# Arquitectura Multiagente para Monitorización Distribuida en Sistemas de Manufaturación Flexible (FMS)

Autores: D Ouelhadj, C. Hanachi y B. Bouzouia

***Resumen***

El artículo presenta una arquitectura de monitorización distribuida inteligente, pensada en Procesos de Fabricación Flexibles (*Flexible Manufacturing Systems FMC)* y basada en el paradigma multiagente.
El desarrollo consiste en modelar el sistema, mediante una sociedad de agentes con conocimiento, encargados de monitorizar una serie de recursos con variados funcionamientos.

Una característica fundamental de los agentes propuestos es tener un conocimiento suficiente para poder tomar decisiones y cooperar con otros agentes del proceso, pensado en el desarrollo de funciones de monitorización del mismo.

La diferencia de esta propuesta es encargar a los agentes de las funciones de monitorización necesarias en el proceso, al igual que la cooperación en tiempo real entre agentes, mediante el *" Contract Net Protocol (CNP)".*

Este modelo de monitorización proporciona flexibilidad (por su capacidad de negociación), posibilidades de ampliación, execución en tiempo real, rápida reacción a fallas y coste de desarrollo.

Se llevaron a cabo experimentos mediante simulaciones, observándose en ellas la factibilidad y potencialidad de la arquitectura multiagente propuesta para monitorización distribuida.

*Introducción*

Las estructuras de producción, necesitan cada vez mas adaptabilidad, debido principalmente al ciclo de vida del producto y a la alta competitividad. Uno de los principales inconvenientes en mejorar estos aspectos, es la existencia de fallas en los procesos, que provocan retardos en la producción y problemas de productividad tales como baja calidad, altos costos y tiempos muertos del producto.

Es por esta razón que las compañias encargadas de la producción, han venido mejorando los procesos, donde se aplican nuevas tecnologías que incluyen capacidades de integración, cooperación, coordinación, robustez, reactividad, flexibilidad, heterogeneidad, autonomía y viablidad, entre las más principales. Pensando en este tipo de procesos complejos, una monitorización centralizada y jerarquica no es la más adecuada, de ahi la necesidad de técnicas inteligentes distribuidas que garanticen la solución de dichos problemas.

Entre las los métodos para desarrollar sitemas inteligentes avanzados flexibles y complejos, el paradigma multiagente uno de los más prometedores.

*Arquitectura Multiagente para Monitorización Distribuida y en Tiempo Real*

Esta arquitectura remplaza un monitoreo centralizado por una red de agentes con conocimiento llamados Agentes de Monitorización de Recursos (RMA). Cada recurso (robots industriales, máquinas de control numérico, herramientas, etc), está bajo la responsabilidad de un RMA, el cual maneja reglas e información del entorno, para llevar a cabo las funciones de monitorización: detección, diagnóstico y manejo de error.

La mayor parte del funcionamiento del sistema no está planeado, sino que emerge mediante la interacción dinámica en tiempo real de los agentes, mediante el uso del CNP. La interface de agente usada por la arquitectura, para recibir las tareas a executar en el proceso, es llamada *Task Manager Agent (TMA).*

Las principales características de esta arquitectura son:
- La monitorización del proceso es distribuida sobre los RMAs
- La monitorización global se hace por el intercambio de mensajes de RMAs.
- El CNP se usa para coordinar y cooperar los RMAs, y para reprogramas las operaciones en falla.

La figura 1 presenta una idea global de la arquitectura propuesta.

