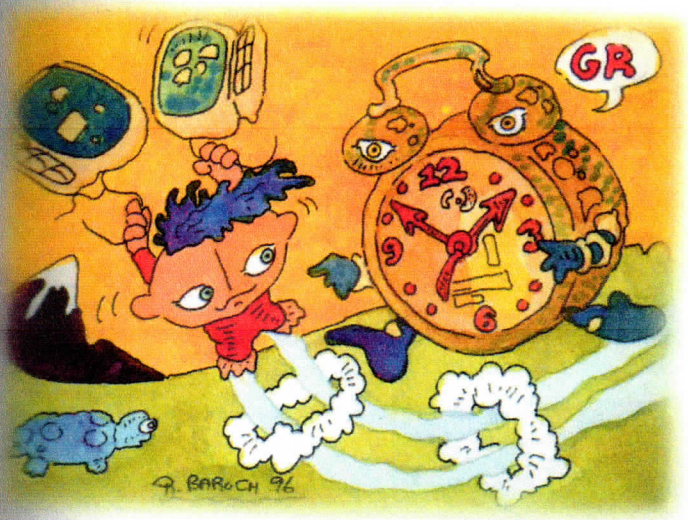
# Paralelismo y Orientación a Objetos




Jorge Luis Ortega Arjona
*DEA-IIMAS-Universidad Nacional Autónoma de México*

***Este artículo pretende introducir a la programación paralela orientada a objetos como una opción importante dentro de las tendencias futuras en la computación. La conjunción de paralelismo y orientación a objetos se basa en los conceptos de objeto y paso de mensaje, esto es, la inclusión de los objetos a una arquitectura multiprocesador que a su vez aporta características de comunicación entre objetos mediante el paso de mensaje como mecanismo básico. Por otro lado, se mencionan los modelos de programación paralela orientada a objetos más comúnmente utilizados, como son el paralelismo de tareas, paralelismo de datos y los objetos comunicantes. Finalmente, se expone en forma breve algunas de las principales ventajas de la programación paralela orientada a objetos, y algunas áreas de aplicación.***

Jorge Luis Ortega Arjona está por obtener el grado de Maestro en Ciencias de la Computación, del IIMAS-UNAM. Trabaja en el Laboratorio de Procesamiento Paralelo, del Departamento de Electrónica y Automatización, IIMAS-UNAM. Areas de interés: procesamiento paralelo, programación OO, procesamiento de imágenes.
Email: JOrtega@leibniz.iimas.unam.mx

## 1 Introducción

Las tendencias de la tecnología de computación actualmente se encuentran dirigidas principalmente por dos que involucran tanto el *hardware* como el *software*: las arquitecturas paralelas y la programación orientada a objetos [7]. Hasta fechas recientes, ambas se habían tratado por separado en sus etapas de desarrollo. Actualmente un amplio grupo de investigadores proponen la conjunción de ambas tendencias en una programación paralela orientada a objetos o, en forma más general, una programación concurrente orientada a objetos. Se han realizado varios trabajos considerando las posibilidades y conveniencias de tal unión en ambientes concurrentes [1]. En contraste, el presente trabajo pretende únicamente considerar algunos de los aspectos que particularmente surgen de la aplicación de la orientación a objetos a una arquitectura paralela.

Es notable que la programación orientada a objetos (POO) se esté diseminando rápidamente en la comunidad de quienes programan sistemas paralelos. Sin embargo, la labor de transferir el conocimiento y experiencia desarrollada para sistems paralelos a una POO paralela no se plantea como una tarea sencilla, debido a que algunos mecanismos desarrollados para la OO requieren aún de evolucionar para ajustarse correctamente al concepto de paralelismo y, por otro lado, otras dificultades aparecen debidas a la inmadurez relativa en el campo del paralelismo, lo que ha dado como resultado una abundancia de prototipos de lenguajes paralelos y OO [3,4].

## 2 Conceptos de la programación paralela y orientada a objetos: Objetos y mensajes.

La programación paralela orientada a objetos se basa en dos conceptos principalmente: objetos, que identifican conocimiento en forma de datos y servicios, y el paso de mensaje como protocolo unificado de comunicación en un sistema multiprocesador (figura 1).

