UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES
DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN

Tesis

Presentada en conformidad con los requisitos para la obtención del título de
Ingeniero Civil en Informática

"Control Distribuido Auto-organizado para un Sistema Flexible de Manufactura"

Por
Cristian R. Durán Faúndez

Profesores Guía:

PEDRO RODRÍGUEZ MORENO	Profesor del Departamento de Sistemas de Información, Facultad de Ciencias Empresariales.
DR. MARIO RAMOS MALDONADO	Profesor del Departamento de Ingeniería en Maderas, Facultad de Ingeniería

Concepción, 16 de marzo de 2005

Dedicatoria

Estando en una etapa de transición en mi vida, al final de este trabajo miro con orgullo y a la vez con nostalgia lo que he logrado. Miro hacia atrás y veo tantos momentos alegres y tristes que me ha puesto al camino el destino. Amigos han llegado y se han ido. Personas que alguna vez consideré importantes para mí han quedado sólo en recuerdos, por esos azares del tiempo y la distancia. Otras, se proyectan en mi futuro como herencia de años de episidios que hemos vivido. Quizás que pasará…

…Pero al mirar hacia el pasado, y al mirar hacia el futuro, siempre está la imagen de dos personas que en silencio se han convertido en el mayor soporte que he tenido hasta ahora. Quienes con mirada alegre me han visto avanzar hacia un futuro que en conjunto hemos forjado.
Este trabajo se lo dedico a dos de las personas más importantes para mí: Gertrudiz Irene Faúndez Riquelme y Juan Carlos Durán Rojas, mis padres, quienes a lo largo de estos 25 años me han entregado todo su amor, afecto, apoyo y enseñanzas de vida que nunca olvidaré, y que día a día moldean el hombre, profesional, padre y esposo en el que me convertiré…

…A mi madre por todo su afecto, dedicación, paciencia y comprensión.
A mi padre por todo lo que me ha enseñado. Él es, sin duda, un modelo de vida para mí.
Todo lo que he recibido es gracias a ellos.
Espero en un futuro, poder hacer tan buen trabajo como ellos.
Les estoy infinitamente agradecido. Espero poder pagar algún día todo lo que me han dado...

…Aunque sé que nunca lo haré.

Agradecimientos

Sin duda que los logros de una persona no son mérito exclusivo de sí. En gran parte se deben a ese conjunto de personas que está ahí cuando lo necesitas. A todos ellos, a los que aparecen en esta página, y a los que no por motivos de espacio (podría escribir una biblia con las personas que han sido parte de esto), muchísimas gracias.

En primer lugar agradecer a mis padres, a quienes dedico este trabajo, por todo lo que ya dije y mucho más.

Al profesor Cristhian Aguilera, quién en muchos aspectos se ha convertido en un modelo profesional a seguir y un guía en varias situaciones, incluyendo el desarrollo de este seminario de título. Así mismo agradecer al profesor Mario Ramos, quién jugó un papel clave en este proyecto como profesor guía, y con quién más de una diferencia de opinión hemos tenido, pero por quien tengo gran respeto y admiración. A estos dos personajes les agradezco haberme dado su confianza para trabajar en conjunto, y proyectar, ojalá, trabajo en conjunto en los años siguientes. De ellos he aprendido mucho.

A mi profesor guía Sr. Pedro Rodríguez, quién a lo largo de la carrera me entregó conocimientos que hoy en día me fueron valiosísimos en la resolución de este trabajo.

A Luis Vera, encargado del Laboratorio CIMUBB, quien hoy en día considero mi amigo. Por toda su ayuda, por su entusiasmo y, sobre todo, por siempre estar ahí con una sonrisa acogedora.

Por último, agradezco a quienes jugaron un papel clave en mi carrera y en mi vida. Gran parte de lo que ahora soy se lo debo a ellos: mis amigos. Y que difícil es decir que uno tiene amigos. Yo sí puedo decirlo. Sé que sin ellos no hubiese podido lograrlo. Podría escribir un libro completo con todos los recuerdos que tengo con ellos, sólo me queda darles las gracias. Lilian Seguel, Karen Araneda, Miguel Vidal, Christian Segura y, muy en especial, a Cristina Riquelme, quien fuera mi compañera durante gran parte de la carrera y quién me obligaba a estudiar y trabajar esas noches en las que sólo quería recostarme y cerrar los ojos, incluso hasta hace pocos minutos de escritas estas palabras.

A todos ellos, y otros tantos que recuerdo ahora… muchas gracias.

"Pueblo que se resigna a tecnologías pasadas
sucumbe en el campo de la ignorancia
y se entierra en sus ideales."

Wily Fiallos

Resumen

En la actualidad, la mayoría de los sistemas productivos automatizados o semi-automatizados están diseñados para trabajar de una forma poco flexible o centralizada. Esto trae consigo varias consecuencias; la dependencia del sistema completo del dispositivo de control central, la dependencia del sistema al estado de los diversos dispositivos (los sistemas automatizados en cadena en su gran mayoría se detienen si falla uno de los componentes intermedios), la pobre o inexistente reacción ante una posible falla en los dispositivos o en los canales de comunicación o ante un accidente fortuito en cualquier proceso (atoramiento, caída de materiales, etc.): en definitiva, la poca "inteligencia" del sistema, al tener que mantenerse operativo en un ambiente de trabajo, en el que las fallas o alteraciones de cualquier tipo atentan con el rendimiento de la productividad.

Por estos motivos se hace necesario el modelamiento, implementación y testeo de un sistema de producción automatizado distribuido, en el que existan individuos virtuales (agentes) capaces de interactuar entre sí, totalmente independientes, proactivos, y con la capacidad de tomar determinadas acciones ante eventualidades que pudieren suscitarse.

Al respecto son numerosas las aproximaciones que proponen los autores. Los sistemas biónicos, fractales y holónicos, entre otros, son motivo de muchos estudios y sus aplicaciones abarcan una infinidad de temas. Pero los que mayor revuelo han causado en la comunidad científica son los sistemas multi-agente, el mayor logro de la Inteligencia Artificial Distribuida.

Estos sistemas son capaces de combinar los esfuerzos de un conjunto de agentes para el logro de un determinado objetivo. La característica principal de este tipo de sistemas es su carácter auto-organizado emergente, es decir, que el funcionamiento global del sistema nacerá de la interacción de los individuos. A esto se le llama enfoque Bottom-Up.

En el presente trabajo se desarrollará, pues, un sistema multi-agente para solucionar la problemática de la programación de la producción en tiempo real, y a su vez, tratar de capear algunas de las deficiencias existentes en los sistemas automatizados jerárquicos centralizados.

Para el desarrollo del proyecto se debe utilizar una metodología orientada a los sistemas multi-agente. Muchos autores se precipitan a utilizar metodologías basadas en la orientación a objetos, por su conocimiento, disponibilidad de herramientas y por los beneficios que ésta trae. Sin embargo, las metodologías orientadas a objetos no satisfacen completamente los requerimientos de determinados sistemas, tal es el caso de los sistemas multi-agentes. Si bien un objeto comparte ciertos atributos con un agente, no son lo mismo.

Se habla entonces de Orientación a Agentes, y se han propuesto muchas metodologías al respecto.

Una de estas metodologías es la extensión en GAIA y AUML, la que se adoptó en este trabajo por convinar una metodología netamente orientada a agentes con los beneficios de la orientaión a objetos.

Durante el desarrollo del trabajo se llega al modelamiento de un sistema de producción automatizado que puede realizar el control de sus diversos componentes en forma distribuida, es decir, independientes a una unidad central (Manager) que organice y distribuya las tareas entre las células subordinadas. En particular se debía concebir un modelo del sistema del control distribuido, que pudiera ser implementado en un ambiente productivo real, dar autonomía a las distintas estaciones que controlan un sistema de producción automatizado, eliminando la dependencia de un control central, que tiene la gran mayoría de los sistemas, y disminuir la tasa de errores producidos por problemas en determinados periféricos, anticipando ciertas situaciones que podrían conducir a la falla total del sistema, entre otros objetivos. Era de particular interés la implementación de la propuesta en el Laboratorio de Sistemas Automatizados de Producción CIMUBB, de la Universidad del Bío-Bío, el que cuenta con un Sistema Flexible de Manufactura, con lo que se podría probar el modelo diseñado.

Indice

Nomenclatura

CIM	:	Computer Integrated Manufacturing (manufactura integrada por computador)
CN	:	Control Numérico
CNC	:	Computer Numerical Control (Control Numérico Computacional).
FMS	:	Flexible Manufacturing System (sistema flexible de manufactura)
IAD	:	Inteligencia Artificial Distribuida
MAS	:	Multi Agent System (sistema multi.-agente)
O.A.	:	Orientado a Agentes
O.O.	:	Orientado a Objetos
RPD	:	Resolución de Problemas Distribuidos
SMA	:	Sistema Multi-Agente
UML	:	Unified Modeling Language (lenguaje unificado de modelamiento)

Introducción

En las últimas décadas la industria de la manufactura ha sufrido cambios vertiginosos. Cambios en cuanto a la velocidad de entrega de los pedidos, diversidad y calidad de los productos, entre otros. Los clientes son cada vez más exigentes y sus gustos más variados; el ciclo de vida de los productos se reduce considerablemente.

Ante esta diversidad, y nuevos requerimientos, las empresas manufactureras se ven obligadas a adoptar medidas enfocadas a acrecentar la productividad, disminuir los costos y, recientemente, a flexibilizar sus flujos de producción. En un mundo competitivo globalizado, como el que prevalece actualmente, las industrias se enfrentan en una contienda en que prevalece la ley del más fuerte; los que son capaces de adaptarse a este cambio abrupto son, por lo general, los que están dispuestos a invertir gran cantidad de recursos y esfuerzo en la reestructuración organizacional y modernización de sus instalaciones. Los otros estarán destinados a extinguirse. En términos de manufactura, esta agilidad se traduce en un conjunto de soluciones, una de las cuales es la denominada automatización.

La automatización es una tecnología dinámica que representa un proceso evolutivo continuo que comenzó muchas décadas atrás (Groover, 1980). En efecto, el ser humano a través de la historia se ha preocupado del desarrollo de técnicas y tecnologías que puedan acelerar los ciclos de producción para disminuir el esfuerzo, minimizar al máximo los costos, satisfacer un mercado en crecimiento, y ganar posicionamiento con respecto a sus pares. Tal vez el crecimiento más significativo, en términos de desarrollo de las tecnologías de manufactura, se ha dado sólo en los últimos tres siglos (CIM and Engineering, 1993; Groover, 1980), sin embargo el desarrollo de la manufactura ha jugado un papel importante durante la evolución de la especie humana. La primera revolución

industrial[1] trajo consigo la incorporación de nuevos conceptos en términos tecnológicos, socioeconómicos y culturales, en un período que circunda el año 1770, aunque ya se visualizaba un acercamiento a los actuales sistemas flexibles de manufactura con trabajos de máquinas para los años 60 (Groover, 1980).

Los sistemas flexibles de manufactura (SFM) se presentan como una solución (hasta cierto punto) a los problemas presentes en el mundo moderno en muchos sectores industriales, existiendo hoy en día variadas configuraciones arquitectónicas instaladas, abarcando las más diversas aplicaciones en la fabricación de productos. Uno de los grandes retos de la fábrica moderna (Rehg, 1994) es el de disminuir el ciclo de vida de los productos; como una forma de enfrentar la filosofía de algunas empresas japonesas de considerar la inversión inicial de un producto como un costo despreciable[2]. Los sistemas flexibles de manufactura ofrecen un compromiso entre productividad y flexibilidad (Ramos, 1998). Con esto, es necesario contar con sistemas rápidamente configurables a la hora de modificar un producto o incorporar uno nuevo.

Un sistema flexible de manufactura consiste en un grupo de estaciones de procesamiento (predominantemente herramientas de máquinas CNC[3]), interconectadas por medio de un sistema de manipulación y almacenamiento de material y controlado por un sistema computacional integrado (Groover, 1980)[4].

Originalmente, este sistema computacional integrado respondía únicamente a una estructura organizativa jerárquica (actualmente ésta es la visión que impera en el mercado comercial de sistemas automatizados de fabricación), donde un administrador central se encarga de las tareas de secuenciamiento globales y envía órdenes a los equipos de trabajo subordinados, quienes controlan procesos más particulares dentro de la cadena de

[1] Otros hitos históricos importantes fueron la creación del Control Numérico (CN) en 1947 en el MIT (CIM and Engineering, 1993), la primera computadora electrónica en 1946 y la creación del primer robot de carácter industrial en 1960 (TrueForce, 2004), entre otros. Estas tecnologías vinieron a cambiar completamente la forma de realizar los procesos de manufactura, y son, actualmente, la base de todo sistema automatizado de producción.
[2] En [Rehg, 1994] lo llaman literalmente *sunk cost*: costo hundido.
[3] CNC: Computer Numerical Control (Control Numérico Computacional).
[4] Otras definiciones de SFM en [Ramos, 1998], [Rehg, 1994] y [CIM & Engineering, 1993].

fabricación. Esta estructuración jerárquica puede repetirse recursivamente una cantidad de veces hasta llegar, desde el nivel de control central, al nivel de máquina[5]. Sin embrago, esta visión trae consigo un conjunto de problemas.

En efecto, la planificación, programación y los mecanismos de control tradicionalmente centralizados están siendo insuficientemente flexibles para responder a los cambiantes estilos de producción y variaciones altamente dinámicas en los requerimientos de los productos. Las organizaciones jerárquicas fuerzan la agrupación de los recursos de manufactura en grupos permanentemente integrados y dependientes, donde la información es procesada en forma secuencial por un software de monitoreo centralizado (Maturana *et al*, 1999). Esta centralización es el principal problema de las arquitecturas actualmente dominantes, ya que el sistema depende completamente de una sola estación de trabajo. Esto incrementa el peligro de una detención completa de los procesos ante cualquier falla en el controlador central o en la comunicación con este. Es más, frecuentemente los sistemas de este tipo dependen completamente del buen funcionamiento de cada uno de los equipos existentes, cualquier falla en las comunicaciones o en la respuesta de una unidad provoca un fallo global; generalmente la planificación es frágil debido a su naturaleza básicamente secuencial.

Ante esto, algunos investigadores, conscientes de esta problemática, comenzaron hace algunas décadas a realizar investigaciones en el campo de la inteligencia artificial distribuida (IAD), para encontrar aplicaciones en el campo de la manufactura. Es así como numerosas propuestas se han realizado hasta ahora, propuestas destinadas a aplicar los paradigmas de la IAD (como los sistemas biónicos, fractales, holónicos o multi-agentes[6]) para eliminar la centralización del control y concebir un modelo en el que las distintas unidades cooperen entre sí en la y conformen una heterarquía, apoyados por la visión de

[5] Esta definición representa, en gran medida, la concepción tradicional de un SFM basado en el paradigma CIM (Computer Integrated Manufacturing). CIM es un concepto que pretende realizar un pacto con la solución de los problemas generales de flexibilidad e integración de un sistema manufacturero. Sin embargo la concepción tradicional de este sistema trae consigo nuevos riesgos y problemas ([Tharumarajah *et al*, 1996], [Choi *et al*,. 2003]).

[6] Se dará una explicación más detallada de estos conceptos en el Capítulo 1 de este documento.

que un sistema manufacturero es, en esencia, un problema de características claramente distribuido.

El presente trabajo pretende realizar una aproximación a un modelo de control distribuido heterárquico, principalmente en el área de los sistemas multi-agente, aplicando una de las metodologías propuestas por los autores que estudian el tema (Iglesias, 1998; Gómez, 2002; Gómez, 2003), tomando como referencia algunas propuestas y experiencias que proponen investigadores de distintas partes del mundo (Parunak, 1988; Ramos, 1998; Wang *et al*,. 1998; Kim y Nof, 2000; Langer y Alting, 2000; Suessmann *et al*,. 2002; Shahin, 2002; Charpentier y Muhl, 2004, entre otros).

Como objetivo final, se pretende estudiar la factibilidad de validar el modelo en las instalaciones del Laboratorio de Sistemas Automatizados de Producción CIMUBB[7], unidad que funciona bajo el alero de la Facultad de Ingeniería de la Universidad del Bío-Bío[8]. Este laboratorio está equipado con modernos dispositivos orientados a la educación; tres células de manufactura compuestas por máquinas CN, brazos robóticos y otros dispositivos. Al estar diseñado bajo el enfoque CIM tradicional, el sistema actualmente presenta los problemas clásicos de este tipo de SFM.

Los sistemas auto-organizados han visto aplicaciones en una infinidad de áreas y sus enfoques practicados en variados aspectos. En particular el trabajo que aquí comienza tiene por objeto el modelamiento de un sistema auto-organizado para el control de un sistema flexible de manufactura en un nivel de floor-shop (nivel de piso o taller).

En el Capítulo 1: La Problemática de la Programación de la Producción en la Industria Automatizada, se explica el estado del arte de la industria fabril moderna y los principales problemas que se deben solucionar. La problemática de los sistemas jerarquizados de control y la problemática general de la programación de la producción. Con esto se exponen explícitamente los factores que debiera solucionar el presente proyecto.

---

[7] Este laboratorio ha sido motivo de gran cantidad de estudios y aplicaciones. Se puede saber algo más de éste en el sitio http://www.cimubb.ubiobio.cl.

[8] En [Morales, 2003] existe un estudio de la arquitectura de software y hardware existente en el laboratorio CIMUBB.

En el Capítulo 2: Sistemas Auto-Organizados de Producción, se describen las distintas arquitecturas de configuración de un sistema automatizado (jerárquicas, oligárquicas, heterárquicas, etc.), además de realizar una breve introducción a la Inteligencia Artificial Distribuida, poniendo especial interés en los sistemas multi-agente.

En el Capítulo 3: Metodología para el Desarrollo del proyecto, se expone un listado de las metodologías de desarrollo de sistemas multi-agente más completas que existen hasta la fecha, y las que mejor han solucionado los problemas de los investigadores. Por último, se procede a la selección y descripción de la metodología a utilizar en el resto del trabajo.

El Capítulo 4: Especificación de Requerimientos, Caso de Estudio CIMUBB, formaliza los requisitos básicos con que debe cumplir la propuesta a implementar, así como un estudio superficial del sistema de manufactura en el que se pretende validar la propuesta, el Laboratorio de Sistemas Automatizados de Producción CIMUBB.

El Capítulo 5: Análisis del Sistema, Identificación de Roles e Interacciones, muestra los primeros modelos de la metodología seleccionada, GAIA, para la definición de las operaciones básicas que deben ser realizadas una vez realizado el sistema.

El Capítulo 6: Diseño del Sistema, Hacia la Identificación y Especificación de los Agentes, pretende ilustrar el proceso de caracterización de los principales agentes involucrados en un sistema manufacturero (y particularmente en el CIMUBB), así como la descripción cabal de sus operaciones y características. Aquí se combinan GAIA y una descripción más acabada con AUML (una ampliación del lenguaje unificado de modelamiento para los sistemas con agentes).

Finalmente, el Capítulo 7: Validación de la Propuesta, Construcción, Implementación y Análisis de Resultados, expone el proceso de construcción del sistema auto-organizado con el que se pretende validar los modelos presentados en el presente documento. Así como la exhibición de los resultados finales.

Un trabajo arduo, con un tema que en la actualidad está en boca de un conjunto considerable de investigadores alrededor del mundo.

Capítulo 1

La Problemática de la Programación de la Producción en la Industria Automatizada

---

Los Sistemas Flexibles de Manufactura (SFM) se proyectaron como la solución a muchos de los problemas existentes en la industria productiva del tercer milenio.

El cambio radical sufrido en el mercado en las últimas décadas, provocado por desajustes en la economía, aumento de la competencia, la exponencial evolución tecnológica y, por sobre todo, cambios en las exigencias de los clientes, ha provocado que las empresas deban replantear sus estrategias y reestructurar radicalmente la organización formal de la institución. Un SFM que opere en condiciones óptimas podría garantizar una rápida respuesta a varios de estos inconvenientes.

Y es que los SFM permiten la correcta combinación de dos conceptos fundamentales: la productividad y la flexibilidad.

En efecto, los SFM incrementan considerablemente la variabilidad de los productos que se puede obtener con ellos, obteniéndose un buen margen de productividad[9] (aunque, obviamente, no tan vasta como el de una línea de transferencia, pero sí lo suficientemente significativa como para satisfacer el mercado para el cual está orientado).

Sin embargo, y pese a todos los beneficios que los SFM han traído a las empresas, no están libres de problemas. Más aún, cuando hablamos de la integración informatizada de los procesos productivos de la empresa (CIM: Computer Integrated Manufacturing),

[9] Piezas o productos, que pueden ser mecanizados, por unidades de tiempo.

concepto que ha tenido gran revuelo en las últimas década, los problemas aumentan exponencialmente.

Los SFM aportan incontables ventajas para la industria moderna, pero la gestión de estos sistemas no es sencilla, las tendencias de mercado actual no se adecuan bien a los requisitos de las empresas y los riesgos implicados son muchos.

En este primer capítulo se pretende hacer una introducción a la problemática que se desea atacar con el presente estudio; los problemas actualmente existentes en las industrias modernas que cuentan con este tipo de tecnología para el desarrollo de sus productos.

1.1 Automatización de la Producción. Antecedentes.

Antes de comenzar a abordar el problema parece propicio realizar una breve introducción al área en la que se desenvolverá el presente documento. Se realizará un primer acercamiento a la problemática que justifica la incorporación de tecnologías automatizadas de producción y cuáles son los beneficios que éstas traen, y la descripción de los distintos enfoques de manufactura moderna, haciendo especial hincapié en la definición de SFM y CIM, términos de los cuales ya se ha hablado en esta primera parte del texto. Esto último tiene particular importancia, ya que es la base de todo lo que recién comienza.

1.1.1 La industria del tercer milenio. Los nuevos desafíos.

Los cambios en el mundo transcurridos en las últimas décadas auguran la necesidad de incorporar en la industria del presente tecnologías capaces de adaptarse rápidamente a las imposiciones del mercado. La habilidad para realizar los procesos de manufactura en forma eficiente y oportuna determina el éxito de una organización (Rehg, 1994). El cliente se convierte en el principal foco de interés.

Según Rehg (1994) los desafíos que las empresas modernas deben enfrentar en la actualidad se pueden clasificar en dos grandes grupos: los *desafíos externos* y los *desafíos internos*.

Los *desafíos externos* son todos aquellos que, presentes en el medio en el cual se desenvuelve la empresa, representan fuerzas o condiciones que pueden generar cierto grado de impacto para la organización, ya sea en forma de oportunidad o de amenaza. Así, una empresa debe asumir el desafío de mantenerse con vida enfrentando sus posibles amenazas y tomar las oportunidades que se presentan, y que podrían ser benéficas para sus fines. Rehg (1994) identifica seis entidades que significan desafíos para la empresa moderna: *la economía globalizada*, *los costos*, *la competencia tradicional*, *nuevos competidores de nicho de mercado*, *los proveedores* y *los clientes*, siendo este último el factor más complicado de enfrentar.

Por otra parte, los desafíos internos son aquellos sobre los cuales la empresa tiene control directo y que se traducen principalmente en la definición de la estrategia a seguir. Para Rehg (1994) estos desafíos son: *la definición de los objetivos corporativos*, *el desarrollo de las estrategias de marketing necesarias para satisfacer los objetivos corporativos*, *análisis de mercado y determinar cómo el producto se ajustará a las condiciones del mercado y a la competencia*, *determinar el proceso a ser utilizado para fabricar los productos* y *proveer la infraestructura de manufactura necesaria para la producción*.

Richard B. Chase *"et al",* en su texto Administración de producción y operaciones: manufactura y servicios (2000), expresa que las empresas formulan las políticas amplias y los planes para utilizar sus recursos según las siguientes prioridades básicas identificadas:

i. *Costos*. La empresa debe ser capaz de ofrecer un precio por su producto que sea óptimo, es decir, que sea capaz de atraer al cliente (o por lo menos no provocar el rechazo) y que sea idóneo para la misma firma.

ii. *Calidad y confiabilidad del producto*. Aquí se puede identificar la *calidad del producto* y *la calidad del proceso*.

    i. La *calidad del producto* es la facultad que posee un determinado producto de satisfacer las necesidades del cliente. Las empresas deben concebir productos con un nivel de calidad óptimo; equilibrio entre una satisfacción del cliente suficiente y un costo aceptable por el mercado (no se consigue nada con fabricar un producto de una calidad tal que no pueda ser adquirido), y

    ii. la *calidad del proceso*, que está directamente correlacionada con lo anterior, y tiene que ver con la forma en la que se elabora que producto.

- *Velocidad de entrega*. La capacidad para entregar sus productos más rápido que sus competidores puede llegar a ser un factor importante para algunas organizaciones.

- *Confiabilidad en la entrega*. Es la capacidad de una empresa para realizar las entregas correspondientes en la fecha determinada.

- *Afrontar los cambios en la demanda*. Según Chase *"et al"*, corresponde a la *capacidad de afrontar efectivamente la demanda dinámica del mercado a largo plazo*, lo cual es un elemento esencial en la *estrategia de operaciones* (Starr, 1989).

- *Flexibilidad y velocidad de introducción de nuevos productos*. Este requerimiento es clave, y se podría decir que es consecuencia directa de todo lo antes mencionado. La flexibilidad tiene que ver con la capacidad de que una compañía sea capaz de ofrecer una gran variedad de productos, que sea capaz de adaptar sus procesos para incorporar rápidamente un nuevo producto y que atienda rápidamente a las necesidades de los clientes, siendo capaces, muchas veces, de generar productos a la medida del cliente.

- Existen otras prioridades que se refieren a productos o situaciones más específicas y no pueden ser clasificables con facilidad, como enlaces y soporte técnico, cumplimiento de fechas de lanzamiento, etc.

Volviendo al tema de la *flexibilidad*, la necesidad de cambiar rápidamente los procesos de producción nacen principalmente a partir de dos factores principales: los clientes y la competencia.

Los clientes, por una parte, exigen mayor variedad de productos, y esta necesidad se vuelve cada vez mayor. Tendrán deseos más individualistas y se verán más y más involucrados en el diseño de los productos. Esto traerá consigo un enorme impacto en los procesos de diseño y manufactura en la industria del futuro (Tharumarajah *et al*, 1996).

Y donde exista necesidades que cubrir, habrá empresas dispuestas a satisfacer estas demandas (incluso a crear otras nuevas). En este mundo globalizado las grandes potencias ya tienen su lugar y los gigantes de la industria luchan día a día por la supremacía.

El gran golpe para las empresas occidentales lo dan algunas empresas japonesas. Muchas de las grandes compañías americanas intentaban sacar el mayor provecho para cada producto que lanzaran al mercado exprimiéndolo al máximo; tratando de capturar el máximo retorno para los gatos de investigación, propios de su concepción. Por el contrario, la filosofía oriental emergente elimina de la mente del inversionista el concepto de recuperar lo que los americanos llaman "inversión", a cambio de difundir la idea de que los "gastos" de investigación y desarrollo de un producto no son más que costos despreciables, y que no se debe esperar el retorno de estos (Rehg, 1994). Esta idea genera la consecuente disminución del ciclo de vida de los productos, lo que se ilustra en la Figura 1.1.

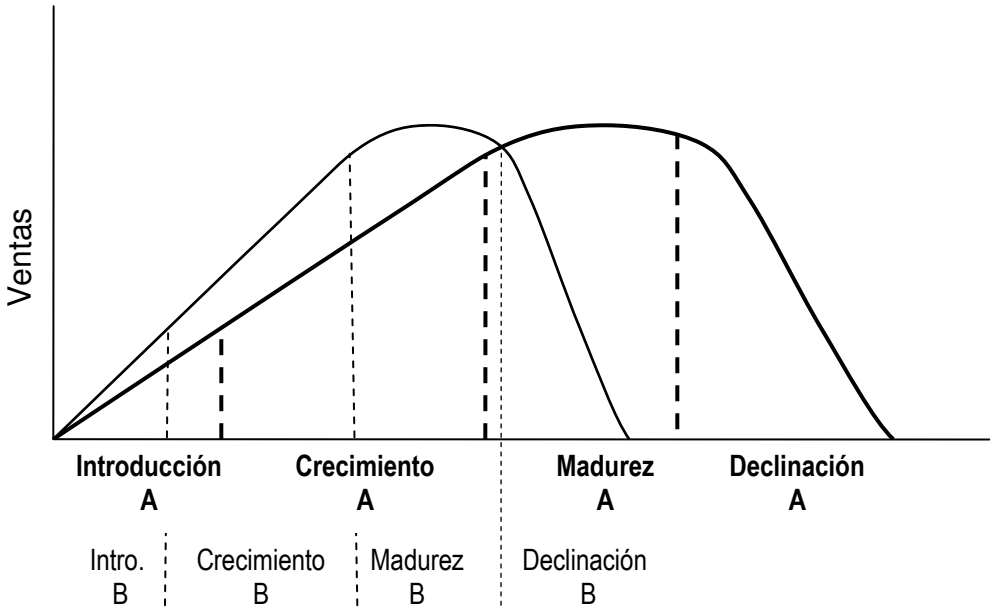


Figura 1.1. Cambio en el ciclo de vida de los productos.

En la figura se exponen las principales etapas del ciclo de vida de un producto: Introducción, crecimiento, madurez y declinación. La curva ancha corresponde a los

tradicionales ciclos de vida americanos, con largos períodos de recuperación de la inversión inicial, en tanto la curva delgada corresponde a los ciclos de vida orientales, mucho más estrechos. Las etapas, expuestas en el eje inferior del gráfico han sido etiquetadas con las letras A y B para representar el ciclo americano y el oriental, respectivamente.

Esto conlleva al actual dinamismo que tiene el mercado actual. Día a día van apareciendo nuevos y mejores productos, acrecentando las tasas de obsolescencia exponencialmente. Con este ritmo tan acelerado las empresas que deseen mantenerse en competencia deben adoptar medidas drásticas.

Para muchas de las empresas actuales, esta problemática se traduce en una solución: Automatización.

1.1.2 Automatización de la Producción.

Hasta el momento se ha utilizado desmesuradamente el término *Automatización*, sin que quizás sea obvio el significado de esta palabra. Se entiende por *Automatización* a una *tecnología referida al uso de sistemas mecánicos, electrónicos y computarizados para operar y controlar la producción* (Groover, 1980).

Groover (1980), indica que existen básicamente 3 tipos de automatización:

i. *Automatización rígida*, que se caracteriza por ser una cadena de procesos autónomos estática para realizar un determinado producto. Este tipo de automatización es el más sencillo de implementar, ya que existe un secuenciamiento de operaciones predefinido y, por lo general, se acota a determinados tipos de productos (muchas veces de un solo tipo). Los sistemas rígidos tienen, por lo general, una muy alta tasa de productividad, ya que los procesos son generalmente en una secuencia única y las decisiones requeridas son mínimas. Sin embargo, la rigidez implica una gran dificultad a la hora de modificar la cadena al tratar de incorporar una nueva máquina, modificar el diseño de un producto o

querer fabricar uno nuevo. Este es el caso típico de las *líneas de transferencia* y de los sistemas de producción continuos.

ii. *Automatización programable* es la otra cara de la moneda. Son los sistemas dotados de equipos altamente configurables debido a su propiedad para cambiar la secuencia o las características de sus operaciones en forma automática, gracias a que responden a instrucciones definidas codificadas. Tal es el caso de las máquinas de control numérico (CN). Estos sistemas están diseñados para permitir un cambio muy rápido en el diseño de los productos y en el secuenciamiento de las instrucciones que deben realizar, sin embargo, la tasa de productividad disminuye considerablemente en comparación a los sistemas rígidos.

iii. Finalmente, *automatización flexible* corresponde a una extensión de la automatización programada, diferenciándose en que en la primera se tiene la capacidad para cambiar parte de los programas y/o la disposición física, sin tiempo perdido de producción (on line).

Existen variados tipos de sistemas automatizados. Cada uno responde a un diferente grado de productividad y flexibilidad, según las necesidades de la empresa y al tipo de servicio que representan. Rembolt *et al* (CIM and Engineering, 1993) ilustra esta situación con el gráfico que muestra la Figura 1.2.

En el gráfico, se muestra una clasificación de algunos de los conceptos de manufactura moderna imperantes en el mercado. Por una parte se encuentran las líneas de transferencia, las cuales implementan una rígida cadena automatizada de trabajos con una elevada tasa de productividad, pero cuyas instalaciones no son capaces de realizar grandes variaciones en los productos que generan. La fabricación de clavos es un claro ejemplo. Y por otra parte están las empresas que cuentan con máquinas CN para elaborar sus productos. Generalmente estas empresas venden a pedido, ya que las máquinas de control numérico son capaces de procesar distintos diseños de piezas con tiempos de configuración mínimos.

En el centro del gráfico aparecen los SFM o Sistemas Flexibles de Manufactura, quienes ofrecen un compromiso entre productividad y flexibilidad (Ramos, 1998), es decir, son capaces de combinar un buen margen de productividad, dado que cuentan con sistemas de asignación y transferencia de material automático, y contar con la flexibilidad que brindan las máquinas de control numérico.

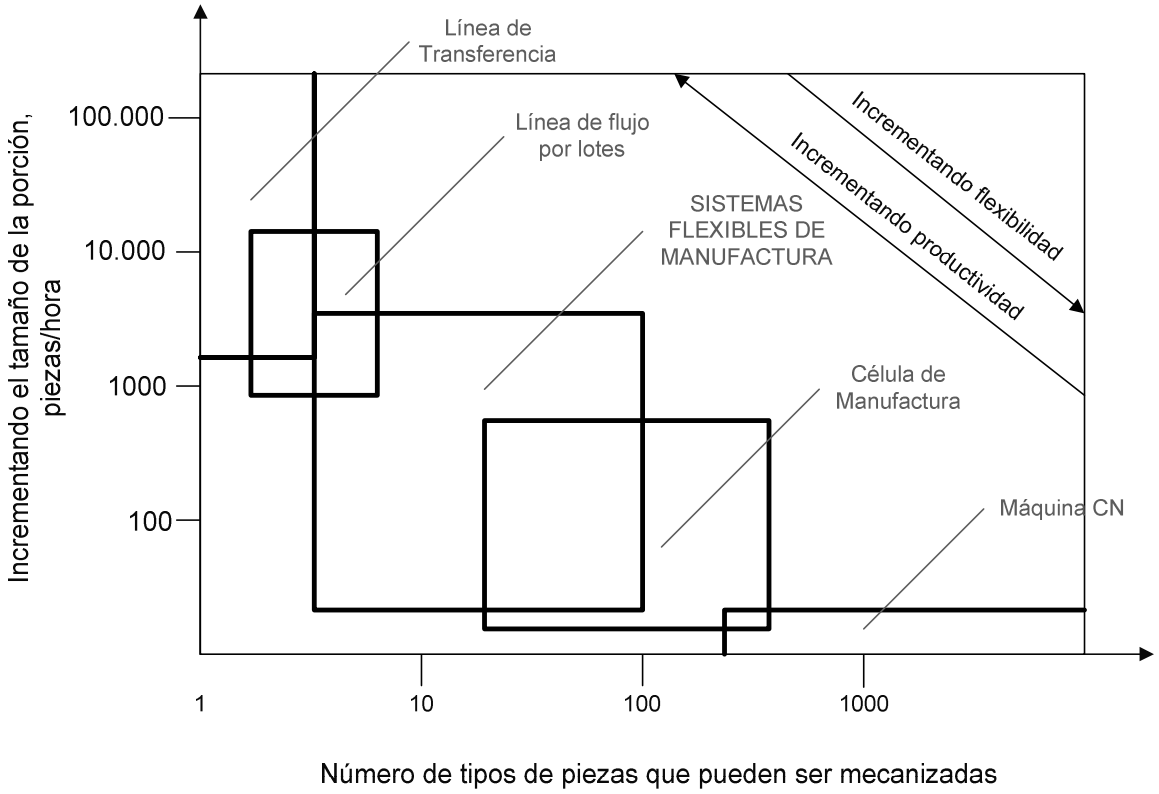


Figura 1.2. Conceptos de manufactura moderna (CIM and Engineering, 1993).

Sea como sea, las razones para automatizar la producción son varias. Groover (1980) destaca las siguientes:

a. El incremento de la productividad,

b. el alto costo de la mano de obra,

c. la escasez de mano de obra,

d. la tendencia de la mano de obra hacia el sector de servicios,

e. seguridad,

f. el alto costo de las materias primas,

g. mejoras en la calidad del producto,

h. reducción de los tiempos de entrega,

i. reducción del inventario, y, por último,

j. el *alto costo de no automatizar* (costo de oportunidad).

1.1.3 Sistemas Flexibles de Manufactura.

Ya se ha hablado un poco de lo que es un Sistema Flexible de Manufactura (SFM), y como se diferencia con los demás sistemas automatizados de producción. Ahora se realizará una aproximación más formal al concepto.

Diversos autores han definido el término Sistema Flexible de Manufactura. Las definiciones concuerdan en varios puntos; máquinas CNC, sistema de transporte, interconexión, etc. Sin embargo, muchas definiciones son aún demasiado vagas.

En el presente documento se utilizará el concepto de SFM como lo ha declarado Mikell P.Groover.

Groover (1980), define un *Sistema Flexible de Manufactura* como un *grupo de estaciones de procesamiento (predominantemente máquinas CNC), interconectadas por medio de un sistema de manipulación y almacenamiento de material automatizado y controlado por un sistema computacional integrado*.

La Figura 1.3 muestra un SFM como el que describe Groover.

Figura 1.3. Sistema Flexible de Manufactura.
(Fuente: CIMware. A Flexible Manufacturing System (FMS) R&D Project (1976/80). [on line] http://www.cimwareukandusa.com/FMSdevelopment.htm [consulta: 25 de julio de 2004]).

1.1.4 CIM: Computer Integrated Manufacturing.

CIM son las iniciales de la palabra Computer Integrated Manufacturing, que hace referencia a un concepto que afecta a toda una organización, y que indica que los procesos y datos de una empresa manufacturera deben ser integrados y coordinados por medio de un sistema computacional que facilite las tareas administrativas[10].

