Patrones para Diseño de Software Paralelo

Jorge L. Ortega Arjona
Departamento de Matemáticas
Facultad de Ciencias, UNAM

Procesamiento Paralelo y Programación Paralela

- **Procesamiento Paralelo**: división de una labor de procesamiento entre múltiples procesadores que operan simultáneamente con un objetivo común.
  - El resultado esperado es, comúnmente, una terminación más rápida del objetivo en comparación con su ejecución en un solo procesador.
  - Su principal ventaja es la habilidad de manejar tareas de una escala que no sería realista o costo efectivo para otros sistemas.

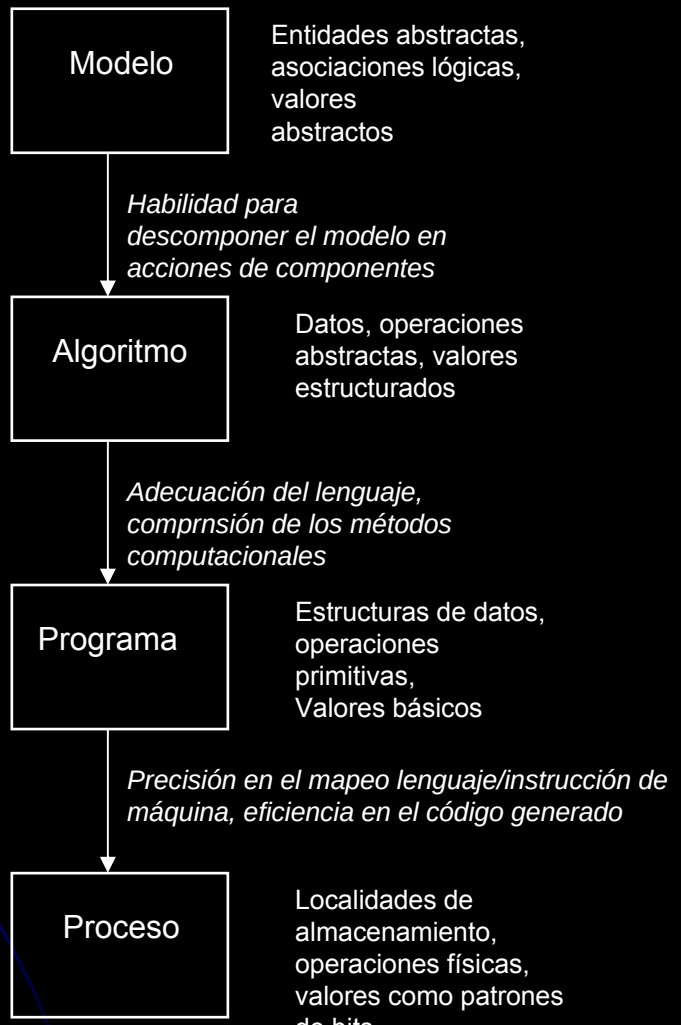
- **Programa Paralelo:** Especificación de un conjunto de procesos que se ejecutan simultáneamente, comunicándose entre sí para lograr un objetivo común.

- **Desempeño:** se refiere a la *responsividad* de un sistema — el tiempo requerido para responder a estímulos (eventos) o el número de eventos procesados en un intervalo de tiempo.

Desempeño de un programa paralelo

- Plataforma Paralela (Hardware y Software)
- Lenguaje de Programación
- Problema a resolver

Problema a Resolver



Programa Paralelo = Coordinación + Procesamiento

- **Coordinación:** conjunto de acciones que permiten la cooperación entre componentes mediante su organización en el tiempo y espacio y el acceso de datos a donde pueden operarse.

  Cambio en el estado de las variables remotas.

  - **Comunicación:** intercambio de datos o solicitud de operaciones entre componentes.

