UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES

Centro de Ciencias Básicas

Depto. de Ciencias de la Computación

Arquitectura de Sistemas Multi-Agente Orientada a Aplicaciones Industriales en Tiempo Real

TESIS QUE PRESENTA:

Luis Fernando Gutiérrez Marfileño

PARA OBTENER EL GRADO DE:

Doctor en Ciencias de la Computación-Inteligencia Artificial

ASESORES DE TESIS:

Dra. Eunice Esther Ponce de León Sentí(UAA)
Dr. Felipe Padilla Díaz (UAA)
Dr. Alejandro Padilla Díaz (UAA)
Dra. Elva Díaz Díaz (UAA)
Dra. Ma. de Lourdes Yolanda Margain Fuentes (UPA)

REVISORA:

Dra. Cora Beatriz Excelente Toledo (LANIA)

18 de junio de 2010

Centro de Ciencias Básicas
UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

M. E.S. LUIS FERNANDO GUTIÉRREZ MARFILEÑO
PASANTE DEL DOCTORADO EN CIENCIAS EXACTAS, SISTEMAS Y DE LA INFORMACIÓN
P R E S E N T E .

Estimado (a) Alumno (a) Gutiérrez:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis titulado: **"Arquitectura de Sistemas Multi-Agente Orientada a Aplicaciones Industriales en Tiempo Real",** hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., 18 de junio de 2010
"LUMEN PROFERRE"
EL DECANO

DR. FRANCISCO JAVIER ÁLVAREZ RODRÍGUEZ

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES
CENTRO DE CIENCIAS BÁSICAS

c.c.p.- Archivo

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Aguascalientes, Ags. Junio de 2010

Dr. Francisco Javier Álvarez Rodríguez
Decano del Centro de Ciencias Básicas
Universidad Autónoma de Aguascalientes
P r e s e n t e:

Por este conducto hago de su conocimiento que el **M.E.S Luis Fernando Gutiérrez Marfileño**, egresado del **Doctorado en Ciencias Exactas, Sistemas y de la Información** del Centro de Ciencias Básicas de la Universidad Autónoma de Aguascalientes, ha integrado satisfactoriamente el documento de tesis titulado **"Arquitectura de Sistemas Multi-Agente Orientada a Aplicaciones Industriales en Tiempo Real"**, por lo que doy mi voto aprobatorio para que continúe con los trámites para presentar el examen de grado reglamentario.

Atentamente:

Dra. Eunice Esther Ponce de León Senti
Directora de Tesis

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Aguascalientes, Ags. Junio de 2010

Dr. Francisco Javier Álvarez Rodríguez
Decano del Centro de Ciencias Básicas
Universidad Autónoma de Aguascalientes
P r e s e n t e:

Por este conducto hago de su conocimiento que el **M.E.S Luis Fernando Gutiérrez Marfileño**, egresado del **Doctorado en Ciencias Exactas, Sistemas y de la Información** del Centro de Ciencias Básicas de la Universidad Autónoma de Aguascalientes, ha integrado satisfactoriamente el documento de tesis titulado **"Arquitectura de Sistemas Multi-Agente Orientada a Aplicaciones Industriales en Tiempo Real"**, por lo que doy mi voto aprobatorio para que continúe con los trámites para presentar el examen de grado reglamentario.

Atentamente:

Dr. Felipe Padilla Díaz
Asesor de Tesis

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Aguascalientes, Ags. Junio de 2010

Dr. Francisco Javier Álvarez Rodríguez
Decano del Centro de Ciencias Básicas
Universidad Autónoma de Aguascalientes
P r e s e n t e:

Por este conducto hago de su conocimiento que el **M.E.S Luis Fernando Gutiérrez Marfileño**, egresado del **Doctorado en Ciencias Exactas, Sistemas y de la Información** del Centro de Ciencias Básicas de la Universidad Autónoma de Aguascalientes, ha integrado satisfactoriamente el documento de tesis titulado **"Arquitectura de Sistemas Multi-Agente Orientada a Aplicaciones Industriales en Tiempo Real"**, por lo que doy mi voto aprobatorio para que continúe con los trámites para presentar el examen de grado reglamentario.

Atentamente:

Dr. Alejandro Padilla Díaz
Asesor de Tesis

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Aguascalientes, Ags. Junio de 2010

Dr. Francisco Javier Álvarez Rodríguez
Decano del Centro de Ciencias Básicas
Universidad Autónoma de Aguascalientes
P r e s e n t e:

Por este conducto hago de su conocimiento que el **M.E.S Luis Fernando Gutiérrez Marfileño**, egresado del **Doctorado en Ciencias Exactas, Sistemas y de la Información** del Centro de Ciencias Básicas de la Universidad Autónoma de Aguascalientes, ha integrado satisfactoriamente el documento de tesis titulado **"Arquitectura de Sistemas Multi-Agente Orientada a Aplicaciones Industriales en Tiempo Real"**, por lo que doy mi voto aprobatorio para que continúe con los trámites para presentar el examen de grado reglamentario.

Atentamente:

Dra. Elva Díaz Díaz
Asesor de Tesis

Resumen

Las empresas manufactureras actuales enfrentan una serie de desafíos para operar eficientemente en los mercados globales y obtener ganancias. Anteriormente estos retos se enfrentaron mediante la organización jerárquica centralizada.

El entorno ha cambiado y ahora deben implementar arquitecturas abiertas que permitan responder en forma ágil y a tiempo fabricando cada vez más nuevos productos con ciclos de vida más cortos y con clientes cada más selectivos.

Los sistemas multiagente (SMA) son un área de la Inteligencia Artificial Distribuida (IAD) que tratan de resolver complejos en forma distribuida, lo que les confiere una gran robustez y flexibilidad al resolver cada agente solo un parte del problema, existen muchos tipos de agentes con muy variadas metas.

La aportación de ésta tésis presenta el diseño de una arquitectura de SMA orientada a aplicaciones industriales en donde se busca que el sistema de producción actúe en forma planificada cuando recibe los pedidos, en forma reactiva, durante la programación de la producción y que integre un control inteligente durante todo el proceso.

A la solución distribuida de los problemas que se presentan en el área de producción se le integra un enfoque multi-objetivo durante todo el proceso, que es como ocurre en la realidad.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

INTRODUCCIÓN

1.1. Introducción

Las empresas manufactureras en la actualidad se enfrentan a requerimientos tales como: la disminución del ciclo de vida de los productos, tiempos de entrega reducidos e incremento en la variedad de productos a fabricar lo que las obliga a disponer de procesos de fabricación fácilmente adaptables (a bajo costo), escalables en pasos pequeños, capaces de reaccionar flexiblemente a órdenes de producción variables, estructurados y organizados robustamente para que el impacto de los disturbios sea limitado, y que sean capaces de reasignar recursos eficientemente [Peeters et al., 1998].
Anteriormente, la forma de enfrentar este tipo de procesos complejos fue la clásica organización jerárquica centralizada, cuya rigidez [Hatvany, 1985] podía dar lugar a la paralización de la actividad de una fábrica por un fallo en cualquier punto de la cadena de producción.
Debido a lo cual, este tipo de empresas necesitan adoptar arquitecturas abiertas que integren sus actividades con proveedores, clientes y servicios en amplias redes de cadena de suministros.
Además, para poder competir de forma efectiva en los mercados globales, en muchas ocasiones se debe asegurar el cumplimiento de restricciones temporales (sistemas de tiempo real (STR)) que pueden afectar desde tiempos de proceso hasta tiempos de entrega de productos, lo cual tiene repercusiones logísticas y económicas.
De acuerdo a [Chase, 2001] una empresa manufacturera está integrada por 3 divisiones básicas que son: Mercadotecnia, Finanzas y la Gerencia de Planta (producción), el problema que este trabajo de investigación pretende resolver se ubica en esta última, específicamente en las subáreas de Control de la Producción y Manufactura cuyas funciones son: la planeación, secuenciación así como el control y la ejecución de la producción, que, como ya se mencionó, en la administración tradicional son centralizados y no son suficientemente flexibles para responder a los nuevos y cambiantes estilos de producción y a las variaciones en los requerimientos del producto [Botti & Giret, 2005].

Figura 1.1: Carta Organizacional de una Empresa Manufacturera

Para cumplir con estos requerimientos, dichos sistemas deben responder con gran adaptabilidad y capacidades de integración tales como cooperación, coordinación, robustez, reactividad, flexibilidad, heterogeneidad, autonomía y viabilidad, entre otras; para resolver esta problemática se han desarrollado técnicas inteligentes distribuidas [Im Cho, 2008].
Entre estos nuevos paradigmas capaces de satisfacer tales requerimientos se encuentran los Sistemas Multiagente (SMA)[Botti & Giret, 2005].
De forma general un Agente se define como cualquier cosa capaz de percibir su medio ambiente con la ayuda de sensores y actuar en ese medio utilizando actuadores [Russell & Norvig, 2006], y un SMA consiste de una población de entidades autónomas (agentes) situadas en una entidad estructurada compartida (ambiente) [Weyns et al., 2005].
La tecnología de Agentes/SMA está realizando importantes aportaciones en la solución de problemas en muy diversos dominios (como el comercio electrónico, subastas electrónicas, medicina, bolsa, etc.), donde aproximaciones tradicionales no han proporcionado soluciones suficientemente satisfactorias [Botti & Giret, 2005].
Y en concreto, la industria manufacturera es uno de los dominios donde la tecnología de agentes es capaz de resolver, de manera natural, problemas que son inherentemente distribuidos y que presentan unos requerimientos, cuyos aspectos específicos a considerar la hacen diferente de soluciones propuestas para otros dominios [Wooldridge, 2002].
Además de que, las aplicaciones industriales son ambientes heterogéneos ya que contemplan una amplia variedad de elementos de producción tales como robots, máquinas, controles de procesos, etc. los cuales pueden formar una sociedad de agentes con conocimientos que pueden encargarse del monitoreo y control de una serie de recursos con variados funcionamientos [Roumeliotis, 2002].

1.2. Planteamiento y descripción general del proyecto

Este trabajo de investigación propone el diseño e implementación de una arquitectura específica de SMA que produzca sistemas capaces de realizar, de forma optimizada, operaciones de planeación, secuenciación, control y ejecución de la producción en tiempo real en empresas manufactureras, lo que implica un nivel de coordinación muy alto entre los diferentes elementos productivos.
Para lograrlo, el comportamiento del sistema debe ser robusto ante la impredecibilidad de las operaciones de manufactura, una aproximación que, de manera descentralizada ha demostrado su potencial en la coordinación mediante politicas adecuadas como en los sistemas adaptativos, es el modelo heurístico bioinspirado llamado Ruteo de Avispas [Cicirello & Smith, 2004] el cual esta basado en un modelo de auto-organización que tiene lugar dentro de una colonia de avispas donde las interacciones entre los miembros de la colonia y el ambiente local resulta en una distribución dinámica de tareas, además de un orden jerárquico social emergente formado por las interacciones entre los miembros de la colonia.
Entre los SMA naturales el modelo de comportamiento de las avispas ha dado mejores resultados que otras sociedades de insectos (hormigas, abejas, etc.) en el caso de problemas dinamicos [Cicirello et al., 2001].
Por lo que es la heurística seleccionada para esta arquitectura, y en la que se va a manejar desde luego, un enfoque de sistema en tiempo real.

1.3. Objetivos de la investigación

1.3.1. Objetivo general

El objetivo general de este proyecto es el desarrollo de una arquitectura para el diseño de SMA orientada a aplicaciones industriales, para resolver los problemas de planeación, secuenciación y control de producción en un entorno de tiempo real.

1.3.2. Objetivos específicos

Para lograrlo se plantean los siguientes objetivos específicos:

- Elaborar una revisión de los requerimientos actuales de los sistemas de manufactura.
- Desarrollar un estudio básico sobre el estado del arte de las arquitecturas actuales de diseño de SMA orientadas a la manufactura.

- Revisar los métodos de optimización en manufactura especificamente mediante una heurística bioinsipirada denominada *Ruteo_de_avispas* [Cicirello & Smith, 2004].
- Plantear una arquitectura de diseño de SMA orientada a aplicaciones industriales que cumpla con los requisitos funcionales y de Ingeniería de Software anteriormente establecidos:
  - ▷ Agentes con la mayor autonomía posible que razonen sobre el comportamiento del sistema de fabricación.
  - ▷ Las unidades de fabricación tendrán un comportamiento basado en rutinas.
  - ▷ Los métodos de programación proveerán encapsulación de datos y procesos.
  - ▷ Los programas de control tendrán una semántica clara y especificarán completamente el comportamiento del agente.
  - ▷ La metodología conducirá de una tarea de control a un programa de agente.

Cuando se determine que dicha arquitectura cumple con los requerimientos de fabricación se va a proceder a:

- Aplicar la arquitectura propuesta en un sistema de simulación que involucre la mayor parte de los elementos planteados y obtenga resultados optimizados de más de una variable medible de desempeño de un sistema de producción.
- Evaluar los resultados obtenidos y de ser necesario se realizarán los ajustes o correcciones necesarios.
- Analizar los resultados finales y se va a determinar el nivel de optimización logrado.

1.4. Novedad y actualidad del problema de investigación

1.4.1. Entorno de las empresas de manufactura

Para enfrentar la competitividad y responder a los cambios del mercado, una empresa de manufactura debe estar integrada con sus sistemas de gestión (compras, diseño, producción, planeación, control, transporte, recursos humanos, materiales, calidad, etc.) que son entornos software y hardware con sistemas de planeación táctica, mediante sistemas distribuidos basados en el conocimiento para relacionar gestión de demanda directamente con recursos y planeación de capacidades [Monfared & Yang, 2006].
Las empresas manufactureras requieren de plena cooperación de sus proveedores y clientes para optimizar el suministro de materiales, la fabricación de accesorios, comercialización del producto final, etc. [Fox et al., 1993], dicha cooperación debe realizarse eficientemente [Christensen, 1994] y responder rápido a cualquier cambio [Deen, 1994], y es además, un requisito imperativo de cualquier modelo funcional para sistemas avanzados de fabricación [HMS, 1994].
En este tipo de empresas las personas y computadoras deben ser integradas para trabajar colectivamente en diversas etapas de desarrollo del producto [Suri, 2003], con acceso al conocimiento [Boose et al., 1990] y a la información requerida.
Para soportar las necesidades y aumentar las capacidades de decisión del sistema hay que integrar fuentes heterogéneas de información.
La globalización y las expectativas de expansión de mercados transforman rápidamente los entornos de fabricación [Gaines, 1992].
Se realizan esfuerzos en la reducción de ciclos del producto, para responder con rapidez a los deseos de los clientes.
En esta escala las corporaciones reorientan progresivamente sus estrategias para expandir su participación de mercado e integrar la fabricación ágil en sus capacidades de producción, lo que es la habilidad de adaptarse en un entorno de fabricación [Ulieru et al., 2002], a continuos e inesperados cambios y esto es un componente clave en las estrategias de fabricación para la competitividad global.
Para conseguir agilidad, las capacidades de fabricación deben establecer asociaciones convenientes con socios heterogéneos.
La escalabilidad es una importante propiedad para los sistemas de fabricación avanzados. Escalabilidad significa que en la organización pueden ser incorporados más recursos cuando son necesarios y debe satisfacer cualquier nodo de trabajo del sistema y cualquier nivel dentro de los nodos [Botti & Giret, 2005].
La expansión de recursos debería ser posible sin romper enlaces organizacionales previamente establecidos.
Cuando nuevos componentes físicos llegan al sistema, son creadas entidades

representativas para actuar como imagen de los componentes a través de sus ciclos de vida. La habilidad de añadir nuevos componentes incrementalmente permite al sistema responder flexiblemente a una amplia variedad de solicitudes.

Tanto las personas como las entidades artificiales que pertenecen a un sistema de fabricación necesitan estar alerta a los cambios del entorno [Ulieru, 2004]. Cada etapa de la planeación de la fabricación se ve afectada por variaciones dinámicas que provienen tanto de fuentes internas como externas. En sistemas de fabricación convencionales, las entradas de los clientes activan una secuencia de eventos, empezando con operaciones de planeación.

En este nivel, los pedidos son procesados de acuerdo a estados preestablecidos (los cuales incluyen la especificación del diseño del producto, gestión de materiales, planeación y disponibilidad de la capacidad de fabricación, y la preparación de los costos de producción) [Botti & Giret, 2005].

El proceso de planeación también activa requerimientos para la subcontratación de servicios externos.

La adquisición y distribución eficiente del conocimiento de cada aspecto de la organización (finanzas, marketing, diseño y fabricación) junto con su uso efectivo, producirá notables avances sobre la investigación de mercados, el desarrollo de productos y procesos, la planeación y secuenciación de la producción, y finalmente la respuesta a clientes.

Los problemas observados en sistemas convencionales de fabricación, son la gestión de información y control de producción muy centralizados, consecuencia de la necesidad de mantener una vista general del sistema con el fin de minimizar costos.

Todas las actividades en sistemas de fabricación convencionales están limitadas por la corrección y estabilidad de los componentes de procesamiento centralizados, produciendo una infraestructura frágil.

Asegurar la viabilidad de la fabricación del producto es el primer paso de la implementación de la ingeniería concurrente.

Especificaciones geométricas y funcionales, disponibilidad de materias primas, y la capacidad y disponibilidad de recursos de fábrica tienen una gran influencia en la viabilidad de la fabricación [Cutkosky et al., 1996]. Un diseño puede ser fabricado bajo una combinación de requerimientos de producto y recursos de planta, pero no bajo otros.

Por lo anterior, los sistemas de fabricación deben adaptarse a software y hardware heterogéneos en sus entornos de fabricación y de información.

Los entornos de información heterogéneos pueden utilizar diferentes lenguajes, representar datos con diferentes lenguajes de representación y modelos, y ejecutarse en diferentes plataformas y, a pesar de ello, sus subsistemas y componentes deben ejecutarse eficientemente.

El sistema debería ser tolerante a fallos tanto a nivel de sistema como de subsistema, así como detectar y recuperarse de fallos del sistema en cualquier nivel y minimizar sus impactos sobre el entorno de trabajo.

En general, el problema de la planeación de la producción puede ser abordado desde tres niveles: la planeación estratégica, cuyas soluciones provienen, normalmente, de la programación lineal, la planeación táctica, mediante métodos exactos, y la planeación operativa (secuenciación) por medio de técnicas heurísticas debida a la explosión combinatoria del espacio de soluciones [Toro Ocampo et al., 2006].

1.4.2. Entorno de Ingeniería de Software

Como los SMA son, en esencia, programas computacionales se requieren técnicas de Ingeniería de Software para estructurar su proceso de construcción. Por lo cual se deben especificar, en primer lugar, los servicios que debe proporcionar la aplicación (requerimientos funcionales). A continuación se enlistan algunos de ellos:

1. Los sistemas de control de fabricación requieren agentes semiautónomos, éstos deben razonar sobre el comportamiento del sistema de fabricación, pero no sobre sus propias actitudes mentales o aquellas de otras unidades de control.

2. Las unidades de control de fabricación principalmente requieren de un comportamiento basado en rutinas que debe ser al mismo tiempo efectivo y oportuno (timely), el cual podría ser configurable o auto adaptativo.

También se deben establecer aquellos requerimientos que se relacionan con el desempeño del sistema, su confibilidad y disponibilidad así como sus restricciones, todos estos se conocen como requerimientos no funcionales, para este caso podemos mencionar los siguientes:

1. Los métodos de programación deben proveer encapsulación de datos y procesos.

2. Los programas de control deben tener una semántica clara, adicionalmente, el comportamiento de un agente debería ser completamente especificado por su programa de control.

3. Un método o metodología de programación debería conducir directamente de una tarea de control a un programa de agente.

Entre los métodos para desarrollar sistemas inteligentes avanzados flexibles y complejos, el paradigma multiagente es uno de los más prometedores.
Dentro de las áreas de desarrollo en la investigacion de agentes se cuentan las mostradas en la figura 1.
La tecnología de agentes se emplea en la integración de empresas de manufactura en la gestión de la cadena de suministro, en planeación, secuenciación

Figura 1.2: Áreas de Investigación

y control de la producción, en manipulación de materiales, y en los sistemas de fabricación holónicos [Christensen, 1994].
Esta tecnología se presenta como una forma natural de superar tales problemas [Belecheanu et al., 2006] y diseñar e implementar entornos de fabricación inteligentes distribuidos y está recibiendo una gran atención en los últimos años y, como consecuencia, la industria de desarrollo de software comienza a interesarse en adoptar esta tecnología para desarrollar sus propios productos.
Sin embargo, a pesar del rápido desarrollo de teorías, arquitecturas y lenguajes de agentes, se ha realizado muy poco trabajo en la especificación (y aplicación) de técnicas para desarrollar aplicaciones empleando tecnología de agentes.
Esta es el área en la que pretende incidir este trabajo de investigación, considerando el esfuerzo realizado en las demás áreas relacionadas con los agentes.
Y uno de sus objetivos es ofrecer al ingeniero de software guías claras y específicas así como fases de desarrollo completas para el ciclo de vida de un SMA.
La investigación dentro de la SMA esta dirigida a buscar la solución de un problema dividiéndolo entre un número de agentes cooperativos, y decidiendo cómo se debe compartir el conocimiento de manera que los agentes puedan trabajar conjuntamente[Moulin, 1996]. Así, las ventajas de un SMA con respecto a un sistema centralizado y monolítico son:

- Rápida solución de problemas debido al procesamiento paralelo.
- Comunicación mínima (transmite solo soluciones parciales de alto nivel a otros agentes en lugar de tener que enviar datos básicos a un sistema central)
- Mayor flexibilidad, (se tienen agentes con diferentes habilidades que en

forma dinámica cooperan entre sí para resolver problemas).

- Mayor confiabilidad, (otros agentes toman las responsabilidades de los agentes que fallen en su operación).

Por lo que, el diseñar e implementar SMA es un problema complejo, y existen una serie de desafíos a los que se enfrenta el diseñador, algunos de los cuales se señalan a continuación [Vlassis, 2003]:

- Cómo descomponer un problema y asignar sub tareas a agentes y sintetizar resultados parciales.
- Cómo manejar información perceptual distribuida y cómo pueden los agentes mantener modelos compartidos consistentes con el mundo.
- Cómo implementar control distribuido y construir mecanismos eficientes de coordinación.
- Cómo diseñar algoritmos eficientes de planeación y aprendizaje para SMA.
- Cómo representar conocimiento y hacer que los agentes razonen sobre sus acciones, planes y el conocimiento de otros.
- Qué lenguajes y protocolos usar para comunicar agentes.
- Cómo pueden los agentes negociar y resolver conflictos.
- Cómo pueden los agentes formar estructuras organizacionales tales como equipos o coaliciones así como asignar roles a los agentes.
- Cómo asegurar un comportamiento coherente y estable de los sistemas.

1.5. Necesidad e importancia del trabajo de tesis

1.5.1. Justificación de la investigación

La tecnología de SMA está recibiendo una gran atención en los últimos años, y como consecuencia, la industria de desarrollo de software está comenzando a interesarse en adoptar esta tecnología para desarrollar sus propios productos. Y aunque se ha dado un rápido desarrollo de teorías, arquitecturas y lenguajes de agentes, entre la comunidad multiagente se menciona que es necesario desarrollar más sobre metodologías [McKean et al., 2007], ya que es poco el trabajo realizado en la especificación de técnicas para desarrollar aplicaciones empleando la tecnología de agentes.

Por lo que, la introducción de la tecnología de Agentes/SMA en la industria requiere de metodologías que asistan en todas las fases del ciclo de vida del sistema de agentes [Pechoucek & Marik, 2008].

La necesidad del empleo de estas técnicas se vuelve más importante en la medida que el número de SMA aumenta [Belecheanu et al., 2006].

Los SMA se desarrollan sobre "middleware" y proporcionan un nuevo nivel de abstracción más intuitivo, y su diseño generalmente, se aborda pensando en los agentes como entes con motivación.

En lugar de modelar un sistema con componentes que ejecutan métodos, el desarrollador tiene que pensar en los objetivos que deben alcanzar los componentes y en las tareas necesarias para resolver sus objetivos.

Al desarrollar los componentes así, el proceso es más intuitivo ya que esta forma de modelar y de razonar de acuerdo con planteamientos cognitivos se halla más cerca del pensamiento humano que los paradigmas de programación tradicionales.

Como la construcción de SMA integra tecnologías de muy distintas áreas de conocimiento: técnicas de ingeniería de software para estructurar el proceso de desarrollo; técnicas de inteligencia artificial para dotar a los programas de capacidad para tratar situaciones imprevistas y tomar decisiones, y programación concurrente y distribuida para tratar la coordinación de tareas ejecutadas en diferentes máquinas bajo diferentes políticas de planeación [Gómez, 2002].

El rango de estas soluciones va desde proporcionar servicios básicos (gestión de agentes, librerías de algoritmos, localización de agentes o movilidad), como JADE, hasta entornos de desarrollo donde se parametrizan armazones (framework) de software, como ZEUS.

Y aunque facilitan el proceso, las plataformas de desarrollo quedan incompletas sin un proceso de desarrollo de software especializado para agentes que haga similar la creación de SMA a la producción de software convencional.

Las técnicas convencionales de ingeniería (RUP [Jacobson et al., 2000], CommonKADS [Tansly et al., 1993]) no tienen en cuenta las necesidades de es-

pecificación de los SMA, como la especificación de planeación de tareas, intercambio de información con lenguajes de comunicación orientados a agentes, movilidad del código o motivación de los componentes del sistema. secuenciación Por ello, se plantean nuevas metodologías basadas en agentes (MaSE, GAIA, TROPOS, INGENIAS, etc.), las cuales parten de un modelo (informal) de cómo debe ser un SMA y dan guías para su construcción [Gómez, 2002].
Inicialmente, se elaborará un estudio de los requerimientos específicos del control de la producción en la actualidad para estas empresas que fundamente la fase de análisis de requerimientos de la metodología propuesta.
Después, y ante el importante número de metodologías desarrolladas hasta hoy se realizará un estudio de estado del arte especialmente sobre aquellas que incursionen en el control de la producción de empresas manufactureras.
Y es que, en estudios comparativos [Cuesta, 2005] se ha demostrado que en la realidad pocos proyectos de desarrollo de agentes utilizan una metodología, la mayoría de los desarrolladores de sistemas basados en agentes utilizan un acercamiento ad-hoc, esto aporta flexibilidad pero la calidad final resulta cuestionable y existe mayor dificultad para reproducir métodos y/o procesos seguidos.
Por lo que existe una necesidad real de desarrollar Metodologías Orientadas a Agentes (MOA) adecuadas, para abordar la solución de problemas complejos, coordinar trabajo de equipo, conducir el proceso de desarrollo y extender la utilización del paradigma de agentes.
Ya que, en un proyecto de desarrollo de software la metodología define ¿Quién debe hacer Qué?, ¿Cuándo? y ¿Cómo? debe hacerlo, la metodología tiene que proporcionar todos los elementos necesarios para el desarrollo de un sistema de software y además debe proporcionar directrices precisas de cómo llevar a cabo cada uno de los pasos a realizar y de los artefactos a producir.
Algunas conclusiones de la aplicación de frameworks de evaluación a esta clase de proyectos sobre agentes son las siguientes [Fipa, 2006]:

- Propuestas sin experimentación en problemas reales.
- Cubren parcialmente el ciclo de vida.
- Algunas sólo cubren aspectos específicos.
- Separación entre análisis y diseño difícil o poco clara (confusión entre actividades de análisis y actividades de diseño).
- Carencia de herramientas de soporte contrastadas.
- Limitaciones para modelar sistemas dinámicos.
- Escasa documentación y disparidad de notaciones.

