# Una Solución Multiparadigma para el Desarrollo de Software

Silvia Amaro, Pablo Quiroga, Luis Reynoso
{samaro, pquiroga, lreynoso}@uncoma.edu.ar
Claudio Vaucheret, Lidia López, Daniel Dolz
{vaucheret, lidiamlopez, daniel.dolz}@gmail.com
Andrea Granados, Ingrid Godoy, Viviana Sánchez, Maximiliano Klemen
{agranados80, ingriduncoma, sanchez.viviana, maximilianoklemen}@gmail.com
Paola Pérez
ppercat@hotmail.com

Dpto. de Ciencias de la Computación
Facultad de Economía y Administración
Universidad Nacional del Comahue, Argentina

## 1. Contexto

La línea de investigación orientada al desarrollo multiparadigma se enmarca dentro del proyecto de investigación ”Técnicas Avanzadas y Análisis para el Desarrollo Multiparadigma”

## 2. Resumen

En el proceso de desarrollo de software, es necesario ser capaces de identificar la naturaleza del problema a resolver para elegir las herramientas y técnicas que mejor se adapten a la situación, y den respuesta a la naturaleza multidimensional de los problemas de software complejos. Una aproximación multiparadigma parece ser lo opción indicada teniendo en cuenta que usualmente un único paradigma no alcanza en el dominio del problema a resolver. Nuestro objetivo es establecer modelos, procesos y técnicas para mejorar el desarrollo de sistemas en un ambiente multiparadigma.
*Palabras clave:* Programación multiparadigma, Lenguajes de Programación, Abstracción, Dominios complejos.

## 3. Introducción

Un producto de software está compuesto de un número importante de diversos artefactos que representan porciones diferentes del software, obtenidos a lo largo del proceso de desarrollo (requerimientos, especificación, diseño, código fuente, casos de testeo, etc.). Los sistemas de software evolucionan en el tiempo, y a medida que el software cambia y crece, todos los artefactos que lo componen evolucionan a su vez, en diferentes formas.

Los problemas que representan las aplicaciones de software son usualmente complejos, y tienen una naturaleza multidimensional. Los desarrolladores tienden a pensar de formas diferentes dependiendo de la naturaleza del problema que les toca resolver. Asimismo es importante lograr los resultados esperados, en el tiempo esperado y con la calidad esperada.

En este contexto es importante aplicar las técnicas y prácticas mas apropiadas para el problema a resolver. Y aquí las abstracciones juegan un rol principal, enfocándose en la esencia del problema y excluyendo los detalles, analizando concordancias y diferencias. La abstracción es una herramienta clave del diseño de software que permite tratar la creciente complejidad de los sitemas informáticos.

La mayoría de los paradigmas de software están arraigados en un modelo clásico de abstracción basado en categorias formadas por propiedades comunes. Para analizar mas allá de este modelo, es necesario mostrar que algunos dominios del mundo real no responden estrictamente al modelo clasico, sino mas bien, que la combinación de características de diferentes paradigmas dentro de un dominio es la vía adecuada para representar problemas y soluciones de dominios reales y alcanzar metas mas altas de independencia de desarrollo y capacidad de mantenimiento [12].

El paradigma orientado a objetos brinda una poderosa herramienta para capturar las abstracciones de muchos dominios de aplicación, presentando una forma natural de mdularizar una aplicación. Sin embargo ningún paradigma es capaz de resolver todos los problemas de forma sencilla y eficiente, por lo tanto es útil poder elegir entre distintas técnicas de programación dependiendo del tipo de problema [10, 6].

En este escenario el desarrollo multiparadigma tiene como principal objetivo proveer al programador con un conjunto variado de herramientas de manera que cuando enfrente un problema no se vea forzado a utilizar una única técnica de programación, sino que tenga la libertad de seleccionar la técnica de solución que mejor se adapte a las características del problema a resolver.

Pero, ¿qué es un paradigma?. En tecnologías de software un paradigma representa las directivas para crear abstracciones. El paradigma es el principio por el cual un problema puede ser comprendido y descompuesto en componentes manejables. En particular un paradigma de programación es el modo de conceptualizar lo que significa realizar una computación y cómo las tareas a llevarse a cabo sobre una computadora deben ser estructuradas y organizadas. El lenguaje en el cual un programador piensa un problema influirá en modo básico fundamental la forma en la cual un algoritmo es desarrollado.

