MDA vs Factorías de Software

Javier Muñoz, Vicente Pelechano
Dept. de Sistemas Informáticos y Computadores
Universidad Politécnica de Valencia
Campus de Vera
46022 Valencia
{jmunoz, pele}@dsic.upv.es

Resumen

La comunidad de la Ingeniería de Software está prestando en la actualidad un especial interés a dos enfoques para el desarrollo de software: la Model Driven Architectura (MDA) y las Factorías de Software. Este trabajo plantea e intenta responder algunas cuestiones respecto a la relación entre ambos enfoques. El trabajo discute las principales diferencias entre ambas propuestas, sus respectivas ventajas e inconvenientes, la posibilidad de integración y se identifican algunos aspectos de interés en el ámbito de la investigación. Las argumentaciones presentadas se ilustran mediante la aplicación de estas propuestas para la definición de un método de desarrollo de sistemas pervasivos.

1. Introducción

La comunidad de la Ingeniería de Software está prestando en la actualidad un especial interés a dos enfoques para el desarrollo de software. Por un lado, la Model Driven Architecture (MDA) [10] propuesta por el Object Management Group (OMG), y por otro, las más recientes Factorías de Software [6] promovidas por grupos de trabajo relacionados con la empresa Microsoft. Ambas iniciativas cuentan con el apoyo de actores muy importantes en la industria del software y, quizá por ello, parecen encontrarse enfrentadas, debido a que los promotores de una, en muchos aspectos, suelen ser detractores de la otra.

El objetivo de este trabajo, en la línea del trabajo presentado el año pasado en este mismo evento [8], es plantear e intentar responder algunas cuestiones que surgen respecto a la relación entre los enfoques MDA y Factorías de Software. En concreto se discute sobre las principales diferencias, sus respectivas ventajas e inconvenientes, la posibilidad de integración entre ambas y las oportunidades para la investigación que pueden plantearse. Las argumentaciones presentadas se ilustran mediante la aplicación de estas propuestas para la definición de un método de desarrollo de sistemas pervasivos.

La estructura del artículo es la siguiente: en la sección 2 se presentan una serie de cuestiones sobre las discutirá en el resto del trabajo. A continuación, la sección 3 describe brevemente las dos propuestas. La sección 4 analiza y discute cada una de las cuestiones planteadas. Finalmente, la sección 5 presenta las conclusiones que se han alcanzado.

2. Cuestiones planteadas

Actualmente existe un intenso debate entre los defensores de las propuestas MDA y Factorías de Software. Artículos defendiendo y atacando cada propuesta se suceden en revistas[1] y *blogs* personales[2]. Habitualmente, los participantes en estos debates pertenecen a alguna de la compañías implicadas en la promoción de ambos enfoques, por lo que en ocasiones es

[1] http://www.bptrends.com/search.cfm?keyword=MDA+journal
[2] http://blogs.msdn.com/stevecook/

difícil dilucidar si las argumentaciones responden a argumentos técnico/científicos o a intereses comerciales. Ante esta situación se considera necesario plantear y estudiar una serie de cuestiones que permitan a la comunidad científica tener una visión clara de estos enfoques. Este artículo trata de responder a las siguientes cuestiones:

- **¿Cuáles son las diferencias básicas entre MDA y las Factorías Software?**¿Qué ofrece cada enfoque que no posea el otro? Se discutirán los puntos críticos que diferencian y causan conflicto entre las dos propuestas.
- **¿Cuáles son las ventajas y debilidades de cada enfoque?** ¿En qué contexto es mejor utilizar uno u otro? De esta manera se facilitará la tarea de selección de una de las dos propuestas en una situación en concreto.
- **¿Son estrategias compatibles?** ¿Se pueden combinar de alguna manera para obtener un método con las ventajas de cada una? La integración de ambas propuestas nos puede llevar a la definición de métodos de desarrollo avanzados.
- **¿Interesan en el ámbito académico?** Mas allá de su posible aplicación industrial, ¿introducen aspectos que necesiten de investigación académica? Se pretende diseñar posibles estrategias de investigación basadas en estos enfoques metodológicos.

