CAPÍTULO V

CONCLUSIONES

Contenidos del capítulo:

CONCLUSIONES

5.1 Conclusiones

En este trabajo se ha realizado la implementación de un modelo orientado a aspectos y basado en componentes como PRISMA utilizando la tecnología .NET.

Para ello, se ha realizado un análisis preliminar de las diferentes aproximaciones existentes orientadas a aspectos, del que se han extraído las principales tendencias de desarrollo, así como las carencias de los prototipos existentes. Todas ellas tienen en común la separación clara del código de aplicación respecto del código de los aspectos, y la definición de los *weavings* de forma independiente al código a unir. Sin embargo, pocas de ellas aportan mecanismos para añadir o eliminar aspectos dinámicamente. Las que sí lo tienen en cuenta, lo hacen a costa de perder la reutilización, bien de aspectos, o bien del código de la aplicación. Además, ninguna de ellas consigue fusionar adecuadamente la programación orientada a aspectos (AOP) con el diseño basado en componentes (CBSD).

En cambio, el modelo PRISMA combina la reutilización proporcionada por AOP y la reconfiguración dinámica de los modelos CBSD. Los aspectos se definen de forma separada a los weavings y son altamente reutilizables. Los componentes se forman por la agregación dinámica e inclusiva de aspectos y pueden ser movidos a otras máquinas en tiempo de ejecución.

La propuesta de implementación realizada se ha materializado en un prototipo desarrollado con la tecnología .NET, del que se han identificado las plantillas de código que deberán ser utilizadas por el compilador para generar código C# a partir de especificaciones PRISMA. Para el diseño del prototipo se han intentado alcanzar los siguientes objetivos, que caracterizan al modelo PRISMA:

- Alta reutilización de los elementos arquitectónicos
- Ejecución concurrente de los distintos elementos
- Soporte a la comunicación distribuida
- Movilidad autónoma (por iniciativa propia) o inducida (a petición de otro elemento), de componentes, conectores y sistemas

- Reconfiguración dinámica de los modelos arquitectónicos para ajustarse a otros entornos
- Evolución dinámica del modelo

Algunos de los objetivos anteriores se han alcanzado completamente, como la reutilización, la ejecución concurrente, la comunicación distribuida y la reconfiguración dinámica. Otros objetivos se han alcanzado parcialmente, pero han sentado las bases de futuras versiones del prototipo desarrollado. Es el caso de la movilidad y de la evolución dinámica. Los componentes pueden ser movidos a otras máquinas, por iniciativa propia o a petición de otro elemento. Sin embargo, aún quedan algunos problemas por resolver para situaciones concretas, como el movimiento cuando la ejecución de un servicio no puede ser interrumpida. La evolución dinámica no ha sido implementada en su totalidad, pero se han definido los mecanismos que permiten modificar en tiempo de ejecución el comportamiento de cada elemento arquitectónico PRISMA, aunque sólo individualmente. Por ejemplo, el componente soporta la modificación en tiempo de ejecución de su estructura interna (puertos, aspectos y weavings), sin necesidad de detener sus operaciones por completo y recompilarse. Sin embargo, los cambios sólo afectan a la instancia a la que se le han solicitado. Para que el prototipo soporte la evolución de forma completa aún tienen que ser introducidos una serie de mecanismos adicionales que permitan la coordinación entre el tipo y sus diferentes instancias y la propagación de los cambios a los distintos *middlewares*.

Además, el prototipo desarrollado ha sido validado con la implementación de dos casos de estudio: la cuenta bancaria y el robot 4U4. El caso de estudio de la cuenta bancaria, cuya especificación se ha mostrado en el capítulo 0 para ilustrar los conceptos del modelo PRISMA, ha permitido validar la implementación de la movilidad y la comunicación distribuida. Por otra parte, el caso de estudio del robot 4U4 [Per04], cuya especificación modela un brazo robótico de ámbito industrial, ha permitido validar la implementación de arquitecturas complejas, concurrentes y altamente reutilizables.