Los objetos representan entidades y conceptos que componen el dominio del problema, los cuales no obedecen a un algoritmo global como modelo del problema, sino que cooperan entre sí intercambiando información mediante mensajes.

Su implementación se basa en conceptos útiles de la POO, como la abstracción (en forma de clases) y la herencia (en forma de subclases), las cuales permiten especificar, clasificar y reutilizar las descripciones de los objetos. De esta forma, el paralelismo es un elemento importante que se incorpora a la POO, presentándose como la habilidad de expresar simultaneidad de acciones dentro de un programa, lo que aumenta considerablemente la eficiencia en su ejecución. Se representa mediante mecanismos de sincronización y comunicación entre procesos paralelos como es el caso del paso de mensaje [2,6].

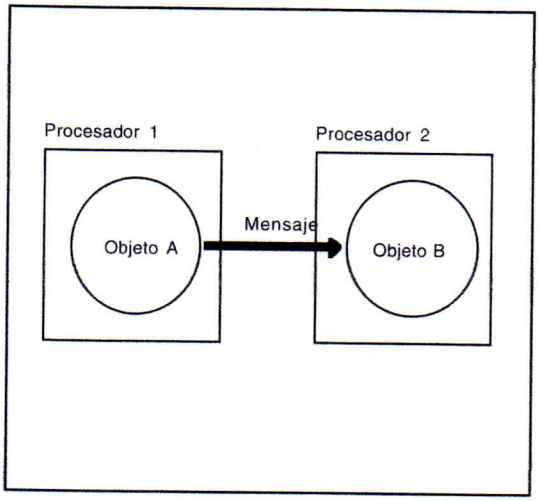



**Figura 1.** Objetos y paso de mensaje.

La integración de la OO con el paralelismo da como resultado la programación orientada a objetos paralela (POOP), la cual aparece como la generalización natural de la POO [3,4]. Esto permite la creación de objetos con una mayor autonomía en su actividad al exhibir paralelismo. Este se expresa a nivel objeto de dos maneras principalmente: una inter-objeto, en la cual cada objeto posee una existencia simultánea tanto en tiempo como en espacio; y el paralelismo intra-objeto, la cual permite a un objeto manejar varias acciones y(o) mensajes concurrentemente. Para lograr ambos efectos, la POOP se basa en algún modelo de programación paralela [2].

## 3 Modelos de programación

Los modelos de programación para procesamiento paralelo se relacionan estrechamente con la naturaleza del problema a solucionar. En función de tal naturaleza, un programa se estructura siguiendo un modelo determinado. Son tres los modelos más utilizados: estructuración explícita en tareas paralelas, tareas estructuradas a partir de los datos, y sistemas de objetos comunicantes [5].

### 3.1 Tareas Paralelas

El paralelismo de tareas se basa en la creación de hebras (hilos) de control independientes, que incluyen elementos para su sincronización y comunicación. Un objeto se trata de una estructura que ejecuta el programa como una colección de tareas paralelas controladas de una manera explícita (figura 2) [5].

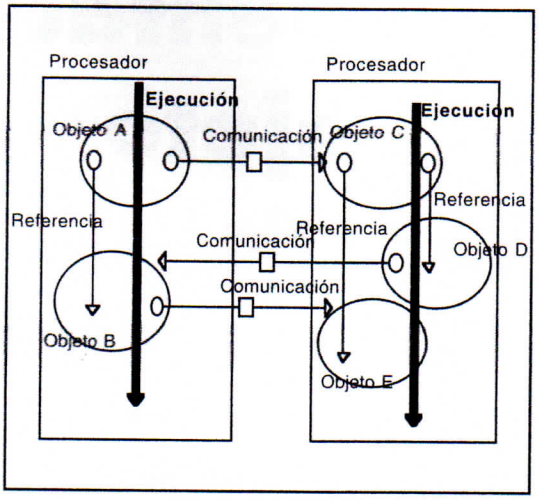



**Figura 2.** Tareas paralelas.

Existe una variedad de elementos y estructuras de sincronización y comunicación en diferentes lenguajes orientados a objetos paralelos. Se basan principalmente en el uso de bloques paralelos (*parbegin/parend*), primitivas de espera de un booleano, mecanismos de suspensión (*futures*), serialización del acceso a las instancias de una clase en particular, referencia de objetos y funciones miembros, y uso de primitivas explícitas para paso de mensajes [5].

