UNIVERSIDAD DE LA LAGUNA

«Aportaciones metodológicas basadas en simulación e inteligencia artificial para la toma de decisiones en la gerencia hospitalaria»

Autor: Rosa Mª Aguilar Chinea
Director: Dr. Lorenzo Moreno Ruiz

Departamento de Física Fundamental y Experimental, Electrónica y Sistemas

D. LORENZO MORENO RUIZ, Doctor en Ciencias Físicas y Catedrático de Física Aplicada del Departamento de Física Fundamental y Experimental de la Universidad de La Laguna,

CERTIFICA:

> Que Dña. Rosa María Aguilar Chinea, Licenciada en Informática, ha realizado bajo mi dirección la presente Tesis Doctoral titulada APORTACIONES METODOLÓGICAS BASADAS EN SIMULACIÓN E INTELIGENCA ARTIFICIAL PARA LA TOMA DE DECISIONES EN LA GERENCIA HOSPITALARIA, para optar al grado de Doctor en Informática.

Con esta fecha autorizo la presentación de la misma.

La Laguna, Septiembre de 1998

El Director

Lorenzo Moreno Ruiz

A mi Madre,
a Fran y a Ana

Con Cariño

Quiero expresar mi mas sincero agradecimiento a todas y cada una de las personas que han colaborado en la realización de este trabajo.

En primer lugar, quisiera agradecer al Dr. D. Lorenzo Moreno Ruiz la dirección de esta tesis, la generosidad y ayuda prestada en la realización de cada una de las líneas de este trabajo. Y un agradecimiento mucho más grande si cabe, por permitirme descubrir la grandeza del mundo universitario.

Al Dr. D. José Demetrio Piñeiro Vera por sus incalculables aportaciones en este trabajo, y por enseñarme que sólo si te haces preguntas puedes encontrar respuestas.

A D. José Ignacio Estévez Damas, en quien he encontrado un gran amigo y que me ha acompañado y motivado durante todo el trabajo.

Al Dr. D. Alberto Hamilton Castro cuyos conocimientos de control contribuyeron a mejorar este proyecto.

Un fuerte agradecimiento a D. Carlos Martín Galán cuya ayuda resultó imprescindible en la dura tarea de programación.

Muchas gracias a todos mis compañeros del grupo de Computadoras y Control: Dr. D. Leopoldo Acosta Sánchez, Dr. D. José Luis Sánchez de la Rosa, Dra. Dña. Soledad Mañas, D. Juan Julian Merino Rubio, Dr. D. Juan Albino Méndez Pérez, D. José Sigut Saavedra, a D. Graciliano Nicolás Marichal Plasencia, Dña. Marta Sigut Saavedra, D. Roberto Marichal Plasencia y a D. Santiago Torres Álvarez, por ser tan buenos conmigo y contribuir a que adore mi trabajo. Y a D. Roberto Bentancor Bonilla personal de administración y servicio del departamento por su disponibilidad continua.

Un muy especial agradecimiento al gerente del Hospital Nra. Sra. de la Candelaria, D. Alberto Talavera Déniz, cuya disponibilidad y asesoramiento ha permitido que este trabajo sea una realidad.

Infinitas gracias a Dña Katy Mary Sánchez, por su ayuda generosa y desinteresada en todo el proceso de adquisición de conocimiento. Y muchas gracias a todo el personal que trabaja con la gerencia del Hospital Nra. Sra. de la Candelaria por su colaboración y buena disposición en todos los problemas que les he planteado.

Muchísimas gracias a Dña. Nieves Gracia Lezcano que con su gran conocimiento sobre hospitales construyó unos cimientos muy firmes en el comienzo de este trabajo.

También quiero agradecer a mis compañeros del proyecto: Dra. Dña. Victoria Jiménez González, Dra. Dña. Beatriz González López, Dr. D. Santiago Rodríguez Feijoó, Dra. Dña. Patricia Barber Pérez y D. Miguel Angel González Lugo por su asesoramiento.

Gracias a D. Pedro Manuel Pérez Méndez y a Dña. Alicia Aránzazu de Torres Capel por su colaboración en la implementación del prototipo desarrollado.

Agradecer al Gobierno de Canarias la financiación de este trabajo que ha sido subvencionado con el proyecto "Diagnóstico económico financiero de la empresa canaria mediante sistemas basados en el conocimiento" nº 93/020 y el proyecto "Diseño de sistemas mediante técnicas de simulación e inteligencia artificial para el análisis de la estructura funcional de gestión en hospitales canarios" nº PI1997/015.

ÍNDICE

INTRODUCCIÓN

Capitulo I
La simulación y la inteligencia artificial en el estudio de sistemas complejos: introducción histórica y conceptual

1.- INTRODUCCIÓN 1
2.- SISTÉMICA 3
2.1.-TEORÍA CLÁSICA DEL CONTROL 5
2.2.- CIBERNÉTICA 5
3.- SISTEMAS 6
3.1.- SUBSISTEMAS 9
3.1.1.- Relaciones entre los subsistemas 10
3.2.- CLASIFICACIÓN DE LOS SISTEMAS 12
3.3.- SISTEMAS COMPLEJOS 14
3.4.- DINÁMICA DE SISTEMAS 20
4.- MODELADO 21
4.1.- TIPOS DE MODELOS 22
5.- SIMULACIÓN 24
5.1.- SIMULACIÓN COMPUTACIONAL 26
5.2.- SIMULADORES 27
5.3.- ANIMACIÓN 28
5.4.- ETAPAS EN EL PROCESO DE SIMULACIÓN 30
5.5.- VENTAJAS Y DESVENTAJAS DE LA SIMULACIÓN 35
5.7.- LENGUAJES DE SIMULACIÓN 38
6.- INTELIGENCIA ARTIFICIAL 42
6.1.- SISTEMAS EXPERTOS 43
6.2.- ÁREAS DE APLICACIÓN 44
6.3.- TIPOS DE APLICACIONES 46
6.4.- DEFINICIÓN DE SISTEMA BASADO EN EL CONOCIMIENTO 48
6.5.- CONSTRUCCIÓN DE UN SBC. METODOLOGÍA KADS. 49
6.5.1.- Principio 1: múltiples modelos 50
6.5.2.- Principio 2: modelado de las capas de conocimiento 52
7.- EL HOSPITAL: UN SISTEMA COMPLEJO 54

7.2.- La sanidad: perspectiva histórica 55
7.3.- El sistema sanitario español 57
8.- Gestión hospitalaria 59

Capitulo II
Modelado y simulación para la toma de decisiones en la gerencia hospitalaria

1.- Introducción 63
2.- Análisis de un hospital 64
3.- Modelado del hospital 66
4.-flujo de pacientes en el hospital 68
4.1.- Modelado 68
4.2.- Simulación 71
4.2.1.- Redes de petri 73
4.2.1.1.- Extensiones de las redes de petri ordinarias 75
4.2- simulación del flujo de pacientes por el hospital basado en redes de petri 79
4.3.- Simulación orientada al proceso 81
4.4.- Selección de la herramienta de programación 83
4.5.- Implementación de un prototipo para la simulación del flujo de pacientes 86
4.6.- Implementación de la dinámica de un hospital 93
4.6.1.- Simulación microscópica 99

Capítulo III
Sistema basado en el conocimiento para la ayuda a la toma de decisiones en la gerencia hospitalaria

1.- Introducción 105
2.-toma de decisiones en la gerencia hospitalaria 107
3.- Modelo organizativo 112
4.- Modelo de tareas 113
5.- Modelo de agentes 117
6.- Modelo experto 119

6.1.- Conocimiento de dominio 119
6.1.1.- Ontología para la gestión hospitalaria 122
6.2.- Conocimiento de control 133
6.2.1.- Toma de decisiones en la gestión hospitalaria 138
6.2.1.1.- Tarea de monitorización 141
6.2.1.2.- Inferencias realizadas en la monitorización 141
6.2.2.- Tarea de diagnóstico 146
6.2.2.1.- Inferencias realizadas en el diagnóstico 148
6.2.3.- Tarea de predicción 153
6.2.3.1.- Inferencias realizadas en la predicción 155
6.2.3.2.- Piezas de conocimiento complejas utilizadas en la predicción 157
6.2.4.- Tarea de diseño 172
6.2.4.1.- Inferencias realizadas en el diseño 172
6.2.4.2.- Piezas de conocimiento complejas utilizadas en el diseño 177
7.- Desarrollo de un prototipo 185

Conclusiones

Apéndice a

Apéndice b

Bibliografía

INTRODUCCIÓN

INTRODUCCIÓN

El presente trabajo se enmarca en una línea de investigación desarrollada por miembros del Grupo de Computadoras y Control del Departamento de Física Fundamental y Experimental de la Universidad de La Laguna, en colaboración con miembros del Departamento de Economía Aplicada de las Universidades de La Laguna y Las Palmas de G.C, para el estudio de la gestión en las organizaciones sanitarias.

La necesidad de acelerar y asegurar la efectividad y eficiencia de las acciones administrativas sanitarias, de desarrollar nuevos instrumentos para la gestión, así como de conseguir un ahorro de dinero mediante la mejor distribución y empleo de los siempre escasos recursos sanitarios hacen que la investigación en gestión hospitalaria tenga una gran repercusión social. Como consecuencia, este estudio es una de las prioridades del Gobierno de Canarias. Este trabajo se desarrolla bajo esta perspectiva, y ha sido subvencionado por el Gobierno de Canarias con el proyecto "Diagnóstico económico financiero de la empresa canaria mediante sistemas basados en el conocimiento" nº 93/020 y el proyecto "Diseño de sistemas mediante técnicas de simulación e inteligencia artificial para el análisis de la estructura funcional de gestión en

hospitales canarios" nº PI1997/015.

Para conseguir los fines anteriormente mencionados, se están diseñando sistemas de información que permiten la estructuración de toda la información que se genera en los hospitales generales. Los modelos basados en patrones son un mecanismo de representación de la información que permite el almacenamiento y manipulación de los datos de forma ordenada, así como la reutilización de componentes diseñados en otras organizaciones. En esta memoria se presenta un sistema de gestión de la información orientado a conseguir organizaciones flexibles ante los cambios, con estructuras horizontales y orientadas al usuario, lo cual precisa del uso creativo de la tecnología de la información.

Este trabajo presenta una metodología, basada en Simulación e Inteligencia Artificial, que permite abordar de forma sistemática la planificación, la distribución de recursos y en definitiva la mejora de las actuaciones dentro de las organizaciones sanitarias.

En la primera fase de este estudio se plantea el análisis de los hospitales. Con el objetivo de conseguir un completo conocimiento de los mismos se diseña un programa de simulación con el que se crea un modelo del hospital bajo estudio. Esta herramienta permite experimentar las distintas situaciones que se presentan en estos sistemas para conocer los efectos que producen las acciones de control. La necesidad de una herramienta con estas características es debida a que los hospitales son sistemas muy complejos formados por muchas partes que se interrelacionan, y el conocimiento de cada una de ellas no es suficiente para determinar el comportamiento global del mismo. La simulación nos permite conocer el comportamiento emergente de todo el conjunto.

Con el modelo el personal de gerencia puede adquirir un conocimiento exhaustivo del funcionamiento de sus organizaciones. La información suministrada en este análisis permite detectar los cuellos de botella y ensayar las consecuencias de las acciones de control necesarias para corregir los funcionamientos erróneos. Esta tarea es muy complicada debido a la gran cantidad de información generada en el proceso de análisis, así como a que el estudio de estos datos se realiza a partir de conocimiento heurístico alcanzado con la experiencia en situaciones similares. Por lo tanto, se hace patente la necesidad de utilizar técnicas de Inteligencia Artificial que faciliten este trabajo

y ayuden en la toma de decisiones.

La segunda fase de la metodología que se presenta consiste en el desarrollo de un Sistema Basado en el Conocimiento (SBC) para la ayuda a la toma de decisiones en la gerencia hospitalaria. Para su diseño se sigue la metodología KADS que permite la estructuración del conocimiento y por lo tanto facilita tanto la construcción como la depuración de la base de conocimiento. Un aspecto igual de importante es que permite la reutilización de piezas de conocimiento utilizadas en otros procesos de razonamiento similares. Esto es posible debido a que todos los expertos en distintos campos del saber resuelven los problemas de la misma forma, independientemente del dominio de aplicación concreto. Así cualquier toma de decisiones se realiza mediante la ejecución de las siguientes cuatro tareas:

* monitorización: se observan las variables en estudio.

* diagnóstico: se detectan los problemas existentes.

* predicción: se determinan las posibles soluciones. En esta fase la simulación de la dinámica del sistema es un elemento imprescindible para detectar las consecuencias de las diferentes acciones.

* diseño: se construye la solución al problema. Toda tarea de diseño puede ser tratada como una búsqueda en un espacio de soluciones. La utilización de funciones heurísticas para recortar el espacio de búsqueda es clave en este proceso.

La cooperación de la Simulación y la Inteligencia Artificial se ha mostrado como un mecanismo muy adecuado en la toma de decisiones de organizaciones tales como las sanitarias.

La presente memoria desarrolla las dos fases de esta metodología de estudio de los hospitales generales, para lo cual ha sido dividida en tres capítulos.

En el Capítulo I se realiza una introducción a los conceptos que se utilizarán posteriormente. El estudio se realiza desde el punto de vista de la sistémica, considerando al hospital como un sistema complejo que necesita ser modelado y simulado. Tras presentar estos tópicos se introduce la Inteligencia Artificial como una ciencia que estudia y construye sistemas complejos. Aunque las técnicas de la IA son

muchas, este capítulo se centra en la presentación de los Sistemas Basados en el Conocimiento puesto que serán el mecanismo utilizado para representar la toma de decisiones.

En el Capítulo II se desarrolla un modelo del hospital que, debido a la complejidad del mismo, es dividido funcionalmente en tres subsistemas: flujo de pacientes por el hospital, modelo estático de los recursos humanos y gerencia hospitalaria. La dinámica del sistema la refleja el flujo de pacientes por los distintos servicios del hospital, que es un sistema de eventos discretos que puede ser modelado siguiendo una estrategia orientada al evento y utilizando Redes de Petri (RdP), o una metodología orientada al proceso. Como las RdP no muestran las características necesarias para representar un sistema de tanta envergadura, la solución consiste en realizar el modelo definiendo los procesos que tienen lugar en el hospital. Esta perspectiva tiene una ventaja añadida, que el estudio se realizada centrado en el paciente y esto resulta ser una pieza clave en cualquier gestión innovadora y moderna. Hemos diseñado un programa de simulación que permite modelizar cualquier tipo de hospital general.

En el Capítulo III se diseña el SBC que ayudará a la toma de decisiones en la gerencia hospitalaria. El primer paso en el desarrollo de este sistema ha sido la adquisición del conocimiento, realizada a través de múltiples entrevistas con personal experto en gestión hospitalaria: enfermeros, médicos, jefes de servicios, economistas y gerentes. Dada la complejidad de la tarea se ha utilizado la metodología KADS que se basa en los dos principios siguientes:

* Múltiples modelos: el estudio del SBC desde distintos puntos de vista permite que el ingeniero del conocimiento se pueda centrar en un problema concreto y abstraerse de la complejidad del desarrollo. Se presenta, por lo tanto, el modelo organizativo que sitúa el SBC dentro de la organización en la que trabajará. El modelo de tareas describirá las funciones a desarrollar por el mismo, y el modelo de agentes indicará con que elementos, de tipo software o usuario, debe cooperar el SBC para su correcto funcionamiento.

* Niveles de conocimiento: El conocimiento que debe tener el SBC es dividido en conocimiento de dominio (el propio de la aplicación concreta que desarrollamos) y

conocimiento de control (define los procesos de razonamiento), y es descrito independientemente de la implementación concreta que tendrá. Este mecanismo facilita la construcción del SBC permitiendo la reutilización de piezas de conocimiento.

En la definición del conocimiento de dominio se describen todos los conceptos y relaciones que necesita conocer el SBC para su trabajo. Esta descripción se realiza utilizando el lenguaje Ontolingua que permite el uso del lenguaje de intercambio de conocimiento (KIF), así como la reutilización de conceptos o relaciones definidos en otras teorías.

Para describir el conocimiento de control se descompone el proceso de razonamiento en los pasos más elementales, inicialmente se divide en tareas que para la toma de decisiones son la monitorización, el diagnóstico, la predicción y el diseño. Y éstas son a su vez divididas en procesos de razonamiento más elementales llamados inferencias.

Este trabajo inicial de definir la estructura de control es equivalente en cualquier toma de decisiones, pero en el siguiente paso se tiene que relacionar este conocimiento de control con el dominio concreto con el que trabaja el SBC que, en este diseño, es la gestión hospitalaria. Para ello, se especifican las estructuras de inferencia que muestra la relación entre los conocimientos de dominio y control.

Finalmente se presenta la selección del software de desarrollo y la construcción del SBC prototipo que sirve para mostrar como la cooperación de la IA y la simulación facilitan el estudio de los sistemas complejos.

La memoria se concluye con dos apéndices. En el primero, se incluye el listado del programa de simulación. En el segundo, se muestran diferentes procedimientos, realizados en MATLAB, para el desarrollo de este trabajo.

Capitulo I
La simulación y la inteligencia artificial en el estudio de sistemas complejos: introducción histórica y conceptual

CAPITULO I

LA SIMULACIÓN Y LA INTELIGENCIA ARTIFICIAL EN EL ESTUDIO DE SISTEMAS COMPLEJOS: INTRODUCCIÓN HISTÓRICA Y CONCEPTUAL.

1.- INTRODUCCIÓN

La Inteligencia Artificial (IA) es un campo multidisciplinario que abarca la informática, la neurociencia, la filosofía, la sicología, la ingeniería y la lingüística; y que intenta reproducir los métodos o resultados del razonamiento humano y de la actividad cerebral [Gregory 1995]. Una consecuencia de ser multidisciplinario es que el término IA significa diferentes cosas a diferentes personas.

En este trabajo se considera la IA desde dos puntos de vista, la IA como ciencia y la IA como ingeniería. Ambas perspectivas resultan al considerar que en todos los campos del saber es posible aplicar el concepto sistema, así como todas las técnicas y propiedades que resultan del estudio de los mismos. Por lo tanto la IA puede ser definida de las dos formas siguientes:

a) La IA es la ciencia que estudia sistemas complejos.- Entre ellos, el más

complejo es el sistema "mente", así que principalmente bajo este título se recogen todas las técnicas de la IA que permiten el estudio de los procesos cognitivos que manipulan la información y el conocimiento.

> *"Los objetivos de la IA son imitar por medio de máquinas, normalmente electrónicas tantas actividades mentales como sea posible y quizás llegar a mejorar las capacidades humanas"*. [Penrose 1989]

b) La IA es la parte de la ingeniería que construye sistemas inteligentes.- Aunque el término inteligente es difícil de definir, vamos a considerar que los sistemas inteligentes son aquellos que exhiben alguna de las características de inteligencia que poseen los humanos.

> *"La Inteligencia Artificial (IA) es la más reciente manifestación de un impulso permanente del hombre por crear artefactos que imiten nuestra propiedad esencial, la inteligencia"*.[McCorduck 1979]

En el estudio de los sistemas se utiliza el modelado y simulación de los mismos. Un modelo representa algunas propiedades del sistema original en un lenguaje específico. La simulación se define como el uso de un modelo para imitar el comportamiento de los sistemas, y por lo tanto, estudiar el rendimiento de los mismos bajo una variedad de circunstancias.

La complejidad de determinados sistemas impide la construcción de modelos matemáticos que los representen y con los que se puedan estudiar los mismos. Con la llegada de los ordenadores se solventa un poco este problema y se simulan sistemas que eran difíciles de estudiar anteriormente porque no tenían modelos matemáticos o éstos eran muy complejos. Pero siguen existiendo sistemas con una dinámica muy complicada, de forma que no se podían simular con las herramientas habituales o que, dada la complejidad de cálculo necesario era muy costosa su resolución. Por ello surge la necesidad de buscar nuevas estrategias de estudio. Encontramos ejemplos de este tipo de problemas, en el modelado de la dinámica de un robot o en la búsqueda y generación de trayectorias de robots móviles en un entorno con obstáculos [Fu 1988]. Para resolver estos problemas se han utilizado con éxito notable distintas técnicas de la IA (redes

neuronales, fuzzy, sistemas basados en conocimiento, algoritmos genéticos, ...) [Sánchez 1998].

En este trabajo, a través del análisis de un sistema complejo como es un hospital, se quiere dejar patente que la inteligencia artificial y la simulación son herramientas que se complementan en el estudio de dichos sistemas. La cooperación entre la IA y la simulación permite avanzar en el conocimiento de sistemas complejos. El primer paso en la descripción de esta cooperación es definir y establecer el marco de trabajo de ambas disciplinas.

2.- SISTÉMICA

El trabajo de la actividad científica consiste en organizar y ordenar los datos que aporta el análisis de la realidad. Al observar un hecho los datos recogidos forman un caos, para comprender la realidad debemos estructurar toda esta información de manera que se pongan de manifiesto las regularidades que presentan los datos. Cada disciplina científica se encarga de realizar las estructuraciones en un campo bien definido de la experiencia humana (el físico, el biológico, ...) de acuerdo con una metodología que a lo largo de la historia han definido esos campos del saber [Aracil 1986b].

La hipótesis de trabajo de la sistémica es que las distintas metodologías utilizadas en las diferentes disciplinas se pueden combinar de manera que sea posible construir una metodología común que haga posible entender y describir mejor cualquier realidad.

A la sistémica no se la considera una "ciencia", ni una "teoría", ni una "disciplina", es una metodología que hace posible la unificación y organización del conocimiento científico.

Esta metodología descansa en el concepto de sistema, considerando que en cualquier campo del saber es posible aplicar el concepto de sistema y que por lo tanto, las propiedades generales de los sistemas se pueden aplicar en las distintas áreas (educación, gestión, política, ...).

El método clásico de explicación de una realidad consiste en aplicar de forma sistemática el criterio de simplificación, el cual implica un división y una reducción. En el paso de división se aíslan los objetos, no sólo unos en relación a otros, sino que también de su entorno y del observador. En la reducción se busca lo elemental o común a una clase variada de problemas, unificando lo diverso y múltiple.

La metodología anterior no se puede aplicar para el estudio de problemas donde la organización o estructura sea tan importante como las partes, esto es, donde el todo sea más que la suma de las partes. En estos casos, la sistémica aporta el mecanismo de estudio necesario. Desde el punto de vista sistémico la realidad se debe explicar utilizando el análisis de la misma, esto es, dividir y estudiar las partes, pero considerando además las interconexiones de esas partes. De esta forma se tiene en cuenta tanto las partes como el todo, sin que las partes impidan considerar el todo, ni el todo impida analizar las partes.

Para terminar de definir la sistémica vamos a presentarla con respecto a otras metodologías con las que habitualmente se confunden:

* La metodología sistémica va más allá de la Cibernética, cuyo principal objetivo es el estudio de control de los organismos vivos y de las máquinas.

* Se debe distinguir de la Teoría General de Sistemas cuyo propósito es describir en lenguaje matemático la totalidad de los sistemas de la naturaleza.

* Los métodos analíticos son complementarios a los sistémicos, debido que la analítica reduce un sistema a sus componentes más elementales para estudiar y entender los tipos de interacciones que hay entre ellos. El análisis de sistemas modifica una de las variables del sistema para determinar las leyes generales que gobiernan su comportamiento. Para ello, se deben poder contemplar la unión de todas las propiedades elementales que aparecen en el sistema. En el caso de sistemas con elementos similares y ligeras interrelaciones entre ellos esta metodología es posible. Pero en sistemas complejos no se puede aplicar dicha metodología y se debe estudiar el sistema globalmente. La simulación nos permite observar en tiempo real los efectos de las diferentes relaciones que hay entre los componentes del sistema y poder diseñar reglas que puedan modificar el mismo.

* Sistémica no es lo mismo que metodología sistemática, en la que la resolución de un problema se obtiene a través de una serie de acciones secuenciales.

Pero la mejor forma de comprender la importancia y el impacto que tiene la sistémica es conocer su nacimiento y desarrollo a lo largo del tiempo.

2.1.-Teoría Clásica del Control

Desde épocas muy tempranas se conciben una serie de inventos que se enmarcan dentro del control automático. De todos ellos, el regulador centrífugo de James Watt (1769) es el más famoso. Este se utilizaba para mantener constante la velocidad de giro de las máquinas de vapor, disminuyendo la presión de la caldera cuando la velocidad aumentaba. Asimismo, los servomecanismos (1868) también tuvieron gran importancia al ser utilizados, inicialmente, para gobernar el timón de grandes barcos.

Todos estos mecanismos de control tenían en común la utilización de una realimentación negativa. Aunque el concepto de realimentación negativa es descubierto en 1927 por H. S. Black, que estudiando un circuito amplificador determina que la amplificación de la diferencia entre la señal de entrada y una fracción de la de salida es mucho mejor que si solamente se amplifica la señal de entrada [Fortmann 1977].

2.2.- Cibernética

Norbert Wiener, profesor de matemáticas en el Instituto Tecnológico de Massachusetts (MIT), en 1940 trabajando con el ingeniero Bigelow en la automatización de un cañón antiaéreo, se pregunta si se pueden encontrar en el hombre los mismos mecanismos de realimentación que hay en los sistemas automáticos.

Después de consultar con el neurofisiólogo Rosenblueth, Wiener dedujo que para poder controlar una acción finalizada (una acción con un propósito) la circulación de la información necesaria para tal control debe formar un lazo cerrado que permita la

evaluación de los efectos de las acciones y la adaptación de la conducta futura en base al comportamiento anterior. Esta cadena causal circular se presenta tanto en el sistema de los cañones antiaéreos como en el sistema nervioso, que ordena a los músculos realizar un movimiento cuyos efectos son entonces detectados por medio de los sentidos que envían esta información realimentada al cerebro.

Wiener y Bigelow generalizan el lazo cerrado de información necesario para corregir cualquier acción (realimentación negativa) a los organismos vivos. Esto es, presenta la analogía entre los dispositivos autogobernados y los procesos que tienen lugar en los seres vivos, tales como coger un lápiz o un vaso, en los que la información obtenida por medio de los ojos, es procesada por el sistema nervioso para gobernar el movimiento de la mano, de modo que alcance su objetivo: coger el vaso.

A partir de este momento se produce una formulación en términos conceptuales de lo que más tarde Wiener denominaría Cibernética [Wiener 1965]. De este modo la cibernética surge de un contacto enriquecedor entre especialistas de distintas disciplinas que observan cómo en sus respectivos campos de estudio se presentan sistemas que, aunque de naturaleza física completamente diferente, presentan modos de comportamiento análogos. Esta analogía entre los modos de comportamiento se explica por el descubrimiento de que en todos estos tipos de sistemas subyace una estructura común: la estructura de realimentación. Este es un ejemplo paradigmático del modo de proceder del movimiento sistémico.

3.- SISTEMAS

El término sistema se ha utilizado con frecuencia en los apartados anteriores, pero ¿qué es un sistema?. Una definición general podría ser:

" Un sistema es una combinación de elementos o componentes interrelacionados formando un todo, y que actúan de manera conjunta para la obtención de un objetivo común." [Matko 1992]

Los componentes de un sistema pueden ser los elementos físicos que lo forman, o las partes funcionales de dichos elementos.

El término interrelación juega un papel muy importante en la definición de sistema porque indica que este puede estar formado por uno o más sistemas, los cuales a su vez están compuestos por otros subsistemas, y así se da una definición recursiva del mismo.

Además, el sistema se puede considerar conceptualmente como una parte del universo con el que interactúa. A los factores externos al sistema que son capaces de causar un cambio en el mismo se denomina entorno del sistema.

Se define estado de un sistema como la colección mínima de información con la cual se puede predecir unívocamente el comportamiento futuro del mismo en ausencia de hechos no esperados.

Cualquier proceso que cambia los atributos (parámetros o variables) de un sistema se llama actividad. El sistema puede cambiar en respuesta a actividades internas o actividades externas al mismo. Las actividades externas al sistema se llaman exógenas, mientras que las internas se denominan endógenas. Aunque es conveniente distinguir en un sistema estos dos tipos de actividades, en muchos casos no es posible debido a que al definir el sistema no queda siempre claro cuáles son los factores externos y cuáles los internos. Además, una actividad exógena puede originar una serie de actividades endógenas, y viceversa.

El sistema interactúa con su entorno por medio de las entradas y las salidas. Las entradas tienen su origen fuera del sistema y es el medio por el cual el entorno del mismo es capaz de producir un cambio en dicho sistema. Las salidas, por otro lado, son generadas dentro del sistema para interactuar con su entorno. Se considera perturbación a una señal que tiende a afectar negativamente el valor de salida del sistema. Si la perturbación se genera dentro del sistema se denomina interna, mientras que una perturbación externa se genera fuera del sistema y constituye una entrada.

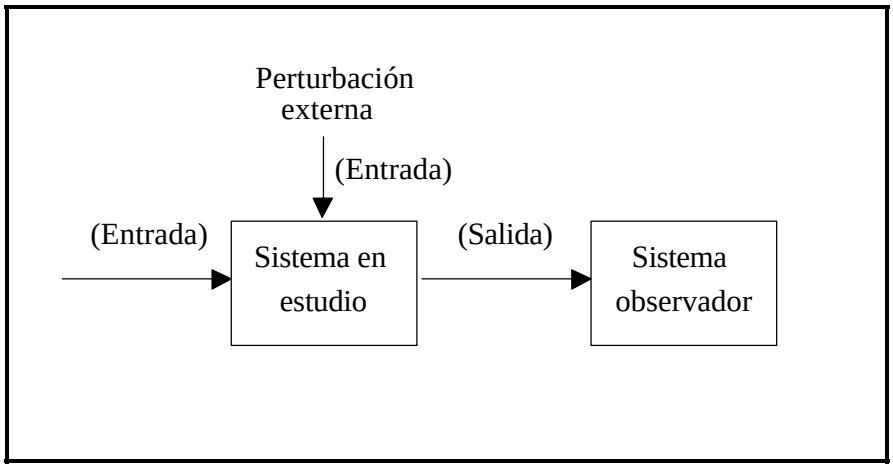


Figura 1.1.- Estructura de entrada y salida de un sistema.

La estructura de entrada y salida (figura 1.1) de un sistema influye en posibilitar el control del mismo. Existen dos conceptos duales que caracterizan las implicaciones dinámicas de la estructura de entrada/salida de los sistemas, estos son la observabilidad y la controlabilidad [Luenberger 1979].

Se dice que un sistema es completamente observable, si dada la salida del sistema en un instante dado se puede determinar el estado inicial (estado del que partió) del mismo.

Un sistema es completamente controlable si para cualquier estado inicial x_0 determinado y para cualquier estado final x_1 deseado, existe una secuencia de entrada que aplicada al sistema lleva a éste del estado inicial x_0 al estado final x_1.

Cuando se realiza una operación de medida sobre las propiedades o patrones de un mismo sistema bajo diferentes condiciones (diferentes instantes de tiempo, diferentes localidades espaciales, ...) y se obtienen distintos valores, se denomina a la propiedad observada, variable del sistema.

En el caso particular de que la medida produzca el mismo valor para todas las condiciones posibles, la propiedad en estudio se denomina parámetro del sistema. Por ejemplo, la masa del péndulo es un parámetro porque su valor permanece constante independientemente de las condiciones de medida [Delaney 1989].

3.1.- Subsistemas

Hay dos aspectos fundamentales que se desprenden de la definición de sistema: la unicidad y la descomposición recursiva.

a) Unicidad.- El sistema es un ente unitario. Esto es, cuando consideramos un sistema tendremos al sistema en sí mismo y al entorno que lo rodea. En este esquema conceptual el sistema se analiza como un todo y no como la unión de partes.

La naturaleza unitaria de los sistemas puede ser clarificada en términos de la relación de invarianza. Un fenómeno se dice que es invariante respecto a otro si permanece invariable mientras el otro cambia. Las relaciones invariantes evidencian los siguientes hechos:

a) La invarianza está limitada a cierto dominio del fenómeno que cambia. Un diamante se puede romper debido a una gran fuerza, sólo es invariante durante un periodo de tiempo, esto es, la invarianza se restringe a cierto dominio del periodo cambiante (tiempo).

b) La invarianza depende del nivel de realidad de los fenómenos considerados, sobre todo de la escala espacio/tiempo. Por ejemplo, la isotropía y homogeneidad de la superficie del océano son invariantes a cierta escala temporal/espacial, pero no a cierta distancia más o menos cercana o considerando instantes más próximos en el tiempo.

c) La invarianza puede estar asociada con la composición y estructura física de un sistema, esto es, los aspectos estáticos del mismo que nos indican lo que es el sistema (un diamante, un edificio, ...).

d) La invarianza puede estar asociada a lo que el sistema hace, independientemente de lo que es (el péndulo oscila, la fábrica hace coches, ...). Estas invarianzas son de naturaleza dinámica o funcional, ya que están asociadas con los patrones de comportamiento del sistema.

Un sistema (su unicidad y existencia como una entidad específica) es reconocible puesto que exhibe ciertas propiedades características o patrones de comportamientos que son invariantes, estas invarianzas son, además, características del sistema completo y no de sus partes.

b) Descomposición recursiva.- Cada uno de los componentes que forman el sistema pueden ser a su vez sistemas, llamados entonces subsistemas (el término componente es usado frecuentemente como sinónimo de subsistema). Estos tienen todas las características de un sistema, entre ellas la de estar formando por subsistemas. Por esto se dice que la definición de un sistema es recursiva. Desde esta perspectiva un sistema es una jerarquía de subsistemas.

Al ser sistemas, los subsistemas deben poseer características invariantes. Sin embargo, cuando forman parte del sistema total, las invariantes de un subsistema quedan sacrificadas en favor de las invariantes del sistema total. Por ejemplo, el patrón de comportamiento de una persona como parte de un equipo de fútbol o de una orquesta será diferente del que ofrece aisladamente en otras ocasiones.

La descomposición de los sistemas es importante debido a que ofrece las bases para entenderlos (explicar sus invariantes, cómo funcionan, ...) en término de sus subsistemas y de sus interrelaciones.

La descomposición puede ser de naturaleza estructural, en cuyo caso la propia naturaleza física del sistema como un objeto compuesto de más objetos elementales y relaciones estructurales entre ellos, establece la descomposición en subsistemas. Por ejemplo, en un edificio cada planta es un subsistema estructural del mismo.

Otro tipo de descomposición en los sistemas es la funcional, en cuyo caso la descomposición en subsistemas se hace atendiendo a criterios funcionales, esto es, se realiza una descomposición en relación a la actividad característica de cada subsistema [Delay 1989].

3.1.1.- Relaciones entre los Subsistemas

Los distintos subsistemas que forman un sistema presentan distintas relaciones entre ellos para determinar las características del mismo.

Las relaciones que se pueden presentar son de tipo estructural, funcional o de realimentación.

a) Relaciones estructurales.-

Son las relaciones existentes entre las distintas partes estructurales de un sistema. Estas relaciones son de tipo jerárquicas, que establecen un ordenamiento entre un sistema y sus subsistemas; y relaciones de tipo espacial, que especifica las posiciones relativas (orden espacial) de los distintos componentes del sistema.

b) Relaciones funcionales.-

Relaciones que implican un ordenamiento del tiempo. Entre ellas se encuentran las relaciones temporales, que generan un ordenamiento en el tiempo pero no implican necesariamente causalidad. Un ejemplo de este tipo de relación sería el tiempo de llegada de dos pacientes a un hospital, ya que involucra un orden temporal pero no una causalidad (la llegada de un paciente no es consecuencia de la llegada de otro).

Otro tipo de relación funcional es la actividad, ya que genera una relación causa-efecto entre su entrada y estado anterior (factores causales) y su nuevo estado y salida (como efecto).

Una interacción es una relación causa-efecto, en la que se involucra una salida (efecto) de un subsistema que es un factor causal (entrada) en otro subsistema.

Tanto las actividades como las interacciones implican orden temporal y causalidad.

c) Relaciones de realimentación.-

La relación de realimentación es la que se establece cuando la entrada de un sistema depende (indirectamente, a través de la intervención de otro sistema) de su propia salida anterior.

La realimentación se dice que es positiva cuando la nueva entrada genera en el sistema una transformación en el mismo sentido que los resultados anteriores, esto es, sus efectos son acumulativos, lo que origina que el sistema tenga un comportamiento divergente. Ejemplos de realimentaciones positivas son: una reacción en cadena, expansión industrial, interés compuesto del capital invertido, proliferación de las células de cáncer, ...