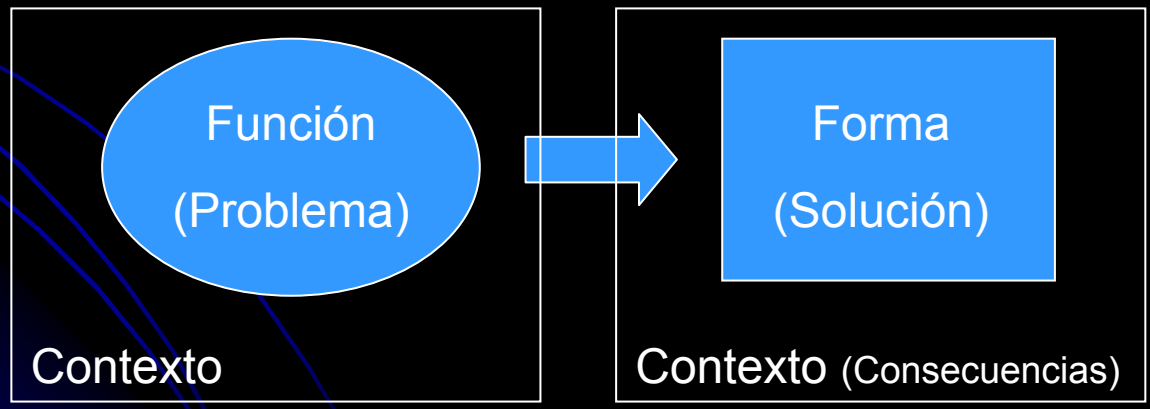
    - **Sincronización:** organización de acciones sobre datos y funciones por parte de diferentes componentes.

- **Procesamiento:** cambio en el estado de las variables locales.

Patrones de Software

Un **patrón de software** (*software pattern*) es una relación forma-función que ocurre en un contexto.

- la función se describe en términos del dominio del problema como un grupo de conflictos de fuerzas sin resolver;
- la forma es una estructura descrita en términos del dominio de la solución que logra un equilibrio adecuado y aceptable entre esas fuerzas.



Categorías de Patrones de Software

- Patrones Arquitectónicos (*Architectural Patterns*)
- Patrones de Diseño (*Design Patterns*)
- Modismos (*Idioms*)

Ejemplo: *Manager-Workers*

- **Name:**
  Manager-Workers pattern.

- **Context:**
  Start the design of a parallel program, using a particular programming language for certain parallel hardware. Consider the following context constraints:

  - The problem involves tasks of a scale that would be unrealistic or not cost-effective for other systems to handle, and lends itself to be solved using parallelism.
  - The parallel hardware platform or machine to be used is given.
  - The main objective is to execute the tasks in the most time-efficient way.

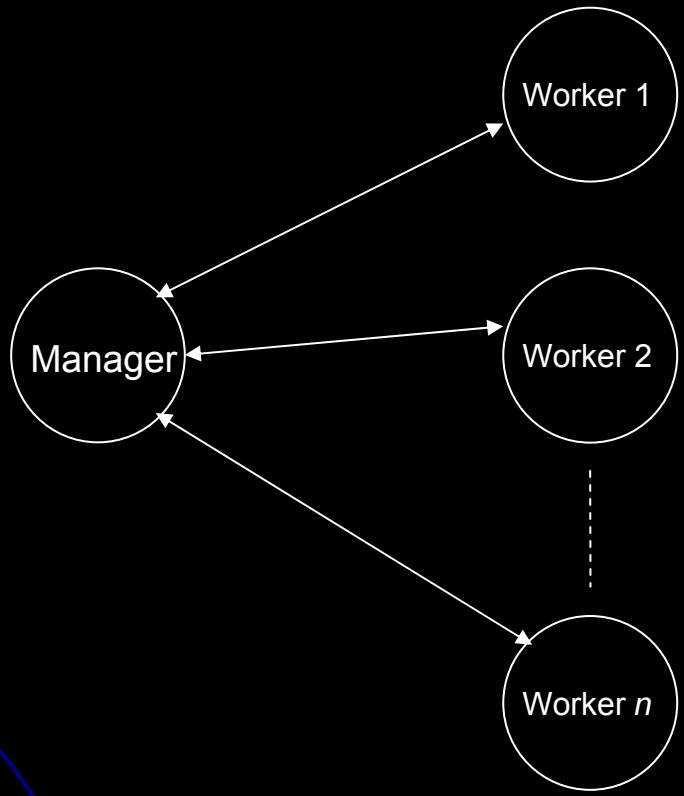
- **Problem:**
  The same operation is required to be repeatedly performed on all the elements of some ordered data. Data can be operated without a specific order. However, an important feature is to preserve the order of data. If the operation is carried out serially, it should be executed as a sequence of serial jobs, applying the operation to each datum one after another. Generally, performance as execution time is the feature of interest, so the goal is to take advantage of the potential simultaneous execution in order to carry out the whole computation as efficiently as possible.