5.2 Trabajos futuros

La plataforma de ejecución de modelos PRISMA aún se encuentra en una fase temprana y no se puede dar el trabajo realizado por concluido. Todavía quedan tareas pendientes por realizar y muchos aspectos que no han sido solucionados en esta primera versión. A continuación se citan los más importantes.

Uno de los trabajos más importantes que quedan por realizar es el desarrollo de un compilador que transforme las especificaciones PRISMA a .NET, cuyas plantillas de código se han obtenido como resultado de este

trabajo. El uso de dicho compilador permitirá ejecutar de forma transparente para el usuario los modelos PRISMA introducidos, así como realizar los cambios en la especificación del modelo, sin hacer uso de la evolución dinámica.

Otro aspecto a considerar, tal y como se ha comentado anteriormente, es que ha de dotarse al prototipo de los mecanismos necesarios para soportar la evolución dinámica a nivel global. Dichos mecanismos deberán permitir:

- Propagar los cambios a todas las instancias de un tipo en la máquina local.
- Almacenar los cambios para que futuras instancias del tipo los incorporen.
- Propagar el tipo modificado a los demás *middlewares* que lo utilicen, además de propagar los cambios a las instancias en ejecución de otras máquinas.

Por otra parte, también es necesario introducir mecanismos para mejorar la parada segura de componentes, conectores y sistemas, de cara a la movilidad. En la implementación actual, existen una serie de situaciones en las que la movilidad no es posible, a menos que finalicen o sean abortadas. Dichas situaciones se producen cuando un componente se encuentra ejecutando un servicio de larga duración, que no puede detenerse, o un servicio que implica la invocación de otros servicios externos.

Por último, también queda como tarea pendiente la implementación del DNS de forma descentralizada, como establece el modelo PRISMA.

BIBLIOGRAFÍA

[Ali03] Ali N.H., Silva J., Jaen J., Ramos I., Carsi J.A., Perez J. *Mobility and Replicability Patterns in Aspect-oriented Component-Based Software Architectures.* In Proc. of 15th IASTED, Parallel and Distributed Systems, Acta Press, ISBN: 0-88986-392-X, ISSN: 1027-2658, pp. 820-826. Marina del Rey, C.A., USA, Noviembre 2003.

[Aosd] *Aspect-oriented software development.* En: http://aosd.net

[Ar02] Archer T., *Inside C#, 2nd Edition.* Microsoft Press, ISBN: 0-7356-1648-5, Redmond, Washington, 2002.

[AsJ03] The AspectJ Team. *AspectJ Programming Guide*, 2003.

[AsJ04] AspectJ Project, en http://eclipse.org/aspectj/

[Bal85] Balzer R., *A 15 Year Perspective on Automatic Programming*. IEEE Transactions on Software Engineering, vol.11, num.11, págs. 1257-1268, Noviembre 1985.

[Dng04] *AspectDNG Project*. En: http://aspectdng.sourceforge.net/

[Gun04] Gunnerson E., *Calling Code Dynamically*. Microsoft Corp., Feb. 2004. En: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncscol/html/csharp02172004.asp

[Jack04] Jackson A., Clarke S., *SourceWeave.NET: Cross-Language Aspect-Oriented Programming*. In Proc. of Generative Programming and Component Engineering (GPCE 2004). Vancouver, Canada, 2004.

[Kic97] Kiczales, G., et al., *Aspect-Oriented Programming*. In European Conference on Object-Oriented Programming (ECOOP'97), Jyväskylä, Finland, 1997, Springer-Verlag, pp.220-242.

[Kim02] Kim H., *AspectC#: an AOSD implementation for C#*. M. Sc. Thesis, Comp. Sci., Trinity College, Dublin, Noviembre 2002.

[Mil91] Milner, R., *π-Cálculo Poliádico: A tutorial*, 1991.

[Mono] *The Mono Project*. En: http://www.mono-project.com

[Laff03] Lafferty D., Cahill V., *Language-Independent Aspect-Oriented Programming*. In Proc. of Object-Oriented Programming Systems, Languages and Applications (OOPSLA 2003). Anaheim, California, USA, 2003.

[Lam02] Lam J., *Cross Language Aspect Weaving*. Demonstration, AOSD 2002, Enschede, 2002.