Como ya se mencionó, el problema a solucionar implica restricciones temporales, situación no considerada en la mayoría de la metodologías actuales

[Julián & Botti, 2003], los casos en los que si se consideran incluyen aproximaciones holónicas [Julián, 2002]. Las cuales difieren esencialmente de la orientación a agentes en que los holones son un tipo especial de agente, así como el que abarcan el funcionamiento total de una empresa, algo que en este trabajo no se pretende realizar. Y es que para la construcción de un sistema de tiempo real se deben especificar en forma precisa los eventos a los que va a responder, las respuestas adecuadas y los períodos de tiempo específicos de cada uno de ellos.
Las relaciones entre los SMA y los STR planteadas en [Julián, 2002]:

> **Definición 1:** *Un agente de tiempo real es un agente con restricciones temporales en la realización de alguna de sus responsabilidades o tareas. Se puede distinguir entre estricto y no estricto:*
>
> - *Un agente de tiempo real es considerado estricto si:*
>
> *Las restricciones temporales asociadas a sus responsabilidades son estrictas. Por tanto el no cumplimiento de dichas responsabilidades dentro de los intervalos de tiempo especificados puede ocasionar que sus acciones no sean útiles, o bien, tengan consecuencias graves.*
>
> - *Un agente de tiempo real es considerado no estricto si:*
>
> *Ninguna de las restricciones temporales asociadas a sus responsabilidades es estricta. El no cumplimiento de sus responsabilidades acotadas temporalmente únicamente supone una perdida de calidad en la respuesta.*
>
> **Definición 2:** *Un sistema multiagente de tiempo real es un sistema multiagente donde al menos uno de sus agentes es de tiempo real ya sea estricto o no.*

En una primera fase de evaluación la metodología se implementará en un ambiente altamente automatizado denominado sistema flexible de manufactura (FMS), en un ambiente controlado de laboratorio, en donde se busca realizar un análisis a fondo de las potencialidades de la metodología.
Y en una fase de evaluación complementaria se implementará un SMA para una empresa real para determinar la eficiencia de la metodología.

1.6. Metodología

De manera general definiremos el término metodología como lo relacionado con los métodos de conocimiento, y específicamente para este trabajo, como el conjunto de métodos o técnicas que ayudan en el desarrollo de un producto de software.
Como los agentes son programas computacionales, su diseño está determinado básicamente por área de la Ingeniería de Software, de acuerdo a ésta, las principales actividades de una metodología [Pressman, 2006] son:

- La definición y descripción del problema que se desea resolver.
- El diseño y descripción de una solución que se ajuste a las necesidades del usuario.
- La construcción de la solución.
- La prueba de la solución implementada.

En base a lo anterior se plantea un proceso que seguirá las siguientes etapas de desarrollo del presente trabajo de investigación:

- Diseño de la metodología.
- Prueba piloto.
- Prueba en una aplicación real.

Los requisitos que según [Dorfman & Thayer, 1997] debe cumplir una metodología y que éste proyecto buscará cumplir son los siguientes:

- La metodología está *documentada*: el procedimiento de uso de la metodología está contenido en un documento o manual de usuario.
- La metodología es *repetible*: cada aplicación de la metodología es la misma.
- La metodología es *enseñable*: los procedimientos descritos tienen un nivel suficientemente detallado y existen ejemplos para que personal cualificado pueda ser instruido en la metodología.
- La metodología está *basada en técnicas probadas*: la metodología implementa procedimientos fundamentales probados u otras metodologías más simples.
- La metodología es *apropiada al problema* que quiere resolverse.

Una definición más precisa de metodología [Heijst, 1994], es aquella en la que se considera como una tecnología de capas donde cada capa puede verse como un bloque de construcción empleado por las capas superiores. Esta definición se asemeja a la definición de Pressman [Pressman, 2006] para la ingeniería de software. En la Figura 2 se muestra una pirámide metodológica que añade una capa de proceso al resto de capas definidas en la pirámide metodológica de CommonKADS. Las capas que se distinguen son las siguientes:

- **Visión del mundo:** perspectiva, principios y asunciones con que se analiza el problema.

  Los enfoques más utilizados para analizar un problema son:

  Enfoques estructurados orientados a funciones, que ven el software como la transformación de información, descomponiendo en funciones con entradas y salidas [Page-Jones, 1980][Powers et al., 1990]. *Enfoques orientados a objetos*, que ven el software como un conjunto de objetos derivados de clases genéricas que se comunican mediante paso de mensajes [Booch, 1998]. *Enfoques formales* que describen el software en términos matemáticos, permitiendo una evaluación de su completitud, consistencia y corrección. *Enfoques de ingeniería del conocimiento* que se centran en el modelado del conocimiento que necesita una aplicación basada en conocimiento para llevar a cabo sus razonamientos y en la descripción de estos mecanismos de razonamiento [Harris-Jones, 1995]. *Enfoques orientados agentes*, que definen el software como un conjunto de agentes autónomos que cooperan para satisfacer los objetivos de la aplicación.

- **Teoría y modelos:** la teoría recoge un conjunto de teoremas del dominio de aplicación de la metodología, mientras que los modelos indican los diferentes puntos de vista con que se analiza el problema (p.ej. estático y dinámico). En el caso de los enfoques formales, éstos suelen ofrecer teoría matemática que permiten su verificación.

- **Proceso de ingeniería de software:** los procesos de ingeniería de software son la base de la gestión del control de los procesos de software, y establecen el contexto en que los métodos técnicos son aplicados, los productos (modelos, documentos, informes de datos, formularios, etc.) son entregados, los hitos del proyecto establecidos, el nivel de calidad asegurada y los cambios son gestionados adecuadamente. Las principales preguntas que debe responder un proceso de ingeniería de software son [Boehm, 2001]: qué haremos a continuación y cuánto tiempo tardaremos en hacerlo.

- **Métodos de ingeniería de software:** se aplican en una gran variedad de tareas, que incluyen el análisis de requisitos, el diseño, la codificación, prueba y mantenimiento.

Los métodos de ingeniería de software [Pressman, 2006] deben incluir actividades de modelado, heurísticos de modelado y técnicas de descripción del modelo. Los métodos de ingeniería de software (a menudo denominados metodologías) se centran en [Boehm, 2001] determinar cómo navegar a través de cada fase, determinando los datos, el control o las jerarquías de uso; asignación de requisitos, etc. y cómo representar los productos de cada fase (diagramas de estructura, diagramas de transición de estado, diagramas de objetos, etc.).

- **Herramientas de ingeniería de software:** proporcionan un soporte automatizado o semiautomatizado para el proceso y los métodos. Las herramientas CASE (Ingeniería Software Asistida por Computadora; Computer-Aided Software Engineering) proporcionan un entorno integrado en que las distintas herramientas que soportan los métodos y procesos pueden compartir la información.
- **Uso y transferencia:** utilización de la metodología y elaboración de métodos y programas para su transferencia y enseñanza.

La metodología que se propone en este trabajo se ubica para cada capa como sigue:

- **Visión del mundo:** *Enfoques orientados agentes*
- **Teoría y modelos:** *Modelo dinámico*
- **Proceso de ingeniería de software:** *Modelo AUML*
- **Métodos de ingeniería de software:** *Análisis de requerimientos, diseño, implementación y pruebas*
- **Herramientas de ingeniería de software:** alguna de las plataformas actuales para SMA tale como *JADE, ZEUS, IDK, etc.*
- **Uso y transferencia:** Manuales de programación y administración de sistemas

1.6.1. Los modelos de proceso de Ingeniería de Software

Los modelos de proceso [Pressman, 2006] deben escogerse dependiendo de la naturaleza del proyecto y de la aplicación, los métodos y herramientas que se van a utilizar, y los controles y productos que se requieran.
Los modelos de ciclo de vida más extendidos [Pressman, 2006] son:

- **Modelo en cascada, lineal o secuencial**: las fases de análisis, diseño, implementación y prueba ocurren de forma secuencial. Su principal problema es que la descripción de los requisitos de usuario se congela en una fase temprana de desarrollo, no se proporciona ningún producto al cliente hasta que casi el proyecto ha terminado.

- **Prototipado**: pretende comprender mejor las necesidades del cliente desde una fase muy temprana, para lo cual se producen prototipos (desechables o no) con los que el cliente puede interaccionar y evaluar si cumplen sus requisitos. Los principales problemas de un prototipado básico consisten en que a menudo los prototipos que iban a ser desechados se intentan aprovechar, generando software de baja calidad. Una variante del prototipado es el prototipado incremental, en el que se desarrolla una versión con las funcionalidades básicas y se va actualizando con nuevas funcionalidades.

- **Modelos formales o de transformación**: se basan en realizar una especificación formal [Vienneau, 1997] de los requisitos que es traducida de forma automática a código. Los principales problemas provienen de la complejidad de su aprendizaje, aplicabilidad y eficiencia del código generado.

- **Modelo de desarrollo evolutivo, en espiral o dirigido por riesgos**: combinan la naturaleza iterativa del prototipado con los aspectos de control sistemático del modelo en cascada. El modelo evolutivo, en espiral o dirigido por riesgos [Boehm, 2001] consiste en desarrollar el sistema a través de varias iteraciones por un conjunto de tareas. En cada iteración se producen versiones incrementales. Las regiones básicas que se identifican en el modelo en espiral son: identificación de objetivos, restricciones y alternativas para el ciclo de desarrollo; evaluación de las alternativas con respecto a los objetivos y restricciones, que conlleva la identificación y clasificación de los riesgos asociados a las alternativas; desarrollo de dichas alternativas a través de un plan de trabajo y, por último, evaluación por parte del cliente de los productos desarrollados en esa fase. Los principales problemas de este modelo es que aún no ha sido probada su eficacia y los clientes pueden ser reacios a su aplicación.

- **Modelo de desarrollo con biblioteca de componentes**: esta técnica [Sommerville, 2002] asume la existencia de partes (componentes) del sistema. El proceso de desarrollo se centra en cómo integrar estas partes en vez de en desarrollar el sistema a partir de cero. Su principal problema es la gestión de los componentes reutilizables.

Para este proyecto se considera que el modelo de prototipos es el más adecuado, especificamente el incremental ya que sigue muy de cerca los requerimientos del cliente y se complementa en forma iterativa.

1.7. Aprobación del de trabajo

Como se estableció en la Sección 1.3 el objetivo de este trabajo de investigación es implementar la arquitectura de un sistema de planeación, programación y control de la producción (nombre del sistema), el problema de la programación es, a nuestro entender el que implica una mayor dificultad, por lo que se realiza un mayor esfuerzo en definir una solución adecuada, ya que la asignación de trabajos en una fáabrica es un problema dinámico (debido a los rearreglos que deben hacerse en los programas de producción a causa de eventos inesperados, tales como falta de materiales, fallas, etc.) el problema se considera NP-duro, [Ericson & Johnson, 2006] por lo que las soluciones determinísticas no son prácticas.
En una revisión sobre aproximaciones en IA, se encontró una metaheurística bioinspirada llamada *Ruteo de Avispas* que presenta un buen desempeño en problemas dinámicos [Cicirello et al., 2001].

Una revisión y una modificación sobre el algoritmo de *Ruteo de Avispas* fue presentada en el Tercer Congreso Internacional de Computación Evolutiva (2007) en la cual se reproduce el modelo especificado en la investigación de [Cicirello & Smith, 2004], nuestro trabajo propuso una modificación al de [Cicirello & Smith, 2004] combinando una heurística (*Ruteo de Avispas*) la cual se empleó para el problema de la coordinación con un algoritmo evolutivo (Algoritmo Genético), en este proceso se consideran 6 maquinas y 6 tipos diferentes de trabajos (un banco de pruebas muy conocido para evaluar este problema fue formulado por Muth y Thompson (1963) y es el que se va a emplear para medir la efectividad del AG propuesto). El criterio de optimización considerado fue la minimización del tiempo total empleado para completar todas las tareas (makespan).

*Gutiérrez Marfileño L.F., Luna Rosas F.J.; Ruteo de avispas para coordinación en ambientes industriales dinámicos; COMCEV'2007; ISBN: 970-728-055-7 Aguascalientes, México; pp. 89-94; 2007.

El siguiente trabajo se presentó en el Quinto Congreso Estatal "La Investigación en el Posgrado"de la UAA 2009, siguiendo con el mismo problema se trabaja sobre la problemática que se presenta cuando dos o más procesadores (cabinas de pintura) licitan por el mismo trabajo algnas formas de otra función objetivo que trata sobre el

Después se buscó aumentar el número de funciones objetivo y tratar el problema con un enfoque multi-objetivo el trabajo que se enviará al MICAI 2010 maneja este enfoque,

Debido a la complejidad del problema (que implica características tales co-

mo concurrencia, interacción, reactividad, temporización) se emplea la herramienta de modelado AUML una extensión de UML, (estándar de facto usado para modelar sistemas informáticos), la que incluye protocolos de interacción (capturan la dinámica del sistema en términos del envió de mensajes entre agentes mediante diagramas de protocolos y de roles) y uso de estereotipos (en este caso agentes en los diagramas de clase). En el trabajo presentado en el 11o Seminario de Investigación Ags. 2010

1.8. Estructura del trabajo

A partir de aquí, el trabajo se estructura de la siguiente forma:
En el **Capítulo 2** se presenta una introducción a los sistemas basados en agentes, iniciando con una serie de características asociadas a la esencia de la agencia y otros atributos complementarios, una serie de clasificaciones basadas en diferentes criterios de los diferentes tipos de agentes y algunas aplicaciones de los agentes. Se define el concepto de ambiente en el paradigma de agentes y sus propiedades clave, su forma de representación y estructura, así como sus responsabilidades. Finalmente la definición, el diseño y las arquitecturas en SMA.
En el **Capítulo 3** se presenta una visión sobre un tipo de sistema de producción, la empresa manufacturera la cual ha sufrido grandes transformaciones debido a las nuevas tecnologías, especificamente la introducción de las computadoras(CIM FMS). También se muestra una caracterización de la problemática que se intenta solucionar específicamente aquellos problemas que tienen que ver con la planeación, la secuenciación y el control de la producción. Además una revisión de los requerimientos sobre los sistemas de tiempo real.
En el **Capítulo 4** se presenta un análisis de 7 arquitecturas (AARIA, SRTA, ARCHON, PROSA, manAge, MASCOT y CORTES) orientadas a aplicaciones industriales en donde se analizan los siguientes elementos:

- Información de referencia
  - Fuentes de donde se obtuvo la información
  - Nombre de la arquitectura
  - Creadores
  - Dominio específico
  - Organización diseñadora
  - Organización receptora
- Objetivos
- Tipo de arquitectura
- Diagramas
- Modelo de agente
- Configuracióon de la comunidad de agentes
- Mecanismo de comunicación

En el **Capítulo 5** se presenta la arquitectura propuesta en este trabajo.
En el **Capítulo 6** se presenta un caso de uso mediante un sistema simulado de una planta de pintura en una empresa automotriz
En el **Capítulo 7** se presentan las conclusiones

Capítulo 2

AGENTES, AMBIENTE Y SISTEMAS MULTI-AGENTE

2.1. Sistemas basados en agentes

Por *sistemas basados en agentes*, entendemos aquellos en los cuales la abstracción clave empleada es el *agente*. Estos sistemas pueden contener un sólo agente ó una mayor cantidad en los llamados *Sistemas Multi-Agente* (SMA) [Wooldridge, 2002].
A continuación se especifica el contexto en el cual se ubican los conceptos de agente, agente de software, esencia de la agencia y otros atributos.

2.1.1. Agentes

En forma general, un *Agente* se define como aquel que actúa o tiene la virtud de actuar [Larousse, 2003], lo cual puede abarcar desde un sistema de control simple (un control de nivel que se activa al detectar el nivel de un líquido), sistemas biológicos fundamentales (organismos unicelulares, bacterias, microorganismos), hasta organimos complejos (como animales, el ser humano o sistemas climáticos).
En Computación un *Agente* es cualquier sistema capaz de percibir su medio ambiente con la ayuda de *sensores* y actuar en ese medio utilizando *actuadores* [Russell & Norvig, 2006].

Agentes de Software
El concepto de *Agente* subyace en algunos de los trabajos de Vannevar Bush (As we may think), Doug Englebart, Carl Hewitt, John McCarthy, Oliver

G. Selfridge, John Von Neumman (Teoría de los Autómatas Autorreproductivos), entre otros.
Con esta noción de agente como una máquina de estados un enfoque más operativo define a un agente como una 4-tupla [Parunak 1997]:

$$Agentes = Agente_1, ..., Agente_n$$

$$Agente_i =< Estado_i, Entrada_i, Salida_i, Proceso_i >$$

Donde:

- $Estado_i$ es el conjunto de valores que defienen totalmente al agente.
- $Entrada_i$ y $Salida_i$ son subconjuntos de $Estado$, cuyas variables se integran al ambiente (mediante sensores y actuadores respectivamente).
- $Proceso_i$ es la ejecución de un mapeo autónomo que cambia el $Estado$ del agente.

2.1.2. La Esencia de la Agencia

La noción de agente debe ser más una herramienta para analizar sistemas y no una caracterización absoluta que divide el mundo en agentes y no-agentes. Cada agente está situado en, y es parte de, algún ambiente al que sensa y sobre el cual actúa en forma autónoma y en donde:

- No requiere de otra entidad para obtener sus entradas ó para interpretar y usar sus salidas. Cada acto está en función de su propia agenda, la cual busca satisfacer pero también puede perseguir metas diseñadas por algún otro agente (como en los agentes de software).
- Cada acción que realiza actualmente puede afectar el sensado posterior, esto es, las acciones afectan al ambiente.
- Finalmente, cada acción se realiza sobre un lapso de tiempo.

Para nosotros estos requerimientos constituyen la esencia de la agencia, los cuales se formalizan en la siguiente definición.

Un agente autónomo es un sistema situado en una parte de un ambiente el cual sensa y sobre el cual actúa al mismo tiempo, en busca de su propia agenda y sobre la cual actuará en el futuro [Franklin & Graesser, 1996].
Para delimitar este trabajo y enfocarlo hacia los agentes computacionales requerimos determinar las propiedades y características de tales sistemas mediante nociones más específicas de lo que en adelante entenderemos por *agencia*.

Noción Débil de Agencia

Quizá la forma más general en la cual se usa el término de agente es para denotar un sistema de cómputo de hardware, ó (más comúnmente) basado en software, que exhibe las siguientes propiedades [Wooldridge & Jennings, 1995]:

- *Autonomía*: agentes que operan sin la intervención directa de seres humanos u otros, y que tienen algún tipo control sobre sus acciones y estados internos.
- *Habilidad social*: agentes que interactúan con otros agentes (y posiblemente seres humanos) vía algún tipo de lenguaje de comunicación de agentes.
- *Reactividad*: agentes que perciben su ambiente, (el cual puede ser el mundo físico, un usuario vía una interfase gráfica o una colección de otros agentes, INTERNET, ó quizás todos ellos combinados), y que responden a tiempo a los cambios que ocurren en él.
- *Pro-actividad*: agentes que no simplemente responden a su ambiente sino que son capaces de exhibir comportamiento dirigido por objetivos tomando la iniciativa.

Esta es la noción asociada a la disciplina de ingeniería de software basada en agentes la cual considera que si se poseen dichas propiedades estamos hablando de un *agente inteligente*.

Noción Fuerte de Agencia

Un agente es un sistema de cómputo que, además de tener las propiedades mencionadas anteriormente es conceptualizado e implementado usando conceptos que son más usualmente aplicados a los seres humanos. Es común en IA caracterizar a un agente usando nociones mentalistas, tales como:

- *Conocimientos*
- *Creencias*
- *Intenciones*
- *Obligaciones (Shoham, 1993)*

Algunos investigadores han ido más allá, y consideran:

- *Agentes emocionales* ([Bates et al., 1992]; [Bates, 1994]).

Otra forma de dar atributos casi humanos a los agentes es representarlos visualmente, usando quizás una caricatura, un icono o un rostro animado ([Maes, 1994]).
Esta noción va más allá de lo que entenderíamos por un programa de cómputo, al manejar atributos que reservamos para el comportamiento humano (creencias, deseos intenciones, etc.) en los llamados *sistemas intencionales*, que es donde se integra el área de IA.

2.1.3. Otros Atributos de la Agencia

Otros atributos son a veces discutidos en el contexto de agencia, por ejemplo:

- *Movilidad* es la habilidad de un agente para moverse alrededor de una red electrónica (White, 1994).
- *Veracidad* es la suposición de que un agente no comunicará información falsa(Galliers, 1988b, pp159-164);
- *Benevolencia* es la suposición de que un agente no tendrá metas conflictivas, y que cada agente siempre tratará de hacer lo que se le pidió (Rosenschein and Genesereth, 1985, p91); y
- *Racionalidad* es la suposición de que un agente actuará para alcanzar sus metas, y que no actuará de tal forma que impida que esas metas sean alcanzadas - en tanto sus creencias se lo permitan (Galliers, 1988b, pp49-54).

La racionalidad tiene que ver con la acción de optimizar una función objetivo determinada por el usuario [Vlassis, 2003].

2.1.4. Diseño de agentes

Considerando el agente como entidad que interactúa con su entorno el diseño de un agente requiere estudiar [Fernández et.al., 2003]:

- ¿Cómo percibir el ambiente?
- ¿Cómo representar el ambiente?
- ¿Cómo definir a los actuadores?

El estudio de estos elementos ha derivado en:

- Arquitecturas, desde la experimentación en la construcción de sistemas
- Lenguajes, desde el estudio teórico de los agentes, principalmente con lógicas modales

El avance en la experimentación de diseño de sistemas ha progresado hacia soluciones más orientadas a la industria:

- Plataformas de desarrollo de agentes
- Arquitecturas reusables
- Entornos de desarrollo
- Metodologías

Según [Wooldridge & Jennings, 1995] estos elementos generan:

- *Teoría de agentes* que son esencialmente *especificaciones*. Los teóricos sobre agentes deben responder preguntas tales como:

  ¿Cómo conceptualizamos a los agentes?

  ¿Qué propiedades deben tener, y cómo representamos formalmente y razonamos sobre estas propiedades?

- *Arquitecturas de agentes* representan el paso de la especificación a la implementación. Quien trabaje en esta área debe responder a:

  ¿Cómo construimos sistemas de cómputo que satisfagan las propiedades especificadas por la teoría de agentes?

  ¿Qué estructuras de hardware y/o software son apropiadas?

  ¿Cómo separar adecuadamente estas funciones?

  Una arquitectura de agentes se caracteriza por su estructura interna, esto es, el principio de organización el cual se mueve hacia el arreglo de sus varios componentes [Ferber, 1999].

- *Lenguajes de agentes* son los lenguajes de programación que pueden incorporar los principios propuestos por los teóricos. Quien trabaje en esta área debe responder a:

  ¿Cómo se programan los agentes?

  ¿Cuales son las primitivas adecuadas para esta tarea?

  ¿Qué tan efectivamente se compilan o ejecutan los programas de agentes?

Los lenguajes de agentes parten, en su mayoría, de modelos operacionales que definen la semántica de sus instrucciones [Gómez, 2002].

2.1.5. Clasificación de agentes

Así, basándose en las funciones que pueden realizar podemos clasificar los agentes de acuerdo a la tabla 2.1 de [Franklin & Graesser, 1996].
Otra clasificación propuesta por [Wooldridge & Jennings, 1995] basada en las aplicaciones de los agentes (ver tabla 2.2).

Basada en las tres características básicas que debe tener un agente Nwana plantea los siguientes cuatro tipos de agentes:

Figura 2.1: Clasificación de agentes de Nwana

Propiedad	**Otros Nombres**	**Significado**
Reactivo	(sensa y actúa)	Responde a tiempo a los cambios en su ambiente.
Autónomo	—	Ejerce control sobre sus propias acciones.
Orientado a metas	Pro-activo Propositivo	No simplemente actúa en respuesta.
Continuo temporalmente	—	Es un proceso que corre.
Comunicativo	Hábil socialmente	Se comunica con otros agentes, incluso con seres humanos.
Que Aprende	Adaptativo	Cambia su comportamiento basado en experiencias previas.
Móvil	—	Capaz de transportarse a sí mismo de una maquina a otra.
Flexible	—	Sus acciones no están preestablecidas.
De carácter	—	Personalidad evidente y estado emocional.

Tabla 2.1: Clasificación funcional de agentes

Agente	**Descripción**	**Ejemplos de Aplicaciones**
Gopher()	Ejecutan tareas directas basadas en reglas preestablecidas y en suposiciones.	Informar sobre una variación de precios, o señalar que se ha alcanzado un punto de reorden de un artículo en un almacén.
Servidores	Ejecutan tareas bien definidas a petición del usuario.	Buscar el boleto de avión más barato o programar una reunión ejecutiva.
Predictivos	Informan o sirven en forma *voluntaria* a un usuario sin que sean explícitamente solicitados	Monitorear grupos de noticias de un tema específico en Internet o enviar información que pueda ser atractiva para un usuario

Tabla 2.2: Clasificación basada en aplicaciones

2.1.6. Algunas aplicaciones de agentes

Los agentes se han empleado en las siguientes áreas [Wooldrige y Jennings 1996]:

Administración de información personal

El desarrollo explosivo del intercambio de información a través de Internet hace que sea muy difícil a veces encontrar la información apropiada o enviarla a los remitentes seleccionados, por lo que un programa que se encargue de todas estas tareas es de suma utilidad.

Comercio electrónico

También en Internet es posible comprar y vender todo tipo de artículos, en tales ambientes los agentes de software son parte integral de todo el sistema, hay agentes usuario que actúan en conjunto con los agentes de negocios que representan a los proveedores.

Administración de procesos de negocios

Los agentes de software no sólo sirven para ayudar a los individuos sino incluso a las organizaciones para que funcionen de manera adecuada y eficiente, la razón de este desarrollo es que los procesos de las organizaciones se vuelven cada vez más complejos.

Pseudo código de un agente

```
1 Siempre
2       Percibe (sensa) el ambiente
3       Actualiza el modelo del ambiente
4       Reflexiona sobre una nueva meta
5       Busca los medios para diseñar un plan que logre la meta
6 Ejecuta el plan
```

2.2. Ambiente

2.2.1. Definición y propiedades clave del ambiente

Como ya se mencionó un agente requiere de un *ambiente* para existir y operar. El cual se define según [Parunak et.al. 2001] como: *Aquello que establece las condiciones bajo las cuales un agente puede existir.*

Propiedades clave del ambiente

- *Totalmente observable vs. Parcialmente observable*: indica si los agentes tienen acceso al estado completo del ambiente o no.
- *Determinístico vs. Estocástico*: indica que el cambio de estado del ambiente está únicamente determinado por su estado actual y las acciones seleccionadas por el agente o no.
- *Episódico vs. Secuencial*: indica que el ambiente puede cambiar a un agente en forma deliberada o no.
- *Estático vs. Dinámico*: indica que el ambiente puede cambiar a un agente en forma deliberada o no.
- *Discreto vs. Continuo*: indica que el número de percepciones y acciones está limitado o no.
- *Agente individual vs. Multi-agente*: indica que el ambiente puede cambiar a un agente en forma deliberada o no.

[Russell & Norvig, 2006]

Las clases más complejas de ambientes son las que son inaccesibles, no-determinísticas, dinámicas y continuas, las primeras tres propiedades de la lista ocurren típicamente en los SMA.

2.2.2. Representación del ambiente

Formalmente un ambiente se puede expresar por medio de la tupla [Parunak et.al. 2001]:

$$Ambiente =< Estado_e Proceso_e >$$

Donde el $Estado_e$ es un conjunto de valores que definen completamente el ambiente.
El estado incluye a los objetos y los agentes de todo el ambiente.
$Proceso_e$ es el mapeo que se ejecuta autónomamente y que cambia el estado del ambiente.

Algunos conceptos relacionados con la capa de aplicación de los SMA, sobre sus funcionalidades lógicas explicitas, los cuales están relacionados con la estructura del ambiente y con su actividad:

2.2.3. Estructura del ambiente

Estructuramiento
Los agentes en un SMA comparten un ambiente común, en el cual están dinámicamente interrelacionados, el rol del ambiente es definir las reglas bajo las cuales estas relaciones pueden existir y evolucionar.
El estructuramiento puede ser espacial, organizacional, basado en entidades, etc..