En el fondo, CIM es la integración total de todas las ciencias de la ingeniería, agrupadas en torno a un objetivo: automatizar la empresa para obtener mejores resultados. En términos prácticos, esta integración consiste en la unificación de todos los procesos existentes en el modelo de *gestión de la producción* (CIM and Engineering, 1993), desde la

[10] Aunque el término CIM es, más o menos, uno sólo, diversas instituciones proponen conceptos y modelos propios. Tal es el caso de compañías como la IBM, Digital Equipment Corporation, Siemens y organizaciones como NIST (Nacional Institute Of Standards and Technology).

planificación de la producción, el diseño de productos y la gestión operativa, en general, hasta la manufactura y la entrega al cliente.

Polakoff (1991) habla de que la implementación de CIM en una empresa acarrea de un 5 a un 15% de reducción en los costos de personal, de un 15 a un 30% de los costos de ingeniería de diseño, de un promedio de 250% de aumento de la calidad de los productos y de hasta un 3.500% en el aumento de la productividad ingenieril, entre otros resultados. Rembold *et al* (CIM and Engineering, 1993), agrega que la reducción de los costos de diseño varía entre el 15 y el 30% del valor original, un incremento de la productividad del 40 al 70%, la reducción de desechos de productos entre un 20 y un 50% por conceptos de calidad y una mejora sustancial en el diseño de los productos (mediante la utilización de tecnologías de apoyo a la ingeniería).

El concepto de CIM acarrea a los SFM como la forma de automatizar la producción, de una forma flexible, de una manera eficiente. Existe un concepto fundamental en todo esto, y es que para CIM, debe existir dos cosas: primero, un grado de automatización importante (ya se ha visto el tema de la automatización de la producción y los SFM), y segundo, que las máquinas deben ejecutar diferentes tareas en diferentes piezas con tiempos de configuración despreciables.

Esta integración es realizada a través de un sistema informático. La idea es unificar la información y coordinar las distintas tareas en forma eficiente. En CIM, esto se realiza a través de una red de computadores interconectados, con bases de datos locales y distribuidas. Esta situación es ilustrada en la Figura 1.4.

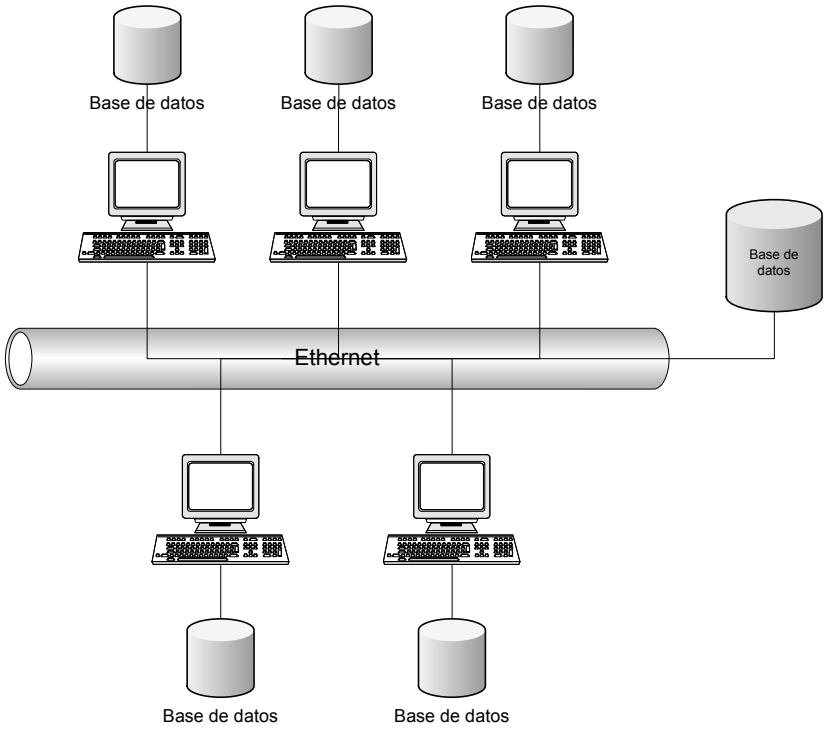


Figura 1.4. Integración de la empresa por medio de una red de computadores.

Sin embargo, pese a toda esta maravilla tecnológica que está sacudiendo el concepto de empresa manufacturera, los problemas no están ausentes. La siguiente sección menciona algunos de estos inconvenientes.

1.1.5 La problemática en la fábrica CIM.

Ante todo, CIM, como concepto tradicional, trae en su definición un concepto que no deja de ser trascendente: el de la centralización.

En CIM se conciben sistemas de manufactura flexibles de administración completamente centralizadas. Muchos autores defienden esta postura, acudiendo al hecho de que la centralización de las decisiones lleva a la definición de una programación lógica y concisa, que la simulación es mucho más fácil, etc. (CIM & Engineering, 1993).

Sin embargo, el dominio centralizado de las labores de gestión de un sistema flexible de fabricación trae una cantidad sustancial de problemas, que no dejan de ser

importantes tratándose de un sistema que debe responder con altos niveles de rendimiento debido a su costo. Al respecto, Botti y Giret (2003) alegan que:

i. Los sistemas CIM tienen una jerarquía de control fija que no soporta cambios,

ii. la reconfiguración y extensión de sistemas existentes es difícil,

iii. la eficiencia de producción no se mantiene fuera de las condiciones normales de producción,

iv. los datos para diagnosis correspondientes a las maquinarias son difíciles de acceder, y

v. el control es completamente automatizado y la intervención humana se excluye.

El enfoque utilizado recibe el nombre de *top-down*, que quiere decir que el modelamiento de la estrategia se concibe desde los niveles superiores de la jerarquía, hasta el nivel de campo, en el último eslabón.

En efecto, la implementación tradicional de CIM dificulta la modificación de su estructura jerárquica y de la configuración de software y hardware, obligando a los inversionistas a depender, generalmente, de un único proveedor (y obligarse en el futuro para su mantención). *(Problemas i. y ii.)*. Estos problemas son de significación en el largo plazo, ya que las instalaciones podrían funcionar en forma óptima, en el mejor de los casos, con una misma configuración durante mucho tiempo.

Sin embargo, en los puntos *iii.*, *iv.*, y *v.* se tienen problemas en realidad graves, pues son básicamente problemas de concepción que pueden traer consecuencias de magnitud en el corto plazo.

Generalmente, en los sistemas de fabricación se determina un algoritmo de asignación de tareas, que está definido de antemano, para resolver la problemática de la

producción (que se explicará en la siguiente sección de este capítulo). Estos sistemas funcionan bien en condiciones normales, el problema ocurre cuando se produce un disturbio inesperado, y el controlador es incapacitado para ejecutar su trabajo (Bussman *et al*, 2001).

1.2 Programación de la Producción.

Uno de los principales problemas con los SFM, y en general con todos los sistemas de manufactura existentes (automatizados o no), es la programación de la producción o "scheduling", esto es, la asignación de un conjunto de recursos limitados en un cierto tiempo (Pinedo, 1995).

Puede que el problema expuesto no suene tan complicado en un comienzo, pero existen muchos factores que solucionar.

1.2.1 La problemática general de la programación de la producción.

Para definir la problemática existente en las empresas manufactureras, acerca de la programación de la producción, se debe considerar lo siguiente: un proceso productivo, cuenta con un conjunto de *m* máquinas, $\{M_1, M_2, \ldots, M_m\}$, las que son capaces de procesar una serie de trabajos. Además se cuenta con un conjunto de *n* trabajos (jobs) a realizar, $\{J_1, J_2, \ldots, J_n\}$. Cada uno de estos trabajos está definido como un conjunto de operaciones *i* sobre máquinas *j*, es decir:

$$J_k = O_{i,j} \qquad \text{Fórmula 1.1.}$$

Cada operación $O_{i,j}$, tiene asignado un tiempo de proceso (process time) $P_{i,j}$. Por último, existe una fecha de llegada (ready time), r_i, que es cuando la tarea *i* está lista para relizarse.

En definitiva: el problema general de la programación de la producción es *asignar las tareas a los procesos, respetando un conjunto de restricciones (precedencia y tiempo), garantizando un cierto criterio de eficiencia*.

Este grado de eficiencia del sistema puede ser:

i. La fecha en que se completa una orden,

ii. el tiempo medio de las piezas en el sistema,

iii. el atraso máximo,

iv. el atraso promedio,

v. el número de trabajos atrasados, etc.

Esta problemática será el objetivo final a solucionar en el presente proyecto. El lograr definir un modelo, que sea capaz de solucionar (hasta cierto punto), el problema de asignar las diversas tareas a ser realizadas en un sistema manufacturero automatizado.

1.2.2 Secuenciamiento de procesos de fabricación.

De forma general se pueden distinguir tres tipos de programación de los procesos de fabricación en una industria manufacturera. Estos son los llamados: *open shop*, *flow shop* y *job shop*.[11].

a. Open shop (Figura 1.5). Este tipo de programación es la más fácil de solucionar. Se cuenta con m máquinas, pero cada trabajo Jk puede ser realizado en cualquier máquina. No se cuenta con restricciones de precedencia. El programador determina la ruta a seguir por cada trabajo.

[11] Una descripción más detallada de estos conceptos y sus aplicaciones se pueden encontrar en [Pinedo, 1995].

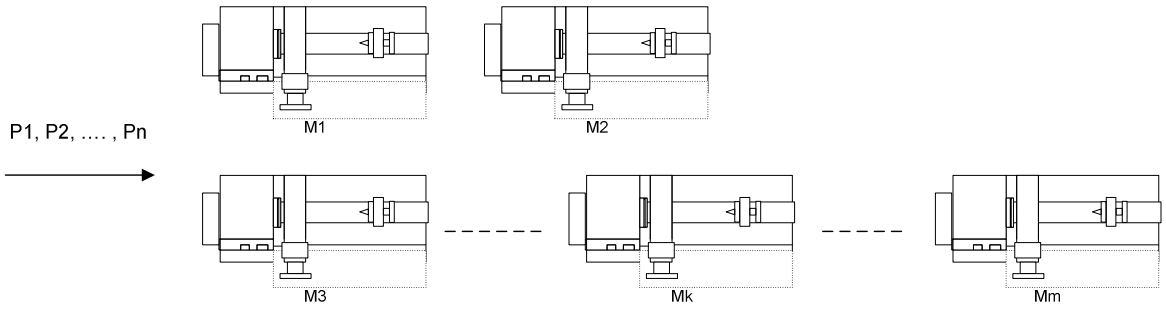


Figura 1.5. Open Shop.

b. Flow Shop (Figura 1.6). En flow shop, cada trabajo sigue siempre la misma secuencia. Se cuenta con una sucesión de operaciones ordenada, las que son sin restricción de precedencia y sin retornos. Esto quiere decir que si alguna máquina se encuentra ocupada en el momento en que llega otra, la nueva tarea, y las que lleguen detrás de ella, deberán esperar su turno de ingresar a la máquina para ser procesadas. Fuera de la máquina, los trabajos esperan su turno, e irán ingresando, cuando la máquina se desocupe nuevamente, según alguna lógica o algoritmo determinado, por ejemplo, declarar una secuencia de entrada como FIFO (first in first out), LIFO (last in first out), SPT (short process time), etc.

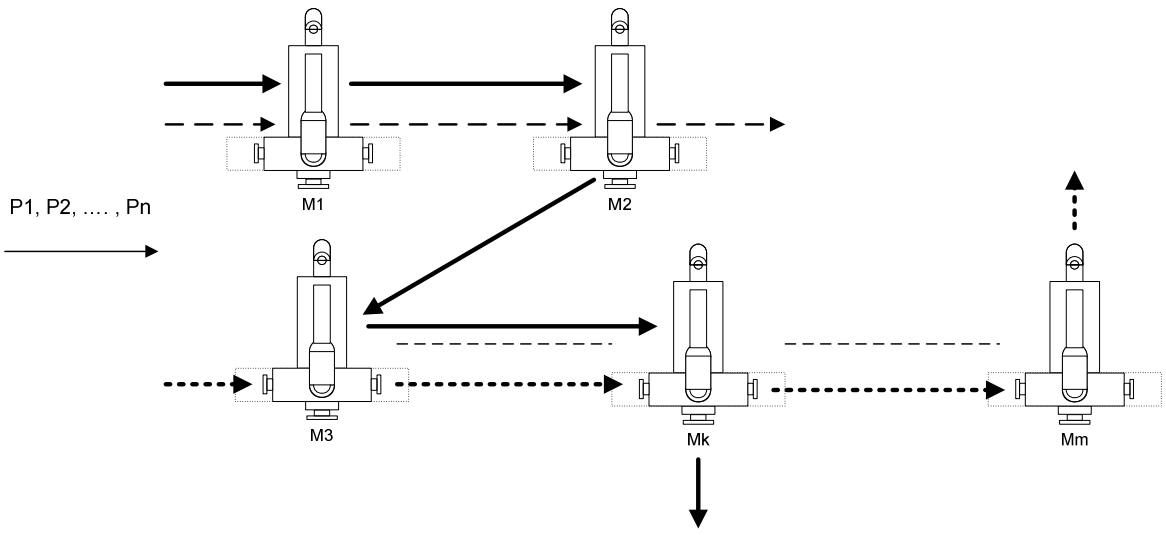


Figura 1.6. Flow Shop.

c. Job Shop (Figura 1.7). Job Shop es la forma de programación más compleja. Con Job Shop cada trabajo sigue su propia ruta.

   Se deben distinguir tres tipos de Job Shop: uno en el que cada trabajo visita una máquina M_j una vez como máximo (sin recirculación), la segunda es cuando una tarea puede volver a la misma máquina Mi varias veces. En este último caso se habla de que el proceso es con *recirculación*. Finalmente se encuentran los FMS, los cuales también soportan la recirculación, pero además se puede contar con máquinas paralelas (una máquina Mi en el modelo puede representar en realidad dos o más máquinas del mismo tipo).

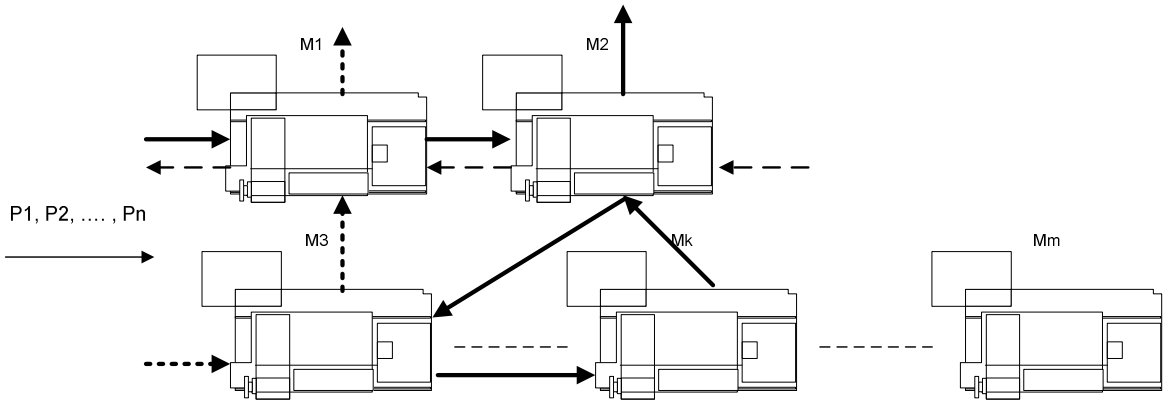


Figura 1.7. Job Shop.

1.2.3 On-line vs. Off-line.

Como se ha visto hasta el momento los factores a considerar son bastantes. Primero los datos generales como las tareas a realizar, las máquinas disponibles y los tiempos de procesamiento. Segundo el tipo de programación a implementar, que en este caso, se tata de una programación Job Shop. Hasta acá ya es bastante complejo, pero falta un detalle.

Existen dos tipos clave de scheduling (programación), dependiendo del momento en el que se realice; estos son: el scheduling *off-line* y el scheduling *on-line*.

La programación *off-line* es aquella que se realiza, ya sea en forma automática o no, fuera de todo proceso de fabricación que se encuentre operando en un momento dado. La idea, es llegar a determinar una solución para el problema de secuenciamiento antes de arrancar un proceso productivo.

La programación *on-line* es la que se realiza durante el proceso de manufactura propiamente tal. En el enfoque más tradicional de CIM, un solo computador central realiza esta tarea, obteniendo el estado del sistema fabril completo y ordenando la secuencia de entrada a cada máquina disponible en tiempo real. Este equipo envía las órdenes correspondientes a otros equipos subordinados quienes se encargaran de controlar los dispositivos de campo para manipular y transformar el material, manteniéndose todo el tiempo informado del estado de la maquinaria y la posición de cada pieza que se encuentre en el sistema. La programación *on-line* es constante durante el proceso de fabricación.

Muchos autores han realizado investigaciones en el tema del secuenciamiento *off* y *on-line*, obteniendo excelentes resultados en ambas estrategias. Por nombrar algunas experiencias se puede citar, en el tema de la programación off-line, el trabajo de Koller (2004), quien desarrollo un modelo para la programación *off-line* de un aserradero. En programación *on-line*, se puede mencionar a Ramos (1998), con su trabajo en sistemas auto-organizados[12].

Wan (1993) realizó una comparación entre ambas opciones, exponiendo las falencias de cada una. En todo caso, es mejor no tratar de hacer competir ambas tendencias, sino llegar, tal vez, a una correcta integración.

[12] Ambos autores mencionados se apoyaron en la Inteligencia Artificial Distribuida para realizar sus experiencias.

Capítulo 2

Sistemas Auto-Organizados De Producción

Al observar el entorno es posible notar que las diferentes entidades que pertenecen y participan de él siempre presentan cierto grado de interacción con otras, sean éstas similares o no. En efecto, una persona que se cruza con otra caminando por la acera de una avenida (en una situación normal[13]), percibe ciertos aspectos de la otra, tal como su distancia, su tamaño, rasgos faciales generales, ropaje, si lleva algún artefacto adicional (un bolso, una carga, etc.), su aspecto en general, la velocidad con la que camina, etc., con el objeto de analizar una posible situación de impacto, reconocer al sujeto, predecir una mala intención (como un asalto), etc.. Obviamente la interacción es mayor si se detienen a conversar. A la vez estas personas interactúan con otras más en mayor o menor grado; y no sólo con personas, también con entes de otras especies (animales, el tráfico, edificios, el sol, una piedra en el camino, etc.), en general, con toda entidad presente en el entorno directo (viéndose afectada además, en cierto, grado por entidades de su entorno indirecto).

Muchos problemas del mundo real han podido ser solucionados mediante la formulación de complejos cálculos matemáticos, otros mediante el diseño de algoritmos de características secuenciales (como máximo, concurrentes) que realizan ciertas rutinas en determinadas situaciones. La resolución de estos permite a las personas aproximarse al resultado o estado intermedio del comportamiento de determinadas entidades o situaciones de las más distintas naturalezas. Ejemplos de esto existen en casi todas las ciencias: la estadística aplicada, elementos finitos, simulación de procesos, realidad virtual, etc.

[13] Pensando en que ambas personas no presentan discapacidades de ningún tipo (o por lo menos las que les prohíban ser incluidas en el presente ejemplo), o simplemente que no vayan caminando totalmente distraídos.

Inclusive técnicas de la misma Inteligencia Artificial (IA) basan su funcionamiento en algoritmos y cálculos secuenciales[14].

Pero en determinadas ocasiones paradigmas de este tipo no son suficientes. Las necesidades de simular múltiples entidades, comunicaciones paralelas, división y asignación de tareas y recursos, cooperación, etc., son tareas complejas que necesitan de otra visión para ser representadas.

Es aquí donde entran en juego los sistemas auto-organizados, tema del que habla el presente capítulo.

[14] Por supuesto que muchos problemas pueden dividirse para resolverse en forma paralela, pero esto es, la mayor parte de las veces, sólo para disminuir la carga computacional existente. Tal es el caso de la llamada Inteligencia Artificial Paralela (Parunak, 1988).

2.1Distribución del control en un sistema de fabricación

Los sistemas automatizados de producción han traído grandes beneficios a las empresas y a las personas, en general. La máquina ha reemplazado al hombre en casi todo ámbito de participación en los procesos manufactureros de carácter industrial en las empresas más modernas del mundo.

Inicialmente maquinarias completamente mecánicas requerían la asistencia de una o más personas para operar. Eventualmente, la llegada de máquinas con dispositivos de control electrónico facilitaron las tareas, incorporando la programación de ciertas rutinas en un determinado lapso de tiempo. Hoy en día las grandes compañías están dotadas de modernos equipos de última generación.

En términos de control, las arquitecturas actualmente vigentes emplean dispositivos de control de campo, principalmente los PLC (Programmable Logical Controller). Además, en las últimas décadas, se ha incorporado la utilización de computadores para realizar determinadas tareas de monitoreo y administración, facilitando aún más la tarea de los operarios, quienes pueden supervisar el estado de cualquier dispositivo dentro de un sistema de producción y tomar las medidas correspondientes con sólo un clic de mouse. Este esquema de trabajo es adoptado, hoy en día, no sólo por la industria manufacturera, sino también por las empresas que explotan recursos naturales, los centros de salud, el ejército, etc.

Básicamente toda configuración de control distribuido debe obedecer a una de las clasificaciones expuestas en la Figura 2.1.

En la ilustración, cada nodo corresponde a un dispositivo computarizado programable, capaz de controlar ciertos procesos. Estos dispositivos pueden ser computadores, PLC's, etc. (entidades de manufactura, como robots, máquinas CNC, y otros, no aparecen representados en la figura).

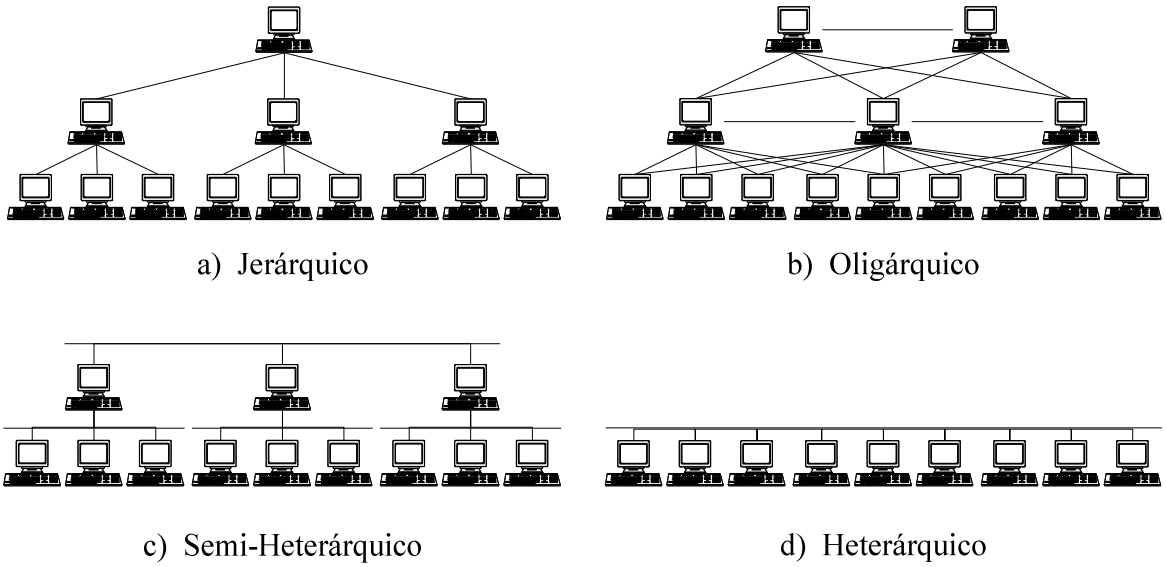


Figura 2.1. Espectro de los sistemas de control de manufactura distribuidos (según Duffie y Prabhu, (1996)).

Un sistema de control jerárquico (Figura 2.1.a) implica una arquitectura en la que una unidad de central se encarga de organizar las tareas del sistema completo a un nivel global, destina órdenes a los diversos dispositivos subordinadas y se realimenta con la información reportada por éstas. Inicialmente, la primera arquitectura de control fue centralizada, esto es, que un solo equipo dirigía directamente las labores de todos los dispositivos de campo existentes, como muestra la siguiente figura:

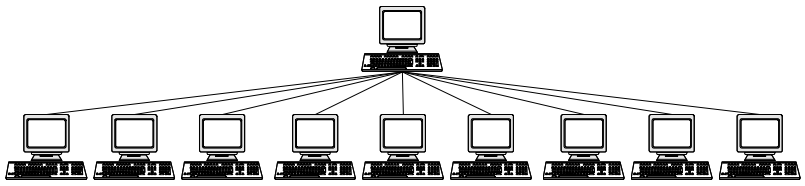


Figura 2.2. Arquitectura de control centralizada.

Definitivamente esto significaba una excesiva carga para la unidad de control. Posteriormente se evolucionó a una arquitectura apropiadamente jerárquica, que se explicaría con la ilustración de la Figura 2.1.a. Este último caso es propio de la implementación tradicional de un sistema CIM, donde el nivel intermedio correspondería al nivel de control de células flexibles de manufactura (CFM).

Según Frayret (2002) y Frayret *et al* (2004) las ventajas de un sistema centralizado son:

i. Toma de decisiones global optimizada

ii. Complejidad limitada

En tanto, sus desventajas son las siguientes:

i. Lenta velocidad de respuesta

ii. Pobre confiabilidad

iii. Modificaciones, adiciones, y extensiones son difíciles de realizar.

Un sistema jerárquico (como el de la Figura 2.1.a) mejora la propuesta anterior, incorporando otros beneficios, sin embargo, agrega otros inconvenientes.

Los beneficios de un sistema apropiadamente jerárquico son las siguientes:

i. Unidades de control coordinan objetivos

ii. Implementación del sistema facilitada

iii. Complejidad limitada

iv. Tiempo de respuesta mejorado

Y sus deficiencias:

i. Inadecuado para responder ante eventos inesperados

ii. Pobre capacidad de extender el sistema y realizar modificaciones imprevistas.

iii. Pobre confiabilidad (parálisis de los niveles debajo de un punto de falla)

iv. Pobre tolerancia a las fallas

Distribuyendo la carga de un controlador centralizado, y el riesgo de falla de una parte importante del sistema (generalmente el sistema completo) se encuentran los sistemas oligárquicos (Figura 2.1.b), los que siguen practicando una arquitectura de control multi-capa, pero, con la salvedad de que se incorpora más de un controlador superior, existe una comunicación horizontal (con unidades del mismo nivel) y la inclusión de la subordinación de un nodo cualquiera, no a un solo aparato del nivel inmediatamente superior, sino que a varios.

El sistema semi-heterárquico (Figura 2.1.c) incorpora sub-sistemas heterárquicos ligados a una heterarquía de control global.

Finalmente, en la Figura 2.1.d se expone una arquitectura de control heterárquica.

Los sistemas heterárquicos fueron inspirados por los sistemas biológicos y economías de mercado. En éstos, el funcionamiento global del sistema se acuerda por homologación de todas (o parte de) las unidades que componen el sistema, existiendo la posibilidad de comunicar cada una de estas directamente.

En [Frayret, 2002] y [Frayret *et al*, (2004)], se identifican las siguientes ventajas para una heterarquía:

i. flexibilidad y reactividad a los disturbios

ii. Son facilitadas las extensiones del sistema y las modificaciones imprevistas

iii. Complejidad reducida

iv. Tolerancia a fallos

Como se aprecia, muchas de las falencias de los sistemas jerárquicos son solucionadas al distribuir la responsabilidad en una heterarquía. Sin embargo, aún los sistemas de control heterárquicos presentan falencias:

i. Falta de previsibilidad

ii. Pobre habilidad para definir cargas óptimas

iii. Falta de soluciones analíticas

iv. Posibilidad de deadlock (callejón sin salida)

Existen otras arquitecturas, variaciones de las anteriormente nombradas, las cuales eliminan ciertas desventajas de los modelos originales. Tal es el caso de las *jerarquías modificadas* o las estructuras *quasi heterárquicas*.

2.2Sistemas auto-organizados en la industria de la manufactura

Como ya se ha dicho, un sistema flexible de manufactura normalmente está compuesto de un conjunto de máquinas CNC y otros dispositivos de transformación, manipulación, transporte, almacenamiento, etc.

Ante tal cantidad de entidades involucradas, la concepción de un sistema de control jerárquico no siempre resulta ser la mejor opción. Sobre todo cuando las fábricas crecen y los hechos fortuitos se vuelven más frecuentes. Pensar en un control centralizado parece una locura, y las jerarquías son propensas a provocar una falla global (lo que a la larga, para una empresa competidora, significa grandes problemas y hasta pérdidas).

Una solución, podría ser la de concebir un modelo en el que cada entidad, que tenga algún grado de participación en el proceso, pueda interactuar con el resto siguiendo sus propios objetivos, consiguiendo metas y llevando esas metas a conseguir un objetivo global, pero un objetivo global que nace de la emergencia de los objetivos de cada componente del sistema completo (este término es conocido como enfoque botton-up).

Numerosos autores han hecho analogía de diversos fenómenos naturales y matemáticos para encontrar solución a esta problemática. Es así como nacen conceptos como sistemas *celulares*, *biónicos*, *genéticos*, *fractales*, *holónicos*, *NetMan framework* etc. (Tharumarajah *et al*, 1996; Frayret, 2002; Frayret *et al*, 2004), cada uno de los cuales, aplica de forma diferente el enfoque *bottom-up*, y presenta diversos enfoques de solución.

2.3 Inteligencia Artificial Distribuida. Sistemas basados en agentes

La inteligencia artificial distribuida (IAD) está definida como el subcampo de la Inteligencia Artificial (IA) que se centra en los comportamientos inteligentes colectivos que son producto de la cooperación de diversas entidades denominadas agentes (Iglesias, 1998). La necesidad de cooperación entre estos agentes se justifica básicamente por uno (o ambos) de los siguientes motivos: Primero, el problema es demasiado complejo y cada agente no es capaz de resolver el problema por sí mismo; y segundo, es más rentable la solución conjunta.

Pese a que las investigaciones en el tema de los agentes y, particularmente, en el campo de la IAD han aumentado considerablemente en las últimas décadas, abarcando aplicaciones en casi todas las áreas de las ciencias, actualmente no existe una definición formal comúnmente aceptada en la comunidad científica para el término *agente*. De hecho, al hacer un barrido por la bibliografía, es posible visualizar que el significado de esta palabra difiere drásticamente de ámbito en ámbito. Sin embargo, la mayor parte de los investigadores concuerdan en varios puntos. Lo que sigue corresponde a los fundamentos teóricos necesarios para comprender el tema en que se centra el presente trabajo de

investigación, desde la definición formal de agente adoptada en el presente texto, hasta una introducción al ámbito central de la IAD: los sistemas multi-agentes.

2.3.1 Áreas de la Inteligencia Artificial Distribuida. Una breve introducción.

Muchos problemas pueden ser implementados como un conjunto de tareas que operan en forma paralela, generando procesos que apresuran considerablemente el resultado obtenido. Sin embargo, estas técnicas no son consideradas usualmente como técnicas de IAD. Parunak (1988) sostiene que estas técnicas deberían ser clasificadas como *Inteligencia Artificial Paralela* (IAP). La diferencia es clara; mientras la IAP se concentra en la distribución de una sola unidad de software en diversos procesadores, la IAD se enfoca en la distribución de tareas y objetivos de varias unidades independientes.

La IAD, propiamente tal, posee dos grandes áreas ampliamente difundidas a través de años de investigación (Durfee y Rosenschein, 1994): La Resolución de Problemas Distribuidos y los conocidos Sistemas MultiAgente. Las diferencias entre ambos enfoques son leves pero significativas[15]. Ambos enfoques se describen brevemente a continuación:

2.3.1.1 Resolución de Problemas Distribuidos

Las técnicas de Resolución de Problemas Distribuidos (RPD) se basan en la suposición de que un problema particular puede ser solucionado por un conjunto de entidades (agentes) que cooperan en al dividir y compartir conocimiento sobre el problema y su solución. Estos sistemas se caracterizan porque cada agente o unidad tiene un comportamiento fijo, con tareas prefijadas de antemano. Además, existe un plan centralizado de resolución del problema. En este tipo de problemas suele haber un agente que ejerce un control global sobre un conjunto o el total de agentes involucrados. Para

[15] Estas diferencias no hacen de estas dos aproximaciones polos opuestos, o completamente excluyentes. Según la primera "visión" de Durfee y Rosenschein (1994) la Resolución de Problemas Distribuidos podría ser considerada como un subconjunto de los Sistemas MultiAgente.

Durfee y Rosenschein (1994), un claro ejemplo de esto podría ser el enfoque Contract-Net para descomponer y asignar tareas en una red[16].

2.3.1.2 Sistemas Multiagente

Un área más evolucionada de la IAD que las técnicas de RPD son los Sistemas Multiagente. Como este es el tema en el que se centra el presente capítulo, no se pretende dar aún una definición más detallada de este enfoque. Esto se tratará en amplitud el tema en la sección 2.3.4.

2.3.2 Agentes: Definición y Características

Maes (1994) define un agente como un *sistema que intenta satisfacer un conjunto de metas en un ambiente dinámico complejo*. Posteriormente, Russell y Norvig (1996) postulan que un agente es *todo aquello que puede considerarse que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores*.

De lo anterior se rescatan algunos conceptos importantes, y que están presentes en la mayor parte de los trabajos. Primeramente el agente está definido como un sistema. Si se entiende por sistema un *conjunto de partes* (subsistemas) *coordinadas y en interacción para alcanzar un conjunto de objetivos*[*], se extiende en demasía el término "agente". Ya que se está trabajando en el área de la Inteligencia Artificial (y la palabra artificial sugiere algo que no es natural), en el presente documento se entenderá que un agente es un ente artificial, como un programa de computador o hardware o ambos[17]. El concepto de *meta* u *objetivo* es primordial. Básicamente la acciones que realice un agente deben ser impulsadas por el logro de uno o varios objetivos determinados. El *ambiente* no es otra cosa que el

---

[16] Esto último, como se demostrará más adelante, no es necesariamente así. Mucho depende de la conceptualización y aplicación del método. Investigadores han demostrado la aplicación real del protocolo Contract-Net en Sistemas Multiagente autoorganizados, propiamente tal, como el enfoque propuesto en Ramos (1998).

[*] Definición de "sistema" en [JOHANSEN B., O. 1986. Introducción a la teoría general de sistemas. México, Limusa. 167p].

[17] Algunos autores sostienen que los seres humanos deben ser consideradas *agentes* (como en [Franklin y Graesser, 1996]). En el presente texto se entenderá por agente un sistema computacional según la definición de Wooldridge y Jennings (1995b), a no ser que se indique explícitamente lo contrario.

conjunto de los factores externos al agente que lo afectan directa o indirectamente, y sobre los cuáles puede tener alguna incidencia. Este ambiente provee las condiciones en las cuales un agente puede existir (Odell *et al*, 2002). El agente debe ser capaz de percibir el ambiente en el que está inmerso y para esto debe estar dotado de mecanismos denominados sensores. Además, el agente debe poder tener la facultad de actuar sobre el medio y modificar determinados parámetros de éste. Para esto,.un agente cuenta con uno o varios efectores.

Esta definición es acertiva en muchos aspectos, sin embargo se limita a lo esencial. Abarca demasiados conceptos en lo que propone como agente. Según esta definición cualquier aplicación distribuida o que opere conectada en una red en comunicación con otras aplicaciones sería un agente. De hecho, en estricto rigor, cualquier programa computacional lo sería (pensando en los sensores como las entradas de parámetros desde el usuario o el sistema operativo y en los efectores como la consecuencia de la ejecución de cualquier comando sobre la máquina).

Una de las definiciones que ha tenido mayor acogida entre los investigadores es la que se extrae de [Wooldridge y Jennings, 1995b]. En este documento un agente es un sistema computacional dirigido por sus objetivos, capaz de interactuar con su entorno de forma flexible y autónoma.

Y es que esta simple definición abarca mucho más de lo que aparenta. La clave está en las palabras *"flexibilidad"* y *"autonomía"*. *Autonomía* es la capacidad de los agentes de operar con la intervención directa de seres humanos u otros sistemas, y la de poseer algún tipo de control sobre sus acciones y su estado interno. Por otra parte, en este contexto, *flexibilidad* es la unificación de tres conceptos fundamentales: *habilidad social*, *reactividad* y *pro-actividad*.

i. *Habilidad social* es la capacidad de un agente de interactuar con otro agente (y posiblemente con seres humanos), por medio de un lenguaje de comunicación de agentes (Genesereth and Ketchpel, 1994).

ii. *Reactividad* es la capacidad para reaccionar frente a determinados estímulos del entorno. Estos estímulos guían el comportamiento de un agente.

iii. Por último, la *pro-actividad* es la capacidad de un agente de actuar motivado por sus propios objetivos (no necesariamente actuar frente a estímulos del entorno, si no por iniciativa propia).

En la Figura 2.3 se aprecia el modelo de un agente como un ente virtual, capaz de percibir estímulos de su entorno, y actuar sobre éste.

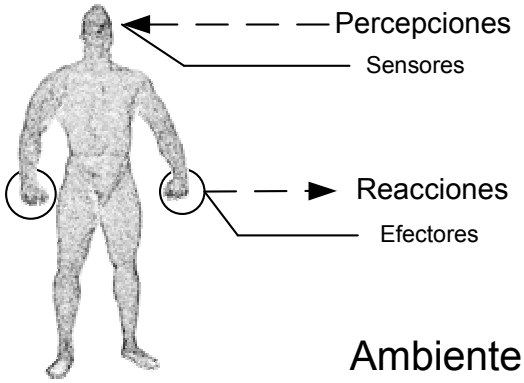


Figura 2.3. Modelo de un agente según la definición de Wooldridge y Jennings (1995b).

Aparte de las mencionadas características se han identificado otras cualidades, las cuales pueden ser atribuidas a los agentes en menor o mayo grado:

i. *Racionalidad*: Un agente es capaz de razonar en base a determinados datos percibidos con el objeto de optimizar un resultado que de solución a un problema.

ii. *Adaptabilidad*: Propiedad de los agentes para aprender bajo determinadas situaciones y modificar su propio comportamiento en base a dicho aprendizaje.

iii. *Movilidad*: La capacidad de un agente para moverse en una red.

iv. *Veracidad*: Todo agente debe entregar información fidedigna y no falsificarla deliberadamente.

v. *Benevolencia*: La capacidad para ir en auxilio de otro agente (siempre que esto no juegue en contra de sus objetivos).