3. Características básicas de ambos enfoques

Antes de discutir las cuestiones planteadas es necesario describir las principales características de los dos enfoques. Para ello, se han tomado como fuentes de información básicas, el documento «MDA Guide Version 1.0»[10] publicado por el OMG y el libro «Software Factories. Assembling Applications with Patterns, Models, Frameworks and Tools»[6].

3.1. ¿Qué es MDA?

La propuesta MDA la promueve el Object Management Group (OMG), un consorcio de empresas (IBM, Borland, Hewlett-Packard o Boeing entre otras) que produce y mantiene una serie de especificaciones para permitir la interoperabilidad entre aplicaciones software.

Según la guía de MDA, *MDA is an approach to using models in software development* [10, 2-1]. La principal característica diferenciadora de MDA respecto a los enfoques tradicionales para el desarrollo de software se encuentra en el uso de los modelos como el recurso principal en el proceso de desarrollo. MDA propone que los sistemas software sean generados directamente a partir de modelos de dicho sistema software. OMG propone y promueve el uso de diversos lenguajes relacionados con la creación y gestión de modelos (UML, MOF, CWM y QVT) como mecanismos básicos para soportar esta estrategia.

Como comenta en [8], algunos autores [11, 3] separan por un lado el enfoque de Desarrollo Dirigido por Modelos, y por otro la propuesta MDA de OMG. Siguiendo este criterio podemos diferenciar entre:

Model Driven Development (MDD)
: que es una **aproximación al desarrollo de software** basado en el modelado del sistema software y su generación a partir de los modelos. Al ser únicamente una aproximación, sólo proporciona una estrategia general a seguir en el desarrollo de software, pero no define ni técnicas a utilizar, ni fases del proceso, ni ningún tipo de guía metodológica.

Model Driven Architecture (MDA)
: que es un **estándar de OMG que promueve el MDD y agrupa varios lenguajes que pueden usarse para seguir este enfoque**. MDA posee el valor añadido de proporcionar lenguajes con los que definir métodos que sigan MDD. Por lo tanto, MDA tampoco define técnicas, etapas, artefactos, etc. MDA sólo proporciona la infraestructura tecnológica y conceptual con la que construir estos métodos MDD.

En resumen, podemos afirmar que la propuesta MDA proporciona una infraestructura para la construcción de métodos de desarrollo de software que sigan el MDD.

3.2. ¿Qué son las Factorías de Software?

Las Factorías de Software constituyen un enfoque para el desarrollo de software promovido principalmente por Microsoft. Aunque el término Factoría de Software no es nuevo [7, 4], desde la publicación del libro «Software Factories. Assembling Applications with Patterns, Models, Frameworks and Tools»[6] ha adquirido una especial relevancia en desarrollo basado en modelos.

Una Factoría de Software, tal y como se define en [6], es *una línea de productos software que configura herramientas extensibles, procesos y contenido [...] para automatizar el desarrollo y mantenimiento de variantes de un producto arquetípico mediante la adaptación, ensamblaje y configuración de componentes basados en frameworks.* Por lo tanto, las Factorías de Software se centran en el desarrollo de sistemas similares promoviendo la reutilización de arquitecturas, componentes software y conocimiento.

Para alcanzar este objetivo, las Factorías Software integran varias técnicas conocidas en la Ingeniería Software. Las principales actividades que promueven las Factorías Software son:

- **Construcción de familias de software similar.** Esta actividad supone el análisis y diseño de una arquitectura común para un conjunto de sistemas, y el desarrollo de un framework que soporte esta arquitectura.
- **Ensamblado de componentes.** La construcción de un nuevo sistema supone el uso, ensamblado y/o configuración de los componentes proporcionados por el framework.
- **Desarrollo de lenguajes de modelado y herramientas específicas para el dominio.** Los desarrolladores utilizan estos lenguajes para describir los requisitos específicos de un miembro de la familia de sistemas. A partir de estas especificaciones se genera automáticamente el código de la parte específica del sistema.
- **Uso de una planificación basada en restricciones y guía activa.** Todos los pasos del proyecto de desarrollo deben realizarse de acuerdo a un proces bien definido y adaptado al dominio.

En esta estrategia, tanto la arquitectura, como el framework de soporte y los lenguajes específicos del dominio deben hacer uso en la medida de los posibles de patrones que encapsulan conocimiento sobre el dominio.