Hablamos de realimentación negativa cuando la nueva entrada produce un resultado en la dirección opuesta a los resultados previos, con lo cual se mantiene el sistema en equilibrio. Ejemplos de este tipo de realimentaciones son: concentración de glucosa en sangre, servomecanismos, termostato, ...

Principio de causalidad: Mediante la relación de realimentación el estado de un sistema puede influenciar sus estados posteriores. Sin embargo, el principio de causalidad nos dice que la cadena de causas y efectos tienen valor durante un tiempo finito (básicamente porque las actividades que ocurren en el sistema consumen tiempo). Esto implica que el estado en cierto instante de tiempo puede únicamente depender en el sentido causal de sus estados anteriores.

3.2.- Clasificación de los Sistemas

Hay muchas formas de clasificar sistemas. Una clasificación obvia atendiendo a la naturaleza de los mismos distingue entre los sistemas naturales (un árbol, un caballo, ...) y los sistemas hechos por el hombre o sistemas artificiales (un coche, un libro, ...).

Otras clasificaciones que pueden ser usadas presentan los sistemas continuos frente a los discretos, los deterministas frente a los estocásticos, o los sistemas abiertos frente a los cerrados.

a) En base a su evolución con el tiempo existen los Sistemas Continuos y los Sistemas Discretos

Los términos continuo y discreto son aplicados a un sistema para hacer referencia a la naturaleza o comportamiento de los cambios del estado del sistema respecto al tiempo. Los sistemas que pueden cambiar su estado en cualquier instante de tiempo se les denominan continuos. Mientras que aquellos sistemas que sólo pueden cambiar su estado en cuantos finitos de tiempo, o a saltos, se les llaman sistemas discretos [Gordon 1978].

Atendiendo a la regularidad de los intervalos en los que se produce los cambios de estado en un sistema discreto nos podemos encontrar:

- Sistemas de tiempo discreto: los cambios de estado se producen en intervalos regulares de tiempo, por ejemplo un ordenador.
- Sistemas de eventos discretos: los cambios de estado ocurren en intervalos no regulares de tiempo, un ejemplo sería la llegada de pacientes a un hospital.

Los sistemas que presentan variables que pueden variar continuamente en el tiempo, y que tienen otras variables que varían discretamente en el tiempo son los sistemas híbridos. Un ejemplo sería una cola de automóviles en una autopista. Si consideramos cómo variable de interés el número de coches de la cola tendríamos un sistema discreto. Pero si la variable de interés es la distancia media entre automóviles, el sistema lo consideramos continuo. Si determinamos que las variables de interés para el estudio del sistema son ambas, el sistema sería híbrido.

b) Atendiendo a la causalidad están los Sistemas Estocásticos y los Sistemas Deterministas

Un sistema se dice determinista cuando su nuevo estado es completamente determinado por su estado anterior y por las actividades. Esto es, un sistema determinista tiene completamente definida la manera de pasar de un estado a otro como respuesta a una actividad dada.

En los sistemas estocásticos no se puede determinar el siguiente estado del sistema, dado el estado actual y un estímulo (actividad). Esto es, un sistema estocástico tiene aleatoriedad en la transición de un estado a otro. Esta aleatoriedad está producida por la presencia de una perturbación de tipo aleatorio, o porque el sistema presenta parámetros desconocidos o inciertos. El estudio de estos sistemas se tiene que realizar recurriendo a la teoría de la probabilidad.

c) Dependiendo del motivo o la causa de la evolución existen Sistemas Abiertos y Sistemas Cerrados

Un sistema cerrado es un sistema en el cual todos los cambios de estado son ocasionados por actividades endógenas.

Los sistemas abiertos son aquellos que cambian de estado en respuesta a tanto actividades exógenas como endógenas.

3.3.- Sistemas Complejos

La mayoría de las definiciones dadas para sistemas complejos [Waldrop 1992] nicamente en un dominio concreto por ejemplo en algoritmos de computación. En la búsqueda de una definición de sistema complejo comenzamos estudiando el significado del término complejidad.

Complejidad significa "entrelazado". Esta definición puede ser interpretada de la siguiente forma: para poder tener complejidad en un sistema se necesitan dos o más componentes, los cuales están unidos de tal forma que es difícil separarlos.

La definición que da el Diccionario Oxford dice que complejo es aquello que está hecho de partes (usualmente varias) estrechamente conectadas. Se observa por lo tanto la dualidad que existe entre las partes que son a la vez "distinción" y "conexión".

Las dos dimensiones que caracterizan la complejidad de los sistemas son la "distinción" y el "acoplamiento" de las partes. La distinción nos informa sobre la variedad, la heterogeneidad, el hecho de que distintas partes del sistema funcionen de forma diferente. La característica "distinción" nos lleva al límite del caos o a la entropía. Mientras que acoplamiento corresponde con las restricciones, el hecho de que las diferentes partes del sistema no son independientes, y que el conocimiento de una parte y sus interrelaciones permite determinar las características de las otras partes. Las interrelaciones nos conduce al orden, a la negentropía.

Por lo tanto, un sistema complejo se caracteriza por estar formado por numerosas partes o elementos que se interrelacionan y que muestran propiedades emergentes no exhibidas por los elementos individualmente. Debido a que esta definición es muy general, bajo esta disciplina se estudian problemas tan diversos como:

- Adaptación: fenómeno que ocurre cuando la estructura o función de un sistema cambia de forma que estos cambios mejoran su existencia en el entorno.

- Agentes: Los agentes son los elementos individuales de un sistema complejo que interactúan de muchas formas. Cada agente tiene su propio estado interno y sus reglas o estrategias que determinan su comportamiento en el entorno.

- Vida artificial: Alife (Artificial Life) es un nuevo campo de estudio donde se investigan los organismos vivos utilizando simulación computacional.

- Caos: La idea central del caos dice que pequeños cambios en las condiciones iniciales son amplificadas en el sistema de manera que tienen una gran influencia en el comportamiento futuro del mismo.

- Límite del caos: La complejidad existe cuando se producen dos fenómenos, el desorden (que se describe estadísticamente) y el orden (que se describe por métodos determinísticos). Por ello estos problemas se encuentran entre el orden y el desorden, o en los límites del caos.

- Algoritmos genéticos: Abstracciones computacionales de la evolución biológica.

- Auto-Organización: Fenómeno que ocurre cuando elementos en un sistema interactúan de forma que se están acomodando mutuamente para generar una nueva estructura a más alto nivel.

Sin embargo, en esta memoria nos queremos centrar en la idea de que sistema complejo es aquél formado por muchas partes interactuando entre sí. Intuitivamente un sistema será más complejo si tiene más partes que pueden distinguirse y si existen más conexiones entre ellas. Más partes a ser representadas implica modelos más extensos, los cuales requieren mucho tiempo de estudio y cómputo. Como los componentes del sistema no se pueden separar sin destruirlo, el método de análisis por descomposición

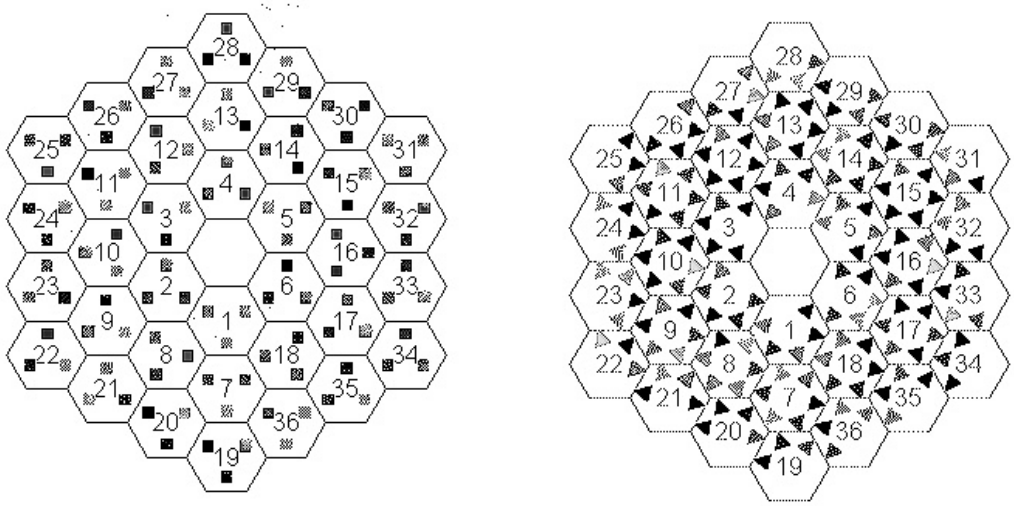
en módulos independientes no se puede usar para el estudio o simplificación de tales sistemas. Esto significa, que los sistemas complejos serán difíciles de modelar y los modelos resultantes son difíciles de usar para la predicción o el control, y que por lo tanto, esos sistemas son difíciles de resolver. Por ello se justifica la connotación de "dificultad" que se le asocia al término complejidad.

El proceso de incremento de la variedad puede llamarse diferenciación y al proceso de incremento en el número de conexiones se le denomina integración. La complejidad producida por diferenciación e integración en la dimensión espacial se conoce como complejidad estructural. Mientras que si esta complejidad se produce en la dimensión temporal se le denomina complejidad funcional.

El estudio de los sistemas complejos se realiza sobre las abstracciones que se realizan del mismo. Esto es, se crean clases de más alto nivel donde se agrupan fenómenos similares, permitiendo obviar las diferenciaciones que existen entre los elementos de esas clases. Dependiendo de las clases creadas se tendrá una variedad y dependencia mayor o menor, y determinará el grado de complejidad estudiado.

Se puede concluir que la complejidad se incrementa cuando la variedad (distinción) y la dependencia (acoplamiento) de las partes de un sistema se incrementan. A continuación veamos dos ejemplos de sistemas complejos, en el primer caso por estar formado por un gran número de componentes y en el segundo por incrementar las interrelaciones entre sus componentes.

A los sistemas que tienen un gran número de componentes también se les denomina sistemas de gran escala [Jamshidi 1983]. Un telescopio de gran tamaño sería un caso de sistemas de gran escala. Por ejemplo, el telescopio Keck (Hawai) o el Grantecan (Canarias). Ambos necesitan un espejo primario muy grande y la solución tecnológica ha consistido en que este espejo esté formado por 36 espejos hexagonales que actuarán a efectos ópticos como uno solo. Para ello se controla la posición de cada segmento por medio de 3 actuadores, y la posición relativa entre segmentos se mide mediante sensores de desplazamiento colocados en las transiciones entre segmentos. Con lo que se dispone de un total de 168 sensores y 108 actuadores como muestra la figura 1.2.



a) Los cuadrados representan los 108 actuadores b)Los triángulos representan los 168 sensores

Figura 1.2.- Espejo primario segmentado de un telescopio de 10 m. de diámetro

En el siguiente ejemplo se observa como el incremento de las relaciones de los componentes del sistema hace que la dinámica del mismo sea más compleja.

El sistema objeto de estudio está compuesto de una rejilla de N * N subsistemas interactuantes, que en su posición de equilibrio se encuentran equidistantes entre sí. Los subsistemas pueden considerarse partículas en dos dimensiones caracterizadas por su posición y velocidad. Las coordenadas de posición de la partícula i, son (x^i, y^i) y la velocidad viene representada por $(\frac{dx^i}{dt}, \frac{dy^i}{dt})$. Llamaremos x_i^i, a la coordenada x de la partícula a la izquierda de la partícula i, x_d^i a la coordenada x de la partícula a la derecha de la partícula i, y_u^i a la coordenada *y* de la partícula una fila por encima de la partícula i e y_d^i a la coordenada *y* de la partícula una fila por debajo de la partícula i. En las gráficas 1.3 y 1.4 se presentan la simulación del mismo sistema con distintos grados de acoplamiento:

a) Interacción sin acoplamiento. En este caso, cada partícula- subsistema Sij se descompone en 2 subsistemas Xij e Yij. El primero tiene como entradas las posiciones en el eje horizontal de las partículas inmediatamente a la izquierda y a la derecha, y como salida la coordenada x de Sij. La ecuación que modela al subsistema Xij es:

$$\frac{d^2x^i}{dt^2} = k(x_d^i + x_i^i) - 2kx^i$$

El subsistema Yij es afectado de manera análoga por las partículas de las filas superior e inferior, teniendo como salida la coordenada en el eje vertical de Sij. La dinámica del sistema cuando se produce una perturbación en el subsistema 3,3 se muestra en la figura 1.3.

b) Interacción con acoplamiento. La descomposición de Sij en dos subsistemas Xij e Yij también se produce en este caso, pero la interacción con los primeros vecinos de Sij es más compleja. Esto es debido a un término de acoplamiento que se introduce entre el conjunto de variables que determinan la posición de las partículas en el eje horizontal y las variables que representan las posiciones en el eje vertical.

En este caso el subsistema Xij recibe entradas procedentes no sólo de subsistemas X correspondientes a primeros vecinos en el eje horizontal, sino también de subsistemas Y correspondientes a primeros vecinos en el eje vertical. El mismo tipo de interacción es añadida en los subsistemas Yij de forma simétrica. Las ecuaciones que describen ambos subsistemas son:

$$\frac{d^2x^i}{dt^2} = k(x_d^i + x_i^i) - 2kx^i + k'(y_a^i - y_u^i) - 2k'Ln$$

$$\frac{d^2y^i}{dt^2} = k(y_u^i + y_a^i) - 2ky^i + k'(x_d^i - x_i^i) - 2k'Ln$$

Si producimos la misma perturbación que en el caso anterior (subsistema 3,3) se observa que la dinámica es mucho más compleja como consecuencia del incremento en el número de interrelaciones figura 1.4.

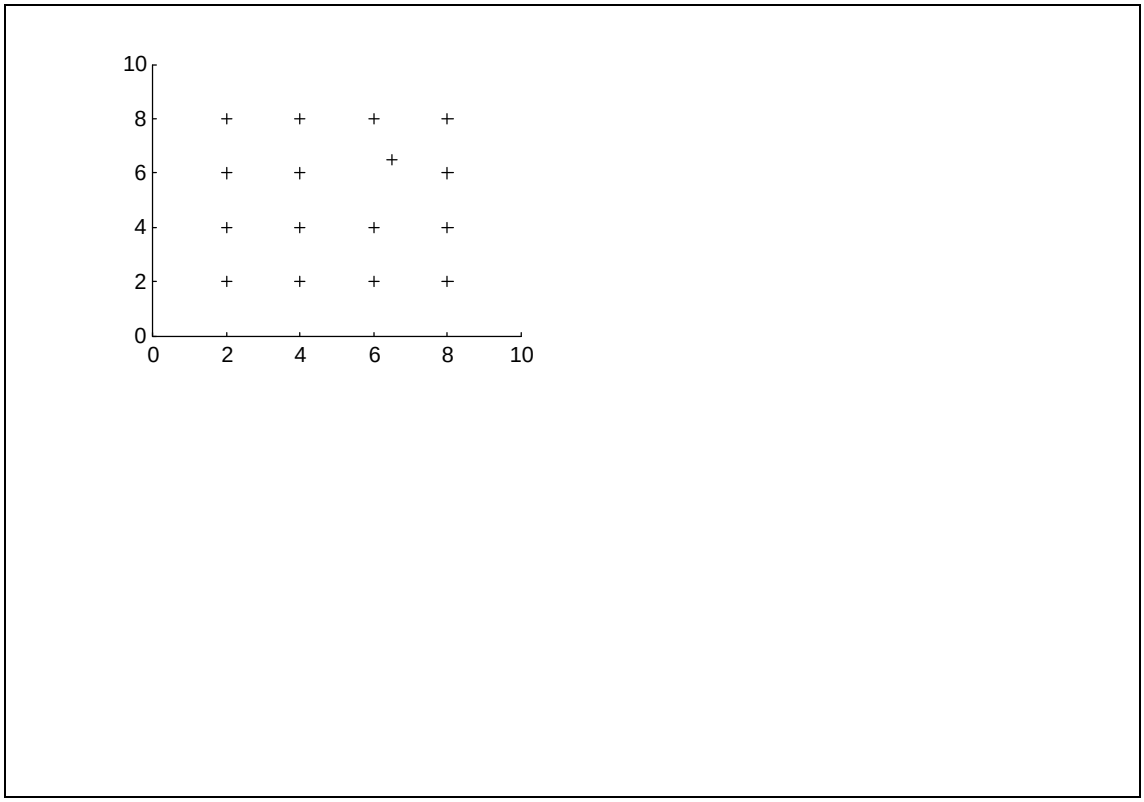


Figura 1.3.- Interacción sin acoplamiento de los elementos del sistema. Evolución del sistema cuando se produce una perturbación en el subsistema 3,3.

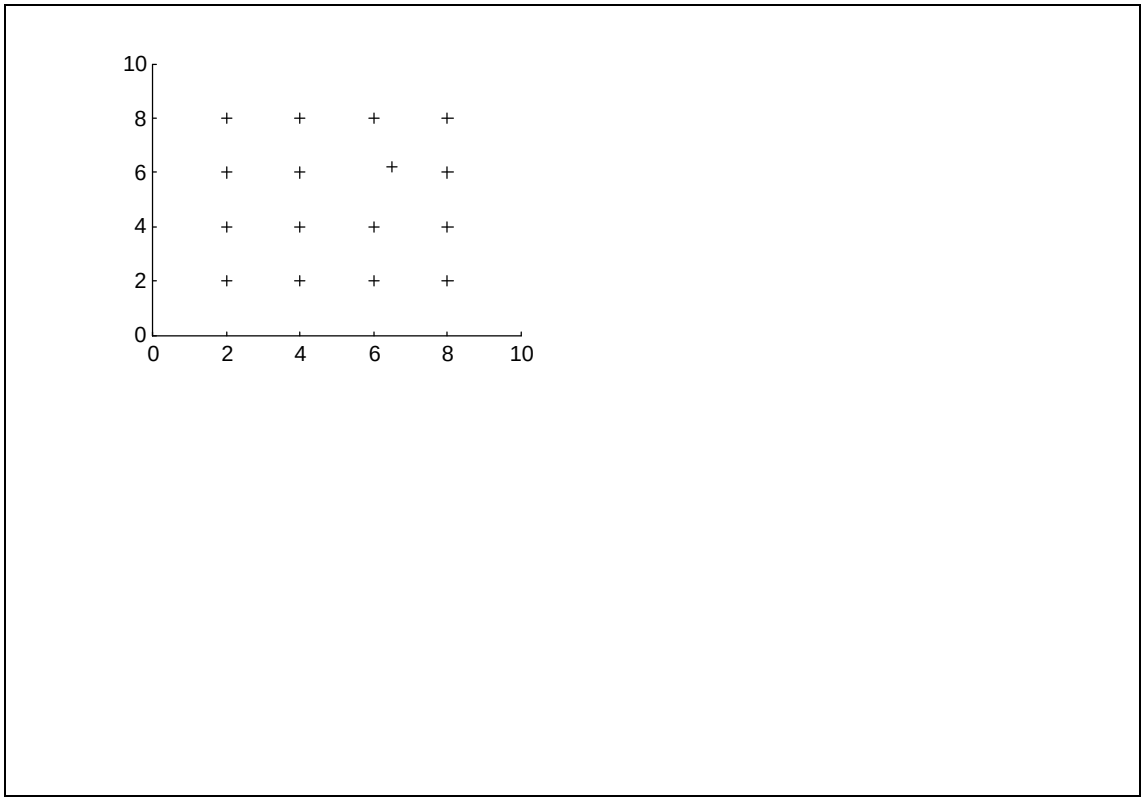


Figura 1.4.- Interacción con acoplamiento de los elementos del sistema. Evolución del sistema cuando se produce una perturbación en el subsistema 3,3.

Como en el caso de los ejemplo anteriores, si observamos un hospital encontramos un sistema formado por muchas partes (pacientes, recursos humanos, recursos materiales) que interactúan entre sí. Tanto el gran número de partes, como las muchas interrelaciones que existen entre ellas, hace que un hospital sea un sistema complejo y que por lo tanto sea difícil su estudio. Para esta tarea, en este trabajo, se han utilizado técnicas de Simulación y de Inteligencia Artificial que nos han permitido su análisis y la obtención de herramientas de ayuda a la toma de decisiones en la gerencia hospitalaria.

3.4.- Dinámica de Sistemas

El término dinámica se refiere al fenómeno que produce que determinados patrones cambien en el tiempo, por lo tanto, la dinámica de sistemas estudia la evolución en el tiempo de los sistemas. Esta evolución viene especificada por las interrelaciones entre las partes del sistema [Aracil 1986a].

Muchos de los sistemas que nos rodean tienen importantes características dinámicas, por ejemplo: el comportamiento de la economía de un país, la evolución de la población de una especie, ... Todos estos ejemplos ilustran situaciones dinámicas e indican la importancia de mecanismos que permiten representar y analizar comportamientos dinámicos. Las matemáticas nos pueden ofrecer un lenguaje formal y un marco teórico para el estudio de estos sistemas.

Cuando se estudia un sistema se puede obtener una colección de observaciones cuantitativas de atributos o propiedades del mismo. Si se considera el comportamiento dinámico, entonces estos datos se encuentran parametrizados con relación al tiempo debido a que el carácter dinámico del sistema considera primordial su evolución temporal. En esta evolución las variaciones que se producen en el sistema son consecuencia, fundamentalmente, de las propias interacciones que constituyen la estructura del sistema. Así que desde el punto de vista de la dinámica, el comportamiento dinámico de un sistema está determinado por su estructura. Esta estructura tiene una importancia mayor en la evolución del mismo que la naturaleza de cada uno de los elementos individuales que lo componen.

Un primer paso en el estudio dinámico de los sistemas es realizar un diagrama causal de los mismos. Este diagrama permite conocer la estructura del sistema, que viene dada por la especificación de los elementos que aparecen, y por el establecimiento de la existencia o no de una relación entre cada par de elementos, para posteriormente estudiar la naturaleza de esta relación.

Los sistemas socioeconómicos son un ejemplo de un sistema dinámico, en los que, al contrario de lo que ocurre con los procesos tecnológicos, se desconocen o no se pueden formular matemáticamente las leyes que rigen las interacciones elementales que se producen en el seno de los mismos. Sin embargo, hay que constatar que todas las organizaciones sociales o económicas muestran un comportamiento dinámico y una fuerte interacción entre sus partes. Es decir, según pasa el tiempo, las variables con que se mide su estado, tales como ventas, habitantes, producción, etc... fluctúan considerablemente, como consecuencia de las interacciones que se producen entre ellas. Como en el caso de un hospital, donde las relaciones que se producen entre los pacientes, los recursos materiales y los recursos humanos generan una producción determinada que determina el estado de salud de una población.

4.- MODELADO

Se entiende por modelo una representación de un determinado aspecto de la realidad de un sistema, en un lenguaje específico. El modelo es una representación de algo (material o abstracto, real o normativo) a lo que está ligado por una relación asimétrica, si M es un modelo de X, X no tiene por qué serlo de M [Aracil 1986b].

El objetivo de un modelo es reproducir algunas propiedades del sistema original, pero no todas; en este último caso se tendría el propio sistema real. Todo modelo es selectivo, y no existe ninguno que pretenda ser perfectamente fiel a la realidad que modela. Su construcción se hace adoptando exclusivamente aquellos aspectos que son relevantes, en función del uso que se pretende hacer del modelo.

La definición más simple de un modelo se puede enunciar diciendo: un objeto M es un modelo de X, para un observador O, si O puede emplear M para responder a cuestiones que le interesan acerca de X. En esta definición no sólo se considera el objeto real X, sino que también está involucrado el propio observador O que decide aquellas propiedades o manifestaciones de X que son relevantes y deben figurar en el modelo M [Aracil 1986b].

Para construir un modelo es necesario especificar los elementos (entidades con atributos y que realizan actividades) que lo componen, esto es, su composición y su entorno, y las relaciones entre ellos, su estructura.

Los componentes que forman el modelo son los elementos considerados relevantes para caracterizar y describir el sistema bajo estudio. El modelo presupone una reducción de la complejidad de la realidad a unos niveles manejables, así pues, el modelo es siempre más simple, y cualitativamente más claro, que la propia realidad, aunque en la simplificación se pierdan algunos aspectos de ella.

Una vez decididos los elementos que forman parte del modelo se procede a establecer las relaciones entre ellos. Las matemáticas nos ofrecen en algunos casos herramientas que nos permiten definir estas relaciones, por ejemplo, a través de las ecuaciones diferenciales cuando la relación es de tipo dinámico. Pero en la mayoría de los casos, estas relaciones no pueden ser definidas matemáticamente y hay que hacer uso de otras herramientas tales como gráficos, algoritmos, lenguajes, etc.

4.1.- Tipos de Modelos

Los modelos pueden ser divididos en muchos tipos [Fishwick 1995]. A continuación vamos a dar una de las posibles clasificaciones:

a) Modelos Físicos

Los modelos físicos son representaciones de sistemas físicos; cuya construcción en muchos casos es cara, consume gran cantidad de tiempo o son imposibles de construir.

Los modelos físicos que tienen carácter estático pueden ser:

modelos a escala: modelos de tamaño reducido de coches, edificios, barcos, etc...

modelos de imitación: estructura de moléculas, muñecas, dibujos, etc...

Los modelos que presentan características estáticas se pueden dividir en:

Modelos análogos: los sistemas modelados son representados con la ayuda de la correspondiente analogía, la cual es adecuada por alguna razón. Por ejemplo, las ratas y los monos representan una clase de modelo análogo a los humanos en una prueba de una nueva droga.

Prototipos: copias de tamaño reducido del sistema real, por ejemplo, laboratorios y plantas pilotos de diferentes procesos industriales, sistemas de autopistas en miniatura, modelos de coches para probar sus características aerodinámicas en un túnel de viento, etc...

b) Modelos Mentales

Los modelos mentales tienen carácter heurístico e intuitivo y existen únicamente en la mente humana. Este tipo de modelo suele ser incierto, impreciso y con problemas para comunicarlo. Los modelos mentales suelen representar la acumulación de experiencia de una persona en procesos de planificación y toma de decisiones. Puntos de vistas personales de un objeto o un evento son causados por un modelo mental, también la habilidad humana de interpretar operaciones, etc...

c) Modelos Matemáticos

Los modelos matemáticos son menos problemáticos de manipular y construir que un modelo físico.

Dentro de los modelos matemáticos están los que representan magnitudes físicas y sus relaciones, modelos matemáticos propiamente dichos. Y los modelos que representan conceptos (entidades) y sus relaciones, que llamamos modelos simbólicos.

Los modelos simbólicos pueden ser lingüísticos (descripciones verbales o escritas de eventos, experiencias, escenas, etc...), gráficos (pinturas, dibujos, gráficos, etc...), esquemáticos (diagrama de flujo, mapas, diagrama de redes, etc...) o algorítmicos

(conjunto finito y ordenado de operaciones cuya aplicación permite resolver un problema). Estos modelos tienen la propiedad de que frecuentemente es bastante complejo obtener información precisa de ellos, especialmente los modelos expresados verbalmente.

Por muchas razones los modelos denominados matemáticos son los más importantes y los más utilizados de todos los modelos. Son concisos, no ambiguos y unívocamente interpretables, además su manipulación y la evaluación de alternativas es relativamente barata. Un modelo matemático puede ser definido como la traslación de las relaciones entre las variables físicas del sistema a ser modelado en la correspondiente estructura matemática. Cuando estas relaciones son establecidas para el estado estacionario únicamente, los modelos tienen carácter estático y se representan con ecuaciones algebraicas. Por otro lado, un modelo matemático dinámico representa el transitorio, así como el comportamiento en estado estacionario del sistema. Se describe utilizando sistemas de ecuaciones diferenciales y un conjunto de restricciones.

Aunque los modelos matemáticos son los preferidos para estudiar los sistemas de cualquier área de la ciencia (medicina, química, biología, sociología, economía, ...), este tipo de modelo es escaso y en la mayoría de los sistemas en estudio es muy difícil encontrar la estructura matemática que permita el modelado del sistema. Es por ello, por lo que debemos recurrir a utilizar un modelo simbólico.

5.- SIMULACIÓN

Para conocer lo que es una simulación vamos a dar algunas definiciones que nos mostrarán los aspectos importantes de la misma:

- Simulación es la técnica de construir y ejecutar un modelo de un sistema real para estudiar su comportamiento sin perturbar el entorno de dicho sistema real.
- La simulación de procesos dinámicos es el método iterativo capaz de estudiar las propiedades de un sistema mediante la experimentación con el modelo correspondiente de la planta real.
- Simulación es el proceso de imitar los aspectos importantes del comportamiento de

un sistema en tiempo real, comprimiendo o expandiendo el tiempo mediante la construcción y experimentación de un modelo del sistema real.

- En comparación con los métodos analíticos, la simulación es más realista y más fácil de entender, pero únicamente cuando se usa correctamente. Por contra, sólo se puede experimentar con casos particulares.
- La simulación por computador significa ejecutar un programa especial sobre un computador, el cual genera la respuesta temporal del modelo que imita el comportamiento del proceso a estudiar.
- La simulación es el proceso de resolver ecuaciones diferenciales mediante integración.

En todas las definiciones anteriores aparecen íntimamente ligados los términos modelo y simulación, incluso muchas veces estos términos se utilizan como sinónimos. Aunque su significado está relacionado, modelado y simulación no son exactamente lo mismo. Los modelos son los elementos esenciales o característicos de una simulación. La simulación es simplemente el uso de un modelo para imitar el comportamiento de los sistemas y permitir estudiar sus rendimientos bajo una variedad de circunstancias. La simulación es frecuentemente utilizada para determinar algunos aspectos del sistema. Por ejemplo, si quisiéramos saber cómo el número de trabajadores en un banco afecta al rendimiento del mismo. El primer paso sería la construcción del modelo que replique la llegada y atención de clientes. Este modelo podría usar variables aleatorias para generar cantidades tales como número de clientes, tiempo de llegada de los clientes, tiempo de atención a los clientes, etc. El siguiente paso sería hacer evolucionar el modelo, esto es, realizar la simulación del proceso. A continuación se estudian los datos obtenidos acerca de los clientes que se han atendido y se determinan cuántos trabajadores necesitamos para tener máximo rendimiento (mayor número de cliente atendidos al menor coste posible, esto es, con el menor número de trabajadores).

Una de las clasificaciones que se puede realizar de las simulaciones es en base al uso final que tendrá la misma (cuando la simulación tenga más de un uso se considerará el uso dominante). Atendiendo a esta clasificación tenemos los 5 tipos siguientes:

a) Investigación y desarrollo: son las que se utilizan para el diseño y desarrollo de

sistemas nuevos.

b) Testeo y evaluación: simulaciones que se usan para complementar los testeos sobre los nuevos sistemas. La principal ventaja que ofrece el testeo por simulación es que los sistemas son sometido a ciertos factores externos y situaciones, a los que no se les podría someter en la realidad por razones técnicas, de costo, de seguridad, etc.

c) Producción y logística: Ayudan en la determinación de requerimientos, en la valoración de sistemas de producción, distribución de recursos, etc, facilitando la toma de decisiones tanto administrativas como operativas.

d) Educación y entrenamiento: se utilizan en clase, en seminarios o en sesiones de entrenamiento. Permiten al alumno entender el uso de distintos procedimientos, técnicas y tácticas, así como facilitan el desarrollo de habilidades.

5.1.- Simulación Computacional

La simulación también la relacionan frecuentemente con los computadores, y así se observa en las definiciones dadas anteriormente. En los últimos años la simulación computacional se ha convertido en una de las principales técnicas en el estudio de sistemas complejos. Esto es debido a que los ordenadores son cada vez más baratos y con mayores prestaciones, así como la mejora en las herramientas para simular por computador, que han hecho que se puedan estudiar sistemas complejos. Los sistemas de gran escala que no podían ser resueltos con esfuerzos razonables, debido a que las técnicas convencionales de modelado, análisis, control y diseño fallaban; son ahora estudiados con ayuda de la simulación computacional [McHaney 1991].

Si consideramos la simulación como el uso de un modelo para extraer conclusiones sobre el comportamiento de cualquier elemento del mundo real, la simulación por computadora exige que el modelo sea representado mediante un programa de ordenador.

En muchos casos, el proceso de describir sistemas complejos del mundo real utilizando modelos analíticos puede ser una tarea difícil e incluso imposible. Es en estos casos cuando se hace necesario la simulación computacional para imitar o simular las operaciones de los procesos del mundo real mediante un programa (modelo simbólico). Este modelo puede ser manipulado para ayudar en el análisis del comportamiento dinámico del sistema. Los resultados del programa de simulación son evaluados numéricamente sobre un periodo de tiempo, y se utilizan estos datos para estimar las características del sistema. Como en cualquier experimento, el estudio de los datos se hace estadísticamente.

5.2.- Simuladores

Aunque los términos modelo y simulación son algunas veces usados como sinónimos, los términos simulador y simulación no se deben usar de la misma manera pues su significado no es equivalente. Un simulador es algo que simula, especialmente un dispositivo que genera un test sobre operaciones. Los simuladores son usados frecuentemente para el entrenamiento de profesionales, es este el caso de los simuladores de vuelos que sirven para entrenar a los futuros pilotos.

También se entiende por simuladores a los paquetes software que facilitan a los directivos de las empresas, analizar problemas y formular soluciones para un amplio espectro de aplicaciones. Estas herramientas permiten al usuario, que no necesita ser un ingeniero de sistemas, representar las operaciones del mundo real de manera muy intuitiva. Las principales características que poseen este tipo de software son: no se tiene que programar, posee una interface gráfica que facilita la construcción del modelo y se puede obtener el modelo en pocas horas de desarrollo. Ejemplos de simuladores son: Witness, MicroSaint, etc.

Este tipo de software utilizado en el contexto adecuado puede ser una herramienta de ayuda, pero cuando se utiliza mal puede dar malos resultados debido a:

1.- Gran simplificación: Los simuladores tienden a simplificar demasiado la representación del sistema. Esto es debido a que es un software que sirve para aplicar a

una gran variedad de sistemas por lo que debe ser muy general y no permite la representación de los hechos concretos de nuestro sistema.

2.- Representación del sistema de forma inflexible: debido a que la base del modelo ya ha sido construida y de una forma muy general, que no se puede cambiar.

3.- Fomenta el salto de etapas en la construcción de los modelos: debido a la naturaleza del simulador permite construcción fácil de los modelos; el usuario tiende a saltarse los pasos intermedios en la construcción de los mismos y realizar un análisis rápido que lleva a errores.

Concluimos que utilizar un simulador no automatiza todo el proceso de simulación, únicamente facilita la construcción del modelo, pero este debe ser, posteriormente, analizado con las técnicas adecuadas para su validación y verificación [McHaney 1991].

5.3.- Animación

Se define como animación el uso de gráficos en un ordenador para representar los elementos simulados y sus actividades. La animación puede ejecutarse en tiempo real o en modo de post-procesamiento. El término de tiempo real es utilizado para indicar que la animación corre en la pantalla del ordenador mientras el programa de simulación se está ejecutando. Esto es, los resultados de la simulación se observan en el momento en el que son obtenidos. La desventaja de esta animación es el gasto de tiempo necesario en la ejecución así como que siempre que se corra el programa de simulación se debe estar viendo la animación. En el modo de post-procesamiento la animación es mostrada después de que la ejecución del programa de simulación ha terminado. Para ello, la simulación debe crear un fichero de datos que es utilizada como entrada al programa de animación. Este mecanismo presenta las ventajas de poder visualizar hacia delante, hacia atrás o cambiar la velocidad en que se muestra la animación. Además la animación puede ser fácilmente transportada sin el programa de simulación, requiriendo únicamente el programa de animación y el fichero de datos.

Debido al incremento del uso de los ordenadores, así como el amplio uso de gráficos en muchas aplicaciones de áreas diferentes se ha ocasionado un fuerte incremento de esta parte de la simulación.

Las razones que han ocasionado el uso de la animación son las siguientes:

- Las salidas producidas por la simulación se pueden presentar de una forma amigable. Observar como las entidades simuladas evolucionan en la pantalla facilitan una mejor entendimiento de la operación del sistema que si se observa una salida estática.

- Facilita la validación porque el experto puede visualmente examinar el modelo.

- Se puede utilizar como una herramienta de depuración ya que inconsistencias en el modelo se pueden detectar más fácilmente con los gráficos evolucionando en la pantalla que con estadísticas.