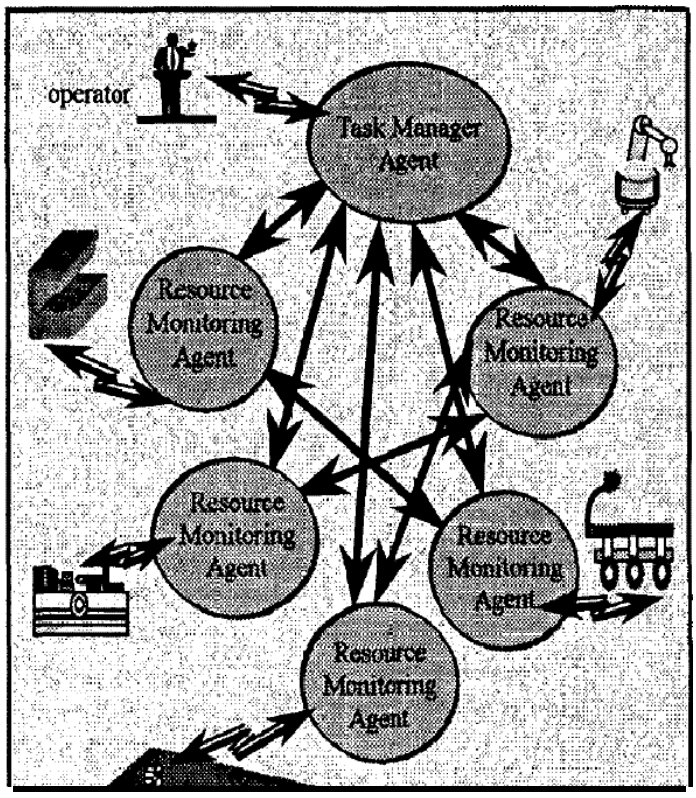


Figura 1. Arquitectura Multiagente para Monitorización Distribuida y en Tiempo Real

*Modelo de Agente de Monitorización de Recursos (RMA)*

El modelo de agente cognitivo tiene la capacidad de distinguir entre: conocimiento, control(razonamiento), cooperación y comunicación.

- *Conocimiento:* Lo requiere el RMA para desarrollar sus actividades externas e internas. Se distinguen dos actividades: de conocimiento seguro y de otros conocimientos. Las de conocimiento seguro son información propia del agente para monitorizar su recurso (base de datos del estado, reglas de producción de diagnóstico, reglas de producción de manejo de error), mientras que las de otros conocimientos, son información acerca de otros RMAs, que pueden cooperar con el.

- *Comunicación.* Es síncrona y se hace mediante mensajes, los cuales son de tres tipos: De requisición de información, de alerta de falla, y de coordinación y cooperación.

- *Control.* Es la capacidad de los RMAs de tomar decisiones, para llevar a cabo funciones de monitorización y cooperar con otros agentes. El control incluye funciones de monitorización tales como detección, diagnóstico, manejo de error y protocolo CNP.

La *detección* se encarga de seguir el estado actual de cada recurso asi como partes próximas a los sensores e interactúa con la base de datos para detectar situaciones anormales que ocurran sobre cada recurso.

El *diagnóstico* identifica y reconoce fallas usando un sistema experto basado en reglas. La base del conocimiento son reglas de producción expresadas con la siguiente logica:

**si** *evento a* **y** *evento b* **entonces** *falla a,.....,falla m*

El *manejo de error* determina las acciones correctivas que manejen la(s) falla(s) previamente detectada(s). El mismo sistema experto basado en reglas usado para diagnóstico, se utiliza para proponer las acciones correctivas. La base de conocimiento consiste de una serie de reglas de producción, dadas de la siguiente manera:

**si** *falla a* **entonces** *acción correctiva a,.....,acción correctiva n*

Ejemplo: **si** *falla máquina de control numérico* **entonces** *transite a modo degradado y llame al operador*

Las acciones correctivas son: reprogramación, parada urgente, transite a modo degradado, llame al operador y omita la acción (salto).

La reprogramación consiste en acomodar localmente operaciones que se pueden executar por el recurso en falla, usando el protocolo CNP.

Cuando un recurso falla, su RMA: cancela peticiones enviando mensajes "*cancel bid* ", Cancela los contratos establecidos sobre acciones enviando mensajes "*cancel contract".* Los RMAs que reciben la cancelación de la petición seleccionan otras peticiones o renegocian las operaciones en falla, con otros RMAs que esten en capacidad de executarlas, mientras que los RMAs que

reciben la cancelación del contrato renegocian las operaciones en falla, con otros RMAs que esten en capacidad de executarlas.

*Architectura de Software y Hardware propuesta*

La arquitectura implementada asigna un computador personal para cada recurso del proceso. Cada agente es un software executándose en el computador aignado al recurso. El lenguaje de implementación del software es Java y las estaciones de trabajo se conectan por TCP/IP a través de red Ethernet.