- **Forces:**
  The following forces should be considered:

  - Preserve the order of data. However, the specific order of operation on each piece of data is not fixed.
  - The operation can be performed independently on different pieces of data.
  - Data pieces may exhibit different sizes.
  - The coordination of the independent computations has to take up a limited amount of time in order not to impede performance of other processing elements.
  - Mapping the processing elements to processors has to take into account the interconnection among the processors of the hardware platform.

- **Solution:**
  Introduce activity parallelism by having multiple data sets processed at the same time. The most flexible representation is the Manager-Workers approach. This structure is composed of a manager component and a group of identical worker components. The manager is responsible of preserving the order of data. On the other hand, each worker is capable of performing the same independent computation on different pieces of data. It repeatedly seeks a task to perform, performs it, and repeats; when no tasks remain, the program is finished.

Ejemplo: *Manager-Workers*



Patrones para Diseño de Software Paralelo

- Conjunto de patrones de software que desarrollan la coordinación, la comunicación y la sincronización de un sistema de software paralelo.

- Método de diseño:
  1. Análisis del problema
  2. Diseño de la coordinación
  3. Diseño de la comunicación
  4. Diseño detallado
  5. Implementación y pruebas

Patrones para Diseño de Software Paralelo

Coordinación:

- ***Architectural Patterns for Parallel Programming***

  - *Parallel Pipes and Filters*
  - *Parallel Layers*
  - *Communicating Sequential Elements*
  - *Manager Workers*
  - *Shared Resource*

Patrones para Diseño de Software Paralelo

Comunicación:

- ***Design Patterns for Communication Components***

  - *Shared Variable Pipe*
  - *Message Passing Pipe*
  - *Shared Variable Channel*
  - *Message Passing Channel*
  - *Multiple Local Call*
  - *Multiple Remote Call*
  - *Local Rendezvous*
  - *Remote Rendezvous*