2.3.3 Clasificación de Agentes

Nwana (1996) presenta una primera aproximación a lo que sería una tipología de agentes, según los atributos que éstos debieran manifestar. Esta tipología es representada gráficamente en la Figura 2.4.

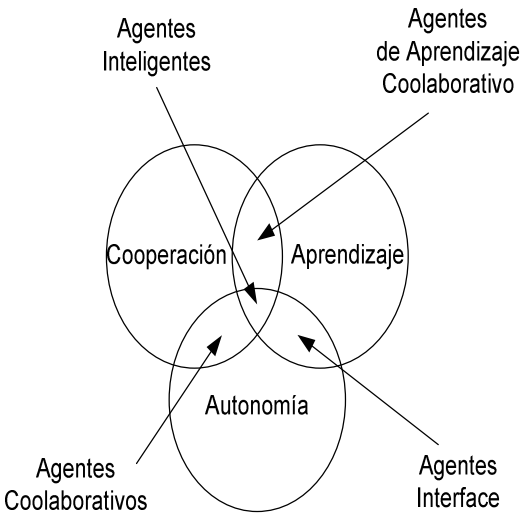


Figura 2.4. Tipologías de Agentes según sus atributos (Nwana ,1996).

Así, un agente "Inteligente" (en inglés "*smart agent*") es un agente que es autónomo, coopera con otros agentes, y además es capaz de aprender. Más adelante en su texto, Nwana propone una clasificación de los tipos de agente que se pueden identificar, abarcando todas (o la gran mayoría de las posibilidades). Según lo que indica en su trabajo, los agentes pueden subclasificarse en 8 tipos (Figura 2.5):

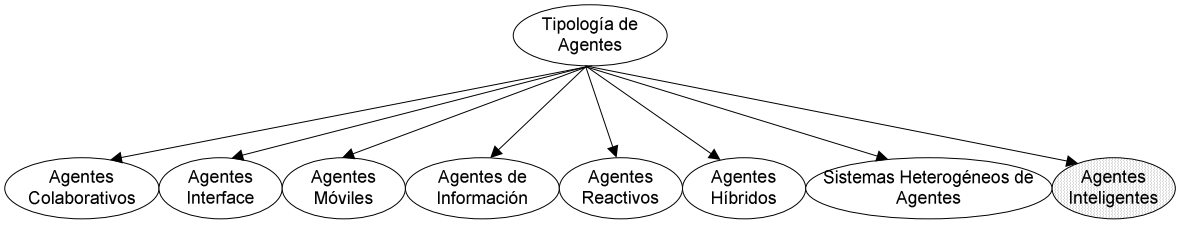


Figura 2.5. Tipologías de Agentes (Nwana ,1996).

i. Agentes Colaborativos (Collaborative Agents): Son agentes autónomos que se caracterizan por la cooperación con otros agentes (con otros agentes). Para coordinarse acerca de cómo resolver un problema, los agentes colaborativos podrían necesitar *negociar*[18], para llegar a un acuerdo de la forma en la que deberían resolver un determinado problema. Estos agentes son la base de los sistemas multiagente (que se explicarán en la sección 2.3.4). Se basan en la premisa de que en conjunto se puede conseguir resultados más allá que las capacidades de cualquiera de sus miembros.

ii. Agentes Interface (Interface Agents): Son agentes utilizados principalmente como "asistentes" para un usuario en una determinada tarea. Utilizados tanto para entrenar como para suplir a un usuario en quehaceres que requieran cierta supervisión computacional. El agente se acomoda al nivel de un usuario para entrenarlo en ciertas labores, o aprende de él, para suplirlo en tareas repetitivas y tediosas.

iii. Agentes Móviles (Mobile Agents): Son agentes capaces de movilizarse a través de una red de computadores. Partir de un host de origen con alguna misión, visitar varios ordenadores buscando información o realizando alguna operación, y luego volver a su lugar de origen. Hoy en día, con el enorme catastro de aplicaciones en la Internet, los agentes móviles se han convertido en uno de los principales focos de investigación para la comunidad informática en el área de la IAD.

[18] Un número considerable de propuestas se han realizado hasta la fecha para resolver este problema de la negociación entre los agentes.

iv. Agentes de Información (Information Agents): Los agentes de información cumplen la importante misión de recopilación, procesamiento o análisis de información de varias fuentes (generalmente distribuidas).

v. Agentes Reactivos (Reactive Agents): Generalmente utilizados en tareas muy específicas, se caracterizan por no poseer un marco simbólico interno de su entorno, es decir, necesitan muy poca información del ambiente, limitándose a reaccionar de ciertas formas ante determinados estímulos externos.

vi. Agentes Híbridos (Hybrid Agents): Estos son agentes que combinan dos o más filosofías en sí mismos. Por ejemplo, un agente que es colaborativo y reactivo a la vez. Nwana sostiene se pueden obtener beneficios "ideales", al combinar los beneficios de las distintas filosofías de cada tipo (base) de agente en un agente híbrido. Por lo general la gran mayoría de los agentes encontrados en la literatura poseen dos o más características básicas, por lo que pueden ser fácilmente catalogados de agentes híbridos.

vii. Sistemas Heterogéneos de Agentes (Heterogeneous Agent Systems): He aquí una "especie" de sistema multiagente. Se describe como un conjunto de dos o más agenentes que corresponden a dos o más clases de agentes diferentes.

viii. Agentes Inteligentes (Smart Agents): Tal como se mostró en la Figura 2.4, un agente Inteligente es aquel que cumple con las características de cooperación, autonomía y aprendizaje, conceptos que ya se han explicado ampliamente. Utopía o no, este tipo de agentes es el más representativo de los objetivos de la Inteligencia Artificial: la posibilidad de generar entes computacionales capaces de "pensar" por sí mismos tal (o casi) como lo hace un ser humano.

2.3.4 Los Sistemas multi-agente

Este es el principal foco de interés de la Inteligencia Artificial Distribuida. Un sistema multiagente (SMA) es un sistema compuesto por dos o más agentes (cada uno con

atributos y objetivos propios), que interactúan entre sí para conseguir un objetivo global. Obviamente, los agentes involucrados en un SMA pueden ser de diversos tipos.

La característica principal es que la solución global es emergente (Ramos, 1998), enfocándose la solución del problema en los distintos nodos. Esto implica que un SMA cumple con lo siguiente:

i. No existe un sistema de control centralizado.

ii. Los datos se encuentran distribuidos.

iii. La computación (el procesamiento) de los datos es asíncrona.

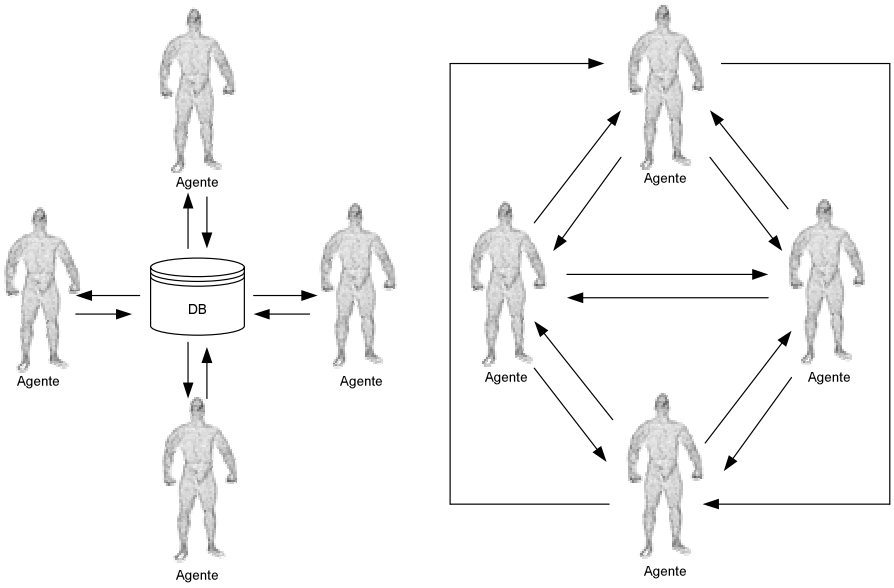
iv. La asignación de tareas es consecuencia de la interacción dinámica de los agentes; entre todos se ponen de acuerdo en qué tareas se realizarán y quiénes las realizarán.

2.3.5 Métodos de Comunicación de Agentes.

Existen varios métodos para realizar la comunicación en un sistema multi-agente. De estos métodos depende la forma en que se tenga que representar el entorno en que habitan los agentes. Los métodos más conocidos son:

i. Arquitectura de Pizarra / Blackboard (Figura 2.5. – parte a): Es cuando los agentes se comunican utilizando un servicio de datos centralizado. En este método de comunicación se representa el ambiente como una base de datos global, en la que los distintos elementos del entorno se encuentran representados, así como el estado de cada agente. Luego, un agente interactúa con el resto realizando modificaciones sobre esta base de datos.

ii. Comunicación por Paso de Mensajes (Figura 2.5. – parte b): Se da cuando los agentes se comunican directamente con otro agente (cuya identidad debe conocer) a través de mensajes.



a) Arquitectura de Pizarra a) Comunicación por Paso de Mensajes

Figura 2.5. Métodos de Comunicación entre Agentes.

Una alternativa a las soluciones antes mostradas (y la solución al problema de exceso de carga en el sistema de comunicaciones o la complejidad de la programación) podría ser el enfoque encontrado en la obra de Genesereth y Ketchpel (1994). En ésta, se expone el concepto de "sistema federado", el cuál se describe como un sistema en el cual los agentes seden cierta autonomía al estar imposibilitados de comunicarse unos a otros directamente (como en la arquitectura de blackboard). En cambio, la comunicación es realizada por entidades llamadas facilitadotes. Estos facilitadotes ejercen las veces de interlocutores entre los agentes ubicados en el mismo sistema computacional, o en otras máquinas, tal como muestra la Figura 2.6.

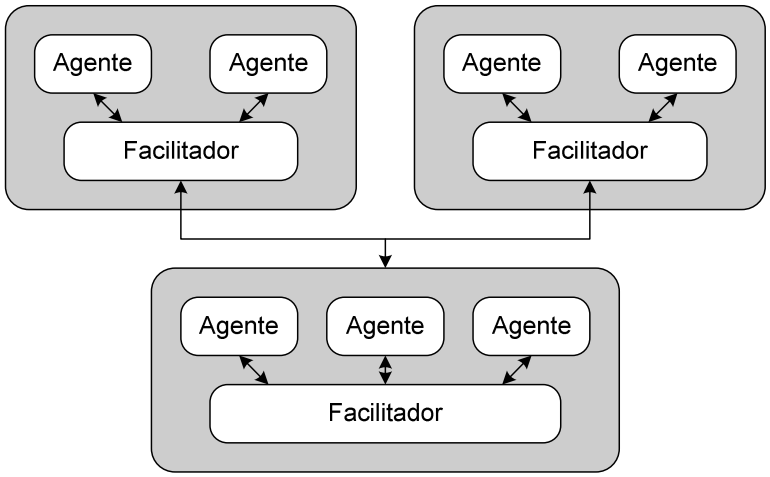


Figura 2.6. Sistema Federado (Genesereth y Ketchpel, 1994)

2.3.6 Aplicaciones de los sistemas multi-agente

Hoy en día los sistemas multi-agente se han aplicado en prácticamente todas las ciencias. Sistemas médicos, Internet, aeronáutica, robótica, aplicaciones de realidad virtual, el ejército y las empresas forestales son algunos de los intereses que esta temática aborda.

En el área de la manufactura integrada han sido numerosas las aproximaciones de investigadores de todo el mundo a modelos enfocados en ciertos temas. Por ejemplo: en el campo de las bases de datos (Wunderli *et al*, 1996), de la planificación de la producción (Koller, 2004) y a las operaciones propiamente tal. En este último aspecto, un grupo de investigadores se ha dedicado al modelamiento de las operaciones desde un nivel macro, considerando toda la empresa, o unidades muy amplias. Otros han decidido enfocarse en problemas más puntuales como las fábricas mismas (Parunak, 1988; Ramos, 1998; Wang *et al*, 1998; Kim y Nof, 2000; Langer y Alting, 2000; Suessmann *et al*, 2002; Shanin, 2002; Charpentier y Muhl, 2004, entre otros). Ese precisamente es el objetivo del presente trabajo. Llegar a un modelo autoorganizado multi-agente de un sistema shop-floor.

Capítulo 3

Metodología Para El Desarrollo Del Sistema

Si bien la Inteligencia Artificial Distribuida ya lleva más de 2 décadas de existencia, su desarrollo se ha centrado básicamente en la solución práctico-técnica de problemas, y la poca teoría formal que existe no pasa, muchas veces, de la simple explicación de conceptos. Esto implica que los investigadores que se mueven en el mundo de la IAD no cuentan muchas veces con metodologías de desarrollo establecidas y probadas rigurosamente para el desarrollo e implementación de sus modelos. La gran mayoría opta por proponer la propia o utilizar una de las metodologías existentes clásicas en el área de la ingeniería de software. Las más utilizadas son aproximaciones orientadas a objetos.

Desde hace pocos años, una nueva línea se ha desarrollado en el ámbito de la formulación de métodos que cubran el proceso de desarrollo de sistemas multi-agente, como una forma de capear la falta de formalismos de desarrollo. Se habla ahora de métodos Orientados a Agentes (OA).

Son varios los autores que han publicado metodologías de apoyo al desarrollo de sistemas multiagente. En este capítulo se da un vistazo general a algunas de ellas, junto con la identificación de sus principales pro y contras. Así mismo, se procede a la selección de la metodología que se utilizará para desarrollar el actual trabajo de investigación, y su descripción. Este paso es fundamental en el desarrollo proyecto.

3.1 Metodologías para el desarrollo de agentes

Sin duda que las técnicas de modelamiento y programación de sistemas ha evolucionado enormemente en las últimas décadas. Como muestra la Figura 3.1, el espectro de técnicas de programación ha variado enormemente. Desde la Programación Estructurada, y con el crecimiento desmesurado de los sistemas informáticos, las metodologías de desarrollo comenzaron a incorporar lenguajes gráficos para la representación de los procesos y los datos involucrados.

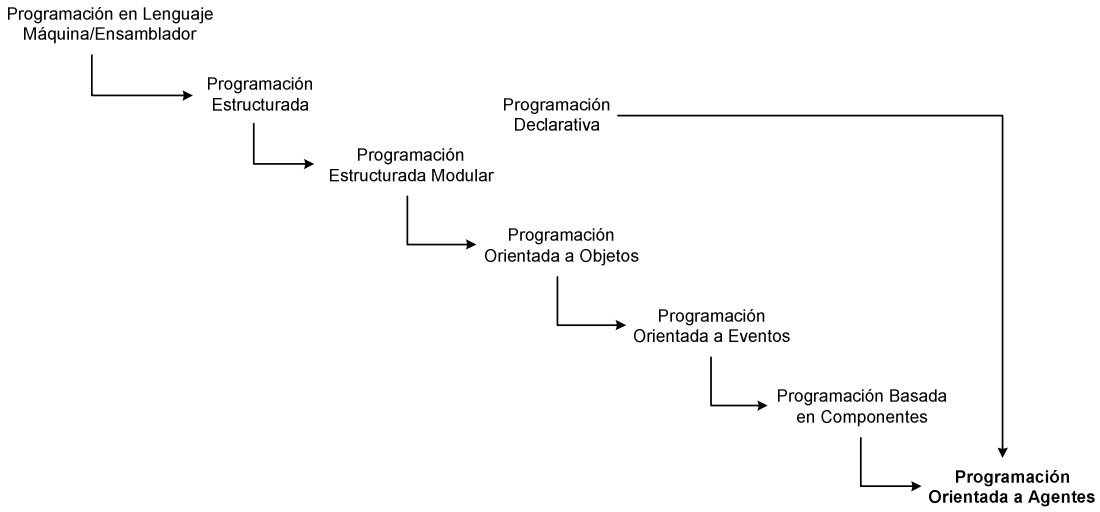


Figura 3.1. Evolución de las técnicas de programación.

Así, a la *Programación Estructurada Modular*, se le asociaron técnicas y metodologías para el desarrollo de aplicaciones, basadas fuertemente en el modelamiento de procesos y datos. Son conocidos en este ámbito los diagramas de flujo y los DFD (Diagramas de Flujo de Datos). Con el nacimiento del concepto de <<clase>> nace la *Orientación a Objetos*, y con esta, el surgimiento de un amplio catastro de tecnologías y metodologías asociadas, como el RUP (The UML: User Guide, 1999) y el lenguaje de modelamiento unificado UML. Más adelante, con la llegada de los llamados lenguajes de cuarta generación, con aplicaciones que corren en sistemas operativos gráficos, la O*rientación a Eventos* tomó forma. Finalmente, la tecnología que actualmente está en boga, están los sistemas *Basados en Componentes*. Por otra línea evolutiva se desarrolló la

P*rogramación Declarativa*, siendo quizás los exponentes más conocidos de ésta, el lenguaje de consulta de bases de datos SQL y un lenguaje de implementación de sistemas expertos basados en la lógica de predicados conocido como Prolog.

Todas estas tecnologías han venido a desembocar hoy en día en la conocida Programación Orientada a Agentes, donde ya no se limita la concepción a unidades individuales reactivas con objetivos específicos, si no que a la conceptualización de unidades muy complejas, autónomas, capaces de realizar varias tareas a la vez, capaces de enfrentar sucesos imprevistos y comportarse en forma no siempre predecible, dotadas muchas veces de cierto grado de inteligencia. Estas unidades ya no son simples objetos o componentes, si no que se transforman en sistemas completos, capaces de interactuar con otras unidades. Con este nuevo paradigma las metodologías tradicionales (Pressman, 2002) fallan. Nacen entonces, como ocurrió con todos los anteriores, las metodologías Orientadas a Agentes (O.A.)

El concepto de metodologías Orientadas a Agentes ya está, sin duda, en el vocabulario de gran parte de la comunidad científica que trabajan en el área. Numerosas publicaciones se han realizado al respecto. [Demazeau, 1995], [Glaser, 1997], [Iglesias, 1998], [Wooldridge et al, 2000], y [Gómez, 2002], entre otros, proponen una serie de pasos modelos para llevar a cabo el proceso de construir sistemas basados en la IAD, todos apoyándose en la idea de que las metodologías puramente Orientadas a Objetos (O.O.) no satisfacen por completo los requerimientos de un sistema multia-gente.

3.1.1 Modelamiento Orientado a Objetos

Sin duda que la orientación a objetos ha tenido un fuerte impacto en la comunidad del desarrollo de software. Ésta tiene considerables ventajas; facilita la incorporación de cambios ocurridos en los objetos físicos representados en el modelo, facilita el logro de altos niveles de abstracción y encapsulamiento y facilita la reutilización, entre otras (Morales, 2003). La esencia del desarrollo orientado a objetos es la identificación y

organización de los conceptos de uso y dominio, más que su representación final en un lenguaje de programación, orientado a objetos o no (Oriented-Object Modeling and Design, 1991).

El Lenguaje Unificado de Modelamiento, UML (Unified Modeling Language), se ha convertido en una de las herramientas más utilizadas para el diseño de aplicaciones O.O., y se han desarrollado diversas metodologías en torno a éste. Básicamente, UML es un lenguaje (conjunto de notaciones) de apoyo a la visualización, especificación, construcción y documentación de un sistema (The UML: User Guide, 1999), no representa un método de desarrollo por sí sólo.

En la problemática del modelamiento de sistemas auto-organizados, UML y la orientación a objetos, han tenido gran acogida por parte de un conjunto de investigadores que han utilizado sus herramientas para resolver ciertos problemas. Tal es el caso de Ramos *et al* (1997), Ramos (1998), y recientemente Choi *et al* (2003) y Koller (2004), entre otros.

Sin duda que los conceptos de objeto y agente pueden parecer similares a la hora de modelar e implementar un sistema. Agentes y objetos, por igual, encapsulan (ocultan) su estado y su comportamiento, y se comunican vía paso de mensajes (Wooldridge, 1997). Sin embargo, agentes y objetos tienen grandes diferencias, y las técnicas O.O. como las que utilizan UML no son suficientemente eficaces. Los objetos sólo son iguales a los agentes en el caso de los agentes puramente reactivos.

Jennings y Bussmann (2003) identifican tres importantes diferencias entre los objetos y los agentes:

i. Primero, *los objetos por naturaleza son generalmente pasivos*. En efecto, un objeto se mantiene esencialmente estático en la espera de un mensaje que active alguno de sus métodos.

ii. Segundo, *los objetos encapsulan el estado de su comportamiento* (la ejecución de sus métodos)*, pero no encapsulan la activación de estos comportamientos*. Cualquier objeto puede invocar alguno de los métodos públicos de otro objeto.

iii. Tercero, *la orientación a objetos falla a la hora de proveer un set de conceptos y herramientas adecuadas para modelar sistemas complejos*. La orientación a objetos representa con excesivo detalle el comportamiento, y la invocación de métodos es un mecanismo muy primitivo para describir los tipos de interacción que se llevan a cabo.

Otro factor importante es que los agentes deben operar paralelamente, y debe ser capaz de realizar varias tareas a la vez. En cambio, en una aplicación O.O. las acciones se realizan en forma secuencial.

Un agente es un sistema de toma de decisiones racional, estos deben ser capaces de tener un comportamiento reactivo y pro-activo (Wooldridge, 1997).

Con esto, algunos investigadores han visto la necesidad de idear nuevas formas de representación de los agentes.

3.1.2 Metodologías Orientadas a Agentes

Gómez (2002 y 2003), expone en su trabajo una selección de las metodologías orientadas a agentes más relevantes de los últimos años. Su lista abarca las metodologías Vowel Engineering (Demazeau, 1995), BDI (Kinny *et al*, 1996), MAS-CommonKADS (Iglesias, 1998), GAIA (Wooldridge *et al*, 2000), INGENIAS (Gómez, 2002), MaSE y ZEUS, de las cuales sólo se describirán brevemente las cuatro primeras por ser, a juicio del autor del presente trabajo, las más interesantes y llevaderas a la práctica.

3.1.2.1 Vowel Engineering.

Esta metodología plasma los esfuerzos del grupo MAGMA [19] por idear una formalización del proceso de desarrollo de sistemas multi-agente. La metodología Vowel Engineering es descrita ampliamente por Demazeau (1995). Demazeau explica que en Vowel Engineering un sistema multi-agente es la conglomeración de cuatro conceptos identificados por las cuatro primeras vocales (de ahí el nombre de Vowel: vocal): Agentes (A), Entornos (E), Interacciones (I) y Organizaciones (O).

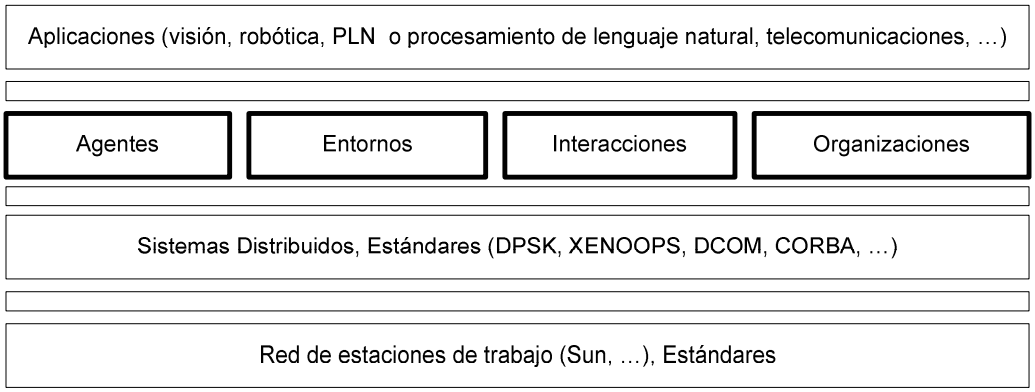


Figura 3.2. Arquitectura de capas de Vowel Engineering (Demazeau, 1995).

Básicamente esta metodología consiste en la selección de los modelos que describen sus conceptos base: un modelo de agente, un modelo de entorno, un modelo de interacciones y modelos organización (Figura 3.). El ordenamiento de estos 4 modelos da como resultado el tipo de sistema a implementar (un sistema altamente interactivo, como sería comenzar con la letra I, un sistema basado en la emergencia del comportamiento global en base a agentes aislados, como sería comenzar con la letra A, etc.).

La gran falencia de Vowel está en que la forma de realizar los modelos dados queda abierta; no especifica nada más que las vocales. Gómez (2003) agrega que la forma de desarrollo de Vowel facilita la reutilización. Como sea esta es una de las primeras metodologías en modelar sistemas a partir de la combinación de diferentes aspectos.

[19] Más información del grupo MAGMA en http://www-leibniz.imag.fr/MAGMA/

3.1.2.2 BDI

En BDI (Kinny *et al*, 1996) se realiza una representación del mundo de cómo se les presenta el entorno a los agentes. Los agentes son estimulados por diversos sensores que se encuentran presentes en su mundo, y estos estímulos generan un cambio en la perspectiva que los agentes tienen del medio. Para BDI, los agentes son guiados por sus propios deseos.

BDI ofrece un conjunto de modelos, los cuales se identifican desde un punto de vista externo y otro interno. Desde un punto de vista externo se detallan dos modelos: un modelo de agente y un modelo de interacción. El primero muestra la identificación de los agentes y su estructura jerárquica, identificando además las instancias para cada tipo de agente, su multiplicidad, etc. El segundo muestra los servicios que provee cada agente, las asociaciones de interacción y las relaciones de control existentes.

En un punto de vista interno se detalla un modelo de creencias, un modelo de metas, y un modelo de planificación.

BDI es un de las metodologías más respetables, por cuanto presenta un conjunto de modelos dependientes de gran utilidad. Es una de las metodologías más formales que se pueden encontrar.

3.1.2.3 MAS-CommonKADS

MAS-CommonKADS (Iglesias, 1998) es el resultado de la adaptación del método CommonKADS[20], que se utilizaba para el modelado de agentes individuales. MAS-CommonKADS, al igual que CoMoMAS (Glaser, 1997), es una metodología orientada fuertemente a modelar sistemas multi-agentes basados en el conocimiento.

Esta metodología ha innova con respecto a las demás, ya que incorpora por primera vez la integración de un método orientado a agentes con un ciclo de vida de software. Iglesias (1998) indica que MAS-CommonKADS puede ser utilizado para proyectos pequeños con un ciclo en cascada y para proyectos grandes con un espiral dirigido por

[20] La sigla MAS viene de Multi-Agent System (Sistema Multi-Agente).

riesgos (Presuman, 2002), básicamente siguiendo las siguientes fases de desarrollo: conceptuación, análisis, diseño, codificación y pruebas, integración, y operación y manteniemiento.

En MAS-CommonKADS se identifican siete modelos fundamentales:

i. Modelo de Agente. Especifica las características de un agente (capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupo al que pertenece y clase de agente).

ii. Modelo de Organización. Describe la organización de los agentes y su relación con el entorno.

iii. Modelo de Tareas. Describe las tareas que los agentes pueden realizar.

iv. Modelo de Experiencia. Describe el conocimiento necesitado por cada agente para el logro de sus objetivos.

v. Modelo de Comunicación. Describe la interacción entre un usuario y un agente de software.

vi. Modelo de Coordinación. Describe las interacciones entre los agentes de software.

vii. Modelo de Diseño. Utilizado para describir la arquitectura y diseño del sistema antes de la implementación.

MAS-CommonKADS propone una notación gráfica que es una ampliación de las de UML. Identifica básicamente dos tipos de agente: un agente humano y un agente máquina.

3.1.2.4 INGENIAS

Con la misma tendencia de MAS-CommonKADS de elaborar metodologías de desarrollo de sistemas multi-agente detalladas, pero en el bando de la ingeniería de software, se encuentra INGENIAS (Gómez, 2002).

INGENIAS incorpora cinco metamodelos. Estos son:

i. Meta-Modelo de Agente.

ii. Meta-Modelo de Interacción.

iii. Meta-Modelo de Objetivos y Tareas.

iv. Meta-Modelo de Organización.

v. Meta-Modelo de Entorno.

Esta metodología es integrada por Gómez (2002) al proceso unificado de desarrollo RUP (The Unified Modeling Language, 1999). Además, Gómez incorpora una práctica notación para la generación de sus metamodelos.

Uno de los aspectos fundamentales a destacar, es que además cuenta con una herramienta de software de apoyo: el INGENIAS IDE, la que se puede descargar de http://ingenias.sourceforge.net.

3.1.2.5 GAIA

GAIA (Wooldridge *et al*, 2000) es una metodología que se enfoca en la concepción de sistemas multi-agentes que tengan por objetivo la maximización de una o más medidas de calidad globales. Plantea ofrecer al analista una forma de ir sistemáticamente desde los requisitos iniciales hasta un diseño que sea lo suficientemente detallado como para ser implementado directamente. La fase de captura de requerimiento ha sido independizada de los paradigmas utilizados para el análisis y el diseño.

El analista parte con la captura de requerimientos, una vez especificados se puede ir pasando a niveles de especificidad mayor. Los modelos que GAIA propone aparecen en la Figura 3.3.

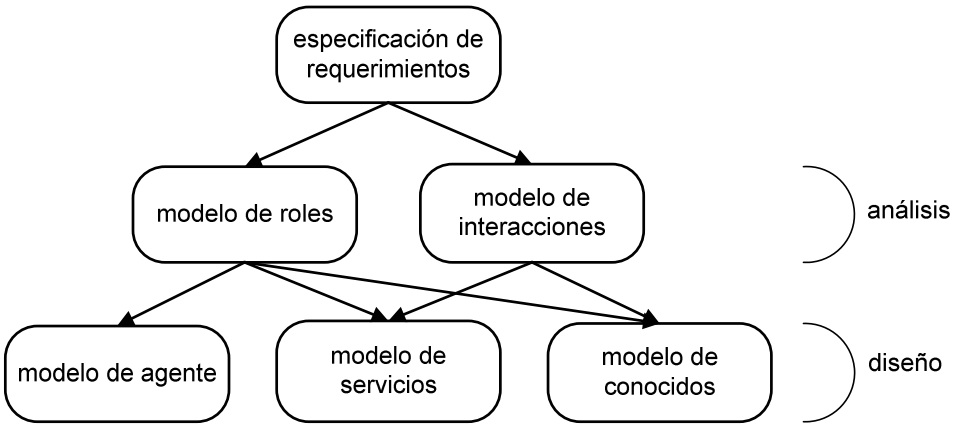


Figura 3.3. Relaciones entre los modelos de GAIA.

GAIA pone especial énfasis en la identificación de roles, más que en la de los mismos agentes. Bussmann *et al* (2001) defiende esta postura, agregando que una metodología para el desarrollo de sistemas multi-agente debe estar orientada a definir el comportamiento del sistema y de los agentes, y que modelar los roles da plena libertad para asignarlos y reasignarlos entre los agentes.

Lamentablemente la metodología no completa el ciclo de desarrollo del sistema. Se queda en la etapa de diseño del sistema. Además, la falta de una herramienta informática de apoyo al modelamiento dificulta un poco las cosas a la hora de adoptar esta metodología.

3.2 Seleccionando una metodología

Del universo de posibilidades de metodologías existentes se han seleccionado sólo cuatro para su descripción, por ser las que han dado pie a una mayor cantidad de desarrollos y las que mejor elaboradas se encuentran.

Claramente aquí se distinguen dos líneas de orientación: una basada en la Ingeniería del Conocimiento, como en el caso de MAS-CommonKADS y otras basadas en la Ingeniería del Software.

En todas se cuenta con experiencias en el desarrollo de aplicaciones en los más diversos ámbitos. Por ejemplo, los trabajos realizados por Arenas y Barrera-Sanabria (2002), en MAS-CommonKADS, y el de Davidsson y Wernstedt (2002) en GAIA.

Sin embargo, la complejidad de los modelos que proponen MAS-CommonKADS e INGENIAS dificultan en demasía su entendimiento. Además, en el caso de INGENIAS, se alarga demasiado el proceso de desarrollo (aunque es bastante exhaustivo en su propuesta).

Dadas las restricciones del presente trabajo, sobre todo en cuanto a tiempo y extensión se refieren, la simplicidad de GAIA hubiese sido la carta ideal, pero la incompletitud de esta metodología deja mucho que desear.

Por otra parte se debe pensar en la facilidad del entendimiento de los modelos que el presente proyecto arroje.

Recientemente se están haciendo esfuerzos por unificar la simplicidad y consistencia que entrega GAIA con los beneficios del modelamiento con UML. En el trabajo de García-Ojeda *et al* (2002; 2004) se realiza una extensión de la metodología GAIA con el lenguaje Agent-UML (AUML); una extensión del lenguaje unificado de modelamiento para aplicarlo en sistemas con agentes (Odell *et al*, 2000a y 2000b). Esta propuesta es resumida en la Figura 3.4.

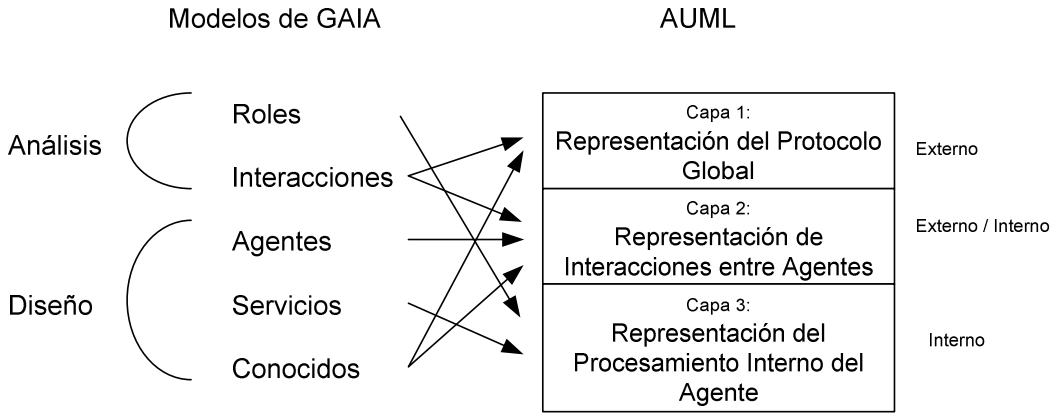


Figura 3.4. Relaciones entre GAIA y el modelamiento con AUML.

Esta figura fue extraída del trabajo de Arenas *et al* (2004), la cual es reformulada en García-Ojeda *et al* (2004) incluyendo explícitamente la utilización de la propuesta de Bernhard Bauer (Bauer, 2001), para la concepción de diagramas de clases utilizados en el contexto de los sistemas multiagente, y un complemento del modelo utilizando la extensión del modelo AALAADIN propuesta por Parunak y Odell (2002). En la figura se aprecian las relaciones existentes entre la metodología de GAIA y el modelo de 3 niveles (capas) de AUML[21] (Odell *et al*, 2000a).

3.3 Metodología de Desarrollo

La metodología a utilizar será la extensión de GAIA y AUML (Arenas *et al*, 2002; García-Ojeda, 2002 y 2004). Con esto el trabajo presentará cuatro grandes fases fundamentales: especificación de requerimientos, análisis del sistema, diseño del sistema y validación de la propuesta; las que a continuación se detallan.

3.3.1 Especificación de Requerimientos

En la primera etapa del desarrollo del proyecto se realizará la toma de requerimientos del sistema. Se debe saber de antemano cuáles son las expectativas de los usuarios para con un sistema de control como el que se está abordando en el presente trabajo; los atributos que deseables para ellos. Gran parte de éste tópico es inducible por la amplia bibliografía existente hoy en día. Sin embargo, a parte, la toma de requerimientos se realizará a través de la realización de entrevistas a distintos académicos de la Universidad del Bío-Bío, peritos en el área de la automatización de la producción y automatización industrial. El caso de estudio y lugar donde se realizará la validación del sistema es el Laboratorio de Sistemas Automatizados de Producción CIMUBB.

[21] Cabe destacar que AUML obtiene todas las virtudes del modelamiento con su padre, UML. Sin embargo, los aportes realizados a este conjunto de notaciones son de gran valor a la hora de modelar sistemas con agentes.

Los pasos a seguir en esta etapa son los siguientes:

i. Entrevistas a académicos de la Universidad del Bío-Bío para capturar los requerimientos del sistema. Además se realizará una charla con el actual Ingeniero de Soporte del Laboratorio CIMUBB, quien juega el papel de usuario directo del sistema.

ii. Establecimiento de los requerimientos del sistema mediante casos de uso.

iii. Estudio de la arquitectura existente en el Lab. CIMUBB, y de otros organismos.

3.3.2 Análisis

Luego de la especificación de los requerimientos del sistema, se debe establecer la forma en la que va a operar este. Esta etapa tiene por objetivo indicar cómo va a operar el sistema, qué protocolos se llevarán a cabo para realizar las distintas tareas y qué roles principales se identifican en este.

i. Identificación de los principales roles involucrados en los sistemas automatizados de producción (tomando como marco de referencia la arquitectura del CIMUBB).

ii. Realización de los modelos de roles y de interacciones de GAIA (Wooldridge *et al* ,2000; Zambonelli *et al*, 2003). En este punto, particularmente al modelar las interacciones entre los roles involucrados, se determinará el protocolo de interacción seleccionado para utilizar en el sistema.

3.3.3 Diseño del Sistema

En esta etapa se realizará el modelamiento general del sistema, y se irá detallando hasta llegar a un nivel suficiente para pasar a la siguiente etapa (como propone GAIA).

Esta etapa comprende las siguientes etapas:

i. Realización del modelo de agentes. Donde se identifican los agentes que conformarán el sistema multi-agente y se le asocia un conjunto de roles a cada uno que deben cumplir. Además, se pretende evaluar la factibilidad de utilizar los modelos de agentes tipo que propone Parunak *et al* (1998).

ii. Realización del modelo de servicios. Donde se expone las tareas que puede realizar cada agente, y qué necesita para que dicha tarea sea realizada exitosamente.

iii. Realización del modelo de conocidos. Donde se detallan los agentes con los que puede interactuar un agente.

iv. Complementar con diagramas de secuencia, de colaboración, de actividad y de estados de AUML.

3.3.4 Validación de la Propuesta

Esta corresponde a la prueba de los modelos con la implementación de la propuesta en algún lenguaje de programación o herramienta de simulación.

