# Planeación de Requerimientos de Materiales incorporando Lógica Difusa para tratar la incertidumbre

Alejandro Domingo Velázquez-Cruz[1], Hilarión Muñóz-Contreras[1] , Jorge Armando Rojas-Ramírez[2], Ulises Jesús López-Maldonado[1]

[1] Instituto Tecnológico de Orizaba, Departamento de Posgrado e Investigación, Avenida Instituto Tecnológico #852, Colonia Emiliano Zapata. Orizaba, Veracruz.
[2] Instituto Politécnico Nacional, Escuela Superior de Ingeniería Mecanica y Electrica, Avenida Instituto Politécnico Nacional s/n. Delegación Gustavo A. Madero, Colonia Zacatenco, México, D.F.
lap.alejandro@acm.org, hilarionm@itorizaba.edu.mx, jrojasr@ipn.mx, lopezu@acm.org

**Resumen.** Este trabajo revisa los beneficios de incorporar lógica difusa en la planeación de requerimientos de materiales (Material Requirements Planning, MRP) para separar el desarrollo del sistema en abstracciones separadas para aislar las diferentes funcionalidades en diversas capas y enlazarlas con la funcionalidad original a través de una capa intermedia. Este tipo de diseño arquitectónico permite tratar la complejidad en diferentes abstracciones. Una de las abstracciones resuelve el problema del MRP mientras las otras manejan la incertidumbre a través de la lógica difusa.

**Abstract.** This work reviews the benefits of incorporate fuzzy logic into material requirements planning to treat uncertainty, for achieving such purpouse is necessary to separate the development of the system into separate abstractions to isolate the different functionalities in layers, and to link with the original functions throught an intermediate layer. This kind of architectural design allows to threat complexity in diffent abstractions. One of the abstractions resolves the MRP problem, while the other handles uncertainty throught fuzzy logic.

**Palabras Clave:** MRP, Lógica Difusa, incertidumbre, determinación, características.

## 1 Introducción

En México existen empresas con problemas para organizar sus procesos productivos, teniendo una baja calidad, retrasos en la entrega, así como una inadecuada planeación de sus operaciones. Algunos de los problemas que ocasiona tal situación son altos inventarios, así como desperdicio de materiales y tiempo, por ello es importante mejorar los sistemas productivos, mejorando la planeación y control de operaciones.

M.A. Cruz-Chávez (Ed): CICos 2011, ISBN. 978-607-00-5091-6. pp. 233 – 240, 2011

Para poder reducir los inventarios y el correspondiente desperdicio de materiales y tiempo es necesario tener una adecuada planeación de la producción, así como de los requerimientos de materiales, dicha información incide de forma definitiva en la capacidad de cualquier empresa para poder tener una eficiente gestión de recursos.

Es por ello que en toda empresa que se dedique a la producción, el conocer la cantidad de materiales que serán necesarios para cubrir las necesidades de su entorno es información vital que incide de manera definitiva en la capacidad de la empresa de gestionar de manera eficiente sus recursos.

La planeación de los requerimientos de materiales es una actividad de suma importancia para gestionar los requerimientos materiales que previene incrementos innecesarios en el inventario, además de que permite calcular los elementos que serán necesarios para llevar a cabo una determinada producción.

El MRP proporciona con información confiable, que permite una adecuada toma de decisiones. Los sistemas MRP han probado ser de suma utilidad en reducir el desperdicio de insumos en producción. Sin embargo, necesitan valores numéricos precisos y no pueden manejar variables que tengan un cierto grado de incertidumbre, lo que limita la posibilidad de aplicarlos cuando alguno de los valores es incierto.

Para poder aplicar el MRP en aquellos sistemas en los que existe incertidumbre se utiliza la Lógica Difusa (LD), la cual permite tratar la incertidumbre del sistema, permitiendo entonces desarrollar sistemas MRP que trabajen con valores inciertos aprovechando la teoría de los conjuntos difusos para tratar esos valores.

Sin embargo la incorporación de LD en un MRP ya existente puede resultar en algo muy complicado de implementar a menos de que desde su concepción la arquitectura contemple la posibilidad de extender su funcionalidad para incorporar nuevos componentes al sistema proveyéndole de la capacidad de evolucionar.

## 2 Planeación de Requerimientos Materiales

Un sistema MRP permite planear los requerimientos de insumos en base a la necesidad real, reduciendo los costos ya que solo se compra aquello que es necesario, su principal objetivo es controlar el proceso de producción donde existen múltiples etapas intermedias en las que los materiales empleados sufren diversas transformaciones, donde una vez terminado un producto el mismo puede convertirse en un componente de otro hasta la terminación del producto final.