*Protocolo Multi-Contract Net propuesto*

Está basado en construir dinámica y jerárquicamente la programación mediante el envío de mensajes (requisiciones, peticiones, contratos, admisiones) entre agentes.

Distribuye de la función de programación sobre todos los agentes del sistema multiagente, brindando una arquitectura multi agente modular asi como habilidad para programar diferentes tareas simultáneamente y en tiempo real.

Se provee un agente con alto comportamiento oportunista, que toma ventaja de la mejor oporunidad que se de por la flexibilidad del ambiente de fabricación, además de la habilidad para manejar la programación de conocimiento incierto.

Este protocolo está basado en dos reglas:

*Regla 1.* El agente recurso propone la mejor estimación para la localización de su recurso ante una operación de tarea dada, sin tener en cuenta las situaciones de conflicto.

*Regla 2.* En el caso de una situación de conflicto el agente recurso puede cancelar una o mas de sus decisiones previas, si asi lo requiere. El ciclo de programación de una tarea se descompone en cuatro etapas: propagación de requisición, realimentación de petición, propagación de contrato y realimentación de admisión.

Estos cuatro pasos se pueden describir mediante un ejemplo:
*Paso 1: Propagación de Requisición.* El agente de recurso coordinador, extrae la última operación de la tarea y construye, usando conocimiento sobre la configuración del material, la lista de posibles posibles recursos útiles para la operación. Cada agente recurso candidato, enviará el mensaje de requisición en la operación (N-1) correspondiente a la subjerarquía de operaciones. La propagación de requisiciones, es repetida hasta que no hay operación por negociar.