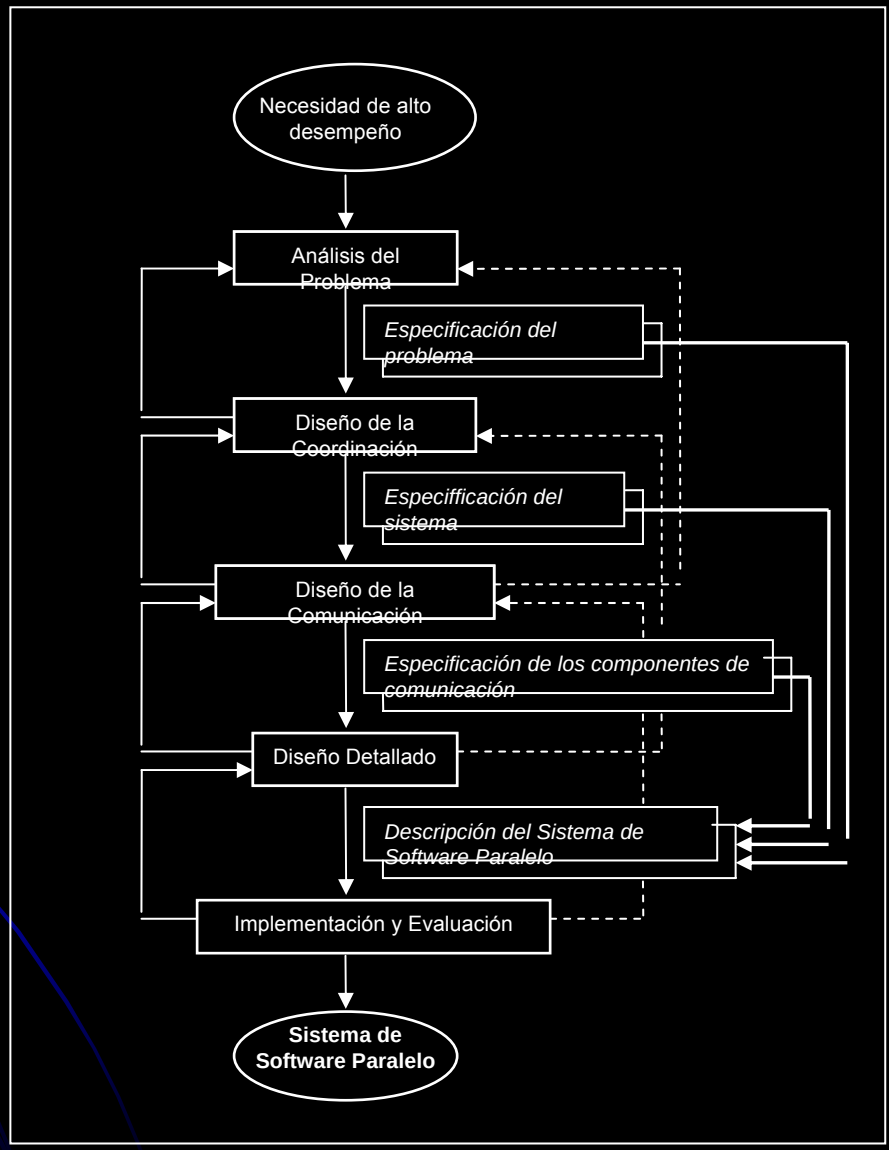
Patrones para Diseño de Software Paralelo

Sincronización:

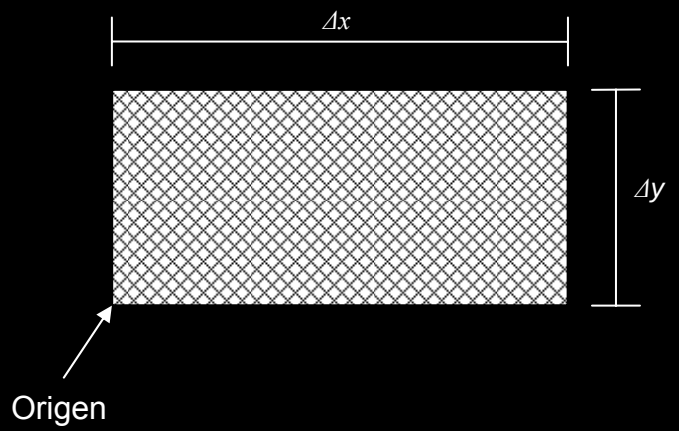
- ***Idioms for Synchronization Mechanisms***

  - *Semphore idiom*
  - *Critical Region idiom*
  - *Monitor idiom*
  - *Message Passing idiom*
  - *Remote Procedure Call idiom*