### 3.2 Paralelismo de datos

El paralelismo de datos es el término utilizado para describir la aplicación en paralelo de un operador atómico o secuencial sobre un conjunto de datos. Se considera una extensión del concepto de función miembro de la POO. Esto quiere decir que en el modelo de paralelismo de datos una función miembro perteneciente a una clase tiene la capacidad de ejecutarse en paralelo sobre un conjunto de datos [2,5]. Un objeto realiza en paralelo tareas estructuradas respecto a los datos que manipulan (figura 3). Frecuentemente, el compilador utilizado se encarga de realizar los ajustes necesarios para organizar las operaciones sobre los conjuntos de datos definidos [5].

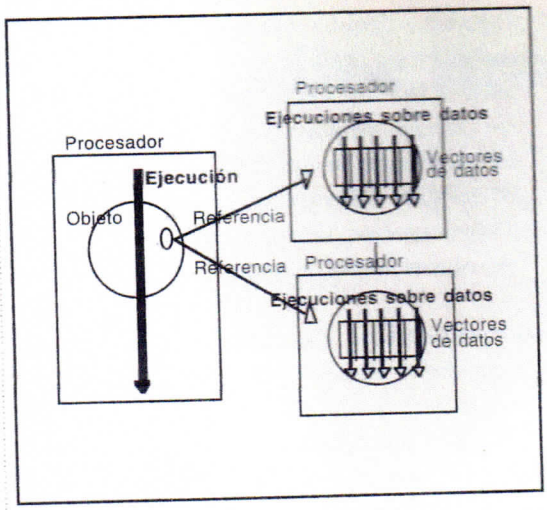



**Figura 3.** Tareas respecto a los datos.

todos que pueden ser ejecutados concurrentemente.

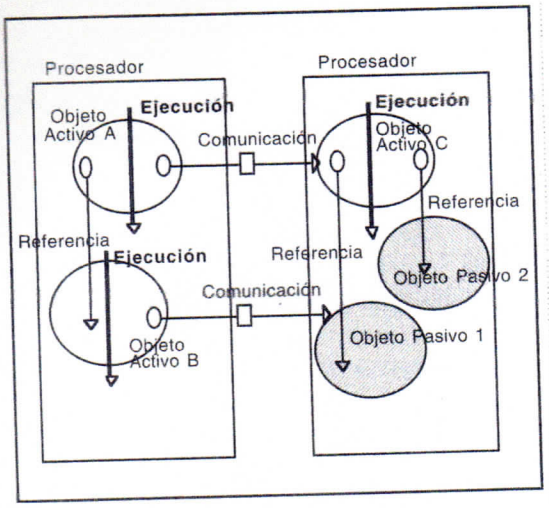



**Figura 4.** Objetos comunicantes.

### 3.3 Objetos comunicantes

El modelo más aceptado entre quienes realizan POOP es el modelo de objetos comunicantes. En este modelo, cada objeto representa una unidad de procesamiento autónoma que se comunica activamente con los demás (figura 4). De esta manera, es posible distinguir dos tipos de objetos: pasivos y activos [2,5].

Los objetos pasivos son en realidad muy semejantes a aquéllos creados en programación secuencial. Constan de un conjunto de datos y métodos que pueden ser ejecutados concurrentemente.

Los objetos activos difieren de los objetos pasivos en que además de contar con estados y operaciones, contienen un proceso que permite ejecutar sus funciones en paralelo con el resto del programa. Son la integración de los conceptos de la POO y el paralelismo, ya que tienen la capacidad de realizar labores de sincronización y comunicación a través de mensajes, y de contro-

lar la ejecución de sus métodos concurrentemente. Esto permite mantener una cierta simplicidad y modularidad, reforzando los conceptos de autonomía y autocontención de los objetos [5].

Un programa basado en el modelo de objetos comunicantes se observa como un conjunto de objetos activos que en general se ejecutan en paralelo en diferentes procesadores, junto con un conjunto de objetos pasivos que son utilizados y manipulados por los objetos activos. Esto permite una mayor expresividad, en el modelado orientado a objetos, de aplicaciones.