La etapa comprende:

i. Simulación del comportamiento entre dos agentes. Identificar alguna herramienta de simulación que sea capaz de emular el comportamiento de dos agentes modelados en la etapa de diseño. El objetivo es obtener una primera idea acerca del futuro desempeño del modelo. Este paso será opcional en caso de no encontrar alguna herramienta que sea capaz de emular el comportamiento de los agentes.

ii. Construcción de los agentes en Lenguaje en Programación. Se procederá a la codificación de los agentes que operarán el CIMUBB, así como el método de mensajería seleccionado, ya sea por paso de mensajes o por pizarrón, o ambos. Antes se realizará la determinación de los detalles de bajo nivel implicados en la programación, como la arquitectura de comunicaciones a utilizar (CORBA,

simplemente sockets, etc.), sistema(s) operativo(s) (Windows, Linux, etc.), motores de base de datos, etc. Además, se debe ver la factibilidad de la opción seleccionada.

iii. Análisis de los resultados obtenidos.

Capítulo 4

Especificación de Requerimientos. Caso de Estudio CIMUBB.

Luego de una breve presentación de los contenidos teóricos en que se sustenta el presente trabajo de investigación, al fin comienza el desarrollo del proyecto propiamente tal.

Se ha mencionado que el objetivo principal del trabajo es el diseño de un sistema basado en el paradigma de la Inteligencia Artificial Distribuida, particularmente un sistema multi-agente, capaz de resolver la problemática de la programación de la producción, y realizar el control en tiempo real de un sistema de manufactura flexible. El proyecto obtendrá como resultado un modelo de carácter general, capaz de adaptarse a una arquitectura de implementación cualquiera (por lo menos dentro de un rango razonablemente amplio), para luego abocarse a la aplicación práctica de la propuesta en un sistema real. En el caso particular de este trabajo, es de especial interés la evaluación de la propuesta en el Laboratorio de Sistemas Automatizados de Producción CIMUBB, perteneciente a la Facultad de Ingeniería de la Universidad del Bío-Bío.

En presente capítulo se desarrolla la primera fase de la metodología propuesta; la definición y análisis de los requerimientos necesarios para conceptuar el sistema. Se expondrán los principales requerimientos recopilados, deseados para este tipo de sistema, lo que reforzará la idea de utilizar una estrategia de control descentralizada para la administración de la producción, para luego, comenzar de lleno en el modelamiento del sistema, partiendo por la identificación de los roles involucrados en el sistema productivo. Con este objetivo, se realizará un análisis de algunas configuraciones de sistemas

automatizados de producción, identificando un espectro de tecnologías utilizados lo más amplio posible, para luego, realizar un estudio completo de la actual arquitectura de hardware y software del laboratorio CIMUBB, lugar donde se pretende validar la propuesta en curso.

4.1 Especificación de Requerimientos del Sistema

La identificación de los requerimientos es uno de los pasos más importantes de un ciclo de desarrollo de software. Prácticamente una mala definición de éstos puede hacer que todo el desarrollo posterior sea resuelto con un enfoque incorrecto y, por lo tanto, que el resultante sistema no satisfaga, en gran parte, al cliente.

Así, la especificación de los requerimientos del sistema es también la primera fase de la metodología GAIA (que se aplicará en el presente trabajo).

Para la empresa manufacturera del presente, los requisitos de un sistema de fabricación son varios, entre estos, debe asegurar una calidad que le permita competitividad en el mercado, adaptarse rápidamente a un nuevo producto, etc. Sin embargo, gran parte de estos requisitos ya han sido solucionados (o por lo menos encontrar una nueva solución no es el objetivo del presente trabajo) mediante la implementación de sistemas flexibles de manufactura, dotados de maquinarias de alta precisión y por la automatización y aumento de la rigurosidad en el control de calidad de los productos. Los avances en las nuevas tecnologías que integran hardware y programas en dispositivos de bajo nivel, y la posibilidad de enlazar estos últimos con computadoras personales, en las que se pueden utilizar software de diseño y administración de altísimo nivel, resuelven en gran medida los requerimientos de flexibilidad y calidad antes mencionados. Se podría decir, a ojos del presente trabajo, que las operaciones locales ya son óptimas. El problema ahora es que la estrategia de control prevaleciente en el mercado no satisface los requisitos de la industria moderna.

Desde el punto de vista de un sistema de administración y control de dispositivos a nivel global (la verdadera problemática que se debe tratar en este proyecto), se debe garantizar el cumplimiento de los principales requerimientos de una organización de estas características en este aspecto, principalmente en lo que ha robustez se refiere.

4.1.1 Requisitos de las Empresas Manufactureras de la Nueva Generación.

Botti y Giret (2003) definen 11 requisitos fundamentales para las empresas manufactureras de la "*nueva generación*", entre estas, la integración de empresas, la cooperación, la integración de personas con software y hardware, y la agilidad (que como se vio anteriormente está siendo resuelta por los SFM).

Desde el punto de vista de la manufactura integrada, de los sistemas que controlan la producción a nivel de campo, los requisitos fundamentales en este sentido serían seis: la escalabilidad, la reconfiguración dinámica, distribución del conocimiento, entornos heterogéneos, interoperabilidad y la tolerancia a los fallos.

i. **Escalabilidad.** La capacidad de incorporar al sistema nuevos dispositivos físicos sin romper los enlaces organizacionales previamente establecidos es conocida como *escalabilidad*. Esto le brinda al sistema la flexibilidad suficiente para adecuarse a nuevos requisitos de producción de la empresa. Lo mismo ocurre al querer eliminar uno de los dispositivos (ya sea por caducidad o por mantención del mismo). En definitiva, la escalabilidad consiste en contar con mecanismos que permitan *proporcionar la integración de nuevos componentes o la eliminación de los mismos* sin modificar la estructura general del sistema global.

ii. **Reconfiguración dinámica.** Aunque en [Botti and Giret, 2003] se enfoca este tópico a la reconfiguración de los procesos organizacionales ante cambios en el entorno (ya sea internos o externos) que afectan la planificación, es posible aplicar esto a la reconfiguración dinámica de los sistemas de control y monitoreo de la fabricación. Los sistemas manufactureros actuales, al igual que los rígidos procedimientos preestablecidos a nivel organizacional, dificultan en demasía la adecuación del sistema ante cambios en su configuración o sucesos imprevistos. Si bien la escalabilidad, es decir, la posibilidad de ir modificando incrementalmente el sistema sin afectar los enlaces previamente establecidos, está presente en gran parte de los sistemas de manufactura (principalmente en fábricas que producción no continua), los estrechos vínculos en cuanto a comunicaciones y dependencias de las

unidades hacen complicada la reconfiguración. En muchos casos esto se debe a la tendencia de los fabricantes por construir sistemas especializados, donde prácticamente son necesarios varios equipos con software y hardware de la misma firma. Sin embargo, uno de los principales problemas nace a causa de la preponderante estrategia de administración centralizada. Con esto, una reconfiguración de la cadena productiva acarrea generalmente una reestructuración del modelo virtual con el que se generan los planes de producción, lo que se convierte en un proceso empalagoso y que requiere muchas veces un tiempo considerable.

iii. **Capitalización y Distribución del Conocimiento.** Antiguamente los sistemas de control de las organizaciones se basaban en la utilización de grandes bases de datos corporativas centralizadas, que almacenaban los datos provenientes de todas las áreas de la empresa. De hecho, actualmente la mayoría de los softwares aplican esta ideología. En efecto, la centralización de los datos en potentísimos computadores facilita enormemente el procesamiento de los datos y la programación de las distintas tareas. El concepto de CIM (sección 1.1.4) representa muy bien esta tendencia. En términos de control de la producción (a un nivel de floor shop) esto se traduce en la implementación de sistemas altamente dependientes de una unidad central de programación de tareas y gerencia de órdenes hacia dispositivos subordinados. Los beneficios y perjuicios de esta estrategia ya han sido mencionados (sección 2.1). La tendencia es hoy hacia la distribución del conocimiento, para agilizar y modularizar el procesamiento de los datos, al realizarlo de forma concurrente.

iv. **Entornos Heterogéneos e Interoperabilidad.** Por su estrecha relación estos dos puntos se revisarán en conjunto. Al complejizarse la configuración de los sistemas de manufactura, es necesario que estos se adapten de forma eficaz a ambientes con hardware y software heterogéneos. Cada vez más, se hará necesario que los sistemas de control operen haciendo uso de aplicaciones escritas en distintos lenguajes y en distintas plataformas.

v. **Tolerancia a Fallos.** En los tiempos que vivimos donde la competencia es fuerte y el tiempo se traduce en dinero, es crítico que el sistema sea capaz de sobreponerse a fallos en cualquier nivel, ya sea de control o de procesos. Se debe contar con mecanismos que detecten estos fallos y los reporten en forma oportuna. El propio programa de producción debe ser tolerante a fallos. Es muy frecuente en los sistemas que actualmente funcionan planificando y controlando la manufactura, ya sea en forma automática o semi-automática, que en el momento en que falla uno de los dispositivos de software o de hardware, el sistema completo se ve bloqueado, teniendo que reconfigurarse manualmente para seguir ejecutando las operaciones correspondientes. Esta reconfiguración es, por lo general, demorosa, teniéndose muchas veces que volver a levantar todo el sistema y a realizar el plan de producción nuevamente.

4.1.2 Requerimientos de Desarrollo del Sistema.

Los investigadores Botti y Giret (2003) identificaron cinco requisitos deseables en el desarrollo de todo sistema de control de la manufactura para las empresas de fabricación del presente. Estos requisitos aparecen clasificados en 2 categorías: requisitos funcionales y requisitos de ingeniería de software.

a) Requisitos Funcionales

Requisito 1: *"Los sistemas de control de fabricación requieren agentes semi-autónomos. Los agentes deben razonar sobre el comportamiento del sistema de fabricación, pero no sobre sus propias actitudes mentales o aquellas de otras unidades de control*[22]*"*.

Requisito 2: *"Las unidades de control de fabricación principalmente requieren de un comportamiento basado en rutinas que es al mismo tiempo efectivo y oportuno (timely). Este comportamiento puede ser tanto configurable o auto-adaptativo"*.

[22] Esto último no excluye necesariamente la idea de que sí puedan hacerlo (que los agentes de los sistema de control puedan razonar sobre sus propias actitudes mentales o de otras unidades).

b) Requisitos de Ingeniería de Software

Requisito 3: *"Los métodos de programación deben proveer encapsulación de datos y procesos"*.

Requisito 4: *"Los programas de control deben tener una semántica clara. Adicionalmente, el comportamiento de un agente debería ser completamente especificado por su programa de control"*.

Requisito 5: *"Un método o metodología de programación debería conducir directamente de una tarea de control a un programa de agente"*.

Los primeros requisitos indican derechamente la necesidad de concebir el sistema como un conjunto de unidades que en conjunto, y valiéndose de protocolos de interacciones, se organicen a sí mismas para realizar el control global del sistema. Los segundos, son requisitos propios de los métodos de desarrollo utilizados en las etapas de desarrollo del sistema a construir. De esta forma Botti y Giret defienden de antemano la idea del desarrollo de un sistema de control de la producción multiagente (o cualquier otro sistema de carácter auto-organizado), y de la utilización de un método como GAIA para el desarrollo del mismo.

4.1.3 Requerimientos del Sistema.

Los requisitos expuestos antes en esta sección del presente documento pretenden ilustrar en cierta forma la situación general que se vive actualmente en las empresas fabriles. Todos estos han sido mencionados explícita o implícitamente antes a lo largo del escrito. Lo que a continuación se presenta son los requisitos más relevantes identificados para un sistema como el que se pretende abordar y que se considerarán en el modelo propuesto.

Junto con la revisión de un conjunto de publicaciones disponibles que hablan al respecto se realizaron algunas entrevistas a funcionarios de la Universidad del Bío-Bío para capturar los requerimientos que estos consideraran relevantes desde varias perspectivas. Entre estos, el Doctor Mario Ramos Maldonado, académico del Departamento de

Ingeniería en Maderas de la Facultad de Ingeniería, indicó que para un empresario, un sistema automatizado de producción debe que cumplir con 2 elementos fundamentales, en consecuencia con la primera prioridad de una organización que debe ser la satisfacción de sus clientes. Según él, un sistema de producción de cualquier tipo debe cumplir con 2 requisitos fundamentales:

> **Requerimiento 1:** *"El sistema debe asegurar la calidad de los productos que en él se fabriquen"*.
>
> **Requerimiento 2:** *"El sistema debe asegurar en lo posible el cumplimiento de las fechas de entrega"*.

Con respecto al primero, ya se ha dicho algo al comienzo de este capítulo. Las sofisticadas tecnologías en hardware, como las máquinas CNC, de gran precisión aseguran (hasta cierto punto) la calidad del trabajo. Sin embargo el modelo propuesto debe asegurar contar con los módulos necesarios para realizar comprobaciones de la calidad de un determinado trabajo según la configuración del sistema existente (como los modernos procesos de control de calidad automatizados existentes hoy en día).

Lo segundo es crucial. El plan de trabajo debe estar resuelto de tal forma que asegure una entrega lo más fiel a lo acordado con los clientes.

Por otra parte, una entrevista realizada al actual Encargado de Soporte del Laboratorio de Sistemas Automatizados de Producción CIMUBB, de la Facultad de Ingeniería, el Ingeniero Sr. Luis Vera Quiroga, reflejó un conjunto de requerimientos desde el punto de vista de un usuario del sistema. Luis Vera lleva algún tiempo a cargo del soporte técnico de este laboratorio, y su experiencia para con el sistema es en conocimiento y operación de todo el hardware involucrado y del ambiente de control Open-CIM (del cuál se hablará más adelante en este capítulo). Según él, un sistema de control de manufactura debería cumplir con las siguientes características:

Requerimiento 3: *"El sistema de control debe ser resistente ante las fallas, es decir, la falla de uno de los dispositivos no debe generar la caida total del sistema (como pasa con el sistema Open-CIM)"*.

Requerimiento 4: *"El software de control debe ser robusto"*.

Requerimiento 5: *"La configuración del sistema, es decir, la agregación o eliminación de unidades hardware en el sistema debe ser simple"*.

El *Requerimiento 3* es fundamental. Después de todo, es deseable que la falla de una pequeña unidad no afecte al sistema en su conjunto, y que pueda seguir operando dentro de lo posible. El sistema debe sobreponerse a las fallas.

El cuarto requerimiento tiene una solución casi obvia, y esta es seguir procedimientos de ingeniería de software para asegurar la optimización del código del sistema. Así como valerse de sistemas de carácter distribuido para evitar la caída por exceso procesamiento de una sola unidad.

Por último es clara la necesidad de evitar largos procesos de reconfiguración del software de control cada vez que es necesario agregar o quitar una unidad de hardware del sistema (como una máquina CN, una nueva unidad de transporte o una cámara de control de calidad).

A parte de esto, se mencionan otros requerimientos que no serán considerados a la hora de diseñar o construir el sistema, como la *amigabilidad* del ambiente.

Con respecto a la funcionalidad del sistema, éste debe ser capaz de controlar el funcionamiento de la fábrica completa ante la orden de un responsable humano al cual se denominará *Encargado de Producción*, previa definición de las órdenes de trabajo.

El caso de uso de la Figura 4.1 muestra al actor Encargado de Producción y la interacción que debería tener para con el sistema. Con esto, el Encargado de Producción define las órdenes de trabajo según las guías de pedido de los clientes en el sistema, para

luego ejecutar el inicio del proceso de manufactura, que se realizará desde este momento en forma automática, interfiriendo sólo, de vez en cuando y según corresponda las políticas y procedimientos de la empresa, para monitorear el funcionamiento del sistema mediante la información arrojada por el sistema.

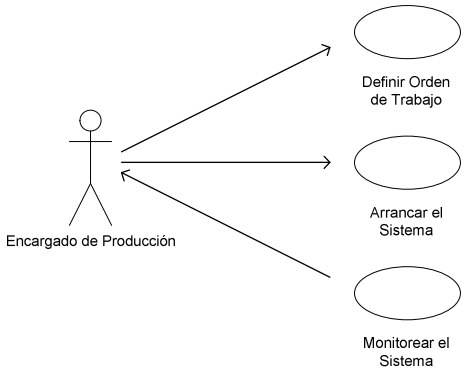


Figura 4.1. Caso de Uso: Ejecución de la Producción.

En el caso de uso de la Figura 4.2 se muestra la actividad de un ente humano conocido (para efectos de este informe) como Ingeniero de Producción, quien se encarga de configurar el sistema cuando proceda, especificando los datos del ambiente de producción, como ubicación y características de las máquinas, rutas de los productos, materias primas disponibles, etc.

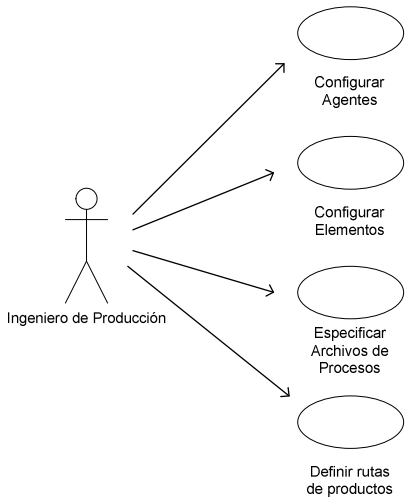


Figura 4.2. Caso de Uso: Configuración del Sistema.

4.2Implementaciones de Estudio. Arquitecturas de Hardware y Software

Como se mencionó en el primer capítulo del presente documento, un SFM es <<*un grupo de estaciones de procesamiento (predominantemente máquinas CNC), interconectadas por medio de un sistema de manipulación y almacenamiento de material automatizado y controlado por un sistema computacional integrado*>>. Apoyados en esta definición, es obvia la identificación de ciertas entidades (posiblemente agentes) que están presentes en todo sistema productivo que pueda ser llamado de esta forma. Entonces, los conceptos de "estaciones de procesamiento", "manipuladores", "almacenes de material" y "sistema de control" se agregan tentativamente a la lista inicial de entes a modelar. Sin embargo, existe la necesidad de ser más específicos, y llegar a un arsenal de actores generalizado que abarque la mayor cantidad de posibilidades. Ya en Parunak *et al* (1998b) se propuso un espectro de los agentes involucrados.

Todas las implementaciones de estudio o industriales encontradas durante el transcurso de este trabajo operan actualmente bajo una lógica de control centralizado jerárquico (que es la que mantiene el laboratorio CIMUBB, del cual se ha hablado en contadas ocasiones y que se adjudica gran parte del presente capítulo). Por ejemplo, la Essex County College, de New Jersey, Estados Unidos, la Galatasaray University, de Istanbul, Turquía, el Institud Universitaire Professionalisé en Brest, Francia, el Centro de Educacao Tecnologica Termomecanica – CETT en Brasil, y, por supuesto, la Universidad del Bío-Bío, en Chile, así como tantos otros cuentan con completísimas instalaciones de estudio para la preparación de sus estudiantes en materias de automatización y robótica (ver Figura 4.3). En [*Fuente: CIMware. A Flexible Manufacturing System (FMS) R&D Project (1976/80). [on line]* http://www.cimwareukandusa.com/FMSdevelopment.htm *[consulta: 25 de julio de 2004]*] es posible ver un proyecto industrial de implementación de las tecnologías que abordamos en este documento.

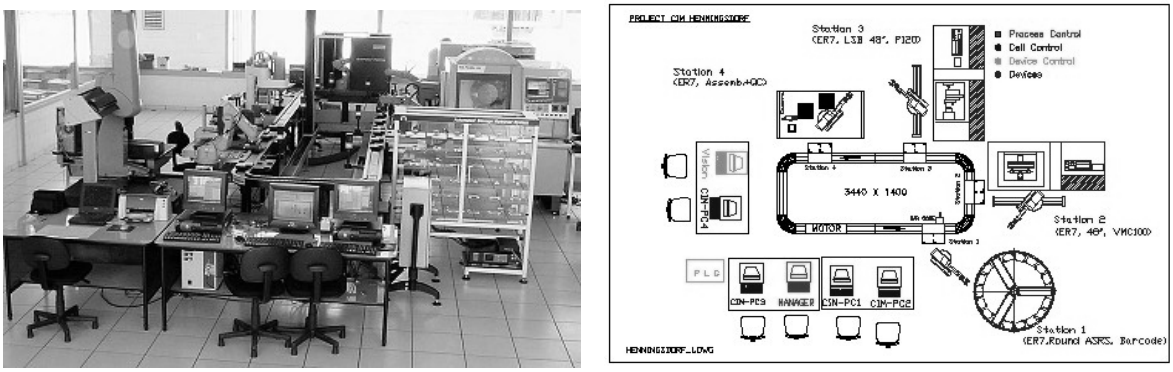


Figura 4.3. .Configuraciones de Estudio. A la izquierda, el laboratorio del Centro de Educacao Tecnologica Termomecanica – CETT de San Bernardo do Campo, Brasil, con 7 células flexibles, y a la derecha, el diagrama esquemático del laboratorio del Oberstufenzentrum Oberhavel II, en Hennigsdirf, Alemania.

Dentro de todas las implementaciones encontradas es fácil categorizar las maquinarias y artefactos que participan de los procesos. Esto será útil más adelante, ya que permitirá definir el modelo de control con agentes de una forma global (o lo más global posible), de tal forma que sea aplicable a un conjunto de ambientes reales. Las categorías de tecnologías en automatización son las siguientes:

i. Manipuladores. Categoría conformada por la gran variedad de brazos mecánicos existentes. Se caracterizan por la capacidad para asir materiales y transportarlos de un lugar a otro, además de realizar tareas de manipulación para realizar ciertos procesos sobre los productos (como soldar, cepillar, etc.).

ii. Unidades de transporte múltiple. En las que podemos encontrar cintas transportadoras, trenes de carga, etc. Todo aquello capaz de transportar continua o discontinuamente varias partes desde varios orígenes a varios destinos.

iii. Máquinas de mecanizado o modificación del material. Principalmente las máquinas de control numérico, que hoy en día se desempeñan en un gran espectro de procesos. Ejemplos de esto son las fresadoras, tornos, etc. Además de otro tipo de máquinas o estaciones automáticas como hornos, secadoras, etc.

iv. Unidades de control y supervisión automatizados. Diferentes unidades como sistemas de control de calidad, lectores de código de barra, medidores de calor, presión y humedad, etc.

v. Almacenes de material y productos terminados. En sus distintos modelos (almacenes matriciales, almacenes de carrusel, feeders, etc.).

4.3 Laboratorio de Sistemas Automatizados de Producción CIMUBB. Estudio de Configuración y Operación.

El Laboratorio de Sistemas Automatizados de Producción CIMUBB (Figura 4.4) funciona en la Universidad del Bío-Bío, bajo el amparo de la Facultad de Ingeniería de esta casa de estudios, desde el año 2000. Fue levantado con el apoyo de la propia facultad antes mencionada, la Universidad, y el Ministerio de Educación de Chile.

Actualmente es utilizado principalmente en el impartimiento de asignaturas de carácter electivo, todas ellas enfocadas en la enseñanza de tecnologías ligadas a la automatización de la producción e industrial. El laboratorio reúne a académicos y estudiantes de niveles superiores de varios departamentos, realizándose en él investigaciones de altísimo nivel. El *grupo* tiene por misión el *"conocer", "desarrollar", "adaptar", "aplicar" tecnologías y soluciones integrales a problemas de la manufactura* (CIMUBB, 2004), todo esto en el quehacer propio de la Universidad, es decir, docencia, investigación y extensión. En este lugar se pretende implementar el trabajo en curso.

Figura 4.4. Fachada exterior Laboratorio CIMUBB.

Hasta la fecha el laboratorio se mantiene operativo con las tres células flexibles con que fue concebido desde su adquisición. Está compuesto por un conjunto de máquinas a escala (máquinas CNC, robots, etc.) que emulan una cadena productiva completa, desde el almacenamiento de materias primas, hasta mecanizado y control de calidad, todo controlado por computadores. A continuación se explica en detalle la composición y estructura organizativa y funcional de los elementos que lo componen.

4.3.1 División celular del CIMUBB. Configuración e Implementación de Células Flexibles.

El laboratorio CIMUBB fue concebido como un sistema flexible de manufactura que responde fielmente al enfoque de integración que acarrea el concepto de CIM (Sección 1.1.4), esto es, todo el sistema productivo está controlado completamente por un sistema computacional centralizado. Este sistema de computación centralizado radica en un computador personal que recibe el nombre de *CIM Manager*.

Funcionalmente los dispositivos con que cuenta el laboratorio CIMUBB han sido organizados en 3 células flexibles[23]: una célula flexible de almacenamiento de material, una célula flexible de mecanizado y una célula flexible de ensamble y control de calidad. Esta organización celular es clásica en los sistemas de manufactura flexibles que imperan actualmente en la industria.

En la Figura 4.5 se expone un esquema del laboratorio CIMUBB indicando la distribución física de las 3 células flexibles, y la ubicación de la mesa de control central[*].

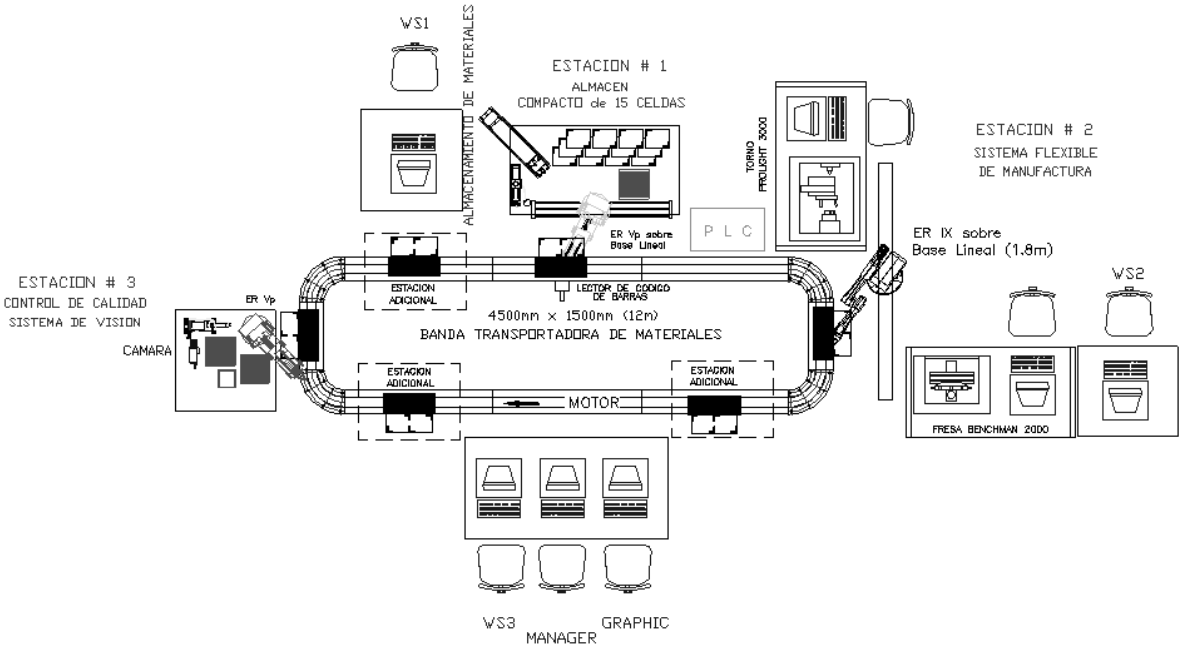


Figura 4.5. Implementación del Laboratorio de Sistemas Automatizados de Producción CIMUBB.

Estas células flexibles, antes mencionadas, y los dispositivos que las componen, aparecen descritas a continuación:

[23] Una célula flexible es una división de máquinas organizadas en un mismo espacio físico, y bajo una dependencia jerárquica común, que trabajan conjuntamente para cumplir uno o varios objetivos afines y comunes y, que en conjunto, presentan cierta autonomía con respecto a las demás células flexibles que pueden conformar el sistema total.

[*] [ESHED ROBOTEC. 2000. OpenCIM Universidad Bío-Bío. [AutoCAD file]. Modelo bidimensional].

4.3.1.1 Célula Flexible de Almacenamiento

Esta célula implementa un almacén de materias primas y templates[24] de transporte. La célula está compuesta por los siguientes elementos físicos:

a. Una estación de trabajo, conocida como Work Station 1 (WS1), que realiza las funciones de control global de la célula sobre los dispositivos de campo subordinados. En esta estación reside el software de control de la célula que se comunica con el computador administrador del sistema completo.

b. Un almacén vertical de 15 celdas (3 niveles con 5 celdas cada uno). Este tipo de almacén es conocido como ASRS (Automated Storage and Retrieval System). Aquí se disponen inicialmente los templates necesarios para el transporte del material.

c. Un robot SCORBOT-ER 5 Plus[25] de 6 grados de libertad[26] (brazo + robot controller). Básicamente este robot cumple la función de cargar los templates disponibles en el ASRS con las diversas partes que puede mecanizar o analizar el sistema, para luego ponerlos sobre la cinta transportadora.

d. Dos feeder (cargadores) de piezas por resorte, uno para almacenar piezas rectangulares para fresado y otro para almacenar piezas cilíndricas para torneado.

e. Un feeder neumático de cilindros para visión artificial.

f. Un lector de códigos de barra (LSM: Laser Sensor Meter), para verificar los códigos impresos sobre las etiquetas de los templates, según la configuración que registra el sistema.

La célula de almacenamiento se muestra en la Figura 4.6.

[24] Un template es una bandeja utilizada por los robots para fijar la posición del material y facilitar su manipulación.
[25] Este modelo fue diseñado para trabajos en laboratorio y aplicaciones de entrenamiento (INTELITEK, 2004).
[26] 5 grados de libertad del brazo más un grado correspondiente al desplazamiento por una base deslizante lineal.

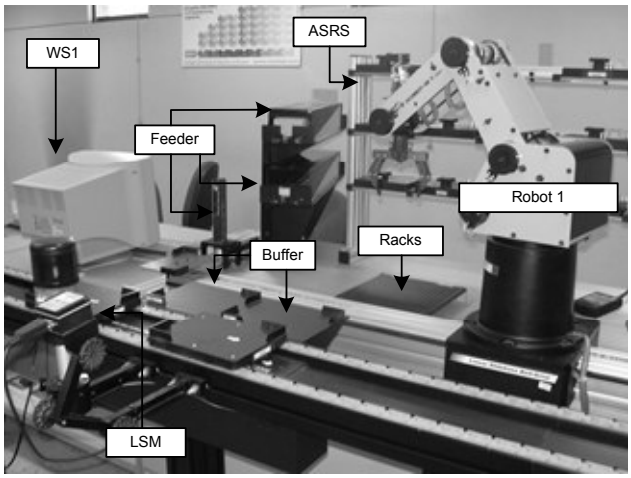


Figura 4.6. Célula Flexible de Almacenamiento.

4.3.1.2 Célula Flexible de Mecanizado

Esta célula posee las máquinas necesarias para el mecanizado o transformación de las partes en algún producto, intermedio o final. Actualmente esta célula cuenta con dos máquinas de control numérico. Los elementos existentes en esta división son los siguientes:

a. Una estación de trabajando conocida como Work Station 2 (WS2), que realiza las funciones de control global de la célula sobre las estaciones de trabajo subordinadas que controlan los dispositivos de campo y el controlador ACL del robot de la estación. En esta estación reside el software de control de la célula que se comunica con el computador administrador del sistema completo.

b. Dos estaciones de trabajo que se comunican directamente con las máquinas CNC, conocidas como Lathe PC (en la que se encuentra instalada una tarjeta del fabricante que controla el torno CNC) y Mill PC (en la que se encuentra instalada la tarjeta que controla la fresadora CNC).

c. Un robot SCORBOT-ER 9[27] de 6 grados de libertad (brazo + robot controller), que cumple la función de recibir los templates, cargados en la célula de almacenamiento, desde la cinta transportadora y depositar los materiales en las máquinas CN para su mecanizado, y luego devolver el trabajo terminado a la cinta.

d. Un torno CNC proLIGHT 3000.

e. Una fresadora CNC de mecanizado BENCHMAN 2000.

La Figura 4.7 expone la disposición física de los dispositivos de la célula flexible de mecanizado.

Figura 4.7. Célula Flexible de Mecanizado.

4.3.1.3 Célula Flexible de Ensamble y Control de Calidad

Una vez realizado el mecanizado de una pieza es deseable comprobar la calidad del producto elaborado. Esta célula está equipada con una cámara profesional con la que, mediante un sistema de visión artificial, se puede detectar la correspondencia de ciertos

[27] Este robot, de alta precisión, está diseñado para trabajos de entrenamiento en instalaciones industriales.

patrones en las piezas recibidas. Los dispositivos de la célula flexible de ensamble y control de calidad son los siguientes:

a. Una estación de trabajando conocida como Work Station 3 (WS3), que realiza las funciones de control global de la célula sobre los dispositivos de campo de la estación. En esta estación reside el software de control de la célula que se comunica con el computador administrador del sistema completo.

b. Un robot SCORBOT-ER 5plus de 5 grados de libertad (brazo + robot controller).

c. Una prensa neumática.

d. Una cámara web profesional.

e. Un compartimiento para depositar las piezas que registren errores (etiquetado como Trash).

En la Figura 4.8 aparecen algunos de los elementos más importantes de la célula flexible de ensamble y control de calidad.

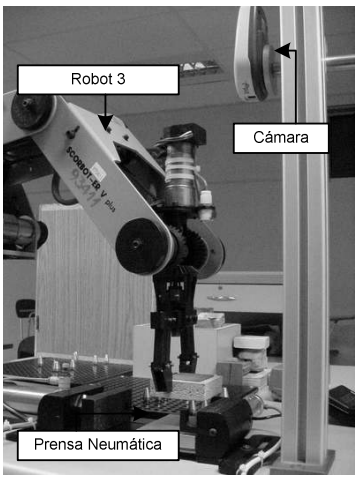


Figura 4.8. Célula Flexible de Ensamble y Control de Calidad.

4.3.1.4 Otros dispositivos

Fuera de las células flexibles antes descritas; de almacenamiento, mecanizado y ensamble y control de calidad, el CIMUBB incorpora otros dispositivos que completan la integración del sistema. Estos dispositivos son:

a. Una cinta transportadora o *Conveyor*, que cumple la función de transportar los palletes cargados con templates desde y hacia las distintas células del sistema; todo controlado por un PLC. Este conveyor cuenta con 6 unidades o estaciones de parada, 3 para las 3 células disponibles, y 3 más para futuras expansiones.

b. Un computador, denominado *CIM-Manager*, en el que reside el software de administración global del sistema.

c. Otros equipos computacionales y hardware de conectividad, como una computadora destinada a trabajo con software de diseño (CAD y CAE), o los switch a los que se conecta toda la red.

4.3.2 División celular y concepto de jerarquización del CIMUBB. Configuración e Implementación de Células Flexibles.

Como se mencionó en la sección 1.1.5 de este documento, el concepto de integración CIM implica jerarquización y centralización en la estrategia de control. Las ventajas y desventajas de esta estrategia ya han sido expuestas en la sección 2.1. Así, en el laboratorio CIMUBB, todo el control del sistema se aloja en una sola estación de trabajo, el CIM-Manager, quien envía órdenes a todas las células subordinadas. Estas células, análogamente a lo que ocurre en el sistema global, también están dominadas por una estación de trabajo, las llamadas Work Station 1, 2 y 3, donde se alojan los softwares de control de los dispositivos de campo de mayor nivel. Además, estos dispositivos de campo programables, como el PLC o los Robots Controller, también ejercen dominio sobre el funcionamiento de otros dispositivos de más bajo nivel.

Esta red jerárquica aparece representada por en la Figura 4.9[28].

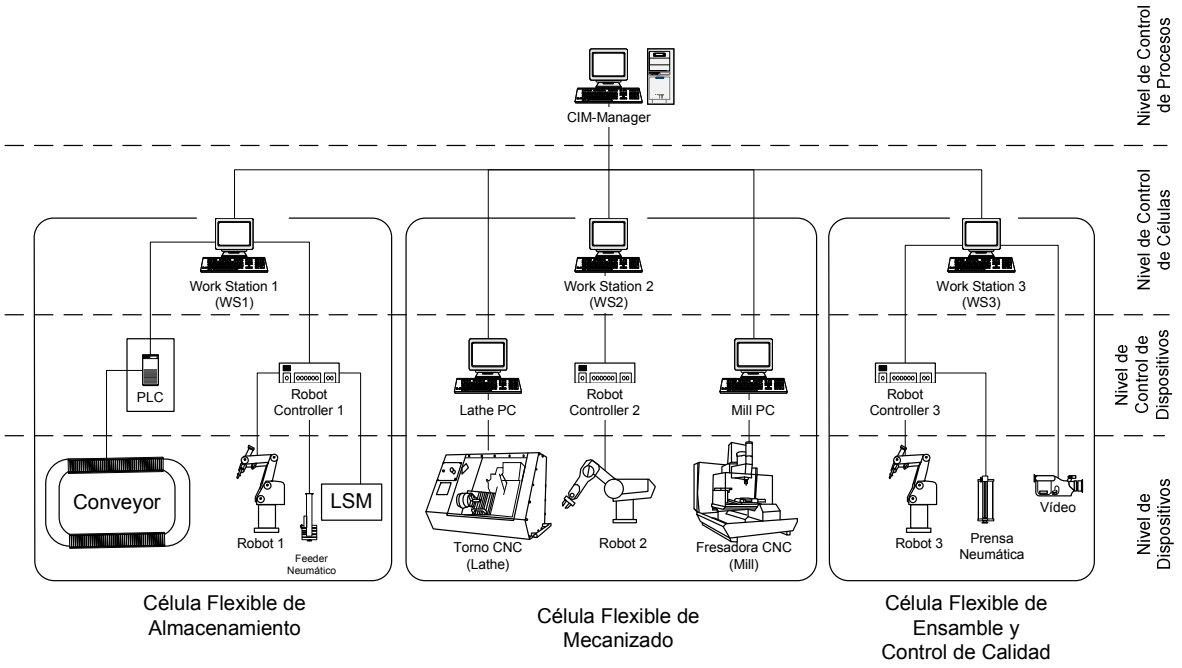


Figura 4.9. Laboratorio de Sistemas Automatizados de Producción CIMUBB.