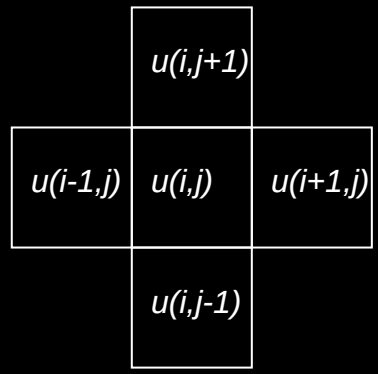
Método de Diseño



Un ejemplo: La Ecuación Bi-dimensional de Calor



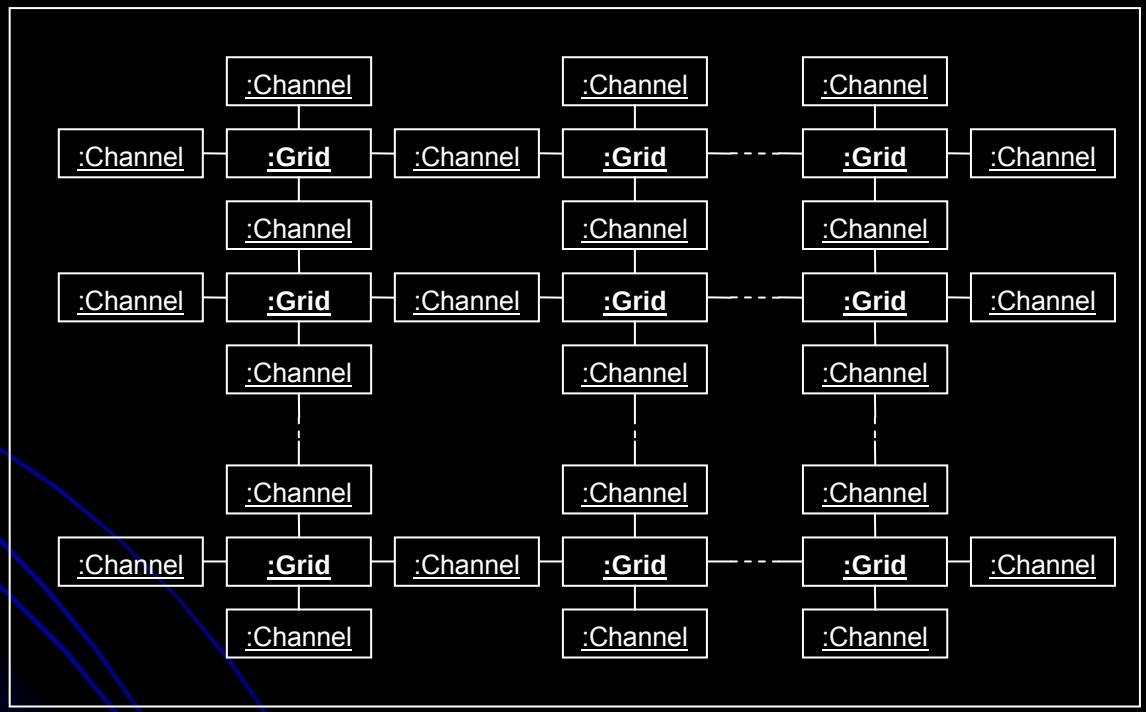
Análisis del problema



$$u(i, j) \approx u(i - 1, j) + u(i + 1, j) + u(i, j - 1) + u(i, j + 1) / 4$$

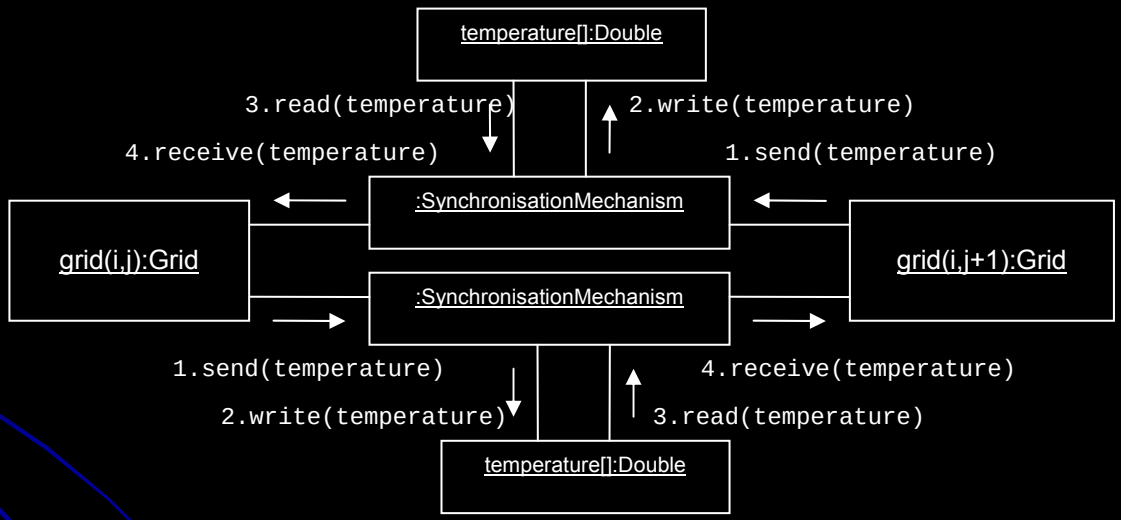
Diseño de la Coordinación

Communicating Sequential Elements pattern



Diseño de la Comunicación

Shared Variable Channel pattern



Implementación de la Sincronización

Monitor Idiom

```
import java.util.Vector;
class Monitor {
        private int numMessages = 0;
        private final Vector temperatures = new Vector();
        public final synchronized void write(double temp){
                if(temp == null) throw new NullPointerException();
                numMessages++;
                temperatures.addElement(temp);