Por lo tanto, el enfoque de las Factorías de Software propone el desarrollo de métodos específicos para cada tipo de aplicación. Utilizando estos métodos, los desarrolladores especifican cada sistema utilizando un lenguaje de modelado. A partir de estas especificaciones se genera automáticamente el código que extiende o configura el framework que sigue la arquitectura que será compartida por todos los sistemas desarrollados utilizando la factoría.

4. Discusión

Una vez descritas las principales características de los dos enfoques, se procede a discuir las cuestiones planteadas en la sección 2. Con el objetivo de ilustrar esta discusión, las respuestas de cada cuestión se aplicarán a un ejemplo en concreto: la creación de un método para el desarrollo de sistemas pervasivos.

4.1. Caso de estudio: Un método para el desarrollo de sistemas pervasivos

Los Sistemas Pervasivos pretenden crear entornos en los que los elementos de computación desaparezcan pero su funcionalidad se encuentre disponible para los usuarios en un entorno determinado. La esencia de esta visión, descrita inicialmente por Weiser [13] a principios de la década de los 90, se encuentra en la creación de entornos saturados de computación y comunicaciones integradas adecuadamente con los usuarios humanos.

A diferencia de los sistemas de información tradicionales en los que la principal funcionalidad es la gestión de datos, los sistemas pervasivos deben proporciona servicios de control, multimedia, seguridad física y comunicaciones en un entorno determinado. En los sistemas pervasivos cobra una relevancia fundamental la integración con dispositivos de todo tipo y con sistemas de información externos.

Por lo tanto, se plantean los siguientes requisitos para la construcción de un método de desarrollo de sistemas pervasivos:

- El método debe permitir especificar un sistema pervasivo utilizando primitivas conceptuales adecuadas para ese tipo de sistemas.
- El método debe generar sistemas funcionalmente operativos a partir la especificación del mismo.
- Los sistemas generados deberán soportar la integración de las distintas tecnologías utilizadas en los sistemas pervasivos.

Una vez descrito el caso que se utilizará como ejemplo, se procede a realizar la discusión de las cuestiones planteadas.

4.2. Diferencias entre estrategias

Antes de iniciar la construcción de un método de desarrollo para sistemas pervasivos utilizando el enfoque MDA o Factorías Software, se plantea la necesidad de identificar cuales serían las diferencias entre utilizar una u otra estrategia.

Por ejemplo, siguiendo el enfoque propuesto por MDA, para construir un método de desarrollo de software para sistemas pervasivos deberíamos:

- definir una técnica de modelado que permita especificar Sistemas Pervasivos de manera independiente de plataforma. Para crear esta técnica se puede optar por extender UML realizando un *profile* o definir un nuevo lenguaje, para lo cual se construiría un metamodelo utilizando MOF. En cualquier caso, los modelos deberían almacenarse utilizando el formato XMI.
- elegir una de las estrategias que propone MDA para transformar los modelos independientes de plataforma (PIM) en modelos dependientes de plataforma (PSM) a partir de los cuales poder generar el código de la aplicación. Una vez elegida la estrategia, será necesario especificar detalladamente cómo se lleva a cabo esta transformación. El estándar de OMG propuesto para llevar a cabo esta tarea es QVT, aunque la versión definitiva todavía no está aprobada, por lo que no existen herramientas software que sigan el estándar. Por lo tanto, será necesario utilizar alguna otra técnica/herramienta.
- proporcionar herramientas de soporte que permitan usar el método de una manera sencilla y eficiente. Para la construcción de los modelos se puede optar por la extensión de alguna herramienta CASE actual que soporte UML (como Rose o ArgoUML) o la construcción de una nueva herramienta utilizando alguna librería para la construcción de metamodelos con MOF (como EMOF, para Eclipse, o MDR, para NetBeans).

Por otra parte, siguiendo el enfoque de las Factorías de Software, para construir el método deberíamos:

- definir una arquitectura común para todos los sistemas pervasivos que se van a construir.
- definir un lenguaje de modelado que permita especificar los requisitos particulares de cada sistema pervasivo a desarrollar.
- diseñar un framework que implemente la arquitectura definida anteriormente, proporcionando constructores que permitan definir correspondencias con las primitivas conceptuales utilizadas en el lenguaje específico para sistemas pervasivos. En el

caso del ejemplo, el framework debería facilitar la integración de las distintas tecnologías utilizadas en el desarrollo de este tipo de sistemas.

