heterogeneidad presentados anteriormente. XML presenta un mecanismo para definición de datos y organización, pese a que no provee un método para invocación remota de métodos entre aplicaciones. Sin embargo, una especificación guía, SOAP (*Simple Object Access Protocol),* provee una estructura de mensajería para intercambiar mensajes XML entre plataformas heterogéneas (SOAP, 2002). Aunque ofrece soporte significativo para la interoperabilidad de sistemas, XML también comparte algunas de las limitaciones encontradas en otras aproximaciones, como la falla para dirigir el espectro total de las heterogeneidades, y la falta de facilidades para correlacionar diferentes modelos de la misma entidad del mundo real.

Análisis de los problemas remanentes

Si bien existe gran variedad de plataformas, se puede observar que se crean varios inconvenientes a la hora de comunicar sistemas heterogéneos, ya que, por residir en un sistema operativo distinto, tener un *hardware* de otro tipo o simplemente no poseer las mismas librerías de tiempo de ejecución, hacen que no exista compatibilidad entre estos sistemas y, por tanto, queden aislados de otros con los que pudieran compartir información común.

Por otro lado, existen varios paradigmas de programación, entre los que se encuentran el imperativo o por procedimientos, que se considera el más común y lo representan, por ejemplo, los lenguajes C o Java; y el paradigma lógico, con lenguajes como Prolog, que son la base para la construcción de los sistemas expertos. Ante esto, no es posible comunicar sistemas heterogéneos con uno de tales paradigmas solamente, ya que dichos lenguajes operan de manera diferente. Esta diversidad sintáctica no permite reutilizar reglas o compartir conocimiento y, por tanto, exige a los desarrolladores de reglas de una empresa a construir desde cero la lógica completa del negocio, que es similar con reglas descritas en entornos de lenguajes de reglas distintas (Janssen, 1994; Buchmann, 1995; Hanson, 1996; Cho *et ál.*, 2002; Comani, 2003; Chavarría y Báez, 2004; Grosof, 2008; BRML, 2008; Boley *et ál.*, 2008).

Finalmente, existen aproximaciones de métodos que aplican reglas en conjunto con XML (Bonifati *et ál.*, 2001; Kiyomitsu *et ál.*, 2001), como es el caso de *Xcerpt* (Schaffert, 2004). Sin embargo, solo presentan integración por medio de consultas. La aproximación de (Zapata *et ál.*, 2007) utiliza una combinación de un lenguaje procedimental (JAVA) con un lenguaje declarativo (XPATH) para lograr comunicar dos sistemas.

Conclusiones

Este artículo mostró la existencia de diferencias de modelado entre sistemas desarrollados independientemente, citando las causas principales de tales diferencias, seleccionando un conjunto de criterios para la evaluación de aproximaciones de interoperabilidad con el fin de comparar su éxito al resolver dichas heterogeneidades. Las aproximaciones actuales para lograr interoperabilidad entre sistemas heterogéneos incluyen varias limitaciones:

-No proveen un medio para resolver el espectro completo de modelado de diferencias encontrados entre sistemas heterogéneos.

-No proveen asistencia para determinar cuándo los modelos de sistemas diferentes se refieren a la misma entidad del dominio del problema.

-Para acceder a otro componente o un estado del sistema, la mayoría de aproximaciones actuales requieren que el sistema que hace la petición utilice el mismo modelo del proveedor para acceder a su información. Esto, usualmente, requiere la modificación de los sistemas existentes para habilitar la interoperabilidad, limitando significativamente la aplicabilidad de la aproximación cuando se construye un conjunto de sistemas de componentes existentes donde el costo, principalmente, restringe la modificación de componentes.

-La mayoría de aproximaciones utilizan procesos de conversión punto a punto para resolver diferencias entre sistemas. Para un conjunto de más de dos sistemas, las aproximaciones punto a punto requieren un número mayor de traducciones por definir.

-La mayoría de aproximaciones no proveen soporte para el desarrollo de las traducciones requeridas en la resolución de diferencias de modelado entre sistemas.

-La mayoría de aproximaciones se enfocan en la resolución del modelado de diferencias para información intercambiada entre sistemas y no proveen la capacidad de resolver posibles diferencias en las firmas usadas para acceder el comportamiento de los correspondientes métodos en diferentes sistemas.

Como trabajo futuro, se podría encontrar solución a dichas limitaciones. Además, se podría hacer la especificación de un lenguaje que permita crear sistemas de *software* y a su vez permita la integración con sistemas existentes por medio de la migración de sus reglas y estructura de modo que sean compatibles y, por tanto, interoperables.