- Una de las más populares razones para utilizar la animación es el impacto visual o el sentido de realidad que genera en un proceso de modelado. Es mucho más fácil aceptar un sistema que puede ser visto en la pantalla de un ordenador que aquel que está representado por un pila de documentos con datos estadísticos.

Pero si se utiliza la animación de forma no correcta puede llevarnos a errores. La animación se debería usar como un suplemento en cualquier estudio de simulación pero no dirigir este estudio ("Observar dibujos sobre una pantalla no puede sustituir a un buen trabajo estadístico", J.O. Henriksen). Los errores en los que se puede caer cuando se utiliza la animación son los siguientes:

- Genera una sobre confianza sobre el sistema simulado, esto es erróneo porque un modelo es solamente tan preciso como lo sean los datos y las presunciones de las que se parte. La animación hace que el sistema parezca tan realista en la pantalla que ocasiona que los usuarios se olviden de las presunciones hechas y de las técnicas utilizadas en el desarrollo del modelo, considerando que el sistema está modelado adecuadamente.

- Excesivo gasto de tiempo en la construcción de una bonita animación, restando por lo tanto tiempo en la construcción del modelo. La animación debe ser utilizada

como suplemento a la simulación y no como punto central.

- Se sustituye el trabajo estadístico por la animación. Se toman decisiones en base a la animación observada durante un corto periodo de tiempo, mientras que lo correcto seria realizar un adecuado análisis sobre toda la simulación aunque ello necesite más tiempo. La porción de simulación observada en la animación puede no ser representativa del verdadero comportamiento del sistema.

5.4.- Etapas en el Proceso de Simulación

Para realizar una simulación tenemos que llevar a cabo las siguientes las siguientes tres fases [Delaney 1989]:

1.- Planificación o premodelado.-

Es la fase inicial en la resolución del problema. En esta etapa se realiza un análisis del sistema para determinar exactamente qué tipo de problema se va a tratar, el contexto del mismo, así como la importancia de una adecuada solución. Esto es, hay que familiarizarse con los aspectos más relevantes del sistema. En este proceso de adquisición de conocimiento se debe recurrir tanto a la literatura sobre el tema, como a entrevistas con personal cualificado de forma que se comprendan totalmente el problema bajo estudio.

Además, se deben determinar los recursos (tiempo, personal, equipos, ...) necesarios para la resolución del problema. En el caso de que no existan los recursos necesarios se debe replantear el mismo de forma que pueda ser resuelto con los recursos disponibles. En muchos casos, esta restricción ocasiona el tener que resolver un problema menos ambicioso.

2.- Modelado.-

En la segunda fase del proceso de simulación hay que realizar el modelado del sistema, esto es, la construcción de un modelo que represente al sistema real. Las características del modelo deben ser representativas del sistema real. Debido a que el modelo presupone una reducción de la complejidad del sistema a unos niveles

manejables, uno de los objetivos en esta fase es seleccionar el conjunto mínimo de características del sistema que debe tener el modelo del mismo, así como las relaciones existentes entre estas características.

Para realizar una simulación digital, se tendrá que obtener un modelo algorítmico (conjunto de instrucciones a realizar por el ordenador para resolver un problema). Como paso previo se debe construir un modelo del sistema, matemático o simbólico, que ayude a la obtención del programa y a la realización de la documentación del mismo.

En la elección del tipo de modelo hay que tener en cuenta que como los modelos matemáticos son los más precisos, siempre que las relaciones entre las variables del sistema correspondan con estructuras matemáticas se utilizarán estos modelos. Pero en la mayoría de los casos, los sistemas a estudiar son tan complejos que no existen relaciones matemáticas suficientemente sencillas que nos permitan el modelado del sistema. En estos casos hay que recurrir a los modelos simbólicos (esquemas, diagramas de flujo, lenguajes, ...).

Si el modelo final es un programa de computador, en la fase de modelado hay que realizar la selección del lenguaje en el cual se va a escribir el programa. En esta elección hay que tener en cuenta las siguientes condiciones:

- Dificultad para trasladar el modelo y sus interrelaciones al lenguaje.
- La presencia o ausencia de facilidades en el lenguaje para realizar determinadas actividades tales como la gestión de colas, la generación de números aleatorios, etc.
- Posibilidad de interface con otras aplicaciones tales como *shell* de sistemas expertos, bases de datos, etc.
- Adecuada respuesta en el tiempo de ejecución.
- Interface amigable y facilidad en la depuración del programa.

Cuando el sistema en estudio es tan complejo que no se puede utilizar un modelo para representarlo, se debe recurrir al método conocido como reducción del problema o modelado de subsistemas. Esta metodología consiste en dividir el sistema en un conjunto de subsistemas menos complejos. Luego, cada subsistema es modelado y el modelo global resultará de la adecuada unión de los modelos de los subsistemas. Los

requisitos necesarios para poder llevar a cabo esta técnica son que los subsistemas puedan ser identificados, divididos y modelados.

Para identificar o dividir los subsistemas podemos utilizar tres mecanismos diferentes:

a) La división del sistema se hace en base al flujo de individuos o información a través del mismo. Los componentes del sistema vienen dados por las entidades a través de las cuales fluyen los ítems (individuos, información) del mismo.

b) Un segundo método es dividir el sistema atendiendo a la funcionalidad de los subsistemas. Esta técnica se utiliza cuando no hay entidades directamente observables que fluyan por el sistema.

c) Un tercer caso es identificar los sistemas atendiendo a los cambios de estado del mismo. Este procedimiento es útil en sistemas que están caracterizados por un gran número de variables que se interrelacionan, y que deben ser examinadas a intervalos regulares de tiempo para detectar cambios en el sistema. Las características del sistema que responden de la misma manera a un estímulo o conjuntos de estímulos se agrupan formando subsistemas.

3.- Verificación, validación y aplicación.-

Para que un modelo sea útil y los experimentos realizados sobre él sirvan para obtener conclusiones, tenemos que tener confianza en sus predicciones y en sus resultados. Validando y verificando el modelo se obtiene esta confianza.

Validar un modelo no es lo mismo que verificarlo. La validación consiste en demostrar que el modelo es una representación adecuada de la realidad. La verificación conlleva testear la consistencia del diseño (bondad y adecuación a nuestro problema de las metodología utilizadas en el modelado, de los algoritmos, de los programas de computación, ...), esto es, debemos probar que el modelo trabaja como hemos propuesto.

La validación de un modelo no es una tarea fácil ya que no existe un procedimiento o algoritmo que indique los pasos a dar en la validación. Además, debido

a que los sistemas reales no son completamente conocidos, así como que los modelos nunca son una representación exacta de la realidad, la validación se complica aún más.

Dado que el objetivo de la validación es establecer el grado de fidelidad del modelo o el nivel de exactitud con el que representa la realidad, en este proceso debemos tener en cuenta las siguientes consideraciones:

a) Validar los conceptos: Debido a que el modelo es un conjunto de deducciones lógicas procedentes de una serie de teoremas o axiomas cuya veracidad es incuestionable, en esta fase nos debemos preguntar acerca de las presunciones sobre las que se basa el modelo o de las evidencias empíricas que ocasionaron las suposiciones hechas.

b) Validar la metodología: Se debe examinar las metodologías utilizadas en la formulación del modelo así como en las soluciones dadas a los distintos problemas (aproximación de problemas no lineales mediante métodos lineales, representación de sistemas continuos mediante su equivalente discreta, exactitud en el uso de las metodologías computacionales, etc). Es evidente que un error de metodología puede ocasionar soluciones absurdas.

c) Validar los datos: Los datos utilizados en el modelado pueden ser defectuosos como consecuencia de errores de observación, errores de calibración, interpolación/extrapolación, inexactitud en la estimación de parámetros, etc. Por todo esto los datos deben ser cuidadosamente establecidos antes de poder llegar a conclusiones sobre los mismos.

d) Validar los resultados: El grado de adecuación entre la respuesta del modelo y los resultados teóricos o medidos juegan un papel importante en la validación (validación predictiva del modelo). Este grado de adecuación es obtenida utilizando métodos estadísticos tales como análisis de varianza, regresión, análisis espectral, etc., los cuales nos indican la utilidad de los datos para la interpretación de los resultados.

e) Validar las inferencias: Las inferencias son correctas cuando las conclusiones realizadas por varios expertos coinciden con las del modelo.

El modelado no es precisamente una ciencia, por lo tanto los criterios para testear la robustez de las teorías científicas no se pueden aplicar estrictamente a los modelos.

Las técnicas utilizadas para asegurar que las condiciones anteriores se cumplen son las siguientes:

1.- Comparar los resultados de la simulación con los datos históricamente producidos por el sistema real (validación retrospectiva).

2.- Utilizar la simulación para predecir resultados, y éstos compararlos con los resultados producidos por el sistema real en un periodo de tiempo futuro (validación predictiva).

3.- Asegurarse de que el modelo refleja fielmente el comportamiento interno del sistema real (validación estructural).

Después de validar el modelo se debe observar que éste posee las siguientes características:

- Exactitud.- Si su respuesta es correcta o muy próxima a la respuesta del sistema real.
- Descripción realista.- Si está basado en proposiciones correctas, lo que significa que se han extraído de descripciones correctas del sistema real.
- Precisión.- Si sus predicciones son números definidos (o curvas) y no un rango de números (o conjunto de curvas).
- Robustez.- Si es relativamente inmune a errores en los datos de entrada.
- Generalidad.- Si se puede aplicar a una gran variedad de situaciones.
- Productivo.- Si sus conclusiones son útiles o es un punto de partida para mejores modelos.

Todo lo visto indica que las decisiones tomadas en la validación del modelo son las más problemáticas del proceso de modelado, pero ineludiblemente se tienen que realizar si se quiere asegurar un correcto uso del modelo.

Una vez que el modelo ha sido adecuadamente validado, éste puede ser utilizado para resolver problemas , esto es, experimentar con él y llegar a conclusiones.

Un hecho importante a tener en cuenta en el momento de utilizar los modelos es que la gran mayoría de ellos tienen únicamente un rango limitado de validación. Fuera de éste rango la integridad del modelo está en peligro, por lo tanto, no se deben realizar experimentos fuera del dominio de validación. Para ello se debe elegir cuidadosamente los parámetros de cada experimento (condiciones iniciales, variables a observar, tiempo de ejecución, datos recogidos, ...).

5.5.- Ventajas y Desventajas de la Simulación

En los últimos años la simulación se ha aplicado en muchas de las áreas de actividad humana. Una pregunta que nos surge es: "Si la simulación es tan buena ¿por qué se utilizan otras técnicas en el estudio de los sistemas?". La respuesta es que la simulación no se puede aplicar en muchos casos, y en otros casos que si se puede existen otras técnicas más fáciles y más baratas para resolver el problema por ejemplo se puede acceder al sistema real a bajo costo. Para determinar cuándo es adecuado utilizar la simulación y cuándo no, debemos conocer cuales son las ventajas y desventajas de su uso [McHaney 1991].

a) Ventajas

1.- Permite experimentar sin perturbar el sistema bajo estudio.- En el sistema el testeo de nuevas ideas puede ser difícil, costoso o imposible. Por ejemplo, en una línea de ensamblaje de coches que trabaja 24 horas al día, 7 días a la semana, parar la producción para probar modificaciones que pueden o no aumentar la velocidad del proceso puede resultar muy caro. En este tipo de situaciones la simulación ofrece grandes ventajas ya que permite, después de tener construido y validado el modelo del sistema, examinar el efecto de cualquier modificación que se desea realizar y decidir si es conveniente o no llevarla a cabo en el sistema real.

2.- Las ideas (conceptos) pueden ser testeados antes de su implementación.- En el diseño de nuevos sistemas una simulación permitirá testear las ideas antes de llevarlas a la práctica. De esta manera se pueden detectar errores imprevistos y el diseñador puede corregirlos antes de su implementación. Si estos errores se detectan después de instalado el sistema, los cambios necesarios para eliminar estos fallos

resultan mucho más costosos y difíciles de realizar. Un ejemplo lo tendríamos en una empresa que decide cambiar su sistema de tratamiento automático del stock del almacén. Si este sistema no se ha simulado anteriormente para detectar posibles errores podría ocurrir que no estuviera preparado para actualizar el material del almacén en el tiempo necesario y que después de varios meses funcionando con el nuevo sistema se observara que no se puede continuar con la producción por la no existencia de material. Este hecho reportaría grandes pérdidas a la empresa que no hubiera tenido si previamente se hubiera experimentado con el nuevo sistema.

3.- Aumento en el conocimiento sobre el sistema.- Un beneficio que es directamente el resultado de un proceso de simulación es el incremento del conocimiento acerca del sistema. En todo proceso de simulación, especialmente en el modelado de sistemas complejos, el conocimiento está disperso entre diferentes personas (cada individuo es un experto en un área particular). Para realizar la simulación se debe recoger y analizar toda esta información, y esto hace que se aumente el conocimiento sobre el sistema.

4.- Fuerza la definición del sistema.- Para conseguir un buen modelo del sistema es necesario que todos sus aspectos sean totalmente conocidos. Si la definición del sistema es incorrecta o incompleta el modelo resultante será inexacto y no se podría utilizar como herramienta de análisis. Por lo tanto el desarrollo de la simulación fuerza a definir completamente todos los elementos del sistema y sus interrelaciones. Si ciertos aspectos no pueden ser determinados con exactitud se deben estudiar con mucho cuidado.

5.- Mayor velocidad en el análisis de los sistemas.- La simulación permite un análisis más rápido debido a que el trabajo del sistema sobre un periodo largo de tiempo puede ser simulado en unos pocos minutos (compresión del tiempo). Muchas plantas de manufactura mantienen un sistema de simulación de su proceso de ensamblaje, de forma que cada mañana le introducen las entradas y predicen cómo funcionará la fábrica durante todo el día.

6.- Aumenta la creatividad de los diseñadores.- Por ejemplo un diseñador concibe dos posibles soluciones a un problema. Una de las soluciones garantiza la solución pero es más cara. La otra es una nueva tecnología y es más barata pero ofrece un mayor riesgo. Si no existiera una simulación que permitiera el análisis de las dos soluciones se optaría por llevar a la práctica la más conservadora, mientras que con la simulación la creatividad del ingeniero puede ser ejercitada sin riesgo de fallos.

7.- Es una herramienta eficaz de entrenamiento.- En muchos casos es conveniente que los nuevos expertos no se entrenen sobre el sistema real ya que esto es peligroso o caro. Por ejemplo, si queremos enseñar a los nuevos operadores de una planta nuclear cómo realizar su trabajo, es conveniente que se realice sobre una simulación con lo que se evita la interacción con el sistema de trabajadores novatos que pueden originar situaciones de peligro.

Todas estas ventajas de utilizar la simulación como una herramienta de análisis tienen un punto en común: reducir el riesgo. Esto es debido a que la incertidumbre acerca de los sistemas es eliminada y reemplazada con conocimiento acerca de cómo va a operar el nuevo sistema o cómo afectaran los cambios en el sistema existente.

b) Desventajas

1.- Puede resultar un proceso costoso.- Realizar un simulación puede resultar costoso en término de horas de trabajo de expertos, así como de necesidades de hardware y software.

2.- Consumo de tiempo.- El modelado de sistemas no produce respuesta rápidas ante las cuestiones planteadas. En la mayoría de los casos, recoger los datos, construir el modelo, analizar el sistema y generar los informes oportunos ocasionan un considerable gasto de tiempo. El proceso de simulación puede ser más rápido si se utilizan las siguientes técnicas:

a) reducir el nivel de detalle: Cuando se reduce el nivel de detalle se puede llegar a la solución de las preguntas generales sobre el sistema en menor tiempo pero se tiene

que tener cuidado con la exactitud o fidelidad del modelo.

b) utilizar librerías de código genérico: En situaciones donde se realizan simulaciones similares se pueden utilizar librerías de funciones genéricas que nos permitirán disminuir los costes del proceso de simulación.

3.- Genera soluciones aproximadas.- En las soluciones de sistemas de eventos discretos se utilizan números aleatorios para generar la entrada del modelo. Debido a esto, existe cierta incertidumbre asociada con la salida. Para que los resultados de la simulación sean significativos la salida debe ser interpretada usando métodos estadísticos. Las salidas son únicamente estimaciones del verdadero comportamiento del sistema.

4.- Dificultad para validar.- La validación es asegurarse que el modelo representa fielmente la realidad bajo estudio. Hay situaciones, como cuando el sistema no existe, que este proceso resulta una tarea complicada. La validación en estos casos se hace en base a las opiniones de los expertos y es necesario tomar una actitud conservadora de forma que se pueda asegurar que el procedimiento de simulación no genere una salida mejor que la del sistema real.

5.- Sus resultados son considerados como dogmas.- Los usuarios del proceso de simulación pueden considerar la salida de la simulación como una verdad absoluta y esto puede llevar a errores. Aunque el modelo esté validado, el usuario debe realizar un riguroso análisis de los informes generados por la simulación. Ya que muchas veces se detectan errores en la simulación cuando se examinan unos experimentos concretos.

5.6.- Lenguajes de Simulación

Cuando la simulación se va a realizar sobre un ordenador, hay un paso importante en el proceso de modelado que es la elección de la herramienta de simulación que nos permitirá construir el programa. Atendiendo a las características del problema a estudiar se debe elegir una u otro lenguaje de simulación. Una clasificación de lenguajes de simulación aparece en la figura 1.5.

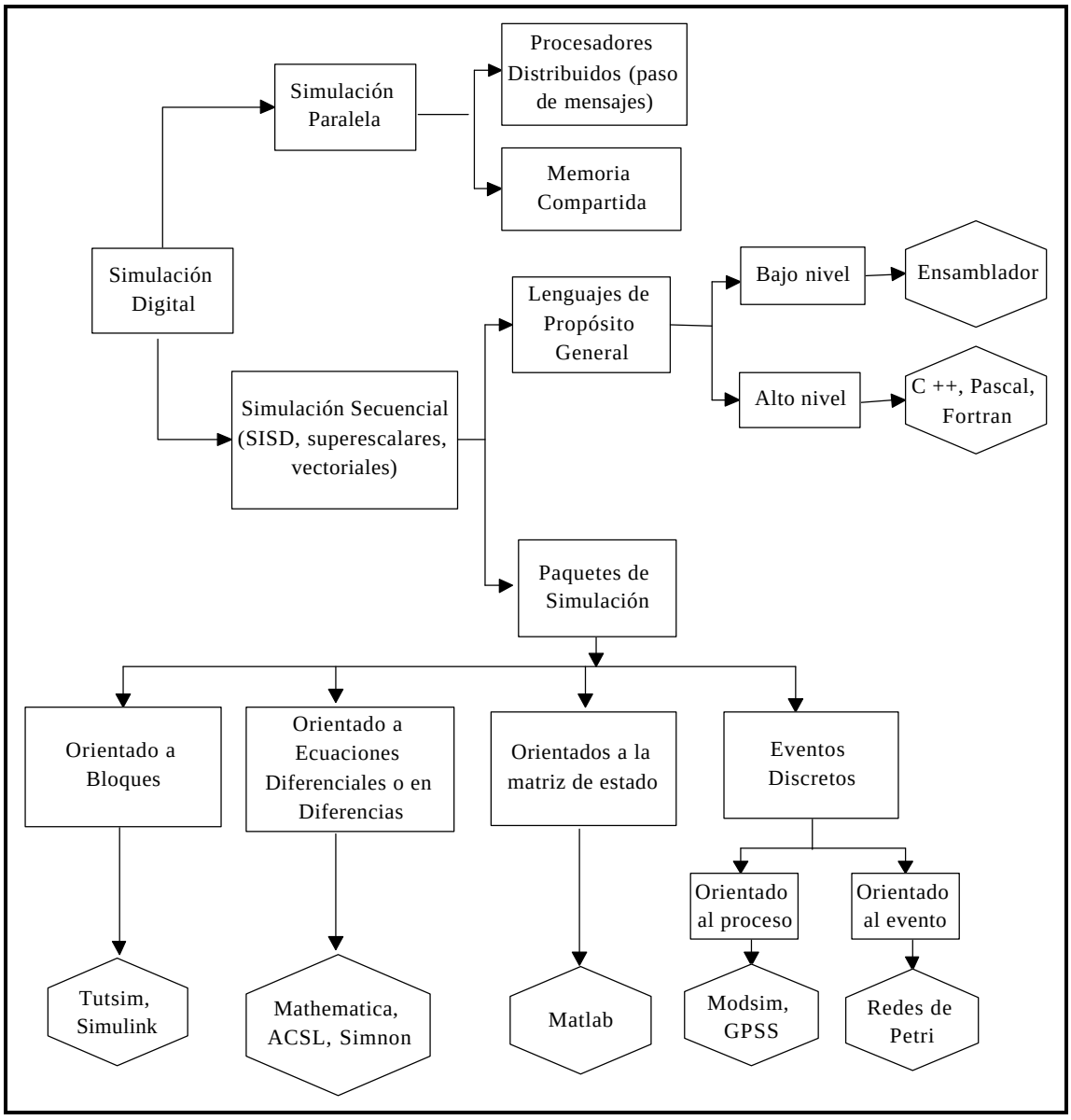


Figura 1.5.- Lenguajes de simulación

Los lenguajes de programación que nos permiten la construcción de simulaciones los podemos dividir en dos grandes grupos:

1.- Lenguajes para simulaciones paralelas y distribuidas, que permiten acelerar el tiempo de ejecución mediante el uso de un sistema multiprocesador. Se conoce como simulación paralela (SIMD, simple instrucciones mútiples datos) la ejecución de la simulación en P procesadores bajo un control central, la comunicación se suele realizar con arquitecturas de memoria compartida. Tenemos simulación distribuida (MIMD, múltiple instrucciones múltiples datos) cuando la simulación se realiza con P procesadores cada uno de ellos realizando un conjunto de procesos de

forma asíncrona, con lo que se consigue un aumento en el nivel de paralelismo asi como en el número de comunicaciones las cuales se realiza mediante el paso de mensajes.

La simulación se puede paralelizar a distintos niveles:

- Nivel de aplicación: La forma más obvia de acelerar los experimentos necesarios para explorar en un espacio de búsqueda es asignar réplicas independientes del mismo modelo de simulación y con diferentes parámetros a los distintos procesadores.

- Nivel de subrutina: Cuando los distintos experimentos no son independientes y no se puede asignar una réplica a cada procesador una manera de acelerar la ejecución es distribuir las distintas subrutinas que constituyen la simulación (procesar evento, salvar estado, realizar estadísticas, ...) en los procesadores disponibles.

- Nivel de componente: Se utiliza el paralelismo existente en el sistema modelado para acelerar la ejecución. En un sistema diseñado siguiendo una metodología orientada al objeto una descomposición natural sería que cada instanciación de una clase de objeto, que corresponde con un componente del sistema real, sea asignado a un procesador.

- Nivel de evento: Se distribuye cada evento que constituye la simulación entre los distintos procesadores. Esta paralelización se puede realizar de dos maneras:

a) Con una lista de eventos centralizada que es gestionada por un procesador "master" encargado de mandar a ejecutar los distintos eventos en los procesadores "slave".

b) Los eventos se distribuyen arbitrariamente por los procesadores con lo cual se consigue un mayor grado de paralelismo pero se incrementa las sincronizaciones necesarias y por lo tanto el número de comunicaciones a realizar.

2.- Lenguajes para simulaciones secuenciales. Dentro de los lenguajes englobados en el segundo grupo tenemos los lenguajes de propósito general y los lenguajes de propósito específico.

Los lenguajes de propósito general nos permiten implementar cualquier tipo de programa y por lo tanto no tienen características propias para simulaciones. Se pueden

dividir en lenguajes de bajo nivel (ensamblador) pocos utilizados por la complejidad de su manejo; y los lenguajes de alto nivel (C++, Pascal, Fortran) más sencillos de programar.

Los lenguajes o paquetes de simulación son herramientas diseñadas específicamente para realizar simulaciones. Atendiendo a la forma en la que es representado el modelo del sistema en el paquete de simulación tenemos los siguientes lenguajes:

- Orientados a bloques (Tutsim, Simulink, etc.): Cuando el modelo del sistema viene dado por una ecuación diferencial o en diferencias se puede utilizar estos tipos de lenguajes que permiten representar las ecuaciones por medio de bloque de tipo integración, suma, etc.

- Orientados a ecuaciones diferenciales o en diferencias (Mathematica, ACSL, Simnon, etc.): En esta clase de lenguajes la ecuación diferencial es descrita por medio de expresiones matemáticas, lo que significa que las ecuaciones son incluidas directamente en el programa. El inconveniente es que requieren mayor tiempo de aprendizaje por parte del usuario. Entre los ejemplos anteriores, Mathematica tiene la característica de utilizar métodos analíticos para resolver las ecuaciones, frente a los métodos numéricos utilizados por los otros lenguajes.

- Orientado a la matriz de estado o función de transferencia (Matlab): Es una herramienta que permite describir el sistema mediante su matriz de estado. Otra facilidad el número elevado de toolboxes o librerías existentes para diferentes situaciones: Control, Procesamiento de Señales, etc.

- Orientados a eventos discretos: Los sistemas de eventos discretos se pueden modelar siguiendo dos enfoques:

a) Simulación orientada al evento: en este caso se definen todos los eventos y lo efectos que éstos producen en el sistema. Las Redes de Petri son las herramientas matemáticas utilizada para este tipo de modelado. Los programas (SimNet, CPN, etc.) utilizados para la simulación de estos sistemas se caracterizan por permitir el diseño y simulación de distintos tipos de Redes de Petri.

b) Simulación orientada al proceso: Son lenguajes (Modsim, GPSS, etc.) que permiten el modelado y simulación del sistema mediante la descripción de los procesos existentes en el mismo, esto es, la secuencia ordenada en el tiempo de eventos interrelacionados.

6.- INTELIGENCIA ARTIFICIAL

Aunque no hay una definición clara de Inteligencia Artificial, la podemos describir como el intento de construir máquinas que piensen y actúen como los humanos, y que sean capaces de aprender y utilizar su conocimiento para resolver problemas.

Por lo tanto en IA se estudia cómo trabaja la mente humana para poder simularla. En esta línea, los investigadores de IA se han dividido en dos grandes grupos que engloban dos filosofías diferentes de simular el cerebro. Las dos ramas en las que se divide la IA son:

a) conexionista: Explica el funcionamiento del cerebro humano como un mecanismo de neuronas interconectadas, intentando reproducir electrónicamente cada neurona y sus interrelaciones [Freeman 1991] [Wasserman 1989] [Azoff 1994].

b) simbólica: Construye sistemas inteligentes preocupándose de los mecanismos de representación y manipulación de la información, obviando los mecanismos neurofisiólogos involucrados en tales procesos [Weis 1984] [Kline 1989].

Un aspecto importante en la construcción de un sistema inteligentes es verificar si ha cumplido el objetivo marcado y por lo tanto tiene las características de inteligencia. Clasicamente se ha comprobado sometiendo al sistema al Test de Turing [Turing 1950] que fue inventado por Alan Turing en 1950 y consiste en lo siguiente:

"Una máquina sería inteligente si fuera capaz de realizar las capacidades cognitivas necesarias para conversar con un juez humano y hacer creer a éste que su interlocutor es una persona, imitando las facultades humanas necesarias para ello. El juez interacciona con el sistema examinando a través de un terminal para evitar cualquier pista sobre su aspecto físico. La conversación debe durar un tiempo razonable y

transcurrido éste, el juez debe indicar si al otro extremo de la línea se encuentra una máquina o una persona. Si el juez no es capaz de discriminar se considera el sistema como inteligente"

La figura 1.6 muestra la breve pero productiva historia de la Inteligencia Artificial.

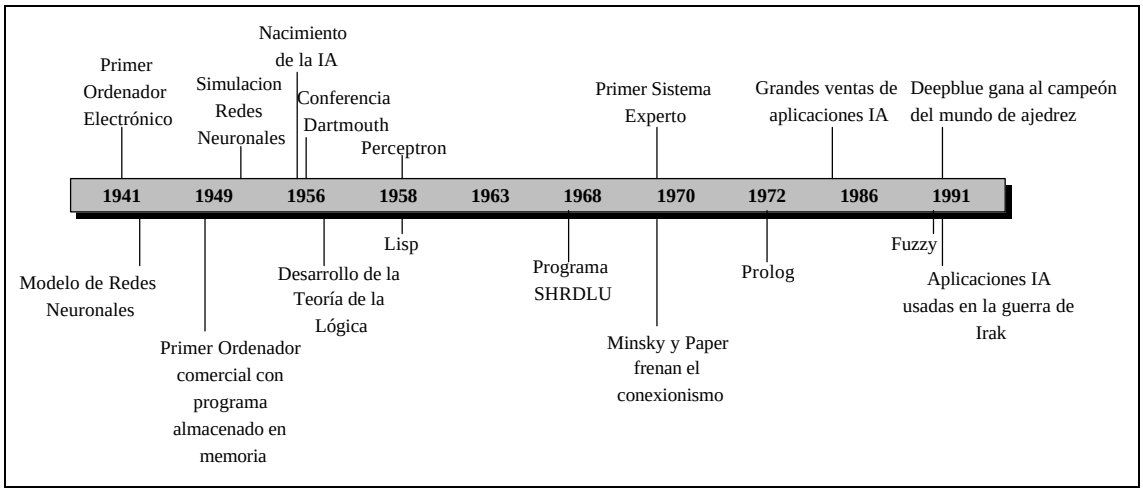


Figura 1.6.- Historia de la Inteligencia Artificial

6.1.- Sistemas Expertos

Los primeros trabajos en IA los realizaron en 1943 el neurofisiólogo Warren McCulloch y el matemático Walter Pitts, que para explicar el funcionamiento del cerebro humano propusieron un modelo de neuronas artificiales, cada una caracterizada por estar "abierta" o "cerrada" en respuesta al estímulo de las neuronas vecinas. Sin embargo, no es hasta la conferencia de Darmonth en 1956, donde se reúnen investigadores como Marvin Minsky, John McCarthy, Herbert Simon y Allen Newell; y se acuñan el término de Inteligencia Artificial.

Desde entonces se trabaja en aplicaciones inteligentes que constituyen la historia de la IA, textos donde se pueden encontrar los diversos trabajos que forman parte de ella son [McCorduck 1979] [Piñeiro 1996]. Debido a la importancia que tienen los Sistemas Expertos en la presente memoria nos centraremos a continuación en el nacimiento y desarrollo de estas aplicaciones.

A mediados de los sesenta, los investigadores de Stanford trabajan en un proyecto que va a dar lugar al nacimiento de una nueva industria en el campo de la IA. Su logro fue desarrollar un programa de ordenador (más tarde llamado Dendra) que podía elucidar la estructura de complejas moléculas a partir del espectograma de la masa; y su rendimiento se situaba al nivel de los expertos humanos. La metodología utilizada por estos científicos, la cual era única en esos momentos, fue codificar el conocimiento obtenido de un experto químico y usarlo para dirigir el programa.

El éxito de este proyecto se acentuó debido a que era la primera vez que se desarrollaba un programa inteligente de ordenador. Un programa cuya principal característica era que tenía conocimiento acerca del problema, más que el tipo de algoritmo de búsqueda utilizado. La era de los Sistemas Expertos (SE) había comenzado.

Un SE es un producto de la Inteligencia Artificial que incorpora el conocimiento de una persona experimentada en algún campo del conocimiento, debiéndose separar claramente de este área aquellos sistemas que incorporan conocimientos algorítmicos o sistematizados. Una de las condiciones necesarias, para que un sistema informático esté contenido en este área, es que el conocimiento incorporado sea esencialmente heurístico. Un SE, en base al conocimiento almacenado, es capaz de interactuar con el mundo real en la misma forma que lo haría el experto humano y además, es capaz de justificar las razones que le llevan a actuar como lo hace.

6.2.- Áreas de Aplicación

Un Sistema Experto es una herramienta para la ayuda a la toma de decisiones humanas. Se utilizan en tareas que requieran mucho conocimiento heurístico. Por lo tanto, donde encontremos humanos haciendo tales actividades: diagnóstico [Moreno 1995] [Piñeiro 1995], diseño de estructuras, tutorizar un estudiante, ...; hay una fuente para la utilización de esta tecnología. Así encontramos SE desarrollados para una amplia gamas de campos [Pigford 1990] [Sierra 1995] [Harmon 1988]. En la figura 1.7 se muestra la mayoría de las áreas de aplicación dónde se han desarrollados de una manera natural. La figura también muestra el número de sistemas desarrollados para cada área

[Durkin 1996].

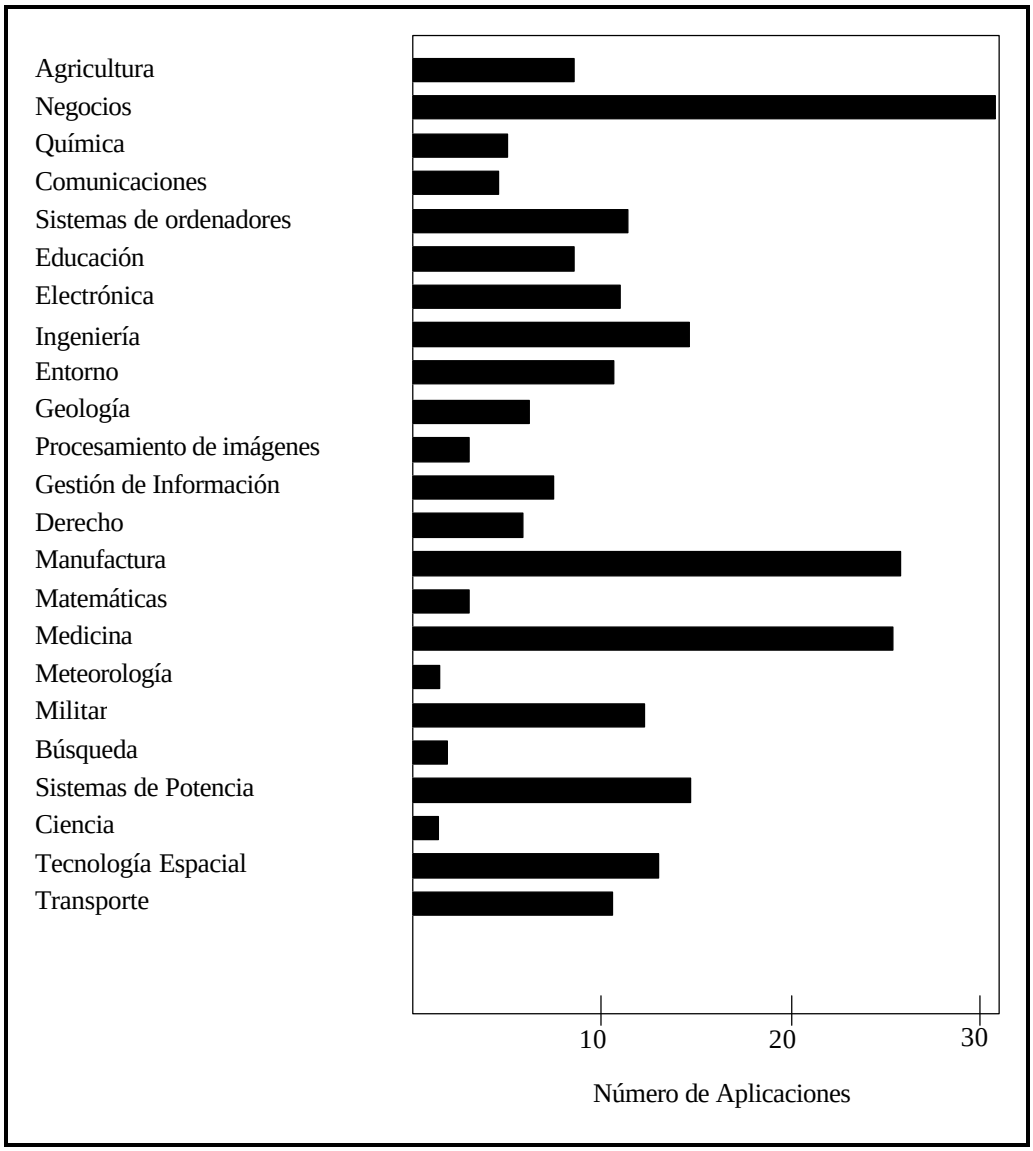


Figura 1.7.- Número de Sistemas Expertos desarrollados en las distintas áreas. (Estudio realizado sobre 2.500 sistemas, alrededor del 20% del total de sistemas realizados)

6.3.- Tipos de Aplicaciones

Los expertos humanos, cuando resuelven problemas tales como diagnóstico, planificación, etc; realizan un conjunto genérico de tareas. Independientemente del área de aplicación y dado un tipo de problema, el experto recoge y razona con información similar. Los SE realizan de la misma manera los trabajos genéricos sobre la base del tipo de problema. En la siguiente figura 1.8 observamos los diferentes tipos de problemas.