Recursos
Lo mismo que los agentes, el ambiente incluye diferentes tipos de objetos o recursos (lógicos), es su responsabilidad controlar el acceso a estos recursos.

Ontología
Un ambiente debe especificar una ontología que provea una representación conceptual del dominio que va a manejar, la cual debe cubrir la estructura del ambiente así como las características observables de objetos, recursos y agentes y sus interrelaciones.
En general, los recursos pueden ser leídos/percibidos, escritos/modificados o consumidos por los agentes.

Conceptos relacionados con la actividad
Incluye las actividades realizadas por lo agentes así como, derivado de las responsabilidades del ambiente, las relacionadas con los recursos y objetos.

Comunicación
La comunicación puede tomar diferentes formas, desde la transferencia de mensajes directos sobre comunicación anónima mediada vía un espacio compartido hasta la comunicación indirecta a través de forrajeo.

Acciones
Sí se permite que múltiples agentes actúen en un ambiente en paralelo, se requerirán modelos explícitos.

Percepción del ambiente
La percepción es la habilidad de un agente para observar su vecindad, y cuyo resultado es la percepción del ambiente. Una percepción describe el ambiente sensado en forma de expresiones que pueden ser entendidas por el agente. Los

agentes usan las percepciones para actualizar su conocimiento sobre el mundo o para directamente tomar decisiones.
En el caso de agentes situados en el mundo físico, la percepción puede ser implementada en hardware: por ejemplo, por medio de una cámara de video o por un sensor láser. Para agentes de software situado en ambientes virtuales, la percepción debe ser implementada en software donde debe ser modelada explícitamente [Weyns & Parunak, 2005].
La interacción indirecta a través del ambiente incrementa el poder y la expresividad de los SMA, habilitando soluciones que de otra forma serían imposibles o prácticamente muy complejas [Babaoglu, 2002].

Procesos del ambiente
Junto con la actividad de los agentes, los recursos u objetos pueden producir actividad en el ambiente, mantener tales dinámicas es una funcionalidad importante del ambiente.

2.2.4. Responsabilidades del ambiente

Intuitivamente, el ambiente puede verse como el lugar donde los agentes interactúan con los objetos del dominio, los recursos y con otros agentes [Weyns, 2005].
Derivado de los conceptos anteriores es posible determinar las siguientes responsabilidades del ambiente:

- *Estructuramiento.* El ambiente es el primer espacio común de los agentes el cual estructura el sistema completa. Tal estructuramiento puede ser espacial [Bandini, 2005], u organizacional [Ferber, 2003]. Propiedades específicas diferentes se pueden definir separadamente para cada espacio, tales como posiciones, localidades, grupos o roles.

- *Administración de recursos y servicio.* El ambiente actúa como un contenedor de recursos y servicios a ser accesados. Los recursos son objetos con un estado específico. Los servicios son considerados como entidades reactivas encapsuladas funcionalmente. El ambiente está a cargo de habilitar y controlar el acceso a los recursos y servicios.

- *Procesos de mantenimiento ambiental.* Además de las actividades de los agentes, el ambiente puede asignar actividades particulares a los recursos. Ejemplos de estos son la agregación, difusión y evaporación de feromonas digitales, o la recolección de basura de datos inútiles. Mantener tales dinámicas es una función importante del ambiente.

- *Habilitar la comunicación.* El ambiente define los medios concretos para que los agentes se puedan comunicar. La mayoría emplea esquemas de

estilo pase de mensajes de un a gente a otro. En la generación de una comunicación indirecta los agentes producen comunicación con objetos dentro del ambiente.

- *Reglamentar los sistemas multiagente.* El ambiente puede definir diferentes tipos de reglas o leyes sobre todas las entidades de los SMA. Las reglas pueden restringir el acceso a recursos específicos ó servicios a tipos particulares de agentes, ó determinar el resultado de las interacciones de los agentes. Las reglas del ambiente pueden ser una herramienta muy poderosa para expresar las capacidades que un ambiente necesita para asegurar consistencia en el sistema [Noriega & Sierra, 2002].

- *Proveer de monitoreo.* Contrario a los agentes, el ambiente debe estar pendiente de que los agentes tengan acceso a los recursos y servicios. Los agentes pueden, incluso ser capaces de observar las acciones de otros agentes. La observabilidad relacionada es la descripción semántica de recursos y servicios. En un sistema abierto, puede ser útil para los agentes el que sean capaces de entender en operación un nuevo ambiente que están descubriendo [Chang, et al. 2005]. Esto puede hacerse definiendo una ontología del ambiente que lo describa, sus recursos y servicios, y posiblemente sus regulaciones.

2.3. Sistemas Multi-Agente

Los SMA forman parte de una de las tres grandes ramas de la Inteligencia Artificial Distribuida (IAD), las otras dos son la Solución de Problemas Distribuidos (SPD) y la Inteligencia Artificial Paralela (IAP) [Nwana, 1996].

2.3.1. Definición

Un SMA consiste de una población de entidades autónomas (agentes) situadas en una entidad estructurada compartida (ambiente) [Weyns, 2005]. La definición de un SMA que plantea [Ferber, 99], amplía estos conceptos donde un SMA es un sistema que reúne los siguientes elementos:

- Un entorno

- Un conjunto de objetos. Estos objetos se encuentran integrados con el entorno, es decir, es posible en un momento dado asociar uno de estos objetos con un lugar en el entorno. Estos objetos son pasivos, pueden ser percibidos, creados, destruidos y modificados por agentes.

- Un conjunto de agentes que se consideran como objetos especiales que representan las entidades activas del sistema.

- Un conjunto de relaciones que unen objetos, y, por lo tanto, agentes.

- Un conjunto de operaciones, que hacen posible que los agentes perciban, produzcan, consuman, transformen y manipulen objetos.
- Operadores que representan la aplicación de operaciones sobre el mundo y la reacción de éste al ser alterado. Estos operadores se pueden entender como las leyes del universo.

2.3.2. Diseño de Sistemas Multi-Agente

La construcción de SMA integra tecnologías de distintas áreas de conocimiento: técnicas de ingeniería de software para estructurar el proceso de desarrollo; técnicas de inteligencia artificial para dotar a los programas de capacidad para tratar situaciones imprevistas y tomar decisiones, y programación concurrente y distribuida para tratar la coordinación de tareas ejecutadas en diferentes máquinas bajo diferentes políticas de planeación [Gómez, 2002].
Los SMA proporcionan una aproximación para descomponer un sistema en un número de entidades autónomas, embebidas en un ambiente el cual coopera para alcanzar los requerimientos funcionales y no-funcionales del sistema [Weyns, 2005].

2.4. Arquitecturas de Sistemas Multi-Agente

Definición de arquitectura: Es la identificación, descripción y especificación de la descomposición de interrelaciones entre los componentes de un sistema a lo largo de múltiples perspectivas y su desempeño como sistema.
La arquitectura de un SMA es un sistema de arquitectura típica que tiene a los agentes inteligentes como componentes del sistema y a la comunicación social como las interacciones entre él. Estructurar la arquitectura de un SMA es un proceso en el cual se definen los roles de los agentes y sus relaciones [Tianfield et al., 2003].
Se puede considerar que una arquitectura de agente es una metodología particular para construir agentes de forma individualizada [Maes, 1991] [Iglesias Fernández, 1998].
Clasificación de arquitecturas

- Reactivas
- Deliberativas
- Híbridas

Arquitectura de Agentes Reactiva
Agentes Puramente Reactivos
El proceso de un agente puramente reactivo es un ciclo de percepción-acción (estímulo-respuesta) que reacciona a la evolución del entorno.

No hay una representación explícita del entorno, de los otros agentes, sus capacidades etc..
Las decisiones no tiene en cuenta ni el pasado (no hay historia), ni el futuro (no hay planificación).

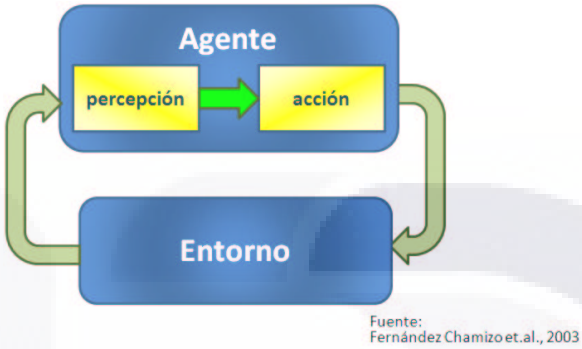


Figura 2.2: Agentes reactivos

Agentes Reactivos que mantienen su Estado Interno
Este tipo de agentes deciden la acción a realizar teniendo en cuenta su historia de interacciones con el entorno.
La secuencia de estados del entorno (secuencia de percepciones) se guarda como el estado interno del agente.

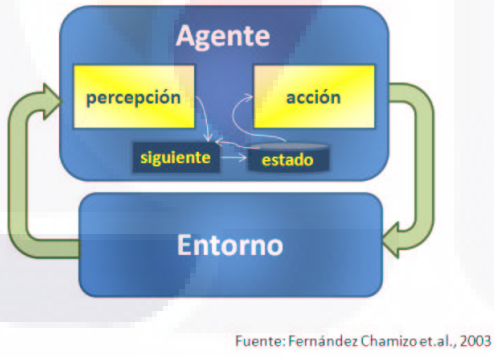


Figura 2.3: Agentes reactivos con estado interno

Ciclo de ejecución de un agente reactivo:

```
Reglas: condición-acción
Conjunto de percepciones
mientras (verdadero){
estado = interpretar entrada(percepción);
regla = correspondencia (estados, Reglas);
ejecutar (regla, acción);
}
```

Agentes Reactivos (subsunción)

La arquitectura de subsunción fue propuesta por [Brooks, 1986], y está formada por un conjunto de módulos de comportamiento que realizan tareas, aquí no hay representación ni razonamiento simbólico, y el comportamiento se puede implementar como un conjunto de reglas situación !acción donde situación se toma directamente de la percepción (sin ningún tipo de transformación o representaciones simbólicas).

Jerarquía de subsunción

Se pueden ejecutar varios comportamientos simultáneamente, para elegir entre ellos se usa la jerarquía de subsunción, los comportamientos están ordenados por capas, los de las capas más bajas (mayor prioridad) inhiben a los de las capas superiores.

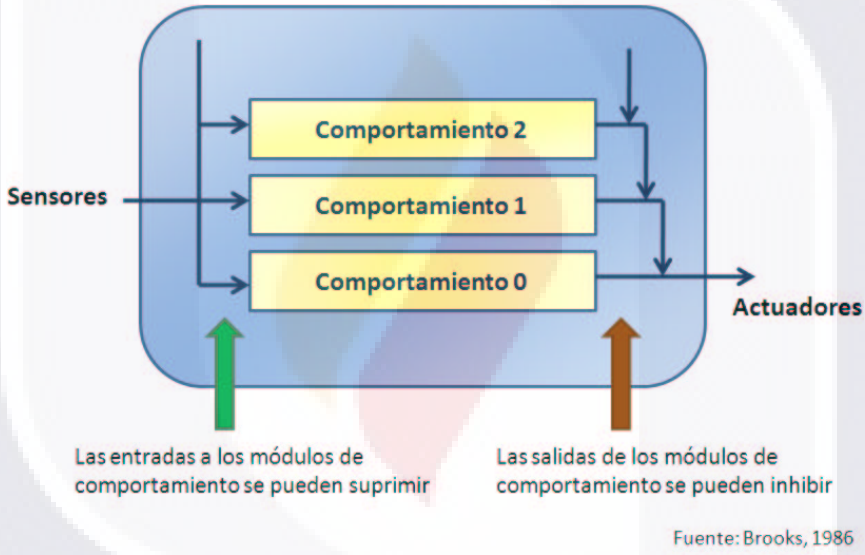


Figura 2.4: Arquitectura de subsunción

Arquitectura de subsunción cooperativa

En este tipo de agente reactivo la cooperación no emplea comunicación directa sino que ésta se realiza a través del entorno, hay dos comportamientos que se realizan en paralelo:

- Comportamiento de manejo de objetos
- Comportamiento de movimiento

Clasificación de agentes reactivos

Este tipo de sistemas está constituido por numerosos agentes homogéneos, sencillos, exibles, tratables (computacionalmente), robustos y tolerantes a fallos, y la inteligencia emerge del SMA.

Agentes organizados		Auto-organización
Agentes reproductores		Mecanismos de reproducción
Agentes cooperativos		Mecanismos de agrupación
Acciones coordinadas		Mecanismos de inhibicióon/Activacióon
Estímulo/respuesta		Autómatas de estados finitos

Tabla 2.3: Clasificación de agentes reactivos

Problemas para diseñar sistemas de agentes reactivos

- Los agentes necesitan conocer suficiente información sobre su entorno para actuar adecuadamente.
- La visión del agente es a corto plazo ya que está basada únicamente en información local.
- Es difícil el aprendizaje y la mejora de las capacidades de los agentes con el tiempo.
- Es difícil desarrollar agentes con muchas capas de comportamiento.

Arquitectura de Agentes Deliberativa

Este tipo de sistemas extienden las arquitecturas cognitivas de la IA. El proceso del agente introduce una función deliberativa entre la percepción y la ejecución para elegir la acción correcta.
Ciclo de ejecución de un agente deliberativo:

```
EstadoMental s;
ColaEventos eq;
...
s.inicializa();
mientras (verdadero){
opciones = generar opciones(eq,s);
seleccionando = delibera (opciones,s);
s.actualiza estado (seleccionando);
ejecutar(s);
eq.mira eventos();
}
```

Para actuar, este tipo de sistemas requieren de dos procesos:

Figura 2.5: Arquitectura deliberativa

- Decidir qué objetivos perseguir (deliberación).
- Decidir cómo alcanzar dichos objetivos (razonamiento basado en medios y fines).

Se basan en el razonamiento práctico (decidir en cada momento la acción a realizar para facilitar la consecución de los objetivos).
Un ejemplo de este tipo de arquitectura es la BDI (Beliefs-Desires-Intentions) en ésta las intenciones del agente juegan un importante papel en el razonamiento práctico:

- Dirigen el razonamiento basado en medios y fines
- Restringen las deliberaciones futuras
- Persisten
- Influencian las creencias en las que se basa el futuro razonamiento práctico

Cada cierto tiempo el agente deberá replantearse sus intenciones, abandonado aquellas que considera que no va a alcanzar, aquellas que ya ha alcanzado y aquellas cuya justificación ha desaparecido.
Aquí existen dos extremos:

- El agente que no reconsidera suficientemente a menudo sus intenciones (atrevido)
  - Comportamiento dirigido por objetivos

- El agente que continuamente reconsidera sus intenciones dedicando así un tiempo insuficiente a su consecución (precavido).
    - Comportamiento dirigido por eventos (reactivo)

Lo difícil es encontrar el equilibrio entre ambos comportamientos

- En entornos estáticos el comportamiento dirigido por objetivos es más adecuado
- En entornos dinámicos es necesario considerar cierta reactividad

Componentes de un agente BDI:

- Conjunto de creencias actuales que tiene el agente acerca de su entorno.
- Función de revisión de creencias (frc) que actualiza las creencias en base a las percepciones.
- Función de generación de opciones (deseos) a partir de sus creencias e intenciones.
- Conjunto de opciones actuales.
- Función filtro correspondiente al proceso de deliberación.
    - Determina las nuevas intenciones en función de sus creencias, deseos e intenciones.
- Conjunto de intenciones actuales.
- Función de selección de acciones que determina la acción a ejecutar a partir de las intenciones actuales.

El proceso de generación de opciones de un agente BDI es realmente un proceso recursivo de generación de una jerarquía de planes.
En cada paso se generan intenciones más especifícas hasta llegar a acciones directamente ejecutables, así el proceso de deliberación de un agente BDI debe ser ajustado en función de la variabilidad del entorno.
En la práctica el problema está en encontrar implementaciones eficientes de esta arquitectura.

Arquitecturas de Agentes Híbridas

Arquitecturas basadas en capas

Unas capas o subsistemas implementan el comportamiento, reactivo y otras el comportamiento deliberativo, las capas horizontales están conectadas a la entrada y salida del agente y en las capas verticales la entrada y la salida están conectadas a una única capa del agente.
A continuación algunos ejemplos de este tipo de arquitecturas.

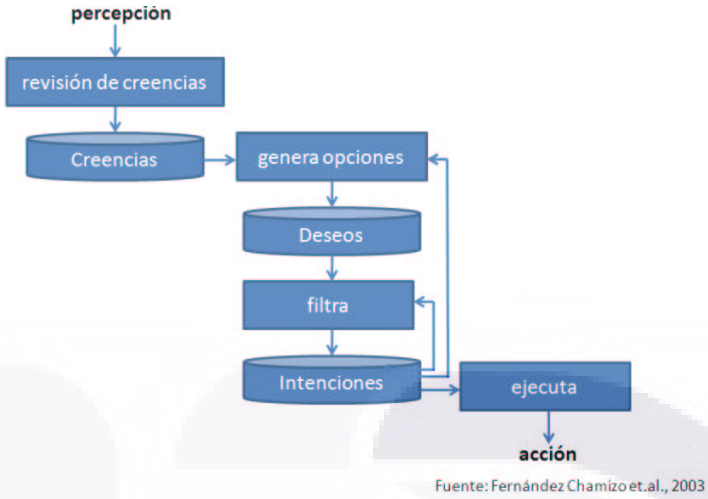


Figura 2.6: Arquitectura BDI

Arquitecturas de capas horizontales
Máquinas de Turing
Esta arquitectura [Ferguson, 1992] consiste de dos subsistemas, el de percepción y el de acción, y está formada por 3 capas horizontales:

- Modeladora (modelo del mundo para anticipar conflictos)
- Planificadora (comportamiento proactivo basado en esquemas)
- Reactiva (genera cursos potenciales de acción en respuesta a eventos)

Finalmente un subsistema de control basado en reglas que pueden inhibir entradas y salidas. Cada capa es un proceso independiente, productor de actividades y de ejecución concurrente.

Arquitectura de capas verticales
InterRap
Esta arquitectura [Muller, 1996] está formada por 3 capas verticales con 2 pasos:

- Capa de comportamiento (reactivo)
- Capa de planificación (pro-activo)
- Capa de cooperación (interacción social)

Uno de éstos pasos contiene la base de conocimiento y el otro contiene varios componentes de control. La interfaz está basada en comportamiento y su propósito es implementar y controlar la capacidad reactiva básica del agente.

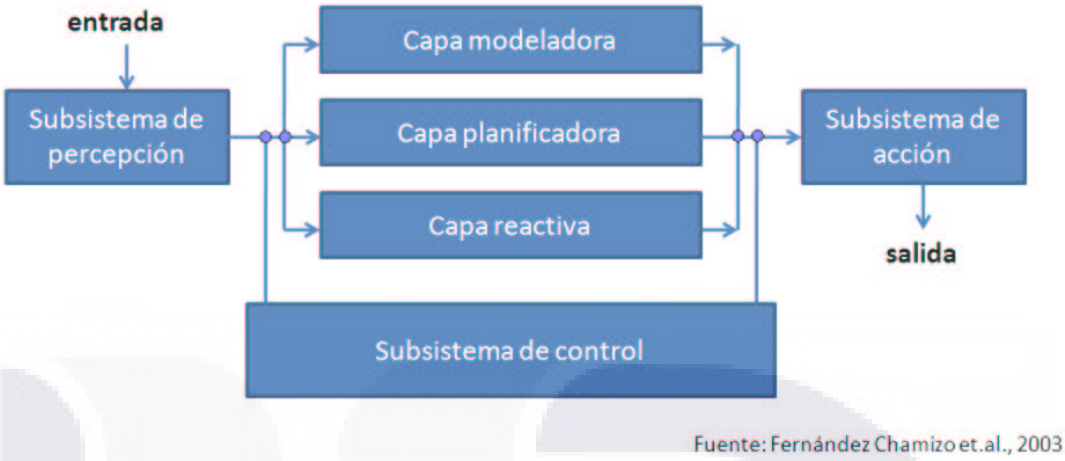


Figura 2.7: Máquina de Turing

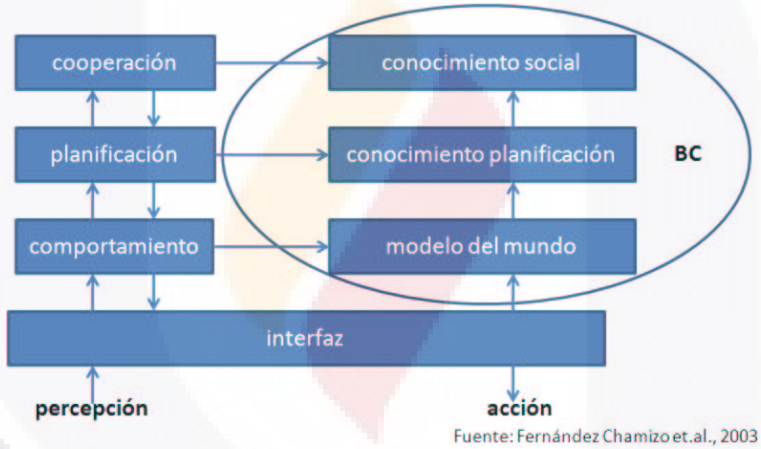


Figura 2.8: Arquitectura de InterRap

Las arquitecturas híbridas son de carácter general, y las más utilizadas. La propia arquitectura de subsunción de Brooks es otro ejemplo de este tipo de arquitecturas, utilizan una descomposición natural de la funcionalidad del agente, no tienen la claridad conceptual y semántica de otras arquitecturas, y la gestión de interacciones entre capas puede ser compleja.
Para las arquitecturas de capas verticales (Arquitecturas de uno o dos pasos) una ventaja es la complejidad de las interacciones entre capas se reduce, un inconveniente es que para tomar una decisión hay que pasar el control a cada una de las capas (No tolerante a fallos).
Para las arquitecturas de capas horizontales una ventaja es su simplicidad conceptual, si se necesita que un agente tenga n tipos de comportamientos, se implementan n capas diferentes, un inconveniente es que el comportamiento global puede no ser coherente, se suele añadir una función de mediación que

decide qué capa tiene el control en cada momento (cuello de botella).

Capítulo 3

EMPRESAS MANUFACTURERAS

3.1. Sistemas de producción

Como se establece en la Sección 1.4 el entorno de este trabajo de investigación son las *empresas manufactureras*, las cuales son sistemas de producción.
De manera general, podemos decir que la producción es el proceso por medio del cual se crean productos o servicios.
Y que implica un sistema con una gran cantidad de actividades interdependientes, formadas por distintas entidades.
Un sistema de producción es un sistema complejo, porque está formado de muchos elementos distintos (materiales, máquinas, energía y personas), algunos de los cuales son difíciles de predecir y controlar, como el suministro y el costo de materias primas, cambios en el mercado, en la conducta y en el desempeño humano.
Es difícil modelar un sistema tan complejo, por la falta de datos detallados o confiables sobre muchas de las variables que intervienen.
De acuerdo con el CAM-I (Consortium for Advanced Manufacturing International) *manufactura* es una serie de actividades y operaciones interrelacionadas que involucran diseño del producto, maquinarias y herramientas, planeación de procesos, materiales, compras, fabricación, servicios de apoyo, mercadeo, ventas, transporte y servicio al cliente [Giret, 2005].
La administración de la producción se ocupa de la toma de decisiones relacionadas con los procesos de producción de modo que los productos resultantes se produzcan de acuerdo con las especificaciones, en las cantidades y distribución requeridos y al costo mínimo.

Así, un sistema de producción utiliza recursos operacionales para transformar insumos en algún tipo de resultado deseado. Un insumo puede ser materia prima, un cliente, o un producto terminado proveniente de otro sistema [Chase, 2001].
El recinto que contiene físicamente a estos sistemas se denomina *fábrica*.
Una fábrica es un ambiente dinámico complejo donde las organizaciones de manufactura deben enfrentarse constantemente con la necesidad de rearreglos de producción [Cicirello & Smith, 2004].

3.2. Nuevas formas de manufactura

3.2.1. El sistema de manufactura

Un sistema de manufactura es el conjunto de procesos, maquinas, y fábricas donde se manejan materias primas para transformarlas en productos manufacturados de mayor valor.

3.2.2. Computadoras y manufactura

El empleo de computadoras en áreas de manufactura inicia durante los años 60

3.2.3. La Manufactura Integrada por Computadora (CIM)

En general, podemos decir que la Manufactura Integrada por Computadora (Computer Integrated Manufacturing) es una estrategia que incluye las metas de organización, formas de realizar el trabajo, desarrollo tecnológico y nuevos requerimientos de competencia, diseño, ingeniería, fabricación, lógistica, almacenamiento y distribución en una empresa de manufactura.

Figura 3.1: Pirámide CIM

Modelos CIM

- Modelo IBM (E.U., 1973) ⇒
- Modelo NIST (E.U., 1981)
- Modelo Digital Equipment Corporation (E.U., 1988)
- Modelo Siemens (Alemania, 1989)
- Modelo ESPRIT CIM-OSA (C.E., 1989)
- Modelo Amherst-Karlsruhe CIM (E.U. y Alemania 1989)

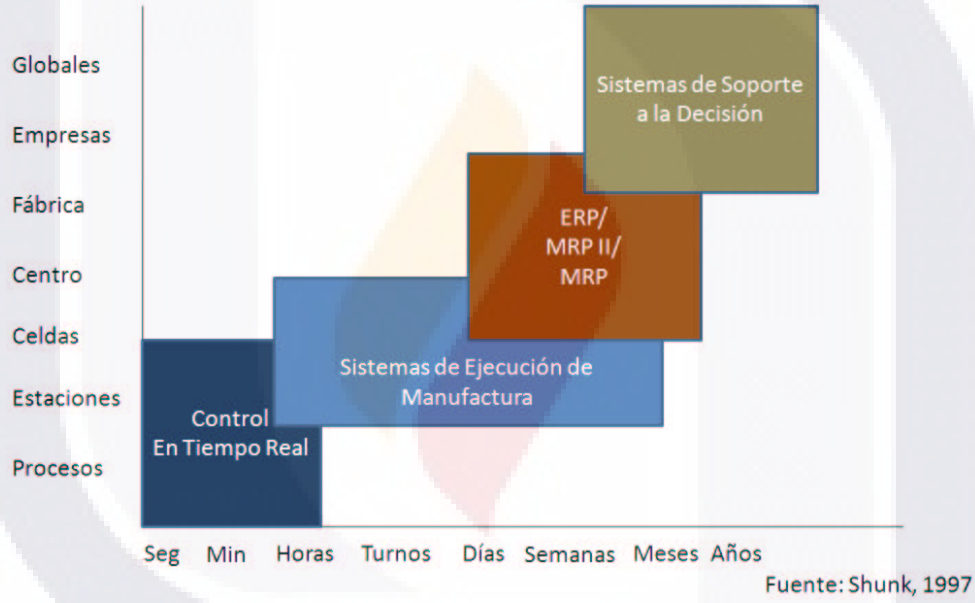


Figura 3.2: Niveles de automatización del CIM

Sistemas Flexibles de Manufactura (FMS)

Bajo el concepto CIM, una manera de organizar la producción son los Sistemas Flexibles de Manufactura (Flexible Manufacturing System).
Un FMS consiste en un grupo de máquinas controladas por computadoras y sistemas automáticos de manejo, carga y descarga de material, y de operación directa sobre el material; todo ello, controlado por un computador supervisor. Los elementos de este sistema son muy flexibles y versátiles, lo que permite una fabricación muy variada en el mismo momento [García & Grillo, 2005]. Los FMS están constituidos por células flexibles, las cuales, a su vez están formadas por:

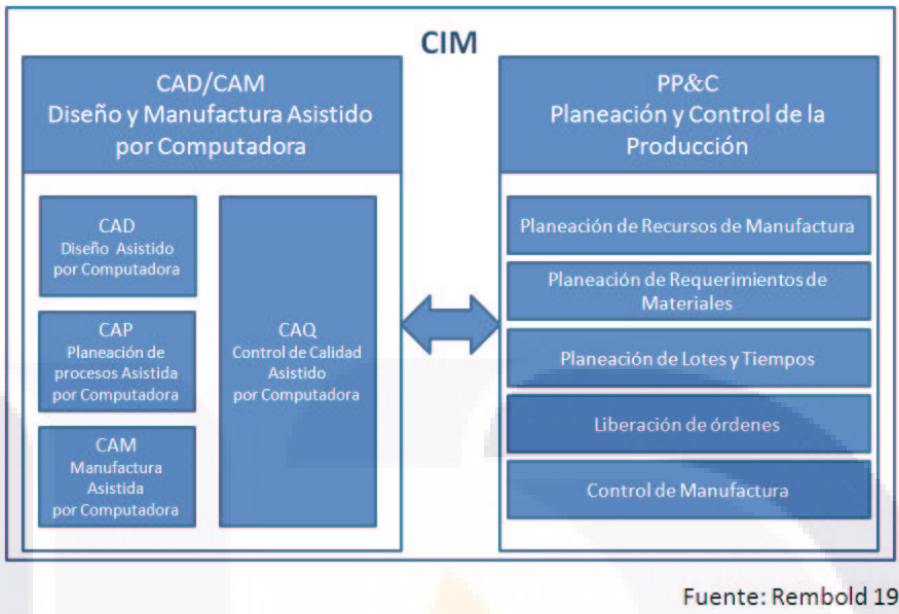


Figura 3.3: Esquema funcional de un CIM

- Máquinas CN
- Sistemas de transporte
- Sistemas de comunicación
- Computadoras

Características de un FMS

- Un grado importante de automatización
- Las máquinas deben ejecutar diferentes tareas en diferentes piezas
- Las máquinas deben tener tiempos de configuración despreciables
- El sistema debe tener muchas posibilidades de configuración

3.2.4. Sistemas de Manufactura Inteligente

Los sistemas de manufactura inteligente son organizaciones altamente distribuidas, donde la inteligencia se distribuye sobre las entidades individuales (agentes) los cuales son entidades autónomas, cooperativas e inteligentes.
El concepto de manufactura inteligente combina la habilidad de los sistemas de soporte a la toma de decisiones en sistemas generativos para obtener conocimiento, estos sistemas manejan la habilidad de aprender por sí mismos, así como la generación de la información necesaria para controlar el sistema de producción.