La comparación entre paradigmas se basa en la facilidad de uso por el programador que no puede disociarse de la naturaleza del problema a resolver. Un lenguaje de programación puede soportar distintos paradigmas de programación con el objetivo de que un programador utilice el más conveniente a la hora de resolver un problema. Un lenguaje de programación multiparadigma es un lenguaje de programación que soporta mas de un paradigma de programación.

Timothy Budd, en [17] sostiene que: ”la idea de un lenguaje multiparadigma es proveer un framework en el que los programadores puedan trabajar en una variedad de estilos, mezclando libremente las construcciones características de los diferentes paradigmas. La meta de diseño de tales lenguajes es permitir a los programadores utilizar la mejor herramienta para un trabajo, admitiendo que ningún paradigma resuelve todos los problemas en la forma mas fácil o mas eficiente”.

Varios lenguajes han demostrado que orientación a objetos y programación funcional pueden complementarse, formando un equipo exitoso [16]. Por su parte el desarrollo orientado a aspectos se presenta apropiado para asistir al desarrollador en aislar puntos de variación en un programa. Esto ayuda al programa a evolucionar y adaptarse a nuevos requerimientos de cambio durante el ciclo de vida del programa. Si bien la programación orientada a aspectos en general se ha presentado como extensión de la orientación a objetos, y básicamente en las situaciones en que los lenguajes soporten herencia y polimorfismo de subtipos, es de interés investigar los resultados que se pueden obtener al tomar otros paradigmas anfitriones.

En particular, un lenguaje que se está tomando como caso de estudio y plataforma de desarrollo dada sus características únicas es *Ciao Prolog* [3]. *Ciao* es un lenguaje multiparadigma basado en expansiones sintácticas que implementan distintos paradigmas mediante la técnica de compilación de control.
También se esta analizando el lenguaje multiparadigma Oz [7, 11], que fue diseñado para combinar en forma armónica conceptos que tradicionalmente están relacionados con paradigmas de programación diferentes. Oz tiene subconjuntos que representan un lenguaje lógico, un lenguaje funcional, un lenguaje orientado a objetos y un lenguaje concurrente de flujo de datos, y mas. Otros lenguajes de interás son: *Leda*, soporta los paradigmas imperativo, orientado a objetos, lógico y funcional [17]; *Python*, combina el paradigma imperativo,funcional, orientado objetos y orientado a aspectos; *Ruby, Scala*, y mas.

# 4. Líneas de Investigación y Desarrollo

Para lograr los objetivos planteados se están abordando distintas temáticas en el contexto del desarrollo multiparadigma, como:

- *bases para la definición del paradigma orientado a agentes* Se pretende establecer las bases para dar origen a una definición definitiva del paradigma orientado a agentes. Se esta trabajando sobre Ciao FLUX, una primera aproximación a una extensión del lenguaje multiparadigma Ciao Prolog que permite dotar a agentes con capacidades cognitivas de alto nivel, mediante una adaptación del método de programación de agentes FLUX. La bondad más importante que brinda Ciao, sin embargo, es la facilidad con la que permite realizar extensiones del lenguaje, ya sea a nivel sintáctico, semántico o de control [9, 8]. Se eligió Ciao Prolog como lenguaje anfitrión por su facilidad de extensión y porque posee un gran repertorio de módulos que permiten implementar autonomía, reactividad, proactividad, habilidad social y movilidad, características fundamentales a la hora de programar agentes.

- *Análisis y adecuación de técnicas orientadas a aspectos en paradigmas básicos.* En general, al trabajar con lenguajes de AOP, la lógica de los aspectos se entrelaza en la aplicación destino, y en esta situación múltiples aspectos, desarrollados de forma independiente pueden producir comportamiento inesperado en el sistema. Resulta de interés analizar el beneficio que pueda producir la aplicación de conceptos de orientación a aspectos en lenguajes no orientados a objetos y multiparadigma [4, 2].

- *Análisis de diversos paradigmas de desarrollo de software* Nos interesa determinar cuales son los efectos de considerar varios paradigmas en el proceso de desarrollo de software. Consideraremos, el desarrollo de software orientado a aspectos, ingeniería de software basada en componentes, desarrollo de software orientado a características [5, 1, 15], etc.

- *Análisis y comparación de lenguajes multiparadigma* Interesa estudiar la posibilidad de extender algunos lenguajes, incorporando características multiparadigma. Se está realizando un análisis de los lenguajes que tienen características multiparadigma (Ciao Prolog, Ruby, Oz, Python, Scala, etc) para definir un entorno comparativo de los mismos [6, 16, 14].