Bibliografía

Adler, S., *W3C,* Extensible Stylesheet Language (XSL) Version 1.0 W3C Candidate Recommendation., 15 Oct., 2001.

Batini, C., Lenzerini, M., Navathe, S., A Comparative Analysis of Methodologies for Database Schema Integration., ACM Computing Surveys, Vol. 18, No. 4, 1986, pp. 323-364.

Berg, C., Advanced Java 2 Development for Enterprise Applications., 2d ed., Sun Microsystems Press, Prentice-Hall, Inc., Upper Saddle River NJ, 2000.

Boley, H., Rule Markup Language., Disponible en: http://www.dfki.unikl.de/ruleml/ . Consultado en agosto de 2008)

Bonifati, A., Ceri, S., Paraboschi, S., Pushing Reactive Services to XML Repositories using Active Rules., 10th International World Wide Web Conference, 2001, pp. 633-641.

Bray, T., *W3C,* Extensible Markup Language (XML) 1.0., 2nd Ed. W3C Recomm. Oct., 2000.

Buchmann, A. P., Deutsch, A., Zimmermann, J., Higa, M., The REACH active OODBMS., Proc. of the ACM SIGMOD International Conference on Management of Data, 1995, pp. 476.

Chavarría-Báez, L., Li X., Measuring triggering interactions complexity on active databases based on conditional colored Petri net model., 1st Intl. Conf. on Electrical and Electronics Eng. and 10th Conf. on Electrical Eng, (ICEEE/CIE), Acapulco, México, Sept. 8-10, 2004.

Cho,E., Park,I., Hyun,S., ARML: an Active Rule Markup Language for Sharing Rules among Active Information Management Systems en RuleML 2002: Proc. of the Intl., Workshop on Rule Markup Languages for Business Rules on the Semantic Web, Sardinia, Italia, 2002

Comani, S., Tanca, L., Termination and Confluence by Rule Prioritization., IEEE Trans. on Knowledge and Data Engineering, Vol., 15, No. 2, 2003.

Díaz O., Piattini M., Calero C., Measuring Triggering-Interaction Complexity on Active Databases., Information Systems, Vol. 26, 2001. pp. 15- 34

Galliers, R. D., Markus, M. L., Newell, S., (Eds)., Exploring Information Systems Research Approaches., New York, NY: Routledge, 2006.

Gatziu, S., Dittrich, K. R., Events in an Active Object-Oriented Database System., In Proceedings of the 1st International Workshop on Rules in Database Systems, 1993, pp. 23-39.

Gatziu, S., SAMOS: An active object-oriented database system., IEEE Data Engineering, Special issue on active databases, 15, 1-4, 1992,pp.23-26.

Grosof, B., Chan, H., IBM CommonRules home pages., Disponible en: http://www.research.ibm.com/rules/ y http://alphaworks.ibm.Com. Consultado en agosto 2008.

Hammer, J., McLeod, D., Resolution of Representational Diversity in Multidatabase Systems., Management of Heterogeneous and Autonomous Database Systems, 1999.

Hanson, E., The design and implementation of the Ariel active database rule system., IEEE Transactions on Knowledge and Data Engineering. Vol. 8, No. 1, 1966, pp. 157-172.

Holowczak, R., Li, W., A Survey on Attribute Correspondence and Heterogeneity Metadata Representation., First IEEE Metadata Conference (IEEE, Silver Spring), 1996.

Janssen, B., Spreitzer, M., ILU, Inter-language unification via object modules., In: Workshop on Multi-Language Object Models, Portland, OR, (in conjunction with OOPSLA94), Aug, 1994.

Kahng, J., McLeod D., Dynamic Classificational Ontologies: Mediation of Information Sharing in Cooperative Federated Database Systems., Cooperative Information Systems, Trends and Directions, Academic Press, 1998.

Kaplan, A., Lunn, J., FlexXML: engineering a more flexible and adaptable web., IEEE Information Technology: Coding and Computing, 2001, pp. 405-410.

Kim, W., Seo, J., Classifying Schematic and Data Heterogeneity in Multidatabase Systems., IEEE Computer, December, 1991, pp. 12-18.

Kiyomitsu, H., Takeuchi, A., Tanaka, K., ActiveWeb: XML-based Rules for Web View Derivations and Access Control., ITVE 2001, IEEE. Vol. 23, No. 6, 2001, pp. 31-39.

C4ISR Arch., Working Group., Levels of Information Systems Interoperability (LISI), March, 1998.

Li, X., Medina-Marín, J., Chapa, S. V., Applying Petri nets on active database systems., IEEE Trans. on System, Man, and Cybernetics, Part C, 2005.