La Figura 4.9 muestra al PLC (y conveyor) como pertenecientes a la célula flexible de almacenamiento. En estricto rigor esto no es así; se debe considerar a la cinta transportadora como una unidad aparte de toda estación de trabajo, utilizada para el transporte de material de una a otra célula. Fue graficada de esta forma por la relación que existe entre el PLC y la WS1, ya que el software de control y comunicaciones con el primero se encuentra alojado en la estación mencionada.

4.3.3 Arquitectura de Software del CIMUBB: El sistema Open-CIM

La estructura jerárquica del sistema CIMUBB, no se visualiza claramente al observar la arquitectura de hardware existente (a nivel de conectividad); esta

[28] Esta figura expande un poco la presentada por Morales (2003), aunque sigue mostrando solamente la jerarquía identificada en este sistema; en ningún caso pretende representar comunicación física alguna. Esto último será discutido en la sección 4.3.4.

jerarquización se visualiza más claramente al conocer las dependencias existentes entre los software instalados que gobiernan el sistema global.

El Laboratorio de Sistemas Automatizados de Producción CIMUBB opera bajo el esquema de administración del sistema Open-CIM, un software de control centralizado que se ocupa de las tareas de programación, asignación y monitoreo de todas las operaciones que se ejecutan entre los distintos dispositivos involucrados[29].

Morales (2003) realizó un estudio completo de las comunicaciones de alto nivel llevadas a cabo en el laboratorio CIMUBB, y con esto, pudo modelar la secuencia de mensajes TCP/IP (Comer, 1996) y seriales que se realizan durante la ejecución del proceso bajo el esquema Open-CIM. Es vital para la etapa de implementación del presente trabajo conocer a cabalidad la forma en la que opera el actual sistema (ya se explicará esto más adelante en el Capítulo 7: Validación de la Propuesta: Construcción, Implementación y Análisis de Resultados).

Como se visualiza en el esquema de la Figura 4.10[30], el software de control central se encuentra instalado en el computador conocido como CIM-Manager, donde genera el plan de trabajo del sistema en base a las órdenes de manufactura, definición de partes y configuración del almacén existente. Este software se comunica a través de la red Ethernet (Comer, 1996) con unas aplicaciones conocidas como Devices Drivers, las cuales radican en las 3 estaciones de trabajo conocidas como Work Stations 1, 2 y 3 (las que, como se muestra en la Figura 4.9, controlan las células flexibles de manufactura en forma local). Open-CIM genera una carta Gantt de todas las secuencias de operaciones que se deben realizar, para luego, enviar las órdenes respectivas a los Device Drivers para realizar la ejecución de las tareas individuales. Cada Device Driver toma las órdenes provenientes de la unidad administradora y realiza pequeños bloques de tareas en forma local, comunicándose directamente con los controladores de los robots (Robot Controller) y el

[29] Open-CIM, así como el resto de las aplicaciones con las que se comunica (en el nivel de control de células), funciona actualmente en el CIMUBB con la plataforma Microsoft Windows 98.

[30] Este esquema se ha simplificado. Sólo se han incluido los programas que radican en los PC's, y no los programas de los dispositivos de bajo nivel, como los programas Pick and Place escritos en lenguaje ACL (Robot Controller) o el programa controlador del conveyor (PLC).

PLC, a través del puerto serial, los que realizan los comandos Pick and Place (ver ANEXO II) y demás operaciones respectivas[31].

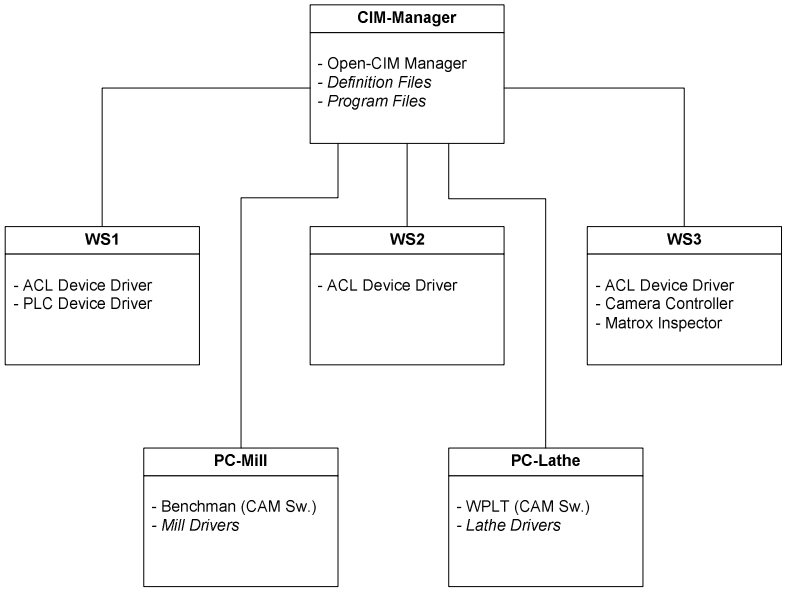


Figura 4.10. Diagrama de software de la arquitectura Open-CIM.

Con el esquema mostrado en la figura anterior, todas las estaciones de trabajo involucradas en el proceso de manufactura *"dependen"*, hasta cierto punto, de la unidad *Manager*. En realidad esto último no es tan estricto. Si bien las estaciones denominadas *"Work Station"* dependen en forma directa de la unidad *Manager* (en el esquema Open-CIM) debido a la comunicación de esta última estación con los Device Drivers (Morales, 2003), es fácil darse cuenta que algunas dependencias no son obligatorias. El ejemplo más claro es la dependencia observada entre el CIM-Manager y las estaciones PC-MILL y PC-LATHE. Esta dependencia está dada únicamente porque los archivos de programas NC de procesamiento de partes están almacenadas en directorios del Manager. Estos programas son utilizados por los software CAM (Computer Aided Manufacturing), Benchman (que controla la fresadora) y WPLT (que controla el torno). Sin embargo, la sola copia de estos archivos en las estaciones PC-MILL y PC-LATHE (junto con la modificación de los programas ACL que contienen la ubicación de los archivos NC) independiza

[31] Considérense aquí lecturas de códigos de barra, control de celdas de detención del conveyor, etc. (aparte de los mencionados Pick and Place, en los cuales se ejecutan rutinas que activan brazos robóticas, jigs y otros).

completamente estas unidades del administrador central (como se verá en la siguiente sección, esto es a causa de que en realidad la comunicación de las estaciones PC-LATHE y PC-MILL es con los Robot Controller, y no con el CIM-Manager, como podría pensarse).

4.3.4 Estudio de Comunicaciones.

En síntesis: el Laboratorio CIMUBB está conformado por 3 células flexibles de manufactura, cada una comandada por una estación de trabajo conocida como Work Station (1, 2 o 3), las cuales comandan a cada célula con las instrucciones provenientes de la estación de control global conocida como CIM-Manager.

El diagrama de la Figura 4.11[**] muestra una síntesis de las comunicaciones físicas existentes en el CIMUBB. Todos los equipos se encuentran conectados en una LAN Ethernet 100 Base T. Por este medio el PC CIM-Manager se comunica con los Device Driver de las Work Stations con mensajes TCP/IP (Comer, 1996).

Por otra parte, las Work Station se comunican con los dispositivos de control de campo, el PLC y los Robot Controller, por medio de comunicación serial utilizando la norma RS232. El PLC ejecuta instrucciones que activan rutinas (ver ANEXO I) de las celdas de detención de palletes (mencionadas antes en la sección 4.3.1.4) y los Robot Controller se encargan de mover los servo-motores de los robot, los jigs y el lector de código de barras. Y he aquí algo bastante singular: contrario a lo que se puede pensar los movimientos de las máquinas CNC son provocadas por el Robot Controller 2, que, por medio de instrucciones ACL ejecuta subrutinas alojadas en las máquinas PC-Mill y PC-Lathe, a través de envío de mensajes por las puertas seriales. Incluso la ejecución de los programas de control numérico es ordenada inicialmente por instrucciones ACL. En efecto, el programa de código numérico (que en el caso del sistema del CIMUBB está etiquetado como un número de 1 a 9 con la extensión *.NC) es seteado por el Robot Controller por medio de variables de entorno denominadas PRGN1 o PRGN2 para la fresadora y el torno respectivamente. Estas variables son tomadas luego por los programas STR1 y STR2 para

[**] Extraida de [ESHED ROBOTEC, 2000].

realizar el procesamiento correspondiente (estos programas, STR1 y STR2, más las variables de entorno fueron definidos por el personal que implementó el laboratorio).

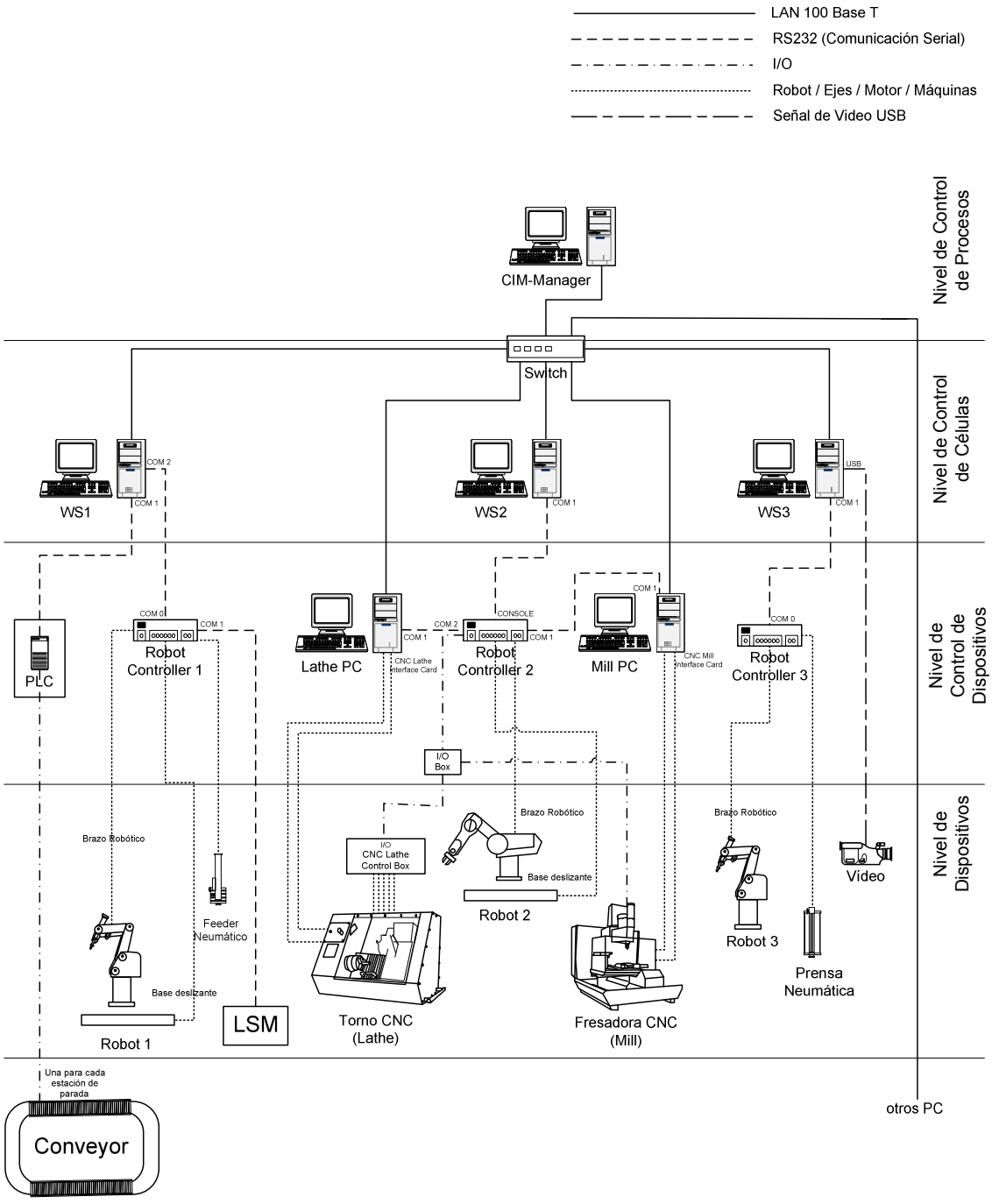


Figura 4.11. Disposición de Comunicaciones del CIMUBB.

STR1 y STR2 son programas escritos en ACL que generan señales a través de comunicación serial hacia las estaciones PC-Mill y PC-Lathe, en las que yace funcionando el software CAM (Computer Arded Manufacturing) Benchman 2000, que controla las máquinas por medio de la tarjeta propietaria instalada en los equipos.

Esto último es crucial para el trabajo[32]. Se puede entonces operar la estación de mecanizado completamente utilizando solamente instrucciones ACL enviadas desde la Work Station 2 al Robot Controller 2 por la puerta serial (que en el momento de la ejecución del presente trabajo se encuentra configurada en el primer puerto de comunicaciones) sin enviar en ningún momento mensajes de alto nivel TCP/IP a las estaciones PC-Mill y PC-Lathe. Con esto, entonces, será factible, en la etapa de implementación, dejar operando los equipos PC-Mill y PC-Lathe tal como funciona el actual esquema Open-CIM, con el mismo sistema operativo Microsoft Windows y el software Benchman, mientras que los agentes podrían operar perfectamente desde el equipo Work Station 3.

Por ejemplo, se puede ejecutar un proceso de fresado (previa deposición de una parte en la máquina) con el código numérico almacenado en el archivo 3.NC enviando al Robot Controller las siguientes instrucciones:

```
> SET PRGN1=3

> RUN STR1
```

Esto explica lo que se mencionó anteriormente: *Existe independencia entre las estaciones PC-Mill y PC-Lathe y el CIM-Manager*.

La configuración de los puertos de comunicaciones para los Robot Controller y el PLC aparece resumida en la siguiente tabla:

[32] El hecho de la ejecución de procesos de mecanizado por instrucciones ACL es crucial, no así la implementación de los programas STR1 y STR2, que fueron implementados por los proveedores del Laboratorio y que pueden ser modificados a voluntad.

Tabla 4.1. Configuración de los puertos seriales.

Item	Robot Controller (1,2 y 3)	PLC
Baud Rate (velocidad de transmisión)	9600	9600
Data Bits (Bits de datos)	8	7
Parity (Paridad)	None (ninguna)	Even (par)
Stop Bits (Bits de parada)	1	2

Capítulo 5

Análisis del Sistema: Identificación de Roles e Interacciones

---

Con el auge de la teoría de agentes, se genera la necesidad de diseñar formas de asegurar el entendimiento y la cooperación entre los actores al momento de ser necesaria la solución de un problema. Al respecto, han sido varias las propuestas de los investigadores, siendo posible identificar protocolos de cooperación, negociación e intermediación entre agentes. Ejemplos de protocolos hay por doquier, y se pueden mencionar, por ejemplo, *Request*, *Query*, *English Auction*, *Brokering*, *Recruiting*, *Subscribe* y *Propose*, entre otros.

Uno de los protocolos que mayor cantidad de experiencias a originado es el protocolo *Contract-Net*. Para el presente trabajo se escogió este protocolo como esquema general para la interacción entre los agentes, por su simplicidad y robustez.

En el capítulo que recién comienza se iniciará la generación de los modelos propios de la metodología GAIA. Se debe recordar que GAIA se caracteriza por, a diferencia de otras metodologías orientadas a agentes, dar mayor importancia inicialmente a la identificación de roles, más que a la de los agentes mismos, característica cuya utilidad se verá reflejada en el presente capítulo.

La etapa de Análisis de GAIA, que se desarrollará en este capítulo, está orientada a conocer cómo funciona (o funcionará) el sistema y su estructura, definiendo principalmente qué roles se pueden identificar en el sistema y qué interacciones son distinguibles entre estos roles. Todo esto sin considerar aún ningún detalle de implementación.

5.1 El Protocolo Contract-Net

El 1983 aparece en la revista Artificial Intelligence un artículo titulado "Negotiation as a metaphor for distributed problem solving" (citada en [Ramos, 1998]). Ahí, los investigadores Davis y Smith propusieron el protocolo contractual, conocido como "red de contratos" o *Contract-Net*.

El protocolo de negociación Contract-Net, que se ilustra en la Figura 5.1, se basa en una idea bastante sencilla. Cuando un agente necesita la ayuda de otro agente o delegar una tarea a uno de estos, existiendo un conjunto de agentes capaces de realizar esta tarea, envía una solicitud al conjunto de postulantes. Cada uno de los agentes que puede realizar la tarea, entonces, envía un valor de *oferta*, que depende del ámbito en el que se esté trabajando, y que corresponde a la aptitud que indica, según el agente oferente, cómo se comportaría el sistema, o cuál sería el beneficio, de serle entregada a él la tarea.

Una vez que todos los postulantes han entregado su oferta al agente demandante, éste último escoge de entre el conjunto de ofrecimientos, el que represente el mayor beneficio. Entonces, envía un mensaje de adjudicación al oferente correspondiente para realizar el contrato. El oferente adjudicado, entonces, obtiene un compromiso para efectuar dicho trabajo cuando corresponda.

La principal característica que se puede visualizar es que no existe interacción entre los agentes oferentes; existe un agente que se adjudica el rol de "tomador de decisiones", el agente demandante. Esto ha traido ciertas controversias con respecto a la clasificación del protocolo Contract-Net. Por ejemplo, en [Durfee and Rosenschein, 1994], Contract-Net aparece inicialmente catalogado como un protocolo correspondiente a la RPD y no a un método de comunicación para Sistemas Multi-Agente. Para Ramos (1998) y otros, el protocolo contractual corresponde a la categoría de protocolos de negociación, y esta será la postura adoptada por el presente trabajo, puesto que no existe real cooperación entre los agentes, si no que sólo se negocian servicios (trabaja por oferta y demanda). En otros trabajos la red de contratos es considerada un protocolo de cooperación.

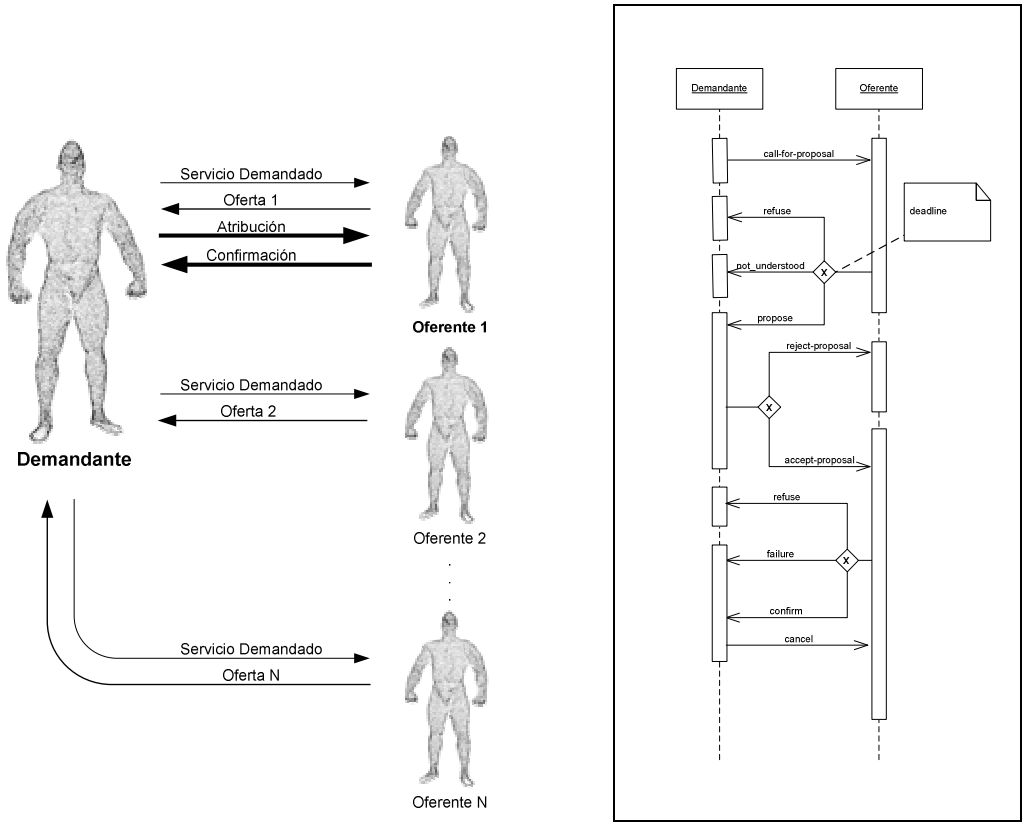


Figura 5.1. Protocolo de Negociación Contract-Net.

A la izquierda, una idea general de la oferta realizada por un agente y la selección del mejor oferente. A la derecha, un diagrama de secuencia de UML explicando la interacción entre demandante y oferente.

Como se puede apreciar en el diagrama de secuencias de la Figura 5.1, el protocolo contractual es algo más complejo de lo que se expuso antes. Por ejemplo, está considerada la situación en que el oferente no entienda la petición del demandante, así como los posibles errores que puedan provocarse durante la negociación, y hasta la posibilidad de cancelar el contrato.

Contract-Net es uno de los protocolos de interacción entre agentes incorporado en la amplia biblioteca de protocolos FIPA [33], una institución no lucrativa enfocada en la producción de estándares para la ínter operación de agentes de software heterogéneos.

[33] Más información de la institución en http://www.fipa.org.

5.2 Hacia la Identificación de Roles

En la identificación de roles, tradicionalmente, lo que propone GAIA es identificar la estructura de las actividades que normalmente se realizan en la organización de una forma, inicialmente, informal. Frecuentemente se identifican los departamentos, actores u organizaciones que participan en los procesos involucrados, y los roles que cumplen en esto (García-Ojeda *et al*., 2002).

El propósito de este trabajo es definir el modelo de un sistema que controle en forma distribuida la producción de una fábrica en un ambiente automatizado. Con esto, la cantidad de personajes humanos es mínima, por lo que los actores involucrados son, en su mayoría, equipos hardware que participan en el proceso, pudiendo identificarse: *transportadores de material*, *maquinarias de mecanizado de material*, *unidades de ensamble*, *almacenes de materias primas y/o productos terminados* y *unidades de control de calidad*. Además, puede identificarse la actividad de un ente humano, responsable del funcionamiento de la fábrica (que bien podría ser una persona o un equipo de funcionarios, dependiendo de la organización), al que se conocerá como *Encargado de producción* (ver caso de uso de la página 66).

El resumen de esta primera etapa se puede visualizar a groso modo en la Tabla 5.1.

Al visualizar los roles identificados, es posible notar la similitud con varios de estos. Por ejemplo, el rol de *Controlador de cola* se repite en las *Maquinarias de Mecanizado*, *Unidades de Ensamble* y en los *Controladores de Calidad*. En el *Almacén* podría haber cierto parentesco con el *Controlador de almacén*. Con esto, para el presente trabajo, estas unidades serán englobadas bajo el nombre de Servicios[34], y se le adjudicarán los roles de Controlador de Servicio y Controlador de Cola.

[34] La utilidad de generalizar una unidad conocida como Servicio es obvia: se puede contar con un ente general que ejecute alguna tarea en el sistema independiente en la programación de la producción de qué tipo de tarea se trate. Es decir, este Servicio podría tratarse de una unidad de mecanizado, control de calidad, o cualquier otra que se haya escapado a la investigación realizada en este trabajo o se genere producto de la propia evolución tecnológica en el futuro. La implementación de los roles no es importante aún.

Tabla 5.1. Identificación informal de roles del sistema.

Unidades	Rol(es)	Descripción Informal
Transporte de Material	Transportador	Trasladar un elemento desde un punto de origen a un punto de destino
Maquinaria de Mecanizado	Controlador de mecanizado	Ejecutar y controlar un proceso de mecanizado, y dar aviso una vez terminado éste, u otros posibles eventos.
	Controlador de cola	Llevar el registro de las piezas que llegan a la máquina y que están en espera de procesamiento, ordenándolas según la estrategia definida.
Unidad de Ensamble	Controlador de ensamble	Ejecutar y controlar un proceso de ensamble o armado de partes, y dar aviso una vez terminado éste, u otros posibles eventos.
	Controlador de cola	Llevar el registro de las piezas que llegan a la unidad, y que están en espera de ensamble, ordenándolas según corresponda.
Almacén	Controlador de almacén	Llevar el registro de las piezas disponibles en el almacén para algún uso o de los productos terminados o semi-acabados que contenga.
	Controlador de llegada de piezas al sistema	Provocar entrada de las piezas al sistema según los pedidos registrados.
Controlador de Calidad	Controlador de proceso de QC	Ejecutar y controlar el proceso de control de calidad y dar las instrucciones o sugerencias según corresponda.
	Controlador de cola	Llevar el registro de las piezas que llegan a la estación de control de calidad y que están en espera de procesamiento, ordenándolas según la estrategia definida.
Encargado de Producción	Registrador de órdenes	Definir las órdenes de manufactura según las especificaciones de los clientes.
	Ejecutor de la producción	Ejecutar el proceso de producción.
	Monitor de la producción	Monitorear el proceso de fabricación analizando los datos generados por el sistema.

Además, como se trabajará con el protocolo de negociación Contract-Net, es necesario agregar, por ahora informalmente, el rol de Oferente de Servicio y Demandante de Servicio, cuyas definiciones de forma general se explican por sí mismas.

5.3 Modelo de Interacciones

El modelo de roles es utilizado para especificar las relaciones y dependencias entre varios roles en una organización multi-agente (Wooldridge *et al*, 2000).

En la Figura 5.2 se visualiza la negociación realizada entre un rol demandante de servicio y un rol oferente según el protocolo contractual. En este, un demandante de servicio solicita la oferta de servicio (en este caso, el tiempo en el que se realizaría el procesamiento de la pieza de serle adjudicada la tarea) a un agente oferente (como se mencionó esto se realiza con un conjunto de posibles oferentes). El oferente realiza varias consultas, incluyendo información entregada por el rol de ControladorDeCola, para saber la situación de la cola del servicio en ese momento. Además, consulta su propia memoria para conocer, según las capacidades de la unidad, el tiempo en que se demora el servicio en realizar la operación solicitada. El OferenteDeServicio entrega al demandante el valor de oferta, quién la considera (además de agregarle el tiempo de transportar la pieza hasta el servicio oferente), en conjunto con todas las ofertas que llegan, y selecciona al más óptimo. Luego, el DemandanteDeServicio adjudica el trabajo al mejor oferente, quién verifica que aún se encuentre en condiciones de cumplir con el ofrecimiento, luego de lo cual confirma la prestación del servicio con el demandante.

Hasta aquí el modelo de interacciones de GAIA parece bastante conveniente, puesto que su lectura y comprensión son bastante sencillas. Sin embargo, la secuencialidad de este modelo impide considerar (por lo menos de una forma visualmente rápida de captar) la incorporación de ciertas excepciones que puedan suscitarse a lo largo de la ejecución del protocolo. En este sentido, los diagramas de secuencia de AUML presentan un soporte más poderoso.

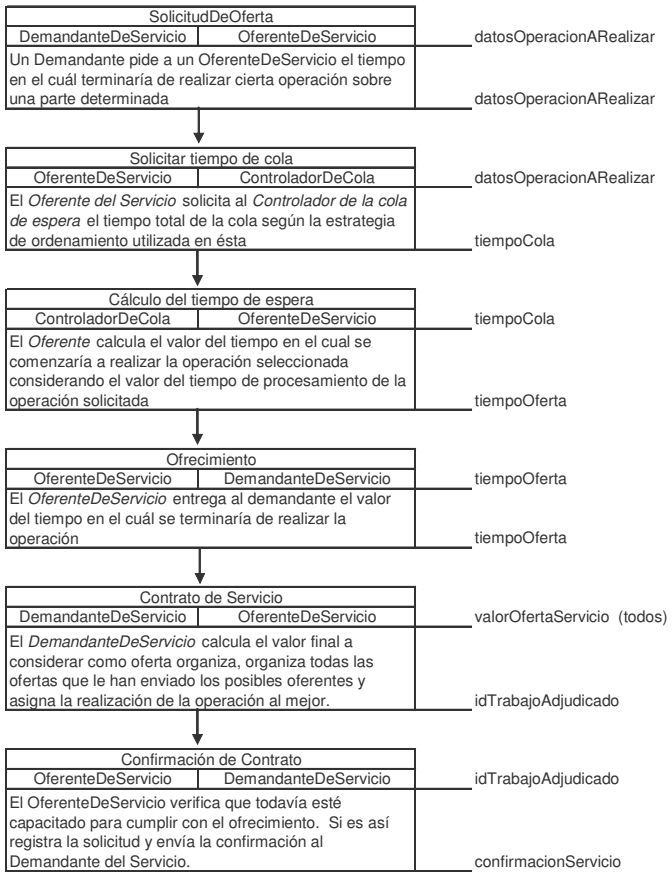


Figura 5.2.Protocolo *NegociacionDeTrabajo* (utilizando Contract-Net) asociado a los roles OferenteDeServicio, ControladorDeCola y DemandanteDeServicio.

Los modelos de la Figura 5.3 y la Figura 5.4 muestran los protocolos que se realizan al momento de trasladar una pieza desde una unidad demandante hasta una unidad oferente a la que se le a adjudicado un servicio, y en el momento en que una unidad de servicio solicita una pieza de la cola para su procesamiento, respectivamente.

Figura 5.3.Protocolo *TransporteDePartes* asociado a los roles DemandanteDeServicio, Transportador y ControladorDeCola.

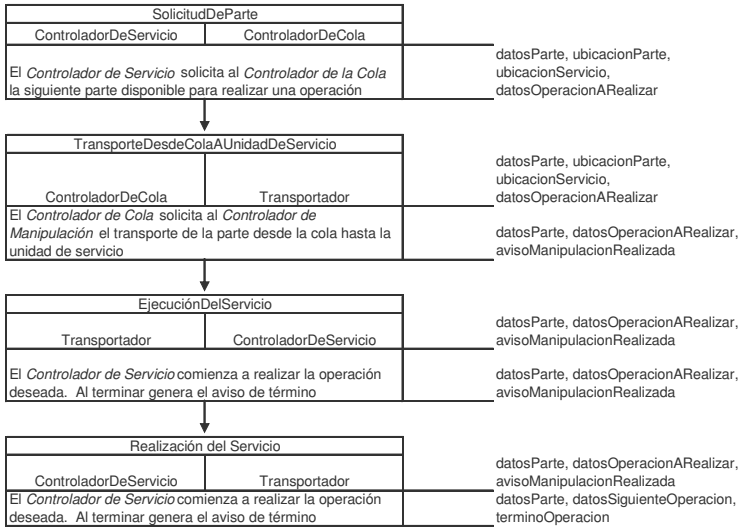


Figura 5.4.Protocolo *SolicitudDePiezaEnCola* asociado a los roles ControladorDeServicio, ControladorDeCola y Transportador.

5.4 Modelo de Roles

El modelo de roles es la especificación de los principales roles que se pueden distinguir en el sistema. Estos roles ya fueron identificados de una forma bastante informal en la sección 5.2 de este capítulo. En esta etapa, se visualiza a un rol como una descripción abstracta de una función que se espera realice una entidad (Wooldridge *et al*, 2000), caracterizándose por 2 tipos de atributos:

i. Los permisos asociados con el rol.

ii. Las responsabilidades del rol.

Los primeros atributos corresponden a la capacidad de un rol para acceder (permisos denominados *reads*), modificar (permisos denominados *changes*) o generar (permisos denominados *generates*) ciertos recursos de información necesarios para ejecutar sus tareas.

Con respecto a lo segundo, Wooldridge *et al* (2000) indican que la funcionalidad de un rol está definida por sus responsabilidades. Las responsabilidades son todo aquello para lo cual fue creado un rol. En GAIA, estas responsabilidades son divididas en dos categorías: las responsabilidades denominadas *liveness* (de ejecución) y las responsabilidades denominadas *safety* (de seguridad). Estas últimas permiten que la ejecución del rol se mantenga dentro de condiciones normales de funcionamiento.

Esquema del Rol: DEMANDANTEDESERVICIO			
Descripción: Solicita el servicio de otros agentes para realizar alguna operación sobre la pieza que tiene actualmente, asignándosela al agente que le reporte la mejor utilidad.			
Protocolos y Actividades: Contract_Net, TransporteDePartes			
Permisos:			
	reads	*ofertaServicio*	// El valor de la oferta de servicio de otro agente
	generate	*asignacionServicio*	// La asignación de a un agente para realizar una tarea
Responsabilidades Liveness: DEMANDANTEDESERVICIO = (Contract_Net. TransporteDePartes) Safety: • statusOferente = OK • statusSigTransporte = OK			

Figura 5.5.Modelo del rol *DemandanteDeServicio*.

Esquema del Rol: OFERENTEDESERVICIO			
Descripción: Retorna un valor de oferta a un agente que realiza el rol DemandanteDeServicio.			
Protocolos y Actividades: Cálculo_de_Oferta, Contract_Net			
Permisos:			
	reads	*demandaServicio*	// El valor de demanda de servicio de otro agente
		asignacionServicio	// La asignación para realizar una tarea determinada
	generate	*ofertaServicio*	// El valor de la oferta de servicio para enviarla a un demandante de servicio.
Responsabilidades Liveness: OFERENTEDESERVICIO = (Contract_Net || Cálculo_de_Oferta) Safety: • statusDemandante = OK			

Figura 5.6.Modelo del rol *OferenteDeServicio*.

Esquema del Rol: CONTROLADORDECOLA			
Descripción: Mantiene organizados los datos de las partes que se encuentran en la cola de una máquina, según una estrategia de ordenamiento.			
Protocolos y Actividades: Contract_Net, TransporteDePartes, SolicitudDePiezaEnCola			
Permisos:			
	reads	*datosPieza*	// El conjunto de datos de la parte que recién llega a la cola.
	changes	*registroCola*	// Actualiza los datos de las piezas que se encuentran en la Cola, ordenándolas además según una estrategia de ordenamiento.
	generate	*tiempoEsperaEnCola*	// El tiempo total que se demorará en ser atendida una pieza en el servicio.
Responsabilidades Liveness: CONTROLADORDECOLA = (Contract_Net. TransporteDePartes) | (SolicitudDePiezaEnCola) Safety: • true			

Figura 5.7.Modelo del rol *ControladorDeCola*.

Esquema del Rol: TRANSPORTADOR			
Descripción: Controlar y organizar los procesos de movilización de partes desde un lugar a otro.			
Protocolos y Actividades: TransporteDePartes, SolicitudDePiezaEnCola			
Permisos:			
	reads	*datosPieza*	// El conjunto de datos de la parte que recién llega a la cola.
		datosServicios	// Los identificadores de las unidades de fuente y destino del transporte.
	generate	*avisosDeTransporte*	// Avisos generados para informar acerca de la llegada de una pieza o de un posible error.
Responsabilidades Liveness: CONTROLADORDECOLA = (Contract_Net. TransporteDePartes | SolicitudDePiezaEnCola) Safety: • statusDestino = OK • statusSiguienteTransporte=OK			

Figura 5.7.Modelo del rol *Transportador*.

Capítulo 6

Diseño del Sistema. Hacia la Identificación y Especificación de los Agentes

---

En el capítulo anterior se identificaron y especificaron los roles existentes en un sistema productivo automatizado, de forma general. En la etapa que recién comienza, se asignarán estos roles a diferentes entidades autónomas, para que puedan ser realizados en pro de la resolución de la problemática en curso. Estas entidades, o *agentes*, interactuarán unas con otras según los roles adjudicados y los protocolos definidos en la fase anterior.

La etapa de Diseño propuesta por GAIA incluye el desarrollo de 3 modelos básicos: el modelo de agentes, el modelo de servicios y el modelo de conocidos; en los que se especifica la identificación de agentes y las asociaciones a los roles a realizar.

En esta etapa la GAIA aún no se muestran detalles de implementación, sólo un bosquejo general de lo que sería el sistema a realizar, y lo que se necesita tener en consideración. Hasta aquí llega esta metodología, y es por eso que existe la necesidad de utilizar modelos más concretos para pasar luego de forma más clara a la etapa de implementación. AUML será la solución a esta problemática, aplicada en el presente trabajo para complementar el ciclo de desarrollo del sistema.

6.1 Modelo de Agentes

En el modelo de agentes, como bien se ha dicho antes en este escrito, se identifican los principales agentes, asociándoles un conjunto de roles que deben ser ejecutados por ellos.

En el capítulo anterior, se establecieron principalmente cinco roles genéricos fundamentales (una vez más se insiste en la idea en que este es un modelo general y la implementación futura de estos puede variar): *OferenteDeServicio*, *DemandanteDeServicio*, *ControladorDeCola*, *ControladorDeServicio* y *Transportador*.

Parunak et al (1998) definen 6 tipos de agentes en un sistema manufacturero tipo: *Unit Process* (unidade de proceso), *Resource* (recurso), *Manager* (administrador), *Part* (parte), *Customer* (cliente) y *Supliré* (proveedores). De esta amplia gama de agentes, en el contexto en el cual se está trabajando, varios no son necesarios. Clientes y proveedores no son actores activos en un sistema de control de la producción (tal vez sí en un sistema de más alto nivel organizacional).

Con la identificación de los principales actores en un sistema de producción automatizado (expuestos en la sección 5.2 del pasado capítulo) es posible generalizar y establecer tan sólo dos tipos de agentes: agentes de servicio (que Parunak et al (1998) llaman *recurso*) y agentes de transporte. La definición anterior especifica dos clases genéricas de agente: *AgenteServicio* y *AgenteTransporte*. La Figura 6.1 muestra estos dos tipos de agente y los roles que se le asocian a cada uno.

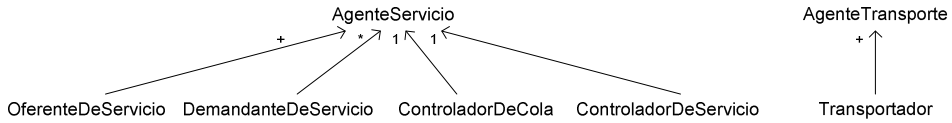


Figura 6.1. Modelo de Agentes del sistema.