Tipo de Problema	Descripción
Control	Gobierno del comportamiento del sistema
Diseño	Configurar objetos que verifiquen determinadas restricciones
Diagnóstico	Inferir el mal comportamiento de un sistema a partir de observaciones
Instrucción	Diagnóstico, depuración y corrección del comportamiento del estudiante
Interpretación	Inferir descripciones de situaciones a partir de los datos adquiridos
Monitorización	Comparar las observaciones con las salidas esperadas
Planificación	Diseñar acciones
Predicción	Inferir consecuencias probables de situaciones dadas
Prescripción	Recomendar soluciones a funcionamientos erróneos en los sistemas
Selección	Determinar la mejor elección de una lista posible
Simulación	Modelar la interacción entre los distintos componentes de un sistema

Figura 1.8.- Tipo de Problemas que resuelven los Sistemas Expertos

La figura 1.9 muestra el porcentaje de aplicaciones para cada tipo de problema. Muchas aplicaciones realizan más de una actividad, por ejemplo, un sistema de diagnóstico puede primero interpretar los datos disponibles, y después prescribir un remedio para el error detectado.

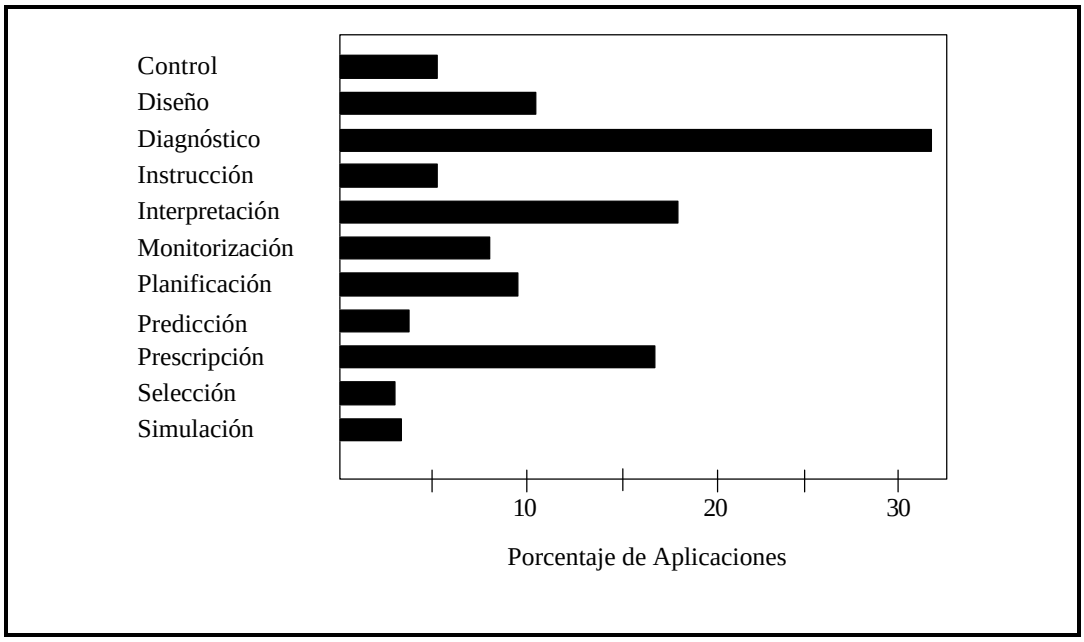


Figura 1.9.- Porcentaje de Sistemas Expertos realizados por cada tipo de problema. (Estudio realizado sobre 2.500 sistemas, alrededor del 20% del total de sistemas realizados)

Como muestra la figura 1.9, los SE han jugado un papel muy importante en las tareas de diagnóstico. Una de las razones ha sido que es el trabajo que realizan la mayoría de los expertos. Otra de las razones para el gran porcentaje de sistemas de diagnóstico es que su desarrollo es relativamente fácil. Esto es porque la mayoría de los problemas de diagnóstico tiene una lista de posibles soluciones, además se necesita una cantidad limitada de información para alcanzar la solución.

El gran porcentaje puede también deberse a las consideraciones prácticas de introducir una nueva tecnología en una organización. La mayoría de las empresas prefieren correr poco riesgo cuando se trata del uso de nuevas tecnologías. Así, prefieren los proyectos que con mínimo esfuerzo se obtienen máximos beneficios. Debido a que los sistemas de diagnósticos son relativamente fáciles de construir, son atractivos para que las empresas se aventuren en este campo.

6.4.- Definición de Sistema Basado en el Conocimiento

En los últimos años el término "Sistema Experto" ha sido reemplazado por el término "Sistema Basado en Conocimiento" (SBC), de esta forma, el énfasis original que se le daba a la transferencia de la experiencia humana a un computador ha sido sustituido por un término más moderado en el que se deja patente el uso de todos los recursos disponibles en la construcción del sistema. Sin embargo, sigue siendo una característica importante el que un SBC sea un sistema diferente a cualquier sistema de información avanzado.

La ingeniería de conocimiento puede ser vista como una parte de la ingeniería de sistemas de información, pero un sistema de información que le da un énfasis particular al carácter epistemológico (el saber sobre el conocimiento) de la información que es clasificada como conocimiento, esto es, "creencia verdadera imposible de desmentir con las leyes de la naturaleza" o "creencia verdadera adquirida a través de los sentidos". Estas definiciones nos dan un contexto en el cual la credibilidad y la derivación de la información son significantes, y tienen que ser tenidas en cuenta. Esto es, los componentes basados en conocimiento de un sistema de información son aquellos que requieran meta-información (conocimiento sobre la información) y meta-proceso de información (conocimiento acerca de la manipulación de la información). Tales sistemas tendrán subsistemas que no sólo procesen información sino que también procesen meta-información acerca de la información y de su procesamiento.

Una de las características más importantes de los primeros Sistemas Expertos, tales como MYCIN [Shortliffe 1976] y de los entornos de desarrollo relacionados como TEIRESIAS [Davis 1982] era la capacidad que tenían de ser reflexivos y responder a las cuestiones de por qué y cómo, y usar esta capacidad para guiar al usuario final en la evolución del sistema y para entender mejor el dominio.

Desde la perspectiva del procesamiento de la meta-información un SBC puede ser caracterizado como un sistema de información esencialmente reflexivo [Maes 1988]. Esto sugiere que la diferencia entre sistemas de información convencionales y los SBC es que éstos últimos incluyen procesamiento de la meta-información.

6.5.- Construcción de un SBC. Metodología KADS.

Inicialmente, los constructores de SBC no tenían una serie de pasos bien definidos para seguir a la hora de construir un SBC debido a la complejidad inherente al proceso de construcción. Como resultado, los SBC se desarrollaban de manera incremental, mejorando paso a paso la organización y la representación del conocimiento del sistema.

El resultado usual en un desarrollo incremental es que éste llega a un punto en el cual la base de conocimiento alcanza un tamaño inmanejable; el control no es efectivo y además es lento, y el sistema parece que está construido a partir de parches y construcciones no integradas. Ante este panorama, es necesario considerar el rediseño y la reimplantación del sistema. El ingeniero del conocimiento y el experto en el dominio deberán reexaminar el problema y rehacer su esquema de representación.

Esto dejaba patente que la construcción de SBC utilizando la metodología de construcción rápida de prototipos presentaba serios problemas.

Además, las bases de conocimientos que se construían se diseñaban para resolver un trabajo específico en una aplicación, y tenían poca o ninguna generalidad. De esta forma, el conocimiento que incorporaba un SBC no podía ser reutilizado para un propósito más general.

Otro problema existente era debido a la forma en que se representaba el conocimiento (principalmente con reglas de producción) que ocultaba importantes propiedades del proceso de razonamiento y de la estructura del conocimiento en el dominio de la aplicación. Esto dificultaba la adquisición y refinamiento del SBC, así como la reutilización del conocimiento, el poder reflexivo (para justificar sus resultados) del SBC y su relación con otros sistemas.

Ante tal panorama, sobre 1983 un grupo de investigadores comienzan a trabajar en una nueva metodología que se concentra en los métodos de adquisición del conocimiento, en el modelado del conocimiento y en el desarrollo estructural de los SBC. A esta metodología se le conoce con el nombre de KADS (Aunque actualmente el acrónimo KADS se utiliza como nombre propio, este se puede interpretar como "Knowledge Analysis and Documentation System" o como "Knowledge Analysis and

Design Support") [Wielinga 1994].

Con esta metodología se pueden construir los SBC de una forma comprensiva. Para ello sigue dos principios:

- el principio de múltiples modelos;
- y el principio de describir, de forma independiente de la implementación, las distintas capas de conocimiento que existen cuando se resuelve el problema.

6.5.1.- Principio 1: Múltiples Modelos

La construcción de un SBC es una tarea complicada. La idea que subyace detrás del primer principio de la metodología KADS, es la de facilitar la construcción de los SBC considerando el problema como formado por un conjunto de modelos, de forma que cada modelo enfatiza ciertos aspectos del sistema que se va a construir, y se abstrae de otros. Estos modelos permiten la descomposición de las tareas del ingeniero del conocimiento: mientras construye un modelo puede olvidarse temporalmente de otros aspectos. Esto es, el ingeniero del conocimiento sigue la estrategia de divide y vencerás.

Este proceso de modelado no se centra únicamente en el conocimiento experto con el que trabajará el SBC, sino que tiene en cuenta las características de cómo ese conocimiento es añadido y usado en la organización.

Los modelos a construir no son considerados como pasos necesarios para desarrollar el SBC, sino como productos independientes que juegan un papel importante durante el ciclo de vida del SBC.

Aunque el KADS propone un conjunto de modelos para el desarrollo del SBC, sin embargo el número y grado de elaboración de los modelos dependerán del contexto del proyecto específico y de la planificación y control realizado en el proceso de construcción [Hoog 1994].

El KADS define los siguientes modelos:

- Modelo organizativo

- Modelo de tareas
- Modelo experto
- Modelo de agentes
- Modelo de comunicación
- Modelo de diseño

Modelo Organizativo

Un modelo organizativo genera un análisis del entorno socio-organizativo en el cual el SBC tendrá que funcionar. Esto incluye una descripción de las funciones, tareas y cuellos de botellas en la organización. Además, debe incluir los problemas que el SBC debe resolver y cómo influirá su uso en la organización y en las personas que en ella trabajan.

Modelo de Tareas

El modelo de tareas describe las actividades, de forma abstracta, necesarias para realizar las funciones en la organización. Establecer una relación directa entre función y tareas no es trabajo fácil, ya que dado el objetivo que el sistema debe lograr podrían haber varias formas de alcanzarlo. Qué alternativa es la adecuada para una aplicación determinada depende de las características de la propia aplicación, sobre la disponibilidad de conocimiento y datos, y sobre los requisitos impuestos por el usuario o por factores externos.

Modelo Experto

La construcción del modelo experto es la actividad central en la construcción de un SBC. Esto es lo que distingue el desarrollo de un SBC del desarrollo de sistemas convencionales. Su objetivo es especificar la experiencia de resolución de problemas requerida para realizar las tareas de resolución de problemas asignadas al sistema. La metodología KADS considera el modelo experto como el comportamiento que mostrará el sistema, y se centra en el tipo de conocimiento necesario para generar tal comportamiento, abstrayéndose de los detalles de cómo el razonamiento es realizado en la implementación.

Modelo de agentes

En el modelo de agentes se describen todas las propiedades relevantes de los agentes que realizan las tareas identificadas en el modelo de tareas. Un agente es un ejecutor de una tarea. Un agente puede ser un humano, o un programa de computadora, o cualquier "entidad" que ejecute una tarea. Los agentes más involucrados suelen ser los usuarios y el SBC. Un agente muy común es el sistema de bases de datos que almacena y proporciona información.

Modelo de Comunicación

Los SBC deben cooperar con otros agentes (usuarios, programas de ordenador, ...) para realizar su trabajo. Esta distribución de tareas define comunicaciones entre los agentes para poder lograr todos los objetivos. El modelo de comunicación es el encargado de describir los conceptos y mecanismos necesarios en cada una de las comunicaciones.

Modelo de Diseño

El modelo de diseño describe las técnicas computacionales que el SBC debería usar para tener el comportamiento especificado. Este modelo tiene en cuenta los requisitos del entorno que no son considerados en los otros modelos, tales como requisitos de velocidad, hardware y software.

6.5.2.- Principio 2: Modelado de las Capas de Conocimiento

El paso más importante en la construcción de un SBC es encontrar la respuesta adecuada a la pregunta de cómo construir el modelo experto. El segundo principio del KADS nos dice que este modelo está formado por diferentes capas de conocimiento, y éstas son independientes de cualquier implementación concreta.

Esto es, se asume que en la construcción de un SBC es posible y útil distinguir entre distintos tipos de conocimiento de acuerdo a los diferentes roles que juegan en el proceso de razonamiento. Además, se asume que estos tipos de conocimiento se pueden organizar en distintas capas.

Una primera distinción de tipos de conocimiento sería el conocimiento de dominio (conocimiento estático, formado por los conceptos, relaciones y hechos que son necesarios para razonar acerca de una determinada aplicación), y el conocimiento de control (conocimiento acerca de cómo controlar el proceso de razonamiento en la resolución de un problema).

El conocimiento de control a su vez se divide en conocimiento de inferencias (describe las operaciones básicas a realizar) y conocimiento de tareas (conocimiento acerca de las tareas necesarias en la obtención del objetivo perseguido).

Los tipos de conocimientos utilizados por un SBC quedan reflejados en la figura 1.10:

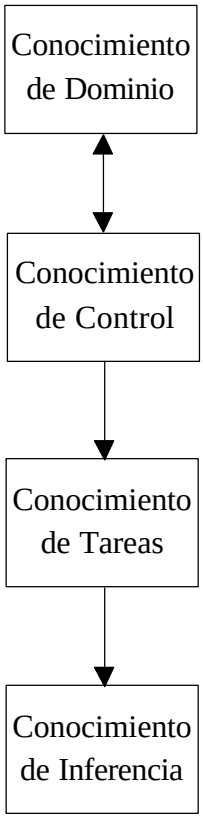


Figura 1.10.- Distintos tipos de conocimiento de acuerdo a los diferentes roles que juegan en el proceso de razonamiento, en la construcción de un SBC.

La separación del conocimiento, en conocimiento de dominio y conocimiento de control, origina una importante pregunta: ¿cuál es la dependencia entre estos dos tipos de conocimiento?. Para responder a esta cuestión Chandrasekaran formuló la hipótesis de la interacción en la que dice que estos dos conocimientos son altamente dependientes: no se puede definir el conocimiento del dominio sin conocer que modelo de tarea se va a realizar, y viceversa [Bylander 1988]. Asimismo, las interacciones que presenta son limitadas debido a que el conocimiento de dominio se puede dividir en el que es verdaderamente del dominio general y aquel que es conocimiento de dominio particular al problema que se está resolviendo y que tiene en cuenta el estado interno del problema. Los que propugnan la metodología KADS sostienen que existe alguna interacción, pero que éstas se pueden distinguir en diferentes tipos. Para resolver esta cuestión, proponen una serie de meta-modelos (denominadas vistas del dominio) que especifican los papeles que juega cada pieza de conocimiento del dominio en cada método de resolución de problemas. Por ejemplo, en el dominio de la mecánica de coches, el hecho "rueda desinflada" puede ser usado como evidencia para establecer un diagnóstico más complejo o bien como una hipótesis de avería final.

7.- EL HOSPITAL: UN SISTEMA COMPLEJO

El hospital, independientemente de su propiedad pública o privada, es una empresa de servicios. Como tal ha evolucionado más lentamente que otras organizaciones en buena medida por la falta de definición operativa de su producto. La gestión hospitalaria ha emergido en los últimos tiempos y su importancia será aún mayor en los próximos años.

Debido a que el modo de proceder de los hospitales se modifica a medida que evoluciona el sistema sanitario que determina su entorno, comenzamos el estudio de los hospitales públicos españoles describiendo el Sistema Sanitario dentro del cual se desarrollan los mismos.

7.1.- La Sanidad: Perspectiva Histórica

Hasta hace relativamente pocos años y en países de un desarrollo sociocultural similar al nuestro, la Salud Pública agrupaba todas las acciones sanitarias emprendidas por la Administración o el Gobierno para la defensa de la salud colectiva quedando al margen la atención a los problemas de la salud individual que no tenían una trascendenci pública o social. Las grandes epidemias o catástrofes sanitarias eran las desencadenantes de los movimientos sanitarios. Es la época primaria de la Salud Pública, pero que todavía hoy es la única Salud Pública existente en gran parte del mundo. En España la responsable de las actividades de Salud Pública tradicionales era la Dirección General de Sanidad dependiente del Ministerio de la Gobernación (Ministerio del Interior), estando regulado por la Ley de 28 de Noviembre de 1855 que extiende su vigencia durante una larguísima época aunque no en razón a sus excelencias sino a la imposibilidad de llegar a un acuerdo sobre una nueva ley sanitaria. La reforma siguiente se establece por Real Decreto del 12 de Enero de 1904, que apenas alteró la organización de 1855, cambiando por épocas el nombre de Dirección General de Sanidad por el de Inspección General de Sanidad [Beltrán 1997].

El desarrollo social y económico hace que vayan imponiéndose otros conceptos en que predomina la Justicia Social; son los tiempos de la Medicina Preventiva y Social, la cual, unida a la Higiene, pasa a ser el esqueleto de lo que se denominó Salud Pública en el segundo y tercer cuarto de nuestro siglo en los países con un nivel de desarrollo por lo menos de tipo medio. En España es regulada por la Ley del 25 de Noviembre de 1944, permaneciendo la idea de que la responsabilidad de la Administración Pública es atender aquellos problemas sanitarios que puedan afectar a la colectividad quedando al margen los problemas de la salud individual.

Esta concepción de Medicina Social que se vincula únicamente a las actividades de Prevención y Promoción y no a las de asistencia a la enfermedad, lleva consigo que en estas actividades tenga cabida casi cualquier acción humana que sea "buena" para las personas porque de una forma u otra repercutirá en salud; son los tiempos de las diferentes definiciones de Salud y de Salud Pública que abarcaban desde la salud física y psíquica hasta la felicidad individual y colectiva.

Junto a estas globalizaciones en las soluciones que se imponen en la década de

los setenta y principios de los ochenta, va adquiriendo una importancia creciente la Atención

Primaria de Salud como eje del Sistema Sanitario sobre el cual habrá de girar el mismo. Estos planteamientos progresan con gran rapidez principalmente en los países en que sus servicios de salud son precarios o en aquellos como es el caso del nuestro en que la Atención Primaria había quedado descolgada del desarrollo general del Sistema Sanitario y mantenía las consultas de medicina general con escasos medios y dedicación, y haciendo descansar toda la asistencia sobre los hospitales que se habían construido y dotado aprovechando el desarrollo económico habido desde la década de los sesenta. No es extraño pues que la reforma de la Atención Primaria se convirtiese en una bandera de los cambios sanitarios. Pero la Atención Primaria que se propone va mucho más allá del cambio del modelo asistencial; es una concepción de Atención Primaria Integral que trabaja con la colectividad, y no solo con el individuo, todos los aspectos de la Prevención de la enfermedad, la Promoción de la Salud, la Asistencia Sanitaria y la Rehabilitación y ello con actuaciones sobre el medio para atajar los factores de riesgo existentes en el mismo. Es la época florida de los diagnósticos de salud, de la contratación de técnicos de Salud Pública por parte de la entidad gestora de la asistencia para poder coordinar todas estas actividades, etc. La Prevención y Promoción como entidades específicas y con vida propia van perdiendo importancia salvo en lo que concierne a las inspecciones de sanidad ambiental y alimentaria y acciones a nivel regional de proveedores de vacunas y recopilaciones epidemiológicas con mayor o menor profundidad. Regulado en España con la Ley de 25 de Abril de 1986 que da respuesta a dos razones de máximo peso por provenir de nuestra Constitución, el reconocimiento en el artículo 43 y en el artículo 49 del derecho de todos los ciudadanos a la protección de la salud.

Otra novedad que aparece algún tiempo después es la denominada "Nueva Salud Pública" que en nuestro país llega con retraso, al revés de lo que sucedió con la nueva atención primaria, probablemente porque las condiciones eran distintas. La "nueva Salud Pública" es la salud pública de siempre pero queriendo abordar todos los factores que inciden en la salud. El discurso de que los factores que condicionan la morbimortalidad en los países desarrollados se deben en más de un 40% a los estilos de

vida, en un 25-30% a la biología humana, en un 20% al medio ambiente y solo en un 10% a la actuación de los servicios sanitarios, siendo real y aún poniendo el énfasis en los programas de prevención de la enfermedad y Promoción de la salud ha resultado poco beneficioso para los servicios sanitarios en general, y probablemente han sido los servicios específicos de salud pública los que se han visto más afectados, pues todo y todos han podido intervenir en estos campos [Oñorbe 1997].

7.2.- El Sistema Sanitario Español

Uno de los mayores retos de los sistemas sanitarios de los países industrializados es cómo hacer frente a las expectativas y demandas sociales generadas en su entorno. El desarrollo progresivo de los sistemas de asistencia sanitaria, junto al envejecimiento poblacional, al aumento del nivel de vida y a los avances científicos y tecnológicos en el campo de la medicina y en el tratamiento de las enfermedades han producido un considerable incremento en los recursos que los diferentes países destinan a sus sistemas sanitarios. A esta situación se han unido los cambios en los patrones de morbi-mortalidad y la aparición de nuevas enfermedades de carácter crónico e invalidante cuya resolución requiere la acción conjunta de los servicios sanitarios y sociales.

En este contexto es un tópico afirmar que los recursos en sanidad, como en otras muchas prestaciones sociales, son limitados y que las necesidades no solamente son ilimitadas, sino que tienden a crecer de modo exponencial, resultando además que este crecimiento está más ligado a las expectativas sociales que a la aparición de nuevas técnicas o procedimientos sanitarios.

A medida que los sistemas sanitarios han adquirido un mayor desarrollo, técnico y estructural, y que los ciudadanos disponen de una mayor información, las necesidades sanitarias tienden a expresarse en forma de demanda a mayor velocidad, de tal forma que siempre estarán por delante de la capacidad de respuesta de los sistemas sanitarios, y de los recursos que la sociedad les asigna. Ante esta situación, que se traduce en una tensión entre los recursos disponibles y las necesidades expresadas, el compromiso de la Administración Sanitaria y de los Gestores es incrementar, moderada y ordenadamente, el gasto sanitario, dentro de los límites que la riqueza del país pueda asumir, y, de otra

parte, mejorar la productividad de los recursos disponibles en el sistema.

Las respuestas que han dado a este problema los distintos países de nuestro marco social y económico básicamente se centran en poner límites a las prestaciones sanitarias, para disminuir el aumento de la demanda; en implantar criterios de eficiencia en la gestión para asegurar un mayor rendimiento de los recursos disponibles, y, por último, en implicar a los profesionales sanitarios en la asignación de recursos [Romay 1997].

Cada país ha optado por la solución que era más acorde a su esquema estructural y a la tradición y organización de su sistema sanitario. Así, por ejemplo, en Holanda, desde el año 1987, se ha acometido una profunda reforma de su estructura y de la financiación de sus prestaciones, así como de su organización interna. Suecia, ha avanzado por la vía de reformar su sistema e incrementar la participación del usuario en el coste de las prestaciones. Alemania, ha hecho un mayor énfasis en el control de los gastos, a través de determinadas prestaciones, como es el caso de la prestación farmacéutica. En el Reino Unido, se ha puesto en funcionamiento una fórmula organizativa basada en el control público de todos los escalones, desde la financiación hasta la decisión de qué tipo de servicios deben prestarse, iniciando un proceso que se conoce como de regulación de competencia en un mercado interno.

En España tenemos un Sistema Nacional de Salud, formado por los Servicios de Salud del Estado y de las Comunidades Autónomas, que representa una apuesta firme y comprometida de los poderes públicos en favor de un sistema equitativo, universal y solidario, que garantiza la cohesión social, en un entorno al que los ciudadanos confieren un valor extraordinario, como es el de la salud.

El grado de equidad conseguido convierte a nuestro sistema sanitario en un instrumento importante que contribuye de manera significativa al bienestar de la población, al permitir a los ciudadanos una enorme tranquilidad frente a las enfermedades que, como es sabido son ciertas, acumulativas en determinadas etapas de la vida e imposibles de predecir.

Dentro de este contexto la eficiencia de los servicios es vital para el buen funcionamiento de la sanidad pública. La introducción, en los hospitales públicos, de

técnicas de gestión originarias del mundo de la empresa privada permitirá mejorar áreas como la gestión de compras, el manejo de los "stocks", la organización de los servicios centrales o los sistemas de información, permitiendo avances en la productividad de los centros.

Otro aspecto a tener en cuenta es que nuestro sistema asistencial público se caracteriza por responder a un modelo organizativo de estructuras rígidas, burocráticas, y centralizadas, y que como demuestra la experiencia acumulada a nivel nacional e internacional tiene serias limitaciones para alcanzar niveles deseables de eficiencia y rentabilidad social, presentando además problemas de acceso de los usuarios a determinadas prestaciones, tal y como se evidencia al analizar los flujos de ingresos hospitalarios o la evolución de las listas de espera.

Frente a este panorama, la empresa sanitaria debe responder con un proceso de innovación continua que permita la transición desde estructuras sanitarias integradas verticalmente hacia modelos más horizontales, descentralizados, orientados hacia el usuario y comprometidos con la calidad y eficiencia de resultados. Las razones de este cambio son el recorte en el crecimiento del gasto sanitario junto a la propia dinamicidad de otros factores críticos del entorno (demográficos, epidemiológicos, políticos, tecnológicos, culturales, garantías sociales, ...). Este proceso de cambio debe realizarse de manera ordenada y teniendo en cuenta que un hospital es una organización compleja de servicios, y por lo tanto, debe seguir las metodologías y herramientas que están dando buenos resultados en el mundo empresarial [Silva 1998].

8.- GESTIÓN HOSPITALARIA

La empresa sanitaria como organización compleja de servicios se encuentra inmersa en un proceso de innovación continua. Toda estrategia de cambio debe comenzar con un diagnóstico de la empresa, analizando los factores internos y externos que serán decisivos para que el proyecto de transformación consiga los objetivos previstos de calidad y eficiencia. Una vez establecido el escenario global, se ha de determinar a qué nivel se va a implementar la innovación. Finalmente se llevará a cabo dicho cambio. En todas las fases de desarrollo, es pertinente contar con el apoyo de

técnicas y herramientas para el análisis de decisiones.

Pero ¿por qué la necesidad de cambio?. Los hospitales, al igual que el resto de las organizaciones, se han estructurado siguiendo el principio de la división del trabajo descrita por Adan Smith en "The Wealth of Nations" en 1776.

Smith, filósofo y economista, reconoció que la tecnología producida durante la revolución industrial había creado una oportunidad sin precedentes para incrementar la productividad disminuyendo los costos, pero para ello se debía utilizar el principio de la división del trabajo. Este principio dice que un número de trabajadores especializados, cada uno de ellos realizando un único paso del proceso de producción, pueden generar más resultados que el mismo número de trabajadores cada uno haciendo el proceso completo.

Las compañías se fueron estructurando teniendo como base este principio, y creando grupos de trabajadores especializados con gestores que coordinaban todo el trabajo. Pero los procesos cada vez son más complicados, el número de tareas a realizar ha crecido y por lo tanto la gestión es más difícil. Esto junto con los cambios que han ocurridos en el entorno de las empresas, han originado que este tipo de organización no sea la adecuada para las compañías actuales.

Los hechos que han cambiado en el entorno empresarial son los siguientes:

*Los clientes.- Los clientes exigen un producto con un precio y en un tiempo determinado. No hay clientes conformistas, por lo que la empresa debe centrarse en ellos para tenerlos satisfecho.

*La competencia.- Las empresas que produzcan un mejor producto a un precio más bajo serán las que vendan el producto.

*Los cambios.- Los cambios han llegado a ser tan persistentes como generales. Con la globalización de la economía crece el número de competidores y con ello las innovaciones en el número de productos y servicios introducidos en el mercado. Además, la rapidez de los cambios tecnológicos también originan innovaciones en el mercado.

Por ello, la estructuración de las compañías basadas en organizar el trabajo según

una división del mismo ha quedado obsoleta. El trabajo orientado a la tarea en el mundo actual no genera buenos resultados. Las compañías se deben organizar alrededor de sus procesos. La reingeniería es el mecanismo para realizar estos cambios [Hammer 1993].

Un elemento decisivo en todo el proceso de cambio es el uso creativo de la tecnología de la información. Particularmente centrándonos en el mundo de la sanidad, debido a la gran cantidad de información requerida por estas organizaciones, se están desarrollando sistemas de información que permitan el almacenamiento y manipulación de los datos de forma estructurada. Esto es, se diseñan sistemas centrándose en los problemas a resolver y aislándose de la implementación concreta. Para ello se realizan modelos utilizando los "patterns" (un patrón es un elemento que ha sido útil en la resolución de un problema en un contexto concreto), de forma que la implementación sea inmediata y que permita la reutilización de estos componentes o patrones en otros programas [Fowler 1997].

En este trabajo se pretende avanzar en el desarrollo de los sistemas de gestión de la información. El análisis del flujo de información, como la primera fase en cualquier sistema en desarrollo, se centra en el uso de la Simulación como una herramienta de adquisición del conocimiento imprescindible en sistemas complejos. Este mecanismo facilitará la creación de sociedades (hospitales) virtuales donde estudiar el comportamiento emergente de los mismos, esto es, se pretende crear un sistema de información que genere la dinámica de la organización en estudio.

Esta primera fase de análisis genera una gran cantidad de información que puede resultar confusa si no es analizada y no se le proporciona al usuario ordenadamente. En este proceso es donde se hace patente la necesidad del uso de técnicas de IA para automatizar el tratamiento de toda la información que se produce en cualquier organización.

Por lo tanto, el objetivo de este trabajo es proporcionar herramientas de simulación e IA que ayuden en la consecución del proceso de cambio necesario en todas las organizaciones sanitarias.

Como la organización en estudio son los hospitales generales, y los usuarios del sistema de gestión de la información el personal de gerencia de dichos hospitales, se

desarrolla en el siguiente capítulo una herramienta de simulación que le permita al gerente un conocimiento más exhaustivo del sistema con el que trabaja.

En el tercer capítulo se diseña un SBC que automatiza el análisis de los datos obtenidos y que proporciona las recomendaciones necesarias, a la gerencia del hospital, para conseguir los objetivos de calidad y eficiencia exigidos.

Capitulo II
Modelado y simulación para la toma de decisiones en la gerencia hospitalaria

CAPITULO II

MODELADO Y SIMULACIÓN PARA LA TOMA DE DECISIONES EN LA GERENCIA HOSPITALARIA

1.- INTRODUCCIÓN

Los modelos son un mecanismo potente para explicar la realidad, es por ello que los científicos en las distintas áreas los utilizan para entender los hechos que estudian. Sin embargo en el estudio de sistemas sociales [Aracil 1986a], esto es, sistemas en los que aparecen colectividades, el uso de modelos no era práctica común. Con la llegada de ordenadores y lenguajes de programación potentes se hace posible la construcción de modelos computacionales para la simulación de los procesos sociales. Esta herramienta consiste en representar el sistema social como un programa de ordenador, permitiendo así experimentar con sociedades artificiales [Gilbert 1993].

En este trabajo se plantea el estudio de la gestión hospitalaria. La importancia del estudio de este sistema se refleja en la repercusión social que tiene, ya que el bienestar de una comunidad en el capítulo de salud está en función del rendimiento de sus

hospitales. El hospital es un sistema social cuyo objetivo fundamental es el correcto tratamiento del colectivo de pacientes para mejorar su estado de salud. Los mecanismos existentes para conseguir este objetivo es la correcta utilización de los recursos materiales y humanos, estos últimos son el colectivo de trabajadores del hospital cuya gestión es una pieza básica para el adecuado funcionamiento de la organización. Asimismo, el hospital es un sistema complejo debido al gran número de elementos que lo forman así como, el número de interrelaciones que hay entre ellos.

El estudio de este sistema complejo se desarrolla en los siguientes dos pasos:

a) Análisis: estudio y definición del problema a resolver.

b) Modelado y Simulación: resolución del problema mediante el uso de técnicas de simulación computacional.

2.- ANÁLISIS DE UN HOSPITAL

El hospital, independientemente de su propiedad pública o privada, es una empresa de servicios. Como tal ha evolucionado más lentamente que otras organizaciones en buena medida por la falta de definición operativa de su producto. La gestión hospitalaria ha emergido en los últimos tiempos y su importancia será aún mayor en los próximos años [Silva 1998].

El primer paso en el estudio de la gestión hospitalaria es identificar y conocer de forma global los flujos que configuran la estructura del hospital y obtener un diseño completo y preciso de los procesos críticos (procesos básicos relacionados con el servicio que se ofrece sin los cuales la organización no podría seguir sobreviviendo) y los secundarios, así como sus interrelaciones [Fowler 1997].

Este análisis riguroso del hospital será el primer paso para el diseño e implementación de una herramienta de gestión que ayude en la obtención de los objetivos de calidad y eficiencia deseados. Debido a que en las organizaciones de servicios sanitarios se requiere una información integrada, intensiva y disponible por parte de los equipos multidisciplinares, se desarrollará un sistema de gestión de la información que permita alcanzar estos objetivos [Moreno 1996b].

Un hospital es un establecimiento dedicado al tratamiento de enfermos. El hospital general es aquel que se organiza como suma de especialidades médicas y quirúrgicas, divididas en distintas unidades clínicas denominadas "servicios". Los enfermos se distribuyen entre las unidades clínicas según la patología que padezcan.

Existen departamentos para las especialidades médicas (cardiología, gastroenterología, reumatología y pediatría especializada), las especialidades quirúrgicas (neurocirugía, cirugía pulmonar, oncología, oftalmología, urología, otorrinolaringología, traumatología y ginecología), y las técnicas auxiliares (radiología, radioterapia, analítica, medicina nuclear, ecografía, tomografía axial computerizada, etc.), a éstas últimas se les denominan "servicios centrales".

En los hospitales existen unidades especializadas (UCI, URPA) donde se ingresa a los enfermos graves y postoperatorios que requieren una vigilancia constante o cuidados especiales.

Otro servicio es el de urgencias que atiende las necesidades de asistencia urgente, tanto del propio hospital como de pacientes externos.

Se dispone también de consultas externas para los servicios de asistencia ambulatoria, y a la que acuden los pacientes que después de ser dados de alta necesitan un tratamiento posterior, y aquellos otros que no necesiten ser internados. Estos servicios están formados por consultorios.

Para el control de este sistema, el gerente de un hospital se enfrenta diariamente a la toma de múltiples decisiones, tanto en el campo administrativo como sanitario, que afectan a la buena marcha del hospital. Las decisiones se deben tomar a partir de un gran número de variables que se actualizan semanalmente y que tienen un alto grado de interrelación, por lo que a la rapidez con que hay que tomarlas hay que añadir la limitada capacidad humana para analizarlas conjuntamente. Se hace por lo tanto patente la necesidad de elaborar una herramienta de ayuda a la toma de decisiones en la gerencia hospitalaria [Moreno 1997a].

La implementación de la simulación del hospital se presenta como el mecanismo necesario para entender dicha organización. Esta herramienta permitirá por un lado obtener información sobre el estado del sistema mediante gráficos e informes, esto es,

elaborará y preparará los datos del hospital para que sean fácilmente analizados por el personal de gerencia. Por otro lado, la simulación del hospital permitirá ensayar diferentes acciones de control para conocer cuál será la evolución del sistema, y así determinar la mejor decisión a tomar frente a las distintas situaciones [Moreno 1996a].

Una vez conocida la importancia de una herramienta de simulación para la ayuda a la gerencia hospitalaria, vamos a modelar el sistema como siguiente paso en la construcción de dicha herramienta.

3.- MODELADO DEL HOSPITAL

Debido a que el hospital es un sistema muy complejo, para su modelado recurrimos al método de reducción del problema (divide y vencerás), y dividimos el sistema en un conjunto de subsistemas menos complejos. Luego se modelará cada subsistema identificado y la adecuada unión de estos modelos generará el modelo global del hospital [Moreno 1997b].