# 5. Resultados esperados

El desarrollo multiparadigma permite la construcción de sistemas mediante la selección del paradigma apropiado en cada situación. El producto de la presente propuesta puede ser de utilidad para el desarrollo de aplicaciones en dominios particulares y que deban cumplir con

propiedades particulares, tales como eficiencia, seguridad y calidad, brindando apoyo al desarrollador para la evaluación y selección de las mejores técnicas y paradigmas para la construcción de sistemas en gran escala.

# 6. Formación de Recursos Humanos

El mayor impacto del presente proyecto se centra en la formación de recursos humanos, consolidación de grupos de investigación e interacción entre grupos interdisciplinarios. Parte de los autores están dando sus primeros pasos en investigación. Actualmente se están desarrollando en el contexto del proyecto 3 tesinas de grado, cuya finalización está prevista para el segundo semestre del corriente año. Asimismo se están desarrollando 1 tesis de magister y 2 tesis doctorales y se prevee la iniciación de nuevos estudios de posgrado.

# Referencias


[1] Guido Soldner and Rudiger Kapetza. *AOCI: An Aspect-Oriented Component Infraestructure*. Workshop on Component Oriented Programming WCOP, ECOOP 2007, Berlin, Alemania.

[2] Dong Ha Nguyen and Mario Südholt. *Property-Preserving Evolution of Components Using VPA-Based Aspects*. In Workshop on Reflection, AOP and Meta-Data for Software Evolution, RAM-SE, ECOOP 2007, Berlin, Alemania.

[3] C. Vaucheret and F. Bueno. *More precise yet efficient type inference for logic programs.* In International Static Analysis Symposium, LNCS 2477, pp 102-116. Springer-Verlag, September 2002.

[4] Robert E. Filman, Tzilla Elrad, Siobhán Clarke, and Mehmet Ak¸sit, editors. *Aspect-Oriented Software Development.* Addison-Wesley, Boston, 2005.

[5] Éric Tanter and Jacques Noyé. *A Versatile Kernel for Multi-Language AOP.* In Proceedings of the ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering (GPCE 2005), LNCS, Springer-Verlag, Tallin, Estonia, September, 2005.

[6] K. Davis and J. Striegnitz. *Multiparadigm Programming in Object-Oriented Languages: Current Research*. In Patrick Eugster (eds.): Object-Oriented Technology, LNCS 5475, Programming and Software Engineering, pp 104-115, 2009.

[7] F. Spiessens and P. Van Roy. *The Oz-E PRoject: Design Guidelines for a Secure Multiparadigm Programming Language.*. http://www.info.ucl.ac.be/ pvr/oze.pdf

[8] M. Thielscher. *FLUX: A logic programming method for reasoning agents*. In Theory and Practice of Logic Programming, Special Issue on Constraint Handling Rules, 5(4&5):533-565, 2005.

[9] M.V. Hermenegildo, F. Bueno, M. Carro, P. López, J.F. Morales, and G. Puebla. *An overview of the ciao multiparadigm language and program development environment and its design philosophy*. LNCS 5065, Concurrency, Graphs and Models. pp 209-237, 2008.

[10] M. Fowler, *Domain Specific Languages* work in progress, available at http://www.martinfowler.com/dslwip/

[11] *The Oz 1.1 Programming System*. Available at http://www.ps.uni-sb.de/oz1/

[12] J. O. Coplien. *MultiParadigm Design*. Vrije Universiteit Brussel, Faculteit Wetenschanppen - Departement Informatica.

[13] P. Van Roy. *Programming Paradigms for Dummies: What Every Programmer Should Know* Department of Computing Science and Engineering - Université catholique de Louvain (UCL) à Louvain-la-Neuve. Available at http://www.info.ucl.ac.be/ pvr/VanRoyChapter.pdf

[14] P. Van Roy, P. Brand, D. Duchier, S. Haridi, M. Henz and C. Schulte. *Logic Programming in the Context of Multiparadigm Programming: the Oz Experience*. Available at http://www.info.ucl.ac.be/ pvr/tutFinal.pdf

[15] S. Apel and C. Kästner. *An Overview of Feature Oriented Software Development*. In Journal of Object Technology, vol 8, no 5., pp 49-84. Julio-Agosto 2009. Available at http://www.jot.fm/issues/issue_2009_09/article5.

[16] Narbel. *Functional Programming at work in Object-Oriented Programming*. In Journal of Object Technology, vol 8, no 6., pp 181-209. Septiembre-Octubre 2009. Available at http://www.jot.fm/issues/issue_2009_07/column5.

[17] T. Budd. *Multiparadigm Programming in Leda*. Addison Wesley 1995. ISBN 0-201-82080-3.