De la figura anterior, al *AgenteServicio* se le asociaron los roles de *OferenteDeServicio*, *DemandanteDeServicio*, *ControladorDeCola* y *ControladorDeServicio*, mientras que al *AgenteTransporte* se le asoció el único rol de *Transportador* (el hecho de que al *AgenteTransporte* no se le haya asociado un rol específico de *ControladorDeCola* no quiere decir que no pueda recibir varias solicitudes de transporte y ordenarlas según estime conveniente; el rol de *Transportador* lo incluye por sí mismo).

En el caso particular del Laboratorio de Sistemas Automatizados de Producción CIMUBB, las instancias de las clases genéricas identificadas a implementar aparece en la siguiente tabla:

Tabla 6.1. Agentes identificados en el CIMUBB.

Clase de Agente	Instancia	Descripción
AgenteServicio	*AgenteASRS*	Agente encargado de llevar el control de las materias primas que contienen los feeders, así como de la administración de los templates ubicados en el almacén matricial.
	AgenteMill	Agente encargado del control de las operaciones de mecanizado de la fresadora CN.
	AgenteLathe	Agente encargado del control de las operaciones de mecanizado del torno CN.
AgenteTransporte	*AgenteRobot1*	Agente encargado de controlar las operaciones de transporte de partes entre la estación de almacén y la cinta transportadora que realiza el robot de la célula flexible de almacenamiento.
	AgenteRobot2	Agente encargado del control de las operaciones de transporte de partes entre el conveyor y las máquinas de control numérico, que realiza el robot de la célula flexible de mecanizado.
	AgenteConveyor	Agente encargado del control de las operaciones de transporte entre estaciones por medio de la cinta transportadora o 'conveyor'

En el sistema a implementar, a la clase genérica *AgenteServicio* pertenecen el '*AgenteASRS'*, el '*AgenteMill'* y el '*AgenteLathe'* [35] , mientras que a la clase *AgenteTransporte* pertenecen '*AgenteRobot1'*, '*AgenteRobot2'* y '*AgenteConveyor'*, todos los cuales adquieren de la clase genérica sus roles y características principales. Estos son los agentes que se pretende construir finalmente para dar solución a la problemática del presente trabajo.

En el modelamiento de agentes, Bauer (2001) indica que un agente debe ser definido a través de características como nombre de la clase del agente, descripción de estados, acciones métodos, capacidades descripción de servicios, protocolos soportados, etc., varios de los cuales GAIA explica en forma separada (Bauer propone para esto una modificación de los diagramas de clases de UML). Así mismo, Parunak (1999) propone la caracterización del agente a través de una tupla de cuatro elementos: estados, entradas, salidas y procesos. Como se puede apreciar, formalismos para describir un agente abundan.

Cabe destacar que en el presente trabajo sólo se considera Agente a todo ente capaz de recibir estímulos de su ambiente y generar respuestas en forma activa, es decir, mediante la ejecución de algún proceso. Se incorpora el término 'objeto de información' para referirse a todas aquellas entidades que no realizan trabajo activo, y que sólo son convocadas por los agentes para conocer algún dato o almacenar datos de algún tipo. Tal es el caos de las partes, templates, palletes, etc.

6.2 Modelo de Servicios

El modelo de servicios se realiza con el objeto de ejectuar una definición más acabada de un rol, a través de la identificación y descripción de los servicios que es capaz de ejecutar. En GAIA, el modelo de servicios es una tabla donde a cada servicio identificado de un rol, se describen sus entradas (datos que necesita para realizar el

[35] Inicialmente se pensaba en la concepción de un *AgenteQC* (para realizar procesos de Control de Calidad, sin embargo fue descartado por motivos de tiempo.

servicio), salidas (los datos que genera), y sus pre y post-condiciones (requisitos que debe hacer para ejecutar el servicio.

Tabla 6.2. Modelo de Servicios asociados al rol DemandanteDeServicio.

Servicio	Entradas	Salidas	Pre-Condición	Post-Condición
Solicitar ofertas de servicio	datosOperacionARealizar, datosAgentesServicios	tiempoOfertaMax	numOferentesDisponibles > 0	tiempoOferta >= 0
Adjudicar trabajo	listaOfertas	idAgenteAdjudicado	listaOfertas ≠ NULL	statusAgenteAdjudicado = OK

Tabla 6.3. Modelo de Servicios asociados al rol OferenteDeServicio.

Servicio	Entradas	Salidas	Pre-Condición	Post-Condición
Ofrecer Servicio	datosOperacionARealizar, tiempoEsperaEnCola	tiempoOferta	solicitudServicio = Done.	colaServicio=Disponible
Realizar compromiso	adjudicacionTrabajo	confirmacion	colaServicio=Disponible	true

6.3 Modelo de Conocidos

De la amplia gama de modelos que pueden ser utilizados (y no sólo de GAIA, si no de gran parte de las metodologías y lenguajes gráficos), el modelo de conocidos es, quizás, uno de los más sencillos de emplear.

El modelo de conocidos, llamado frecuentemente modelo de familiaridad, simplemente muestra qué agentes comparten información o se comunican; no se visualiza qué mensajes ni datos se comparten.

En la Figura 6.2 se visualiza un sencillo modelo de conocidos, que incorpora las dos clases genéricas de agente identificadas. Básicamente un *AgenteServicio*, como el que se

ha definido en este trabajo (en un sistema manufacturero), conoce a por lo menos un *AgenteTransporte*.

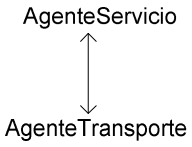


Figura 6.2. Modelo de Conocidos Simplificado del sistema.

La Figura 6.3 muestra un modelo más general. Es posible visualizar que en un nivel superior, todos los agentes de la clase *AgenteServicio* se conocen entre sí, ya sea para realizar negociación (en el esquema contractual demandante/oferente) o para compartir información. Por otra parte, cada *AgenteTransporte* conoce a otros agentes de esta misma clase, y no necesariamente (puede darse el caso en que un *AgenteTransporte* no conozca a otros de esta clase). Tampoco es obligatorio el hecho de que un *AgenteTransporte* conozca por lo menos a un *AgenteServicio* (un ejemplo concreto se puede observar en la Figura 6.4, con el AgenteConveyor). Un *AgenteServicio* puede conocer a uno o varios agentes de transporte.

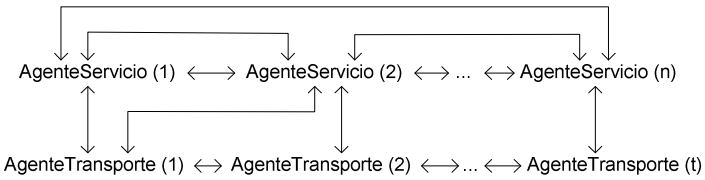


Figura 6.3. Modelo de Conocidos General.

Un nivel más específico de este modelo se puede apreciar en la Figura 6.4, en el que se aplica el modelo de conocidos directamente al sistema que se pretende implementar en el laboratorio CIMUBB. Aquí, como se mencionó en el modelo general, todos los agentes de servicios se conocen entre ellos. Además, el *AgenteASRS* conoce únicamente a un *AgenteTransporte*: el *AgenteRobot1*. En el laboratorio CIMUBB el robot de la primera célula flexible es el puente obligatorio para cualquier parte que entre o salga de la estación.

Por otra parte, el *AgenteRobot1* conoce además al *AgenteConveyor*, quien solo mantiene contacto con los dos agentes de la clase *AgenteTransporte* y no con alguno de la clase *AgenteServicio*.

El *AgenteRobot2* mantiene comunicación directa con los dos agentes que controlan las máquinas CN: el *AgenteMill* y el *AgenteLathe*.

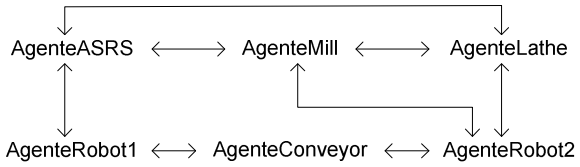


Figura 6.4. Modelo de Conocidos del sistema a implementar en el CIMUBB.

6.4 Hacia la generación de un modelo más concreto del sistema. Aplicación de AUML.

Sin duda que GAIA es una metodología poderosa en la conceptuación de un sistema multi-agente. La variedad y riqueza de la propuesta permiten al usuario identificar rápidamente los roles (concepto que sin duda está en boca de gran parte de la comunidad que trabaja en el área), sin preocuparse de qué agente lo realizará, y luego apoya la identificación y descripción de protocolos y agentes del sistema. Sin embargo, lamentablemente GAIA aún no es una metodología completa, puesto que sólo llega hasta la etapa de diseño. Y no sólo eso, si no que aún los diagramas son demasiado vagos para conducir al desarrollador a una etapa de construcción fluida y sustentada.

AUML parece la solución (o por lo menos un fuerte apoyo). Al ser una extensión del lenguaje unificado UML, no solo hereda sus beneficios (ver sección 3.1.1), si no que además incorpora una serie de modificaciones que lo hacen bastante más usable para modelar sistemas con agentes que su antecesor Orientado a Objetos.

En la propuesta de Odell *et al* (2000a), se aplica gran parte de los diagramas clásicos de UML (como los diagramas de secuencia, estado y actividad) para representar el sistema multi-agente, y a los agentes mismos, desde 3 perspectivas o niveles de abstracción: a) un primer nivel de representación del protocolo global, b) un nivel de representación de las interacciones entre los agentes que actúan en el sistema multi-agente, y c) un nivel de representación del procesamiento interno del agente.

6.4.1 Nivel 1: Representación del Protocolo Global.

En el primer nivel de abstracción de AUML se modela la interacción principal de los agentes que comprende el sistema. Para esto, Odell *et al* (2000a) proponen la utilización de dos de las técnicas de UML que mejor representan un protocolo para su reutilización: paquetes y plantillas.

Como una forma de ilustrar esta aplicación, se seleccionó utilizar plantillas (templates) para modelar los principales protocolos encontrados en el sistema a modelar. Del diagrama de secuencia generalizado que ilustra la Figura 6.6, se pueden extraer los dos protocolos más importantes del sistema: el protocolo NegociacionDeTrabajo (que es básicamente el protocolo Contract-Net) y el protocolo de TransporteDePartes, definidos ambos en la Figura 6.5.

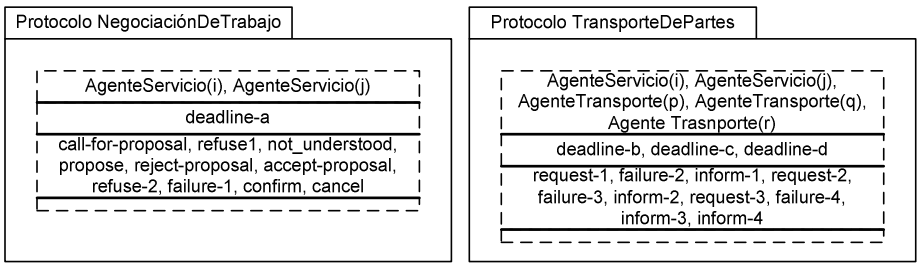


Figura 6.5. Plantillas de los principales protocolos del sistema: NegociacionDeTrabajo y TransporteDePartes.

Nótese la utilización de palabras como call-for-proposal, refuse y accept-proposal en los diagramas recién ilustrados. Estos vocablos, que seguirán siendo utilizados por lo menos a través de lo que queda del presente capítulo, son extraídos de los llamados actos del habla, definidos en [FIPA, 2005], y son reconocidos como un estándar para representar comunicaciones entre agentes.

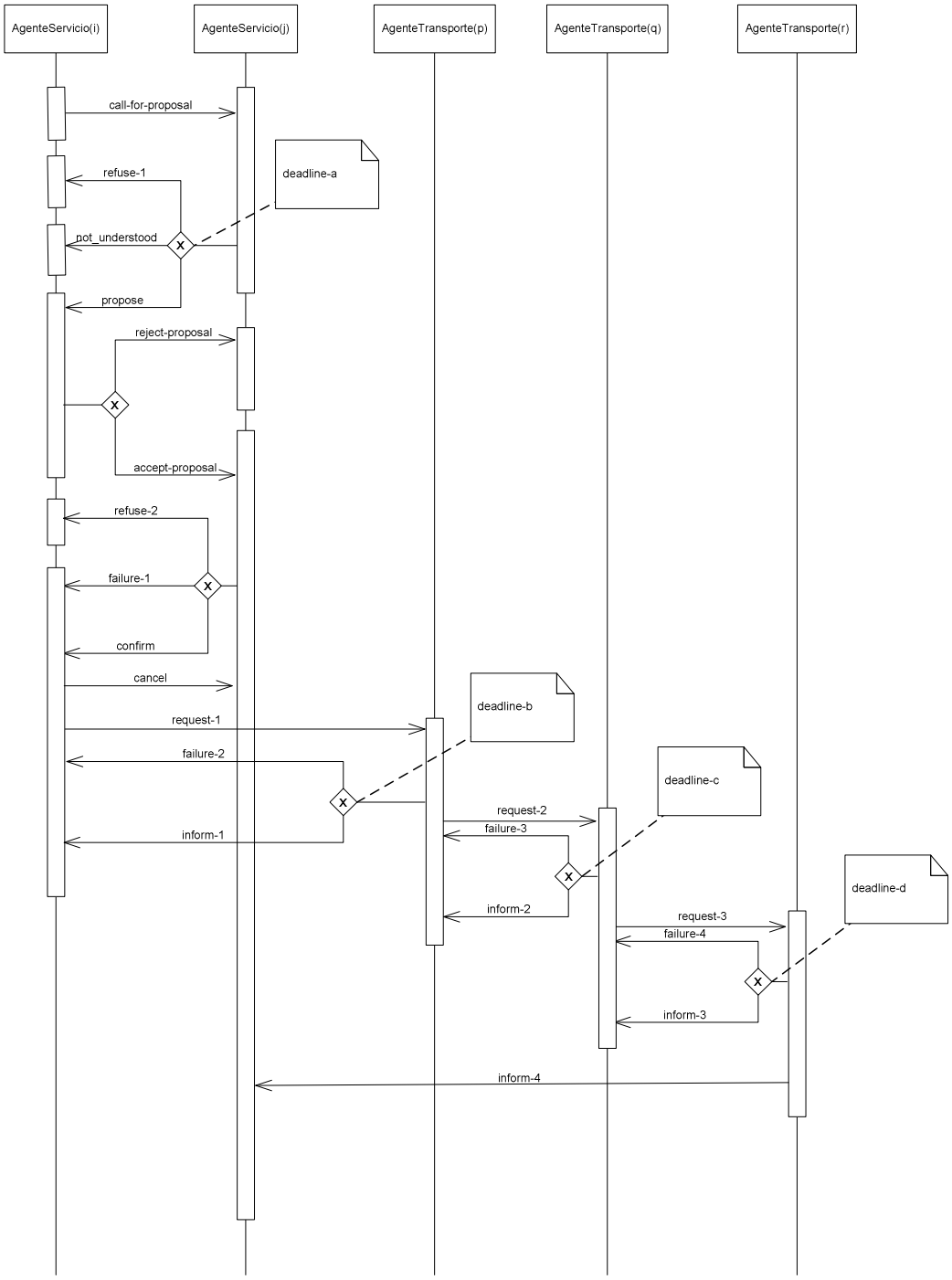


Figura 6.6. Diagrama de Secuencia del Protocolo Global del Sistema.

6.4.2 Nivel 2: Representación de Interacciones Entre Agentes.

En el nivel de Representación de Interacciones Entre Agentes, se muestran los principales flujos de información que se llevan a cabo entre los agentes, detallando además, los roles que interactúan. Para esto, Odell *et al* (2000a) sugieren la utilización de diagramas de secuencia, diagramas de colaboración, diagramas de actividad y cartas de estado.

El diagrama de la Figura 6.7 muestra las interacciones existentes desde el momento de generarse una negociación entre un AgenteServicio(i) quién inicialmente realiza el rol de DemandanteDeServicio y un AgenteServicio(j) que inicialmente realiza el rol de OferenteDeServicio. Luego de la negociación comienza el protocolo de TransporteDePartes, para movilizar la pieza adjudicada a la cola del AgenteServicio(j). Si todo marcha bien, el agente registrará y ordenará la pieza dentro de la cola (rol ControladorDeCola). Una vez que el rol de ControlDeServicio solicite la siguiente pieza, se solicitará a un transporte[36] lleve la pieza hasta la unidad de servicio (el lugar físico). Cuando termina el servicio el agente adopta el rol de DemandanteDeServicio para volver a repetir el ciclo.

Otra forma de representar las interacciones puede visualizarse en el diagrama de colaboraciones de la Figura 6.8, en el cual se visualizan los mismos mensajes que en el diagrama de secuencias de la Figura 6.7, sólo que no es posible visualizar el orden en que se ejecutan. En estricto rigor, esto debe ayudar al desarrollador a visualizar que mensajes llegan a cada rol de un agente y quienes los envían. En esto complementa un poco la idea del modelo de conocidos de GAIA (ver sección 6.3).

La carta de estados de la Figura 6.9 muestra un conjunto de estados a través de los cuales pasa el sistema o los agentes en la ejecución de una o varias tareas.

[36] Aunque en algunos sistemas automatizados la misma máquina que representa el rol de servicio tiene la capacidad de ingresar automáticamente la pieza desde la cola hasta el lugar de procesamiento. Esto no invalida el modelo, sólo se debe modificar el flujo hacia el transporte externo por el del controlador del mecanismo de transporte de la máquina.

Todas estas formas de representación están enfocadas a mostrar la mensajería que se realiza en el ambiente distribuido, y visualizar, aún de una manera muy pobre, las reacciones de los agentes ante estos estímulos del ambiente.

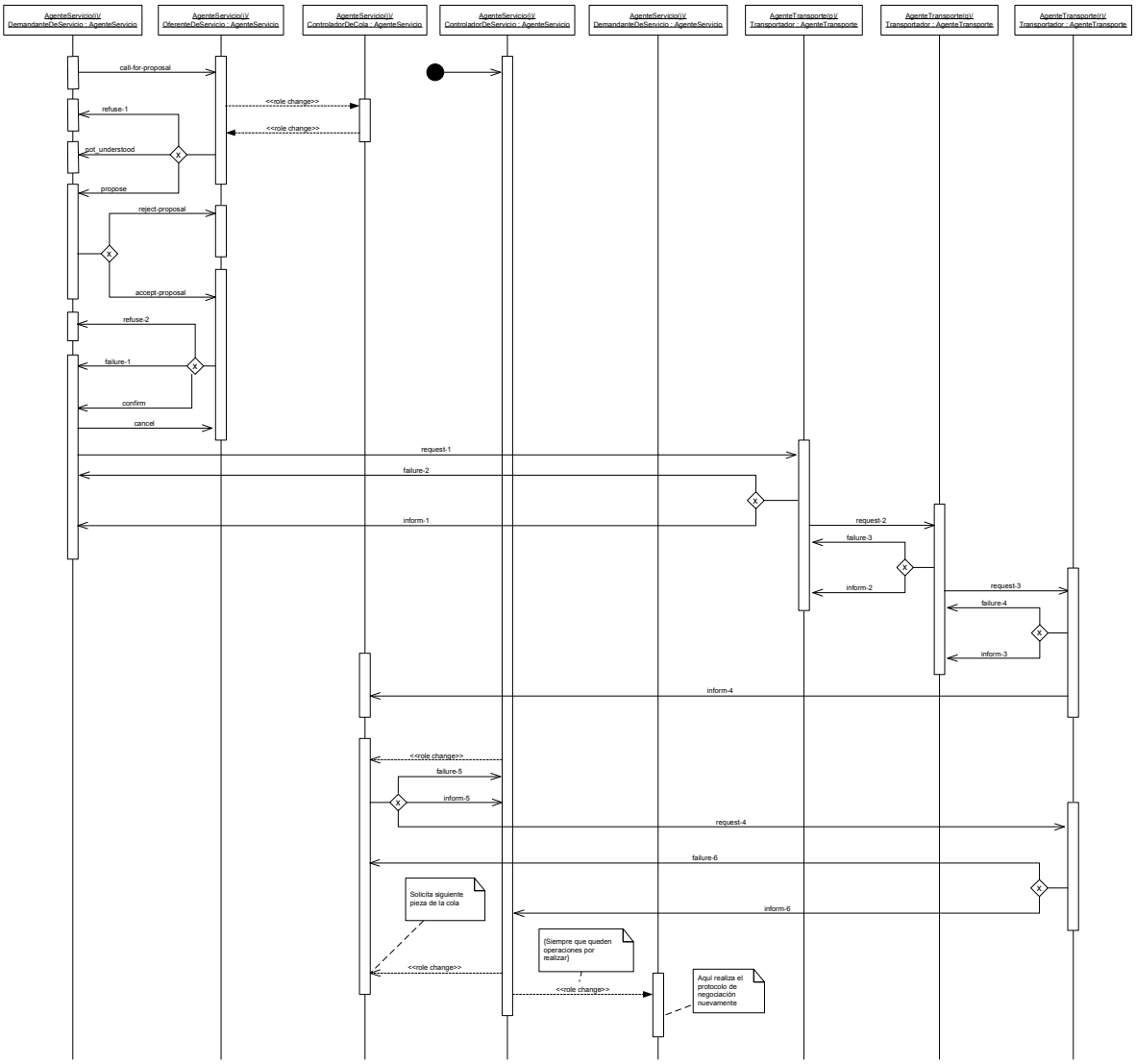


Figura 6.7. Representación de Interacciones entre los Agentes.

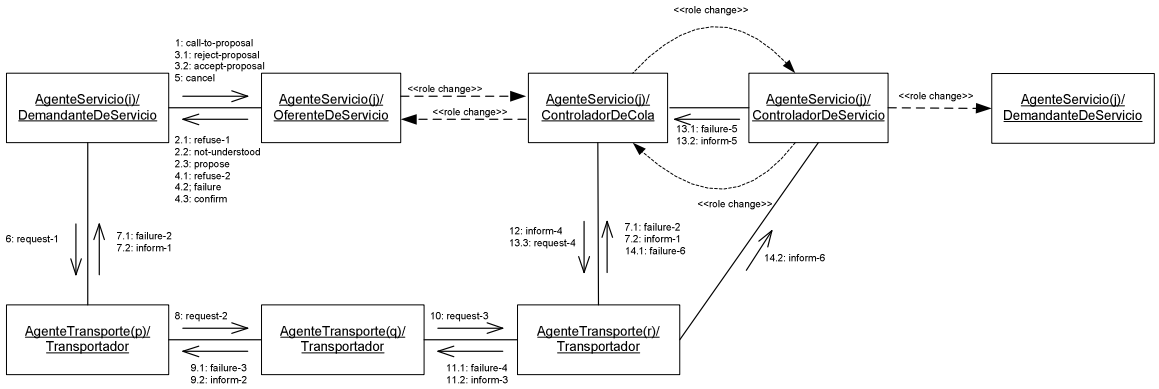


Figura 6.8. Diagrama de colaboración de las interacciones entre los agentes.

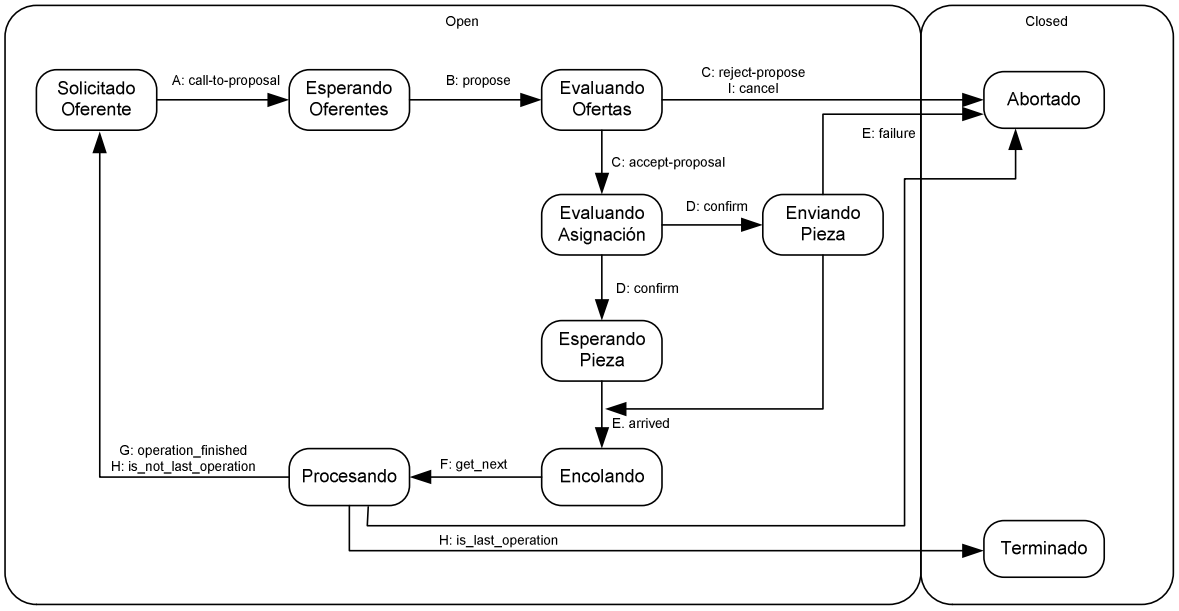


Figura 6.9. Carta de Estados representando las situaciones por las que pasan los agentes al interactuar.

El diagrama de secuencia de la

Figura 6.10 muestra un nivel más detallado de las comunicaciones. En este modelo se reemplazaron los vocablos de actos del habla por los mensajes reales a utilizar en el sistema a implementar. En el diagrama, las líneas paralelas emergentes de un único agente corresponden a hilos, es decir, tareas que se ejecutan en forma “paralela”. En el modelo se

incluyeron, además, otras aplicaciones y dispositivos físicos utilizados directa o indirectamente por los agentes. Tal es el caso de los Devices Drivers y los dispositivos de control de campo (Robot Controllers y PLC).

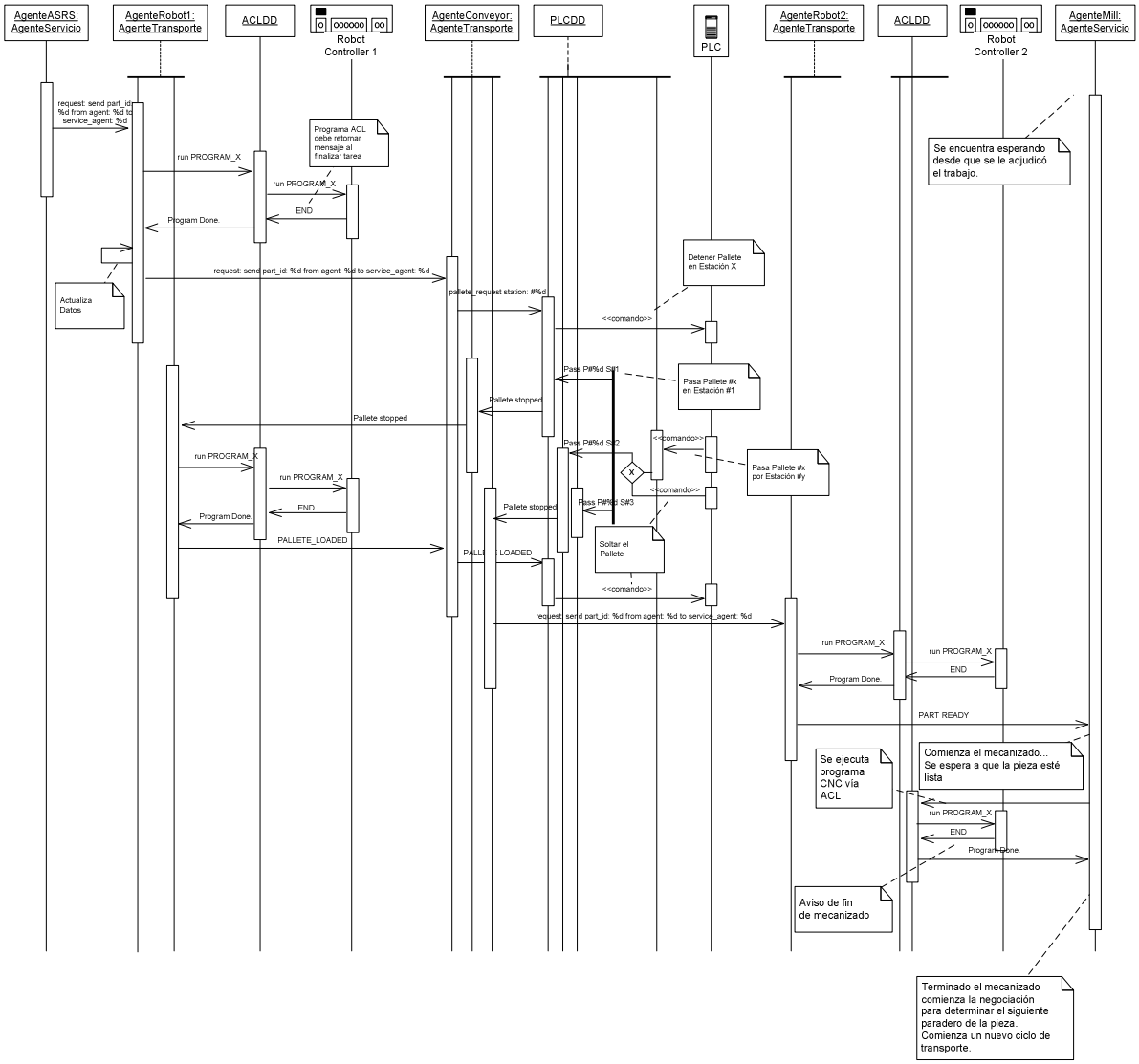


Figura 6.10. Representación de Interacciones entre los agentes del sistema a implementar.

6.4.3 Nivel 3: Representación del Procesamiento Interno del Agente.

Finalmente, el nivel de Representación del Procesamiento Interno del Agente, clarifica el funcionamiento de cada agente, exponiendo qué procesos o actividades realiza a través del tiempo y qué cambios en el entorno gatillan sus cambios de estado.

Odell *et al* (2000a) recomiendan para esto la utilización de diagramas de actividad y cartas de estado. En particular, se considera de especial utilidad el modelamiento mediante diagramas de actividad, por ser de fácil lectura y áltamente representativo de la secuencia de procesos ejecutados por el agente.

Una de las características más importantes del diagrama de actividad es que, a diferencia del resto, muestra únicamente una única rama de procesos del agente, es decir, ilustra una situación particular de comportamiento del agente ante determinadas situaciones. Por ejemplo, el diagrama de actividad de la Figura 6.11 muestra el caso en el que a un AgenteServicio le es adjudicado un trabajo, y este lo recibe sin problemas en la cola de espera. En la figura, los recuadros con línea continua son actividades propias del agente, mientras que los enmarcados con una línea discontinua son procesos ejecutados desde agentes o sistemas externos.

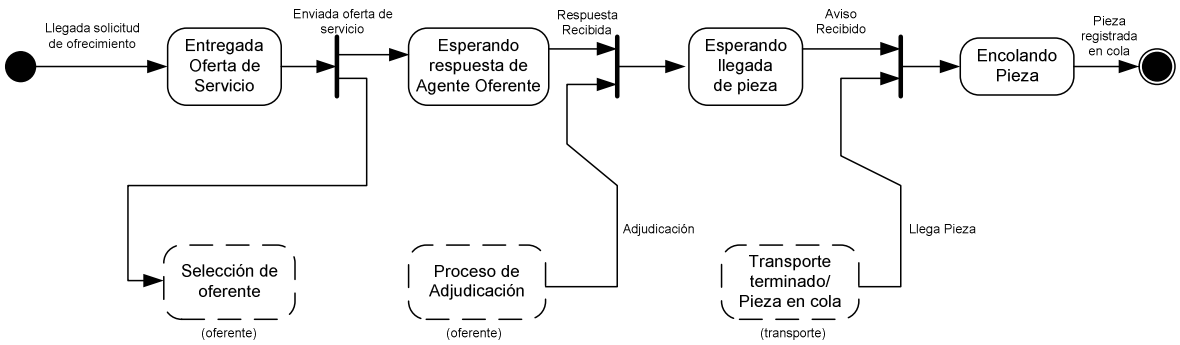


Figura 6.11. Diagrama de Actividad del AgenteServicio al Recibir una Adjudicación de Trabajo.

Así mismo, el diagrama de actividad de la Figura 6.12 muestra el proceso en el que un AgenteServicio termina de realizar cierto servicio sobre una parte y solicita la intervención de otro agente, adjudica luego el trabajo a otro y despacha satisfactoriamente la pieza.

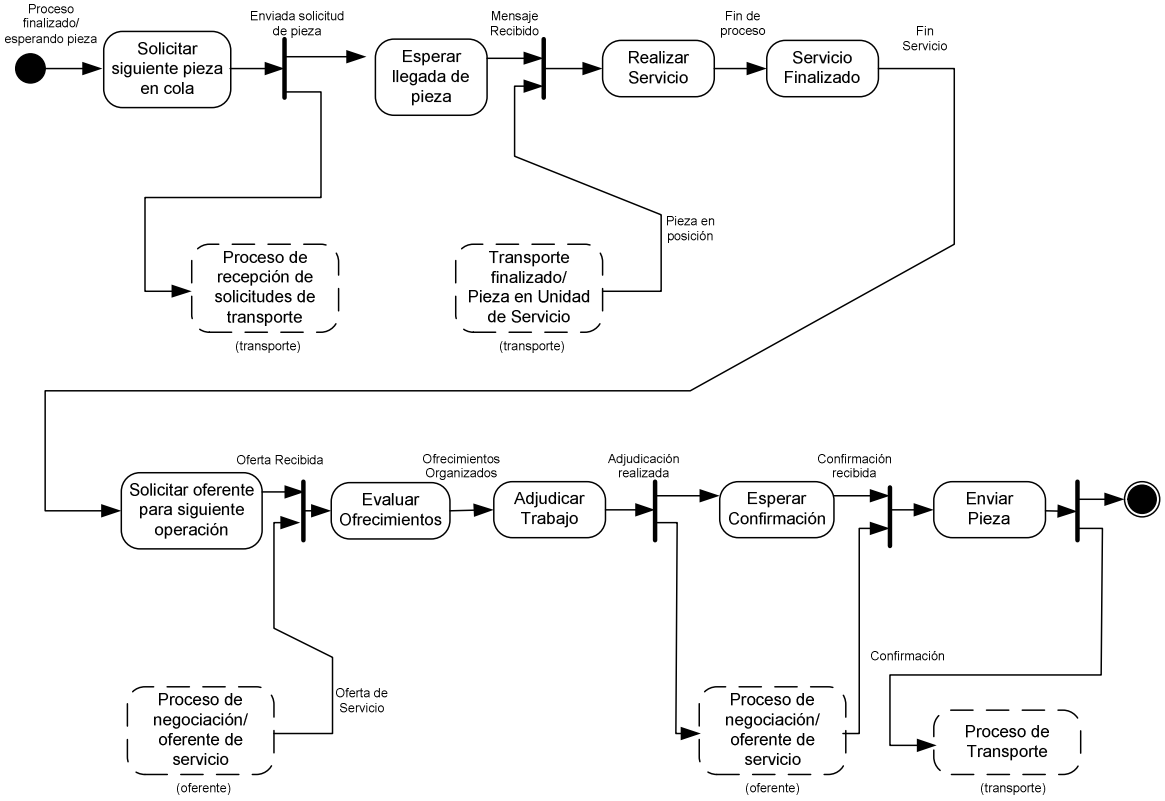


Figura 6.12. Diagrama de Actividad de AgenteServicio que Adjudica un Trabajo a Otro y Despacha una Pieza.

Finalmente, el diagrama de la Figura 6.13 ilustra el proceso en que un AgenteTransporte recibe una solicitud de transporte de una parte de un lugar a otro. En el diagrama se explicita que el proceso cumple con los pre-requisitos para el transporte y realiza las actividades correspondientes, dando al fin, el aviso de transporte realizado a los agentes interesados en conocer este evento.

El proceso de diseño llega hasta este punto. Estos diagramas darán pie a la concepción de un conjunto de agentes de software capaces de solucionar la problemática de la programación de la producción (ver sección 1.2.1) con un esquema heterárquico distribuido. En el siguiente capítulo se expondrán los resultados de una implementación que se realizó en el Laboratorio de Sistemas Automatizados de Producción CIMUBB, para validar la propuesta. Queda abierta la posibilidad al lector de seguir explorando nuevas formas de modelar e idear un sistema multi-agente; bibliografía abunda en esta temática.

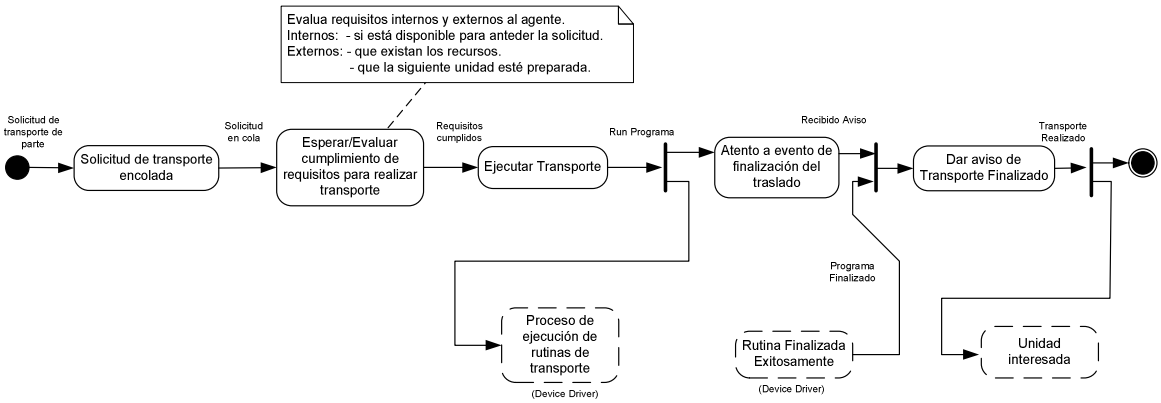


Figura 6.13. Diagrama de Actividad de AgenteTransporte que Ejecuta una Operación de Traslado de una Parte.

Capítulo 7

Validación de la Propuesta: Construcción, Implementación y Análisis de Resultados

---

Armados de los modelos expuestos en los capítulos anteriores, ya sea para verificar la presencia de errores o incoherencias en la propuesta, o para guiarse en la construcción del sistema, comienza el capítulo cúlmine del presente proyecto: la Validación de la Propuesta.