Para identificar los distintos subsistemas que intervienen en el hospital recurrimos a la funcionalidad de los mismos. Resultando el hospital formado por los tres subsistemas que muestra la figura 2.1.

El subsistema 1 representa el flujo de pacientes en el hospital, tiene como entrada no controlada (perturbación) la llegada de pacientes y como entrada de control las distintas acciones que realiza el gerente para llevar el hospital al estado deseado. La salida es el conjunto de variables que indican el estado del hospital. Debido a que la finalidad de un hospital es atender a todos los pacientes que llegan, la variables que nos indican su correcto funcionamiento son el tiempo medio y las desviaciones respecto a la media que deben esperar los pacientes para ser atendidos, es por ello que el estado de este sistema, desde el punto de vista de la gerencia, vendrá definido por el tiempo medio de todas las colas generadas.

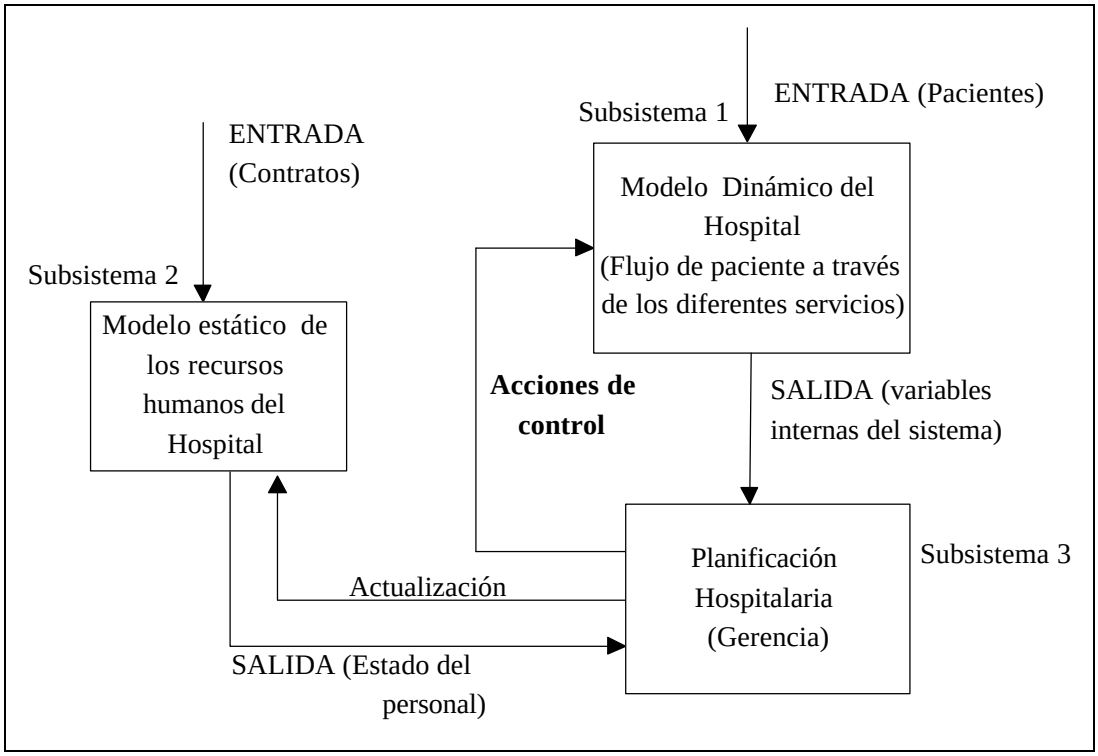


Figura 2.1.- Descripción funcional del hospital Nª Sra. de la Candelaria

El subsistema 2 es el modelo de los recursos humanos de los que se dispone para atender a los pacientes. Como entrada recibe las altas y bajas de personal que se realizan por el subsistema 3 para corregir mal funcionamientos del hospital. Como salida produce el conjunto total de recursos disponibles en un momento determinado.

El subsistema 3 describe la gerencia hospitalaria. Recibe como entrada los valores de las distintas colas de los servicios que se originan por la escasez de recursos para atender a un volumen determinado de pacientes. Además, recibe como entrada los recursos humanos de los que se dispone para mantener o mejorar el rendimiento del hospital. Como salida produce las acciones de control necesarias (redistribución de recursos) para corregir estados no correctos del sistema. El modelado y simulación de la gestión es realizado mediante la construcción de un Sistema Basado en el Conocimiento que se describe en el capítulo III.

4.-FLUJO DE PACIENTES EN EL HOSPITAL

La dinámica del hospital, esto es, su evolución con el tiempo va a venir determinada por la llegada de pacientes al sistema. Dependiendo de la patología que tengan estos pacientes necesitarán un conjunto de recursos específicos. Una escasez en este conjunto de recursos o un aumento en el número de pacientes con la misma patología harán que el estado del hospital no sea el adecuado. La señal que indica que el rendimiento del hospital no es el deseado es el aumento por encima de un umbral de referencia del tiempo medio de espera de un paciente para ser atendido. Cuando esto ocurre, el gerente del hospital debe determinar la mejor distribución de recursos para minimizar el problema existente. La implementación de un programa de ordenador que simula el flujo de pacientes por el hospital será una herramienta de ayuda a la toma de decisiones porque el gerente podrá experimentar distintas acciones de control y analizar sus efectos.

4.1.- Modelado

El subsistema 1 de la figura 2.1 representa el modelo dinámico del hospital, formado por un conjunto de subsistemas y sus interrelaciones. Para determinar los subsistemas que lo forman observamos el flujo de individuos a través del hospital, esto es, los subsistemas serán las distintas unidades a las que los pacientes solicitan sus servicios. Mediante el esquema de la figura 2.2 se observa como circulan los pacientes por los distintos unidades del hospital Nra. Sra. de la Candelaria, aunque el modelo es genérico para cualquier otro hospital general.

La entrada no controlada de este subsistema es la llegada de pacientes al hospital que se produce cuando en Consultas Externas de algún Centro de Salud determinan que el paciente debe ser atendido en el hospital. Dependiendo del tipo de patología que presente el usuario necesita ser tratado en Consultas Quirúrgicas o Consultas Médicas.

El primer paso al llegar al hospital es ser atendido en Primera Consulta (médica o quirúrgica) donde realizará su historial, y tras ser observado se le determinan el conjunto de pruebas que debe hacerse en los Servicios Centrales para volver a ser examinado.

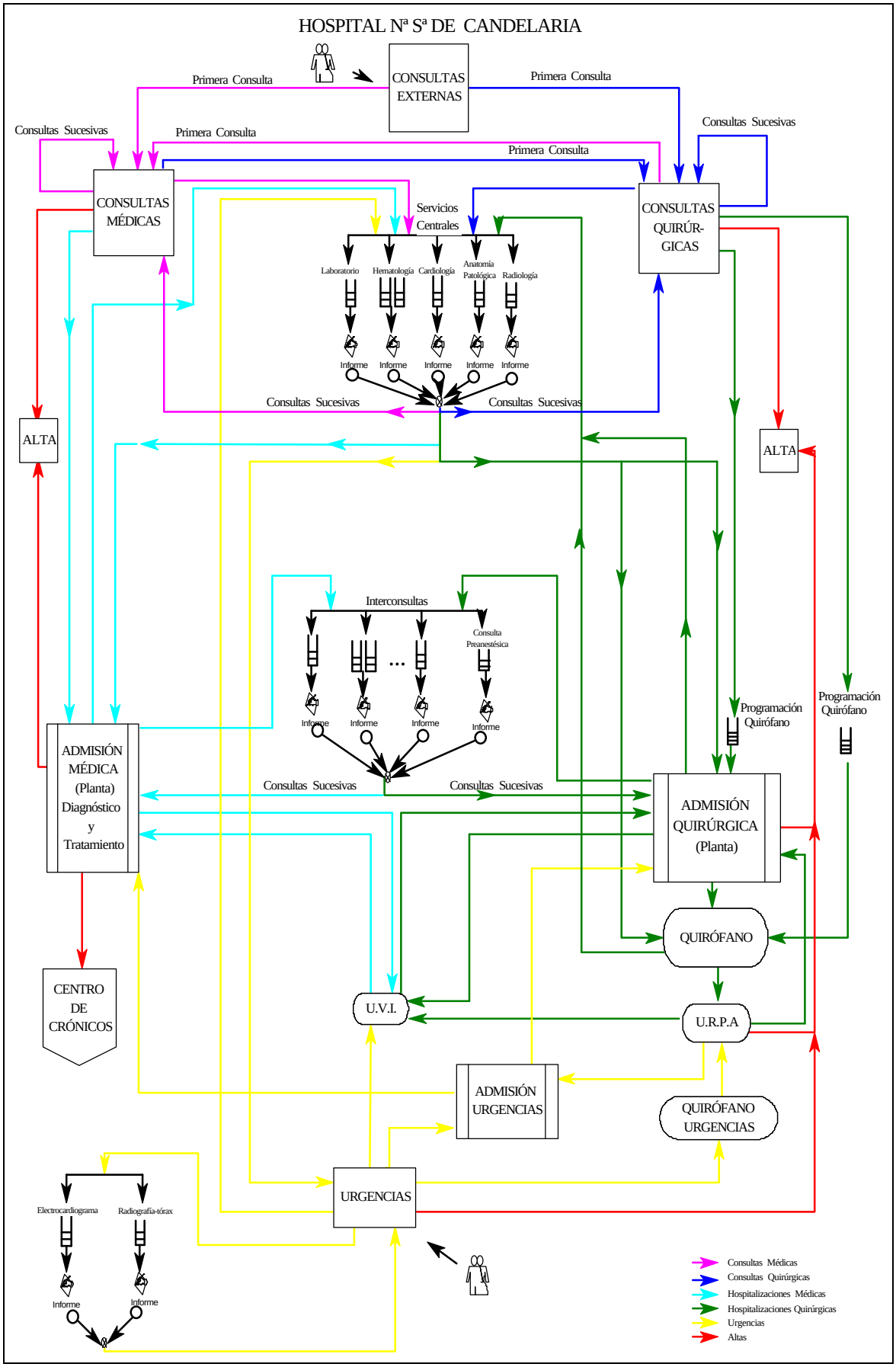


Figura 2.2.- Flujo de los pacientes en el Hospital Nª Sra. de la Candelaria

Las distintas pruebas clínicas (análisis, radiografías, electroencefalogramas, ...) se llevan a cabo en los Servicios Centrales, que son unos servicios que se caracterizan por no tener sus propios pacientes, es decir, que no tratan pacientes con determinadas patologías, sino que atienden a los enfermos procedentes de todos los servicios (médicos y quirúrgicos) del hospital.

Cuando el usuario ya tiene el resultado de las pruebas marcadas, debe volver a consulta pero el proceso ahora es el de una Consulta Sucesiva, donde estudiarán los resultados de las pruebas y determinarán el tratamiento a seguir por el enfermo, o si necesita realizarse nuevas pruebas. El paciente debe volver a Consultas Sucesivas cuando finalice el tratamiento o cuando obtenga los resultados de las nuevas pruebas, y seguirá en este bucle hasta que se le da de alta o sea ingresado en Planta.

Si el paciente es ingresado debido a una patología médica, debe pasar por admisión médica donde le indicarán cuando puede entrar en la Planta Médica para seguir el tratamiento adecuado. Después de su tratamiento el paciente es dado de alta y sale del hospital. Si su enfermedad pasa a ser crónica el paciente sale del hospital hacia un Centro de Crónicos.

Si el paciente es ingresado por causa de una patología quirúrgica, pasa por admisión quirúrgica para conocer cuando entra en Planta Quirúrgica para ser operado. En planta permanece un tiempo prequirúrgico durante el cual se le realizan las pruebas necesarias para preparar su entrada en quirófano. Luego, en quirófano permanece el tiempo necesario tras lo cual pasa a URPA (Unidad de Reanimación PostAnestésica). De URPA el paciente regresa a Planta donde permanecerá el tiempo postquirúrgico necesario para su total recuperación, después de la cual al paciente se le dará el alta y saldrá del hospital.

En cualquier momento, durante el recorrido del paciente por los distintos servicios del hospital, si éste sufre un empeoramiento de su enfermedad es ingresado en la UVI (Unidad de Vigilancia Intensiva). Cuando se recupera es llevado a planta para seguir el camino comentado anteriormente.

La llegada al hospital de los pacientes también se puede realizar por Urgencias. En este caso al paciente se le realizan las pruebas necesarias (algunas de ellas se llevan a cabo en Urgencias y otras en los Servicios Centrales) y después de indicarle un tratamiento se le da de alta. En el caso de que el paciente tenga que ser ingresado en planta pasa por Admisión de Urgencias donde lo direccionan a la Planta Quirúrgica o Médica que corresponda para seguir el flujo habitual de los pacientes ingresados. El tercer caso que se puede presentar en Urgencias es tener que operar al paciente, para ello utilizan el Quirófano de Urgencias y de ahí a URPA; después de su recuperación postanestésica pasa a planta parar seguir el camino habitual de los enfermos en planta.

4.2.- Simulación

La simulación del flujo de pacientes por el hospital, nos servirá para conocer cómo evoluciona el hospital con el tiempo. Para ello, tenemos que convertir el diagrama del flujo de pacientes a un modelo algorítmico (un programa) que nos permita conocer para situaciones específicas el estado de las distintas colas.

El subsistema en estudio es un sistema de eventos discretos que cambia de estado cada vez que se produce un nuevo evento [Gordon 1978]. Para este sistema no disponemos de una ecuación que defina el estado del mismo, sino que debemos describir el algoritmo o flujo de acciones que generan los eventos que describen la evolución del sistema. Para simular esta dinámica, necesitamos encontrar una herramienta que nos permita realizar experimentos sobre el hospital de forma que se pueda adquirir conocimiento y determinar las acciones de control a realizar para mejorar su rendimiento.

Previamente a la selección de la herramienta de simulación debemos tener en cuenta los aspectos del problema que influyen notablemente en la elección de la misma [Pressman 1988]:

- En el sistema, los pacientes son considerados elementos discretos y su flujo a través del mismo es representado como un sistema de eventos discretos, sistema dinámico que cambia de estado cuando ocurre un evento (llegada de

un paciente al hospital, un paciente que es atendido en consulta, entra un paciente a quirófano, ...).

- Los pacientes deben ser caracterizados por un conjunto de atributos tales como tipo de patología, pruebas clínicas a realizar, etc. Los valores de estos atributos no serán fijos sino que variarán durante el tiempo de simulación.
- Necesitamos conocer en cada instante la situación de todos los pacientes del hospital por lo que debemos realizar una microsimulación (simulación microscópica).
- Debido a que se pretende hacer experimentos con el modelo, éste debe permitir que la simulación se realice a intervalos de tiempo, y en cualquier momento se pueda volver a una situación anterior para probar acciones de control diferentes.
- En todo momento se debe conocer cual es el estado del sistema para poder actuar en consecuencia.
- Asimismo, la simulación nos debe suministrar información referente a las colas generadas en el hospital ya que son las variables que nos indican el rendimiento del sistema.

Junto a las características propias del problema las especificaciones que se necesitan en cualquier herramienta de simulación de eventos discretos son las siguientes:

- Gestión del tiempo de simulación de manera sencilla
- Fácil manejo de diferentes sistemas de colas (con prioridad, múltiples servidores, ...)
- Generación de estadísticas
- Posibilidad de interface con otras aplicaciones, tales como *shell* de sistemas expertos, bases de datos, etc.
- Adecuada respuesta en el tiempo de ejecución
- Interface amigable y facilidad en la depuración del programa

Hemos estudiado diferentes herramientas buscando que cumplan las características anteriores. Básicamente se pueden clasificar estas herramientas en dos tipos:

1.- Herramientas de simulación orientadas al evento (basadas en las Redes de Petri) [Silva 1985].

2.- Herramientas de simulación orientadas al proceso.

4.2.1.- Redes de Petri

Las redes de Petri (RdP) se utilizan con el objeto de modelizar el comportamiento dinámico de sistemas discretos [Brams 1986a] [Brams 1986b]. Se puede describir una RdP de dos formas equivalentes.

La primera es el formalismo matemático que define una RdP como la cuádrupla R= < P,T; Pre, Post > , siendo P un conjunto finito de cardinal n, T un conjunto finito de cardinal m, Pre y Post son dos aplicaciones de P x T $\Rightarrow$ $\mathbb{N}$. Los elementos de P se llaman "lugares", los de T "transiciones" y las aplicaciones Pre y Post aplicaciones de incidencia previa y posterior.

La segunda forma de definir una RdP es con un formalismo gráfico que tiene dos partes, una estática y otra dinámica. La parte estática es un grafo orientado en el que intervienen dos clases de nodos, los lugares (representados por circunferencias) que definen una condición y las transiciones (representados por segmentos rectilíneos) que describen un evento, unidos alternativamente por arcos. Un arco une un lugar con una transición, o viceversa, pero nunca dos transiciones o dos lugares. La parte dinámica de la RdP está formada por las marcas y las reglas de disparo de las transiciones. Un lugar puede contener un número finito o nulo de marcas que se representan por un punto en el interior del círculo correspondiente al lugar. El conjunto de marcas asociados en un instante dado a cada uno de los lugares constituye el marcado de la red y representa el estado del sistema. Para definir la dinámica del sistema mediante RdP debe existir una regla que indique cómo puede evolucionar el marcado. Para ello debemos primero definir lugar de entrada, lugar de salida y transición sensibilizada.

- un lugar *p* se dice lugar de entrada de una transición *t* si existe un arco orientado de *p* hacia *t*;

- un lugar *p* es lugar de salida de la transición *t* si existe un arco orientado de *t* hacia *p*

- una transición está sensibilizada si todos los lugares de entrada están marcados

La regla de evolución de marcado, y por lo tanto que nos permite definir la dinámica del sistema, es la siguiente:

Una transición sensibilizada es disparada si el evento que le está asociado se verifica. El disparo de una transición consiste en quitar una marca a cada uno de los lugares de salida. La figura 2.3 ilustra los conceptos de sensibilización y disparo de una transición.

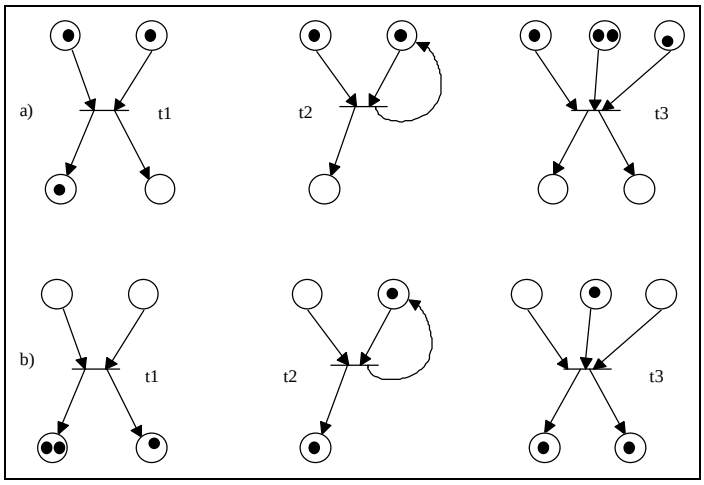


Figura 2.3.- a) Ejemplos de transiciones sensibilizadas. b) Marcados obtenidos después de sus disparos

Las propiedades básicas que presenta una RdP son las siguientes:

a) Una transición es viva, para un marcado M_0 si para todo marcado M que se pueda alcanzar a partir del marcado inicial M_0 existe un marcado M' sucesor de M a partir del cual se puede disparar esa transición. Una RdP es viva, para un marcado dado, si todas sus transiciones son vivas para ese marcado.

Una red de Petri no viva es aquella que tiene alguna transición no viva, esto es, a partir de un marcado hay alguna transición que no puede evolucionar más (está bloqueada) esto permite sospechar que el modelo del sistema objeto de estudio es incorrecto.

b) Para un marcado inicial dado, una RdP es binaria si cualquier marcado alcanzable es tal que ningún lugar posee más de una marca.

Si una RdP es no binaria, el número de marcas de un lugar puede crecer continuamente debido a que no tiene ninguna restricción sobre la cantidad de marcas posibles, y por ello se dice que es una RdP ilimitada puesto que el marcado en ese lugar no tiene límite finito. Si la RdP que modela un sistema es ilimitada cabe sospechar que el modelo no es correcto. La limitación indica que el número de estados internos del sistema que se modela es finito.

c) Se dice que una RdP es conforme si es binaria y viva.

d) Dos o más transiciones simultáneamente sensibilizadas están en conflicto si descienden de un mismo lugar y éste no dispone de un número de marcas suficientes para dispararlas simultáneamente, figura 2.4. Un conflicto se dice efectivo si los eventos asociados a las transiciones en conflicto se verifican simultáneamente. Un conflicto efectivo corresponde a una ambigüedad en la descripción. Ningún modelado correcto puede poseer conflictos efectivos.

Figura 2.4.- En ambos casos t1 y t2 están en conflicto

4.2.1.1.- Extensiones de las Redes de Petri Ordinarias

Hasta aquí hemos definido las redes de Petri ordinarias. Pero cuando se modeliza un sistema real sucede con frecuencia que el tamaño de una red resulta ser muy importante [Valette 1994]. Este fenómeno de crecimiento puede deberse en particular al empleo repetido de ciertas formas de subredes que modelizan un mecanismo específico

del problema tratado. Entonces puede resultar útil enriquecer la definición de una red para que tal mecanismo se convierta en una operación primitiva de un nuevo modelo, al cual consideramos entonces como una abreviación de una red. Sin cambiar la potencia algorítmica, ni las propiedades, una abreviación de red permite una economía de escritura y de lectura que es a menudo necesaria para la descripción de grandes sistemas [Brams 1986a] [Brams 1986b].

Estos tipos de RdP son las siguientes:

A) Redes de Petri generalizadas

Las RdP generalizadas son análogas a las RdP ordinarias exceptuando que cada arco se etiqueta con un entero natural que se denomina peso del arco. Por convenio, un arco no etiquetado posee un peso unitario. La regla de evolución del marcado para una RdP generalizada es la siguiente:

Disparar una transición sensibilizada t es la operación que consiste en eliminar tantas marcas del lugar de salida como indique el peso del pre arco (arco orientado del lugar a la transición), y añadir tantas marcas al lugar de entrada como indique el peso del post arco (arco orientado de la transición al lugar de salida).

B) Redes de Petri con capacidad limitada

Una extensión de las RdP generalizadas y por lo tanto de las RdP ordinarias son las RdP con capacidad limitada. Este tipo de redes facilitan la tarea de modelización de sistemas complejos, especialmente si existen numerosos almacenes, memorias, etc. (su capacidad siempre estará limitada físicamente).

Una red de Petri con capacidad limitada es una quíntupla <P, T, ***á***, ***ß***, ***Ã***>, donde R=<P, T, ***á***, ***ß***> es una red generalizada y ***Ã*** es una función que asocia a cada lugar su capacidad; es decir, el máximo número de marcas que puede contener.

Esta noción de capacidad modifica las condiciones de disparo de una transición de manera que el marcado que resulte no exceda la capacidad de los lugares. Es decir, para disparar una transición, además de cumplirse las pre y post condiciones, se tiene que verificar que las marcas no excedan la capacidad del lugar que van a ocupar.

C) Redes de Petri con arcos inhibidores

Una RdP con arcos inhibidores es una red a la que se le añaden unos arcos que sólo parten de lugares y van a transiciones, denominados arcos inhibidores. La regla de disparo de una transición en una red de este tipo exige que estén desmarcados todos los lugares que se encuentren unidos a la transición mediante un arco inhibidor, figura 2.5.

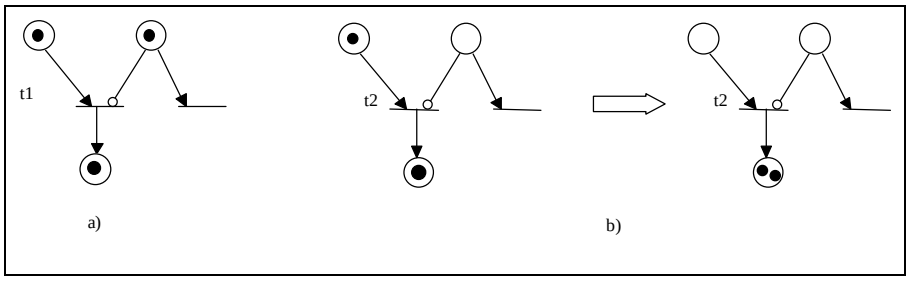


Figura 2.5.- a) La transición t1 no está sensibilizada. b) Disparo de la transición sensibilizada t2.

D) Redes de Petri coloreadas

En una red de Petri coloreada, cada marca puede portar un color que la identifique. A cada lugar y a cada transición se le asigna un conjunto de colores. Una transición puede dispararse respecto a cada uno de sus colores. El disparo de una transición elimina y añade marcas como en las RdP, pero respetando la dependencia funcional especificada entre el color de disparo de la transición y los colores de las marcas. El color de cada marca puede ser cambiado por el disparo de una transición.

Las RdP coloreadas son útiles en la condensación de la descripción y análisis de sistemas en los que se identifican diversos subsistemas con estructura y comportamiento similares, pero que trabajan en paralelo.

E) Redes de Petri temporizadas

El tiempo se puede introducir en una red de Petri de dos formas, tiempo en las transiciones o tiempo en los lugares.

Una RdP temporizada es un par <R, Z> tal que R=<P, T, ***á***, ***ß***> es una red generalizada y Z es una función que asigna un número real no negativo z_i a una transición o a un lugar de la red:

e.1.-) El tiempo se asocia con las transiciones si el número real no negativo z_i es asignado a cada transición de la red, donde $z_i = Z(t_i)$ que se denomina tiempo de disparo de la transición t_i. La regla de evolución del marcado es idéntica a una RdP generalizada. La única cuestión a tener en cuenta es que el disparo de t_i dura z_i unidades de tiempo.

e.2.-) El tiempo se asocia con los lugares si el número real no negativo z_i es asignado a cada lugar de la red. Una marca se puede encontrar en dos estados: dispuesta e no dispuesta. Las reglas de evolución del marcado son idénticas a las de las RdP si las marcas están dispuestas. Si las marcas no están dispuestas es como si no existieran en cuanto a la evolución de la red. Al pasar a un lugar una marca, entra en estado no dispuesto y pasa a estado dispuesto después de $z_i = Z(p_i)$ unidades de tiempo.

Esta clase de redes se utilizan en estudios de rendimiento de sistemas.

F) Redes de Petri con prioridades

Las RdP con prioridades se utilizan cuando se debe elegir entre varias transiciones sensibilizadas. Por ejemplo, si varios procesos necesitan un mismo recurso, una estrategia posible consiste en atribuir este recurso al proceso más lento que tiene necesidad. En este caso, es preciso dar un orden a las transiciones.

G) Redes de Petri con predicados

Las RdP con predicados permiten asociar a cada marca un conjunto de atributos, esto es, cada marca puede poseer un conjunto de colores en vez de un único color como ocurre con las RdP coloreadas. Así, una marca es susceptible de representar una n-upla de parámetros. Los arcos de las transiciones pueden etiquetarse con n-upla de constantes o de variables que precisen los parámetros de las marcas a utilizar o a producir. Además, a cada transición se le asocia un predicado (expresiones acerca de los parámetros) que indica para qué valores de los parámetros la transición es sensibilizada (si una variable no figura en un predicado puede tomar cualquier valor para la sensibilidad de esa transición).

4.3- Simulación del Flujo de Pacientes por el Hospital Basado en Redes de Petri

Para simular un sistema de eventos discretos podemos recurrir a la metodología orientada al evento [McHaney 1991]. Con este mecanismo es necesario identificar todos los eventos diferentes que puedan ocurrir en el sistema. Un evento se define como una ocurrencia instantánea que puede cambiar el estado del sistema.

Para simular un sistema siguiendo esta metodología se recurre al uso de las RdP debido a que éstas poseen un formalismo matemático que facilita la implementación y validación de la simulación.

En el modelado de la dinámica del hospital mediante una RdP, asociamos a cada paciente una marca que transita de unos lugares a otros, donde cada lugar representarían los diferentes servicios por los que pasan los pacientes. Las transiciones que hacen que los pacientes fluyan de un lugar a otro estarían influenciadas por la presencia de otras marcas en otros lugares que representarían los recursos que necesita el paciente para evolucionar. Asimismo, para representar simultaneidad en el uso de determinados recursos, éstos deben ser representados por transiciones, un ejemplo serían los consultorios. La figura 2.6 representa las transiciones en dos consultorios por los que pueden pasar los pacientes que se encuentran en el lugar etiquetado con el nombre pacientes, siempre que exista una marca en el lugar etiquetado enfermera y otra en el lugar etiquetado médico.

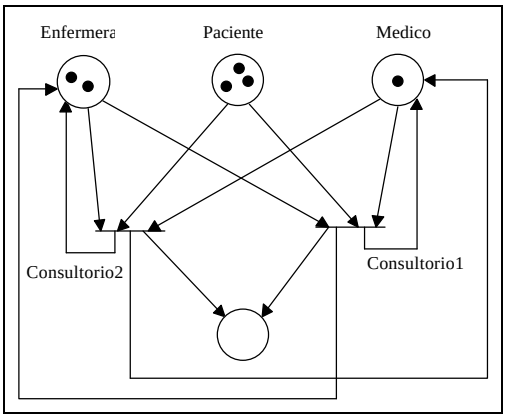


Figura 2.6.- Representación del flujo de pacientes por consulta mediante una RdP

Debido a que según la patología de cada paciente éste transitará por distintos servicios del hospital se utilizaron las RdP coloreadas para poder representar distintos tipos de pacientes. La implementación se realizó con un software de dominio público llamado SIMNET [Garbe 1991]. Este programa presenta las siguientes características:

- Trabaja para PC bajo el sistema operativo Dos.

- Presenta un entorno gráfico que permite la construcción de las RdP muy fácilmente.

- Se pueden diseñar una gran variedad de RdP tales como las generalizadas, con capacidad, con tiempo en las transiciones, con arcos inhibidores y coloreadas.

- Fácil depuración de la RdP permitiendo la traza paso a paso de la simulación.

En conclusión, hay que mencionar que es un software de fácil manejo para aplicaciones sencillas, pero que cuando el número de elementos del sistema se dispara, el trabajo con SIMNET se hace bastante engorroso porque no está totalmente depurado el programa y se producen bloqueos del mismo en determinadas ocasiones.

El modelado del hospital mediante RdP coloreadas no es posible porque los pacientes deben disponer de un conjunto de atributos tales como las pruebas a realizar. Se hace necesario utilizar una RdP que permita asociar a cada marca (paciente) el conjunto de atributos. Además, se debe poder asociar a cada transición acciones, para ello se utilizan las RdP con predicados.

SIMNET no permite el uso de RdP con predicados, así que se utilizó otra herramienta de desarrollo de RdP el CPN (Color Petri Net). Este programa que corre bajo Unix, ofrece una mayor flexibilidad para diseñar el sistema puesto que permite implementar todas las extensiones de las RdP. Por lo tanto facilita el asociar atributos a las marcas y acciones a las transiciones. El problema que plantea esta herramienta es la dificultad de uso ya que la definición de la RdP se realiza en un lenguaje propio del CPN.

El uso de RdP para simular el flujo de pacientes en el hospital presenta los dos problemas siguientes:

1.- El hospital es un sistema muy complejo y aunque utilizáramos RdP generalizadas la estructura que obtenemos es demasiado complicada, lo que dificulta trabajar con el modelo.

2.- El hecho de tener que asociar determinados recursos (por ejemplo, consultorios) con transiciones que son parte de la estructura estática del modelo de RdP, nos obliga en cada experimento, cuando se redistribuyen recursos para obtener mejores resultados, a variar la estructura estática de la red. Esto dificulta cada proceso de simulación.

Estas razones nos han llevado a una simulación orientada al proceso [McHaney 1991] para la implementación del flujo de pacientes por los distintos servicios del hospital.

4.4.- Simulación Orientada al Proceso

Un proceso es una secuencia ordenada en el tiempo de eventos interrelacionados separados por el paso del tiempo. Esta secuencia describe el paso de un ítem (individuo, información, ...) a través del sistema. Para utilizar una aproximación orientada al proceso hay que definir la secuencia de pasos (eventos o serie de eventos) para cada transacción que ocurre en el sistema, por lo que a este tipo de simulación se le denomina microsimulación. Esta secuencia de pasos se puede representar utilizando un diagrama de flujo (flowchart). Los objetos requeridos por una transacción durante su movimiento a través del sistema son definidos como recursos.

La simulación orientada al proceso o basada en el proceso se usa mucho en la simulación de sistemas sociales (ej. un hospital). Cada parte de un sistema tiene un conjunto de reglas que gobiernan el comportamiento de cada individuo durante su ciclo de vida (llegada al hospital, primera consulta, consulta sucesiva, ...). Cuando se representan estas reglas en un ordenador se está creando una instanciación artificial de un componente. En una simulación basada en el proceso, esta instanciación se llama proceso.

Cada uno de estos procesos hacen uso de recursos. A menudo la simulación incluye una competición por los recursos. Los procesos tendrán que esperar en cola antes de utilizar los recursos (por ejemplo, pacientes esperando por el uso del aparato de rayos X). Los recursos no tienen que corresponder necesariamente con entidades físicas del sistema. Para el proceso de una abeja se define cómo recurso el conocimiento acerca de la localización de miel, el cual sólo puede ser obtenido cuando otra abeja hace que el recurso esté disponible (por ejemplo con su baile). La adquisición de recursos es el único método que permite que un proceso se comunique con otro.

Claramente, la modelización basada en proceso captura la esencia de la dinámica de los sistemas sociales mucho mejor que la simulación tradicional ya que en ésta se utiliza un modelo matemático donde se escogen un conjunto de parámetros que definen la dinámica del sistema, pero en estos sistemas hay un gran número de factores que afectan a su dinámica y el efecto de reducir estos factores a un número de parámetros hace que se pierda la esencia de en qué consiste el sistema. Con un modelo orientado al proceso el énfasis se centra en que los procesos intercambian o compiten por recursos. Esto da una limitación al tipo de comunicación que puede haber entre las entidades.

La simulación orientada al proceso es buena cuando se puede reducir el sistema a un conjunto de procesos que adquieren y abandonan recursos sin pensar mucho en el entorno en el cual la entidad existe. Esto es especialmente indicado en simulaciones de sistemas de redes, procesos de manufactura o colas en distintas oficinas. Sin embargo, cuando el sistema no tiene un claro intercambio de recursos entre sus procesos (ej. comunicación de abejas con otras) y existe un profundo sentido del entorno en el cual el proceso vive (ej. colmena de abejas) es difícil construir un modelo que es esencialmente un conjunto de colas. En estos casos para realizar simulaciones se debe incrementar el poder de comunicación a los procesos y definirles claramente el entorno donde viven, por lo que se hace necesario la utilización de agentes [Langton 1995].

En el caso del hospital tenemos un sistema social donde el conjunto de pacientes representan la colectividad de procesos. Estos procesos se comunican mediante la competición de recursos (materiales o humanos), lo que hace que sea un sistema de eventos discretos adecuado para ser simulado mediante una aproximación orientada al proceso.

En esta aproximación se define el comportamiento de todas las entidades del modelo por separado. La ejecución simula el comportamiento de cada entidad haciendo evolucionar el tiempo de manera que permite la concurrencia. Los lenguajes de programación que permiten este tipo de simulación deben estar preparados para ejecutar varias instancias de un mismo objeto (que puedan tener diferentes ejecuciones según el punto de simulación en el que se encuentren), así como de permitir la comunicación entre ellos.

La utilización de programación orientada a objetos [Jacobson 1992] es muy adecuada para simular estos tipos de sistemas, debido a que un objeto es una combinación de variables que definen el estado del mismo, y un conjunto de métodos que implementan su comportamiento. Las estructuras de datos fundamentales en la programación orientada a objetos son las distintas clases de objetos. Por lo tanto, para describir un sistema basado en el proceso se debe modelar cada elemento del mismo con un objeto que debe ser una instanciación de una de las clases posibles existentes en el sistema. Cada instanciación particular de un objeto tiene su propio estado, aunque la definición genérica de su comportamiento haya sido proporcionada por la clase a la que pertenece.