Un sistema MRP es una solución que permite controlar y coordinar los materiales para que estén disponibles cuando se requieran sin la necesidad de tener un inventario de gran tamaño. El MRP determina cuántos componentes son necesarios, convirtiéndolos en órdenes concretas de compra de materiales y de fabricación de cada uno de los productos que se desea obtener [1].

El sistema debe poder asegurar que los materiales necesarios para la producción se encuentren presentes y en caso de que no sea así elaborar las órdenes de compra correspondientes manteniendo el mínimo nivel de inventario posible.

El procedimiento del MRP se basa en dos ideas centrales:

- La demanda de los componentes no es independiente, únicamente lo es la de los productos terminados.

- La necesidad de cada artículo debe poder calcularse a partir de las demandas independientes y la estructura del producto.

Un MRP cuando menos tiene tres subsistemas principales que le proporcionan la información necesaria, que son: el plan maestro de producción (Master Production Schedule, MPS), la lista de materiales (Bill Of Materials, BOM) y el inventario. El insumo prioritario es el plan maestro de producción, ya que debemos tomar los requerimientos necesarios en cada etapa de la producción para convertirlos en requerimientos de componentes individuales.

### 2.1 Limitaciones del MRP para tratar incertidumbre

A pesar de las ventajas de los MRP, también tienen limitaciones que restringen su aplicabilidad en aquellas situaciones que involucran 4 incertidumbre, el poder incluirla en el sistema aumentaría el rango de dominios donde puede utilizarse.

### 2.2 Lógica Difusa (LD)

Se utiliza para resolver una gran diversidad de problemas, en especial en aquellos relacionados con la toma de decisiones, siendo su uso bastante generalizado en la actualidad. De acuerdo con [2] "muchas de las decisiones hechas en el mundo real se deben hacer en un ambiente en el cual las metas, las restricciones y las consecuencias de las posibles acciones no son conocidas con precisión".

En [3] se define la LD como "*una rama de la Inteligencia Artificial que se basa en el concepto todo es cuestión de grado, lo cual permite manipular y procesar información en términos inexactos, imprecisos, subjetivos o de difícil especificación*".

De acuerdo con [3] la LD es útil de aplicar en las siguientes situaciones:

- Si no existe un modelo de solución sencillo.
- Cuando es necesaria la experiencia de un experto para el tratamiento de conceptos imprecisos.
- Si desconocemos algunas partes del sistema.
- Si el modelo de la solución implica imprecisión o incertidumbre.

La incorporación de LD en un MRP permite introducir incertidumbre en el sistema, lo cual nos permite ampliar el campo de aplicación del MRP, pero incrementa la complejidad del diseño y del desarrolló, lo que puede incrementar el tiempo necesario para su conclusión, así como el costo del sistema en general.

## 3 Separación de abstracciones relacionadas con el sistema

El uso de LD permite tratar con variables inciertas, pero la implementación de un sistema MRP que contemple LD puede volverse algo complicado y difícil de manejar, por lo que un mejor enfoque es el de separar ambos conceptos para manejarlos de forma independiente, permitiendo abstraer una parte del sistema, simplificando el desarrollo.

El separar la LD del MRP permite que se diseñe e implemente por etapas, facilitando a su vez el mantenimiento y la extensión del sistema, ese tipo de separación permite que el sistema tenga definido un lugar para la introducción de las extensiones de funcionalidad, brindándole la capacidad de evolucionar conforme el sistema vaya requiriendo la incorporación de nuevos elementos.

Para poder desarrollar un sistema que tenga estas características es necesario identificar los requerimientos que deberá solventar el sistema, es decir, aquellas propiedades que son necesarias para un usuario del sistema.

La Programación Orientada a Aspectos (POA) permite la identificación y separación de las abstracciones relevantes para un problema determinado, facilitando con ello su comprensión. Separar las abstracciones relevantes permite que la LD sea considerada en una abstracción diferente a aquella en la que se defina el MRP.

La POA permite encapsular las diversas abstracciones de un problema en el código de un aspecto, en este caso encapsulando la funcionalidad relacionada con la LD en un componente independiente del sistema MRP facilitando así su mantenimiento y el análisis del problema. CaesarJ es un lenguaje de POA [4], que nos permite conjuntar los objetos originales contemplados en el MRP con aquellos componentes de la LD que proveerán la funcionalidad adicional que permitirá la incorporación de variables con valores inciertos en el sistema. CaesarJ permite reutilizar la mayor parte de código posible, está orientado a componentes y busca integrar componentes independientes en una nueva aplicación, sin requerir la modificación del nuevo componente, así como tampoco del existente [5].

## 4 Diseño y Arquitectura del Sistema

Para poder separar la complejidad del sistema como es requerido por este problema en particular, es necesario desarrollar una arquitectura en capas, como la que se muestra a continuación en la figura 1.