                if(numMessages <= 0) notify();
        }
        public final synchronized double read(){
                double temp = 0.0d;
                numMessages--;
                while(numMessages < 0){
                        try{
                                wait();
                                break;
                        }
                        catch(InterruptedException e){
                                if(numMessages >=0) break;
                                else continue;
                        }
                }
                temp = temperatures.firstElement();
                temperatures.removeElementAt(0);
                return temp;
        }
}
```

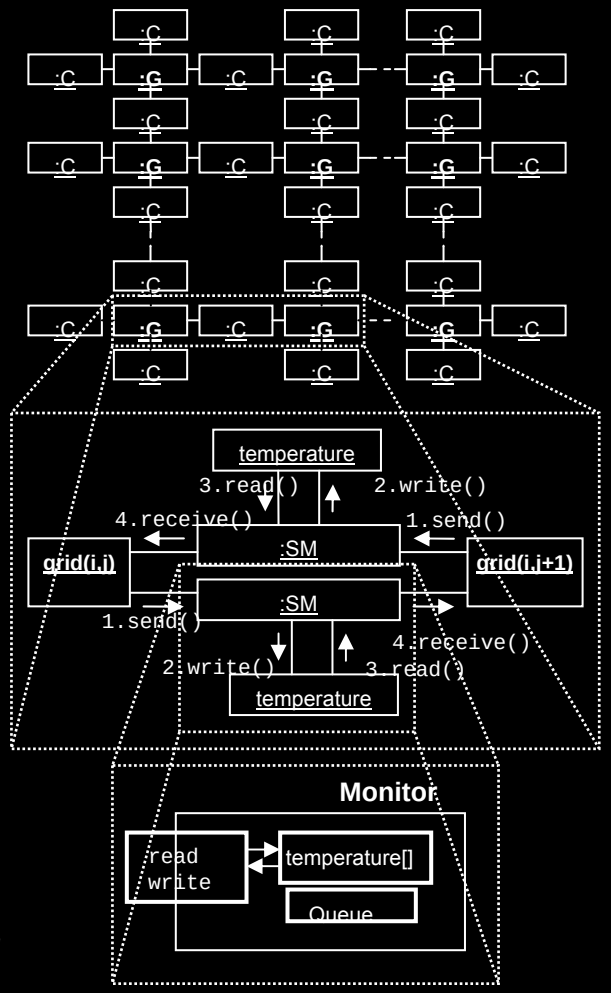
Descripción del Diseño del Sistema Paralelo