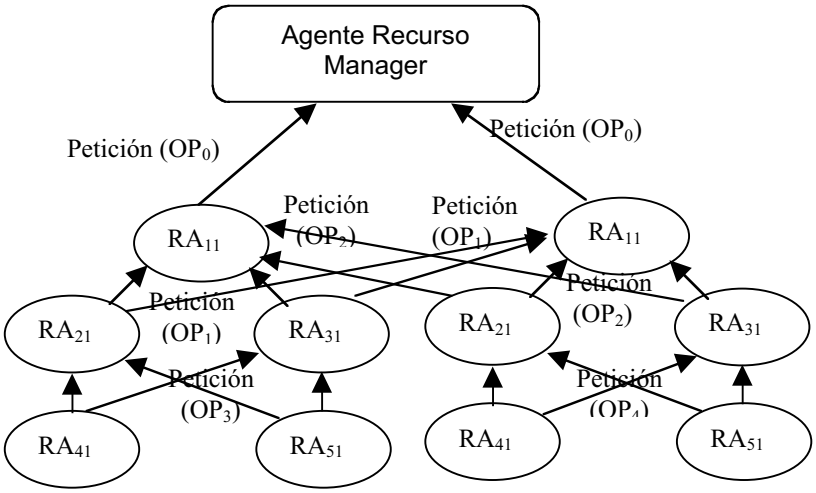


Figura 1. Propagación de requisición

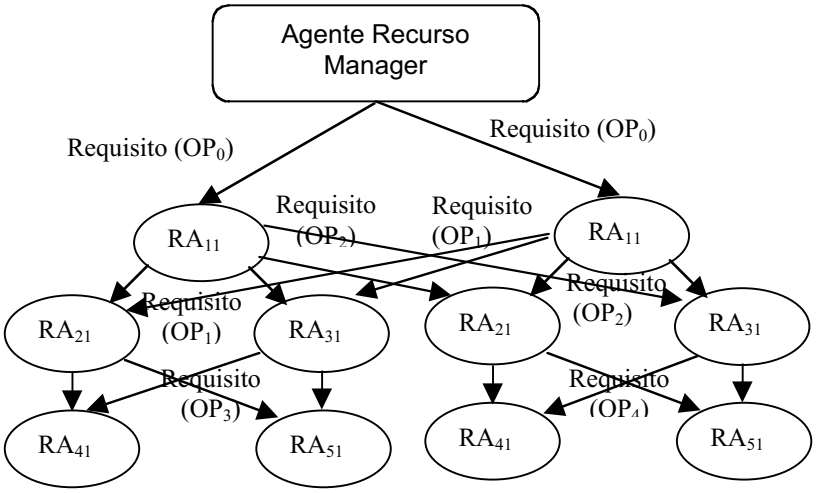


Figura 2. Realimentación de petición

*Paso 2. Realimentación de petición.* Los agentes recurso que reciben el mensage de requisición sobre operaciones elementales, son los primeros que responden con el mensaje de petición del recurso, basado en las dos reglas expuestas anteriormente. El mensaje de petición del recurso tiene el siguiente formato:

(senderagent, idftask, operation, startdate)

Donde senderagent es el agente que envía la petición, idftask es el descriptor de tareas, operation es la operación misma y stardate es el tiempo en el cual la operación debe iniciar. Los agentes recurso que reciben mensages de requisición en una jeraquía de operaciones, esperan asíncronamente hasta que todos los agentes de recurso de la subjerarquía envían sus mensajes de petición.

*Paso 3. Propagación de contratos.* Cuando el agente de recurso manager, recibe todos los mensajes de petición de la anterior operación, el selecciona el agente de recurso que propone la ejecución rápida “star date”. Si el recurso no está en la capacidad de garantizar la linea muerta de una tarea, la negociación se termina sin contrato alguno, siendo enviado un mensaje que indica la falla de la programación. El agente de recurso seleccionado, prosigue estableciendo los contratos a la subjerarquía. La representación esquemática se muestra en la figura 3.

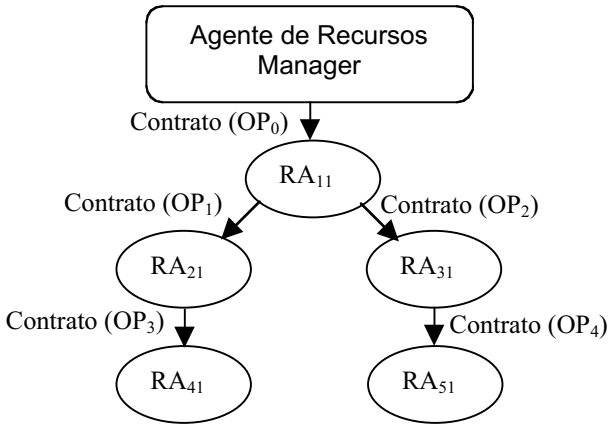


**Figura 3. Propagación de contratos**

El formato del contrato es de la forma: (senderagent, idftask, operation, contractstartdate), donde es el tiempo de inicio de un contrato.

*Paso 4. Realimentación de admisión.* La confirmación de la ejecución de la operación contratada, se hace mediandte un mensaje de admisión o aceptación, con el siguiente formato: (idftask, operation). Además el agente recurso, procede a la cancelación de peticiones relacionadas con la respectiva operación confirmada. Los agentes recurso quienes reciben los contratos sobre la operación, son los primeros agentes en enviar sus acepataciones. Los agentes recurso que reciben el contrato en la jerarquía superior de operaciones, esperan asíncronamente hasta que todos los agentes recurso seleccionados en esta jerarquía, envíen sus acepataciones. El agente recurso manager, al recibir el mensaje de aceptación de la última operación determina el final del proceso de programación de la tarea recibida.

*Conclusión*

Las principales ventajas de la arquitectura propuesta son flexibilidad en un ambiente variable, reconfiguración y adaptibilidad inherente, tolerancia a fallos rápida y eficiente, desarrollo modular y autonomía operacional.

La implementación de la arquitectura multiagente fue llevada a cabo en Java por ser orientada a objetos, distribuida, segura, dinámica y programación de alto desempeño. El *Java Expert System Shell* fue integrado en el software, para el desarrollo del sistema experto basado en reglas, usado en el diagnóstico y manejo de error.

Los resultados experimentales obtenidos en la simulación, demostraron que la arquitectura multiagente is una plataforma de trabajo bastante apropiada para monitorización distribuida y en tiempo real de FMS.