- proporcionar herramientas de soporte que permitan usar el método de una manera sencilla y eficiente. Para la construcción de los modelos, los autores del libro de las Factorías Software [6] recomiendan el uso de las Domain-Specific Language Tools[3] de Microsoft, aunque es posible utilizar otras herramientas como MetaEdit+[4]. También son necesarias herramientas para la generación del código a partir de los modelos construídos con el lenguaje de dominio específico. Estas herramientas suelen ser dependientes de las utilizadas para construir los modelos, ya que utilizan sistemas propios para almacenar y gestionar los modelos.

Como se puede observar, cada uno de los enfoques describe más detalladamente y enfatiza ciertos aspectos del proceso de desarrollo. En general se identifican las siguientes diferencias:

- **Las Factorías de Software proporcionan una mayor guía para llevar a cabo el desarrollo de software**, ya que recomiendan explícitamente el uso de líneas de producto, frameworks de implementación, patrones de diseño, etc.; mientras que MDA se centra en diferenciar entre la descripción del sistema de manera independiente de plataforma y las posibles realizaciones de esa especificación. MDA no proporciona indicaciones sobre cómo se deben diseñar los sistemas, se limita a recomendar que la generación del código se realice a partir de modelos.

- **Las Factorías Software promueven el uso de Lenguajes de Dominio Específico**, mientras que MDA promueve el uso de UML, que es de propósito general. Esta es una de las diferencias más importantes entre los dos enfoques y la que, sin duda, es más polémica. Las Factorías de Software recomiendan la creación de lenguajes (de modelado) que permitan a los desarrolladores utilizar las primitivas más adecuadas para cada tipo de sistema. La OMG defiende que debe usarse UML, utilizando sus capacidades de extensión (profiles), cuando sea necesario, para expresar conceptos que no pueden representarse de forma directa en UML.

- **MDA proporciona lenguajes para la gestión de los modelos** (UML, CWM, MOF, QVT), mientras que las Factorías de Software se limitan a describir técnicas que pueden utilizarse para este fin. Por ejemplo, en [6, cap. 8] se describe la posibilidad de especificar la sintaxis abstracta de los lenguajes específicos de dominio utilizando BNF o metamodelos, para estos últimos utiliza una notación similar a la propuesta por MOF. Sin embargo en el apéndice B de [6] se critica MOF desaconsejando su uso sin proponer ninguna alternativa.

- **MDA promueve explícitamente elevar el nivel de abstracción** a la hora de especificar los sistemas, las Factorías Software permiten la creación de DSLs que representen directamente elementos de plataformas de implementación. En varios ejemplos, por ejemplo algunas figuras de [6, cap. 7], se muestran modelos que utilizan primitivas como «ASP.NET Service» o «Windows Application». Siguiendo la estrategia MDA, un sistema debe especificarse utilizando modelos independiente de plataforma (PIM) que, posteriormente, podrán ser transformados o anotados para convertirse en modelos específicos de una plataforma (PSM).

4.3. Ventajas e inconvenientes

Las diferencias descritas en la subsección anterior indican que cada uno de los dos enfoques posee una serie de puntos fuertes frente al otro. En general, puede decirse que las **ventajas de MDA** son:

[3] http://lab.msdn.microsoft.com/teamsystem/Workshop/DSLTools/

[4] http://www.metacase.com/

MDA	Factorías Software
(√) Indica claramente las técnicas a utilizar (UML, MOF, CWM, QVT)	(×) No propone ninguna técnica en concreto
(×) Sólo define la estrategia general PIM ⟶ PSM	(√) Proporciona más guías metodológicas (DSLs, Líneas de Producto, Frameworks, etc.
(√) Está más tiempo. La primera versión es de 2001	(×) Más "joven". El libro es de 2004
(√) Más herramientas disponibles (OptimalJ, Arcstyler, etc.)	(×) Herramientas no maduras o difundidas (SDL Tools, MetaEdit+)
(×) Carencias en transformación de modelos. Estándar QVT no aprobado y sin soporte de herramientas	(√) Integra varias áreas de conocimiento en Ing. del SW.
(_) Promovido por OMG, un consorcio de empresas	(√) Promovido principalmente por el líder del mercado: Microsoft

Tabla 1: Resumen comparativo de ambos enfoques.