3.3. Caracterización del problema de manufactura

Una arquitectura jerárquica convencional para entornos de planeación y secuenciación de la producción incluye la planeación de órdenes de trabajo, la secuenciación (scheduling) del plan, la ejecución de las órdenes, el control de las máquinas para la ejecución de las órdenes, y el control de dispositivos para la operación efectiva, como se muestra en el siguiente diagrama [Giret, 2005]:

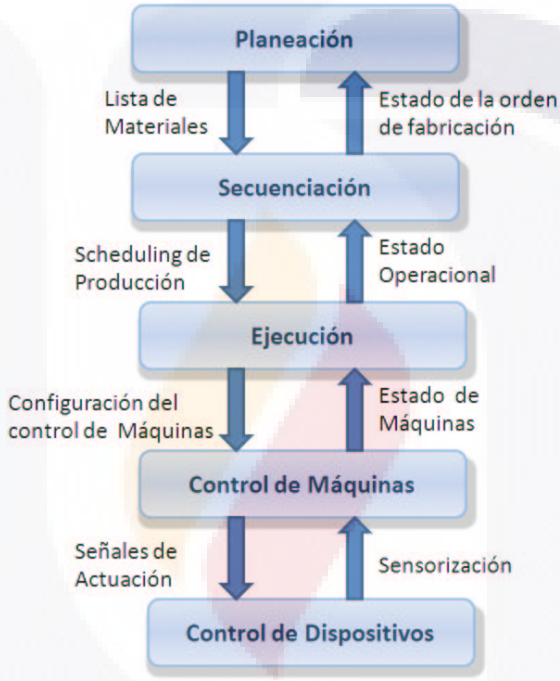


Figura 3.4: Jerarquía de control de fabricación

Planeación

A continuación se describen las actividades de planeación:

- La descomposición de una orden en una secuencia de operaciones de producción
- La asignación nominal de las operaciones a tipos de recursos

Scheduling

El scheduling en un entorno de fabricación discreta incluye:

- La asignación de las operaciones de producción a recursos específicos
- La especificación de tiempos (inicio, duración y finalización) para estas operaciones

Ejecución y Control de Taller
La ejecución y control en el taller involucra el inicio, control, monitorización y terminación de tareas con tiempos y parámetros de producción reales.

Control de Máquina y Dispositivo
El control de máquinas trata con el inicio, coordinación y monitorización de las diferentes funciones de máquina o dispositivos necesarios para la ejecución de las tareas de producción en una máquina. El control de dispositivos se ocupa de la actuación, sensorización y control de retroalimentación de las operaciones físicas que soporta una máquina o una unidad de proceso.

3.3.1. El problema de la planeación de la producción

El objetivo de la *planeación* en IA es la construcción de programas de computadora que sean capaces de obtener, en forma automática, planes de acción adecuados para alcanzar metas determinadas [Castillo Vidal, 1998] .
A partir de la forma en la que se construyen los planes, existen dos tipos de aproximaciones:

- *Planeación por refinamiento.* Aquí el *plan* se construye mediante la adición sucesiva de acciones y restricciones a una plan incialmente vacío.
- *Planeación por transformación.* Aquí el *plan* se construye no sólo añadiendo, sino también eliminando acciones y restricciones a partir de un plan inicialmente vacío.

Considerando el tamaño y la naturaleza de los componentes que se utilizan para construir un plan, podemos clasificar las aproximaciones como:

- *Planeación por generación.* Los componentes con los que se construye el *plan* son acciones y restricciones elementales (la mayoría de las aproximaciones empleadas en este tipo de problemas se ubican en esta clase) y donde se parte de cero para cada problema.
- *Planeación basada en casos.* Los elementos que componen el *plan* se pueden obtener a partir de planes completos o fragmentos de planes elaborados con anterioridad.

La planeación de la producción en empresas manufactureras

La satisfacción de la demanda de productos manufacturados es con frecuencia variable con respecto al tiempo, y las empresas fabricantes deben atender dicha demanda, lo que implica que se deben admitir diversas posibilidades que aseguren que la demanda es atendida convenientemente
[Castillo et.al., 2002].
Existen dos posibilidades:

- **Producción variable**. El fabricante puede producir cada período el número exacto de unidades que le solicitan. Sin embargo, como una producción que varía es costosa de mantener, por los costos de horarios más largos en los períodos de producción alta y los costos asociados al paro del personal y la maquinaria en los de producción baja; este tipo de producción no es eficiente.

- **Producción constante**. El fabricante que debe atender una demanda que cambia con el tiempo puede producir por encima de dicho nivel en periodos de baja demanda y almacenar la sobreproducción para los periodos de demanda mayor. Así, la producción puede mantenerse constante, compensando la demanda alta con la sobreproducción de periodos pasados. Sin embargo, debido a los costos de almacenamiento, tal opción puede no ser deseable si requiere costos altos de almacenamiento durante períodos prolongados.

Problemas de esta naturaleza muestran las dificultades que surgen cuando objetivos contrarios están presentes en un sistema dado. El objetivo de cualquier empresa es llevar a cabo una planeación de la producción que maximice los beneficios después de considerar los costos de las variaciones en la producción y los de almacenamiento.
Los cuatro elementos principales que intervienen en el problema de la planeación de la producción son:

1. **Datos:**

   n: el número de períodos a considerar

   s_0: la cantidad almacenada disponible al principio del periodo considerado

   d_t: el número de unidades (demanda) que se solicita en el período t

   s^{max}: la capacidad máxima de almacenamiento

   a_t: el precio de venta en el período t

   b_t: el costo de producción en el período t

   c_t: el costo de almacenamiento en el período t

2. **Variables:**

   x_t: el número de unidades producidas en el período t

   s_t: el número de unidades almacenadas en el período t

3. **Restricciones:**

   Como la demanda d_t en el período t debe coincidir con el cambio en el almacenamiento, $s_{t-1} - s_t$, más la producción x_t en el período t; la

capacidad de almacenamiento no puede excederse; y la demanda d_t, almacenamiento s_t, y producción x_t deben ser no negativas; se tienen las siguientes restricciones:

$$\begin{aligned} s_{t-1} + x_t - d_t = &\quad s_t; \quad t = 1, 2, ..., n \\ s_t \leq &\quad s^{max}; \quad t = 1, 2, ..., n \\ s_t \geq &\quad 0 \end{aligned} \tag{3.1}$$

4. **Función a optimizar:**

   - Una posibilidad en el problema de la planeación de la producción consiste en maximizar el ingreso después de descontar los costos de la variación de la producción y los inventarios; esto es, maximizar el beneficio.

     $$Z = \sum_{t=1}^{n} (a_t d_t - b_t x_t - c_t s_t) \tag{3.2}$$

     Si el periodo es corto, a_t, b_t, y c_t pueden considerarse constantes, esto es, $a_t = a$, $b_t = b$ y $c_t = c$.
   - Otra posibilidad es minimizar los costos de almacenamiento:

     $$Z = \sum_{t=1}^{n} c_t s_t \tag{3.3}$$

3.3.2. El problema de la secuenciación (scheduling) de la producción

Referencia 1485, 262 El principal objetivo de la secuenciación de la producción es una asignación eficiente de recursos compartidos en el tiempo respetando las restricciones para completar actividades.
La secuenciación de la producción ha sido tema de una cantidad significativa de literatura en el campo de la Investigación de Operaciones desde los años 50´s y su énfasis se ubica en investigar los problemas de secuenciación de maquinas donde los trabajos representan actividades y las maquinas representan los recursos.
El problema no es solamente NP-duro, sino que también tiene una bien ganada reputación de ser uno de de mayor dificultad computacional dentro los problemas de optimización combinatoria considerados hasta no hace mucho tiempo [Applegate & Cook, 1991].
En general, existen tres tipos de secuenciación de la producción en la industria manufacturera, que se muestran a continuación:

- **Producción abierta** (*open shop*): es la más fácil de programar, se cuenta con (m) máquinas, donde cada trabajo (J_k) puede ser realizado en cualquiera de ellas, no existen restricciones de precedencia y el programador determina la ruta a seguir de cada trabajo.

- Producción de flujo (*flow shop*): aquí cada trabajo sigue siempre la misma secuencia, la sucesión de operaciones es ordenada y no hay restricciones de precedencia ni retornos, es decir, si alguna maquina estuviera ocupada en el momento en el que llega una nueva tarea, ésta y las subsecuentes deberán esperar su turno para ser procesadas.

- Producción de taller (*job shop*): requiere la forma de secuenciación más compleja y hay 3 tipos de job shop, en el primero cada trabajo visita una maquina (M_j) una vez como máximo (sin recirculación) en el segundo, es cuando una tarea puede volver a la misma maquina (M_i) varias veces (con recirculación) en el tercero se encuentran los FMS, los cuales también soportan recirculación y pueden contar con máquinas paralelas (una máquina M_i en el modelo puede representar en realidad dos o más máquinas del mismo tipo).

El Problema de la Secuenciación de la Producción de Taller (Job Shop Scheduling Problem)

En el área de producción, secuenciación (*scheduling*) es la asignación de recursos compartidos sobre actividades que compiten en el tiempo.
En la terminología de manufactura, los trabajos (*jobs*) representan actividades y las maquinas (*machines*) representan recursos, el rango de las áreas de aplicación de la teoría de la secuenciación no esta limitado a computadoras y manufactura sino que incluye la de transportación, de servicios etc..
Aquí enfocaremos nuestra atención en la producción de taller determinística (job shop), donde toda la información que define una instancia del problema es conocido como en evolución.

Notación básica
Primero se van a definir algunos de los elementos básicos de notación del dominio [Liu J.S., 1996].

Una **operación** (*operation*) es una tarea elemental a ser realizada.

El **tiempo de procesamiento** (*processing time*) de una operaci on es la cantidad de tiempo requerido para procesar la operación; en muchos casos los tiempos de actualización son independientes de las relaciones de operación y se incluyen en los tiempos de procesamiento.

Un **trabajo** (*job*) es un conjunto de operaciones que están interrelacionadas por restricciones de precedencia derivadas de restricciones tecnológicas.

Una **máquina** (*machine*) es una pieza de equipo, un dispositivo, o un facilidad capaz de realizar una operación.

El **tiempo de recibo (ó fecha de recibo)** (*release time or release date*) de un trabajo es el tiempo en el cual es recibido en el piso de planta; es el menor tiempo en el cual la primera operación de un trabajo puede ser procesado.

La **fecha de entrega** (*due date*) de un trabajo es el tiempo en el cual la última operación del trabajo debe ser completada.

El **tiempo de terminación** (*completion time*) de un trabajo es el tiempo en el cual el procesamiento de la última operación de un trabajo es completada.

Un **programa** (*schedule*) es una especificación de la ejecución de cada operación sobre una maquina particular en un intervalo de tiempo específico.

Un **programa factible** (*feasible schedule*) es un programa que observa todas las restricciones del problema, por ejemplo, tiempo de recibo de trabajo, relaciones de precedencia de operación y capacidad de maquina.

Un **ruteo** (*routing*) de trabajo es un grupo secuencial de recursos a los que el trabajo accede antes de completarse.

El **tiempo de llegada** (*arrival time*) de un trabajo en una maquina es el tiempo en el cual el trabajo llega a la maquina previa en su ruteo, y es equivalente al tiempo de activo de la operación a ser realizada en una máquina.

Descripción del problema

El problema general de la secuenciación de producción de taller:

$$n \times m \tag{3.4}$$

de duración mínima (makespan) es definido por los símbolos:

$$n/m/G/C_{max} \tag{3.5}$$

y en lo adelante, el conocido como $JSSP$ (Job Shop Scheduling Problem) se puede describir como un conjunto de **n** trabajos (jobs)

$$\{J_i\} 1 \leq j \leq n \tag{3.6}$$

procesado por un conjunto de **m** maquinas (machines)

$$\{M_r\} 1 \leq r \leq m \tag{3.7}$$

El problema puede ser caracterizado como sigue:

Cada *trabajo* debe ser procesado por cada *maquina* en un orden dado por una *secuencia tecnológica* de maquinas predefinida. Cada *maquina* puede procesar un solo *trabajo* a la vez.

El procesamiento del *trabajo* J_j en la *maquina* M_r es llamado *operación* O_{jr}. La *operación* O_{jr} requiere el uso exclusivo de M_r en una *duración* interrumpida p_{jr}, que es el *tiempo de procesamiento*, y no se permite la *preadquisición* (preemption).

El *tiempo de inicio* y el *tiempo de terminación* de una *operación* O_{jr} se denota por s_{jr} y c_{jr} respectivamente, y donde un *programa* (schedule) es un conjunto de *tiempos de terminación* de cada *operación* $\{c_{jr}\} 1 \leq j \leq n$, $1 \leq r \leq m$ que satisface sus restricciones.

El tiempo requerido para completar todos los trabajos es llamado *duración* (makespan), el cual se denota por C_{max}.
Por definición $C_{max} = max1 \leq j \leq n,\ 1 \leq r \leq mc_{jr}$.
La *secuencia tecnológica* predefinida para cada *trabajo* puede ser dada colectivamente como una matriz $\{T_{jk}\}$ en la cual $T_{jk} = r$ corresponde a la *k-ésima operación* O_{jr} del *trabajo* J_i de la *maquina* M_r.
El objetivo de optimización del problema es encontrar un programa que minimice C_{max}.
Un ejemplo de un JSP de 3x3 es:

Trabajo			
1	1(3)	2(3)	3(3)
2	1(2)	3(3)	2(4)
3	2(3)	1(2)	3(1)

Tabla 3.1: Ejemplo de JSP

En este caso tenemos 3 trabajos y 3 maquinas, cada columna representa la secuencia tecnológica de maquinas por cada trabajo con los tiempos de procesamiento entre paréntesis.

Matriz de secuencia de trabajos $\{T_{jk}\}$

$$\{T_{jk}\} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$

y matriz de tiempos de procesamiento $\{p_{jk}\}$

$$\{p_{jk}\} = \begin{bmatrix} 3 & 3 & 3 \\ 2 & 3 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

para el problema de 3x3 dado.

Optimización
Es posible considerar otros objetivos de la secuenciación además de la minimización de la duración (makespan), tales como la minimización de la suma de los tiempos totales de todas las operaciones:

$$C_{sum}(C_{sum} = \Sigma 1 \leq j \leq n, 1 \leq r \leq mc_{jr}) \tag{3.8}$$

Gráfica de Gantt
La gráfica de Gantt es una forma conveniente de representar una solución visualmente del JSP.
Esta gráfica muestra las unidades de tiempo en las abscisas y los números de maquina en el eje de las ordenadas.

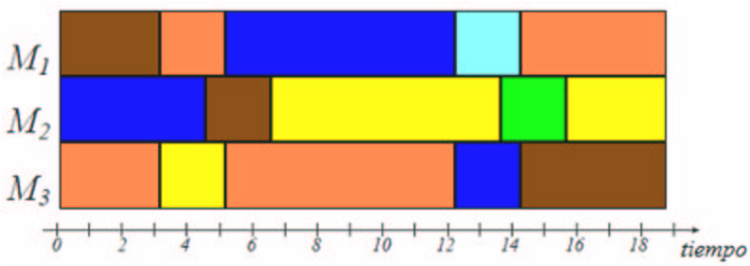


Figura 3.5: Gráfica de Gantt

Un ejemplo de la solución del problema de 3x3 se muestra en la siguiente figura:

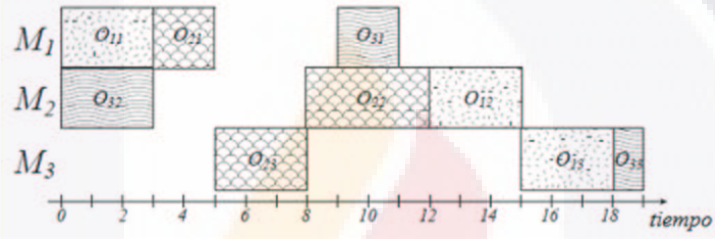


Figura 3.6: Solución al problema 3 x 3

Aquí, cada cuadro representa una *operación* O_{jr} con su extremo izquierdo puesto en s_{jr} como la coordenada x y con su longitud horizontal representando el *tiempo de procesamiento* p_{jr}
La *duración* de este programa es $C_{max} = 19$ unidades de tiempo.

Representación de Grafo Disyuntivo
La representación de la gráfica de Gantt y la de la matriz descrita en el apartado previo son simples y directas para identificar un programa.
Sin embargo no es obvio ver si el programa resultante es factible o no: así, sea que la secuencia de trabajo sobre cada maquina no se contradice con la secuencia tecnológica predefinida de maquinas.
Una formulación mas informativa del problema basada en representaciones gráficas es:
El JSSP se puede describir por un grafo disyuntivo

$$G = (V, C \cup D) \tag{3.9}$$

donde:

- V es un conjunto de nodos que representan operaciones de los trabajos junto con dos nodos especiales, uno fuente (0) y un fondo ($\star$), representando el inicio y el fin de un programa, respectivamente.

- C es un conjunto de arcos conjuntivos que representan secuencias tecnológicas de maquinas para cada trabajo.
- $D = U_{mr} = 1D_r$, donde D_r es un conjunto de arcos disyuntivos que representan pares de operaciones que deben ser realizadas por la misma maquina M_r.
- El tiempo de procesamiento para cada operación es un valor pesado p_v unido al correspondiente nodo v, por el par de nodos especiales

  $p_0 = p_* = 0$.

La siguiente figura muestra un grafo disyuntivo que representa el problema 3x3 dado:

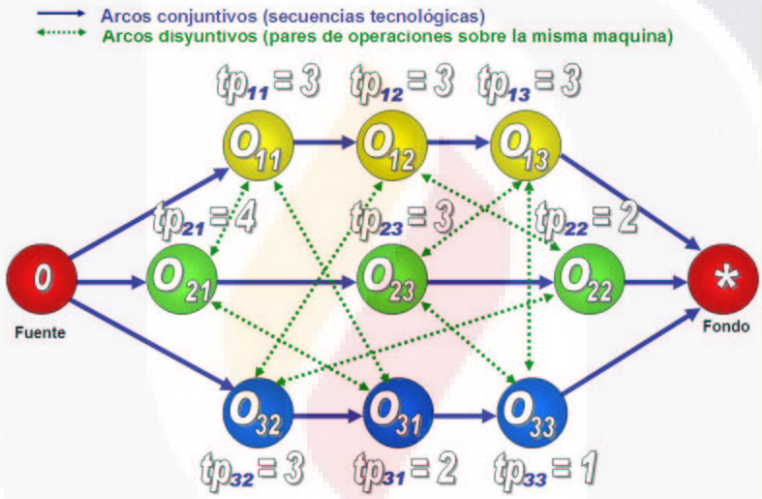


Figura 3.7: Grafo disyuntivo

Medidas de rendimiento

A continuación se definen una serie de funciones objetivo que las organizaciones emplean para optimizar programas de producción basadas en criterios económicos [Liu J.S., 1996].

- **Tiempo de terminación** (*Completion time*) $\rightarrow C_i$ Es el tiempo en el cual un trabajo J_i completa su procesamiento, es el tiempo final de la última operación de J_i.
- **Tiempo de Flujo** (*Flow time*) $\rightarrow F_i$ Es el lapso de tiempo que un trabajo J_i espera en el piso de planta, corresponde al intervalo entre el tiempo de liberación de J_i y el tiempo de terminación de J_i.

$$F_i = C_i - rt_i \tag{3.10}$$

- **Tardanza** (*Tardiness*) → T_i Es el lapso positivo de tiempo en el cual el tiempo de terminación de J_i excede de la fecha de entrega dd_i. Sí J_i se completa antes que dd_i la tardanza es cero.

$$T_i = max[C_i - dd_i] \tag{3.11}$$

- **Tiempo de Flujo promedio** (**Flow time mean**) → F_{mean} Es el tiempo de flujo promedio de un programa.

$$F_{mean} = \frac{1}{m}\sum_{i=1}^{m} F_i \tag{3.12}$$

donde m es el número de trabajos.

- **Trabajo en proceso** (*Work-in-process*) Es el promedio de los tiempos de procesamiento en progreso de un programa.

$$WIP = \frac{1}{m}\sum_{i=1}^{m}(C_i - st_{i1}) \tag{3.13}$$

donde m es el número de trabajos y st_{i1} es el tiempo en el cual un trabajo J_i empieza su procesamiento, o el tiempo de inicio de la primera operación de J_i.

- **Tardanza promedio** (*Mean Tardiness*) → T_{mean} Es la tardanza promedio de un programa.

$$T_{mean} = \frac{1}{k}\sum_{i=1}^{m} T_i \tag{3.14}$$

- **Tardanza promedio condicional** (*Conditional Mean Tardiness*) → CT_{mean} Es la tardanza promedio sobre los trabajos atrasados de un programa.

$$CT_{mean} = \frac{1}{k}\sum_{i=1}^{m} T_i, T_i < 0 \tag{3.15}$$

- **Proporción de trabajos atrasados** (*Proportion of Tardy jobs*) → PT Es el porcentaje de trabajos que están atrasados en un programa.

$$PT = \frac{k}{m} \tag{3.16}$$

donde k es el número de trabajos con retraso diferente de cero, y m es el número de trabajos. Por lo que la tardanza promedio y la tardanza promedio condicional están relacionadas por $Tmean = CT_{mean} \times PT$.

- **Tardanza Pesada** (*Weighted Tardiness*) $\rightarrow T_wt$ Es la suma de la tardanza proporcional multiplicada por la importancia del trabajo.

$$T_{wt} = \sum_{i=1}^{m} w_i T_i \tag{3.17}$$

  donde w_i es el peso de un trabajo J_i

- **Duración de un programa** (*Makespan*) Es la longitud de un programa, o el intervalo entre el tiempo en el cual el programa inicia y el tiempo en el cual el programa termina. Usualmente se marca el inicio de un programa como tiempo 0. Por lo que la duración de un programa es el máximo de los tiempos de terminación del conjunto de trabajos en el programa.

$$Makespan = max[C_i], i = 1, ..., m \tag{3.18}$$

Procedimientos (reglas) de despacho en una planta de producción intermitente

Una forma relativamente simple de elaborar un programa de producción es mediante las llamadas reglas de despacho, las cuales dictan el orden en el cual los trabajos son procesados por cada máquina, a continuación las primeras reglas que están basadas en un solo parámetro (reglas simples).

- **Primero en entrar primero en salir** (**F***irst* **C***ome-***F***irst* **S***erved*) Esta regla de prioridad está basada en el orden de arribo de los trabajos y usa el tiempo final de la operacióon previa como índice de prioridad para la operación que espera a ser procesada, la operación con el índice de prioridad mínimo es rankeada primero para ser despachada, esta regla es muy fácil de implementar.

- **Fecha de vencimiento más próxima** (**E***arly* **D***ue* **D***ate*) Los trabajos son programados de acuerdo con su fecha de vencimiento, el trabajo con la fecha de vencimiento más próxima es el de mayor prioridad.

- **Trabajo actual** Esta regla de prioridad está diseñada guardar los tiempos de actualización de todos los trabajos que requieren la misma secuencia de actualización, así se elimina la necesidad de actualizar operaciones. El problema con esta regla es que los trabajos con una fecha de vencimiento más próxima pueden ser retrasados mientras que otros trabajos menos urgentes se están procesando.

- **El tiempo de procesamiento más corto** (**S***hortest* **P***rocessing* **T***ime*) Esta regla de prioridad trata de minimizar el número de trabajos que esperan al frente de la máquina, procesando los trabajos más cortos primero. El problema es que los trabajos más largos pueden ser completados después mientras que los trabajos cortos son completados antes de su fecha de vencimiento.

Además de las reglas de prioridad simple anteriores, hay una variedad de reglas más complejas entre las que tenemos:

- **Razón critica** (**C***ritical* **R***atio*) Esta regla está basada en la diferencia entre la fecha de vencimiento y la fecha actual divida entre el tiempo requerido para completar el trabajo anterior. Los trabajos con valores pequeños de CR son los que tienen mayor prioridad.
- **Holgura de tiempo restante** (**S***lack* **T***ime* **R***emaining*) Esta regla está basada en la diferencia entre el tiempo restante antes de la fecha de vencimiento y el tiempo requerido para procesar los trabajos restantes. El valor más pequeño de STR es el del trabajo con mayor prioridad.
- **Holgura de tiempo restante por Operación**

  (**S***lack* **T***ime* **R***emaining*/**OP***eraction*) Esta regla está basada en el promedio de la holgura de tiempo restante por operación calculada como la relación entre el STR y el número de operaciones restantes. La mayor prioridad se asigna a los trabajos con el menor valor de STR/OP.

3.3.3. El problema del control de la producción

Para obtener un producto manufacturado, los materiales que lo integran sufren varios tipos de transformaciones realizadas por las maquinas (actuadores) que componen el sistema de manufactura, cuyo funcionamiento es coordinado por un programa de control secuencial y la supervisión de un operador humano.
El diseño del programa de control secuencial que gobierna al sistema de manufactura es un problema complejo que básicamente es realizado en forma manual por ingenieros especializados en este tipo de control, esta forma de trabajo tiene las siguientes desventajas [Castillo Vidal, 1998]:

- Es un proceso lento y complicado que puede dar lugar a errores humanos
- El proceso se complica aún más ante la necesidad de adaptarse a la evolución del ciclo de vida del propio sistema de manufactura

Propiedades de los Programas de Control
Los programas de control secuencial y los sistemas de producción sobre los que actúan presentan las siguientes **propiedades estructurales** que describen las distintas formas en que se pueden llevar a cabo un conjunto de operaciones:

- *Secuenciación* es la posibilidad de realizar una operación a continuación de otra.

- *Concurrencia* es que dos operaciones se puedan llevar a cabo de manera paralela a la vez, lo cual es deseable (en las situaciones donde sea posible) ya que permite una ejecución más rápida que si las operaciones se realizarán secuencialmente.

