MULTIPARADIGMA EN LA ENSEÑANZA DE LA PROGRAMACION

Yanina Paola Pérez – ppercast@hotmail.com
Lidia Marina López – llopez@uncoma.edu.ar
Departamento de Ciencias de la Computación
Universidad Nacional del Comahue
Buenos Aires 1400 – Neuquén – tel. 0299-4430312

Abstract

La Programación Multiparadigma es una práctica que emerge como resultado de la co-existencia de los paradigmas orientado a objetos, procedural, declarativo y funcional buscando mejorar la producción en el desarrollo de proyectos. En años recientes, las instituciones académicas han ido implementando la programación orientada a objetos como el primer paradigma en la enseñanza de la programación en los planes de estudios de las carreras de grado de informática. Este trabajo apunta a presentar la necesidad de analizar los paradigmas con los que se encara la enseñanza de la programación y a la posibilidad de incorporar un lenguaje de programación multiparadigma en su lugar.

Introducción

Los sistemas informáticos desempeñan un papel esencial en la Sociedad de la Información. A medida que la sociedad se implica más en tales sistemas y crece su dependencia hacia ellos, existe un avance en los distintos lenguajes de programación para satisfacer las demandas actuales. Por todo esto se hace presente la necesidad de que los estudiantes de computación incorporen durante su cursado una visión general de los paradigmas de programación que le permita aprovechar los beneficios de cada uno y superar sus dificultades. El fin de ampliar el espectro de enseñanza, es preparar a los futuros profesionales a tener una visión general que les permita ir incorporando nuevos paradigmas ofreciendo mayor competencia y calidad de programas.

El análisis que se inicia a partir de este trabajo, será independiente de un lenguaje de programación específico, lo cual permite que los temas se traten en forma imparcial.

La programación puede ser definida en dos partes esenciales: la tecnología y su fundamento científico. La tecnología consiste en las herramientas, técnicas prácticas y estándares que permiten *hacer* un programa. El fundamento científico consiste en la parte teórica permitiendo *entender* la programación. Enseñar programación correctamente implica enseñar ambas partes: tecnología (herramientas actuales) y ciencia (conceptos fundamentales). Conocer las herramientas prepara al estudiante para el presente, y conocer los conceptos lo prepara para la evolución futura.

Paradigmas de Programación

Existen distintas escuelas de pensamiento, sobre diferentes formas de ver la programación llamadas *paradigmas*. Cada escuela tiene su ciencia.

Un paradigma de programación provee (y determina) la visión y métodos que un programador utiliza en la construcción de un programa o subprograma. Diferentes paradigmas resultan en diferentes estilos de programación y en diferentes formas de pensar la solución de los problemas. Por ejemplo, en *programación orientada a objetos* (POO), los programadores pueden pensar en un programa como una colección de objetos interactuando, mientras que en la programación funcional, un programa puede ser pensado como una secuencia de evoluciones de funciones sin estado.

Un paradigma representa las directivas en la creación de abstracciones, y es un principio por el cual un problema puede ser comprendido y descompuesto en componentes manejables. Un paradigma fija las reglas y propiedades, pero también ofrece herramientas para el desarrollo de aplicaciones.

Diferentes paradigmas de computación se han desarrollado a través de los años. Los primeros lenguajes de programación (código maquina, ensamblador y de alto nivel) se basaron sobre el paradigma imperativo, que consistía en una secuencia de comandos/sentencias con los que se operan los datos almacenados en memoria. Uno de los aspectos más remarcables de la programación imperativa es el mecanismo de *side effect* realizado en la sentencia de asignación. Esta sentencia cambia el estado del programa alterando el contenido de las posiciones de memoria. Debido al gran y continuo uso de este paradigma es que se lo considera un tema vital en la enseñanza de la programación.

Con el paradigma imperativo, los programas de gran escapa se tornaban difíciles de comprender. Con el tiempo se sumó un paradigma organizacional que se agrega al paradigma imperativo para facilitar la construcción de programas. El primero fue el paradigma procedural, en el cual el programa se divide en conjuntos de bloques de código ejecutable, llamados "procedimientos". Esta forma de abstracción permite al programador usar la aproximación "divide y conquistarás" para diseñar el flujo de control de los programas.