- **Define de manera clara la tecnología que recomienda utilizar** para aplicar su enfoque: UML, MOF, QVT, etc. Esto facilita la tarea de construcción de un método a los desarrolladores que quieren aplicar este enfoque. Podemos contar con lenguajes ampliamente conocidos que, en gran parte, no necesitan de descripción y disponen de abundante documentación.

- **Está más tiempo «en el mercado»**, ya que fue presentado con anterioridad a las Factorías de Software. Debido a esto, MDA ha sido más estudiado, discutido y aplicado. Este aspecto facilita la aceptación de un método por parte de la comunidad de la Ingeniería del Software.

- **Ofrece un mayor soporte de herramientas** que están empezando a madurar. Esto ocurre debido a dos causas: (1) OMG está compuesto por muchas empresas que desean ofrecer productos que sigan las especificaciones de OMG, y tal y como se ha dicho en el punto anterior, (2) MDA es conocido desde hace más tiempo. Por ejemplo, Eclipse es un entorno de desarrollo extensible, libre y promovido principalmente por IBM. Este entorno tiene una amplia comunidad que desarrolla extensiones para proporcionar funcionalidad muy diversa. Entre estas extensiones destaca EMF, una extensión para el metamodelado basado en el estándar MOF de OMG capaz de almacenar los modelos en formato XMI 2. También existe una extensión que proporciona una implementación del metamodelo de UML 2 utilizando EMF, disponiendo en estos casos de implementaciones libres de los estándares de OMG. Por otra parte, las Domain-Specific Language Tools que Microsoft propone para construir Factorías de Software todavía se encuentran en fase de desarrollo, por lo que no son muy robustas y no pueden aplicarse en entornos de producción industrial.

Por su parte, consideramos que las **ventajas de las Factorías de Software** son:

- **Proporcionan una completa guía metodológica a los desarrolladores**. Con las Factorías de Software se definen de forma más clara y precisa los pasos que deben llevarse a cabo para construir un método que siga esa aproximación. Esto ha quedado patente en la sección 4.2, donde se ha descrito cómo se deberían aplicar cada una de las estrategias para crear un

método para el desarrollo de sistemas pervasivos. La cantidad de información para la construcción de métodos que proporciona el enfoque de las Factorías de Software es mayor que la proporcionada por MDA, ya que éste último no da ninguna pauta sobre cómo se deberían implementar los sistemas.

- **Integra muchas áreas de la Ingeniería del Software** que ya han sido investigadas y puestas en práctica (líneas de productos, patrones de diseño, construcción de frameworks, etc.). De este modo, este enfoque no parte desde cero, sino que integra el conocimiento de estos temas.

- **El principal promotor es Microsoft**, por lo que su posición dominante en el mercado puede conseguir que el enfoque de las Factorías de Software acabe implantándose en la industria. Pese a ello, se debe tener en cuenta que OMG también auna un número importante de multinacionales del desarrollo de software, por lo que esta ventaja puede ser relativa.

Como puede observarse, cada enfoque posee unas ventajas que resultan muy interesantes para la creación de métodos de desarrollo de software, pero ninguno de las dos resulta claramente mejor que la otra. A la hora de construir un método de desarrollo de software, la elección de un enfoque u otro dependerá de las características requeridas por el método:

- El enfoque MDA es especialmente adecuado cuando la interoperabilidad con otras herramientas o el uso de herramientas existentes (que sigan los estándares de OMG) sea un factor clave.

- El enfoque de las Factorías de Software es especialmente adecuado cuando existe la intención de construir una serie de sistemas similares y/o se va trabajar dentro de un dominio determinado.

En nuestro método para el desarrollo de sistemas pervasivos se satisfacen las dos condiciones, por lo que ninguno de los dos enfoques puede quedar descartado. Es necesario destacar que esta situación es similar para numerosos métodos, por lo que todo lo que se diga aplicado a nuestro ejemplo será fácilmente generalizable. Como se verá a continuación, la aplicación de una estrategia mixta basada en MDA y las Factorías Software puede resultar muy interesante.