McCarthy, D., Dayal, U., The Architecture Of An Active Data Base Management System., Proceedings of the ACM SIGMOD, 1989, pp. 215-223

Mehdi, S., Jorge, P., Michel, L., Information system architectures: where we are? Information and Communication Technologies: From Theory to Applications., Proceedings International Conference, 19-23 April, 2004, pp.509–510.

Microsoft., Distributed Component Object Model (Dcom)., 1993. Disponible en: http://www.microsoft.com/com/default.mspx.

Microsoft., .NET Framework (.NET)., 2003. Disponible en: http://www.microsoft.com/net.

Min, H.J., Design and Implementation of an Object-oriented Rule Management System for Active Database Services., Master Dissertation, ICU, Korea, 2000.

OMG., Object Management Group. 2002., Disponible en: http://www.omg.org/.

Parlanti, D., Pettenati, M.C., Bussotti, P., Giuli, D., Improving Information Systems Interoperability and Flexibility Using a Semantic Integration Approach., Automated solutions for Cross Media Content and Multi-channel Distribution, 2008. AXMEDIS '08. International Conference, 17-19 Nov., 2008, pp.63–67.

Paton, N. W., Díaz, O., Active database system., ACM Computing Serveys, 1999, pp. 63-103.

Pitoura, E., Providing Database Interoperability through Object-Oriented Language Constructs., Journal of Systems Integration, Vol. 7, No. 2, August 1997, pp. 99-126.

Pope, A., The CORBA Reference Guide., Addison-Wesley Longman, Inc., Redding, MA, 1998.

Pressman, R., Software Engineering: A Practitioner's Approach., McGraw-Hill, Boston, MA, 2001.

Robin Cover and OASIS: Business Rules Markup Language (BRML). Disponible en http://www.oasis-open.org/cover/brml.html

Rosenberger, J., Teach Yourself CORBA in 14 Days., Sams Publishing, Indianapolis, 1998.

Schaffert, S., Xcerpt, A., Rule-Based Query and Transformation Language for the Web., PhD thesis, University of Munich, 2004.

SOAP, Simple Object Access Protocol., SOAP Version 1.2 Part 0: Primer, 2002. Disponible en: http://www.w3.org/TR/2002/CR-soap12-part0-20021219.

SQL, Structured Query Language., ANSI/ISO/IEC International Standard (IS)., Database Language SQL—Part 2: Foundation (SQL/Foundation), 1999.

Türker, C., Gertz, M., Semantic integrity support in SQL:1999 and commercial (object-) relational database management systems., VLDB Journal, Vol. 10, No. 4, 2001, pp. 241-269.

UserLand Software, Inc., XML-RPC. Disponible en: http://www.xmlrpc.com. Consultado en Agosto 2008.

W3C., eXtensible Markup Language (XML) 1.1 (2003)., Disponible en: http://www.w3.org/XML/. Consultado en Agosto 2008.

W3C., HyperText Markup Language (HTML) 5.0 (2003)., Disponible en: http://www.w3.org/html/. Consultado en Agosto 2008.

W3C., Standard Generalized Markup Language (SGML). 2.0., Disponible en: http://www.w3.org/MarkUp/SGML/ (accedido en Agosto 2008).

W3C, Web Services. Disponible en http://www.w3.org/2002/ws/. Consultado en Agosto 2008.

Widom, J., The Starburst active database rule system., IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 4, 1996, pp. 583-595.

Wiederhold, G., Intelligent Integration of Information., ACM-SIGMOD 93, Washington, DC, May, 1993, pp. 434- 437.

Wileden, J. C., Wolf, A. L., Rosenblatt, W. R., Tarr, P. L., Specification level interoperability., Communications of the ACM, Vol. 34, No. 5, May, 1991, pp. 73-87.

Young, P., Berzins, V. Ge, J., Luqi, Using an Object Oriented Model for Resolving Representational Differences between Heterogeneous Systems., The 17th ACM Symposium on Applied Computing, Madrid, Spain, March 10–14, 2002.

Young, P., Chaki, N., Berzins, V., Luqi; Evaluation of middleware architectures in achieving system interoperability Rapid Systems Prototyping, 14th IEEE International Workshop, 9-11 June, 2003, pp.108–116.

Young, P. E., Heterogeneous Software System Interoperability Through Computer-Aided Resolution of Modelling Differences., Ph.D. dissertation, Naval Postgraduate School,2002.

Zapata, C., Gonzalez, G., Gelbukh, A., A Rule-Based System for Assessing Consistency Between UML Models., MICAI 2007, LNAI 4827, 2007, pp. 215–224.