4.5.- Selección de la Herramienta de Programación

Una vez decidido que la dinámica del hospital se implementaría siguiendo la metodología orientada al proceso, se debe determinar con que herramienta software se desarrollará la simulación. Se analizaron varias de éstas existentes en el mercado.

Inicialmente se estudió un programa llamado MedModel diseñado para la simulación de hospitales. Este paquete de simulación se encuentra incluido dentro del grupo de programas conocidos como ProModel y que han sido desarrollados para simular distintos tipos de organizaciones.

Para el caso de la simulación de un hospital, MedModel dispone, además de las características de simulación, de una serie de librerías gráficas relacionadas con el tema. El proceso que se debe seguir para construir la simulación con esta herramienta es el siguiente:

Se definen los lugares donde se van a realizar distintas acciones. Estas acciones se describen en un lenguaje de programación propio de MedModel. Asimismo se definen los objetos que intervienen en el proceso: en nuestro caso los pacientes y los recursos. Al definir un objeto como recurso se pueden obtener estadísticas de su uso, esto es, ocupación, estados de espera de los mismos, ...

El siguiente paso es la definición del modelo que se realiza de forma gráfica utilizando librerías gráficas especiales para medicina, en este sentido la herramienta presenta una interfaz muy buena. Los informes se especifican según las necesidades de las simulaciones.

Sin embargo, para la simulación de la dinámica de un hospital general esta herramienta presenta las siguientes desventajas:

a) Sólo se pueden realizar un grupo reducido de acciones por parte de los objetos.

b) El proceso se define mediante la descripción gráfica de la ruta que sigue el elemento.

c) Obliga al modelado de detalles innecesarios, por ejemplo, se contabiliza el tiempo que tarda en llegar un paciente de la sala de espera al consultorio, cuestión que en una simulación de gran envergadura resulta poco práctico.

d) Sólo podemos obtener las estadísticas prefijadas y no se permite por ejemplo conocer el tamaño de una cola de espera.

e) No existe la posibilidad de almacenar el estado del sistema en un instante determinado, impidiendo realizar distintos experimentos a partir de un estado concreto.

f) Ninguna facilidad para comunicarse con otras aplicaciones.

En general se puede decir que la herramienta es muy autocontenida, ofreciendo al usuario poca flexibilidad para definir procesos que se salgan de los estándares.

Después del estudio de esta herramienta se analizó Modsim [CACI 1995a] que fue el paquete de software finalmente utilizado en nuestra aplicación. Modsim basado en el lenguaje modula 2, es un paquete de programación de alto nivel de propósito general, modular y que posee las estructuras básicas de la programación estructurada, además

permite la programación orientada a objetos que facilita la reutilización del código y la programación orientada al proceso.

La estructura modular de Modsim permite que el programa esté formado por un conjunto de librerías. En cualquier parte del programa se pueden importar tipos, variables, constantes y procedimientos de otra librería.

Es un lenguaje compilado, genera código del lenguaje C que es compilado para producir el fichero ejecutable que corre bajo Windows. Esto además de disminuir el tiempo de ejecución hace posible la comunicación con otras aplicaciones a través de ficheros, librerías de enlace dinámico (dll) o intercambio dinámico de datos (dde). Asimismo, permite incluir dentro del programa procedimientos escritos en el lenguaje de programación C.

Una de las características que lo hacen muy útil para la simulación es que permite concurrencia de ejecución de procesos en tiempo de simulación. Con esto, en nuestra aplicación, se puede representar varios pacientes en el mismo tiempo de simulación que están en diferentes situaciones en el sistema consumiendo recursos por separado. Para lograr esto el lenguaje de programación contiene mecanismos para la comunicación entre los objetos, esperas, reservas y adquisición de recursos, sincronismos, etc.

En una simulación de un sistema de eventos discretos se modela una secuencia de eventos ordenada en el tiempo. Un medio para permitir la correcta secuencialización en el tiempo de los distintos eventos es la creación de un reloj de simulación. Este reloj funciona como índice cronológico o planificador de instrucciones que indica cuándo ciertos eventos deben ocurrir. Modsim gestiona automáticamente el tiempo de simulación. Este es adimensional y depende de la interpretación del programador que determinará el nivel de granularidad apropiado a cada aplicación (años, meses, días, horas, minutos, segundos, ...). En cualquier instante del tiempo de simulación pueden haber múltiples actividades ejecutándose, aunque éstas actividades que ocurren en el mismo instante de simulación se ejecutan secuencialmente baja el reloj del ordenador que gestiona el tiempo de ejecución de las instrucciones.

Las librerías incorporadas permiten la utilización de objetos tipo cola (con prioridad, múltiples servidores, ...), así como objetos tipo recursos los cuales son muy utilizados en la simulación del flujo de pacientes por el hospital.

El Modsim II es una herramienta de entorno gráfico (bajo plataforma Windows) que consta de varias utilidades: debugger, programa de diseño de gráficos, editor, browser de objetos, etc. con lo que se facilita la implementación y validación del sistema [CACI 1995b].

4.6.- Implementación de un Prototipo para la Simulación del Flujo de Pacientes

El diseño e implementación de la dinámica del hospital se ha realizado en dos fases. Una primera en la que se ha construido un prototipo o aplicación "toys", que consiste en simular un único servicio del hospital Nuestra Señora de la Candelaria, concretamente el servicio de ginecología. Una vez validado el programa que ha resultado de la primera fase se generaliza el trabajo, esto es, se implementa un programa que simula todos los servicios de un hospital general cualquiera.

El objetivo de desarrollar un prototipo es determinar las estructuras de datos y control necesarias para programar la dinámica del hospital. La idea de realizar una aplicación "toys" antes de diseñar el programa general surge como consecuencia de que un hospital está formado por un conjunto de servicios. Si se simula un servicio específico entonces la implementación del programa total será el mismo proceso pero replicado tantas veces como servicios tenga el hospital. El problema se centra entonces en cómo interconectar estos servicios en el paso de generalización. Esta cuestión se resuelve observando que las interconexiones de los servicios se producen por la existencia de servicios centrales que atienden pacientes de todo tipo de patologías. En el prototipo esta interconexión es sustituida por el tiempo medio que tardan los servicios centrales en atender a un paciente, pero en el programa general se sustituye este tiempo medio por la existencia del servicio central correspondiente.

Para la construcción del prototipo nos basamos en el flujo de pacientes por el servicio de ginecología del hospital Ntra. Sra. de la Candelaria del Servicio Canario de Salud, figura 2.7.

En la dinámica del servicio de ginecología se observa que los elementos que intervienen en el modelo son los pacientes y los recursos. Los pacientes están caracterizados por tener una de las cuatro patologías que se tratan en el servicio de ginecología (funcional, orgánica, oncológica y precoz). Los recursos que pueden ser materiales o humanos tiene como atributo la jornada en la que están disponibles para atender a los pacientes. Para simular este proceso se han definido tres clases de objetos:

- Clase paciente: su recorrido por el servicio de ginecología adquiriendo y dejando recursos marca la dinámica del sistema. El conjunto de variables y métodos que constituyen esta clase de objeto se encuentra en la figura 2.8.
- Clase recurso: es un tipo de objeto definido en Modsim y que representan a los recursos necesarios para atender a los pacientes en los lugares en los que se encuentran. El conjunto de variables y métodos que definen esta clase de objeto se encuentra en la figura 2.10.
- Clase Gestor: Sólo existirá una instanciación de este tipo de objeto y será el encargado de que los recursos cumplan su horario, esto es, pone disponible los recursos al comienzo de la jornada y los deshabilita cuando éstos acaban su horario. El conjunto de variables y métodos que forman esta clase de objeto de encuentra en la figura 2.9.

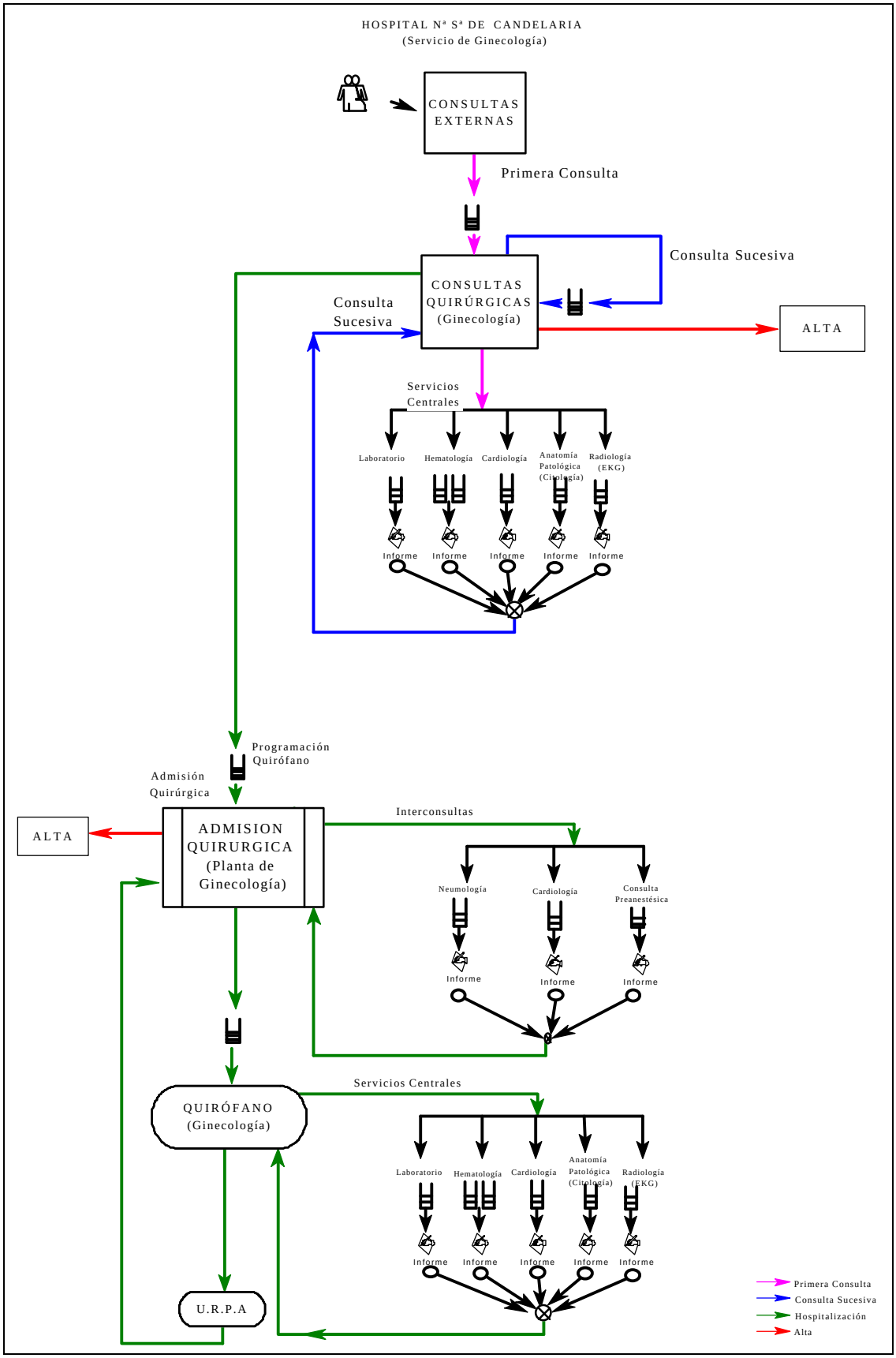


Figura 2.7.- Modelo del Servicio de Ginecología del Hospital Nª Sra. de la Candelaria.

```
PacienteObj = OBJECT              {declaración de la clase paciente}
    IdPac: INTEGER;
    PrimConsulta:INTEGER;
    Tipo: Especialidad;
    NConsSuc:INTEGER;
    Prueba:FIXED ARRAY[1..NPRU] OF INTEGER;
    DiaConsulta:Dia;
    EsperaCama: INTEGER;
    DiaEntrada: INTEGER;
    EsperaQuiro: INTEGER;
    Quiro: INTEGER;
    EntradaQuiro: INTEGER;
    PlusCama: INTEGER;
    ASK  METHOD ObjInit;
    TELL METHOD InitSystem(IN p:INTEGER; IN t:INTEGER; IN d:Dia; IN n:INTEGER; IN
                           q:INTEGER; IN eq:INTEGER; IN ec:INTEGER);
    TELL METHOD Muerte;
    TELL METHOD SolicitaNumero;
    TELL METHOD EsperaPorConsulta;
    ASK  METHOD MarcaPruebas;
    ASK  METHOD ObjTerminate;
    TELL METHOD PacienteEjecucion(IN t:Especialidad);
    TELL METHOD PacientePlanta;
    TELL METHOD Opera(IN planta:INTEGER);
    TELL METHOD PedirQuirofano;
END OBJECT;
```

Figura 2.8.- Definición de la clase de objeto paciente.

```
GestorObj = OBJECT              {objeto que monitoriza las jornadas de los recursos}
    ASK METHOD ObjInit;
    TELL METHOD MonitorizaAdministrativo;
    TELL METHOD MonitorizaConsultorios(IN tip:INTEGER);
    TELL METHOD MonitorizaEnfermeraConsultas;
    TELL METHOD MonitorizaMedicosConsultas;
    TELL METHOD MonitorizaQuirofanoGine;
    TELL METHOD EjecucionGestor;
    TELL METHOD InicializacionCama;
    ASK METHOD ObjTerminate;
END OBJECT;
```

Figura 2.9.- Definición de la clase de objeto gestor.

```
ResourceObj = PROTO
AllocationList   : AllocQueueObj
   PendingList      : PriorityList
   MaxResources     : INTEGER
   Resources        : INTEGER
   PendingResources : INTEGER
    ASK METHOD ObjInit
    ASK METHOD ObjTerminate
    ASK METHOD ReportAvailability() : INTEGER
    ASK METHOD ReportNumberPending() : INTEGER
    ASK METHOD NumberAllocatedTo(IN Object : #ANYOBJ) : INTEGER
    ASK METHOD Create(IN number : INTEGER)
    ASK METHOD IncrementResourcesBy(IN incBy : INTEGER)
    TELL METHOD DecrementResourcesBy(IN decBy : INTEGER)
    WAITFOR METHOD Give(IN Me : #ANYOBJ; IN numberDesired : INTEGER)
    WAITFOR METHOD TimedGive(IN Me           : #ANYOBJ;
    WAITFOR METHOD PriorityGive(IN Me          : #ANYOBJ;
    WAITFOR METHOD GetResource(IN Me        : #ANYOBJ;
    ASK  METHOD TakeBack(IN FromMe        : #ANYOBJ;
    ASK METHOD Transfer(IN From, To    : #ANYOBJ;
    ASK METHOD Cancel(IN Object        : #ANYOBJ;
    ASK METHOD Reset
    ASK METHOD ResetAllocationStats
    ASK METHOD SetAllocationStats(IN on : BOOLEAN)
    ASK METHOD SetAllocHistogram(IN low      : INTEGER;
    ASK METHOD AllocMaximum()     : INTEGER
    ASK METHOD AllocMinimum()     : INTEGER
    ASK METHOD AllocCount()           : INTEGER
    ASK METHOD AllocMean()            : REAL
    ASK METHOD AllocVariance()        : REAL
    ASK METHOD AllocStdDev()          : REAL
    ASK  METHOD AllocWtdMean()     : REAL
    ASK METHOD AllocWtdVariance() : REAL
    ASK METHOD AllocWtdStdDev()   : REAL
    ASK METHOD ResetPendingStats
    ASK METHOD SetPendStats(IN on : BOOLEAN)
    ASK METHOD SetPendHistogram(IN low: INTEGER;IN high: INTEGER; IN interval : INTEGER)
    ASK METHOD PendingMaximum()         : INTEGER
    ASK METHOD PendingMinimum()         : INTEGER
    ASK METHOD PendingCount() : INTEGER
    ASK METHOD PendingMean() : REAL
    ASK METHOD PendingVariance() : REAL
    ASK METHOD PendingStdDev() : REAL
    ASK METHOD PendWtdMean() : REAL
    ASK METHOD PendWtdVariance() : REAL
    ASK  METHOD PendWtdStdDev()   : REAL
      PRIVATE
        outstanding      : INTEGER; (* used in decrement *)
        ASK  METHOD Find(IN Obj : #ANYOBJ) : #ANYOBJ;
        ASK  METHOD Allocate(IN Me: #ANYOBJ; IN number: INTEGER; IN priority : REAL);
END PROTO;
```

Figura 2.10.- Definición de la clase de objeto recurso. Este tipo de objeto viene definido en la librería ResMod.

Otras características que debe tener el programa son:

1.- Facilidad para cambiar el número de recursos del servicio de manera que para un mismo estado del sistema se puedan estudiar distintas redistribuciones de recursos.

2.- Salvar y recuperar el estado del sistema de forma que se pueda volver a un estado anterior y experimentar distintas opciones. Dado que el estado de un sistema debe ser la colección mínima de información necesaria para predecir unívocamente el comportamiento futuro del mismo, el estado del hospital viene definido por el conjunto de pacientes con todas sus características (p.e. si espera para primera consulta, si está en planta, patología que tiene, etc). Asimismo, formará parte del estado el conjunto de recursos materiales y humanos existentes en el hospital.

3.- Producir como resultado el tiempo medio de todas las colas generadas en el servicio de ginecología, que son las siguientes:

- Tiempo medio en primera consulta para oncológica, funcional, orgánica y precoz.

- Tiempo medio en consulta sucesiva para oncológica, funcional, orgánica y precoz.

- Tiempo medio en quirófano para oncológica y funcional/orgánica.

Para cumplir con todas las especificaciones el programa está formado por los seis módulos siguientes:

1.- Módulo Gine2.- es el programa principal que inicializa la simulación.

2.- Módulo Gestor.- encargado de habilitar y deshabilitar los recursos en el horario correspondiente.

3.- Módulo Paciente.- donde se implementa el flujo de los pacientes por los distintos lugares del servicio de ginecología.

4.- Modulo Gine.- incluye un conjunto de procedimientos necesarios para la ejecución del programa, entre ellos se encuentra el procedimiento encargado de guardar y recuperar el estado del sistema. Este trabajo se realiza haciendo uso de un fichero de texto.

5.- Módulo Diálogo.- es la interface con el usuario (gerente) permitiéndolo redistribuir los recursos para analizar las consecuencias de las posibles acciones. La estructura del servicio, esto es, conjunto de recursos que lo forman, se guarda en un fichero de texto. En la figura 2.11 podemos observar la pantalla generada por este módulo.

6.- Módulo Gráfica.- es el encargado de representar los resultados, para ello genera una gráfica por cada una de las colas existentes en el servicio, figura 2.12.

El listado del programa se encuentra en el apéndice A.

Figura 2.11.- Menú de interface con el usuario.

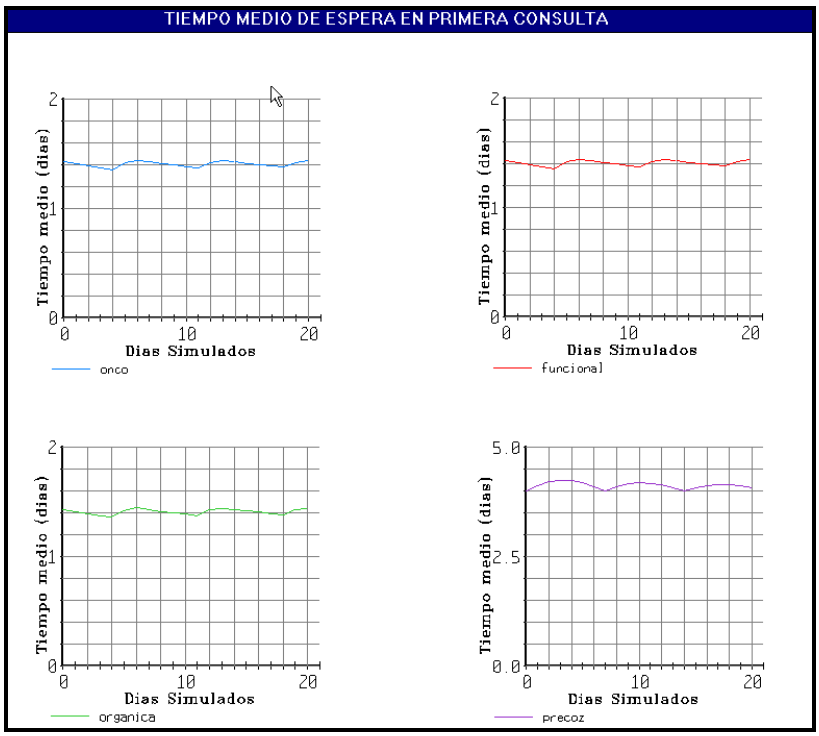


Figura 2.12.- Resultado de las colas de primera consulta del servicio de ginecología.

4.7.- Implementación de la Dinámica de un Hospital

Después de validar el programa que simula el servicio de ginecología se implementó la simulación de la dinámica de todo el hospital. Con este programa se quiere representar cualquier hospital general y por ello debe ser lo más general posible.

La pogramación tiene que ser muy estructurada de forma que se permita la reutilización. Además, esta estructuración permite que con el mismo código se puedan crear distintos hospitales ya que todos tienen los mismos tipos de objetos y los cambios vendrán dados por el número y composición de cada hospital. Por este motivo se determina que la clase de objetos a definir son los siguientes:

- Clase objeto Paciente: La implementación reflejará el flujo del paciente por los distintos servicios del hospital por lo que el objeto principal será el paciente. La característica fundamental de este objeto es el tipo de patología de la que debe ser tratado. El proceso a seguir por el objeto paciente depende de esta patología.
- Clase objeto Servicio: Hay dos tipos de objeto servicio. El objeto servicio general y el objeto servicio central. La diferencia entre los dos tipos está centrada en el hecho de que el servicio general tiene sus propios pacientes con un conjunto de patologías específicas, mientras que el servicio central no posee sus propios pacientes sino que su trabajo es realizar pruebas clínicas a los pacientes procedentes de los servicios generales. Por lo tanto la característica principal del objeto servicio general es el conjunto de patologías que atiende, mientras que la del objeto servicio central es el conjunto de pruebas clínicas que realiza.
- Clase objeto Patología: Cada patología dispone de un conjunto de recursos que el paciente puede utilizar para su tratamiento.
- Clase objeto Prueba Clínica: Idem objeto patología.
- Clase objeto Recurso: Esta estructura describe los recursos que utiliza el paciente en su paso por los distintos lugares del hospital. Los recursos que

son materiales o humanos tienen como característica el número de instancias del recurso existente para cada patología o prueba clínica, así como el horario de disponibilidad de cada instancia de cada recurso. Para que el objeto paciente puede utilizar el recurso debe adquirirlo, en caso de que el recurso no esté disponible se colocará en cola de tipo FIFO con prioridad.

La ejecución del programa comienza cargando la estructura (servicios y recursos) del hospital desde una base de datos implementada en Access. Las tablas y relaciones entre ellas se muestra en la figura 2.13.

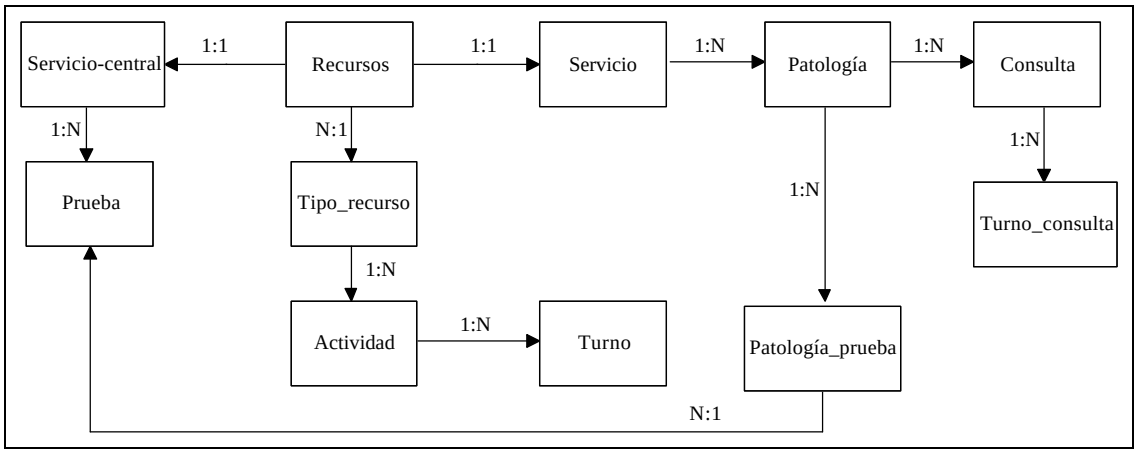


Figura 2.13.- Tablas y relaciones de la base de base de datos que guarda la estructura del hospital.

La base de datos está formada por las siguientes tablas:

- Tabla Servicio: almacena el nombre de todos los servicios no centrales del hospital, así como el número de pacientes diarios que llegan al servicio. Tiene una relación de uno a muchos con la tabla Patología ya que cada servicio atiende a varias patologías.
- Tabla Servicio_Central: guarda el nombre de los servicios centrales. Esta tabla mantiene una relación de uno a muchos con la tabla Prueba debido a que un servicio central puede realizar varios tests clínicos.
- Tabla Patología: se almacenan los datos acerca de todas las patologías que se

tratan en el hospital, esto es, nombre patología, servicio al que pertenece, número de consultas sucesivas asociado a los pacientes de esa patología, tiempo prequirúrgico, postquirúrgico, ... Presenta una relación de uno a muchos con la tabla Patología_prueba, puesto que una patología solicita un número de pruebas.

- Tabla Patología_prueba: relaciona las patologías con las pruebas que éstas solicitan, para ello mantiene una relación de muchos a uno con la tabla Prueba porque una prueba puede ser solicitada por muchas patologías.
- Tabla Prueba: indica el nombre de cada prueba, el servicio central al que pertenece y el tiempo de duración de la misma.
- Tabla Recurso: almacena el nombre de los recursos, el servicio al que pertenecen, el tipo de recursos y el número de instancias existentes. Mantiene una relación de uno a uno con las tablas Servicio y Servicio_Central porque cada recurso sólo puede pertenecer a un servicio. Asimismo tiene una relación de mucho a uno con la tabla Tipo_recurso que hace que varios recursos puedan ser del mismo tipo. Además, la relación uno a mucho que existe con la tabla Turno indica que cada recurso puede tener varios turnos.
- Tabla Tipo_recurso: indica los tipos de recursos. La relación con la tabla Actividad es de uno a muchos ya que un recurso puede realizar varias actividades, por ejemplo, el médico tiene la actividad de consulta, de atención en planta y quirófano.
- Tabla Actividad: guarda las actividades que puede realizar cada tipo de recurso. Mantiene una relación de uno a muchos con la tabla Turno puesto que cada actividad puede realizarse en varios turnos distintos.
- Tabla Turno: almacena el horario, esto es, para cada día la hora de comienzo y de finalización de la jornada de trabajo de cada recurso y para cada una de sus actividades asociadas.
- Tabla Consulta: una tabla de un tipo especial de recurso porque no pertenece a un servicio sino que está asociado a una patología concreta. La relación muchos a uno que mantiene con la tabla Patología indica que pueden existir

varios consultorios donde se atiendan una misma patología. Además guarda una relación de uno a muchos con la tabla Turno_consulta ya que cada consultorio puede tener varias jornadas de trabajo.

- Tabla Turno_consulta: almacena la jornada de trabajo de cada consulta.

La comunicación entre el programa Modsim y la base de datos se realiza mediante el intercambio dinámico de datos o DDE (Data Dynamic Exchange). Un intercambio dinámico de datos siempre tiene lugar entre una aplicación cliente y una aplicación servidora. La aplicación cliente, que en nuestro caso es el programa de simulación, inicializa el intercambio estableciendo una conversación con el servidor que es la base de datos de Acess. Después de iniciada la conversación el cliente envía transacciones al servidor, cada transición será una petición de datos, esto es, el Modsim solicita al Acces los datos sobre el hospital tales como número y nombre de los servicios, patologías que atienden, recursos que tienen, etc. La base de datos responde al cliente suministrándole los datos pedidos. El programa de simulación termina la comunicación cuando no necesita más datos del servidor. Para definir la estructura de un hospital se ejecuta un procedimiento que consiste en formularios Access que modifican la base de datos.

Después de inicializado el hospital, la ejecución la marca la llegada diaria de pacientes con distintas patologías. Dependiendo de la patología el paciente pertenecerá a un servicio u otro. El servicio tiene una serie de recursos con los que atiende a sus pacientes. Además, es el encargado de indicarle al paciente las pruebas clínicas que debe realizarse. Cada una de estas pruebas pertenece a un servicio central determinado que posee los recursos necesarios para realizar los tests clínicos. Por lo tanto, el paciente que llega al hospital sigue el proceso de la figura 2.14.

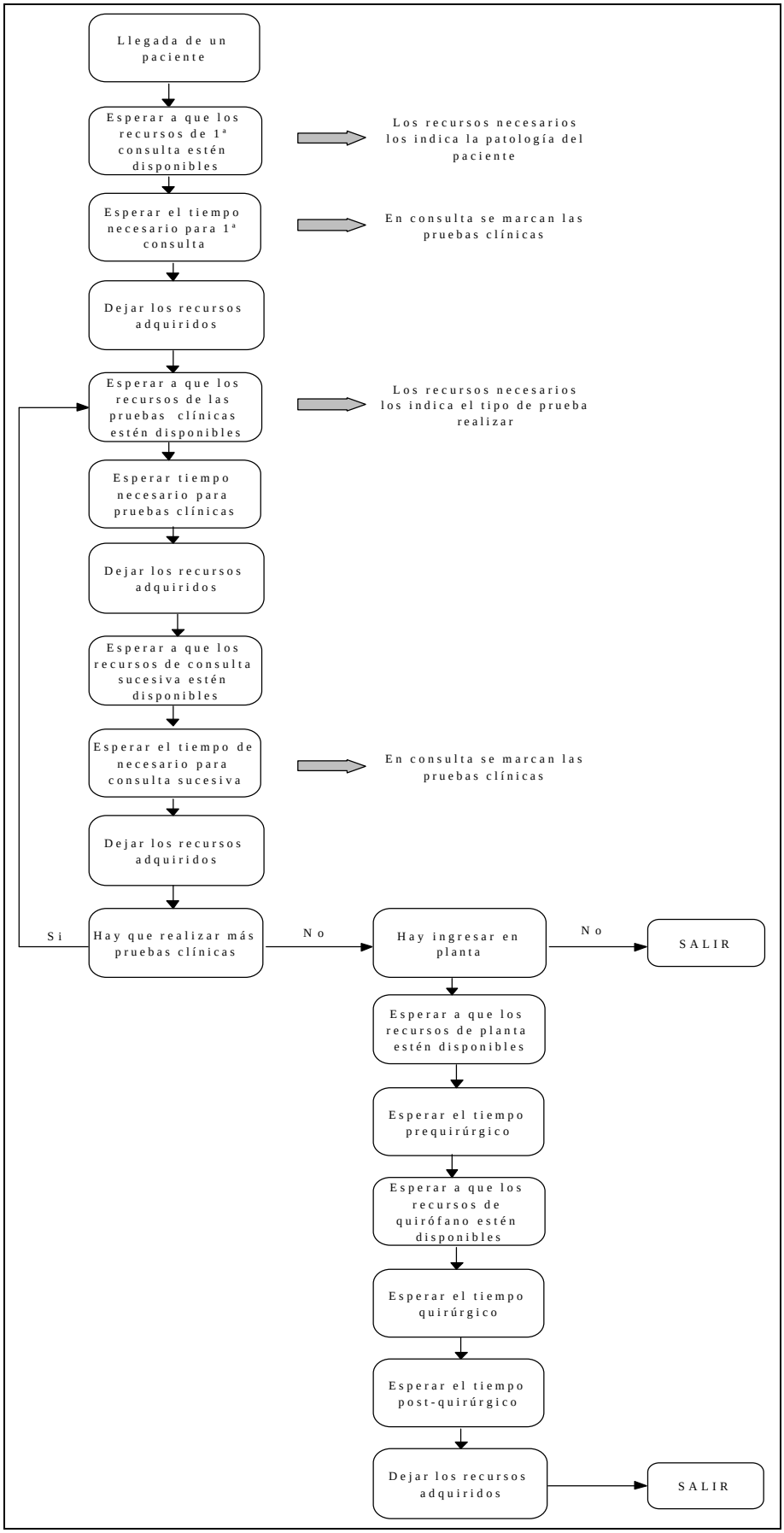


Figura 2.14.- Diagrama de flujo del proceso de cada paciente que llega al hospital.

Como se observa en el flujo del proceso de los pacientes, cuando estos necesitan realizar cualquier actividad (ej. prueba clínica) tienen que adquirir el conjunto de recursos necesarios (local, analista, ...) para ese proceso. La obtención de los recursos puede crear un problema de deadlock (un paciente tiene un recurso pero necesita del que ha adquirido otro paciente para completar el proceso, este segundo paciente tampoco puede realizarse la prueba por no tener todos los recursos). El mecanismo implementado para resolver este bloqueo consiste en hacer que todos los pacientes adquieran los recursos en el mismo orden y que las colas en las que esperan sean de tipo FIFO (First Input First Output). Aun así puede seguir existiendo este tipo de problema por lo que la decisión adoptada consiste en controlar la secuenciación ocasionada por la llegada de los pacientes que interactúan en el mismo tiempo de simulación.

Cuando un servicio central o no central no tiene el número de recursos suficientes para atender a todos los pacientes que le llegan se crean colas de esperas. El tiempo medio de estas colas será la salida del programa puesto que son los valores que le indican al gerente si el hospital funciona correctamente o no.

Uno de los requisitos exigidos al programa es que salve el estado para que se pueda ejecutar a partir de un mismo estado con acciones de control (redistribución de recursos) diferentes, y así analizar sus efectos. Debido a que se realiza una microsimulación el estado del sistema vendrá determinado por el conjunto de elementos que circulan por el mismo. Por lo tanto, salvar el estado consistirá en guardar cada uno de los pacientes con sus características (patología, pruebas clínicas a realizarse, lugar del hospital en el que se encuentra). La manipulación de tal cantidad de información obliga al uso de una base de datos que se implementa en Access. Las tablas asociadas al estados son las mostradas en la figura 2.15.

TABLA	PACIENTES	PRUEBAPENDIENTE
Atributos	identificador-paciente queconsulta patología servicio identificador-estado clave	identificador-paciente prueba identificador-estado clave

Figura 2.15.- Tablas necesarias para guardar el estado del hospital

La comunicación entre el programa de simulación y el Acces se realiza de nuevo mediante DDE, siendo el Modsim el programa cliente que establece la conexión y solicita a la aplicación servidora la ejecución de órdenes de inserción en las tablas correspondiente los datos enviados por la simulación. Para recuperar el estado al comienzo de una simulación, el Modsim vuelve a iniciar la comunicación como cliente y solicita los datos sobre los pacientes a la aplicación servidora, esto es, al Access.

4.7.1.- Simulación Microscópica

Una vez implementado el programa que simula la dinámica de un hospital general se realiza su validación. En esta etapa el problema detectado es que la simulación de periodos de tiempo muy grande necesitan un tiempo de ejecución excesivo. Por lo tanto, si con la simulación se desea estudiar las consecuencias de distintas distribuciones de recursos durante un periodo de un año, el tiempo de ejecución es grande.

En la figura 2.16 se muestran los tiempos obtenidos para distintos experimentos. Cuando se utiliza un procesador Pentium de 100 MHz y con 32 Mb de memoria los tiempos de ejecución empiezan a ser considerables a partir de que se introduzca una llegada de 32 o más pacientes al día y se realice una simulación de más de 20 semanas. Si mantenemos la memoria pero aumentamos la velocidad del procesador al doble, 200 MHz, ocurre que el tiempo de ejecución se disminuye a la mitad, pero como en el caso anterior, cuando se tiene una llegada de pacientes superior a 32 al día se produce un bloqueo del ordenador debido al excesivo intercambio que se realiza con el disco duro por la falta de memoria principal. Para comprobar que la memoria es un factor decisivo en una simulación de este tipo, se realizan los mismo experimentos con un procesador pentium de 200 MHz pero se aumenta la memoria al doble 64 Mb. Los resultados obtenidos muestran que el tiempo de ejecución no varía respecto al ejercicio anterior, las diferencias se presentan en el caso de simulaciones de un periodo muy largo de tiempo (dos años) y con una llegada de pacientes diaria también grande (más de 32 pacientes al día) que en el caso de ordenadores con menos memoria se produce una aumento exponencial en el tiempo de ejecución debido a la necesidad de hacer intercambio de datos con el disco duro, este problema se soluciona con el aumento de la memoria principal.