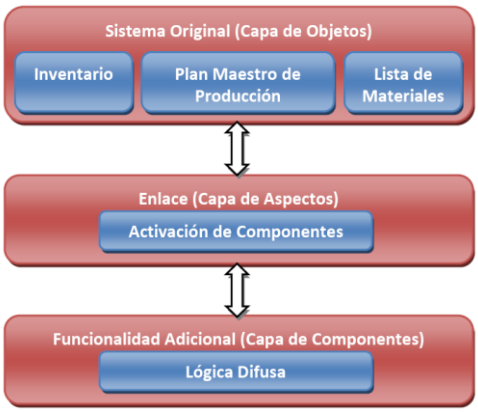


**Fig. 1.** Arquitectura del sistema.

En la capa de objetos se encuentra la funcionalidad del sistema MRP o funcionalidad original que será extendida con la incorporación de la LD, dicha capa contiene 3 módulos, el primer módulo es el Inventario, el cual nos permite llevar un control de los artículos que se encuentran disponibles actualmente.

El segundo módulo es la Lista de Materiales, que posee la información necesaria para determinar que componentes se requieren para poder ensamblar un producto determinado, así como el orden en el que deben tenerse disponibles.

El tercer módulo es el Plan Maestro de Producción, que recibe la información de los productos finales a ensamblar, así como cuales componentes son necesarios y en qué cantidad se requieren. Los tres módulos que se encuentran en la capa de objetos proveen la funcionalidad original del sistema, permitiendo realizar una planeación de los requerimientos materiales para producción.

En la capa de aspectos se encuentran el módulo responsable de la activación de componentes, que tiene como función el detectar los puntos de ejecución en que deberá insertarse la funcionalidad adicional provista por la capa de componentes.

Este módulo determina en tiempo de ejecución en qué momento se activa la funcionalidad adicional, así como el componente adicional que se encargará de proveerla. En la capa de componentes se encuentran los elementos que proveen la funcionalidad adicional al sistema, permitiendo agregar estado y comportamiento a los objetos existentes.

Este tipo de arquitectura nos permite reutilizar la mayor parte posible de código, también evita la necesidad de modificar el existente para añadir nuevas capacidades al sistema, esto se logra abstrayendo la funcionalidad adicional en componentes nuevos, los cuales son activados mediante los aspectos definidos en la capa de enlace.

Este tipo de diseño nos permite evolucionar el sistema conforme los requerimientos van cambiando de acuerdo a las nuevas necesidades de los usuarios, los nuevos requerimientos se encapsulan en componentes adicionales, permitiendo simplificar el desarrollo y brindando mayor claridad al código.

## 5 Implementación del Sistema y la Arquitectura

Para poder probar la arquitectura, se creó un proyecto en el cual se crearon los 3 paquetes propuestos en la arquitectura de referencia, principal que contiene la funcionalidad original, enlace que contiene a los aspectos que enlazarán el funcionamiento original con los nuevos componentes, y la capa de componente que se encuentra conformada por nuevos elementos que son activados de acuerdo al nuevo comportamiento que se espera de la aplicación. A continuación se analiza el código de las diferentes clases que componen la arquitectura.

```
package principal;

public class Inicio {
  public static void main(String[] args) {
    Inventario inventario = new Inventario();
    MPS mps = new MPS();
    BOM bom = new BOM();
```

```
    mps.generarMPS(new int[]{1,2});
  }
}
```

En el código anterior se aprecia el funcionamiento básico de la arquitectura del sistema, en el cual se instancian los principales componentes, Inventario, MPS y BOM, una vez instanciados generamos un MPS para lo cual es necesario enviarle algunos valores, en esta prueba de la arquitectura se le envía un arreglo de enteros, compuesto por 2 enteros 1 y 2.

```
package principal;
public class MPS {
  public void generarMPS(int[] productos) {
    System.out.println("MPS Generado: ");
    for(int i=0 ; i<productos.length ; i++ )
      System.out.println(productos[i]);
    }
    public void eliminarMPS(){
      System.out.println("MPS Eliminado");
    }
    public void consultarMPS(){
    System.out.println("MPS Consultado");
  }
}
```

En la clase MPS que se muestra en el segmento de código anterior, el metodo publico de instancia generarMPS recibe un arreglo de enteros e imprime un mensaje que indica con cuales productos fue generado el MPS, en este caso 1 y 2. Este sería el flujo normal entre los diversos componentes de la arquitectura.

```
package enlace;
import principal.MPS;
import componentes.LD;
public deployed cclass activacionComponentes {
  private int[] nuevosValores;
  pointcut corte();
  execution(void principal.MPS.generarMPS(int[]))&&!cflow
           (within(activacionComponentes));
  void around(): corte() {
    LD ld = new LD();
    MPS mps = new MPS();
    Int[] nuevosValores = ld.ldProcess();
    Mps.generarMPS(nuevosValores);
  }
}
```

Como se aprecia en el segmento anterior de código, se aprecia la clase que contiene el aspecto que modificará la ejecución normal del sistema, esta clase de CaesarJ puede identificarse por la palabra clave cclass. La clase activacionComponentes es la responsable de introducir el nuevo comportamiento. Primero se importan los paquetes principal y componentes para poder tener visibilidad de aquellos elementos sobre los que se actuará. Activamos la clase de CaesarJ agregando la palabra clave deployed a su declaración.