- *Existencia de conflictos* es la existencia de varias alternativas, es decir a la hora de realizar una operación el que existan distintas vías para hacerlo.

La importancia de estas propiedades radica en que el formalismo que se utilice para representar un programa de control debe ser suficientemente rico en expresividad como para poder representar estas caraterísticas estructurales. Además de estas propiedades estructurales, todo buen programa de control debe exhibir una serie de **propiedades de buen funcionamiento**.

- *Compartición de recursos* si dos operaciones o secuencias de operaciones paralelas intentan utilizar un mismo recurso para realizar una acción y este recurso no puede ser utilizado en dos acciones a la vez, la ejecución de estas acciones debe sincronizarse correctamente.

- *Ausencia de bloqueos* en ocasiones, la ejecución de un programa de control puede entrar en un proceso sin salida y bloquearse, con el consiguiente riesgo para el sistema de producción. Un programa de control puede asegurar que esto no ocurra y que ninguna combinación de eventos procedentes del exterior le haga entrar en un bloque.

Un programa de control responde constantemente a los estímulos que recibe del exterior los cuales están originados por el correcto funcionamiento del sistema de producción y su principal función es advertir la evolución de las operaciones

3.3.4. Los sistemas de tiempo real

En este trabajo se va a entender por tiempo real al procesamiento de la información, de manera determinista y acotada en el tiempo [Benítez, 2002]. Actualmente, un sistema de tiempo real (**STR**) es un sistema computacional que responde a eventos o entradas del mundo exterior en un período de tiempo finito específico. Desde este punto de vista, todos los sistemas de control son sistemas de tiempo real, y existen dos tipos, uno de ellos son los sistemas embebidos, es decir la computadora está integrada a alguna parte del equipo o maquinaria [Arzén, 2006] en la cual están ubicados físicamente y a la que controlan, y se pueden encontrar en controles de automóviles, telefónos celulares, electrodomésticos, etc..
El otro tipo son los sistemas de control industrial los cuales se emplean para controlar procesos industriales e incluyen a los sistemas de control distribuido, controladores lógicos programables, y estaciones de trabajo industriales.

Los STRs son sistemas que interactúan con el mundo exterior y dependiendo de la información que reciben del entorno deciden las acciones a realizar. Si el ambiente es dinámico, deben tomar y aplicar las decisiones antes de que las condiciones del entorno cambien. Estos sistemas están sujetos a las restricciones temporales del ambiente en el que se desenvuelven incluyendo tiempos de inicio y tiempos de terminación (*deadline*).
Por lo que, en los STRs, el buen funcionamiento implica no solo la corrección lógica de sus resultados sino también su corrección temporal [Stankovic, 1990].
Así, dos de las propiedades más importantes de un STR son la predecibilidad funcional y la temporal, la predecibilidad temporal es la capacidad de predecir a priori que una tarea se realizará antes de su plazo máximo de ejecución.
Existen dos tipos de requerimientos de trabajos o tareas relacionados con el tiempo:

- *Requerimiento de tiempo real estricto* (**hard**): Es aquel en el que el incumplimiento de su plazo máximo hace que la utilidad de la tarea sea nula, incluso puede provocar la degradación del sistema.
- *Requerimiento de tiempo real no-estricto* (**soft**): Es aquel en el que el incumplimiento de su plazo máximo hace que la utilidad de la tarea disminuya, pero el sistema continua funcionando.

Los primeros STR se utilizaron para controlar pequeñas aplicaciones perfectamente caracterizadas donde se conocían todos los tiempos de ejecución de las tareas a realizar, aquí la planificación de las tareas tenía una importancia fundamental para asegurar que se cumplierán todos los plazos máximos de ejecución, lo que aseguraría que se eviten fallos que tendrían graves consecuencias para el sistema.
Luego se requirieron STRs más complejos para controlar y monitorear procesos que implicaban diferentes modos de operación, y las computadoras resultaron una conexión natural entre los sistemas de control tradicional y estos procesos.
A continuación se mencionan los tipos de eventos a los que debe responder un STR:

- Períodicos
- No-Períodicos
  - a-períodicos
    - frecuencias de arribo ilimitadas
  - esporádicos
    - frecuencias de arribo limitadas

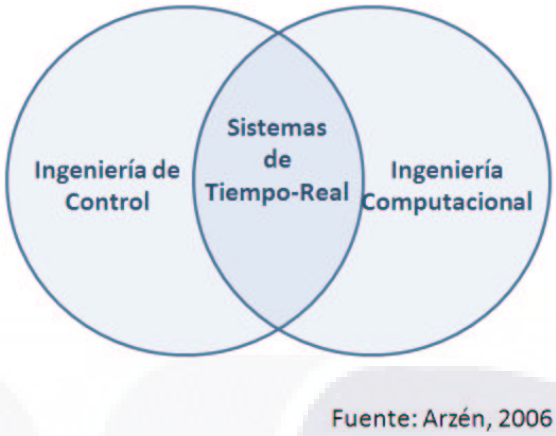


Figura 3.8: Control, Computación y Tiempo Real

Los eventos pueden ser internos o externos y cada evento puede requerir una cierta cantidad de procesamiento y tener un tiempo de terminación.
Todos los STR son *concurrentes*, es decir, los múltiples eventos pueden ocurrir al mismo tiempo y los trabajos que deben realizarse para servir a una tarea son llamados trabajos asociados al evento [Stankovic, 1990].

Capítulo 4

ARQUITECTURAS DE SMA ORIENTADAS A APLICACIONES INDUSTRIALES

4.1. Arquitecturas

En el transcurso de los últimos años los SMA se han utilizado en una gran variedad de aplicaciones, entre las cuales están aquellas que tiene que ver con los procesos industriales.
Una clasificación de este tipo de aplicaciones que considera, tanto el tipo de aplicación como el rol que juegan los agentes dentro de ella, es la planteada por [Caridi & Cavalieri, 2004], donde además se específica el porcentaje de artículos encontrados de cada tipo de aplicación (ver Tabla 4.1).
Como puede apreciarse en dicha tabla, ha habido una gran cantidad de esfuerzos de investigación orientados hacia los sistemas de ingeniería, planificación y programación de la producción (aprox. 45 %), aunque cabe mencionar que el nivel de madurez de dichos trabajos con respecto a las aproximaciones tradicionales en este campo (principalmente las que provienen de las áreas de Ingeniería Industrial e Investigación de Operaciones) es aún muy bajo. El estudio de [Caridi & Cavalieri, 2004] señala, por ejemplo, que sólo se encontr ó un trabajo a nivel de aplicación en fase final de producto en el área de scheduling y monitoreo de procesos continuos [Parunak et.al. 2001b]; en aplicaciones en fase de producción (previa a producto final acabado) una en el área de cadena de gestión de suministro [Sadeh et al. 1999] y varias en el área de scheduling y ensamble [Parunak 1998]; la mayoría de las aplicaciones

Tipo de aplicación	Rol de los agentes	% de articulos consultados
Orden de cotización	Gestor de costos	5 %
Diseño	Agentes de diseño	13 %
Ingeniería	Agente de diseño de procesos o fabricación	9 %
Previsión de demanda	Agente de ventas o marketing	5 %
Gestión de órdenes	Agentes, holones de órdenes	7 %
Programa Maestro de Producción	Agente planificador	6 %
Planeación de Requerimientos de Materiales (MRP)	Agente planificador	9 %
Scheduling	Agente programador	20 %
Aprovisionamiento	Agente de suministro	7 %
Monitoreo	Agente controlador	14 %
Distribución	Agente de almacén	5 %

Tabla 4.1: Tipos de aplicaciones industriales

se encuentran aún en fase de modelado o de emulación.
A continuación se analizan algunas de las arquitecturas propuestas en el dominio de procesos industriales:

4.2. Arquitectura AARIA

4.2.1. Información de referencia

Fuentes: [Parunak et.al. 2001b], [Parunak et al.1997], [Baker et al. 1997]
Nombre: Agentes Autónomos para el Arsenal de Rock Island (Autonomous Agents for Rock Island Arsenal)
Creadores: H. Van Dyke Parunak, Albert D. Baker, Steven J. Clark
Dominio: Programación de la Producción
Organización diseñadora: ERIM, University of Cincinnati, Real World Interface, Inc.
Organización receptora: Arsenal de Rock Island, Ejército de los EEUU.

4.2.2. Objetivos

Mediante un analisis de requerimientos se define el problema que el sistema debe resolver, es decir, se concentra en el *porqué* debe realizarse el esfuerzo más en el *qué* o el *cómo*.
Los requerimientos se dividen en dos categorías, las impuestas por la interface del sistema con su ambiente externo y aquellas operaciones internas necesarias para cumplir con su funcionalidad.
Un sistema de producción requiere facilitar las relaciones entre cuatro grupos diferentes:

- Los **Clientes** quienes crean enlaces de entradas de productos.
- Los **Proveedores** quienes crean enlaces de salidas de productos.
- Los **Operadores** quienes hacen que el sistema de producción funcione.
- Los **Ingenieros de manufactura y administradores** quienes son responsables del diseño, instalación, modificación y mantenimiento de la fábrica.

Para lograr lo anterior se establecen los siguientes objetivos:

Interfaces Externas

- **Apoyar en la interacción Cliente/Proveedor** - cada orden de cliente ocupa un punto en un espacio de 5 dimensiones:
  - Identificación del producto
  - Cantidad
  - Precio
  - Tiempo de entrega
  - Tiempo de respuesta

  El proveedor y el cliente deben trabajar cercanamente para identificar una región en este espacio multidimensional que satisfaga a ambos.
- **Apoyar al personal operativo y equipo** - las organizaciones modernas buscan empoderar al personal operativo proporcionándoles toda la información necesaria para controlar el sistema.
- **Apoyar a los administradores y a los ingenieros de manufactura** - son los responsables de diseñar, instalar, modificar y mantener el sistema. La simulación es una poderosa herramienta para evaluar y selecionar alternativas si se emplea el mismo código que en la operación. Esto extiende el empoderamiento de la producción a la administración.

Operaciones Internas

- **Cambios frecuentes** - un sistema que puede auto-organizarse en respuesta a los cambios frecuentes será más eficiente.
- **Funcionalidad ERP** - el paradigma dominante en la planeación de la manufactura es la Planeación de Recursos de la Empresa, la cual constuye programas de producción en base a las ordenes de producto de materiales, la aceptación de cualquier nuevo paradigma debe contar con esta funcionalidad.
- **Modalidades de control de fábrica diferentes** - ninguna operación puede ejecutarse antes de que materiales y recursos estén disponibles, las modalidades de control se clasifican en programación avanzada, empuje (kanban) y reglas de despacho.
- **Metmorfosis** - un buen sistema debe ser capaz de reconfigurarse (evolucionar) partiendo de su estado actual hacia un estado igualmente eficiente en el futuro.
- **Presentación Uniforme externa/interna** - cualquier sistema de manufactura debe ser capaz de interactuar amigablemente con sus clientes y proveedores y tener funcionalidades tales como escalamiento, transparencia de subcontratación, y alternativas adecuadas sobre la decisión de fabricar o comprar.

4.2.3. Tipo de Arquitectura

AARIA incluye mecanismos que permiten dividir la fábrica en subconjuntos que ocupan una posición en este espacio y cambian su modalidad cuando cambian las circunstancias, a este proceso le llaman Modalidad de Emergencia.

4.2.4. Diagramas

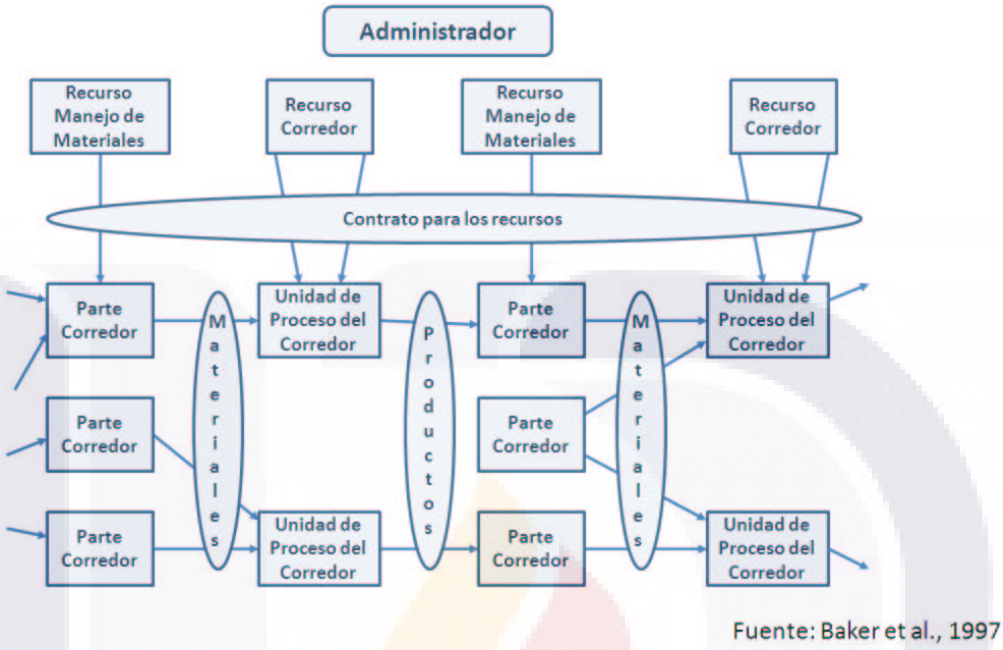


Figura 4.1: Arquitectura AARIA

4.2.5. Modelo de agente

Forma de identificar a un agente

La forma utilizada es la Teoría de casos linguística [Parunak et al.1997], técnica análisis de casos. Se realiza una narración de sentencias de la forma (Joe supervisó la Unidad de proceso 12 sobre la parte 25 del proveedor Acme, usando un Taladro 32, Cortador 86 y Programa de parte 19 produciendo la parte 26 para la orden del ejercito 22). Para validar a los candidatos a agentes se emplea la técnica **Cosa vs Función**, o descomposición funcional, además es deseable que los agentes sean los más pequeños posible, así serán más fáciles de construir y de entender y el impacto en la falla de alguno de ellos sobre el sistema completo será despreciable.

Con un conjunto de candidatos a agentes identificados se define su comportamiento (hipotético) individual y la clase de mensajes que pueden intercambiar.

En AARIA se asigna un agente a cada capacidad de manufactura indiviual que pueden ser recursos (es decir, personas, máquinas, facilidades), tipos de partes o unidades de procesamiento.

Tipos de Agentes

Se tienen dos tipos de agentes:

- *Agentes persistentes* - son aquellos que no dejan, ni se incorporan al sistema sin la intervención humana
- *Agentes transitorios* - representan entidades que tienen un ciclo de vida específico en el sistema (nacimiento, requerimiento, comisión, disponibilidad, actividad, muerte y archivamiento)

4.2.6. Mecanismo de comunicación

AARIA emplea cuatro mecanismos para la comunicación

- Difusión -
- Jeraquía -
- Flujo de proceso -
- Mediación -

El mecanismo mediante el cual interactúan los agentes en AARIA es numérico balanceado, es un mecanismo de mercado empleado extensivamente como un medio para lograr uniformidad.

4.3. Arquitectura ARCHON

4.3.1. Información de referencia

Fuente: [Cockburn & Jennings 1993]
Nombre: Arquitectura para Sistemas Cooperativos Heterogeneos en Línea (ARchitecture for Cooperative Heterogeneous ON-line systems)
Creadores: D. Cockburn, N. R. Jennings **Dominio**: Abastecimiento y distribución de electricidad, control de fábricas complejas, control de aceleradores de partículas y control robótico.
Organización diseñadora: EA Technology, University of London.
Organización receptora: .

4.3.2. Objetivos

El diseño de un sistema ARCHON usa una aproximación mixta de arriba hacia abajo (*top down*) y de abajo hacia arriba (*bottom up*).

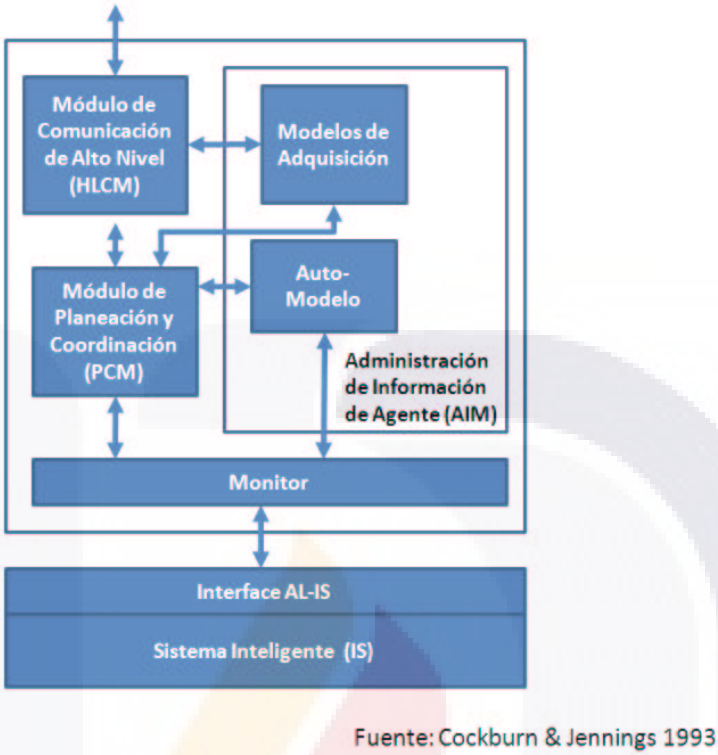


Figura 4.2: Arquitectura de un Agente ARCHON

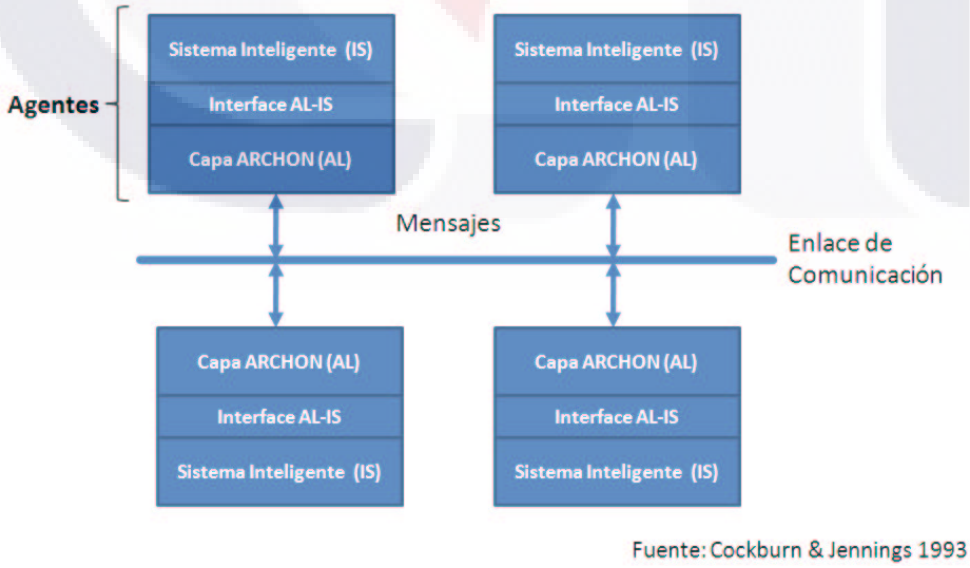


Figura 4.3: Estructura de una Comunidad ARCHON

4.3.3. Diagramas

4.4. Arquitectura PROSA

4.4.1. Información de referencia

Fuentes: [Peeters et al., 1998], [Vestraete et al., 2006]
Nombre: Arquitectura de agentes de Producto-Recurso-Orden-Personal (Product Resource Order Staff Architecture)
Creadores: Patrick Peeters, Paul Valckenaers, Jo Wyns, Luc Bongaerts
Dominio: General.
Organización diseñadora: Katholieke Universiteit Leuven, A.I.Systems Brussels, Daimler-Benz AG, Research & Technology Berlin, University of Cambridge, VTT Automation.
Organización receptora: .

4.4.2. Objetivos

- Los agentes distribuirán información sobre el estado local del sistema. Esto incluye la información sobre la configuración lógica y física del sistema y las habilidades y capacidades de los recursos capaces de responder a cambios y disturbios.
- La infraestructura de la información será accesible a los humanos (a través de interfaces hombre-maquina) para permitir el control guíado por humanos de decisión o monitoreo.
- Las decisiones lógicas se tomarán considerando principalmente cambios y disturbios.
- Las decisiones lógicas deben ser reemplazadas con poco esfuerzo (es decir, sin modificaciones referidas para la infraestructura de la información, o los tipos de agentes) y capaces de adaptarse a los cambios tales como los de las funciones meta o restricciones.
- La estructura deberá ser abierta por lo que el algoritmo puede ser extendido agregando fuentes de información y agentes de toma de decisiones. La infraestructura deberá soportar un comportamiento orientado a la robustez y proactivo.

4.4.3. Modelo de agente

Holon: Es un bloque de construcción autónomo y cooperativo de un sistema de manufactura que transforma, transporta, almacena y/o valida información y objetos físicos, consiste de una parte que procesa información y

una parte física de procesamiento, un holon puede ser parte de otro holon [Brussel et al., 1998].
En PROSA hay tres tipos de holones que tienen las siguientes responsabilidades:

- *holon recurso* - ofrece capacidades de producción y funcionalidad a los holones que lo rodean, contiene una parte física (recurso de producción del sistema de manufactura) y una de procesamiento de información (que controla el recurso).
- *holon orden* - proveen conocimiento a los agentes orden sobre las rutas y las secuencias de procesamiento disponibles que son capaces de crear un producto adecuado.
- *holon producto* - controla a su recurso (actuador) y mantiene su modelo de estado interno sincronizado con el recurso a través de la entrada adecuada (sensor).

4.4.4. Mecanismo de comunicación

Feromonas:

4.5. Arquitectura manAge

4.5.1. Información de referencia

Fuente: [Heikkila et al.1999] **Nombre**: Agentes para manufactura
(manufacturing Agent)
Creadores: Tapio Heikkila, Martin Kollingbaum, Paul Valckeneaers, Geert-Jan Bluemink
Dominio: .
Organización diseñadora: VTT Automation, University of Cambridge, Katholieke Universiteit Leuven.
Organización receptora:

4.6. Arquitectura MASCOT

4.6.1. Información de referencia

Fuente: [Sadeh et al., 1999]
Nombre: Herramienta de Coordinación para Cadena de Suministro Multi-agente
(Multi Agent Supply Chain cOrdination Tool)
Creadores: Norman M. Sadeh, David W. Hildum, Dag Kjenstad, Allen Tseng

Dominio: Cadena de Suministro.
Organización diseñadora: Intelligent Coordination and Logistics Laboratory, The Robotics Institute, School of Computer Science, Carnegie Mellon University and SINTEF Applied Mathematics, Department of Optimization, Oslo Norway.
Organización receptora: .

4.6.2. Objetivos

Los agentes MASCOT son shells que soportan las siguientes funcionalidades:

1. *Coordinación* los agentes MASCOT actúan como cubiertas para módulos de planeación y programación distribuidos a lo largo de una cadena de suministro soportando protocolos de coordinación lateral y vertical.

2. *Integración con módulos heterogéneos de planificación y programación* cada agente MASCOT está configurado alrededor de una arquitectura de pizarra.

3. *Funcionalidades de soprte a la decisión de iniciativa mixta* la pizarra de cada agente MASCOT está particionada en contextos que corresponden a un grupo posible de diferentes asunciones que habilitan a un usuario final a comparar diferentes compromisos tanto localmente como a través de la coordinación con otros agentes en la arquitectura.

4. *Reconfigurabilidad* los agentes MASCOT puede reconfigurarse rápidamente para acomodarse a la introducción de nuevos productos, nuevos flujos de productos, nuevas facilidades etc..

4.6.3. Diagramas

4.6.4. Modelo de agente

Esta arquitectura se construye alrededor de un personalizable y de iniciativa mixta agente envuelto, el cual incluye las siguientes funciones:

4.6.5. Mecanismo de comunicación

Pizarra

4.7. Arquitectura SRTA

4.7.1. Información de referencia

Fuente: [Horling et al., 2006]

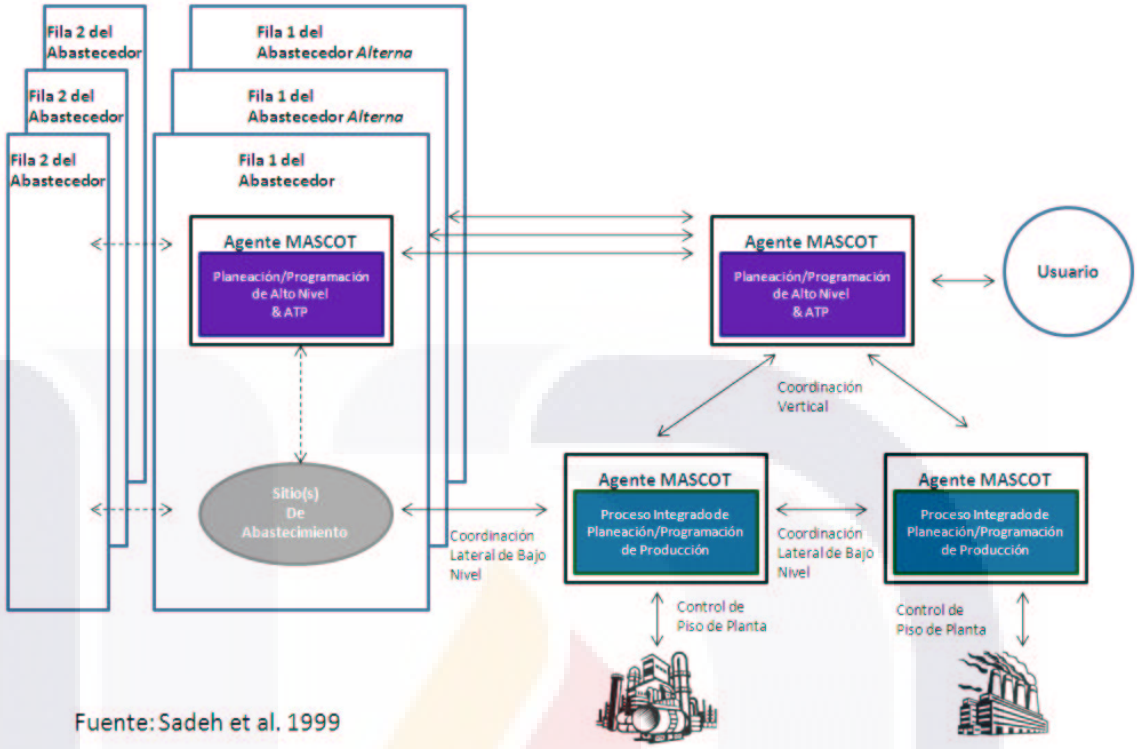


Figura 4.4: Arquitectura MASCOT

Nombre: Arquitectura de Tiempo Real Suave
(Soft Real Time Arquitectura)
Creadores: Bryan Horling, Victor Lesser, Regis Vincent, Thomas Wagner
Dominio: Control en tiempo real.
Organización diseñadora: Department of Computer Science, University of Massachusetts, SRI International, DARPA/IPTO.
Organización receptora: .

4.7.2. Objetivos

- Habilidad para generar rápidamente planes y programas para metas que son adecuadas a los recursos disponibles y las restricciones aplicables, tales como tiempso de inicio más recientes y finales.
- La habilidad para fusionar nuevas metas con las existentes, y multiplexar sus programas solución.
- La habilidad para usar representaciones explícitas de desviaciones inciertas en planes de comportamientos esperados que manejen las variaciones en recursos, uso de patrones y características de acciones inesperadas.