Ordenador Pentium 100 MHz, 32 MB			
Tiempo simulado	Pacientes diarios		
	16	32	48
3 semanas	0.33	0.55	0.83
9 semanas	0.90	1.92	3.60
13 semanas	2.76	4.1	7.13
20 semanas	3.18	9.30	14.28
40 semanas	10.11	35.38	76.48
Ordenador Pentium 200 MHz, 32 MB			
Tiempo simulado	Pacientes diarios		
	16	32	48
3 semanas	0.13	0.25	0.33
9 semanas	0.42	0.85	1.53
13 semanas	0.63	1.81	3.23
20 semanas	1.38	3.60	6.50
40 semanas	4.65	16.13	30.93
Ordenador Pentium 200 MHz, 64 MB			
Tiempo simulado	Pacientes diarios		
	16	32	48
3 semanas	0.13	0.20	0.32
9 semanas	0.35	0.85	1.45
13 semanas	0.66	1.73	3.10
20 semanas	1.16	3.50	6.35
40 semanas	4.71	14.25	30.55

Figura 2.16.- Tablas que muestran el tiempo de ejecución en minutos para distintos experimentos realizados con diferentes recursos computacionales.

El motivo que ha generado este problema es el tipo de simulación realizada. En un sistema social formado por distintos colectivos si realizamos una simulación microscópica [Harding 1990], representamos cada elemento del sistema y simulamos su proceso, resulta que el gasto de memoria es grande debido a que se tiene como elementos activos todos y cada uno de los componentes del sistema, en el caso del

hospital cada paciente y cada servicio con todos y cada unos de sus recursos. Además se tiene que simular el proceso que sigue cada uno de los elementos del sistema por lo cual se necesita mucho tiempo de ejecución. El trabajo fundamental de este tipo de simulación es la gestión de los eventos futuros, en cada proceso hay que determinar cual de ellos evolucionará en un instante determinado. Esta gestión se realiza de forma diferente si la simulación es secuencial o es paralela.

En una simulación secuencial, como la que hemos realizado, el algoritmo de gestión utiliza una estructura de datos ordenada llamada lista global de eventos, donde se almacenan todos los eventos generados y el tiempo en el que tienen que ser activados. Como muestra la figura 2.17 en la lista deben aparecer todos los objetos que intervienen en el sistema y que generan acciones, y se ordenan atendiendo al tiempo en que tienen que ser lazandas las actividad de un objeto. Así el objeto con la actividad a ser ejecutada más próximamente será el primero de la lista. Cuando la primera actividad se realiza esta estructura de datos debe ser de nuevo ordenada de forma que siempre quede en la cabeza de la lista el objeto con la actividad más próxima a ejecutar. La gestión de esta lista hace que la ejecución sea cada vez más lenta, a medida que el número de objetos aumenta así como el número de actividades asociadas a cada uno de ellos.

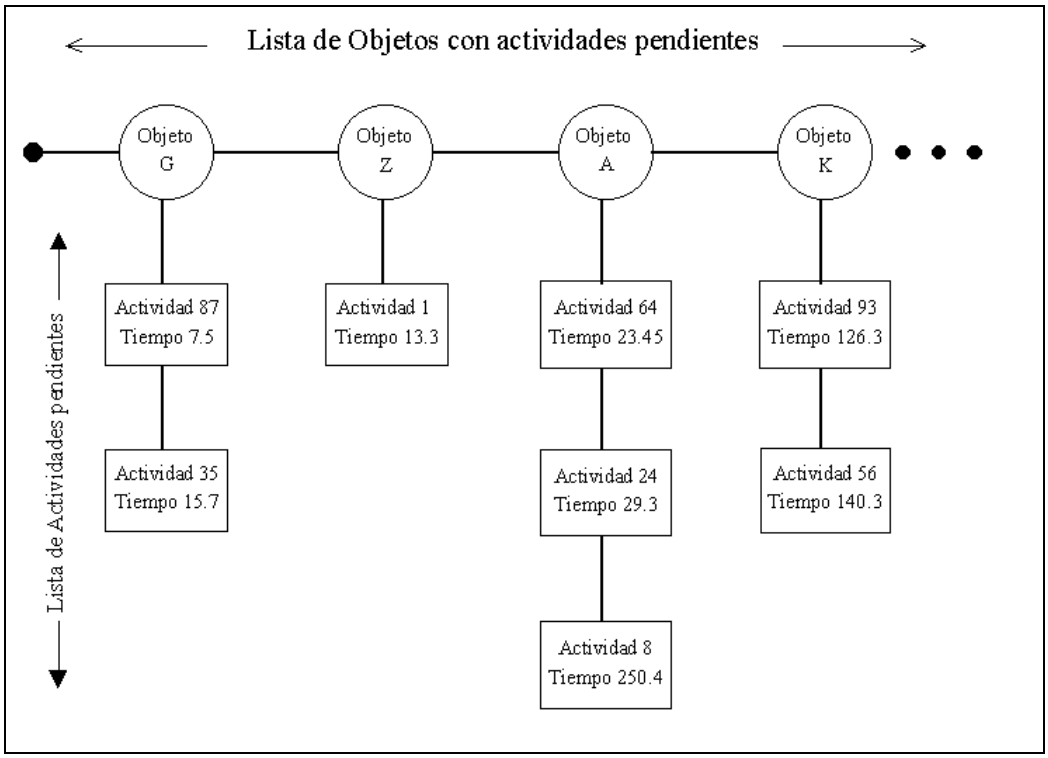


Figura 2.17.- Lista de eventos pendientes en una simulación microscópica.

Una solución al problema es realizar una simulación paralela donde cada entidad del modelo es mapeado en un procesador específico [Unger 1993]. Cada procesador tendrá su propia lista donde almacena los eventos de los objetos que se ejecutan en esa máquina. La comunicación entre procesadores se puede establecer mediante paso de mensajes o a través de una memoria compartida. El principal problema que surge en la simulación paralela es la restricción causal, esto es, conseguir que los eventos que están en cada una de las listas de eventos de cada uno de los procesadores se ejecuten en el tiempo global correspondiente. Los algoritmos de sincronización son los encargados de la resolución de este problema, existen fundamentalmente de dos tipos: los conservadores y los optimistas.

Los conservadores no permiten que se produzca un error de causalidad: cada objeto realiza una actividad solamente cuando el algoritmo puede asegurar que no le llegará otra actividad con un tiempo de simulación menor. Este mecanismo puede producir deadlock.

Los protocolos optimistas permiten procesar eventos en cualquier orden, sin embargo el algoritmo de sincronización se encarga de detectar y corregir los problemas de causalidad producidos. Uno de los mecanismos más simples para lograr esto es que cada procesador vaya guardando periódicamente su estado. Si se detecta un error al haber procesado un evento en el orden no adecuado, el procesador volverá a un estado anterior correcto y procesará a partir de ese punto los eventos en el orden apropiado.

Para la simulación paralela de la dinámica de un hospital se simula en cada máquina un servicio y habrá un host que funcione como gestionador y se encargará de la interconexión de los servicios. Esta tarea de paralelización es una línea abierta de este trabajo.

La solución implementada ha consistido en beneficiarnos de las ventajas de la micro y macro simulación [Dekkers 1995]. El problema planteado surge como consecuencia de realizar microsimulación, ya que existen muchos elementos circulando por el sistema, con un coste en memoria y en tiempo de ejecución asociado. Cuando este coste llegue a ser excesivo se realiza una focalización sobre el servicio que se quiere

analizar, esto es, se sigue realizando una microsimulación del servicio pero se considera ahora el macromodelo del resto del hospital.

El mecanismo que nos permite pasar del micro al macromodelo consiste en la utilización de distribuciones estadísticas tal como valores medios. Esto quiere decir que en el servicio focalizado se realiza un microsimulación y su relación con el resto de los servicios del hospital va a venir dada por el tiempo medio que los servicios tardan en atender a un paciente.

Al utilizar esta metodología mixta se obtienen las ventajas de la microsimulación ya que se conoce con todo detalle el estado y la dinámica del servicio en estudio, y los beneficios de los macromodelos que permiten reducir el tiempo de ejecución y el gasto de memoria a cambio de perder información detallada sobre parte del colectivo del hospital [Moreno 1998b].

Capítulo III
Sistema basado en el conocimiento para la ayuda a la toma de decisiones en la gerencia hospitalaria

CAPÍTULO III

SISTEMA BASADO EN EL CONOCIMIENTO PARA LA AYUDA A LA TOMA DE DECISIONES EN LA GERENCIA HOSPITALARIA

1.- INTRODUCCIÓN

En un principio el término producción se aplicó, únicamente, a la producción de bienes materiales, es decir, bienes físicos y tangibles, como pueden ser: los alimentos, la ropa, los libros,... Según este concepto, no realizan producción las empresas de servicios, tales como un supermercado, una universidad o un hospital.

En la actualidad se considera que toda empresa que desarrolla una actividad económico-social realiza producción, sin importar si el producto final es un bien físico o un servicio. Por lo tanto, una empresa del sector sanitario sí realiza producción y la actividad que desarrolla es un servicio.

Así lo afirma Muñoz Machado [Guadalajara 1994], cuando dice: " En economía producir es efectuar operaciones sobre un bien de modo que se le acerque más a aquella forma en la que puede satisfacer de modo directo o inmediato una necesidad. Un hospital es, en este sentido, una unidad de producción ya que restablece la salud perdida, o atenúa el dolor. Se diagnostica a cada paciente que entra en el hospital, se le prescribe un tratamiento, se desarrolla un proceso que lleva a un resultado más o menos

satisfactorio y en el que se pueden medir los consumos de factores. Estos consumos compondrán el coste y, en un hospital habrá tantos *productos sanitarios*, así medidos, como pacientes distintos reciban atención en el período a considerar."

Por tanto, la gestión hospitalaria como la de cualquier otra empresa de servicios, se debe orientar a conseguir la mejor utilización posibles de los recursos, compatible con una calidad adecuada y la consiguiente satisfacción de los clientes [Pineault 1989].

Para conseguir este fin, el gerente de un hospital se enfrenta diariamente a la toma de múltiples decisiones, tanto en el campo administrativo como en el sanitario, que afectan a la buena marcha del hospital. Las decisiones se deben tomar a partir de un gran número de variables (tiempos de esperas en todas las colas generadas debido al uso de cualquier servicio del hospital) que se actualizan semanalmente y que tienen un alto grado de interrelación, por lo que a la rapidez con que hay que tomarlas hay que añadir la limitada capacidad humana para analizarlas conjuntamente. Se hace por lo tanto patente la necesidad de elaborar una herramienta de ayuda a la toma de decisiones en la gerencia hospitalaria.

El objetivo del presente capítulo es presentar las herramientas metodológicas combinadas basadas en simulación e inteligencia artificial que hemos desarrollado para ayudar al gerente en la toma de esas decisiones. Mediante estas herramientas se ha desarrollado un Sistema Basado en Conocimiento (SBC) que incorporará el conocimiento de varios expertos en la toma de decisiones hospitalaria de forma que ayude al gerente en su trabajo diario [Boy 1991].

La construcción de un SBC se justifica por el hecho de que los gerentes realizan su trabajo con conocimiento heurístico, esto es, para la toma de decisiones se necesita un conocimiento cuyo acceso tiene asociado un alto coste debido a que se adquire con la experiencia y, en el caso de los hospitales, cualquier decision incorrecta genera un gran gasto al sistema sanitario. Además la existencia de un SBC de ayuda permite que se realice un uso más efectivo y eficiente del experto gerente ya que se le eleminan los trabajos tediosos de analizar una gran cantidad de información. Otro motivo para la construcción del SBC es la posibilidad de combinar el conocimiento experto de múltiples expertos en una base de conocimiento compartida que pueda ser estudiada en cuanto a su consistencia y la fiabilidad de sus recomendaciones. Desde el punto de vista

científico, la razón más importante es la de la formalización y la clarificación del conocimiento que se extrae al hacer explícito el experto su razonamiento.

A continuación se describe la metodolía propuesta para la toma de decisiones para posteriormente pasar a describir con detalle el SBC.

2.-TOMA DE DECISIONES EN LA GERENCIA HOSPITALARIA

El gerente de un hospital debe tener un completo conocimiento del mismo para poder realizar una correcta toma de decisiones. Sin embargo, debido a que el hospital es un sistema complejo formado por un gran número de partes que se interrelacionan, su estudio resulta una tarea difícil. En este trabajo, como hemos dicho, se propone una metodología que facilite la toma de decisiones consistente en las dos fases siguientes:

- análisis del hospital

- tratamiento automático de la información

La primera fase implica la adquisición del conocimiento sobre el hospital, esto es, un análisis exhaustivo del mismo para identificar las características importantes del sistema, así como para determinar los conceptos claves y la relaciones que caracterizan el mismo. Para realizar este primer trabajo se plantea la utilización de distintos tipos de recursos tales como libros y entrevistas con diferentes expertos. Debido a la complejidad del hospital se construye un modelo que refleja la dinámica del mismo (capítulo II), de manera que abstrayendo la complejidad de la realidad nos permita el completo conocimiento de las partes del sistema y sus interconexiones. Se realizan distintos experimentos con el programa de simulación de manera que sea una herramienta más en la adquisición del conocimiento.

Esta primera fase sirve para tener un completo conocimiento del sistema, debido a que la información obtenida es abundante y precisa le permite al gerente tener un mayor conocimiento del hospital y por lo tanto realizar una mejor toma de decisiones.

Una vez adquirido el conocimiento acerca de cómo es el hospital, y debido a que la cantidad de información producida es grande, la siguiente fase consiste en el diseño de un sistema que automatice su tratamiento. Esto implica la propuesta de soluciones y la posibilidad de testear las mismas. La simulación se presenta como el elemento

imprescindible en la predicción de las consecuencias de las distintas soluciones permitiendo elegir aquella que lleve al hospital al estado más adecuado. La importancia de conocer de antemano las consecuencias de una acción de control es debida a que en las organizaciones médicas la mayoría de los proyectos no se realizan por el riesgo de alterar negativamente el proceso de atención a los pacientes.

En esta segunda fase se necesita construir un SBC, donde se abstraen los métodos de razonamiento de los gerentes y se describe sistemáticamente cómo, cuándo y donde aplicarlos, que ayude a la toma de decisiones en la gerencia hospitalaria proponiendo soluciones al gerente que cumplan con los requisitos impuestos por el entorno socio-político.

La construcción de un SBC es una tarea complicada en la que hay que extraer, estructurar, organizar y codificar el conocimiento experto. Por ello se utiliza una metodología de desarrollo de SBC que permita el diseño estructurado [Musen 1994] y comprensible del mismo, concretamente la metodología utilizada ha sido el KADS [Schreiber 1993].

La necesidad de una metodología de este tipo es debido a que el SBC que se diseña debe ser útil para la toma de decisiones en el Hospital, considerado como prototipo, Nra. Sra. de La Candelaria, sino que además debe servir para cualquier otro hospital e incluso en diferentes empresas de características similares. Esto es, se quiere diseñar un sistema que además de organizar el conocimiento de una forma comprensiva para facilitar su depuración y justificación, permita reutilizar el conocimiento adquirido. Para tal fin Newell [Newell 1982] propone la hipótesis de los niveles de conocimientos, que establece que la descripción del conocimiento se debe realizar independientemente de su implementación concreta en reglas, frames, etc.

KADS proporciona una metodología que permite cumplir las restricciones anteriormente mencionadas. Su primer principio de "múltiples modelos" [Schreiber 1994] facilita la construcción comprensiva del SBC porque el ingeniero de conocimiento descompone su trabajo en distintas tareas, reduciendo por lo tanto la complejidad del proceso. Para permitir la reusabilidad KADS propone el segundo principio "capas de conocimiento" que dice que en el modelo experto, donde se describe el conocimiento del SBC, el conocimiento debe ser definido independientemente de la implementación

concreta (reglas, frames) centrándose en el comportamiento que el sistema debe tener y en el tipo de conocimiento necesario para tal comportamiento, abstrayéndose de los detalles de como el razonamiento es implementado [Wielinga 1994]. Los tipos de conocimientos están organizados en capas:

- conocimiento de dominio: conocimiento estático que describe la teoría de dominio de la aplicación.

- conocimiento de control: conocimiento acerca de cómo se realiza el proceso de razonamiento. Este se divide a su vez en conocimiento de tareas que define las tareas elementales a realizar, y conocimiento de inferencia que ofrece las operaciones básicas a realizar con el dominio concreto.

En este capítulo, siguiendo la metodología KADS, vamos a diseñar un SBC para la ayuda a la gerencia hospitalaria [Moreno 1998]. Se describirán los modelos considerados importantes para el desarrollo del mismo [Hoog 1994], comenzando con el modelo organizativo que define la empresa en la que trabajará el SBC. A continuación se propone el modelo de tareas donde se determinarán las tareas que debe realizar. El SBC no es un elemento que pueda trabajar independientemente en la organización, por ello se define el modelo de agentes que presenta las relaciones del SBC con otros elementos de la empresa. Finalmente se desarrollará el modelo experto que constituye el núcleo fundamental del SBC.

Para diseñar el modelo experto se sigue el principio de niveles de conocimiento. Por ello el primer trabajo consistirá en la definición del conocimiento de dominio, es decir, todos los conceptos y relaciones que se presentan en la toma de decisiones de la gerencia hospitalaria. Para cumplir con el requisito de la reutilización, se describe la teoría de la gestión hospitalaria en el lenguaje Ontolingua [Gruber 1992], que permite realizar las definiciones a partir de un lenguaje de intercambio de conocimiento (KIF) [Genesereth 1992], así como reutilizar conceptos ya definidos en otras teorías.

Una vez definido el conocimiento de dominio se describe el conocimiento de control. Este tipo de conocimiento tiene la característica de reusabilidad debido a que todos los expertos realizan el mismo conjunto de tareas para realizar los mismos trabajos en los distintos campos del saber. La toma de decisiones es una tarea que se

realiza en dominios muy diferentes: la gestión hospitalaria, el control de tráfico, el control de una planta química, etc., por ello la construcción de un SBC para realizar cualquier de estos trabajos incluirá el mismo conocimiento de control. Para describir este de forma estructurada y comprensiva se debe descomponer el proceso de razonamiento en tareas o pasos a realizar. Toda toma de decisiones consiste de las siguientes tareas [Molina 1996]:

- monitorización: determina el valor de las variables en estudio.

- diagnóstico: indica la existencia o no de problemas en el sistema.

- predicción: la resolución de problemas en el hospital se aborda inicialmente de forma local, esto es, en esta tarea se proponen soluciones locales a las distintas partes del hospital (servicios) sin tener en consideración las interrelaciones existentes. Para ello se sigue una estrategia que en la teoría de control se conoce como PID (acción proporcional- integral -derivativa) [Flanklin 1990].

- diseño: esta tarea debe plantear la solución global al sistema. Se considera un problema de diseño [Chandrasekaran 1990] donde las primitivas son las soluciones aportadas por la tarea de predicción y la configuración de la solución global debe cumplir unas restricciones de costo determinadas. La resolución de este problema se plantea como una búsqueda en un árbol (búsqueda en un espacio de soluciones) siguiendo una estrategia inicial de primero-el-mejor, para seguir con una búsqueda en profundidad intentando corregir la solución propuesta [Rich 1991] [Russell 1995].

Finalmente se describe la relación de estas tareas con el conocimiento de dominio por medio de las inferencias [Aben 1993]. La reutilización del conocimiento de control permite que el conjunto de tareas a realizar en una toma de decisiones estén perfectamente descritos, pero estos pasos deben ser adaptados al dominio concreto con el que se trabaja, para ello se definen las inferencias. Estas son los pasos elementales que hay que realizar en el proceso de razonamiento y que indican concretamente que conocimiento de dominio se necesita de entrada y que conocimiento se proporciona como salida. Todo el trabajo realizado se muestra en el esquema de la figura 3.1.

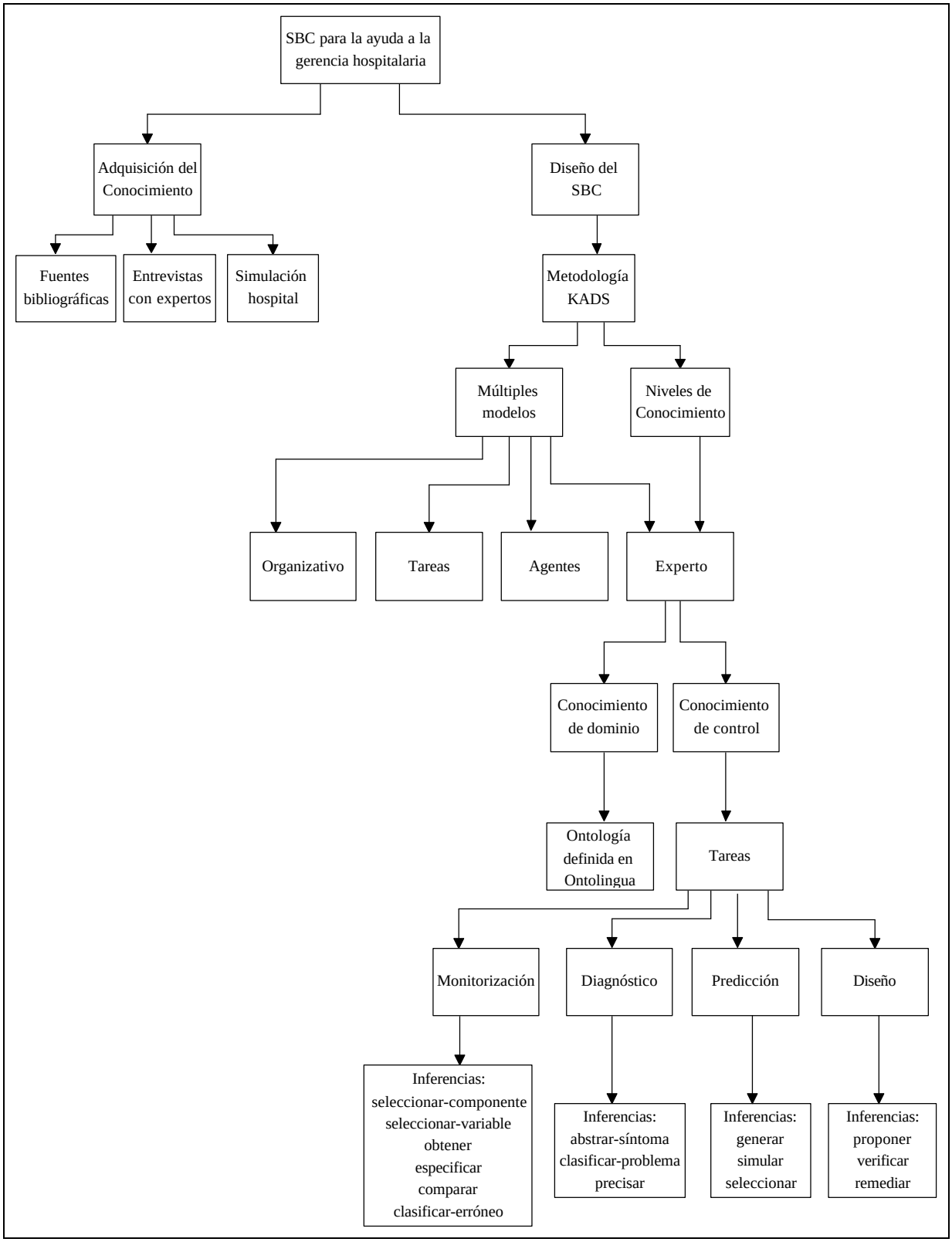


Figura 3.1.- Esquema del desarrollo de un SBC para la ayuda en la toma de decisiones de la gerencia hospitalaria.

En este apartado hemos pretendido hacer una síntesis de la metodología propuesta para el estudio de los hospitales generales. A continuación pasaremos a describir las características del SBC propuesto.

3.- MODELO ORGANIZATIVO

Un hospital es un establecimiento dedicado al tratamiento de enfermos. El hospital general es aquel que se organiza como suma de especialidades médicas y quirúrgicas, divididas en distintas unidades clínicas denominadas "servicios". Los enfermos se distribuyen entre las unidades clínicas según la patología que padezcan [Post 1997].

Existen departamentos para las especialidades médicas (cardiología, gastroenterología, reumatología y pediatría especializada), las especialidades quirúrgicas (neurocirugía, cirugía pulmonar, oncología, oftalmología, urología, otorrinolaringología, traumatología y ginecología), y las técnicas auxiliares (radiología, radioterapia, analítica, medicina nuclear, ecografía, tomografía axial computerizada, etc.), a éstas últimas se les denominan "servicios centrales".

En los hospitales existen unidades especializadas (UCI, URPA) donde se ingresa a los enfermos graves y postoperatorios que requieren una vigilancia constante o cuidados especiales.

Otro servicio es el de urgencias que atiende las necesidades de asistencia urgente, tanto del propio hospital como de pacientes externos.

Se dispone también de consultas externas para los servicios de asistencia ambulatoria, y a la que acuden los pacientes que después de ser dados de alta, necesitan un tratamiento posterior, y aquellos otros que no necesiten ser internados. Estos servicios están formados por consultorios.

Desde el punto de vista de gestión hospitalaria, el hospital está formado por tres elementos fundamentales:

1.- El primer elemento es el flujo de pacientes por los distintos servicios del hospital para ser atendidos. Como consecuencia de la no disponibilidad de los recursos necesarios, los pacientes generan colas de espera. Estas son las variables que la gerencia

del hospital debe mantener dentro de los márgenes permitidos.

2.- Cómo segundo elemento tenemos los recursos que debe gestionar el gerente para el correcto funcionamiento del hospital. Los recursos permiten que el paciente reciba el tratamiento adecuado en cada uno de los servicios del hospital en el que se encuentra. Entre los recursos podemos distinguir: los materiales (consultas, camas, quirófanos, camillas, medicinas, ...); y los humanos (médicos, ATS, cocineros, celadores, administrativos, lavanderos, ...).

3.- El último elemento a considerar es la gerencia hospitalaria donde se trabaja para que el hospital funcione de acuerdo a la planificación establecida. La tarea esencial del planificador consiste en establecer planes, es decir, prever situaciones futuras a partir de la situación actual. Partiendo del estado de salud de una población, el objetivo del planificador es asegurar una adecuación entre las necesidades de salud de una población y los recursos puestos a su disposición.

Por lo tanto, el programa a seguir por el gerente estará formado por los siguientes tres elementos [Pineault 1989]:

1) un objetivo de salud, expresado en términos de estado de salud

2) actividades o servicios, y

3) recursos: físicos y financieros.

4.- MODELO DE TAREAS

Una vez conocida la función del hospital, vamos a estudiar cuáles son las distintas tareas que se deben desarrollar para conseguir una buena planificación y por consiguiente, la satisfacción de los pacientes.

El objetivo que se persigue es realizar una buena planificación, por ello, se debe tener bien claro qué es planificación y cómo y en qué medida realizarla para llevar al hospital al estado deseado [Pineault 1989].

Es difícil proponer una definición de planificación que sea completamente satisfactoria. Cada autor tiene la suya propia. Esta diversidad de definiciones es interesante en la medida que permite identificar ciertos elementos que caractericen el

proceso de planificación.

1.- Para empezar, la planificación concierne al futuro. Esto es, planificar es aplicar un proceso que conduce a decidir qué hacer, cómo hacerlo y cómo evaluar lo que se hará antes de la acción.

2.- Una segunda característica de la planificación implica una relación de causalidad entre la acción tomada y los resultados alcanzados. Es el principio del determinismo. Este principio recurre a los elementos técnicos, es decir, a la relación plausible o demostrada entre las acciones propuestas y los éxitos que se persiguen.

3.- Esto nos conduce a una tercera característica de la planificación, la acción. El plan que resulta del proceso de planificación no puede ser sólo un documento bien escrito, lógico y bien presentado. El objeto último de la planificación es la acción, el cambio.

4.- La planificación además es un proceso continuo y dinámico. El proceso debe adaptarse continuamente a las situaciones particulares presentadas.

5.- Otra característica del proceso de planificación es su naturaleza multidisciplinaria. Este último elemento añade dificultad a la planificación ya que las diferentes disciplinas no utilizan necesariamente los mismos conceptos cuando se refieren a la planificación. El economista, por ejemplo, se interesa por la importancia del rendimiento óptimo de los recursos. Para el administrador, la planificación puede tener como objetivo aumentar la eficacia de la empresa. El político, por su parte, se interesa sobre todo del proceso de toma de decisiones. A pesar de las dificultades que entraña la multidisciplinariedad, está claro que tener en cuenta estas diferentes perspectivas de planificación puede conducir a un análisis más rico de situaciones y decisiones.

6.- El último aspecto de la planificación es que está íntimamente ligada al contexto sociopolítico en el que se aplica.

La planificación se puede definir por la extensión de su proceso. Esto es, la planificación la podemos clasificar teniendo en cuenta qué nivel de objetivos pretende alcanzar, figura 3.2.

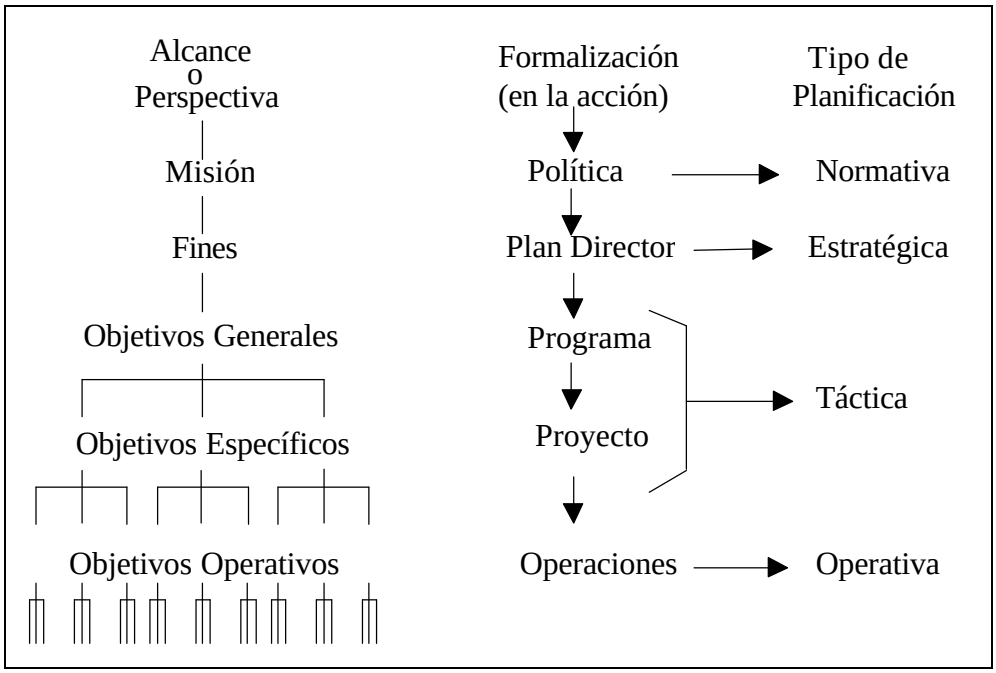


Figura 3.2.- Jerarquización de los objetivos según el nivel de planificación.

Llamamos planificación normativa o política cuando se indica la orientación general que se tiene que adoptar, planteando la misión de un organismo. Es la planificación a más alto nivel y suele ser realizada por los gobiernos o sus diferentes ministerios.

En el segundo nivel se encuentran los fines, que representan las decisiones importantes que tendrán consecuencias a largo plazo. Se trata de la planificación estratégica.

En el siguiente escalón se encuentra la planificación táctica o estructural que se encarga de los objetivos generales y específicos a seguir. La gerencia del hospital será la encargada de realizar tanto la planificación estratégica como la estructural. Como resultado se obtiene un programa o proyecto.

Y por último, hay que fijar objetivos operacionales sobre el plan concreto de operatividad y puesta en marcha del programa. Éste es el objeto de la planificación operativa, es decir, las decisiones a más corto plazo que conciernen al desarrollo de las

actividades, el calendario de ejecución, la gestión de recursos. Es aquí donde trabajará el SBC.

La figura 3.3 resume el objeto de la planificación según estos cuatro niveles.

Nivel de Planificación		Objeto
Normativo		ideales
Estratégico	Fines	metas
Táctico		objetivos
Operativo	Medios	
	Recursos	
	Organización	

Figura 3.3.- Objeto de la planificación según los diferentes niveles.

El SBC a construir debe realizar la planificación operativa del hospital, esto es la gestión de la operaciones para conseguir la utilización correcta de los recursos. Como muestra la figura 3.4.

La correcta planificación de los recursos asegura que los mismos sean suficientes y adecuados para el desarrollo de las actividades y los servicios que permitan alcanzar los objetivos del programa. Se trata de determinar el tipo y cantidad de recursos requeridos y de responder a las siguientes preguntas:

- ¿En qué medida una nueva actividad afecta al número de recursos?
- ¿Hay posibilidad de sustituir tareas o personal con una mayor productividad?
- ¿Cómo mantener la calidad de cuidados y servicios, limitándose a los recursos de la organización o a los que están disponibles en el mercado de trabajo?
- ¿Es posible con los recursos actuales producir los servicios necesarios y satisfacer las necesidades de salud?

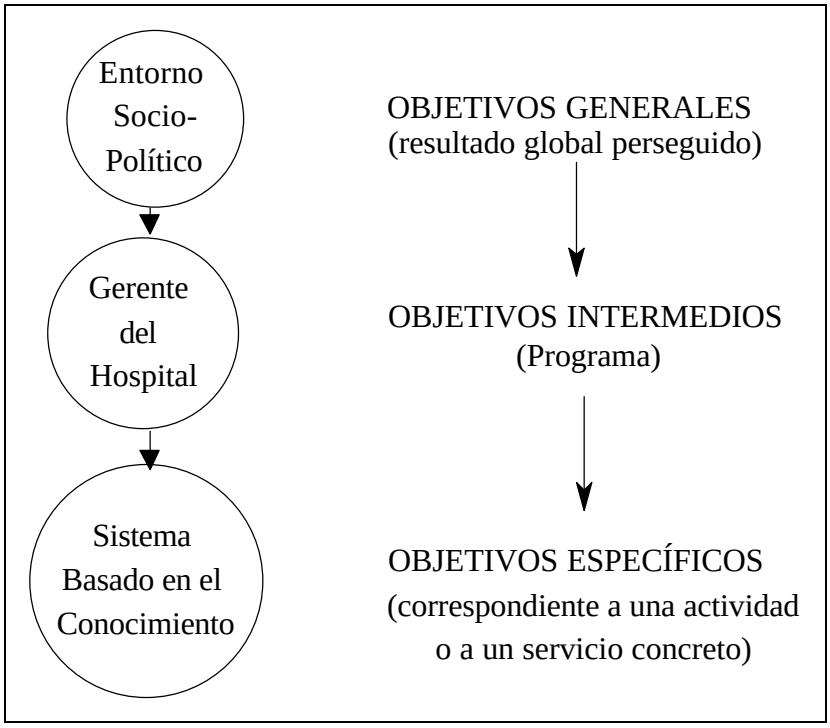


Figura 3.4.- Jerarquización de los objetivos. La superficie de los círculos está en relación directa con la cantidad de objetivos: cuanto más específicos son, más numerosos serán.

5.- MODELO DE AGENTES

En la figura 3.5 observamos los agentes que intervienen en nuestro sistema.

El "Agente Gerencia" es de tipo humano, y es el usuario del SBC. Como hemos visto anteriormente, según el objetivo deseado la planificación debe realizar una serie de tareas. En el caso del hospital, la planificación estratégica, esto es la determinación de las orientaciones y prioridades del sistema serán realizadas por el personal de gerencia que sigue las directrices gubernativas. La planificación táctica será realizada, asimismo, por el gerente para conseguir que se cumpla el plan propuesto. Como resultado de esta planificación se tendrá el programa. Esto es un documento en el que se fijaran los objetivos específicos, correspondientes a una actividad o a un servicio concreto.