Una abstracción organizacional más nueva fue introducida a la programación imperativa con el paradigma orientado a objetos. En programación orientada a objetos, los datos y operaciones están encapsulados en entidades llamadas "objetos". Esencialmente un objeto consiste de datos y métodos, en terminología orientada a objetos formal, los objetos reciben mensajes de requerimientos desde los métodos (funciones) para realizar una computación.

La programación orientada a objetos también referencia a la aproximación basada en la comunicación cliente-servidor.

Un paradigma basado en la teoría de funciones recursiva de computación, es el paradigma de programación funcional. La principal característica de este paradigma es la evolución de expresiones. La programación funcional afirma que cualquier cómputo se puede expresar en términos de una secuencia de evaluación de expresiones.

Otros paradigmas de programación, como programación lógica y los sistemas basados en reglas no son tan usados en la industria.

En los últimos años, algunos autores optan y sugieren la programación orientada a objetos como primer paradigma para enseñar en los planes de estudio de las carreras de computación.

A continuación se presenta el rol crítico de utilizar más de un paradigma de programación en la enseñanza de la programación.

Programación multiparadigma

En los primeros cursos de los planes de estudio de las carreras de computación se enseña el primer lenguaje de programación, los métodos de la abstracción, y distintos paradigmas de manera separada. Las técnicas y las prácticas adquiridas para cada paradigma se aplican sobre problemas sin establecer cuál enfoque sería el más adecuado. Generalmente el problema vuelve a presentarse más adelante, si el problema requiere un acercamiento diferente. Esta situación puede obstaculizar la tarea de los programadores produciendo programas fuera del marco de calidad y tiempo previstos. Una solución posible es preparar a los futuros profesionales para identificar la naturaleza del problema que busca solucionarse y elegir conscientemente las mejores herramientas y técnicas, esto es, enseñar cómo las diferentes clases de problemas se podrían resolver usando el/los

paradigma/s apropiado/s. Como los problemas complejos tienen generalmente una naturaleza multidimensional, en la mayoría de los casos requieren una aproximación multiparadigma.

Los estudiantes de computación deberían ver más de un paradigma de programación durante su primer año de aprendizaje de programación, pues necesitan ver una variedad de opciones que pueden ser usadas en el diseño de sus programas. Hay que enfatizar que no todos los problemas encuadran fácilmente en un único paradigma. Es cierto que los problemas son más fáciles de pensar y de resolver en un modelo en comparación con otros modelos, pero algunos problemas pueden requerir usar múltiples modelos.

La programación orientada a objetos domina ampliamente en el área industrial de la computación, logrando un diseño e implementación que satisface la funcionalidad básica y con calidad aceptable, sin embargo, existen conceptos que este paradigma no maneja debido a que atraviesan todo el sistema, o varias partes de él. Algunos de estos conceptos son: sincronización, manejo de memoria, distribución, chequeo de errores, seguridad, entre otros.

Las técnicas de implementación actuales tienden a implementar los requerimientos usando metodologías de una sola dimensión, forzando los requerimientos a ser expresados en esa única dimensión. Esta dimensión resulta adecuada para la funcionalidad básica y no para el resto de los requerimientos, los cuales quedan diseminados a lo largo de la dimensión dominante. Es decir, que mientras el espacio de requerimientos es de n-dimensiones, el espacio de la implementación es de una sola dimensión. Dicha diferencia resulta en un mapeo deficiente de los requerimientos a sus respectivas implementaciones.

Existen lenguajes que soportan multiparadigmas que contienen los conceptos de los mayores paradigmas de programación, incluyendo lógico, funcional, imperativo, orientado a objetos, por restricciones, distribuido y concurrente. Existen alternativas se puede enseñar programación multiparadigma sin un lenguaje que lo soporte formalmente. Una alternativa es emular las características usando un lenguaje que esencialmente soporta un único paradigma. Otra posibilidad es usar lenguajes separados, cada paradigma tiene una forma de procesamiento (datos, protocolos de llamadas a procedimientos e infraestrutura interoperativa).

Enseñando a programar

Los planes de estudios tienen una carga horaria determinada que no hace posible incluir la enseñanza completa de todos los paradigmas de programación. Los nuevos paradigmas pueden co-existir fácilmente con los viejos. Por ejemplo, la programación orientada a aspectos tiene la misma estructura base que la programación orientada a objetos. El aspecto disminuye la repetición de código dentro de las clases y mejora la modularidad de un programa. La programación procedural también reúne principios del estilo orientado a objetos, tiene varias construcciones básicas como tipos de datos abstractos y clases. La programación funcional también presenta similitudes con la programación orientada a objetos, provee tipos de dato abstracto, clases, encapsulamiento, subtipos y estructuras básicas. Existen otros conceptos comunes en dos paradigmas, como estructuras de datos y funciones genéricas o polimorfismo parametrizable.