[Let98] Letelier P., Sánchez P., Ramos I., Pastor O., *OASIS 3.0: "Un enfoque formal para el modelado conceptual orientado a objeto"*. Universidad Politécnica de Valencia, SPUPV -98.4011, ISBN 84-7721- 663-0, 1998.

[Ossh00] Ossher H., Tarr P., *Multi-Dimensional Separation of Concerns and The Hyperspace approach*. In Proc. of the Symposium on Software Architectures and Component Technology: The State of the Art in Software Development. Kluwer, 2000.

[Pas97] Pastor O. Et al., *OO-METHOD: A software production environment combining conventional and formal methods*. In proc. of 9th International Conference, CaiSE97, Barcelona, 1997.

[Per03] Pérez, J., Ramos, I., *Oasis como Soporte Formal para la Definición de Modelos Hipermedia Dinámicos, Distribuidos y Evolutivos*. Informe Técnico DSIC-II/22/03, Universidad Politécnica de Valencia, octubre 2003.

[Per04] Pérez, J., Cabedo R., et al., *Arquitectura PRISMA para el caso de estudio: Brazo Robot*. Actas del II workshop DYNAMICA – DYNamic and Aspect-Oriented Modeling for Integrated Component-based Architectures, pags. 119-127, junto a Jornadas de Ingeniería del Software y Bases de Datos (JISBD), Málaga, noviembre 2004.

[Ramm02] Rammer, I., *Advanced .NET Remoting (C# Edition)*. Apress, 2002.

[Raj03] Rajan H., Sullivan K., *EOS: Instance-Level Aspects for Integrated System Design*. In Proc. of Joint European Software Engineering Conference (ESEC) and ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE). Helsinki, Finland, Septiembre 2003.

[RRose] Rational Software, *Rational Rose*, http://www.rational.com/products/rose/

[Sch00] Schult W., Polze A., *Aspect-Oriented Programming with C# and .Net*. In Proc. of International Symposium on Object-oriented Real-time distributed Computing (ISORC) 2002, pp. 241-248, Crystal City, VA, USA, Mayo 2000.

[Sch02] Schult W., Polze A., *Dynamic Aspect-Weaving with .NET*. Workshop zur Beherrschung nicht-funktionaler Eigenschaften in Betriebssystemen und Verteilten Systemen, TU Berlin, Germany, Noviembre 2002.

[Schm02] Schmied, F. *AOP with .NET*. En: http://wwwse.fhs-hagenberg.ac.at/se/berufspraktika/2002/se99047/contents/

[Schü02] Schüpany, M., Schwanninger, C., Wuchner, E., *Aspect-Oriented Programming for .NET*. In First AOSD Workshop on Aspects, Components, and Patterns for Infrastructure Software, (Enschede, The Netherlands, 2002), pp.59-64.

[Scott03] McLean S., Naftel J., Williams K., *Microsoft .NET Remoting*. Microsoft Press, 2003.

[SetP04] SetPoint! Project, en: http://www.dc.uba.ar/people/proyinv/setpoint/

[Ser94] Sernadas A., Costa J.F., Sernadas C., *Object specifications trough diagrams: OBLOG approach.*INESC Lisbon, 1994.

[Sparx] Sparx Systems, *Enterprise Architect*. En: http://www.sparxsystems.com/

[Suv03] Suvee, D., Vanderperren, W. and Jonckers, V., *JAsCo: an Aspect-Oriented approach tailored for Component Based Software Development*. In Proc. of international conference on aspect-oriented software development (AOSD), Boston, USA, pp 21-29, ISBN 1-58113-660-9, ACM Press, Marzo 2003.

[Vers03] Verspecht, D., Vanderperren, W., Suvee, D. and Jonckers, V., *JAsCo.NET: Capturing Crosscutting Concerns in .NET Web Services.* In Proc. of Second Nordic Conference on Web Services NCWS'03, Vaxjo, Sweden. Publicado en "Mathematical modelling in Physics, Engineering and Cognitive Sciences", Vol. 8, Noviembre 2003.