En el período final del año 2004, hasta principios de marzo de 2005 comenzó en el Laboratorio de Sistemas Automatizados de Producción CIMUBB, un arduo proceso de construcción del sistema multi-agente para el control de la producción que validaría la propuesta presentada como fondo en este trabajo. El trabajo fue extenso, luego de lo cual, se logró la concepción de un conjunto de aplicaciones de software autónomas (agentes) capaces de realizar cada una un conjunto de tareas y coordinarse para solucionar la problemática de la programación de la producción (discutida en el primer capítulo de este escrito).

La aplicación a realizar era clara. Se pretendía construir un conjunto de agentes que, operando desde varias máquinas, se coordinaran en la resolución de una orden de manufactura ingresada desde una de las estaciones. El producto final, fue un sistema multi-agente capaz de recibir inicialmente una orden de manufactura consistente en operaciones de mecanizado de fresado y torneado, y, una vez lista ésta, ejecutar la producción por medio de los agentes quienes realizan el control de los distintos dispositivos.

El presente capítulo habla acerca de eso; del proceso de construcción e implementación de los agentes. Qué herramientas se utilizaron, cuáles fueron los problemas encontrados y qué soluciones se encontraron. Finalmente se ha de realizar el análisis de los resultados obtenidos, siendo de especial interés, su evaluación al ponerlo frente al actual sistema imperante Open-CIM.

7.1 Discusión Acerca del Resultado Esperado. El Objetivo Final.

En el Capítulo 4 se especificaron los requerimientos que debería cumplir el sistema que resultase de todo este trabajo.

Se rescata de sobremanera el segundo requerimiento expuesto, el de *"asegurar en lo posible el cumplimiento de las fechas de entrega"*. Esto es vital. Como objetivo final, el sistema será rentable para la empresa siempre que sea capaz de responder a los compromisos de la organización con sus clientes. En parte esto último se ha solucionado, como en varios trabajos existentes, aplicando a las colas de las máquinas (llamadas en este trabajo "servicios"), estrategias de ordenamiento[37] de diversa índole.

Por otra parte, el principal punto de interés está en la tolerancia a las fallas. Como se ha mencionado en el primer capítulo, los sistemas actuales altamente centralizados tienden a generar antes de la ejecución de la producción (*off-line*) la programación completa de las actividades que deben realizar los diversos componentes del sistema. Así, cada movimiento de una parte de un lado a otro, cada detención o ejecución de las unidades de transporte y, principalmente, cada procesamiento de los materiales, están determinados antes de dar la orden de inicio.

¿Cuál es el problema?. Es bastante claro: en caso de que un dispositivo que haya sido considerado en la programación inicial caiga, provoca la inutilización de toda la carta de procesos que hasta el momento se estaba siguiendo. El sistema debe ser detenido, el ambiente virtual del planificador central (la definición de las máquinas) debe ser regenerado y la planificación vuelta a realizar.

Contract-Net, como protocolo de interacción entre agentes, soluciona este problema. ¿De qué forma?. Como la programación es *on-line*, simplemente los dispositivos que fallan no son considerados en las siguientes negociaciones.

[37] Estrategias como FIFO, LIFO, SPT o EDD, por mencionar algunas.

El producto a construir debía ser capaz de cumplir a cabalidad con estos requisitos. Un ambiente amistoso de ejecución o la posibilidad de configurar rápidamente el sistema fue descartada del proyecto por considerarse irrelevantes.

Por último, el sistema debe ser capaz de realizar operaciones de torneado y fresado (ver Capítulo 4) con una orden de manufactura tipo. La célula de Control de Calidad no debía ser automatizada (con el propósito de enfocar los esfuerzos en probar la propuesta más que en la variedad de aplicaciones que se podían realizar).

7.2 Especificación de Recursos a Utilizar. Definición de Lenguajes y Plataformas.

La presente sección se reducirá a una corta explicación de qué herramientas fueron utilizadas para realizar el presente trabajo. La falta de estudios en el presente trabajo para tomar decisiones formales para seleccionar un conjunto de tecnologías informáticas hacen que cualquier opinión dada se reduzca a la simple experiencia del autor de este documento. Todo se reducirá a esto:

7.2.1. Selección del Sistema Operativo.

Basta recorrer Internet, o preguntarle a cualquier encargado de soporte para darse cuenta de que la plataforma Microsoft Windows, en cualquiera de sus versiones, presenta serios problemas de robustez. Los equipos que operan bajo esta administración, generalmente tienden a colapsar pasado algún tiempo de utilización, con lo que es necesario reiniciar una y otra vez la máquina esperando se solucione el problema. Obviamente, estas falencias lo hacen ineficiente en un sistema productivo, donde los procesos deben permanecer ejecutándose durante horas ininterrumpidamente y cualquier caída del sistema reporta gastos fatales.

Desde las tinieblas de la falta de lucratividad aparece lo que se muestra como una solución a todos estos problemas.

Los sistemas Linux (basados en el antiguo Sistema Operativo UNIX) están, hoy en día, atrayendo a cada vez más empresas, debido a su robustez y eficiencia en el manejo de los procesos.

Apoyados en este rápido auge, por la experiencia adquirida por el autor del presente proyecto y la abundante bibliografía, además de ser considerado como uno de los sistemas gestores más prometedores en el área de la automatización en el largo plazo, se escogió como ambiente de trabajo el Sistema Operativo Linux, el cual fue instalado en las máquinas Work Station 1 y Work Station 2 (donde se ejecutarían los agentes). Particularmente se escogió la versión 9.0 de la distribución Suse, para facilitar la detección de los periféricos hardware de las máquinas.

7.2.2. Lenguajes de Programación y Otros.

El conjunto de lenguajes de programación que se puede utilizar en Linux es amplio. Sin embargo, pocos cuentan con la capacidad para manejar en forma eficiente protocolos de comunicación de todo tipo.

De los modelos expuestos en el Capítulo 6 se infiere la necesidad de trabajar con un lenguaje de programación capaz de manejar sistemas multi-hilos, o por lo menos trabajar concurrentemente en forma óptima. Además de la obvia necesidad de operar con redes y puertos de comunicaciones, y, posiblemente, motores de bases de datos, la lista se reduce ampliamente.

Para el presente trabajo se escoge la utilización del lenguaje de programación C++, que en Linux cuenta con un compilador denominado g++, además de la utilización del comando *make* y archivos Makefile para la automatización de la compilación de los distintos archivos. Este lenguaje presenta un conjunto significativo de librerías para manejar comunicaciones seriales, de red, procesos, y otros, lo cual lo hace una de las

opciones más convenientes para el presente trabajo, además de la amplia bibliografía disponible.

La capacidad de C++, como lenguaje Orientado a Objetos[38], no fue explotada. Sólo se utilizó para la programación de ciertos objetos para manejar las comunicaciones. La mayor parte de la programación fue inherentemente estructurada.

Una vez solucionadas estas incógnitas (que no dejan de ser importantes, y en gran medida pueden definir el éxito o fracaso de un proyecto), se procede a la preparación de los equipos y la programación propiamente tal.

7.3 Comenzando la Implementación. Preparación de la Plataforma.

Hay que recordar que el sistema actualmente imperante en el laboratorio CIMUBB, Open-CIM, opera bajo la plataforma Microsoft Windows 98. Por otra parte, todos los softwares (ver sección 4.3.3) necesitan, por su configuración, la operación de la estación CIM-Manager.

Debido a la instalación de tarjetas propietarias de control de las máquinas CNC, las estaciones PC-Mill y PC-Lathe deben mantenerse funcionando con la actual plataforma y su software CAM. Sin embargo, se dijo en la sección 4.3.4 que esto era irrelevante, puesto que la ejecución de los programas CN de la fresadora y el torno son provocados por mensajes seriales que envía el Robot-Controller a las estaciones PC-Mill y PC-Lathe y que interpretan las aplicaciones CAM que tienen ejecutándose en cada equipo.

Esto se puede visualizar fácilmente en la Figura 7.1.

En la figura, la Work Station 2 envía un mensaje *s1* vía puerto serial al Robot Controller 2. *s1* es un mensaje en lenguaje ACL que ejecuta un programa del controlador del robot, donde se hace llamado a alguna subrutina relacionada a la máquina CN.

[38] En realidad C++ es un híbrido entre O.O. y Programación Estructurada.

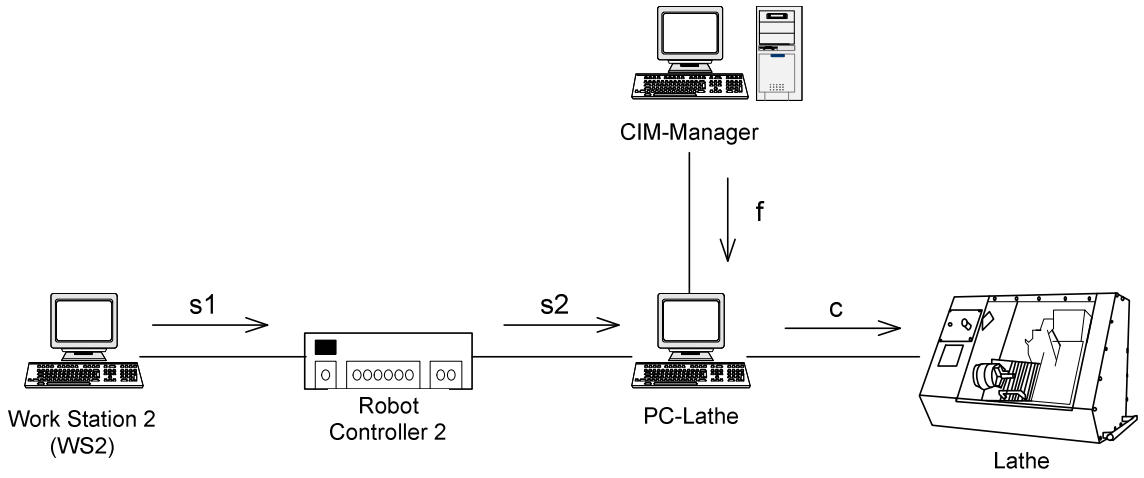


Figura 7.1.Flujo de Comunicaciones para Ejecutar un Programa de Control Numérico

El Robot Controller 2 recibe la instrucción *s1*. Si esta instrucción contiene llamado a alguna subrutina de la máquina CN, la direcciona hacia el PC en el que se aloja el software de control de la máquina (en el caso de la fresadora el PC-Mill con su software Benchman y en el caso del torno del PC-Lathe con su software WPLT). Los software CAM de cada PC leen los puertos seriales. Robot Controller envía la solicitud de ejecución del programa CN a través del puerto serial, representado en la figura como el mensaje *s2*. En el PC correspondiente se lee el mensaje, se busca el archivo (*f*) en el CIM-Manager y se ejecuta el programa de control (etiqueta *c*).

Con este esquema, es fácil darse cuenta de que PC-Mill y PC-Lathe pueden "independizarse" del CIM-Manager a través de la copia de los archivos CN en directorios locales a ellos. Además, se deben modificar 3 programas ACL del Robot Controller 2, que direcciona los programas CN a utilizar por los softwares Benchman y WPLT. Estos programas (en el CIMUBB) son llamados `dir1`, `dir2` y `dirm`. En ellos deben identificarse las nuevas direcciones de los archivos CN.

Una vez hecho esto, el esquema se reduce a lo expuesto en la Figura 7.2. y, con esto, PC-Lathe y PC-Mill pueden mantenerse operando tal y como hasta la fecha.

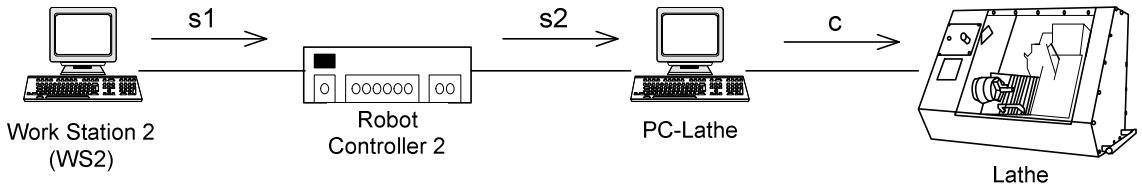


Figura 7.2.Nuevo Flujo de Comunicaciones para Ejecutar un Programa CN sin CIM-Manager

7.4 Construcción de los Agentes.

Finalmente se ha llegado a la etapa más compleja de todas: la construcción del sistema. En esta sección se presentarán las principales consideraciones realizadas al momento de programar los agentes. Cuál es su estructura de programación básica, estructuras de datos y archivos, y protocolos utilizados. En todo caso, esta implementación no es más que una aplicación de los modelos expuestos en los capítulos anteriores, y el lector del presente documento podrá explorar nuevas formas de realizar un trabajo similar.

7.4.1. Estructura General de un Agente.

La primera consideración a realizar es la estructura base de un agente. Sobre todo, un agente de software debe ser capaz de realizar un conjunto de operaciones (roles), muchas veces varias en paralelo, recibir información de su entorno y actuar realizando algún proceso físico o comunicándose con otro agente (o con un usuario humano).

Guessoum y Briot (1999), presentaron una arquitectura base para un agente, en la cual un agente es presentado como el conjunto de varios módulos, que definen su comportamiento. Estos módulos son clasificados funcionalmente en tres tipos: a) módulos de percepción, con los cuales obtiene un comportamiento reactivo, y es capaz de percibir modificaciones en su entorno o recibir mensajes de otros agentes, b) módulos de razonamiento, con los que adquiere un comportamiento deliberativo, es decir, es capaz de

obtener y generar conocimiento y meditar acerca de su propio comportamiento (representa creencias, objetivos, etc.), y c) módulos de comunicación, mediante los cuales es capaz de generar cambios es su ambiente (acciones directas) o enviar información o instrucciones a otros agentes (acciones indirectas). Con estos últimos, los agentes pueden ser reactivos o preactivos utilizando, por ejemplo, los conocidos actos del habla (FIPA, 2005).

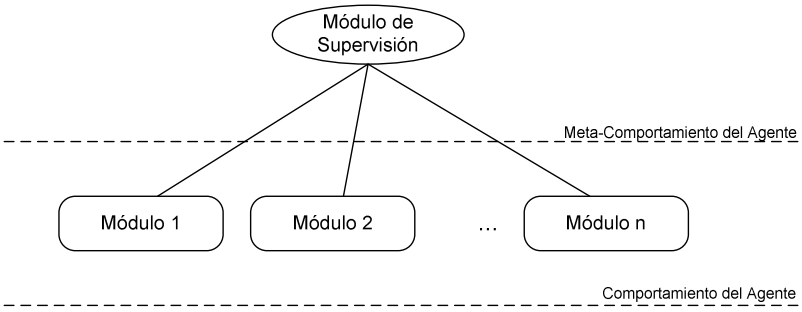


Figura 7.3. Modelo general de un agente propuesto por Guessoum y Briot (1999).

La propuesta de Guessoum y Briot es interesante y fue considerada al momento de programar los agentes para la validación de la propuesta.

Se propone, entonces, concebir a un agente como un conjunto de estos tres tipos de módulos. Sin embargo, se modifica un poco la propuesta agregando el concepto de proceso, en lugar del de módulo. De esta forma, se obliga a conceptuar a un agente como un conjunto de tareas que se ejecutan en forma paralela y asincrónica, con el objetivo de realizar varios de sus roles y/o desarrollar su aprendizaje (propias de los agentes inteligentes), a la vez, o por lo menos en forma concurrente, y a buscar mecanismos de comunicación y coordinación entre estos procesos[39].

La Figura 7.4 presenta el esquema propuesto para un agente de servicio. En esta, se identifica un proceso para la percepción de los mensajes que pueda recibir de algún otro

[39] En estricto rigor, no necesariamente se deben programar todos los procesos. Por ejemplo, se puede considerar a los procesos levantados por el propio sistema operativo para recibir mensajes a través de un puerto del pc (como en una comunicación con sockets) como un proceso utilizado por el agente.

agente, y un proceso de comunicaciones para generar respuesta a los estímulos propios y externos. Además, los roles *Control de Proceso* (se le dio este nombre aquí para hacerlo más genérico; antes se lo llamó *Control de Servicio*), Control de Negociación (que en realidad se refiere al rol de ofrecimiento de servicio), y el rol de Control de Cola.

Además, en la cúspide del agente, un proceso principal de supervisión del comportamiento del agente y de aprendizaje[40].

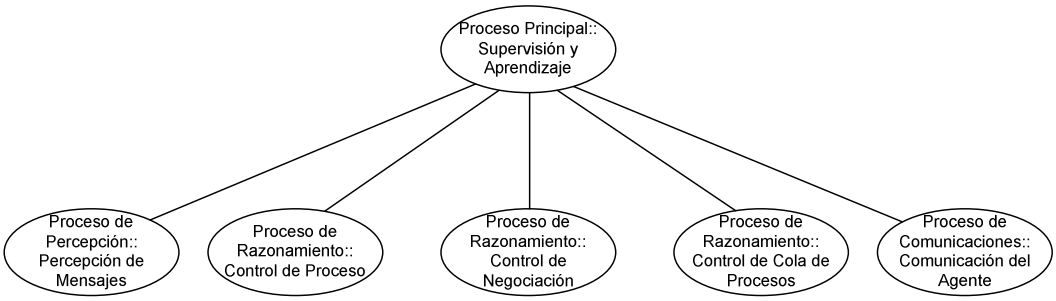


Figura 7.4. Propuesta de esquema de un agente de servicio.

Se debe dejar en claro que este esquema es sólo una estructura propuesta. La mayoría de los procesos no fueron implementados en la propuesta de validación, debido a su poca aplicabilidad en el sistema flexible del CIMUBB. Por ejemplo, el proceso de Control de Cola (el rol de ControlDeCola) no tenía sentido alguno, debido a que no existe depósito físico para implementar una cola (en realidad la cola tiene un tamaño de 1). Por otra parte, la implementación de un proceso de Aprendizaje resulta en extremo compleja, y se dejará para estudios posteriores.

7.4.2. Comunicaciones entre agentes.

Hasta ahora, el modelo es abierto en cuanto a la estrategia de comunicación entre agentes a utilizar. En efecto, los procesos de Percepción de Mensajes y Comunicación del Agente podrían ser procesos de consulta y actualización de una base de datos, como en el caso de una comunicación de *blackboard*, o de recepción y envío de mensajes a través de

[40] El aprendizaje en el agente puede realizarse de diversas formas. Por ejemplo, modificando fuentes de datos a las que los distintos procesos accedan para tomar alguna decisión o realizar cierta actividad.

una red utilizando un protocolo de alto nivel como UDP o TCP (Comer, 1999), como sería una estrategia por *paso de mensajes* (ver sección 2.3.5).

Considerando la idea de los sistemas auto-organizados, y que una de las primeras características de un agente debe ser la de autonomía (ver sección 2.3.2), se considera que el enfoque por *paso de mensajes* es mucho más representativo y útil para esto. Una de las principales ideas de los sistemas auto-organizados es que la información debe ser distribuida (con esto el concepto de *blackboard* atenta contra la idea de un sistema multi-agente real).

Se descarta la posibilidad de construir un sistema con un ambiente federado (Genesereth y Ketchpel, 1994).

Como el presente trabajo debe ser aplicado en un ambiente manufacturero, donde el ambiente es hostil, tanto para las máquinas como para los medios de comunicaciones, se debe optar por la utilización de un protocolo que asegure la llegada de todos los paquetes desde un agente a otro. Por este motivo, se utilizó el protocolo de comunicaciones TCP/IP para comunicar los agentes a través de la red del CIMUBB, con interfaces socket[41], lo que obliga a implementar interacciones con un esquema cliente/servidor entre todos los agentes conocidos (ver modelos de la sección 6.3).

7.4.3. Control con dispositivos de campo.

Con el objetivo de independizar al agente de los métodos de bajo nivel para comunicarse a través de las interfaces seriales, se generaron tres aplicaciones (no consideradas agentes debido a su ausencia en las negociaciones y a ser programas simples de consulta respuesta) que los agentes transportadores utilizan para realizar los movimientos de las unidades de campo. Estos programas fueron denominados, al igual que sus parientes de Windows utilizadas por Open-CIM, Devices Drivers, existiendo uno para cada Robot y uno para el control del Conveyor.

[41] Debido a la sencillez de las comunicaciones que deben existir, y a la falta de necesidad de generar agentes móviles, no se optó por utilizar protocolos de programación distribuida como CORBA u otros.

Los Devices Drivers de los Robots se comunican a través del puerto serial utilizando la configuración mostrada en la tabla de la sección 4.3.4, con mensajes de alto nivel pertenecientes al lenguaje ACL.

El Device Driver de control del PLC se comunica utilizando la configuración de la tabla antes mencionada, y los mensajes que se pueden ver en el ANEXO I.

7.4.4. Encontrando la ruta para el transporte de una parte.

Con algo de inspiración en las tablas de ruteo que contienen los actuales conmutadores de paquetes que operan en la Internet, se ideó un esquema de ruteo de partes a través de un sistema productivo automatizado, utilizando la estructura que a continuación se expone:

```
struct destiny{
      int   id_transport;
      int   metric;
}Destinities[NUMBER_OF_SERVICES+NUMBER_OF_TRANSPORTS]
[NUMBER_OF_SERVICES+NUMBER_OF_TRANSPORTS];
```

Esta estructura es una cuadrada de *n*x*n*, donde cada celda *i,j* corresponde al "puente" (el transporte) para movilizar la pieza desde una ubicación actual *i* a una ubicación final *j*.

En la implementación realizada esta estructura es estática, debido a que la configuración de las unidades de transporte del laboratorio CIMUBB es sencilla.

Con esto, el sistema prevee que en una posible aplicación en una empresa real, las rutas de mover una pieza desde un lugar a otro puedan ser más de una, y se tome siempre la más óptima. Si esto fuera así, sería necesaria la implementación de un mecanismo de actualización de la tabla, cada vez que sea necesario.

7.4.5. Estructuras de Datos de Representación de Agentes y Trabajos.

Para almacenar los datos de los agentes se utiliza la siguiente estructura de datos:

```
struct agent{
      int   id_agent;
      char  *dir_agent;
      char  *tipo_agent;
      int   operations[MAX_OPERATIONS_AGENTS];
} Agent[NUMBER_OF_SERVICES + NUMBER_OF_TRANSPORTS];
```

Así, se cuenta con un mecanismo complete para el almacenamiento de los datos propios de un agente, como su identificación, dirección Ip de la máquina que lo contiene, descripción del tipo de agente al que pertenece y operaciones que soporta (con esto último, un agente puede saber a quienes realizar una solicitud de servicio en una negociación).

El almacenamiento de los distintos trabajos a realizar por los agentes Servicios se realiza en estructuras como la que a continuación se muestra:

```
struct job{
      int   id_job;
      int   operations[MAX_OPERATIONS_JOB];
      int   num_operations;
      int   current_operation;
      int   delivery_date;
} jobs[MAX_JOBS];
```

Con esta estructura, se almacenan, junto con el identificador del trabajo, el conjunto de operaciones a realizar en ese trabajo, el número de operaciones que contiene la lista anterior, la operación que actualmente se le está realizando a la pieza, y la fecha de entrega.

7.4.6. Esquema resultante.

Como resultado del siguiente proyecto, se cuenta con un sistema capaz de realizar las tareas básicas que realizaba el sistema adquirido en el CIMUBB, Open-CIM, pero desde una perspectiva distribuida-heterárquica. La independización de los distintos componentes del sistema flexible de un control centralizado (como el que ejercía el CIM-Manager) trae como consecuencia un sistema mucho más robusto.

El cambio de esquema es visualizado en la Figura 7.5.

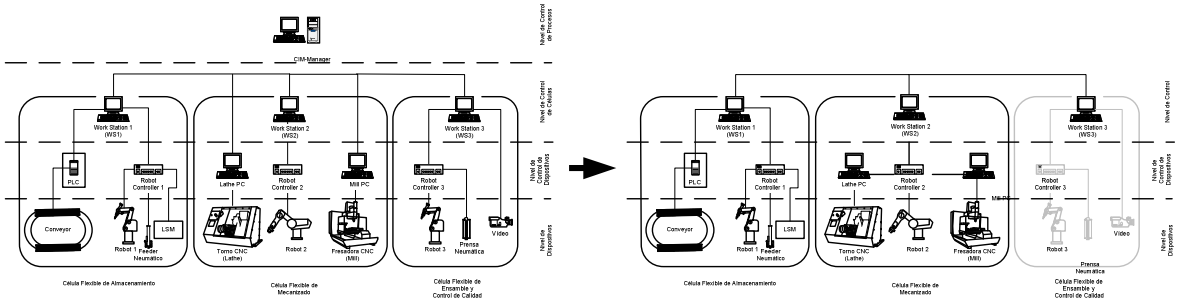


Figura 7.5. Cambio de Esquema, desde una estrategia jerarquizada a una heterárquica.

Como se aprecia en la figura, el cambio de estrategia hace que el esquema mostrado en un comienzo en el capítulo 4 acerca de la estructura jerárquica del CIMUBB, pierda una capa, ya que todo el control es ahora de nivel celular, y el comportamiento global es emergente debido a la interacción de los agentes que trabajan en los computadores principales. El control de los dispositivos de campo es ahora distribuido, teniendo cada uno de los principales un agente que controle sus acciones. El esquema de estos agentes se puede visualizar en la Figura 7.6.

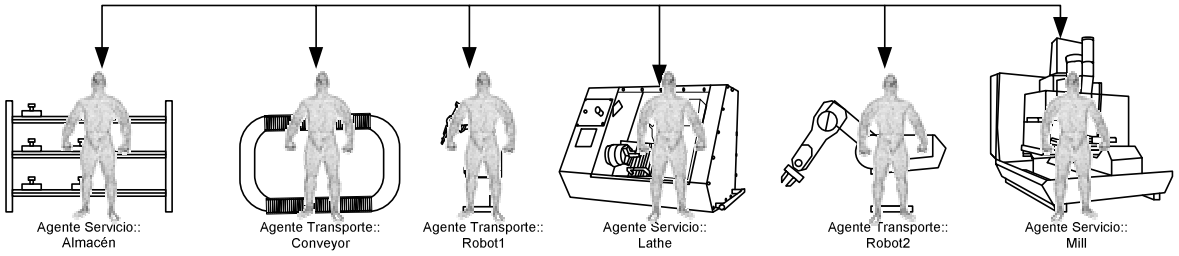


Figura 7.6. Esquema de agentes operativos en el CIMUBB.

El comportamiento del sistema en su conjunto es emergente. De hecho, no es posible afirmar con certeza qué actividades realizará cada agente previamente, aunque el conjunto de decisiones y reacciones sea preestablecido. Esto sucede porque cada agente actúa ante eventos que desconoce a priori, y el tiempo en el que ocurran estos eventos determinará cómo se comportará finalmente. De esto se puede afirmar que no sólo el comportamiento del sistema global es emergente, sino que el de los mismos agentes también. Ya que el conjunto coordinado de procesos y actividades que va realizando hacen que las tareas se vayan realizando en tiempos y secuencias no previsibles.

Así, por ejemplo, el agente Robot1 podría cargar un template con una materia prima y dejarlo inmediatamente en un pallete, si es que lo detuvo el agente Conveyor en un tiempo razonable. Si no ha llegado ningún pallete, el agente Robot1 puede bien cargar otro template con otro tipo de materia prima, aprontándose a una necesidad que verá más adelante, con lo que gana tiempo y, aparentemente, se ve que el sistema es algo más inteligente que la simple ejecución de tareas predefinidas. Se debe notar que esta decisión mencionada, así como otras tantas, no son programadas a priori, si no que resultan por la conjunción de tareas (procesos) que se realizan ante eventos asincrónicos a través del tiempo.

El funcionamiento general es de la siguiente forma:

i. Un usuario ingresa una orden de manufactura, en la que pueden ir varios trabajos con varias operaciones cada una (operaciones, este caso, de torneado

y fresado, cada una de las cuales tiene un tiempo de proceso asignado que maneja sólo el agente de la máquina que la soporta).

ii. El agente *ASRS* (que controla la salida de piezas de los feeders y la configuración actual del almacén matricial) recibe las órdenes de manufactura (rol *ControlDeCola*) y realiza un ordenamiento inicial de los trabajos por fecha de entrega (aunque la estrategia de ordenamiento en realidad podría ser cualquier otra).

iii. Una vez ordenados los trabajos comienza la negociación, en que el agente *ASRS* es el demandante de servicios (servicios de fresado o torneado) y comienza a buscar en sus registros a los posibles oferentes (rol *DemandanteDeServicio*). Se conecta con los servicios de *OfertaDeServicio* que mantienen corriendo los agentes de las máquinas (*Lathe* y *Mill*).

iv. El agente máquina (*Lathe* o *Mill*) retorna el valor del tiempo de proceso de la pieza y *ASRS* calcula el tiempo estimado de término del mecanizado (considerando además el tiempo de transporte). Se realiza entonces, el contrato definido antes en el Capítulo 5. Entonces la pieza es movilizada hacia la máquina por la interacción de los agentes Transporte: *Robot1*, *Conveyor* y *Robot2*. El agente máquina cierra entonces la posibilidad de otras negociaciones, puesto que su cola de espera es 1.

v. Al llegar la pieza es sujeta a un proceso de mecanizado[42].

vi. Una vez finalizado el mecanizado el agente máquina correspondiente cambia al rol de *DemandanteDeServicio*, y busca la unidad en la que se debería realizar la siguiente operación. Hay que decir que el puede hacer negociación incluso consigo mismo, en el caso en que él sea capaz de realizar la siguiente operación. La última operación está definida siempre como un proceso de

[42] En la implementación realizada en el CIMUBB no se utilizó mecanizado real, si no la emulación con un tiempo de retardo (delay).

almacenamiento, soportada por el agente *ASRS*, quien juega, en este caso, el papel de OferenteDeServicio. Cuando la pieza es retirada de la máquina se actualiza la cola del agente y la posibilidad de recibir solicitudes de servicio se abre nuevamente.

Esta secuencia de actividades se realiza hasta que se haya completado la orden de manufactura.

7.5 Análisis de Resultados.

El sistema se ha probado y ha demostrado cumplir eficazmente su función principal: la de controlar los dispositivos mientras se realizan las órdenes de manufactura. ¿Pero qué tan eficiente es el sistema logrado?. Una forma de probarlo será hacerlo competir, con una misma orden de manufactura, frente al actual sistema imperante de gestión: el sistema Open-CIM. Ahora, estrategias de control centralizada y heterárquica se verán las caras en una contienda que promete ser bastante interesante.

Ya se ha mencionado que una de los aspectos positivos de los sistemas de producción de control centralizados es el de la facilidad para planificar la producción. En efecto, los sistemas automatizados jerárquicos se valen de una única unidad central que genera la carta de planificación del sistema global, simplificando al máximo la capacidad desisional de las unidades subordinadas. Esto sugiere la idea que en una batalla entre jerarquía y heterarquía, la primera debería triunfar ampliamente. Esta hipótesis será puesta a prueba para finalizar el presente escrito.

La tarea a ejecutar por los contendientes será sencilla: deberán realizar tres operaciones de fresado y tres de mecanizado, con una misma configuración de almacén y con los mismos programas CNC.

Primeramente se levantó el laboratorio con el actual sistema imperante, Open-CIM. Se configuraron las definiciones de partes y las órdenes de manufactura en el sistema Open-

CIM (Figura 7.7), desde el CIM-Manager y se ejecutó la programación, considerándose como tiempo cero la presión del botón de "RUN" que tiene en su tablero, y como último tiempo, mismo criterio que se utilizaría con el sistema multi-agente, la devolución de la pieza terminada al almacén.

Los resultados son los siguientes: Open-CIM tardo 41 minutos 30 segundos en realizar todas las operaciones. El primer fresado fue terminado en el minuto 12 con 52 segundos, el primer torneado en el minuto 15 con 17 segundos, el segundo fresado en el minuto 26 con 28 segundos, el segundo torneado en el minuto 29, el tercer fresado en el minuto 40 y el tercer torneado en el minuto 41 y 30 segundos.

Desde ya se visualizó una desventaja del sistema multi-agente. El sistema producto del presente trabajo utilió programas ACL que el propio autor del trabajo programó. Por motivos de seguridad las velocidades de los movimientos de los robots fueron reducidas. Open-CIM se vale de las operaciones Pick and Place (ANEXO II) para realizar el transporte de las partes. Estos programas, parametrizados por los fabricantes del sistema utilizan velocidades por mucho más altas que las de los programas generados para probar el sistema multi-agente.

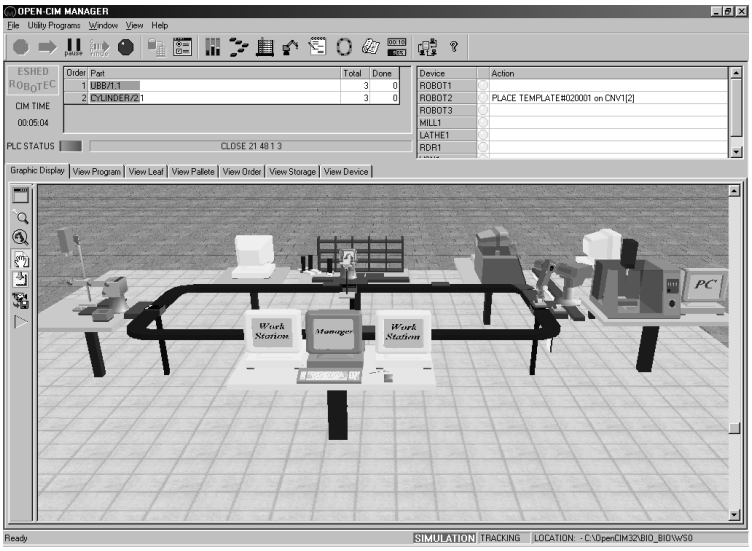


Figura 7.7. Software de Administración y Control Open-CIM.

Con este primer contra, se tuvo inicialmente la idea de manipular los resultados que de la prueba del sistema multi-agente se tuvieran, como una forma de obtener una conclusión más realista a partir de los datos, sin ventaja para ninguno. Afortunadamente, como se verá a continuación, no fue necesario.

El sistema multi-agente superó satisfactoriamente al poderoso Open-CIM. En efecto, la última pieza se devolvío al almacén al minuto 36. Una diferencia de cinco minutos, entre un sistema generado como proyecto de tesis de una universidad chilena, con respecto a un software comercial de una empresa Israelita que es comercializado en más de treinta paises alrededor del mundo. Sin duda un gran logro para el presente proyecto.

En la Figura 7.8 se visualiza esta comparación.

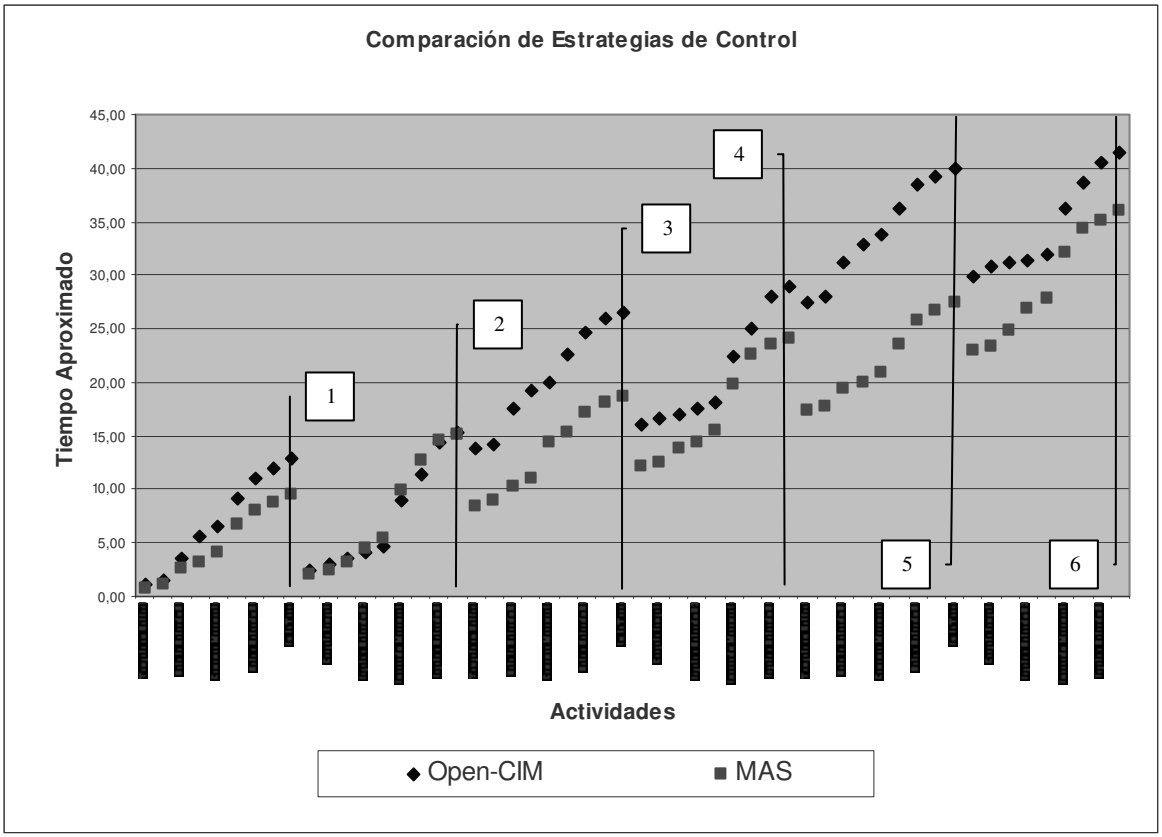


Figura 7.8. Gráfico comparativo entre las actividades realizadas por Open-CIM y el sistema Multi-Agente.

En este gráfico de dispersión, aparecen las actividades medidas para la evaluación de ambas estrategias, frente al tiempo transcurrido desde el momento cero para ambos sistemas.

Las etiquetas con números del 1 al 6 que aparecen sobre la gráfica, muestran los valores asociados a los términos de cada trabajo (hasta el momento de su almacenamiento final). Se puede observar que en todos los trabajos el sistema multi-agente fue muy superior a Open-CIM. Satisfacción inicial en los resultados, pero sólo eso.