Es importante tratar de distribuir la mayor cantidad de paradigmas a lo largo de los distintos cursos de la curricula de computación de manera que el alumno incorpore y relacione las características que cada paradigma provee, brindando los recursos necesarios para su posterior adaptación a nuevos paradigmas.

Es conveniente tratar de no descartar en su totalidad algunos paradigmas ya que un paradigma no sustituye a otro, y excluir alguno podría ser en deterioro de la calidad de los futuros profesionales en programación o ingeniería de software. Por ejemplo, en los primeros cursos de análisis de algoritmos, todas las técnicas son importantes: divide y vencerás, voraces, programación dinámica,

etc., enfatizando la divide y vencerás, en forma similar ocurre con los paradigmas, durante el transcurso de la carrera se puede enfatizar en programación orientada a objetos pero no es adecuado que éste sustituya la enseñanza de otros paradigmas, ya que tendrían una visión parcial del espectro de posibilidades al momento de resolver un problema en programación.

Una de las cuestiones más críticas es en qué momento de la curricula se incluye cada paradigma de programación. Como la programación imperativa es necesaria en casi todos los cursos, es obvio que se deberá enseñar en los cursos introductorios de programación. A partir del segundo curso de programación sería apropiado comenzar con conceptos introductorios de programación orientada a objetos.

La educación de alto nivel debería acentuar la enseñanza teórica y práctica de los nuevos paradigmas y sus herramientas y entornos de programación que soportan el lenguaje. Los estudiantes pueden profundizar su base de conocimiento adquiriendo nuevos estilos y paradigmas de programación, ya que tendrán disponible el uso de nuevas herramientas y tecnologías en sus trabajos futuros. Es muy importante tener en cuenta que un nuevo paradigma nunca deja de lado las experiencias de programación ganadas previamente. La programación estructural sobrepasó las nociones de los primeros lenguajes imperativos. El paradigma orientado a objetos no cancela la implementación estructural en la implementación de sus métodos. El alternar paradigmas incorpora todas las herramientas, métodos y experiencias ya existentes en una unidad estructural superior.

Conclusión y trabajo futuro

Se ha presentado una breve descripción de los distintos paradigmas que se incluyen en los planes de estudios de las carreras de computación. El principal punto que se desea resaltar es que cuando los estudiantes completan los cursos de programación (y aquellos otros cursos que involucren la programación en dictado) deberían tener una cantidad pre-determinada de conocimiento de los paradigmas de programación utilizados durante el cursado de la carrera, que les permitirá determinar cuál sería el adecuado, para lo que se apunta a establecer el alcance que abarca la enseñanza de los distintos paradigmas y el impacto que produciría el uso de un lenguaje multi-paradigma en los primeros cursos de programación.

Desde una perspectiva pedagógica se buscará establecer una lista de problemas adecuados para una paradigma particular y, de la misma manera una lista de problemas que pueden encararse a través de distintos paradigmas.

Existen lenguajes multi-paradigmas que pueden ser usados a nivel académico que aumentan las características fundamentales de cada paradigma.

Se ha seleccionado el lenguaje Oz (http://www.mozart-oz.org/) como lenguaje multi-paradigma para implementar algunos temas de las asignaturas de programación y poder analizar el impacto.

Bibliografía

Van Roy Peter, Haridi Seif – "Concepts, Techniques, and Models of Computer Programming" - MIT Press, 2004

Laxmi P Gewali and John T Minor – "Multi-Paradigm Approach for Teaching Programming" Proceedings of the 2006 International Conference on Frontiers in Education: Computer Science & Computer Engineering, FECS 2006, Las Vegas, Nevada, USA, June 26-29, 2006.

Zoltán Porkoláb, Viktória Zsók - "Teaching Multiparadigm Programming Based on Object-Oriented Programming" - 10th Workshop on Pedagogies and Tools for the Teaching and Learning of Object-Oriented Concepts,TLOOC Workshop, ECOOP 2006, Nantes