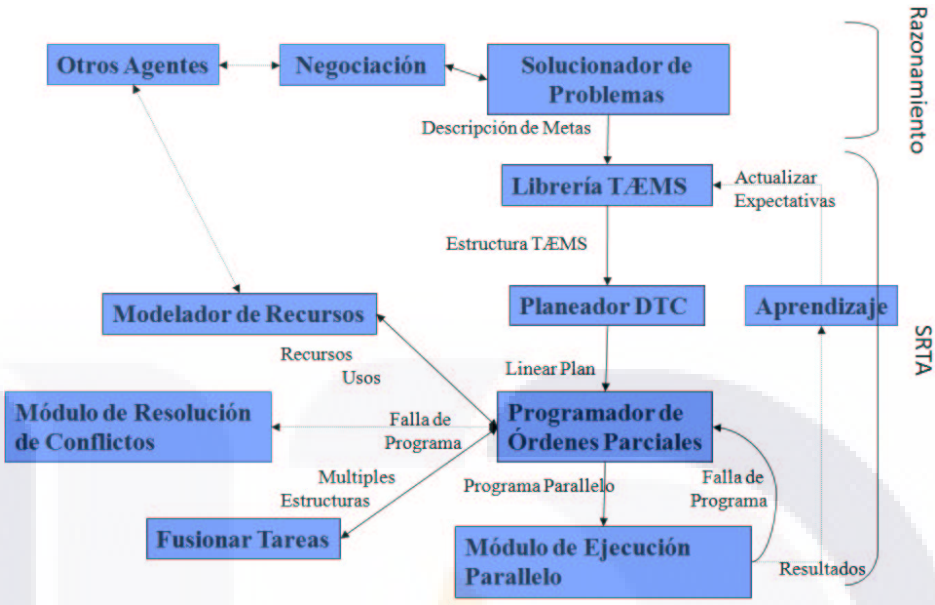


Figura 4.5: Arquitectura SRTA

4.8. Arquitectura CORTES

4.8.1. Información de referencia

Fuente: [Fox & Sycara, 1990]
Nombre:
(C O R T E S)
Creadores: M. S. Fox, Katia P. Sycara
Dominio: Planeación, Programación y Control de la Producción.
Organización diseñadora: Center for Integrated Manufacturing Decision Systems, The Robotics Institute, Carnegie Mellon University.
Organización receptora: The Carnegie Mellon Engineering and Manufacturing Company (CARMEMCO).

4.8.2. Diagramas

El rol de una arquitectura es integrar los métodos de diseño y los algoritmos (soluciones parciales) alrededor de una representación compartida (central) [1735].

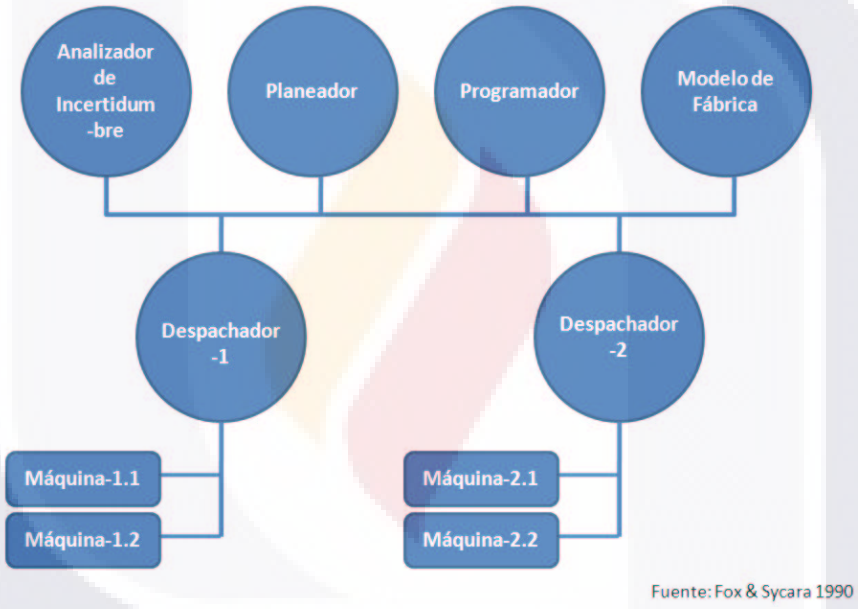


Figura 4.6: La Arquitectura CORTES

Capítulo 5

DISEÑO DE LA ARQUITECTURA PROPUESTA

5.1. Arquitecturas

El diseño de una la arquitectura de un sistema de software puede enfrentarse considerando tres enfoques básicos [Tianfield et al., 2003]:

- Flujo de información
  - Modelo de repositorio
  - Modelo Cliente/Servidor
  - Modelo de Maquina Abstracta
- Control
  - Centralizado
  - Conducido por eventos
- Roles

En este trabajo se utilizó el enfoque orientado a roles por sus características de representación de clasificación dinámica y múltiple y por representar colaboraciones entre objetos.
Un rol además describe las propiedades estructurales y de comportamiento de un objeto (en este caso un agente) en un contexto, además un objeto (agente) puede jugar uno o más roles a lo largo de su existencia y un mismo rol puede ser jugado por diferentes objetos [Andersen, 1997], por lo que los

roles son muy adecuados para la construcción de sistemas con aplicaciones distribuidas.
En el área de producción de una planta de manufactura muchos problemas son esencialmente distribuidos por lo tanto el conocimiento necesario para resolver este tipo de problemas también debe estar distribuido. Emplear un solo elemento para resolver este tipo de problemas requiere múltiples recursos y puede resultar muy riesgoso depositar toda la responsabilidad sobre este único elemento.
Aquí el enfoque distribuido de resolución de problemas resulta muy conveniente y el uso de Sistemas Multiagente permite utilizar a los agentes para resolver cada tipo de problema y lograr un desempeño eficiente y robusto.

La solución de problemas distribuidos implica tres fases:

- La descomposición de problemas en subproblemas
- Resolución de los subproblemas
- La interacción de las subsoluciones

Benchmarks [1601] FJSSP[1285] planeación [1882]

5.2. Caracterización del Área de Producción

Dentro de una empresa de manufactura, el área de producción (*o shop floor*) es el espacio físico en el que se llevan a cabo las operaciones de producción. Está integrado por todos los diferentes elementos que contribuyen en forma directa o indirecta a llevar a cabo la producción de productos terminados.
Considerando la estructura señalada en [Obi, 1999] en la cual un sistema de manufactura está formado por:

1. **Equipos y facilidades**: (máquinas, herramental, equipos y facilidades)
2. **Métodos de producción**: (procedimientos, métodos de producción, control de calidad y control de producción.
3. **Manejo de materiales**: sistemas de transporte de material.
4. **Mano de obra**: personas o trabajadores
5. **Productos**: artículos, bienes producidos

En este trabajo los elementos involucrados en la producción se definen de la siguiente forma:

- **Producto**: Es todo aquel elemento o elementos que van a formar parte de un producto terminado.

- **Operación de producción**: Es toda tarea o grupo de tareas que se realizan sobre un producto para transformarlo, modificarlo o integrarlo a otro producto.
- **Orden de trabajo**: Es todo aquel requerimiento formulado explícitamente que demanda la ejecución de una, o un grupo de operaciones sobre un producto.
- **Fuente**: Es todo aquel elemento que proporciona un producto o grupo de productos a ser utilizados en el área de producción para obtener un producto terminado
- **Procesador**: Es todo aquel dispositivo, maquina, arreglo o ser humano que transforme, modifique, o integre un producto o grupo de productos durante su paso por el área de producción.
- **Cola**: Es todo aquel espacio físico o contenedor que sirve para almacenar productos durante su paso por el área de producción.
- **Transportador**: Es todo aquel dispositivo, maquina, arreglo o ser humano que modifique la ubicación física de un producto dentro del área de producción en su ruta hacia la salida.
- **Salida**: Es el punto del área de producción al cual llegan los productos terminados, en este punto ya no pueden regresar al sistema.
- **Servicios**: Son todos aquellos elementos, dispositivos, maquinas, arreglos o seres humanos que contribuyen en forma indirecta a que se realice la producción.

5.2.1. Ecosistemas Artificiales Industriales

Los ecosistemas artificiales están inspirados en el comportamiento social de seres no-humanos, principalmente insectos [Parunak, 1997], existe una gran similitud entre éstos y los sistemas de producción.
Las sociedades de insectos, a nivel de colonia enfrentan retos muy parecidos a los de una fábrica, por ejemplo tienen que hacer frente a la competencia por los recursos y tienen que enfrentarse a cambios en ambientes imprevisibles (plasticidad) [Robinson, 1992].
Una colonia de insectos funciona como una unidad integrada que no solo posee la habilidad de procesar grandes cantidades de información en forma distribuida, toma de decisiones sobre cómo asignar individuos a varias tareas, coordinar las actividades de decenas o miles de trabajadores o enfrentar enormes proyectos de construcción, sino también exhibir flexibilidad y robustez en respuesta a los cambios externos y las perturbaciones internas [Wilson & Holldobler 1988].

A continuación un breve análsis de comportamiento de insectos sociales que muestran una cierta correspondencia con los realizados en una fábrica [Parunak 1997]:

- *Hormigas: Planeación de rutas*
  - **Comportamiento del sistema** - las hormigas construyen redes de rutas que conectan sus nidos con las fuentes de alimento disponibles que minimizan la distancia recorrida, y por lo tanto, la energía requerida para llevar el alimento al nido, de ello emerge una estructura óptima global a partir de acciones simples de los individuos (hormigas).
  - **Responsabilidades** - cada hormiga que sale en busca de alimento tiene el mismo comportamiento básico definido por las siguientes reglas:
    - *Evitar obstáculos*
    - *Vagar aleatoriamente*
    - *Si encuentra alimento depositar feromona por el camino*
    - *Si no tiene alimento y lo encuentra debe tomarlo*
    - *Si está en el nido y transporta alimento, depositarlo*
  - **Integración** - el movimiento browniano conduce eventualmente a la hormiga a cualquier punto en el plano, vagando la hormiga encontrará alimento, si lo hay (aún sin guía) y si transporta alimento eventualmente encontrará el nido, si una hormiga se mueve en una direcciíon donde no exista alimento, será presa de los depredadores o morirá de hambre.
- *Hormigas: Cuidado de las crías*
  - **Comportamiento del sistema** - un hormiguero contiene diversas clases de cosas (larvas, huevos, capullos, alimento) las cuales están clasificadas y tienen una ubicación específica.
  - **Responsabilidades** - el algortimo individual de una hormiga que maneja los comportamientos clasificadores es muy similar al problema de planeación de rutas.
  - **Integración** - como en la planeación de rutas el movimiento browniano hace vagar a la hormiga para examinar todos los objetos en el nido, lo que hace que las hormigas coloquen objetos similares en los mismos lugares.
- *Termitas: Construcción de nidos*
  - **Comportamiento del sistema** - las termitas tropicales construyen montículos que pueden exceder los cinco metros de altura

y varias toneladas de masa. Estas estructuras de varios pisos almacenan alimentos, resguardan a las crías y protegen a la población del fuego y de los depredadores, la existencia de algunas de ellas sobrepasa los 350 años y a pesar de su complejidad durabilidad y efectividad ninguna termita toma el papel de ingeniero en jefe, planeando su estructura y administrando su construcción.

- **Responsabilidades** - las termitas, al igual que las hormigas, depositan rastros de feromona y vagan en el espacio en busca de alimento.
- **Integración** - un algortimo de probabilidad define la distribución inicial de los dépositos para formar el nido, estos atraen a las termitas que vagan cerca y aumentan la probabilidad de su refuerzo, como el rastro decae en ciertos plazo los dépositos en el centro son los más fuertes y las pilas tienen a crecer hacia arriba formando columnas.

- *Avispas: Asignación de tareas*
  - **Comportamiento del sistema** - las avispas se clasifican dentro del nido en tres tipos, un solo jefe, un grupo de forrajeras y un grupo de enfermeras que se encargan del cuidado de las crías, estos roles son jugados por avispas genéticamente idénticas, la proporción relativa entre forrajeras y de enfermeras varía con la abundancia de alimento y el número de crías, y nadie se encarga de establecer como debe ser dicha proporción.
  - **Responsabilidades** - cada avispa mantiene dos parámetros, uno de Fuerza, que determina su movilidad y otro de Umbral de Forrajeo que determina cómo buscar alimento, las crías manejan un tercer parámetro, la demanda, que estimula a los forrajeros, los comportamientos se derivan de la interacción de éstos parámetros.
  - **Integración** - la confrotación entre avispas modifica la asignación de tareas mediante una forma de comunicación primitiva, el alimento reduce la demanda de las crías mientras que el estímulo reduce los umbrales y acciona así el forrajeo.

Características:

- **Pequeños en tamaño** - Los sistemas naturales tales como las colonias de insectos y las economías de mercado están formados por muchos agentes, en comparación con el sistema completo. Tales agentes son faciles de construir y entender y el impacto de la falla de cualquiera de los agentes es mínimo. Muchos agentes favorecen un espacio rico en comportamientos (comportamiento emergente) el espacio de estados

es exponencial al numero de agentes. La cantidad favorece a la supervivencia. La especialización de entidades pequeñas es mejor que la generalización de entidades mayores usando las técnicas de agregación apropiadas. Es preferible construir un agente para cada mecanismo de una célula de producción que un solo agente para toda la célula.

- **Descentralización** - Los sistemas naturales no reflejan el tipo de centralización que frecuentemente aparece en los sistemas artificiales. La centralización puede deberse a veces a que el diseñador confunde la clase con los agentes individuales se podría estar tentado a representar un grupo de cabinas de pintura con el agente pintura debido a que todas hacen los mismo, ciertamente es una clase pero las cabinas son una instanciación de esa clase.
- **Diversidad y Generalización** - Las comunidades naturales de agentes balancean su diversidad (lo que les permite monitorear un ambiente mucho mayor que el que haría un solo agente) con mecanismos generalizados. Los mecanismos de herencia de clases de las plataformas orientadas a objetos sobre las que se construyen los agentes son un soporte excelente para una generalizacion comparable a través de los agentes que construyen, pero la experiencia muestra que la parte difícil es identificar generalizaciones apropiadas en primer lugar.

5.3. Arquitectura Propuesta

Considerando los requerimientos no funcionales que se señalaron en la Sección 1.4.1 la propuesta de arquitectura se diseñó buscando que reúna las siguientes características:

- Flexibilidad
- Adaptabilidad
- Optimización
- Re-utilidad

Flexibilidad

La flexibilidad se consideró desde varios enfoques:

En cuanto al SMA como un sistema de software la flexibilidad se va a entender como su capacidad para soportar cambios, actualziaciones y mejoras que le permitan ir evolucionando al ritmo de la tecnología.

En cuanto

Adaptabilidad

Optimización

Re-utilidad

5.3.1. Modelo del Agente

Se propone una estructura genérica básica de capas para modelar a los agentes que sirva como plantilla para el desarrollo de cada tipo de agente y a su vez que determine la generación de nuevos agentes, o su remoción, basada en los requerimientos del sistema.
En cada agente existe una capa intermedia que esta diseñada de acuerdo con las funciones y tipos de tareas de las que se va a encargar cada tipo de agente.
El agente esta consituido por capas las cuales se generan en base a los requerimientos de las tareas y funciones que realice el agente.
La primera capa denominada de Interface esta formada, a su vez por tres sub-capas que son:

- **Interface Física** - se encarga de recibir/enviar señales discretas de los dispositivos de control.
- **Interface Lógica** - se encarga de recibir/enviar paquetes de señales que requieran de una decodificación.
- **Interface Inteligente** - se encarga de recibir/enviar información que deba ser interpretada por el sistema.

La siguiente capa se denomina de Función, y es la que se encarga de realizar las funciones específicas del sistema de agentes (planeación, programación, control, apoyo y orden de trabajo).
La capa superior es la denominada de Agente, que se va a encargar de administrar el conocimiento para optimizar la tarea de toma de decisiones.

Capa de Interface
Esta capa se encarga de actuar como el medio de comunicación entre el agente y el ambiente, dicha comunicación se va a

Interface Física
Esta formada por aquellos dispositivos que manejen datos en forma de señales discretas (interruptores, relevadores, sensores, ..).
Interface Lógica
Esta formada por aquellos dispositivos de control que manejen información sobre procesos (controladores de lazo cerrado, controladores lógicos programables,....).

Interface Inteligente
Esta formada por aquellos sistemas que manejen información que deba ser intérpretada (MRP, CASE, SCADA,...).

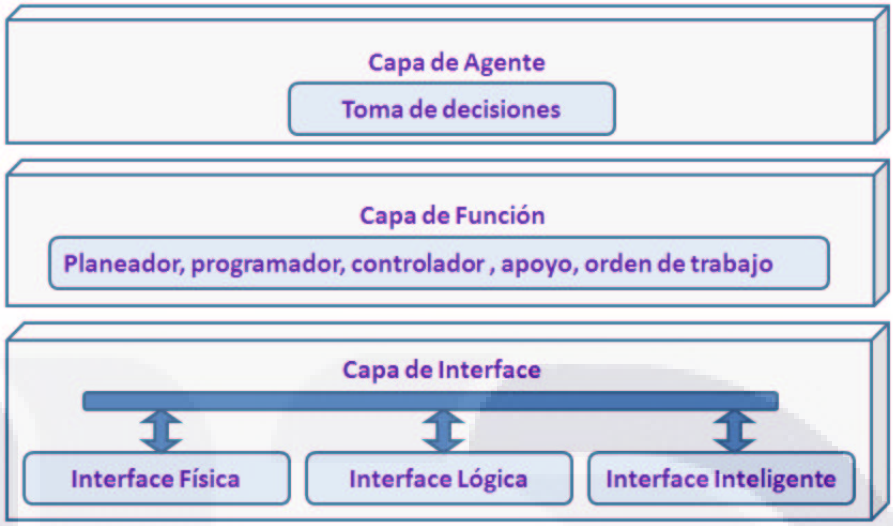


Figura 5.1: Modelo del Agente

5.3.2. Modelo del ciclo de vida del agente

Cómo identificar a los Agentes?

Cualquier parte del sistema que sea capaz de intercambiar información con una capa de interface es candidato a ser controlado por un agente.
En el caso de dispositivos que manejen entradas de datos (señales discretas) se debe considerar un número determinado de ellos (dependiendo de la aplicación pero de preferencia que sea mayor de 3) que justifique el esfuerzo invertido y disminuya la probabilidad de un comportamiento caótico del sistema.
La condición anterior, de que exista un elemento que se pueda comunicar con una capa de interface, es necesaria pero no suficiente,
En el caso de dispositivos de control o sistemas la relación puede ser de 1 a 1 con respecto al agente.
Incluso puede considerarse una combinación de los elementos anteriores para definir un control por agente.

5.4. Subsistema de Planeación

Rol del Agente de Planeación

El subsistema de planeación está formado por agentes deliberativos que se encargan de planeación [1882]
La planeación en un entorno de fabricación discreta involucra:

- La descomposición de una orden de trabajo en una secuencia de operaciones de producción

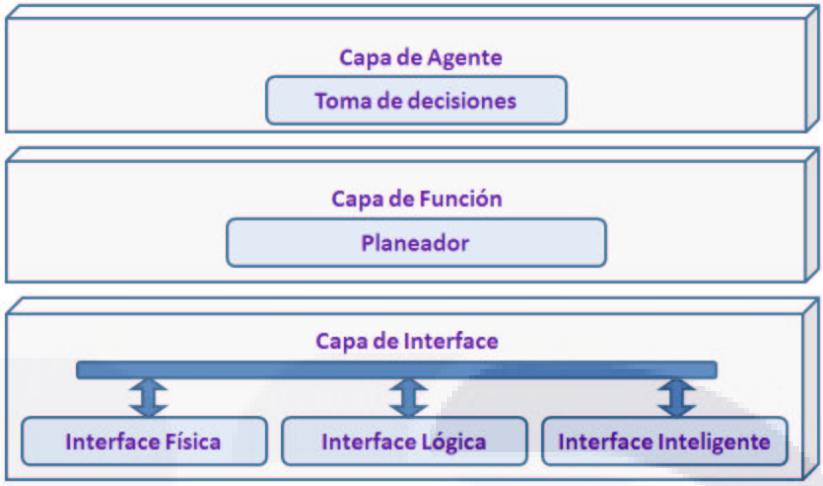


Figura 5.2: Modelo del Agente Planeador

- La asignación nominal de las operaciones de producción a los diferentes tipos de recursos

Los pasos a seguir para elaborar un plan de producción son:

1. Cada agente planeador realiza una descomposición de la especificación del producto en partes itegrantes o sub-ensambles
2. Por cada producto, el agente planeador identifica las operaciones de producción necesarias.
3. Se seleccionan los tipos de recursos necesarios para proveer las operaciones de producción mediante una interaccion entre los agentes planeadores y los agentes orden de trabajo.
4. Se involucra un proceso de comunicación entre los agentes involucrados para determinar el conjunto adecuado de operaciones de producción.
5. Se concreta la secuencia completa de operaciones de producción (plan de producto) y el agente planeador lo almacena.

Formalmente un algoritmo de planeación tiene tres entradas:

- La descripción del mundo en un lenguaje formal
- La descripción de la meta del agente en un lenguaje formal
- La descripción de las posibles acciones que puedan ser realizadas

La salida del agente planeador es una secuencia de acciones la cual, cuando se ejecuta en el mundo, satisfaciendo las condiciones iniciales, alcanzará la meta.
El enfoque que se dió al modelo del agente planeador integra a su comportamiento la planeación multi-objetivo.

5.5. Subsistema de Programación

Rol del Agente de Programación
El subsistema de programación está formado por agentes reactivos que se encargan asignar las tareas a los recursos.

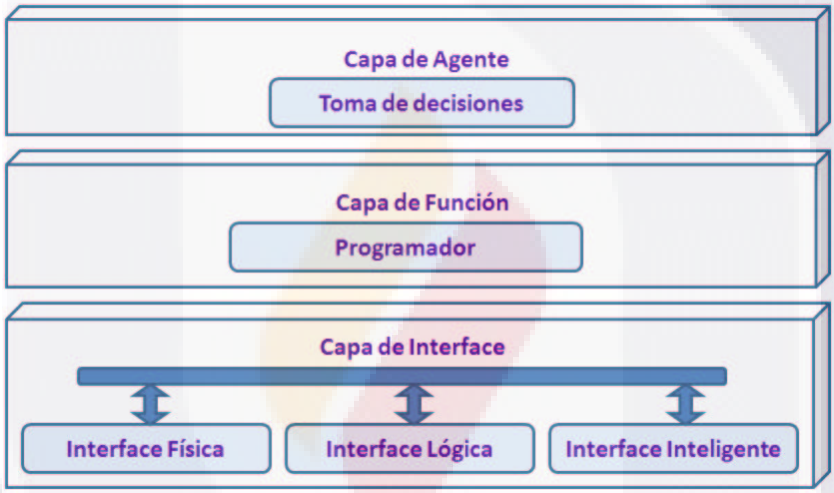


Figura 5.3: Modelo del Agente Programador

La programación en un entorno de fabricación discreta involucra:

- La asignación de las operaciones de producción a recursos específicos
- La especificación de tiempos (inicio, duración y terminación) para estas operaciones

De esta forma a cada agente se le asociará un módulo de toma de decisión local y capacidad de cómputo, un mecanismo o protocolo de intercambio, que se encargará de la comunicación para realizar una estrategia cooperativa y un grado de coordinación específico.

Algoritmo de Programación
El algoritmo empleado para el agente programador surge del modelo de auto-organización que tiene lugar en una colonia de avispas [Theraulaz et al., 1991].

El modelo de comportamiento de la avispa describe la naturaleza de los dos tipos de interacciones que se encontráron para controlar el comportamiento individual en una colonia de avispas (*Polistes dominulus*).
El modelo de asignación de tareas auto-organizada de la colonia usa lo que [Theraulaz et al., 1991] llama *umbrales de respuesta*.
Una avispa individual tiene un umbral de respuesta para cada zona del avispero. Basada en el umbral de avispa para una zona dada y la cantidad de *estímulo* de la prole localizada en esa zona, una avispa puede o no ocuparse de la tarea de alimentación allí.
Un umbral menor de una zona dada equivale a una mayor probabilidad de ocuparse en la actividad dado un estímulo, en [Bonabeau et al., 1998] discuten sobre un modelo en el cual estos umbrales permanecen fijos en el tiempo. Después ellos consideran que un umbral para una tarea dada disminuye durante períodos de tiempo cuando esa tarea es realizada o se incrementa [Theraulaz et al., 1994].
El modelo de comportamiento de la avispa de [Theraulaz et al., 1991] también describe la naturaleza de la interacción avispa-avispa que tiene lugar dentro del avispero. Cuando dos individuos de una colonia se encuentran, hay alguna posibilidad de que interactúen en un *torneo de dominancia*.
Si esta interacción tiene lugar, la avispa con mayor rango social tiene más probabilidades de dominar la interacción. A través de estas interacciones, las avispas dentro de la colonia se auto-organizan en una jerarquía de dominancia. En [Theraulaz et al., 1995] se discuten un número de formas de modelar la probabilidad de interacción durante un encuentro en el cual el rango de algunas interacciones se basa en ciertas tendencias de los individuos.
En [Cicirello & Smith, 2004], se retoma el modelo de la avispa y se usa para controalr la programación en un sistema simulado de una planta automotriz, específicamente en un área de pintado de camiones donde se tiene a la entrada una línea de ensamble de camiones que entrega camiones para ser pintados a una serie de cabinas de pintura, a cada una de ellas se asocia un *Ruteador de Avispa* que determina si la cabina licita o no por cada camión, se tiene varios colores y no se conoce el color a emplear hasta que el camión llega al área de pintura, para el algoritmo del *Ruteador de Avispa* considera los siguientes elementos:

Un umbral de respuesta:

$$\Theta_\omega = \theta_{\omega 0}, ..., \theta_{\omega J} \tag{5.1}$$

Donde:
$\theta_{\omega J}$ es es el umbral de respuesta de la avispa ω a trabajos de tipo J.

Los trabajos en el sistema que no han sido asignados a una maquina, diseminan entre todos los *Ruteadores Avispa* un *estímulo* el cual es proporcional al lapso de tiempo que el trabajo espera para asignación a una cabina. El

estímulo es *etiquetado* de acuerdo al tipo de trabajo. Por lo que el trabajo que más tiempo permanece sin asignar, es el que emite el estímulo más fuerte.
Un *Ruteador de Avispa* ω puede licitar por un trabajo que está emitiendo un estímulo S_j con probabilidad:

$$P(licitar|\theta_{\omega j}, S_j) = \frac{S_j^2}{S_j^2 + \theta_{\omega j}^2} \tag{5.2}$$

De otra forma no lo solicitará. Esta es la regla definida para la asignación de tareas en el modelo conductual de avispas como el descrito en [53].
En esta forma, las avispas tienden a seleccionar trabajos del tipo en los que el umbral de respuesta es menor. Pero pueden seleccionar trabajos de otros tipos si un mayor estimulo es emitido. Este mecanismo para decidir si licita o no un trabajo se ilustra en la la Figura x.