4.4. ¿MDA + Factorías de Software?

Cada uno de los enfoques discutidos hace énfasis en ciertos aspectos del proceso de desarrollo de software. Esto les confiere ciertas ventajas respecto al otro en aspectos específicos y claramente diferenciados. Una cuestión que surge rápidamente es ¿sería posible integrar los dos enfoques? Para ello **deberíamos seguir las recomendaciones de ambos** e integrar de algún modo líneas de productos, modelos de alto nivel de abstracción, frameworks de implementación, lenguajes de dominio especifico, los lenguajes de OMG, etc.

Seguir este enfoque integrador tiene una serie de ventajas:

1. La estrategia general del método está definida: línea de producto + framework de implementación + lenguaje de dominio específico.

2. Las técnicas a utilizar están definidas: MOF + QVT.

3. Existen herramientas de soporte: EMF de Eclipse o MDR de NetBeans, por ejemplo.

4. Se pueden aplicar técnicas conocidas propuestas por las factorías de software, como los "feature models"para la construcción de la línea de producto o los patrones de sieño para el desarrollo de framework de implementación.

Por lo tanto, para desarrollar una método que integre MDA y Factorías de software habría que::

- Hacer uso de la estrategia metodológica que proponen las Factorías de Software; es decir, seguir un enfoque basado en:

1. el desarrollo de una línea de producto,
2. la construcción de un framework de implementación para los sistemas que serán desarrollados siguiendo la línea de producto, y
3. la definición de un lenguaje de dominio específico (DSL) que permita capturar los requisitos específicos de cada uno de los sistemas utilizando las primitivas conceptuales más adecuadas para ese tipo de sistemas.

- Construir el DSL de manera que:
  1. proporcione primitivas conceptuales independientes de plataforma.
  2. se especifique siguiendo las técnicas que propone OMG (creación de un profile UML o definición de un metamodelo con MOF).

Un método que siga estas directivas podrá argumentar que sigue tanto el enfoque de las Factorías de Software como el que propone MDA. Ésta ha sido la estrategia elegida finalmente en nuestro caso, publicada en [9]. Siguiendo esta estrategia mixta, el método para sistemas pervasivos proporciona:

- Un lenguaje para la especificación de sistemas pervasivos: Perv-ML. Este lenguaje utiliza primitivas que permiten representar los conceptos utilizados en sistemas pervasivos como *servicio* o *dispositivo*. Se ha definido el metamodelo MOF del lenguaje.
- Un framework para la implementación de sistemas pervasivos. Los puntos de extensión del framework proporcionan constructores similares a las primitivas conceptuales que ofrece Perv-ML para la especificación de sistemas pervasivos.
- Una implementación de Perv-ML utilizando la extensión EMF de Eclipse. A partir de esta implementación se está desarrollando un motor de transformación de especificaciones Perv-ML a código que extienda el framework. El motor de transformación, a falta del estándar definitivo de OMG, se basa en la herramienta AGG[5], que permite especificar y ejecutar transformaciones de modelos mediante gramáticas de grafos, y el motor de plantillas FreeMarker[6]. Utilizamos esta estrategia mixta (gramáticas de grafos + plantillas) ya que consideramos que las gramáticas de grafos resultan una técnica adecuada para definir transformaciones entre metamodelos, mientras que las plantillas resultan más cómodas para la generación de los archivos de código finales.

4.5. Interés científico

MDA y las Factorías Software son enfoques que necesitan mejorar porque poseen algunos aspectos indefinidos y/o discutibles. Estas propuesta dejan abiertas numerosas cuestiones, ofreciendo oportunidades de trabajo e investigación para la comunidad científica del ámbito de la Ingeniería del Software. Entre otras necesidades, es interesante:

- Crear métodos que sigan las pautas indicadas por estos enfoques. Estos métodos serán útiles intrínsecamente por su aplicabilidad para el desarrollo de software y servirán de ejemplo y experiencia para la construcción de otros métodos nuevos.
- Aplicar estos métodos en entornos reales de producción. Esto servirá para demostrar su viabilidad y para extraer experiencias que ayuden a la evolución de los métodos o la creación de nuevos.
- Demostrar que estos métodos son mejores que los métodos tradicionales. Para ello será necesario realizar establecer métricas y realizar experimentos que permitan llevar a cabo una comparación objetiva de los mismos.
- Proporcionar técnicas y herramientas para dar soporte a estos enfoques. Dentro

[5] http://tfs.cs.tu.berlin.de/agg/
[6] http://freemarker.sf.net

de este campo cabe considerar la gestión de modelos, transformación de modelos, desarrollo de frameworks, construcción de DSLs, etc.