## 4 Ventajas del paralelismo orientado a objetos

Debido a sus características de abstracción y encapsulación, un programa orientado a objetos se compone de pequeños objetos o módulos cooperativos, que se ejecutan eficientemente en paralelo mediante alguna arquitectura multiprocesador. El uso del paso de mensaje como mecanismo de comunicación entre objetos hace que tal programación sea apropiada para arquitecturas con memoria compartida y distribuida. Las ventajas obtenidas son las siguientes [2]:

**Alto nivel de abstracción**. Un programa se realiza considerando la interacción entre diferentes tipos de entidades conceptuales que describen el problema, sin atender a problemas de implementación de sus elementos.

**Descomposición implícita de actividades paralelas**. Las unidades de programación se identifican claramente al descomponer un programa en un conjunto de objetos interactivos. Cada objeto se considera como una unidad autónoma que puede efectuar además acciones concurrentes a nivel interno.

**Transparencia en sincronización**. La sincronización de actividades simultáneas se considera en el diseño de los objetos interactivos. No es necesario hacer consideraciones de sincronización a bajo nivel.

**Localización y autocontenido**. Los objetos son entidades autocontenidas que conjuntan un cierto grado de conocimiento, acciones y recursos. Su localización en diferentes procesadores permite su ejecución paralela, obteniendo la existencia una característica espacio-temporal.

**Dinamicidad**. Dependiendo del sistema multiprocesador, los objetos pueden ser creados y recuperados dinámicamente, adquirir nuevo conocimiento y cambiar su comportamiento respecto a la información que reciban, o pueden ser creados atendiendo a la necesidad de recursos durante la ejecución de un programa. Un conjunto organizado de objetos puede ser dinámicamente reconfigurado atendiendo a requerimientos del sistema.

**Multigranularidad**. Los objetos de diversas granularidades, es decir, de diferente tamaño en cuanto al conocimiento que contienen, pueden cohabitar y cooperar entre sí en el mismo sistema, lo que permite la descomposición de un diseño complejo en varios niveles.

## 5 Conclusiones

Un programa paralelo basado en el modelo de objetos es en realidad un conjunto de abstracciones en forma de clases, las cuales se definen mediante un conjunto de tipos de datos que agrupa, además, operaciones entre tales datos, y que además se comunican entre sí utilizando mensajes. Esto hace que el campo de utilización de esta programación se forme de aplicaciones donde la expresividad de la simultaneidad de acciones como modelo de la realidad sea necesaria [7]. Tales aplicaciones de la POOP se pueden hallar en áreas donde la expresión distribuida del conocimiento tiene un papel importante, como en sistemas operativos, inteligencia artificial, simulación distribuida, bases de datos, sistemas de información, sistemas de tiempo real, procesos de control, etc. Otras áreas de aplicación pueden ser análisis de textos y música por computadora [2]. ▼


**Referencias:**

[1] Akerbaek, T., "C++, Coroutines, and Simulation", *The C Users Journal*, march 1993.

[2] Briot, J.P., "Object-Oriented Concurrent Programming: Introducing a New Programming Methodology" *Proceedings of the 7th International Meeting of Young Computer Scientist* (IMYCS'92), Topics in Computer Science Series, Gordon & Breach, 1993.

[3] Chandy, M., and Kesselman, C., "CC++: A declarative Concurrent Object-Oriented Programming notation", Research Directions in Concurrent Object-Oriented Programming. The MIT Press, Cambridge, Mass., 1993.

[4] Wyatt, B.; Kavi, K., and Hufnagel, S., "Paralelism in Objetc-Oriented languajes: a survey", *IEEE Software*, november 1992.

[5] Furmento, N.; Roudier, Y., and Siegel, G., "Parallélisme et Distribution en C++. Un revue des langages existants", Rapport de Recherche I3S RR95-02.

[6] Lewis, T, and El-Rewini, H., "Introduction to parallel computing", Prentice-Hall, 1992.

[7] Lewis, T., "Where is computing headed?", *Computer*, IEEE Computer Society, august 1994.