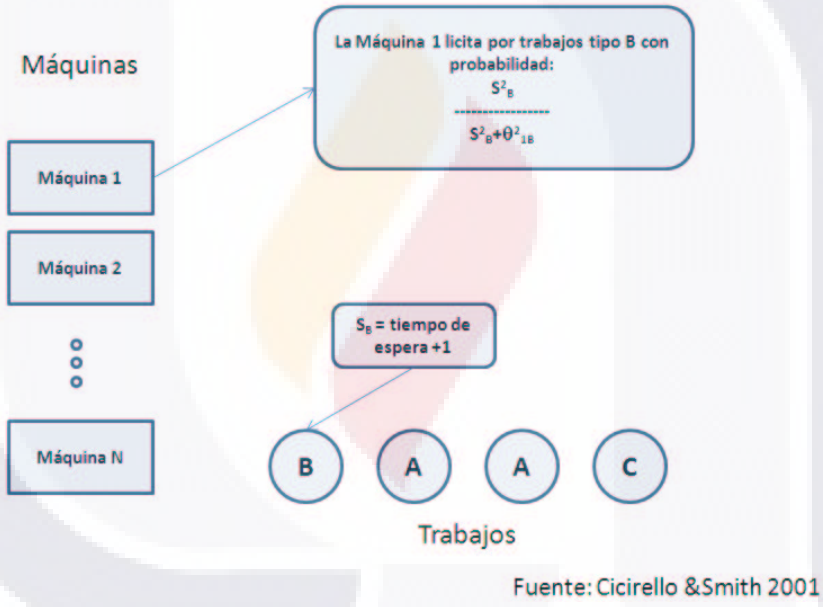


Figura 5.4: Modelo del Ruteador de Avispa

Los valores de umbral $\theta_{\omega j}$ pueden variar en un rango $[\theta_{min}, \theta_{max}]$ y cada *Ruteador de Avispa* sabe, en todo momento, qué es lo que está haciendo su procesador, incluyendo: el estatus de la cola, y sí, o no el procesador está realizando una actualización, el tipo de trabajo que está siendo procesado y si el procesador está ocioso.
Este conocimiento se usa para ajustar los umbrales de respuesta de los varios tipos de trabajos comunes en una fábrica.
Esta actualización del umbral de respuesta ocurre en cada lapso de tiempo. Sí la maquina está procesando un tipo j de trabajo actualmente o si está en el proceso de actualizar para procesar un tipo de trabajo j, entonces $\theta_{\omega j}$ se actualizará de acuerdo a:

$$\theta_{\omega j} = \theta_{\omega j} - \delta_1 \tag{5.3}$$

Sí la maquina está ya sea procesando o actualizándose para procesar un tipo de trabajo diferente de j entonces, $\theta_{\omega j}$ se actualiza de acuerdo a:

$$\theta_{\omega j} = \theta_{\omega j} - \delta_2 \tag{5.4}$$

Sí una maquina esta ociosa actualmente y tiene una cola vacía, entonces para todos los trabajos de tipo j que la maquina pueda procesar el *Ruteador de Avispa* ajusta el umbral de respuesta $\theta_{\omega j}$ de acuerdo a:

$$\theta_{\omega j} = \theta_{\omega j} - \delta_3^t \tag{5.5}$$

Donde:
t es el lapso de tiempo que la maquina ha estado ociosa y es un exponente.

En el modelo de [Cicirello & Smith, 2004], las constantes del sistema $\delta 1$, $\delta 2$ y $\delta 3$ son positivas, por lo que, los umbrales de respuesta para el tipo de trabajo que está siendo procesado actualmente son reforzados para dar mayor valor al *Ruteador de Avispa* para licitar por trabajos del mismo tipo; mientras que los umbrales de respuesta de otros tipos que no están siendo trabajados actualmente son adaptados para disminuir el valor del *Ruteador de Avispa* para licitar ese mismo tipo de trabajos.
Esta especialización del *Ruteador de Avispa* (es decir, del procesador) ayuda a minimizar los tiempos de actualización. Las primeras dos formas en las cuales el umbral de respuesta se actualiza (ecuaciones 5.3 y 5.4) son análogas a las del modelo real de avispas [Theraulaz et al., 1994].
La tercera (ecuación 5.5) se incluye para valorar una avispa asociada con una maquina ociosa para tomar cualquier trabajo que pueda tener para no permanecer ociosa. Esta última regla de actualización reconoce que aunque la especialización puede reducir el tiempo de actualización, la sobre especialización de un tipo de trabajo con baja demanda puede resultar en una baja respuesta del sistema.
Considerando lo anterior se empleó el modelo de la avispa y el umbral de respuesta para que el agente programador formule una política adaptativa para decidir si se licita o no por un trabajo.
Para mejorar este componente (algoritmo *Ruteador de Avispa*) se agregó un enfoque multiobjetivo en el cual se busca no sólo disminuir el número de actualizaciones (setups) de los procesos, lo cual es deseable en la mayoría de las empresas manufactureras, sino también considerar otras medidas de rendimiento comunes en la industria tales como:

- Duración total de un programa (makespan)
- Tardanza (tardiness)

- Tardanza promedio (average tardiness)

Para lo cual se emplea la optimización de frente de Pareto mediante una técnica de búsqueda local de las soluciones presentes a la salida del *Ruteador de Avispa* que determina si existen y cuales son las soluciones, dominadas y no-dominadas,

Plantilla de identificación del agente programador

5.6. Subsistema de Control

El subsistema de control está formado por agentes mixtos que se encargan del inicio, control, monitorización y terminación de tareas con tiempos y parámetros de producción en tiempo real.

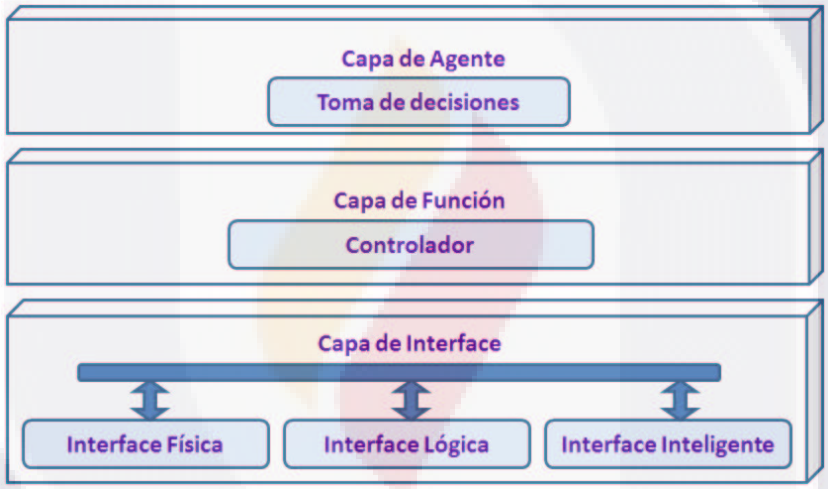


Figura 5.5: Modelo del Agente Controlador

En este SMA el agente controlador se encarga de:

1. Asegurar que se establecen y mantienen las operaciones de producción
2. Que se lleven a cabo las tareas compatibles con los requisitos de producción incluso ante fallos

Rol del Agente de Control

Para modelar el agente de control se va a partir de un lenguaje de modelado muy utilizado en el área de control, las **Redes de Petri**.

Las Redes de Petri surgieron del trabajo doctoral *Kommunikation Mit Automaten* (Comunicación con autómatas) de Karl Adam Petri en 1962, y son una herramienta para el estudio de sistemas, pueden ser analizadas de manera formal y obtener información del comportamiento dinámico del sistema modelado.

La estructura de una red de Petri esta integrada por:

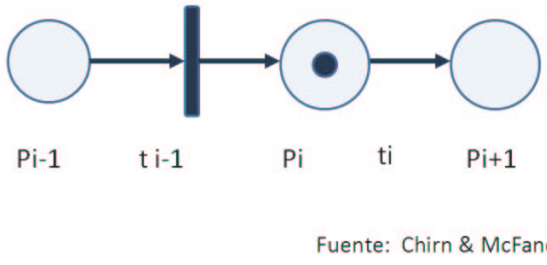
- Un conjunto de nodos
- Un conjunto de transiciones
- Una función de entrada
- Una función de salida

Las funciones de entrada y salida relacionan los nodos y las transiciones. La función de entrada es un mapeo de una transición t_j a una colección de nodos conocidos como los nodos de entrada de una transición.
Formalmente, una red de Petri es una 4-tupla:

$$RP = \{P, T, \alpha, \beta\} \tag{5.6}$$

Donde:
P es un conjunto finito y no vacío de *nodos*
T es un conjunto finito y no vacío de *transiciones*
$P \cap T = \oslash$
$\alpha : T \times P \rightarrow$ es la *función de entrada*
$\beta : P \times T \rightarrow$ es la *función de salida*

Gráficamente [Chirn & McFarlane, 2000], una red de Petri es un grafo bipartito dirigido formado por los siguientes elementos: *nodos* (o lugares) los cuales denotan el estado del sistema y están representados por círculos, *transiciones*, las cuales denotan los eventos o condiciones que pueden ocurrir en el sistema y están representadas por barras, *arcos* los cuales denotan la conexión entre nodos y transiciones y son representados por flechas, y *señales* (tokens) las cuales denotan el nodo actualmente activo y están representadas por puntos negros dentro de los nodos.
Tan pronto como un nodo obtiene una señal, el estado de ese nodo se vuelve activo, y es llamado *nodo marcado*, al mismo tiempo la siguiente transición de este nodo puede ser disparada. Después del disparo, el nodo regresa a su estado inactivo (ocioso) y la señal es transferida al siguiente nodo.



Fuente: Chirn & McFandale, 2000

Figura 5.6: Un modelo simple de una red de Petri

Reglas de disparo de las redes de Petri

La ejecución en una red de Petri es controlada por el número y distribución de las señales (*tokens*) que tiene. Las señales presentes en los nodos controlan la ejecución de las transiciones de la red. Una red de Petri se activa disparando transiciones. Una transición es disparada removiendo señales de los nodos de entrada y creando señales de salida.
De aquí se obtiene la primera condición de disparo en una transición: todos los nodos de entrada de la transición, deben tener al menos el mismo número de señales, que de arcos que van hacia la transición, para que ésta sea disparada, cuando la transición cumpla esta condición se dice que es una transición habilitada.
Las transiciones pueden seguir disparándose indefinidamente hasta llegar a un estado deseado o hasta que ninguna pueda ser disparada.

Redes de Petri Temporizadas

En este tipo de redes se considera al tiempo en el modelo lo cual es una consideración importante ya que en los STR es indispensable para la sincronización de los procesos. El modelo más simple es el que asigna duración a:

1. Los nodos, en el sentido de que una condición es verdadera solo para una cierta cantidad de tiempo.
2. La transición, en el sentido de que un evento toma una cierta cantidad de tiempo en ocurrir.
3. Cuando la duración de los eventos no son fijos, o no pueden ser expresados con valores nominales, simplemente se estiman límites dentro de los cuales el evento puede ocurrir.

Modelado de sistemas en tiempo real con redes de Petri

Las redes de Petri proporcionan los elementos necesarios con restricciones de tiempo para la representación de los STR, pero es preciso representar de manera no ambigua los siguientes aspectos:

- Eventos (tanto internos como externos) a los que responde el sistema,
  - Los patrones de tiempo que controla a estos eventos, por su periodicidad o aperiodicidad
  - Las acciones activadas por sus características temporales
- Las interacciones entre las distintas acciones, que consisten de sincronizaciones, comunicaciones o relaciones de precedencia

En el modelo de redes de Petri temporizado, se requiere considerar el tipo de transiciones que el STR contiene:

- CODEt. Cada una de estas transiciones junto con sus lugares de entrada representa una actividad desarrollada por el sistema (código ejecución). Este código comenzara a ejecutarse en el momento en que la transición sea sensibilizada, es decir, todos sus lugares de entrada están marcados.
- TIMEt. Son transiciones asociadas a alguna actividad temporal, como un time out o la activación periódica de un proceso.
- SYCOt. Son el resto de transiciones, y no tienen significado temporal explicito asociado. Su disparo será inmediato, lo que supone un intervalo temporal implícito, usadas para modelar sincronizaciones y tareas de control.

En los STR la parte operativa del sistema recae sobre las transiciones CODE, mientras que el de la parte de control y de supervisión temporal depende de transiciones SYCO, TIME.
Estos tres tipos de transiciones son la base de lo que se llama unidades de ejecución, o componentes básicos que representan acciones elementales en el STR. Como consecuencia de la anterior clasificación, se distinguen tres unidades de ejecución:

- CODEe: Formada por una transición CODEt y sus lugares de entrada, que modelará la ejecución de un cierto código en el sistema: esta unidad comenzará a ejecutarse cuando se sensibilice su transición, estará en ejecución mientras sus lugares de entrada permanezcan marcados, y terminará en un instante determinado por su intervalo temporal, o si es ocupada se le dice que es abortada,
- TIMEe: Integrada por una transición TIMEt y sus lugares de entrada, que corresponderá a la posible ocurrencia de un evento temporal; la unidad se ejecutará cuando transcurra la cantidad de tiempo especificada por su intervalo temporal, siempre que la transición permanezca ocupada de forma continua durante esa cantidad de tiempo,
- SYCOe: Formada por una transición SYCOt que representa acciones de control o sincronización; se ejecutará inmediatamente en el momento en que se detecte su ocupación.

El resto de elementos, tales como lugares de la red serán utilizados para modelar recursos, datos, condiciones, etc., de acuerdo al modelo del sistema descrito por la red de Petri. Cada lugar puede tener como máximo, una transición CODEt de salida, si bien puede contar simultáneamente con varias TIMEe y varias SYCO. De este modo se evita que puedan existir conflictos entre transiciones CODEt (lo que modela que un sistema está ejecutando simultáneamente dos códigos distintos).

Diagramas de Protocolo de Interacción de Agentes
Los diagramas de protocolos de interacción de agentes describen el comportamiento externo de los agentes en un SMA. Cuando se convierte de un tipo de diagrama a otro se habla de realizar una *transformación*, una *traducción* o un *mapeo*, todos estos términos son adecuados para describir el procedimiento de convertir un diagrama de protocolo de interacción de agente a una red de Petri y viceversa.
Desde el punto de vista matemático, un mapeo se enfoca en el hecho de que un conjunto de elementos de un diagrama son trasladados a un conjunto de elementos de otro diagrama, los elementos de una red de Petri representan el dominio y los elementos del diagrama de protocolo de interacción representan el co-dominio del mapa.

5.7. Ambiente de la Arquitectura

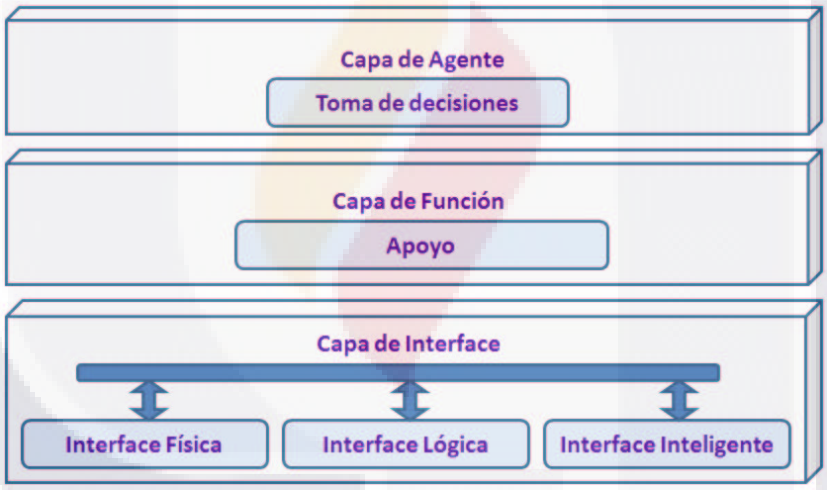


Figura 5.7: Modelo del Agente de Apoyo

Rol del Agente de Apoyo
Rol del Agente de Orden de Trabajo

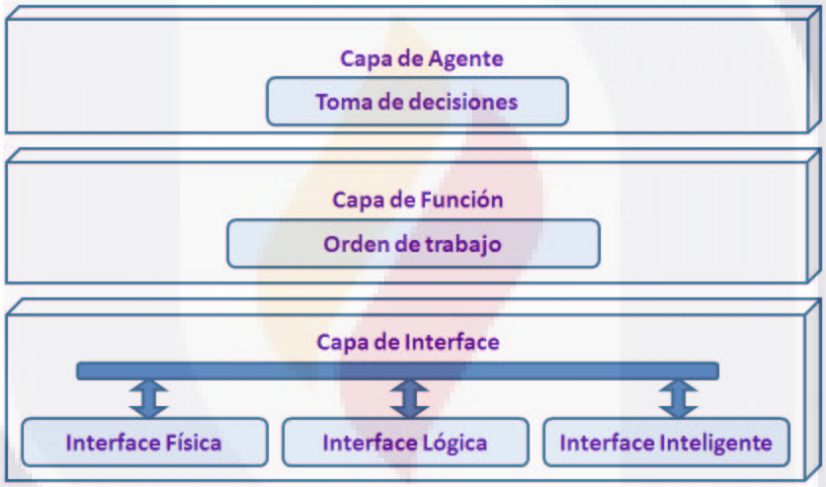


Figura 5.8: Modelo del Agente de Orden de Trabajo

Capítulo 6

CASO DE APLICACIÓN

6.1. Caso

El diseño y desarrollo de este caso de aplicación forma parte del presente trabajo de investigación de acuerdo al objetivo especificado en la sección .
Básicamente, este caso busca validar el algoritmo seleccionado para resolver el problema de la programación de la producción denominado *Ruteador Avispa* [Cicirello & Smith, 2004] el cual está basado en el modelo del comportamiento de una colonia de avispas real, realizado por [Theralauz et al. 1991], en donde las interacciones entre avispas individuales y el ambiente local toman forma de mecanismos de estímulo-respuesta para controlar la asignación de tareas distribuida, además las interacciones entre pares de individuos resultan en una auto-organización de jerarquía de dominancia. Combinando ambos aspectos del comportamiento se logra una efectiva y descentralizada base para coordinar la asignación de trabajos a maquinas en una fabrica.

Tipo de organización a la que se aplicó el sistema diseñado
El tipo de organización que se simuló mediante este sistema es una *empresa manufacturera* que se dedica a la fabricación de camiones de carga, específicamente el sistema se encarga de recibir de la línea de producción los camiones terminados y pintarlos del color especificado en la orden de trabajo, los colores no se conocen hasta el momento en que llegan al área de pintura donde se ubican las cabinas que se encargan de realizar esta tarea.
El propósito de este caso se desarrolló en dos fases, en la primera se reprodujo el sistema de simulación de planta de pintura presentado en el trabajo de investigación de [Cicirello y Smith 2004], denominado *Wasp-like Agents for Distributed Factory Coordination* en el que agentes denominados Ruteadores Avispa, asociados a las cabinas de pintura se encargan de licitar o no por los

camiones que van saliendo de la línea de producción, mediante una combinación de mecanismo de mercado y el modelo de la avispa.

La segunda fase:

Identificación de Requerimientos

¿Cuál es el problema?

Estructura del problema de programación de tareas de una planta de pintura de camiones

Función: asignar dinámicamente camiones a cabinas de pintura

Restricciones:

- El ritmo de entrada de camiones a la línea de pintura es de 1 por minuto
- Hay siete cabinas de pintura
- La cola puede tener un máximo de 3 camiones
- Toma 3 minutos para pintar un camión
- Hay 14 colores diferentes de pintura
- Cada camión requiere un color diferente de pintura y arriban sin ningún orden
- El 50 % de los camiones requieren un solo color
- El otro 50 % requieren colores especificados en forma aleatoria de los otros 13 colores (es decir, 1/26 para cada color)
- Una cabina de pintura puede utilizar un solo color a la vez y hay un costo en tiempo por reconfigurarla para otro color
- El costo en tiempo por reconfiguración es de 1 minuto

Entorno de las cabinas

Cada cabina de pintura tiene asociado a un agente que debe cumplir con las siguientes reglas simples:

1. Intente tomar un camión del mismo color que su color actual.
2. Tome particularmente los trabajos importantes.
3. Tome cualquier trabajo para permanecer ocupado.
4. No tome otro trabajo si la cabina de pintura está fuera de servicio o su cola está llena.

Estas reglas deben ser aplicadas por medio de un sistema de subastas de un mecanismo de mercado (cada carro debe ser pintado). La oferta de las cabinas se basa en su capacidad de hacer un trabajo eficiente a bajo costo y con un retardo mínimo. Una cabina de pintura que esté fuera de servicio o cuya cola esté llena no participa en hacer una oferta. Los parámetros óptimos del sistema para hacer ofertas son determinados con técnicas de cómputo evolutivo.
Un ofrecimiento (licitación) esta basado en tres factores:

1. La longitud de la línea delante de la cabina
2. La prioridad del camión, y
3. La necesidad de eliminar burbujas de la pintura.

Dominio del tiempo

- Naturaleza del dominio - Discreto
- Reglas de propagación - Periódicas
- Dependencia entre los dominios de tiempo - Sincronización
- Tipo de restricción temporal - Temporal

Modelado del sistema

Utilizando AUML y considerando las características anteriores se procede a elaborar el diagrama de clases de agentes [Picard, 2003], el cual emplea el estereotipo de agente (en este caso es un agente reactivo, este es un agente que responde a los cambios en su ambiente).
Además en el área de atributos se incluyeron los estereotipos para especificar:

1. **Percepción** - Expresa un medio por el que el agente recibe información del ambiente físico o social (otros agentes) sus atributos representan datos que provienen del ambiente (visibilidad privada).
2. **Interacción** - Establece herramientas que habilitan al agente a comunicarse con otros agentes o con su ambiente (visibilidad pública).
3. **Habilidades** - Conocimiento especifico del agente para realizar su función (visibilidad privada)
4. **Acción** - Acciones realizadas sobre el ambiente (visibilidad privada)

La línea de ensamble y las cabinas de pintura son objetos que forman parte del ambiente del sistema multiagente las relaciones entre estos elementos son de asociación, la cual no es una asociación fuerte ya que sus tiempos de vida no dependen entre sí.
Casos de uso

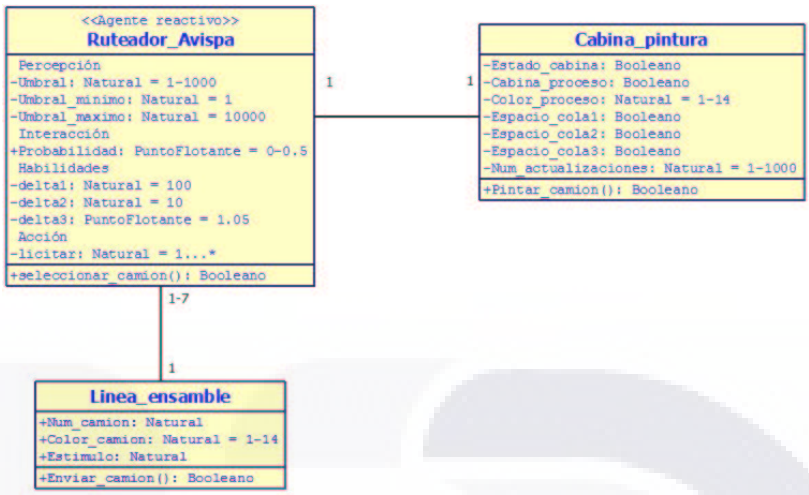


Figura 6.1: Diagrama de clases del Ruteador-Avispa

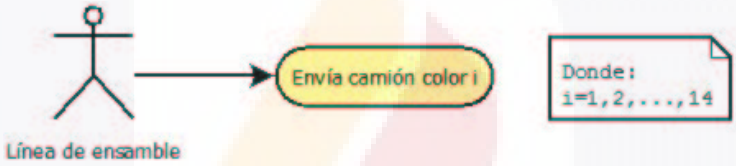


Figura 6.2: Caso de uso del Ruteador-Avispa

Una primera aproximación relaciona la línea de ensamble con las cabinas por medio del Ruteador de Avispa:
A continuación se muestra un ejemplo de Diagramas de interacción entre la línea de ensamble, el Rutedor Avispa y la cabina de pintura:

Figura 6.3: Relación Línea de ensamble-Ruteador-Cabina

Figura 6.4: Diagrama de estados de la Cabina de pintura

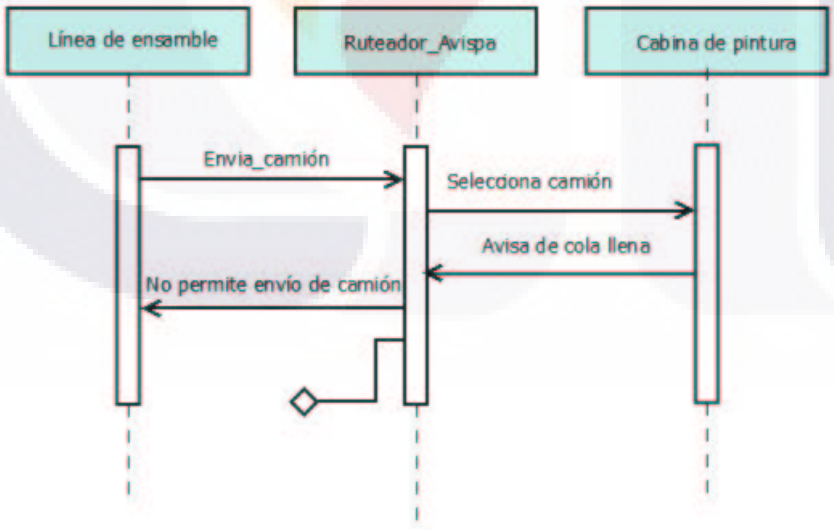


Figura 6.5: Diagrama de interacción Línea ensamble-Rutedor Avispa-Cabina pintura

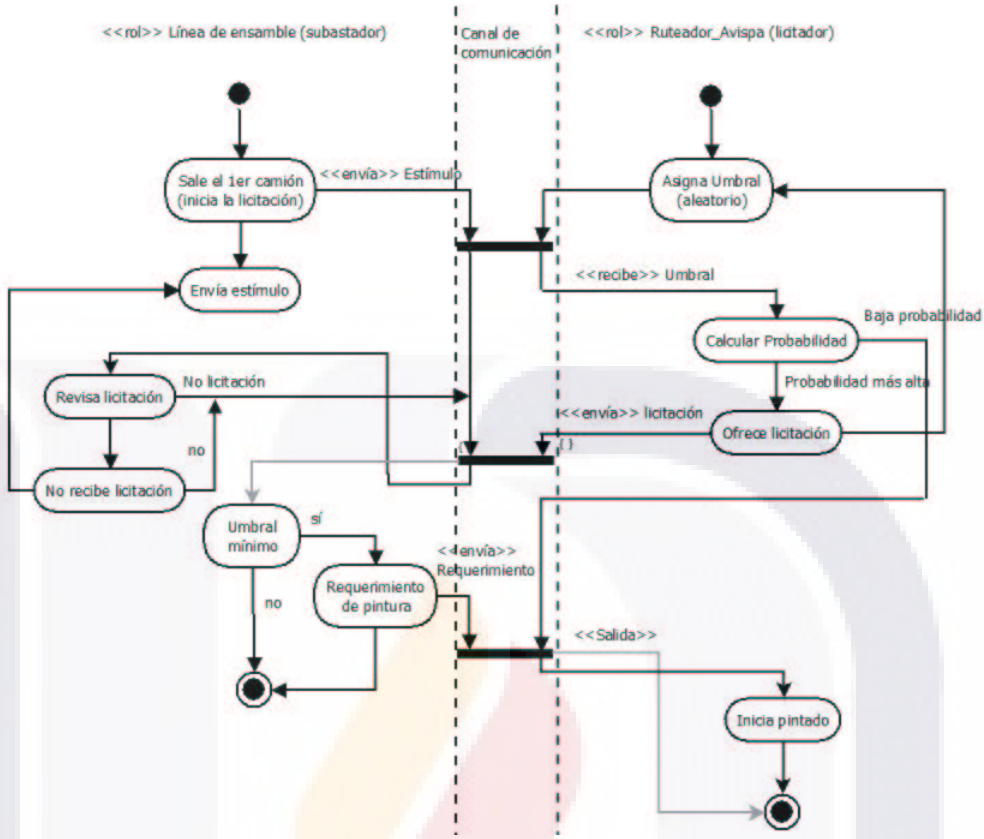


Figura 6.6: Diagrama de actividades del proceso de licitación de los Rutedores Avispa

El proceso de asignar camiones a cabinas utiliza el paradigma de mercado, en donde los Ruteadores Avispa licitan por los camiones que salen de la línea de ensamble, estos generan un *Estímulo* que dispara el mecanismo, conforme transcurre el tiempo, si no se atiende dicho estímulo se va incrementando. El protocolo generado se muestra en la siguiente figura:

6.2. Glosario del Proyecto

Elemento	Concepto	Identificador
Número de camiones	**Constante global** que establece el numero entero camiones que se van a ingresar al sistema de simulación.	*NUM_UNID* .
Número de cabinas	**Constante global** que establece el numero entero cabinas de pintura de que consta el sistema de simulación.	*NUM_CAB*
Línea de producción	**Estructura** que contiene las variables relacionadas con la línea de producción de camiones que entrarán al proceso de pintura.	*Linea_produccion*
Número de camión	**Variable** que asigna un valor numérico entero consecutivo a cada camión que sale de la línea de producción.	*Numero_camion*
Color de camión	**Variable** que asigna un valor numérico entero a cada color de los 14 empleados en la planta de pintura.	*Color_camion*
Estímulo	**Variable** que asigna un valor numérico entero que se incrementa conforme transcurren los pasos de la simulación a cada camión.	*Estimulo*
Camión	**Arreglo** que almacena la estructura de datos de cada camión	*Camion NUM_UNID*
Ruteador Avispa	**Estructura** que contiene las variables relacionadas con los ruteadores avispa asociados a cada cabina que determinarán la licitación o no de cada camión proveniente de la línea de producción.	*RA*
Fuerza	**Variable** que establece el valor numérico natural asociado a la fuerza con la que cada ruteador licita por un trabajo cuando dos o mas de ellos.	*Fuerza*
Umbral	**Variable** que establece el valor numérico entero del umbral del ruteador	*Umbral*
Umbral mínimo	**Variable** que establece el valor numérico natural del límite inferior del umbral	*Umbral_minimo*
Umbral máximo	**Variable** que establece el valor numérico natural del límite superior del umbral	*Umbral_maximo*
Umbral temporal	**Variable** que establece el valor numérico natural para actualizar el valor del Umbral	*Umb_temp*
Probabilidad	**Variable** que establece el valor numérico natural para calcular la probabilidad de licitación	*Probabilidad*
δ_1	**Variable** que establece el valor numérico natural del parámetro delta 1	*delta1*
δ_2	**Variable** que establece el valor numérico natural del parámetro delta 2	*delta2*
**********	**********	**********

Elemento	Concepto	Identificador
δ_3	**Variable** que establece el valor numérico natural del parámetro delta 3	*delta3*
Cabina de pintura	**Estructura** que contiene las variables relacionadas con las cabinas de pintura que procesaran los camiones provenientes de la línea de producción	*Cabinas_pintura*
Estado de la cabina	Variable que establece el valor numérico entero asociado a si la cabina está funcionando (valor=1) o no (valor=0)	*Estado_cabina*
Cabina en proceso	Variable que establece el valor numérico entero asociado a si la cabina está pintando un camión (valor =1) o si está ociosa (valor = 0)	*Cabina_en_ proceso*
Color en proceso	**Variable** que establece el valor numérico entero asociado al color de pintura con el que se ha cargado el sistema de cada cabina.	*Color_en_ proceso*
Espacio 1 de cola	**Variable** que establece el valor numérico entero asociado a si el primer espacio de la cola de una cabina está ocupado por un camión (valor =1) o si está vació (valor = 0).	*Espacio1_cola*
Espacio 2 de cola	**Variable** que establece el valor numérico entero asociado a si el segundo espacio de la cola de una cabina está ocupado por un camión (valor =1) o si está vació (valor = 0).	*Espacio2_cola*
Espacio 3 de cola	**Variable** que establece el valor numérico entero asociado a si el tercer espacio de la cola de una cabina está ocupado por un camión (valor =1) o si está vació (valor = 0).	*Espacio3_cola*
Actualizaciones	**Variable** que establece el valor numérico entero asociado al numero de veces que cambia de pintura una cabina.	*No_ actualizaciones*
Cabina	**Arreglo** que almacena la estructura de datos de cada cabina de pintura.	*Cabina_pintura[]*

Tabla 6.1: Glosario del Proyecto

Capítulo 7

CONCLUSIONES

7.1. Aportaciones

En este trabajo las aportaciones principales se ubican dentro de las prácticas de ingeniería y el área de inteligencia artificial.
La automatización en las áreas de producción es un trabajo complejo que repercute tanto en el cambio de las formas de producción, buscando mejorar la calidad de los productos así como abatir los costos y así atraer a más clientes.
La forma de implementar esta automatización no siempre consigue tales objetivos, la arquitectura propuesta mejora la forma en la que se implementan sistemas de información y control en una fábrica al utilizar las herramientas de modelado comunes para los ingenieros de software y por otro lado también a los ingenieros de manufactura al utilizar herramientas conocidas
La integración del enfoque multiobjetivo a la arquitectura como un elemento integrado

7.2. Limitaciones y líneas de investigación abiertas

Como limitación se debe señalar la necesidad de una mayor validación con nuevos casos de aplicación a diversos sistemas de manufactura, también es de señalar una mayor definición del trabajo en equipo por parte tanto de ingenieros de software como de manufactura para delimitar las áreas de trabajo en vista a una mayor eficiencia en el desarrollo y operación de los proyectos.

7.3. Trabajo futuro

Se enfoca principalmente sobre tres áreas:
La mejora de la arquitectura, afinando detalles y cubriendo zonas no-totalmente definidas.
La mejora en los procesos de generación de código, orientándola hacia lenguajes comerciales gráficos, y su implementación.
La generación de herramientas, para automatizar los proecesos de modelado de los diferentes tipos de agentes.

Bibliografía

[Andersen, 1997] Andersen, E.P.; *"Conceptual Modeling of Objects: A Role Modeling Approach"*; PhD thesis; Faculty of Mathematics and Natural Sciences, University of Oslo, Norway; (1997).

[Applegate & Cook, 1991] Applegate D., Cook W.; *"A Computational Study of the Job Shop Scheduling problem"*; ORSA Journal on Computing; Vol.3 No.2; pp. 149-156; (1991).

[Arzén, 2006] Arzén K.L.; *"Real Time Systems"*; Department of Automatic Control Lund Institute of Technology, Lund University, Sweden; http://www.control.lth.se/ kurstr/ ; (2006).

[Babaoglu, 2002] Babaoglu,O.; *"Anthill: A framework for the development of agent-based Peer-to-Peer System"*; In Proceedings of the 22nd International Conference on Distributed Computing Systems; Vienna, Austria. IEEE Computer Society, Digital Library; pp. 15-22; (2002)

[Bandini, 2005] Bandini S., et al.; *"A spatially dependent communication model for ubiquitous systems"*; In (Weyns, et al. 2005), pp. 74-90; (2005).

[Baker et al. 1997] Baker, A.D., Parunak, H.V., Erol, K.; *"Manufacturing over the Internet and into Your Living Room: Perspectives from the AARIA Project"*; ECECS Dept. Technical Report TR208-08-97; (1997).

[Bates, 1994] Bates, J.; *"The role of emotion in believable agents"*; Communications of the ACM, 37(7); (1994).

[Bates et al., 1992] Bates, J., Bryan Loyall, A., Scott Reilly, W.; *"An architecture for action, emotion and social behavior"*; Technical Report CMU CS 92 144; School of Computer Science; Carnegie-Mellon University Pittsburgh PA.; (1992).

[Belecheanu et al., 2006] Belecheanu Roxana A., Munroe Steve, Luck Michael, Payne Terry; *"Commercial Applications of Agents: Lessons, Experiences and Challenges"*; AAMAS´06 ACM Press, pp. 118-122; (2006).

[Benítez, 2002] enítez Peréz H.; *"Sistemas en Tiempo Real"*; DISCAIIMAS-UNAM; (verificar link); (2002).

[Boehm, 2001] Boehm B. W.; *"The Spiral Model as a Tool for Evolutionary Acquisition"*; In CrossTalk The Journal of Defense Software Enginnering, May:4-11; (2001).

[Booch, 1998] Booch G.; *"Análisis y diseño orientado a objetos con aplicaciones"*; Addison Wesley Longman; México; (1998).

[Boose et al., 1990] Boose John, Hin Motoda, H., Mizoguchi, R., Boose, J. H., and Gaines; *"Knowledge Acquisition Tools, Methods, and Mediating Representations"*; B.R.(Eds.); Proceedings of the First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop: JKAW-90, Ohmsha ,Ltd: Japan; (1990).

[Bonabeau et al., 1998] Bonabeau, E., Theraulaz, G., Deneubourg, J. L.; *"Fixed response threshold and the regulation of division of labor in insect societies"*; Bulletin of Mathematical Biology; 60, pp. 753-807; (1998).

[Botti & Giret, 2005] Botti Navarro Vicente, Giret Boggino Adriana; *"Metodología Multi Agente para Procesos Industriales"*; Actas del CAIP 2005: 7º Congreso Interamericano de Computación Aplicada a la Industria de Procesos, 365-368; (2005).

[Brooks, 1986] Brooks, R. A.; *"A robust layered control system for a mobile robot"*; IEEE Journal of Robotics and Automation; 2(1) 14-23 (1986).

[Bruckner, 2000] Sven Bruckner; *"Return from the ant synthetic ecosystems for manufacturing control"*; Mathematisch Naturwissenschaftlichen Fakultat II Humboldt Universitat zu Berlin; Diseertation zur Erlangung des akademischen Grades doctor rerum naturalium; (2000).

[Brussel et al., 1998] Brussel, H.V., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.; *"Reference Architecture for Holonic Manufacturing Systems: PROSA"*; Computers In Industry, special issue on intelligent manufacturing systems, Vol. 37, No. 3, pp. 255-276; (1998).

[Caridi & Cavalieri, 2004] aridi M., Cavalieri S.; *"Multi-agent systems in production planning and control: an overview"*; Production Planning & Control, 15, 106-118; (2004).

[Castillo et.al., 2002] Castillo E., Conejo A.J., Pedregal P., Garcia R. y Alguacil N.; *"Formulación y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia"*; Wiley and Sons, España; (2002).

[Castillo Vidal, 1998] Castillo Vidal Luis; *"Desarrollo y aplicación de técnicas de planeación no lineal para la programación del control de plantas industriales"*; Tesis Doctoral; Dpto. de Ciencias de Computación e Inteligencia Artificial, Universidad de Granada; (1998).

[Chang et.al., 2005] Chang P., et al.; *"From reality to mind: A cognitive middle layer of environment concepts for believable agents"*, En (Weyns et al. 2005a), pp. 57-73, (2005).

[Chase, 2001] Chase Richard B., Aquilano Nicholas J., Jacobs F.Roberts; *"Operations Management for Competitive Advantage"*; 9a Edición USA, McGraw Hill; (2001)

[Chirn & McFarlane, 2000] Chirn, J.L., McFarlane, D.C.; *"Petri Nets based design of ladder logic diagrams"*; Control 2000; Cambridge, UK; (2000).

[Christensen, 1994] Christensen, J.H.; *"Holonic manufacturing systems: initial architecture and standards directions"*; In Proceedings of First European Conference on Holonic Manufacturing Systems; Hanover, Germany; (1994).

[Cicirello & Smith, 2004] Cicirello Vincent A., Smith Stephen F.; *"Wasp-like agents for Distributed Factory Coordination"*; Journal of Autonomous Agents and Multi-Agent Systems; 8(3):237-266; (2004).

[Cicirello et al., 2001] Cicirello Vincent A., Smith Stephen F.; *"Insect Societies and Manufacturing"*; The IJCAI-01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing; (2001).

[Cuesta, 2005] Cuesta Morales Pedro; *"Ingeniería de Software Orientada a Agentes"*; Grupo Web de Agentes Inteligentes; Universidad de Vigo, España; (2005).

[Cockburn & Jennings 1993] Cockburn D., Jennings N.R.; *"ARCHON: A Distributed Artificial Intelligence System For Industrial Applications"*; EA Technology, Department of Electronic Engineering, Queen Mary and Westfield College, University of London; (1993).

[Cutkosky et al., 1996] Cutkosky, M.R., Tenenbaum, J.M. and Glicksman J. Madefast; *"Collaborative Engineering over the Internet"*; Communication of the ACM, 39(9):78-87; (1996).

[Deen, 1994] Deen, S.M.; *"A cooperation framework for holonic interactions in manufacturing"*; In Proceedings of the Second International Working Conference on Cooperating Knowledge Based Systems (CKBS 94); DAKE Centre, Keele University; (1994).

[Dorfman & Thayer, 1997] Dorfman M., Thayer R. H., editors; *"Software Engineering"*; IEEE Computer Society Press; (1997).

[Ericson & Johnson, 2006] Ericson S., Johnson J.; *"Computational Complexity of Shop Floor Scheduling"*; Technical Report COT-6410; University of Central Florida; (2006).

[Ferber, 1999] Ferber J.; *"Multiagent Systems, An Introduction to Distributed Artificial Intelligence*; Addison Wesley; (1999)

[Ferber, 2003] Ferber J., et al; *"From agents to organizations: An organizational view of multi-agent system"*; In P. Giorgini, J. P. MAuller, & J. Odell (eds.), Agent-Oriented Software Engineering IV, vol. 2935 of Lecture Notes in Computer Science, pp. 214-230, Melbourne, Australia. Springer, Berlin, Heidelberg, Germany, (2003).

[Ferguson, 1992] Ferguson, I. A.; *"TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents"*; PhD thesis; Clare Hall, University of Cambridge, UK; (1992).

[Fernández et.al., 2003] Fernández Chamizo,C., Gómez Sanz,J.,Pavón Mestras,L.; *"Modelos y Arquitecturas de Agentes"*, Dep. de Sistemas Informáticos y Programación; `http://grasia.fdi.ucm.es`; (2003)

[Fipa, 2006] Foundation for Intelligent Physical Agents; *"Framework para evaluación de proyectos de sistemas basados en agentes"*; `http://www.fipa.org`; (2006).

[Franklin & Graesser, 1996] Franklin,S., Graesser, A.; *"Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents"*, Institute for Intelligent Systems University of Memphis(1996)

[Fox et al., 1993] Fox, M.S., Chionglo, J.F., and Barbuceanu, M.; *"The Integrated Supply Chain Management System"*; Internal Report, Dept. of Industrial Engineering; Univ. of Toronto; (1993).

[Fox & Sycara, 1990] Fox, M.S., Sycara K. P.;.; *"Overview of CORTES: A constraint based approach to production planning, scheduling and control"*; Proceedings of The Fourth International Conference on Expert systems in Production and Operations Management, Hilton Head Island, SC; (1990).

[Gaines, 1992] Gaines Brian R.; *"Manufacturing in the Knowledge Economy"*; Proceedings of ICOOMS 92: International Conference on Object-Oriented Manufacturing; 19-36; (1992).

[García & Grillo, 2005] García Jiménez R., Grillo Perelello J.; *"Sistemas de Fabricación Flexible"*; Ingeniería Técnica Industrial esp. Electrónica Industrial (ITIEI); Universitat de les Illes Balears; (2005).

[Giret, 2005] Giret Boggino A.; *"ANEMONA: UNA METODOLOGIA MULTI-AGENTE PARA SISTEMAS HOLONICOS DE FABRICACION"*; Tesis Doctoral; Universidad Politécnica de Valencia, Dpto. de Sistemas Informáticos y Computación; (2005).

[Gómez, 2002] Gómez Sanz Jorge J.; *"MODELADO DE SISTEMAS MULTI-AGENTE"*; Tesis Doctoral; Departamento de Sistemas Informáticos y Programación, Facultad de Informática, Universidad Complutense de Madrid; (2002).

[Harris-Jones, 1995] Harris-Jones C.; *"Knowledge-Based Systems Methods. A Practitioner´s Guide"*; The Practitioner Series. Prentice-Hall; (1995).

[Hatvany, 1985] Hatvany J.; *"Intelligence and cooperation in heterarchic manufacturing systems"*; Robotics and Computer-Integrated Manufacturing 2(2):101-104; (1985)

[Heijst, 1994] Heijst G. Van; *"The Role of Ontologies in Knowledge Engineering"*; PhD thesis; University of Amsterdam; (1994).

[Heikkila et al.1999] HEIKKILA T., KOLLINGBAUM M., VALCKENAERS P., BLUEMINK G.J.; *"manAge: an agent architecture for manufacturing control"*; VTT Automation, University of Cambridge, Katholieke Universiteit Leuven; (1999).

[HMS, 1994] Holonics Manufacturing Systems; *"HMS Requirements"*; Press Release, HMS Server; `http://hms.ifw.uni-hannover.de/index.htm`; (1994).

[Horling et al., 2006] Horling B., Lesser V., Vincent R., Wagner T.; *"The Soft Real-Time Agent Control Architecture"*; Autonomous Agents and Multi-Agent Systems; 12: 35-91; Springer Science+Business Media, Inc; (2006).

[Im Cho, 2008] Im Cho Young; *"Intelligent multiagent application system in an AI system"*; Artificial Life Robotics, 12:6-13; (2008).

[Jacobson et al., 2000] Jacobson, I., Booch, G. y Rumbaugh, J.; *"El Proceso Unificado de Desarrollo de Software"*; Addison Wesley, 303-379; (2000).

[Julián, 2002] Julián Inglada Vicente J.; *"RT-MESSAGE: Desarrollo de Sistemas Multiagente de Tiempo Real"*; Tesis Doctoral; Universidad Politécnica de Valencia; Diciembre (2002).

[Julián & Botti, 2003] Julián V., Booti V.; *"Estudio de métodos de desarrollo de sistemas multiagente"*; Revista Iberoamericana de Inteligencia Artificial; No.18, 65-80; ISSN: 1137-3601; (2003).

[Larousse, 2003] Diccionario Larousse, Edición Premium, EDICIONES LAROUSSE MÉXICO y SPS EDITORIAL BARCELONA, (2003).

[Liu J.S., 1996] Liu Jyi-Shane; *"Coordination of Multiple Agents in Distributed Manufacturing Scheduling"*; PhD thesis; The Robotics Institute, School of Computer Science, Carnegie Mellon University; (1996).

[Maes, 1994] Maes, Pattie; *"Agents that Reduce Work and Information Overload"*; Communications of the ACM, Vol 37(7), pp-36, ACM Press; (1994).

[McKean et al., 2007] McKean MJez, Shorter Hayden, Luck Michael, McBurney Peter, Willmott Steven; *"Technology Diffusion: Analysing the Diffusion of Agent Technologies"*; Journal of Autonomous Agents and Multiagent Systems; 12(2): 1-34, Kluwer Academic Publishers; (2007).

[Monfared & Yang, 2006] Monfared M.S., Yang J.B.; *"Design of integrated manufacturing planning scheduling and control systems: a new framework for automation"*; International Journal of Advanced Manufacturing Technologies; DOI 10.1007/s00170-006-0476-8, Springer-Verlag London Limited; (2006).

[Moulin, 1996] Moulin and B. Chaib-draa; *"Foundations of Distributed Artificial Intelligence: An Overview of Distributed Artificial Intelligence"*; John Wiley & Sons; (1996).

[Muller, 1996] Muller H. J.; *"Multi-agent systems engineering"*; In Second Knowledge Engineering Forum, Karlsruhe; (1996).

[Muller, 1996] Muller J. P., Pischel M.; *"Modelling interacting agents in dynamic environments"*; In Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94); 709-713; (1994).

[Nwana, 1996] Hyacinth S. Nwana; *"Software Agents: An Overview"*; Knowledge Engineering Review, Vol. 11, No 3, pp. 205-244; Cambridge University Press; (1996).

[Noriega & Sierra, 2002] Noriega P. & Sierra C.; *"Electronic Institutions: Future Trends and Challenges"*; In Proceedings of the 6th International Workshop on Cooperative Information Agents, vol. 2446 of Lecture Notes in Computer Science, pp. 14-17. Springer-Verlag, London, UK, (2002).

[Obi, 1999] Obi, S.C.; *"A Framework for Implementing Appropriate Manufacturing Systems in Developing Economies"*; In The Journal of Industrial Technology, volume 15, number 2, pp. 1-6; (1999).

[Page-Jones, 1980] Page-Jones M.; *"The Practical Guide to Structured Systems Design"*; Yourdon Press, New York. Found of UNISYS; (1980).

[Parunak 1997] Parunak H.; *"Go to the Ant: Engineering principles from natural multi-agent systems"*; Annals of Operation Research; 75, pp. 69-101; (1997).

[Parunak et al.1997] Parunak H., Sauter J., Clark S.; *"Toward the Specification and Design of Industrial Synthetic Ecosystems"*; Four International Workshop on Agent Theories, Architectures, and Languages; (1997).

[Parunak 1998] Parunak H.; *"Practical and industrial applications for agent based systems"*; Industrial Technology Institute; (1998).

[Parunak et.al. 2001] Parunak,H.,Odell,J.,Fleischer,M.,Brueckner,S.; *"Modeling Agents and their Environment"*; ERIM and James Odell Associates; (2001).

[Parunak et.al. 2001b] Parunak H., Baker A.D., Clark S.J.; *"The AARIA agent architecture: From manufacturing requeriments to agent-based system design"*; Integrated Computer-Aided Engineering, 8: 45-88; (2001).

[Pechoucek & Marik, 2008] Pechoucek M., Marik V.; *"Industrial deployment of multi-agent technologies: review and selected case studies"*; Auton Agent Multi-Agent Syst DOI 10.1007/s10458-008-9050-0; Springer Science+Business Media; (2008).

[Peeters et al., 1998] Peeters P., Tapio Heikkila, Stefan Bussmann, Jo Wyns, Paul Valckenaers, Hendrik Van Brussel; *"Novel Manufacturing System Requirements in Automated, Line-Oriented Discrete Assembly"*; Katholieke Universiteit Leuven - Mechanical Engineering Department, VTT Automation, Daimler-Benz AG; MASCADA Project; (1998).

[Picard, 2003] Picard, G.; *"UML Stereotypes Definition and AUML Notations for ADELFE Methodology with OpenTool"*; First European Workshop on Multi-Agent Systems, St. Catherine College Oxford; (2003).

[Powers et al., 1990] Powers M.J., P. H. Cheney, and G. Crow; *"Structured Systems Development: Analysis, Design, Implementation"*; Boyd y Fraser, 2nd edition; (1990).

[Pressman, 2006] Pressman R. S.; *"Ingeniería de Software: Un enfoque práctico"*; Editorial McGraw Hill, 6a Ed. México; (2006).

[Robinson, 1992] Robinson G.E.; *"Regulation of division of labor in insect socities"*; Annu. Rev. Entomol.; 37; pp. 637-665; (1992).

[Roumeliotis, 2002] Roumeliotis, S. I. Bekey; *"G. A. Distributed multi-robot localization"*; IEEE Trans. Robotics and Automation, 18(5):781-795; (2002).

[Russell & Norvig, 2006] Russell S., Norvig P.; *"Inteligencia Artificial: un enfoque moderno"*; 2a Edición, España; Pearson Prentice Hall; (2006).

[Sadeh et al., 1999] Sadeh N.M., Hildum D.W., Kjenstad D., Tseng A.; *"MASCOT: An Agent-Based Architecture for Coordinated Mixed-Initiative Supply Chain Planning and Scheduling"*; Third International Conference on Autonomous Agents (Agents '99) Workshop on Agent-Based Decision Support for Managing the Internet-Enabled Supply Chain, Seattle; (1999).

[Sommerville, 2002] Sommerville Ian; *"Ingeniería de Software"*; Pearson Educación, México; (2002).

[Stankovic, 1990] Stankovic, J.A., Ramamritham, K.; *"What is Predictability for Real-Time Systems"*; Journal of Real-Time Systems, 2(4): pp. 247-254; (1990).

[Suri, 2003] Suri Rajan; *"QRM and POLCA: A Winning Combination for Manufacturing Enterprises in the 21st Century"*; Technical Report; Center for Quick Response Manufacturing; University of Wisconsin-Madison; (2003).

[Tansly et al., 1993] Tansley, D. S. W. y Hayball, C. C.; *"Knowledge Based systems Analysis and Design a KADS developer's handbook"*; Prentice Hall; (1993).

[Theraulaz et al., 1991] Theraulaz, G., Goss, S., Gervet, J., Deneubourg, J.L.; *"Task differentiation in Polistes wasp colonies: a modelf or self-organizing groups of robots"*; Proceedings of the first international Conference on Simulation of Adaptive Behavior, On From Animals to Animats; pp 346-355; (1991).

[Theraulaz et al., 1994] Theraulaz, G., Bonabeau, E., Deneubourg, J:L.; *"Response threshold reinforcement and division of labour in insect societies"*; Proc. R. Soc. London B, vol. 265, no. 1393, pp. 327-335; (1998).

[Theraulaz et al., 1995] Theraulaz, G., Bonabeau, E., Deneubourg, J:L.; *"Self-organization of hierarchies in animal societies: The case of the primitively eusocial wasp polistes dominulus christ"*; J. Theor. Biol.; vol. 174, pp. 313-323; (1995).

[Tianfield et al., 2003] Tianfield H., Tian J., Yao X.; *"On the Architectures of Complex Multi-Agent Systems"*; Proceedings of the Workshop on "Knowledge Grid and Grid Intelligence"(ISBN 0-9734039-0-X), held at the 2003 IEEE/WIC International Conference on Web Intelligence / Intelligent Agent Technology, October 13 - 16, 2003, Halifax, Canada, pp. 195 - 206; (2003).

[Toro Ocampo et al., 2006] Toro Ocampo Eliana L., Restrepo Grisales Yov, Granada Echeverri M.; *"Algoritmo Genético modificado aplicado al problema de secuenciamiento de tareas en sistemas de producción lineal - flow shop"*; Scientia et Technica Año XII, No 30, ISSN 0122-1701, (2006).

[Ulieru et al., 2002] Ulieru Mihaela, Brennan Robert W., Scott S. Walker; *"The holonic enterprise: a model for Internet-enabled global manufacturing supply chain and workflow management"*; Integrated Manufacturing Systems 13(8):538 - 550; (2002).

[Ulieru, 2004] Ulieru Mihaela, Este Robert A.; *"The Holonic Enterprise and Theory Emergence: On emergent features of self-organization in distributed virtual agents"*; Cybernetics And Human Knowing; 11(1):79-98; (2004).

[Vestraete et al., 2006] Verstraete, P., Saint Germain, B., Hadeli, K., Valckenaers, P., Van Brussel, H.; *"On applying the PROSA reference architecture in multi-agent manufacturing control"*; K.U.Leuven, Leuven, Belgium; (2006).

[Vienneau, 1997] Vienneau R., *"A review of formal methods"*; In M. Dorfman and R. H. Thayer, editors, Software engineering; IEEE Computer Society, 181-192; (1997).

[Vlassis, 2003] Vlassis Nikos; *"Systems and Distributed AI Intelligent Autonomous Systems. A Concise Introduction to Multiagent"*; Edit. Informatics Institute University of Amsterdam; (2003).

[Weyns, 2005] Weyns, D.; *"Agents are not part of the problem, agents can solve the problem"*; AgentWise, DistriNet, Department of Computer Science, K.U.Leuven, B-3001 Heverlee, Belgium; (2005)

[Weyns et al., 2005] Weyns D., Helleboogh A., Steegmans E., De Wolf T., Mertens K., Boucké N., Holvoet T.; *"Agents are not part of the problem, agents can solve the problem"*; Proceedings of the OOPSLA Workshop on Agent-Oriented Methodologies, (Gonzales-Perez, C., ed.), pp. 101-112; (2005).

[Weyns & Parunak, 2005] Weyns, D.,Parunak,H.V.,Michel,F.,Holvoet,T.,Ferber,J.; *"Environments for Multiagent Systems State-of-the-Art and*

Research Challenges"; D. Weyns et al. (Eds.): E4MAS 2004, LNAI 3374, pp. 1-47. Springer-Verlag Berlin Heidelberg (2005).

[Wilson & Holldobler 1988] Wilson, E.O.,Holldobler B.; *"Dense heterarchies and mass communications as the basis of organizations in ants colonies"*; Trends in Ecology and Evolution; 3, pp. 65-68; (1988).

[Wooldridge, 2002] Wooldridge Michael; *"An Introduction to MultiAgent Systems"*; John Wiley & Sons; (2002).

[Wooldridge & Jennings, 1995] Wooldridge, M., Jennings,R.N., *"Intelligent Agents: Theory and Practice"*, Knowledge Engineering Review, (1995)