Para poder identificar el punto de ejecución en el que se introducirá el nuevo comportamiento, se utiliza un corte en el punto definido por la ejecución del método generarMPS que recibe un arreglo de enteros y se encuentra ubicado en la clase MPS del paquete principal.

Este corte permite que cada que se invoca al método generarMPS se pueda vincular el comportamiento original con el nuevo. El método generarMPS debe de recibir únicamente los valores contenidos en el arreglo original, pero en lugar de ello se invoca nuevamente el método pero se le envía un arreglo compuesto por los valores provistos por el método ldProcess de la clase LD. Creamos una nueva instancia de la clase MPS y le envíamos el nuevo arreglo de valores, reemplazando los valores originales que debía recibir el método de 1 y 2 por los nuevos valores 1, 2 y 3, modificando la impresión en la consola como se puede apreciar en la figura 2.

```
MPS Generado con los productos:
1
2
3
```

**Fig. 2.** Resultado de la ejecución.

## Conclusión

Si los aspectos se consideran desde el inicio del desarrollo del sistema, es posible definir una arquitectura capaz de separar la dificultad del problema en una funcionalidad principal que cubre las principales funciones del sistema, mientras que el resto pueden ser encapsuladas en componentes que se incorporen en tiempo de ejecución.

Este tipo de arquitectura más flexible, también puede ajustarse para adaptarse a las nuevas necesidades que el sistema debe poder cubrir, sin embargo, como se puede apreciar es importante que desde el inicio del diseño del sistema se contemplen estas características.

La Programación Orientada a Aspectos abre interesantes posibilidades para separar un problema en diversos niveles de abstracción permitiendo simplificar el desarrollo, reducir el tiempo de implementación y tener una mayor claridad en la lectura del código.

En el caso específico de la Planeación de Requerimientos Materiales en la cual se requiere incorporar LD para poder tratar con valores inciertos, los cuáles normalmente no pueden incluirse en un sistema MRP, pero su inclusión puede dificultar el desarrollo de forma considerable.

Tomando en cuenta las ventajas que provee el uso de aspectos, al desarrollar la arquitectura una solución que permite un manejo más adecuado es separar la LD del MRP. Esta decisión permite enfocarse las funcionalidades básicas de un MRP, como son el Inventario, el MPS y la BOM, para posteriormente incorporar la LD, la cual se considera en un nivel de abstracción diferente, facilitando el diseño y el desarrollo de la aplicación.

## Trabajo Futuro

La separación del MRP en una abstracción diferente a la de las funcionalidades adicionales permite manejarlas por separado reduciendo la complejidad del mantenimiento del sistema. Una de las principales limitaciones del MRP consiste en que es incapaz de reflejar el conocimiento de un experto, lo cual se resuelve hasta cierto punto con la lógica difusa, pero podría mejorarse su capacidad si se le incorporase una red neuronal que le permitiese aprender.

## Referencias

[1] Domínguez-Machuca, J. A. (1995). Aspectos tácticos y operativos en la producción y los servicios. Mc Graw Hill.
[2] Bellman, R. y Zadeh, L. (1970). “Decision making in a fuzzy environment”, Management Science vol. 17, no. 4.
[3] Escobar Gómez, Elías Neftalí (2009). Modelo Difuso para Planeación Agregada. Tesis de Doctorado, Centro de Ingeniería y Desarrollo Industrial.
[4] CaesarJ Team (2009). Software Development with CaesarJ. CaesarJ.
[5] Mezini, M. ét al (2005). An Overview of CaesarJ. Darmstadt University of Technology.
[6] Timothy J. Ross. Fuzzy logic with Engineering Aplications. 2nd edition. John Wiley Ed. Sons, 1998.
[7] Chow, Mo Yuen (1997). Methodologies of Using Neural Networks And Fuzzy Logic Technologies for Motor Incipient Fault Detection. World Scientific Publishing Company.
[8] Kosko, Bart (1992). Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence. Prentice Hall.
[9] García Mendoza, Consuelo Varinia (2006). Modelado de Software con Lógica Difusa. Instituto Politécnico Nacional.