Una Descripción del Diseño del Sistema Paralelo para resolver la Ecuación Bidimensional de Calor

Coordinación: *The Communicating Sequential Elements architectural pattern*

Componentes de Comunicación: *The Shared Variable Channel design pattern*

Mecanismo de Sincronización: *The Monitor idiom*

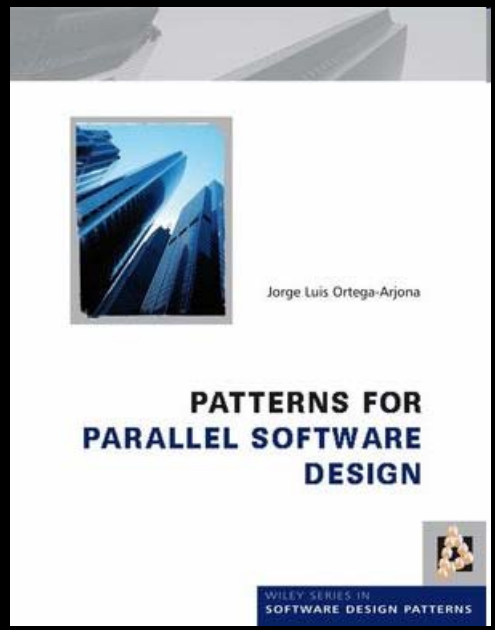


Procesamiento

temperature += (total/4.0 – temperature);

Patrones de Software para Diseño de Software Paralelo

- ***Patterns for Parallel Software Design***
  Jorge L. Ortega-Arjona
  John Wiley & Sons, 2010.



Referencias

1. **Architectural Patterns for Parallel Programming.** Jorge L. Ortega-Arjona and Graham Roberts. *Proceedings of the 3rd European Conference on Pattern Languages of Programming and Computing (EuroPLoP'98).* Kloster Irsee, Germany. 9-12 of july, 1998.
2. **The Communicating Sequential Elements Pattern. An Architectural Pattern for Domain Parallelism.** Jorge L. Ortega-Arjona.*7th Pattern Languages of Programming 2000 (PloP2000).* Allerton Park, Illinois, USA. 13-15 august, 2000.
3. **The Shared Resource Pattern. An Activity Parallelism Architectural Pattern for Parallel Programming.** Jorge L. Ortega-Arjona.*10th Pattern Languages of Programming 2003 (PLoP2003).* Allerton Park, Illinois, USA. 8-12 september, 2003.
4. **The Manager-Workers Pattern. An Activity Parallelism Architectural Pattern for Parallel Programming.** Jorge L. Ortega-Arjona. *9th European Conference on Pattern Languages of Programming and Computing 2004 (EuroPLoP2004).* Kloster Irsee, Germany. 7-11 july, 2004.
5. **The Pipes and Filters Pattern. A Functional Parallelism Architectural Pattern for Parallel Programming.** Jorge L. Ortega-Arjona. *10th European Conference on Pattern Languages of Programming and Computing (EuroPLoP 2005).* Kloster Irsee, Germany, 6-10 july, 2005.
6. **The Parallel Layers Pattern. A Functional Parallelism Architectural Pattern for Parallel Programming.** Jorge L. Ortega Arjona. *6th Latin American Conference on Pattern Languages of Programming (SugarLoafPLoP 2007).* Porto de Galinhas, Pernambuco, Brasil. 25-31 May, 2007.
7. **Design Patterns for Communication Components of Parallel Programs.** Jorge L. Ortega Arjona.*12th European Conference on Pattern Languages of Programs (EuroPLoP 2007).* Kloster Irsee, Germany. 4-8 july, 2007.