Conclusiones y Perspectivas del Proyecto

La problemática de la programación de la producción es, sin duda, un reto para los investigadores del mundo. La complejidad del tema y la imposibilidad de encontrar solución óptima por los métodos tradicionales lo hace un reto digno de abordar. Y es que además, los problemas generados a partir de soluciones de programación off-line, que encuentran excelentes soluciones en ambientes libres de errores, tienden a fallar en instalaciones donde las concecuencias son catastróficamente caras. Se necesita entonces la concepción de nuevas formas de solucionar este problema y los sistemas auto-organizados son, hoy en día, donde se están realizando la mayor cantidad de apuestas. Quizás que depara el futuro.

Como dice el proverbio de Wili Fiallos, sitado al principio del documento que aquí culmina: "Pueblo que se resigna a tecnologías pasadas sucumbe en el campo de la ignorancia y se entierra en sus ideales". Mucho de verdad existe en esta frase.

Las necesidades del mercado obligan a construir nuevos mecanismos de gestión de los recursos, los que además de ser áltamente eficientes, deben ser robustos y minimizar en lo posible el riesgo de altos impactos a causa de los incontables eventos fortuitos del quehacer industrial.

En el presente trabajo se estudió la resolución del problema de la programación de la producción a través de la visión de los que piensan que la inteligencia artificial distribuida se proyectará como la opción escogida en un futuro. Actualmente persiste la visión centralista. Sin embargo, en el transcurso de este proyecto, se ha esclarecido la factibilidad de implementar un sistema distribuido operado por agentes de software, que generan un comportamiento completamente emergente del sistema global y del agente mismo.

El presente proyecto efectuó un barrido por los métodos más conocidos para el desarrollo de sistemas con agentes, dejando abierta la posibilidad al lector de investigar y generar opiniones más acabadas que las que pudiesen ser expuestas en el escrito. La exploración de una de las propuestas más prometedoras en el área de la ingeniería de software de agentes; la metodología GAIA en conjunción con el modelamiento con AUML.

Se ha dado opinión clara de un amplio espectro bibliográfico en cuanto a desarrollo de sistemas distribuidos inteligentes. Además de haber cumplido el objetivo inicial que es el de generar un modelo de un sistema heterárquico distribuido capaz de solucionar la problemática tantas mencionada alrededor de este documento, para luego validar la propuesta realizada con la construcción de los entes autónomos en el Laboratorio de Sistemas Automatizados de Producción CIMUBB, de la Universidad del Bío-Bío, obteniéndose interesantes resultados.

Se concluye que la factibilidad de generar sistemas con agentes para controlar un sistema flexible de manufactura es clara. Todavía queda mucho trabajo por hacer, sobre todo en cuanto a la optimización de la solución. Los sistemas auto-organizados resuelven en gran parte el problema de la robustez, sin embargo se dejan de lado muchas veces los resultados lucrativos que genera la óptima utilización de los recursos a través del tiempo. En este sentido, los algoritmos de solución centralizados *off-line* siguen siendo más poderosos.

De los resultados obtenidos en la etapa final de este proyecto no se puede decretar nada. El ambiente en el que se probó el sistema no es tan complejo como los sistemas automatizados que se pueden encontrar en industrias reales. En efecto, en el laboratorio CIMUBB de la Universidad del Bío-Bío, donde sólo existen dos máquinas, la estrategia heterárquica demostró ser mejor. Esto no implica nada. De la observación hecha durante la ejecución de las distintas pruebas es posible observar que la diferencia de velocidades está dada por las estrategias de transporte, y en ningún caso por la planificación de asignación de las máquinas. En un ambiente industrial real, ésta última debiera ser la prioridad y la que definitivamente haga la diferencia, ya que por lo general se cuenta con

grandes encolaciones de partes antes de cada máquina, con lo que el transporte de partes (el tiempo) es casi despreciable (las piezas llegan mucho antes de que se puedan comenzar a procesar).

Años de investigación abalan la idea de que algoritmos de programación off-line realizan una mejor planificación que los algoritmos de control centralizados y el motivo es obvio. Se puede predecir completamente lo que sucederá y buscar combinaciones que sean mejores. Los algoritmos genéticos, tan citados, son quizás hoy la mejor solución.

Pero la guerra no se acaba aquí. Los investigadores del mundo seguirán luchando por la generación de agentes cada vez más inteligentes y protocolos de comunicación cada vez más efectivos.

Con todo esto, quién sabe si en un tiempo más los proveedores de implementos para la industria automatizada incorporen además un agente instalable o, por qué no, incorporado en el hardware mismo de los dispositivos de campo, capaz de buscar autónomamente a sus pares en una red de máquinas y operar sin la necesidad de reconfigurar un sistema de software centralizado.

Sin duda las posibilidades del tema son muchas.

Perspectivas del Proyecto

Con la investigación e implementación realizada, se espera haber abierto un área de interés en temas de investigación similares. Explorar en otros protocolos mucho más complejos. Contract-Net es sólo uno de muchos, y en ningún caso uno de los más óptimos. Hoy en día se han realizado muchas propuestas en el ámbito de la Inteligencia Artificial Distribuida, y no se puede abandonar el interés en esto.

Las recomendaciones para trabajos posteriores a este son varias:

Primero, se espera la prueba de otros protocolos de interacción entre agentes, que aseguren un mejor rendimiento que el ya expuesto. La aplicación realizada podría ser mejor probada si se incorporaran nuevas aplicaciones en el laboratorio, por ejemplo, espacios físicos para implementar colas en las máquinas.

Mecanismos para que los agentes sean más reactivos a cambios en el entorno son necesarios. Si bien el modelo propuesto en este trabajo considera los errores en la ejecución de ciertas tareas, no están probados mecanismos para detección de fallas, tanto en máquinas como en procesos.

Trabajos multi-disciplinarios deben ser fomentados, generando agentes mucho más eficientes que incorporen otras heurísticas o meta-heurísticas, o intentando embeber el código del agente en el mismo hardware. Por qué no, implementar un conjunto de agentes físicos. Robots y máquinas, con un controlador hardware, completamente autónomas y concientes de su ambiente. Esto es, sin duda, un reto interesante de adoptar.

Se recomienda además trabajar en el área de la ingeniería de software de agentes y, por qué no, realizar aportes en la amplia gama de metodologías utilizadas para el desarrollo de sistemas distribuidos auto-organizados. Tal vez, el lector podría encontrar interesante corregir algunos detalles de las notaciones que hacen que el diseño sea insuficientemente

claro para su entendimiento cabal, y que presente una real ayuda en el proceso de construcción.

En detalles técnicos, tal vez, una gran adelanto, y un real aporte, podría darse en el ámbito de las comunicaciones. TCP como protocolo de capa de transporte es un excelente mecanismo para su aplicación en ambientes industriales operados por computadora, debido a que asegura la entrega de los paquetes a un destinatario final. Sin embargo, la utilización de TCP en los sistemas con agentes puede ser una molestia, ya que no se puede perder el esquema cliente/servidor, tan común en estos sistemas. UDP es una solución descartable, por no poder asugurar la entrega de los mensajes. Es necesario un protocolo que rompa el esquema cliente/servidor y de la posibilidad de generar mensajes multi-cast, por ejemplo, y mensajería asincrónica desde el momento en que se ejecuta el agente hasta su muerte, sin conocer a priori direcciones de red o puertos de comunicación, pero que además sea seguro como TCP. Aquí hay mucho trabajo por hacer.

Sólo recalcar, para finalizar este trabajo, que la tecnología de agentes está en boga hoy en día, y que probablemente siga mucho tiempo así. El trabajo que ahora termina sirvió como una forma de esclarecer los aspectos teóricos que se han desarrollado, además de ilustrar la forma de consebir y contruir un sistema de este tipo. Ha de esperarse, que en un futuro se convierta en paradero obligatorio para otras investigaciones en el área de la, todavía, adolecente Inteligencia Artificial Distribuida.

Referencias

ADMINISTRACIÓN DE producción y operaciones: manufactura y servicios. 2000. By Richard B. Chase "et al". 8ª ed. Santa Fé de Bogotá, MacGraw-Hill. 885p.

ARENAS, A. E. and BARRERA-SANABRIA, G. 2002. Applying the MAS-CommonKADS Methodology to the Flights Reservation Problem: Integrating Coordination and Expertise. In: PROCEEDINGS OF the Fifth Joint Conference on Knowledge-Based Software Engineering (JCKBSE 2002). Maribor, Slovenia. 10p.

ARENAS, A. E., GARCÍA-OJEDA, J. C. and PÉREZ A., J. de J. 2004. On combining organisational modelling and graphical languages for the development of multiagent systems. Integrated Computer Aided Engineering, 11(2):151-163.

BAUER, B. 2001. UML Class Diagrams Revisited in the Context of Agent Based-Systems. In: PROCEEDINGS OF Agent-Oriented Software Engineering (AOSE 01). Montreal, Canada. Springer Verlag. 8p.

BOTTI N., V. J. and GIRET B., A. 2003. Aplicaciones Industriales de los Sistemas Multiagente. Valencia, Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación. 59p.

BUSSMAN, S., JENNINGS, N. R. and WOOLDRIDGE, M. 2001. On the Identification of Agents in the Design of Production Control Systems. In: CIANCARINI, P. and WOOLDRIDGE M. (Eds.). *Agent-Oriented Software Engineering*. Springer Verlag. pp. 141-162.

CHARPENTIER, P., MUHL, E. 2004. From a reactive, heterarchical to a holonic system: an application for optimizing flow in an automotive plant. *Production Planning & Control*, 15(2):166-177.

CHEUNG, H. M. E., YEUNG, W. H. R. and FUNG, S. T. R. 2000. HSCF: a holonic shop floor control framework for flexible manufacturing systems. *International Journal of Computer Integrated Manufacturing*, 13(2):121-138.

CHOI, K.-H., BAE, C.-H. and LEE, S.-H. 2003. Baheviour modelling and control of computer integrated manufacturing. *International Journal of Computer Integrated Manufacturing*, 16(2):128-139.

CIMUBB. Laboratorio de Sistemas Automatizados de Producción. [on line] http://www.cimubb.ubiobio.cl [consulta: 12 de diciembre de 2004].

COMER, D. E. 1996. Redes Globales de Información con Internet y TCP/IP. Principios básicos, protocolos y arquitectura. 3ª Edición. México, Prentice-Hall Hispanoamericana. 621p.

COMPUTER INTEGRATED Manufacturing and Engineering. 1993. By Rembold, U. "et al". Workingham, Addison-Wesley Publishing Co.. 640p.

DAVIDSSON, P. and WERNSTEDT, F. 2002. Characterization and Evaluation of Just-In-Time Production and Distribution. In: PROCEEDINGS OF the AAMAS Workshop on MAS Problem Spaces and Their Implications to Achieving Globally Coherent Behavior. 13p.

DEMAZEAU, Y. 1995. From interactions to collective behaviour in agent-based systems. In: FIRST EUROPEAN Conference on Cognitive Science. Saint-Malo, France. 15p.

DUFFIE, N. A. and PRABHU, V. 1996. Heterarchical control of highly distributed manufacturing systems. *International Journal of Computer Integrated Manufacturing*, 9(4):270-281.

DURFEE, E. H. and ROSENSCHEIN, J. S. 1994. Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples. In: PROCEEDINGS OF the 13th International Distributed Artificial Intelligence Workshop. 10p.

ESHED ROBOTEC. 2000. Communication Layout for Bio Bio University [AutoCAD file]. Modelo bidimensional

FIPA. FIPA Communicative Act Library Specification. [on line] http://www.fipa.org/specs/fipa00037/PC00037E.html [consulta 13 de marzo de 2005].

FRANKLIN, S. and GRAESSER, A. 1996. Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents. In: PROCEEDINGS OF the Third

International Workshop on Agent Theories, Architectures, and Languages. Springer-Verlag, Berlín, Alemania. 10p.

FRAYRET, J-M. 2002. A Conceptual Framework To Operate Collaborative Manufacturing Networks. Thèse de Philosophiae Doctor (Ph.D.). Québec, Université Laval, Faculté des Sciences et de Génie. 385p.

FRAYRET, J-M., D'AMOURS, S., MONTREUIL, B. 2004. Coordination and control in distributed and agent-based manufacturing systems. *Production Planning & Control*, 15(1):42-54.

GARCÍA-OJEDA, J. C., PÉREZ-ALCÁZAR, J. de J. and ARENAS, A. E. 2002. Applying GAIA and AUML to the Selective Dissemination of Information on the Web. In: FOURTH IBEROAMERICAN Workshop on Multi-Agent Systems: Iberagents 2002 (Iberamia 2002). Sevilla, España. 16p.

GARCÍA-OJEDA, J. C., PÉREZ-ALCÁZAR, J. de J. and ARENAS, A. E. 2004. Extending the Gaia Methodology with Agent-UML. In: AAMAS 2004. 2p.

GENESERETH, M. R. and KETCHPEL, S. P. 1994. Software Agents. *Communication of the ACM*, 37(7):48-53.

GLASER, N. 1997. The CoMoMAS Approach: From Conceptual Models to Executable Code. 16p.

GÓMEZ S., J. J. 2002. Modelado de Sistemas Multi-Agente. Tesis Doctoral. Madrid, Universidad Complutense de Madrid, Facultad de Informática. 249p.

GÓMEZ S., J. J. 2003. Metodologías para el desarrollo de sistemas multi-agente. *Revista Iberoamericana de Inteligencia Artificial*, (18):51-63.

GÓMEZ-S., J. J. and FUENTES, R. 2002. The INGENIAS Methodology. In: FOURTH IBEROAMERICAN Workshop on Multi-Agent Systems: Iberagents 2002.

GROOVER, M. P. 1980. Automation, Production Systems, and Computer Integrated Manufacturing. Englewood Cliffs - NY, Prentice-Hall. 808p.

GUESSOUM, Z. and BRIOT, J.-P. 1999. From active objects to autonomous agents. *IEEE Concurrency*, 7(3):68-76.

HENESEY, L. E., NOTTEBOOM, T. E. and DAVIDSSON, P. 2003. Agent-based simulation of stakeholders relations: An approach to sustainable port terminal

management. In: INTERNATIONAL ASSOCIATION of Maritime Economists Annual Conference. Busan, Korea. 17p.

IGLESIAS F., C. A. 1998. Definición de una Metodología para el Desarrollo de Sistemas Multiagente. Tesis Doctoral. Madrid, Universidad Politécnica de Madrid, Departamento de Ingeniería de Sistemas Telemáticos. 293p.

INTELITEK. [on line]. http://www.intelitek.com [consulta: 20 de diciembre de 2004].

JENNINGS, N. R. and BUSSMANN, S. 2003. Agent-Based Control Systems – Why Are They Suited to Engineering Complex Systems?. *IEEE Control Systems Magazine*, 23(3):61-74.

KIM, C-O. and NOF, S. Y. 2000. Investigation of PVM for the emulation and simulation of a distributed CIM workflow system. *International Journal of Computer Integrated Manufacturing*, 13(5):401-409.

KINNY, D., GEORGEFF, M. and RAO, A. 1996. A methodology and modelling technique for systems of BDI agents. In: VAN DE VELDE, W. and PERRAM, J. W. (Eds.). Agent Breaking Away: Proceedings of the Seventh European Workshop on Modeling Autonomous Agents in a MultiAgent World. (LNAI vol. 1038). Berlin, Germany, Springer Verlag, pp. 56-71.

KOLLER L., R. A. 2004. Un Enfoque de Sistemas Multi-Agentes para la Programación de la Producción en la Industria del Aserrío. Tesis de Magíster. Universidad del Bío-Bío, Facultad de Ingeniería. 79p.

LANGER, G. and ALTING, L. 2000. Trends and Perspectives - An Architecture for Agile Shop Floor Control Systems. *Journal of Manufacturing Systems*, 19(4):267-281.

MAES, P. 1994. Modeling Adaptive Autonomous Agent. In: LANGTON, C. G. (editor). *Artificial Life: An Overview*. The MIT Press, Cambridge, MA. pp. 135-162.

MATURANA, F., SHEN, W. and NORRIE D. H., 1999. MetaMorph: An Adaptive Agent-Based Architecture for Intelligent Manufacturing. *International Journal of Production Research*, 37(10):2159-2173.

MORALES O., F. J. 2003. Modelamiento Orientado a Objeto del CIM-UBB. Tesis de título de Ingeniero Civil en Informática. Concepción, Universidad del Bío-Bío, Facultad de Ciencias Empresariales. 108p.

NWANA, H. S. 1996. Software Agents: An Overview. *Knowledge Engineering Review*. 11(3):1-40.

OBJECT-ORIENTED Modeling and Design. 1991. By James Rumbaugh "et al". Englewood Cliffs, N. J., Prentice-Hall. 500p.

ODELL, J., PARUNAK, H. V. D. and BAUER, B. 2000a. Extending UML for Agents. In: PROCEEDINGS OF the Agent-Oriented Information Systems Workshop. pp. 3-17.

ODELL, J. J., PARUNAK, H. V. D. and BAUER, B. 2000b. Representing Agent Interaction Protocols in UML. In: PROCEEDINGS OF the AGENTS'2000. Barcelona, Spain. 21p.

ODELL, J., PARUNAK, H., FLEISHER, M. and BRUECKNER, S. 2002. Modeling Agents and their Environment. In: PROCEEDINGS OF AOSE 2002. Bologna, Italy. 18p.

PARUNAK, H. V. D. 1988. Distributed Artificial Intelligence Systems. In: KUSIAC, A. (editor). *Artificial Intelligence – Implications for CIM*. IFS Ltda./Springer Verlag. pp. 225-251.

PARUNAK, H. V. D. 1994. Chapter 4: Applications of Distributed Artificial Intelligence in Industry. In: O'HARE, P., and JENNINGS, N. R. (Eds.). *Foundations of Distributed Artificial Intelligence*. Wiley Inter-Science. pp. 139-163.

PARUNAK, H. V. D. 1997. Go to the Ant: Engineering Principles from Natural Multi-Agent Systems. In: ANNALS OF Operations Research. 27p.

PARUNAK, H. V. D. 1998a. Characterizing Multi-Agent Negotiation. In: INTERNATIONAL WORKSHOP on Multi-Agent System. 9p.

PARUNAK, H. V. D. 1998b. Practical and Industrial Applications of Agent-Based Systems. 41p.

PARUNAK, H. V. D. and ODELL, J. 2002. Representing Social Structures in UML. In: WOOLDRIDGE, M., WEISS, G. and CIANCARINI, P. (Eds.). *Agent-Oriented Software Enginnering II*. Lecture Notes on Computer Science volume 2222, Springer-Verlag, Berlín, pp. 1-16.

PARUNAK, V., SAUTER, J. and CLARK S. 1998. Toward the Specification and Design of Industrial Synthetic Ecosystems. In: SINGH, M. P., RAO, A. and

WOOLDRIDGE, M. J. (Eds.). Intelligent Agents IV: Agent Theories, Architectures, and Languages. Springer Verlag. Berlin. pp. 45-59.

PATRITI, V., SCHÄFER, K., RAMOS, M., CHARPENTIER, P., MARTIN, P. and VERON, M. 1997. Multi-agent and manufacturing. A multilevel point of view. In: Proceedings of CAPE'97. Detroit, USA. 8p.

PINEDO, M. 1995. Scheduling: Theory, Algorithms, and Systems. Englewood Cliff., N.J.: Prentice-Hall. 378p.

POLAKOFF, J. C. 1991. Computer Integrated Manufacturing: A New Look at Cost Justifications. In: MABERT, V. A. and JACOBS, F. R. (Eds.). Integrated Production Systems: Design, Planning, Control, and Scheduling. 4ª edición. Norcross, Georgia, Institute of Industrial Engineers. pp. 107-116.

PRESSMAN, ROGER S. 2002. Ingeniería del Software. Un enfoque práctico. Madrid, McGraw-Hill. 601p.

RAMOS M., M. 1998. Vers un pilotage auto-organisant de Systèmes Flexibles de Fabrication. Thèse de Docteur. Nancy, Université Henri Poincaré, Faculté des Sciences. 132p.

RAMOS M., M., CHARPENTIER, P. and MARTIN, P. 1997. Reactive Distributed and self-organized scheduling of Flexible Manufacturing Systems. In: ANNALS OF the 7th Euro Conference Decision Support Systems: march 24-27 1997. Bugres, Belgium. 10p.

REHG, J. A. 1994. Computer-Integrated Manufacturing. Englewood Cliffs - NJ, Prentice-Hall. 460p.

RODRÍGUEZ M., P. and IBACACHE R., L. 2002. El Lenguaje de Programación C. 152p.

RUSSELL, S. and NORVIG, P. 1996. Inteligencia Artificial: un enfoque moderno. México, Prentice Hall Hispanoamericana. 979p.

STARR, M K. 1989. Managing Production and Operations. Englewood Cliffs – NJ, Prentice-Hall. 646p.

SHAHIN, R. 2002. A Practical Representation of Holonic Manufacturing Systems. In: BASYS 2002. Cancún, México. 8p.

SUESSMAN, B., COLOMBO, A. W. and NEUBERT, R. 2002. An Agent-Based Approach Towards the Design of Industrial Holonic Control Systems. In: BASYS 2002. Cancún, México. 8p.

THARUMARAJAH, A., WELLS, A. J. and NEMES, L., 1996. Comparison of the bionic, fractal and holonic manufacturing system concepts. *International Journal of Computer Integrated Manufacturing*, 9(3):217-226.

THE UNIFIED Modeling Language: User Guide. 1999. By Ivar Jacobson "et al". Reading, Massachusetts, Addison-Wesley. 482p.

TRUEFORCE ™. History Timeline of Robotic. [on line] http://trueforce.com/Articles/Robot_History.htm [consulta: 06 de mayo de 2004]

WAN, Y.-W. 1993. Which is Better: Off-Line or Real-Time Scheduling?. *Me Research Bulletin*, Department of Manufacturing Engineering, City Polytechnic of Hong Kong, 1(1):125-133.

WANG, L., BALASUBRAMANIAN, S. and NORRIE, D. H. 1998. Agent-based intelligent control systems design for real-time distributed manufacturing environments. In: AGENT - BASED manufacturing Workshop – Autonomous Agents'98: may 9-13. Minneapolis, St. Paul. 8p.

WOOLDRIDGE, M. 1997. Agent-Based Software Engineering. *IEEE Proceedings on Software Engineering*, 144(1):26-37.

WOOLDRIDGE, M. and JENNINGS, N. R. 1995a. Agent Theories, Architectures, and Languages: A Survey. In: *Intelligent Agents: Theories, Architectures, and Languages*. Springer-Verlag. Heidelberg, Alemania. pp. 1-39.

WOOLDRIDGE, M. and JENNINGS, N. R. 1995b. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115-152.

WOOLDRIDGE, M., JENNINGS, N. R. and KINNY, D. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285-312.

WUNDERLI, M., NORRIE, M. C. and SCHAAD, W. 1996. Multidatabase agents for CIM systems. *International Journal of Computer Integrated Manufacturing*, 9(4):293-298.

ZAMBONELLI, F., JENNINGS, N. R. and WOOLDRIDGE, M. 2003. Developing Multiagent Systems: The Gaia Methodology. *ACM Trans on Software Engineering and Methodology*, 12(3):317-370.

ANEXO I. Comunicación con PLC

Para realizar comunicación con el PLC instalado en el Laboratorio CIMUBB, se debe setear una interfaz de comunicación con el puerto serial (vía protocolo RS232), por el primer puerto de comunicaciones[xliii], con la siguiente configuración:

Baud Rate (velocidad de transmisión)	=	9600 Baudios
Bits de Datos (Data Bits)	=	7
Paridad (Parity)	=	Even (par)
Stop Bits (Bits de Parada)	=	2

El control del Conveyor a través de este PLC, requiere manejar solamente tres tipos de mensaje (todos los cuales están compuestos por una palabra de varios valores hexadecimales que siguen al símbolo arroba): mensajes para saber cuando un pallete pasa por alguna estación de trabajo, mensajes para detener un pallete en alguna estación de trabajo y mensajes para soltar el pallete cuando corresponda.

Mensajes de paso de Palletes por Estaciones de Trabajo.

Cuando un Pallete pasa por una estación de trabajo, los sensores de la unidad de detención retornan al PLC el valor del número de identificación de éste (lo cuál se realiza mediante sensores magnéticos). El PLC devuelve al ordenador una palabra compuesta por el símbolo arroba (@), seguido por varios valores hexadecimales. Lo relevante del mensaje (en el caso de la configuración del laboratorio CIMUBB) es el octavo y doceavo valor hexadecimal, los que indican la estación y el número de pallete respectivamente. Así, el mensaje de retorno

[xliii] COM1 en Windows y tS0 en Linux.

@00EX000**3**000**1**00025D*

Puede ser interpretado como "ha pasado el pallete número 1 por la estación de trabajo número 3" (PASS S#3 P#1).

Mensajes de Detención de Palletes.

Para detener un Pallete en una estación en particular se debe enviar al PLC, por medio del puerto serial, un mensaje compuesto por el símbolo arroba, más una cadena de 14 valores hexadecimales, un asterisco (*) y un retorno de carro (el carácter 13). De este mensaje, lo relevante son el séptimo y octavo valores decimales, que deben formar los valores 96, 97 y 98, para referirse a las estaciones 1, 2 y 3, respectivamente. Así, para detener un pallete en las estaciones de trabajo, podrían ser enviados los siguientes mensajes:

@00WD00**96**00015D*↲ (Detener un pallete en la estación #1)

@00WD00**97**00015C*↲ (Detener un pallete en la estación #2)

@00WD00**98**000153*↲ (Detener un pallete en la estación #3)

Mensajes para Soltar un Pallete

Similar a los mensajes para obtener un Pallete en una estación de trabajo, los mensajes necesarios para soltarlo cuando ya está detenido están compuestos por los mismos valores, y al igual que en el caso anterior, los valores relevantes son el séptimo y octavo valor hexadecimal, los que deben formar los valores 48, 49 y 50, para indicar al PLC que suelte el pallete detenido en la estación 1, 2 y 3 respectivamente. Así, para soltar un pallete en las estaciones de trabajo, podrían ser enviados los siguientes mensajes:

@00WD004800015E*↲ (Soltar el pallete en la estación #1)

@00WD004900015F*↵ (Soltar el pallete en la estación #2)

@00WD0048000157*↵ (Soltar el pallete en la estación #3)

ANEXO II. Lenguaje ACL.

El lenguaje de programación ACL es un estándar utilizado actualmente para la programación de robots.

Inicialmente se pretendía utilizar programas Pick and Place para la utilización de los robots, sin embargo, finalmente se tendió a la programación desde cero de rutinas de movimientos en ACL.

Básicamente todo programa ACL comienza escribiéndose en una pantalla terminal conectándose con el Robot Controller a través de alguna puerta serial. En particular, en el laboratorio CIMUBB, estas puertas son configuradas con los siguientes parámetros: Baudios=9600, Bits de Datos=8, Sin paridad y 1 como bit de parada.

El comando ACL "*edit + nombre del programa"* ordena al robot controller disponer de una shell muy básica en la que se puede ir editando un programa ACL.

El lenguaje ACL es complejo, y se puede realizar un conjunto importante de operaciones de comunicaciones con dispositivos y almacenamiento de datos. En particular en el trabajo del cual se desprende este anexo, básicamente se utilizaron las siguientes instrucciones ACL: *attach* (atachar un vector de posisiones para modificarlo), *edit* (crear o editar un programa), *move* (realizar movimiento articular del brazo robótico), *movel* (realizar movimiento articular del brazo robótico), *moveld* (realizar movimiento lineal con espera de comando final), *moved* (realizar movimiento articular con espera de comando final), *open* (abrir pinzas), *close* (cerrar pinzas), *gosub* (llamada a sub-programas), *run* (ejecutar un programa), *set* (crear o modificar una variable) y *println* (retornar un mensaje de texto).

Nota: las instrucciones denominadas move son realizadas llamando a posisiones espaciales del brazo robótico previamente grabadas en un vector. En el caso de los Pick & Place de Open-CIM, los vectores son denominados CIM y CIMB para el brazo y el eje horizontal, respectivamente.

Operaciones Pick and Place

Pick and Place es el nombre que el modelo Open-CIM da a los programas parametrizados construidos en lenguaje ACL para el traslado de partes con los brazos robóticos.

Codificación de las partes.

Los principales códigos asignados en el sistema Open-CIM a las partes o piezas utilizadas en el CIMUBB aparecen descritas en la tabla siguiente:

Tabla 1: Codificación de Open-CIM de las partes más utilizadas

Código	Parte	Descripción
0	TEMPLATE	Bandeja para fijar materias primas o piezas con algún mecanizado, y facilitar su manipulación y transporte.
1	BOX_SUPPLY	Pieza de madera rectangular destinada a procesos de fresado.
11	CYLINDER_SUPPLY	Pieza cilíndrica destinada a procesos de torneado.
21	CYLINDER_FOR_VISION	Cilindro metálico con marcas de aceptación y falla en sus caras, utilizadas para realizar pruebas de control de calidad por visión artificial.
31	BASE_FOR_VISION	Base rectangular ahuecada para ensamblarla con un CYLINDER_FOR_VISION en la estación de ensamble y control de calidad y así facilitar su fijación en la prensa para tomar las imágenes pertinentes para el proceso de visión.

Codificación de Fuentes y Destinos

Open-CIM codifica cada uno de los lugares identificados en las estaciones y las máquinas de control numérico como aparece en la siguiente tabla:

Tabla 2: Codificación de Open-CIM de fuentes y destinos.

Célula	Código	Lugar	Rango de Índices	Descripción
Ninguna (Dispositivo de uso general)	1	CNV1 (Conveyor)	1	Corresponde a un pallete detenido en una de las estaciones de trabajo desde donde el robot puede coger una parte o ponerla en este sitio.
Célula 1 (almacén)	12	BFFR1 (Buffer1)	1-2	Corresponde a los buffers o estaciones temporales ubicadas en la célula 1 (de almacenamiento) junto a la cinta transportadora. Estos "espacios" son utilizados por el robot para almacenar temporalmente un template y cargar sobre ellos las piezas requeridas.
Célula 1	13	RDR1 (Reader)	1	Corresponde al lector de códigos de barra instalado en la célula de almacenamiento. Este código le indica al brazo robótica que debe posicionar el template que sostenga en ese momento en una posición frente al reader, lugar donde se realiza la la lectura del código de barra.
Célula 1	14	NXMAS1 (Almacén)	1-15	Corresponde al almacén ASRS. El rango de índices indica la celda del almacén matricial, organizado en 3 filas de 5 columnas cada uno.
Célula 1	15	FDR1 (Feeder 1)	1	
Célula 1	16	FDR2 (Feeder 2)	1	
Célula 1	17	FDR3 (Feeder 3)	1	
Célula 1	18	RACK1	1-6	Corresponde a un rack ubicado junto al robot, en el cual se pueden depositar hasta 6 piezas de distintos materiales.
Célula 1	19	RACK2	1-6	Idem a RACK1
Célula 2 (mecanizado)	22	BFFR2 (Buffer 2)	1-2	Corresponde a los buffers o estaciones temporales ubicadas en la célula 2 (de

				mecanizado) junto a la cinta transportadora. Estos "espacios" son utilizados por el robot para almacenar temporalmente un template y cargar sobre ellos las piezas requeridas.
Célula 2	23	MILL1 (Fresadora)	1	Corresponde a la fresadora CNC. Al seleccionar este lugar, el brazo robótica toma o deja una pieza cilíndrica dentro del torno, en el lugar dispuesto para su sujeción.
Célula 2	24	LATHE1 (Torno)	1	Corresponde al torno CNC. Al seleccionar este lugar, el brazo robótica toma o deja una pieza rectangular en (o desde) la prensa del torno, donde se fija para su mecanizado.
Célula 3	32	BFFR3 (Buffer 3)	1-2	Corresponde a los buffers o estaciones temporales ubicadas en la célula 3 (de ensamble y control de calidad) junto a la cinta transportadora. Estos "espacios" son utilizados por el robot para almacenar temporalmente un template y cargar sobre ellos las piezas requeridas.
Célula 3	34	JIG1 (Prensa)	1	Corresponde a la prensa neumática que fija las partes a analizar mediante control de calidad.
Célula 3	35	RACK3	1-6	Ídem a Rack 1 y 2
Célula 3	36	RACK4	1-6	Ídem a Rack 1, 2 y 3
Célula 3	37	TRASH1	1	Corresponde al compartimiento destinado a depositar las piezas que no hayan cumplido con el control de calidad.

Ejecución de un PICK AND PLACE en ACL.

Pick and Place (seleccionar y posicionar) es el nombre de los programas básicos escritos en lenguaje ACL de los Robot Controller. Están compuestos por un conjunto de llamadas a posiciones espaciales de un brazo robot, grabadas previamente en vectores en un controlador específico.

Los programas Pick and Place se deben ejecutar en el orden y según las especificaciones que se exponen en la siguiente tabla:

Tabla 3. Ejecución de un programa PICK AND PLACE.

Nº	Flujo del Mensaje	Mensaje (Comando)	Descripción
1	Desde el ordenador al Robot Controller	**run PCPLC↵**	Ordena la ejecución del programa PCPLC (Pick and Place)
2	Desde el Robot Controller al ordenador	**?**	Solicita valor de siguiente parámetro.
3	Desde el ordenador al Robot Controller	Un número de 6 cifras correspondiente al código de la operación a ejecutar. Por ejemplo: **220000↵**	Se ingresa el código de la operación a ejecutar. Esto se realiza con el objetivo de asignar un identificador a la operación en curso. El código debe ser un número de 6 dígitos.
4	Desde el Robot Controller al ordenador	**%PAR? ↵** **?**	Solicita valor de siguiente parámetro.
5	Desde el ordenador al Robot Controller	El código de la parte que el robot va a tomar. Por ejemplo **0↵** (correspondiente a un TEMPLATE)	Código de la parte que se va a tomar. Los valores válidos especificados en el sistema Open-CIM, se encuentran especificados en la Tabla 1
6	Desde el Robot Controller al ordenador	**?**	Solicita valor de siguiente parámetro.
7	Desde el ordenador al Robot Controller	El código del lugar desde el cuál se debe tomar la parte. Por ejemplo: **14↵** (correspondiente al almacén, NXMAS1)	Código del lugar de origen de la parte a tomar. Los valores válidos especificados en el sistema Open-CIM, se encuentran especificados en la Tabla 2.
8	Desde el Robot Controller al ordenador	**?**	Solicita valor de siguiente parámetro.
9	Desde el ordenador al Robot Controller	El código de la posición específica en el lugar donde se va a coger la pieza. Por ejemplo: **1↵** (correspondiente a la primera celda del almacén).	Código de la posición específica de la parte que el robot va a coger, dentro de un conjunto de posiciones o subdivisiones que puede tener el objeto o dispositivo que la contenga. Los valores válidos especificados en el sistema Open-CIM, se encuentran especificados en la Tabla 2.

10	Desde el Robot Controller al ordenador	**?**	Solicita valor de siguiente parámetro.
11	Desde el ordenador al Robot Controller	El código del lugar en el cuál se debe va a depositar la parte. Por ejemplo: **12↵** (correspondiente al Buffer de la primera estación, BFFR1)	Código del lugar de destino de la parte asida. Los valores válidos especificados en el sistema Open-CIM, se encuentran especificados en la Tabla 2.
12	Desde el Robot Controller al ordenador	**?**	Solicita valor de siguiente parámetro.
13	Desde el ordenador al Robot Controller	El código de la posición específica en el lugar donde se va a depositar la pieza. Por ejemplo: **1↵** (correspondiente a la primera celda del Buffer de la primera estación).	Código de la posición específica de la parte que el robot va a coger, dentro de un conjunto de posiciones o subdivisiones que puede tener el objeto o dispositivo en el que se va a depositar la parte. Los valores válidos especificados en el sistema Open-CIM, se encuentran especificados en la Tabla 2.
14	Desde el Robot Controller al ordenador	**?**	Solicita valor de siguiente parámetro.
15	Desde el ordenador al Robot Controller	**↵**	Se envía un salto de carro para indicar que no se enviarán más parámetros.
16	Desde el Robot Controller al ordenador	**>**	El Robot Controller indica que puede recibir más instrucciones.
17	Desde el ordenador al Robot Controller	La orden de ejecutar el programa "GET" para obtener la pieza según los parámetros de origen ingresados anteriormente. Por ejemplo: **run GT014↵** (correspondiente al programa "GET" para tomar una pieza desde el almacén).	El código del programa "GET" (obtener) para coger la parte especificada en el primer parámetro ingresado luego del código de la operación, desde la posición especificada en los parámetros descritos de origen. Este corresponde a un programa almacenado con el nombre de GT más un número de 3 dígitos correspondiente al código del lugar de origen (Tabla 2). ***Nota:*** *En este momento el robot comienza a moverse en busca de la pieza especificada.*
16	Desde el Robot Controller al ordenador	**>**	El Robot Controller indica que puede recibir más instrucciones.
18	Desde el ordenador al Robot Controller	La orden de ejecutar el programa "PUT" para posicionar la pieza tomada según los parámetros de	El código del programa "PUT" (poner) para dejar la parte tomada en la posición especificada en los parámetros descritos

		origen ingresados anteriormente. Por ejemplo: **run PT012↵** (correspondiente al programa "PUT" para dejar una pieza en el Buffer 1).	de destino. Este corresponde a un programa almacenado con el nombre de PT más un número de 3 dígitos correspondiente al código del lugar de origen (Tabla 2). ***Nota:*** *El controlador del robot esperará a que termine la operación "GET" y, seguidamente, comenzará la ejecución del programa "PUT".*
19	Desde el Robot Controller al ordenador	La palabra "%START" más el código de la operación. Por ejemplo: **%START 220000↵** (comienza la operación de código 220000)	Una vez que comienza el programa "PUT" la palabra %START más el código de la operación indica el comienzo de la tarea especificada.
19	Desde el Robot Controller al ordenador	Las palabras "%FINISH" y "%END" más el código de la operación. Por ejemplo: **%FINISH 220000↵** **%END 220000↵** (termina la operación de código 220000)	Una vez que el programa "PUT" ha finalizado las palabras %START y "%END" más el código de la operación, indica el fin de la tarea especificada.
16	Desde el Robot Controller al ordenador	**>**	El Robot Controller indica que puede recibir más instrucciones.