Como puede observarse, el interés para la comunidad científica se extiende cada vez más como lo demuestra la creación de numerosos eventos científicos relacionados con estos temas (EWMDA [5], MDAFA [12], WiSME [2], Generative Techniques in the context of Model Driven Architecture [1]).

5. Conclusiones

En este trabajo se han presentado y discutido una serie de cuestiones que estudian la relación existente entre MDA y las Factorías de Software. Una conclusión importante de este análisis es que **no se trata de enfoques tan incompatibles como el enfrentamiento Microsoft vs OMG podría hacer suponer**. Si nos fijamos en las principales recomendaciones de cada una de las propuestas, se puede constatar que es posible compatibilizarlas, aunque es cierto que miembros del OMG desaconsejan el uso de lenguajes de dominio específico para no caer en un babel de lenguajes, y Greenfield et al. en [6] desaconsejan el uso de los lenguajes de OMG por resultar imprecisos.

En general, como se ha comentado en la subsección 4.3, la elección de uno de los dos enfoques dependerá de los requisitos y los beneficios concretos que se desee de obtener en cada situación. También se debe tener en cuenta que, debido a que ambos enfoques están siendo apoyados por participantes importantes de la industria del desarrollo de software, **el éxito final de uno o de otro puede depender de factores ajenos a sus características y virtudes** como, por ejemplo, a estrategias de márketing o la posición en el mercado de sus respectivos promotores.

Referencias

[1] *Generative Techniques in the context of MDA*, October 2003. Anaheim, California, USA. http://www.softmetaware.com/oopsla2003/mda-workshop.html.

[2] *Workshop in Software Model Engineering*, October 2004. Lisbon, Portugal. http://albini.xactium.com/wisme/.

[3] Colin Atkinson and Thomas Kühne. Model-Driven Development: A Metamodeling Foundation. *IEEE Software*, 20(5):46–51, September/October 2003.

[4] Michael A. Cusumano. *Japan's Software Factories. A Challenge to U.S. Management.* Oxford University Press, 1991.

[5] D.H.Akehurst, editor. *Second European Workshop on Model Driven Architecture (MDA)*, September 2004. Canterbury, UK. http://www.cs.kent.ac.uk/projects/kmf/mdaworkshop/.

[6] Jack Greenfield, Keith Short, Steve Cook, and Stuart Kent. *Software Factories.* Wiley Publising Inc., 2004.

[7] Ivan Aaen and Peter Bøtcher and Lars Mathiassen. The Software Factory: Contributions and Illusions. In *Proceedings of the Twentieth Information Systems Research Seminar in Scandinavia*, Oslo, 1997.

[8] Javier Muñoz and Vicente Pelechano. MDA a Debate. In *I Taller sobre Desarrollo de Software Dirigido por Modelos, MDA y Aplicaciones (DSDM'04)*, pages pp. 1 – 12, 2004.

[9] Javier Muñoz and Vicente Pelechano. Building a Software Factory for Pervasive Systems Development. In João Falcão e Cunha Oscar Pastor, editor, *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17*, volume 3520 of *Lecture Notes in Computer Science*, pages 329–343. Springer-Verlag GmbH, May 2005.

[10] Object Management Group. Model Driven Architecture Guide, 2003.

[11] Stephen J. Mellor and Anthony N. Clark and Takao Futagami. Guest Editors' Introduction: Model-Driven Development. *IEEE Software*, 20(5):14–18, 2003.

[12] Uwe Abmann, editor. *Proceedings of Model-Driven Architecture: Foundations and Applications*, June 2004. Linköping University, Sweden. http://www.ida.liu.se/ henla/mdafa2004/.

[13] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, Sept. 1991.