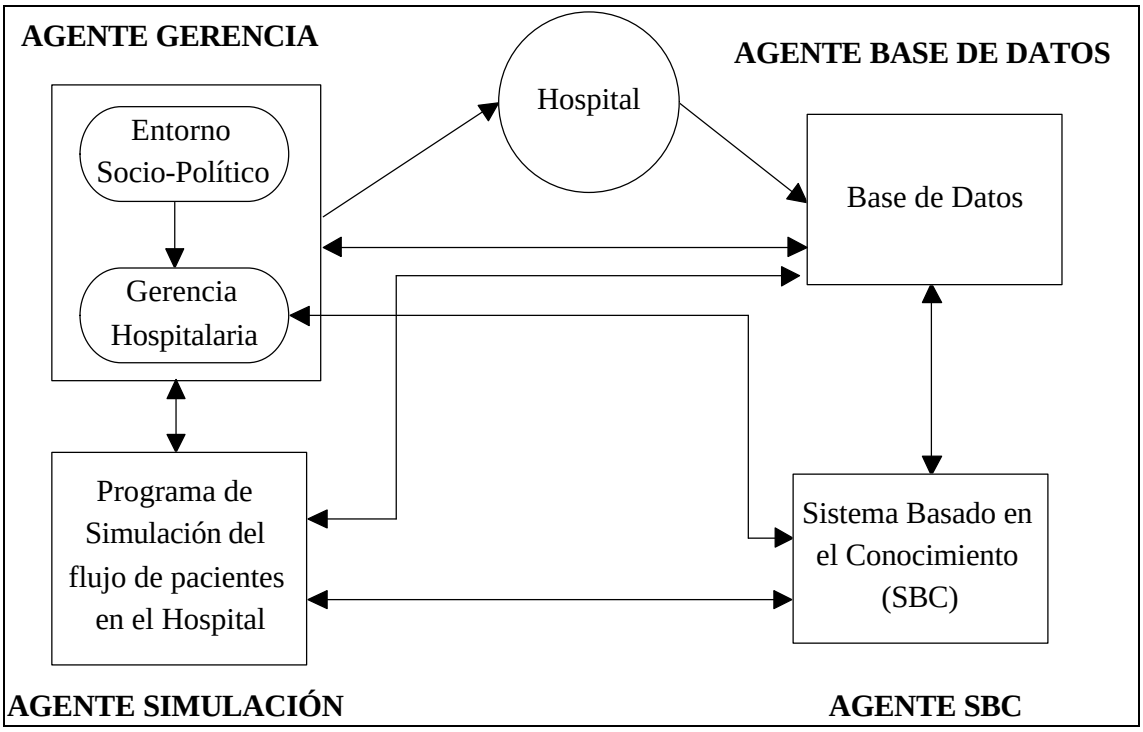


Figura 3.5.- Agentes que intervienen en el SBC para la ayuda a la gerencia hospitalaria.

El SBC será el agente (de tipo software) encargado de hacer cumplir el programa, realizando la planificación operativa. Para ello, el gerente del hospital debe comunicar al SBC el programa y éste tras un estudio del estado del hospital llegará a la solución más adecuada. En conclusión, el SBC debe realizar un proceso continuo de previsión de recursos y de servicios necesarios para conseguir los objetivos determinados según un orden de prioridad establecido, permitiendo elegir la o las soluciones óptimas o suboptimas entre muchas alternativas. Esta elección tomará en consideración el contexto de dificultades, internas y externas, conocidas actualmente o previsibles en el futuro.

Tanto el gerente como el SBC necesitan conocer y almacenar datos sobre el estado actual o deseado del hospital, por tanto surge la necesidad del "Agente Base de Datos" que guarda y permite acceder a la información del hospital.

Por último, el modelo se completa con el "Agente Simulación" el cual es necesario debido a que el SBC requiere hacer simulaciones para predecir el comportamiento del hospital y llegar a la solución más adecuada [Simon 1995]. Estas simulaciones se realizan sobre el Agente Simulación que es un programa que simula el flujo de pacientes a través de los distintos servicios del hospital y cuya realización fue

explicada en el capítulo anterior.

6.- MODELO EXPERTO

El modelo experto es el que diferencia a un SBC de cualquier otro sistema de información [Wielinga 1994]. Es el modelo en el que se describe el conocimiento que debe tener el SBC para realizar su tarea. El conocimiento lo podemos clasificar en conocimiento de dominio y conocimiento de control. El conocimiento de dominio describe el contexto de trabajo del SBC y el conocimiento de control los métodos de resolución de problemas.

El conocimiento de control se divide a su vez en el conocimiento de tareas, que define los trabajos a ejecutar en la resolución del problema, y el conocimiento de inferencia que indica los pasos elementales o inferencias a realizar con el conocimiento de dominio para llevar a cabo todas las tareas.

6.1.- Conocimiento de Dominio

El conocimiento del dominio incluye la conceptualización de un dominio para una aplicación particular en forma de teoría, esto es, un cuerpo de conocimiento representado formalmente se basa en una conceptualización: objetos, conceptos y otras entidades que puedan existir en algún área de interés, y las relaciones que hay entre ellas. Una conceptualización es una abstracción con una visión simplificada del mundo que deseamos representar.

Debido al gran resultado que ha dado definir el conocimiento de control mediante métodos generales e independientes del dominio, se ha intentado aplicar la misma metodología para facilitar la adquisición y modelado del conocimiento del dominio. Así se han definido diferentes ontologías[1] que buscan la reutilización del

[1] Una ontología es una especificación explícita de una conceptualización. El término es adoptado de la filosofía donde ontología es el estudio (la metafísica) de la existencia. Para los sistemas de la IA, lo que existe es aquello que se puede representar. Cuando el conocimiento de un dominio es representado con un formalismo declarativo, el conjunto de objetos que se representan se llama universo del discurso. Este conjunto de

conocimiento del dominio [Gruber 1993a]. Formalmente una ontología es la declaración de una teoría lógica [Gruber 1993b]. Ejemplos de estas son:

CyC: cuyo objetivo es la construcción de una gran base de conocimiento de sentido común. Este tipo de ontología se podría utilizar para que el nuevo SBC tenga un conocimiento general sobre parte del mundo que no conoce, pero que por otro lado no necesita conocer para resolver el problema para el cual fue creado.

UMLS : su objetivo es el desarrollo de un sistema inteligente que facilite y mejore el acceso a conocimiento médico.

Ontolingua: la principal idea de Ontolingua [Gruber 1992] es compartir el conocimiento. Traduce ontologías escritas en el lenguaje estándar KIF (Knowledge Interchange Format) a un sistema de implementación.

El objetivo general de cualquier ontología es compartir y reutilizar el conocimiento. Esto está motivado por la disminución de los costes de construcción, testeo y mantenimiento de los SBC.

Para ser reutilizable, una ontología debería ser una descripción genérica del dominio, esto es, una descripción independiente del problema a resolver. Así, el conocimiento descrito en una ontología reutilizable no debería ser adecuado para resolver un problema concreto, con un método determinado, sino que debería describir únicamente el universo del discurso.

Pero, ¿puede el conocimiento de dominio de una ontología reutilizable ser usada en un SBC para un método de resolución específico? ¿Puede este conocimiento ser parte de un SBC, sin ninguna modificación, mientras la descripción del dominio es dada de forma independiente al problema a resolver?

---

objetos y relaciones entre ellos se describe como un conjunto de definiciones en un vocabulario formal. Estas definiciones asocian los nombres de las entidades en el universo del discurso (por ejemplo, clases, relaciones, funciones, etc) con texto entendible por un humano, de forma que describen lo que los nombres significan, y los axiomas que restringen la interpretación y el buen uso de estos términos.

El conocimiento del dominio de un SBC corresponde a todo el conocimiento sobre el dominio que un experto necesita conocer para resolver un problema. Este conocimiento tiene las siguientes tres características:

a) Conocimiento limitado.

Un experto no necesita manipular todos los conceptos que existen de un dominio, únicamente aquellos que son necesarios en un razonamiento. Esto cubre una parte del dominio, que está limitado por su uso.

El área de conocimiento que cubre el modelo del dominio está limitada y definida por la habilidad del SBC en resolver un problema. El nivel del dominio está compuesto por conceptos y relaciones necesarias para resolver un problema.

b) Conocimiento personal.

El conocimiento que se incluye en un SBC se adquiere de expertos que son especialistas en resolver problemas específicos de un dominio. Estos expertos, con la experiencia adquieren mucho conocimiento heurístico. El SBC trabaja con este conocimiento heurístico, que está íntimamente relacionado con el experto.

Algunas veces la experiencia es adquirida de varios expertos. En este caso el conocimiento adquirido puede ser más comprensivo, más preciso y menos distorsionado. El problema está en las dificultades para que los expertos lleguen a un consenso.

c) Conocimiento relativo.

El conocimiento del dominio es relativo al problema que tiene que ser resuelto. De una forma más precisa, esta clase de conocimiento es:

* relativo al contexto de un problema particular,

* relativo al objetivo a resolver.

Por esto, el conocimiento del dominio en un SBC es dependiente del problema a resolver.

En conclusión, el conocimiento del dominio no está formado por piezas que sean reutilizables de una forma inmediata. Así, las ontologías de conocimiento reutilizables y

compartibles no se pueden utilizar de manera directa al ser un modelo de conocimiento del dominio con un nivel de abstracción superior al utilizado en la construcción de los SBC. Una buena idea es utilizar estas ontologías de conocimiento como guías para la construcción de nuevos SBC.

6.1.1.- Ontología para la Gestión Hospitalaria

El conocimiento de dominio del SBC para la gerencia hospitalaria descrito a través una red semántica se muestra en la figura 3.6. Los conceptos que con los que trabaja el SBC son los siguientes:

- Hospital: Sistema a planificar.
- Servicios: Las distintas unidades en las que se divide el hospital.
- Servicios Centrales: Los servicios del hospital encargados de realizar las pruebas clínicas.
- Servicios No Centrales: Los servicios del hospital que atienden a pacientes que sufren determinadas patologías.
- Pruebas: Examen médico realizado a un paciente para determinar la naturaleza de su enfermedad.
- Especialidades: Las partes en las que se divide un servicio y que se encargan de atender a pacientes que presentan un conjunto de patologías específicas.
- Actividad: Los diferentes trabajos que se realizan en cada servicio.
- Pacientes: La persona que recibe tratamiento médico en el hospital.
- Patologías: Las distintas enfermedades que sufren los pacientes.
- Recursos: Las distintas posesiones del hospital necesarias para que puedan realizar su función.
- Recursos Humanos: El personal que trabaja en el hospital.
- Recursos Materiales: Los dispositivos que son necesarios en el hospital para su funcionamiento.
- Cola: Un conjunto de pacientes esperando a ser atendidos.
- Tiempo medio: Tiempo medio de espera en cola.

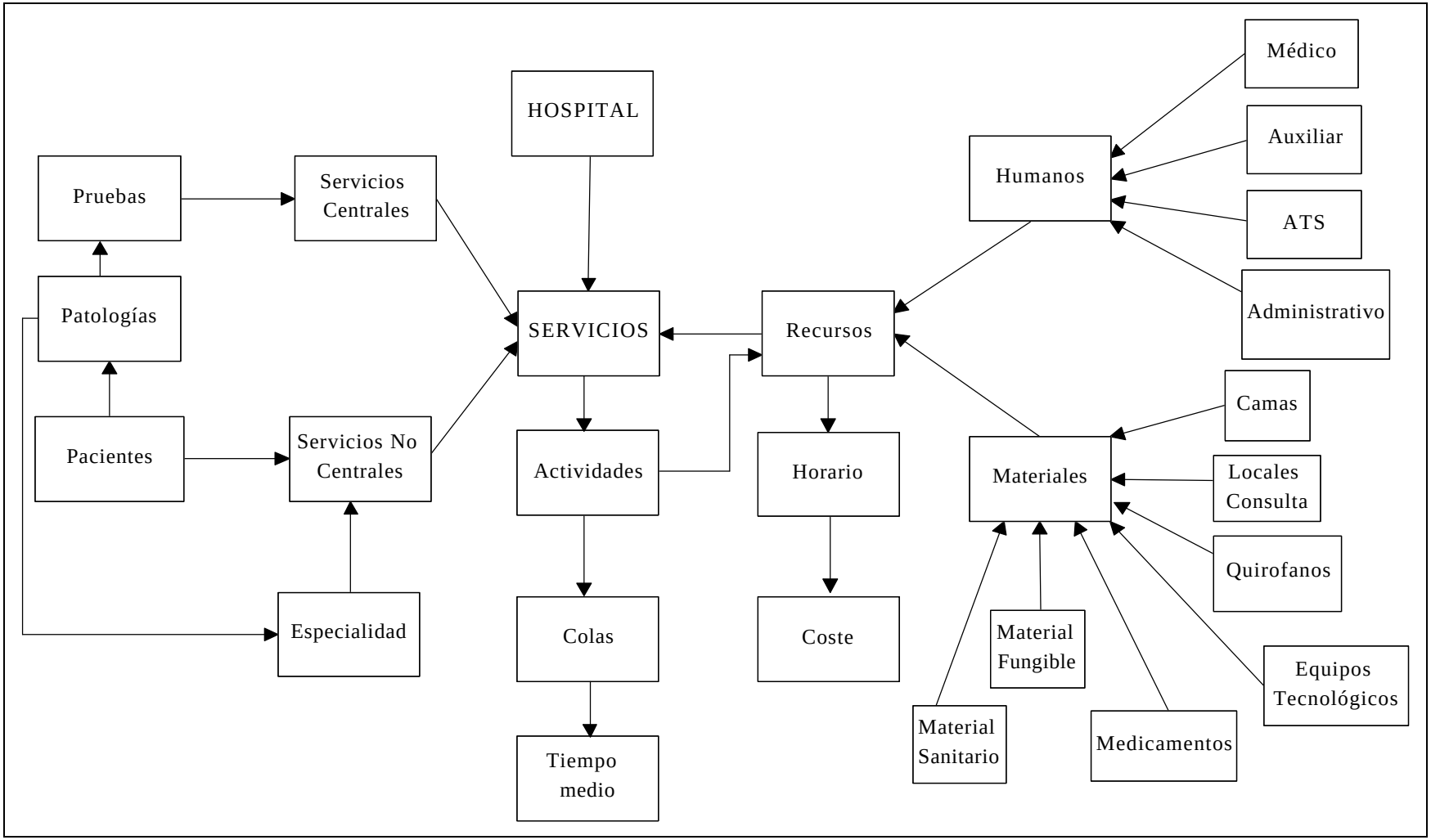


Figura 3.6.-Especificación del Dominio de un SBC para la ayuda en la Toma de Decisiones en la Gestión Hospitalaria.

En este trabajo se utiliza Ontolingua[2] como un lenguaje de descripción con el que se puede definir la Ontología del SBC para la ayuda a la toma de decisiones en la gerencia hospitalaria. En este sentido se considera Ontolingua como una herramienta de construcción de librerías de bases de conocimiento. Para especificar clases y relaciones se utilizan un conjunto de definiciones. El cuerpo de estas definiciones son sentencias KIF que expresan la semántica de los términos. Además de KIF, Ontolingua usa el "frame-ontology" que ofrece primitivas estándares para organizar el conocimiento. KIF y el frame-ontology se pueden ver como una ontología base o como un lenguaje especializado para la representación de conocimiento [Genesereth 1992] [Gruber 1993a], utilizado para la definición de ontologías más específicas. Asimismo, se reutilizarán ontologías ya definidas que tengan descritos conceptos o relaciones presentes en la teoría que se va a desarrollar. A continuación se describe la ontología asociada a un SBC para la gerencia hospitalaria.

[2] Ontolingua es un software formado por un analizador sintáctico de KIF (lenguaje para el intercambio de conocimiento entre distintos programas), herramientas para el análisis de ontologías, y un conjunto de traductores para convertir los ficheros fuentes escritos en ontolingua a un formato en el que se puede implementar un SBC tales como CLIPS, EPIKI, LOOM.

```
/* Primero necesitamos establecer el paquete donde se va a incluir esta ontología. Ontolingua-user es un buen paquete para escribir ontologías porque permite exportar símbolos desde KIF y paquetes escritos en ontolingua. Esto permite al usuario el uso de todos los símbolos definidos en la ontología KIF (por ejemplo: AND, OR, SETOF, ...) y en Frame-ontology (por ejemplo: VALUE-TYPE, HAS-ONE, ...) */
(in-package "ONTOLINGUA-USER")
/* A continuación definimos la teoría de la gestión hospitalaria. Esta teoría incluye el frame-ontology, la teoría llamada component-assemblies [Gruber 1994] y la teoría jat-generic [Hama 1992] que definen relaciones presentes en la descripción de un hospital desde el punto de vista de la gerencia hospitalaria */

(define-theory GERENCIA-HOSPITALARIA (frame-ontology component-assemblies jat-generic)
   "La ontología gerencia-hospitalaria define los términos usados para realizar la planificación hospitalaria. Esta teoría define las clases básicas (conceptos) usados en cualquier gestión hospitalaria, tales como hospital, servicio,... ."
)

(define-class HOSPITAL (?h)
   "Un hospital es un sistema que está formado por al menos un servicio"
      :def (and
            (component ?h)
            (value-type ?h HAS-SUBCOMPONENT Servicio)
            ( >= (value-cardinality ?h HAS-SUBCOMPONENT 1))))

(define-class SERVICIO (?s)
   "Un servicio puede ser de dos tipos diferentes y tiene recursos"
      :def (and
            (component ?s)
            (value-type ?s HAS-SUBCOMPONENT Recurso)
            ( >= (value-cardinality ?s HAS-SUBCOMPONENT 1))
```

```
            (EXHAUSTIVE-SUBCLASS-PARTITION ?s Tipo-servicios)
            (value-type ?s REALIZAN Actividades)
            ( and
                    (>= (value-cardinality ?s REALIZAN 1))
                    (>= (value-cardinality ?s REALIZAN 3)))))

(define-class TIPO-SERVICIOS (?ts)
   "Los servicios pueden ser servicios centrales o servicios no centrales"
      :axiom-def
            (SUBCLASS-PARTITION
                Tipo-servicios
                   (SET OF  Servicio-central Servicio-no-central)))

(define-class SERVICIO-CENTRAL (?sc)
   "Los servicios centrales son una clase de servicios que realizan pruebas clínicas"
      :def (and
            (SUBCLASS-OF ?sc Servicio)
            (value-type ?sc HAS-SUBCOMPONENT Pruebas-Clínicas)
            (>= (value-cardinality ?sc HAS-SUBCOMPONENT 1))
            (member ?sc (Setof Anatomía Patológica Bacteriología Farmacia
                    Hematología Laboratorio Medicina Nuclear Medicina Preventiva
                    Neurofisiología Oncología Radioterápica Radiodiagnóstico
                    Anestesiología-Reanimación UVI)))

(define-class SERVICIO-NO-CENTRAL (?snc)
   "Los servicios centrales son una clase de servicios que atienden las patologías de los
pacientes"
      :def (and
            (SUBCLASS-OF ?snc Servicio)
            (SUBCLASS-PARTITION ?snc Especialidad)
            (>= (value-cardinality ?snc SUBCLASS-PARTITION 1)
            (value-type ?snc HAS-SUBCOMPONENT Pacientes)
```

```
        (>= (value-cardinality ?sc HAS-SUBCOMPONENT 1))
        (member ?snc (Setof C-Vascular C-General Maxilofacial C-Pediátrica
            Tórax Ginecología-Obstetricia Neurocirugía Oftalmología
            Otorrinolaringología Traumatología Urología Alergia
            Braquiterapia Cardiología Endocrinología Dermatología Digestivo
            Hematología Custodiados Med-Interna Nefrología Neumología
            Neurología Oncología Radioterapia Psiquiatría Rehabilitación
            Reumatología Pediatría Urgencia)))

(define-class ACTIVIDAD (?act)
   "Los servicios realizan tres tipos de actividades donde atienden a los pacientes con el
uso de determinados recursos, la no disponibilidad de recursos origina una cola en la
actividad"
      :def (and
            (EXHAUSTIVE-SUBCLASS-PARTITION ?act Tipo-actividad)
            (value-type ?act UTILIZA Recurso)
            ( >= (value-cardinality ?s UTILIZA 1))
            (value-type ?act GENERA Cola)
            (=(value-cardinality ?act GENERA 1))))

(define-class TIPO-ACTIVIDAD (?ta)
   "Los servicios realizan actividades en consulta y en quirófano"
      :axiom-def
            (SUBCLASS-PARTITION
                Tipo-actividad
                    (SET OF  Consulta Quirúrgica)))

(define-class CONSULTA (?c)
   "Las actividades relacionadas con la consultan son Primera Consulta cuando el
paciente va por primera vez a ese servicio y Consulta Sucesiva cuando repite consulta el
paciente"
      :axiom-def
```

```
            (SUBCLASS-PARTITION
                Consulta
                    (SET OF  Primera-Consulta Consulta-Sucesiva)))

(define-class QUIRÚRGICA (?q)
   "Las actividades quirúrgica incluyen una operación en quirófano, se clasifican en actividades quirúrgicas en las que los pacientes tienen que ser ingresados en planta y actividades quirúrgicas en la que los pacientes no ingresan en planta "
        :axiom-def
            (SUBCLASS-PARTITION
                Quirúrgica
                    (SET OF  Planta-ActividadQuirurgica   ActividadQuirúrgica)))

(define-class RECURSO (?r)
   "Los recursos son objetos utilizados por el servicio para atender a los pacientes"
        :def (and
            (individual ?r)
            (SUBCOMPONENT-OF ?r Servicio)
            (TIEMPO-SERVICIO ?r Servicio Duration)
            (value-type ?r HAS-SUBCOMPONENT Horario)
            (=(value-cardinality ?r HAS-SUBCOMPONENT 1))
            (EXHAUSTIVE-SUBCLASS-PARTITION ?r Tipo-recurso)))

(define-class TIPO-RECURSO (?tr)
   "Los servicios pueden ser servicios centrales o servicios no centrales"
        :axiom-def
            (SUBCLASS-PARTITION
                Tipo-servicios
                    (SET OF  Recurso-humano Recurso-material)))
```

```
(define-function TIEMPO-SERVICIO (?recurso ?servicio ) -> ?duration
   "Cada servicio necesita sus recursos durante un periodo de tiempo"
      :def (and
            (Recurso ?recurso)
            (Servicio ?servicio)
            (Duration ?duration)))

(define-class HORARIO (?h)
   "Cada recurso tiene un horario que es un rango de tiempo durante el cual el servicio puede utilizar el recurso"
      :def(and
            (SUBCOMPONENT-OF ?h Recurso)
            (TIME-RANGE ?h)
            (value-type ?h HAS-SUBPART-SLOT Costo)
            (= (value-cardinality ?h HAS-SUBPART-SLOT 1))))

(define-relation GENERA (?actividad ?cola)
   "Cuando el recurso tiene un horario inferior al tiempo de servicio requerido por el servicio se genera una cola"
      :def (and
            (Actividad ?actividad)
            (UTILIZA ?actividad ?recurso)
            (SUBCOMPONENT-OF ?recurso ?servicio)
            (HAS-SUBCOMPONENT ?recurso ?horario)
            (Cola ?cola)
            (< ( TIME-RANGE.DURATION ?horario
                  (TIEMPO-SERVICIO ?recurso ?servicio))))

(define-relation UTILIZA (?actividad ?recurso)
   "Para realizar una actividad es necesario utilizar recursos"
      :def (and
            (Actividad ?actividad)
```

```
            (Recurso ?recurso)))

(define-class COLA (?cola)
   "Cada recurso tiene asociado una cola de pacientes esperando por el recurso. Luego la cola viene dada por un rango de enteros que nos informa del número de pacientes en espera"
      :def (and
            (individual ?cola)
            (INTEGER-RANGE ?cola)))

(define-class ESPECIALIDAD (?esp)
   "Cada servicio está dividido en un conjunto de especialidades que tratan determinadas patologías"
      :def (and
            (SUBCLASS-OF ?esp Servicio)
            (value-type ?esp HAS-SUBCOMPONENT Patología)
            (>= (value-cardinality HAS-SUBCOMPONENT 1))))

(define-class PACIENTE (?pac)
   "Los pacientes son atendidos en las especialidades atendiendo al tipo de patología que tienen"
      :def(and
            (SUBCOMPONENT-OF ?esp Servicio)
            (value-type ?pac HAS-SUBPART-SLOT Patología)
            (= (value-cardinality HAS-SUBPART-SLOT 1))))

(define-class PATOLOGÍA (?pat)
   "Las patologías caracterizan a los pacientes que según ésta son atendidos en una especialidad u otra"
      :def (and
            (individual ?pat)
            (SUBCOMPONENT-OF ?pat Especialidad)
```

(SUBPART-SLOT-OF ?pat Paciente)
(value-type ?pat REQUIERE Pruebas-Clínicas)
(>= (value-cardinality REQUIERE 0))))

(define-relation REQUIERE (?pat ?prucli)
"Atendiendo al tipo de patología se necesitan realizar al paciente un conjunto de pruebas clínicas que informen sobre su estado"
:def (and
(Patología ?pat)
(Pruebas-Clinicas ?prucli)
(domain REQUIERE ?pat)
(range REQUIERE ?prucli)))

(define-class PRUEBAS-CLINICAS (?prucli)
"Las pruebas clínicas las realizan los servicios centrales a los pacientes según la patología que tengan"
:def (and
(individual ?prucli)
(SUBCOMPONENT-OF ?prucli Servicio-Central)))

(define-function TIEMPO-MEDIO (?cola) -> ?tiempo
"Las colas tienen asociado un tiempo medio de servicio que indica el número de días que están esperando en media los pacientes en cola"
:def (and
(Cola ?cola)
(Duration ?tiempo)))

(define-class COSTO (?c)
"El horario de cada recurso tiene asociado un costo"
:def (and
(SUBPART-SLOT-OF ?c Horario)
(value-type ?c HAS-SUBPART-SLOT Valor)

```
            (= (value-cardinality ?c HAS-SUBPART-SLOT 1))))

(define-class VALOR (?v)
   "El valor del costo del horario de un recurso es un número real positivo que indica el
costo en pesetas necesarias para tener dicho recurso disponible en ese horario"
      :def (and
            (real-number ?v)
            (>= ?v 0)))

(define-class RECURSO-HUMANO (?rh)
   "Las personas que trabajan en el hospital"
      :def
            (Recurso ?rh)
      :axiom-def
            (subclass-partition
                  RECURSO-HUMANO
                  (setof medico, ATS, Administrativo, Auxiliar)))

(define-class RECURSO-MATERIAL (?rm)
   "Los materiales utilizados en el hospital"
      :def
            (Recurso ?rm)
      :axiom-def
            (subclass-partition
                  RECURSO-MATERIAL
                  (setof Camas Quirófanos Locales-consulta Equipos-Tecnológicos
                        Medicamentos Material-sanitario Otro-material-fungible)))
```

6.2.- Conocimiento de Control

El conocimiento de control describe cómo se realiza el razonamiento con el conocimiento del dominio, en términos de operaciones elementales de razonamiento sobre el conocimiento de dominio, y en términos de estructuras de control y descomposición del todo en un conjunto de tareas.

KADS distingue, dentro del conocimiento de control, entre el conocimiento de inferencia (conocimiento sobre cómo usar el dominio en los pasos elementales) [Aben 1995], y conocimiento de tareas (que describe cómo descomponer los trabajos en subtareas y cómo se realiza el control sobre las operaciones elementales) [Chandrasekaran 1983].

La capa de inferencia relaciona el conocimiento de control con el conocimiento de dominio, esto es, el conocimiento de inferencia describe las operaciones básicas que queremos realizar sobre el conocimiento de dominio. Una inferencia opera sobre algún dato de entrada y tiene la capacidad de producir una nueva pieza de información como salida. Los distintos tipos de inferencia describen la forma en que los conceptos, relaciones o estructuras pueden ser usadas para hacer una inferencia. Estas consisten de una entrada, una fuente de conocimiento o cuerpo, y una salida. La fuente de conocimiento o cuerpo de la inferencia lleva a cabo una acción sobre los datos de entrada y produce nueva información como salida. Una clasificación de los tipos de inferencia comúnmente utilizados se muestran en la figura 3.7.

Las inferencias nos definen las relaciones entre el conocimiento de control y el de dominio a través de los roles de conocimiento [Coelho 1996b].

Un role de conocimiento es una etiqueta abstracta que indica el papel que juega un conocimiento de dominio en un proceso de inferencia.

La inferencia es la definición declarativa de la relación entre los roles de conocimiento de entrada y los roles de conocimiento de salida en un proceso de inferencia. La única referencia que una inferencia hace al conocimiento de dominio es mediante los roles de conocimiento de entrada, de salida y los estáticos. El único conocimiento de dominio que es afectado por una inferencia es el referido en el role de

conocimiento de salida. Las inferencias tienen al menos un role de conocimiento de entrada y exactamente un único role de conocimiento de salida.

La estructura de inferencia es un diagrama que muestra la interconexión conceptual de las inferencias por medio de los roles de conocimiento [Aben 1995]. En este diagrama los rectángulos representan los roles de conocimiento, los óvalos las acciones de las inferencias y las flechas se usan para indicar las dependencias de los roles de conocimiento de entrada/salida. La estructura de inferencia no especifica el flujo de control entre las inferencias, el cual es definido en la capa de tareas.

Los roles de conocimiento no tienen estructura interna, son definidos por su nombre. Sin embargo, para describir las inferencia de otra manera que con las relaciones abstractas de las etiquetas, los roles incluyen los requerimientos ontológicos que sobre el conocimiento del dominio tiene un role particular.

Como una inferencia es una descripción declarativa de las relaciones de entrada/salida, no debería especificar ningún control que indique cómo aplicar la inferencia. Por ejemplo, si la aplicación de una inferencia implica una búsqueda en la capa de dominio, la especificación de la técnica de búsqueda particular no es parte de la inferencia, sino del método que realiza la inferencia.

Las inferencias deberían describir únicamente los procesos de inferencia interesantes, esto es, el nivel de detalle que se debe especificar es una decisión de modelado y depende de la situación particular.

Se han definido las inferencias como la relación entre los roles de entrada y de salida, sin embargo, para definir claramente las inferencias debemos especificar sus constituyentes:

a) Precondiciones: condiciones bajo las cuales una inferencia es aplicada. En la terminología utilizada en este texto se utilizará la palabra reservada **:cond** para especificar las precondiciones de una inferencia.

b) Cuerpo de la inferencia: la operación realiza por la inferencia. Especifica la relación entre los roles de entrada y salida de la inferencia. En este texto se describe por **:body**.

c) Postcondiciones: las condiciones del role de conocimiento de salida, u otras condiciones que se deben mantener después de haber aplicado la inferencia. Definido por **:result**.

Intuitivamente, la definición de la operación debería estar en el cuerpo de la inferencia y las consecuencias en las postcondiciones.

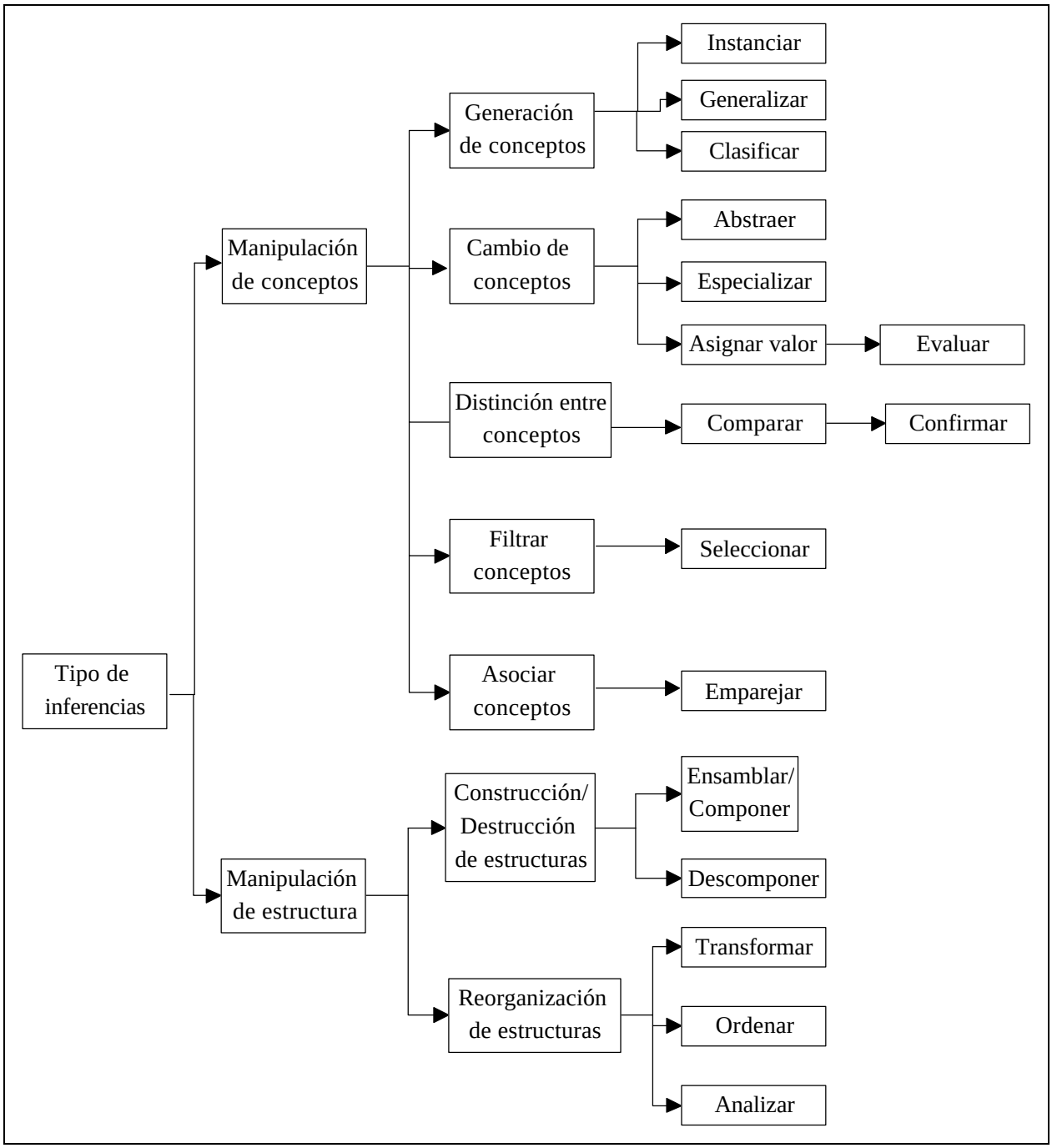


Figura 3.7.- Taxonomía de los tipos de inferencia comúnmente usados.

La capa de tareas está por encima de la capa de inferencia y describe cómo varios tipos de inferencia son combinados para lograr un objetivo concreto. El conocimiento de tareas describe cómo descomponer las tareas de alto nivel y cómo generar el control sobre las operaciones elementales. Por lo tanto, se puede definir tarea como un conjunto de actividades coherentes que realiza un agente para conseguir un objetivo.

El conocimiento acerca de como controlar el proceso de razonamiento en la resolución de un problema puede venir en una de las dos formas siguientes: un modelo completamente instanciado de como realizar el proceso de razonamiento para una aplicación específica, llamado modelo genérico de tareas [Breuker 1994]. O en forma de conocimiento que describe como descomponer un problema en subproblemas y cómo lograr subproblemas primitivos, denominado método de resolución de problemas [Benjamins 1995].

Un método de resolución de problemas puede ser considerado una receta de como construir parte de un modelo de tareas, y el modelo de tareas un método de resolución de problemas completamente instanciado.

Los modelos genéricos de tareas han sido desarrollados para problemas específicos [Cañamero 1996], y estas tareas se pueden utilizar como base en la construcción de otros SBC. Ejemplos de modelos de tareas desarrollados son: planificación, configuración, diseño y diagnóstico.

Los métodos de resolución de problemas se pueden clasificar en análisis, modificación y síntesis. Los métodos de análisis incluyen problemas en los que hay que identificar soluciones. En los métodos de modificación se consideran tareas de cambio de las condiciones del sistema tal que el sistema resultante puede o no ser el mismo que en el estado anterior, sin embargo, si que es de la misma clase que el sistema original. Cuando se habla de métodos de síntesis, el sistema resultante puede ser radicalmente diferente que el considerado inicialmente. En este caso en el sistema inicial se producen una combinación de cambios, algoritmos y adquisición de componentes, de forma que el sistema resultante puede o no ser de la misma clase o categoría que el sistema fuente. Métodos de resolución de problemas de este tipo sería el diseño, la configuración, la planificación, programación y modelado, figura 3.8.

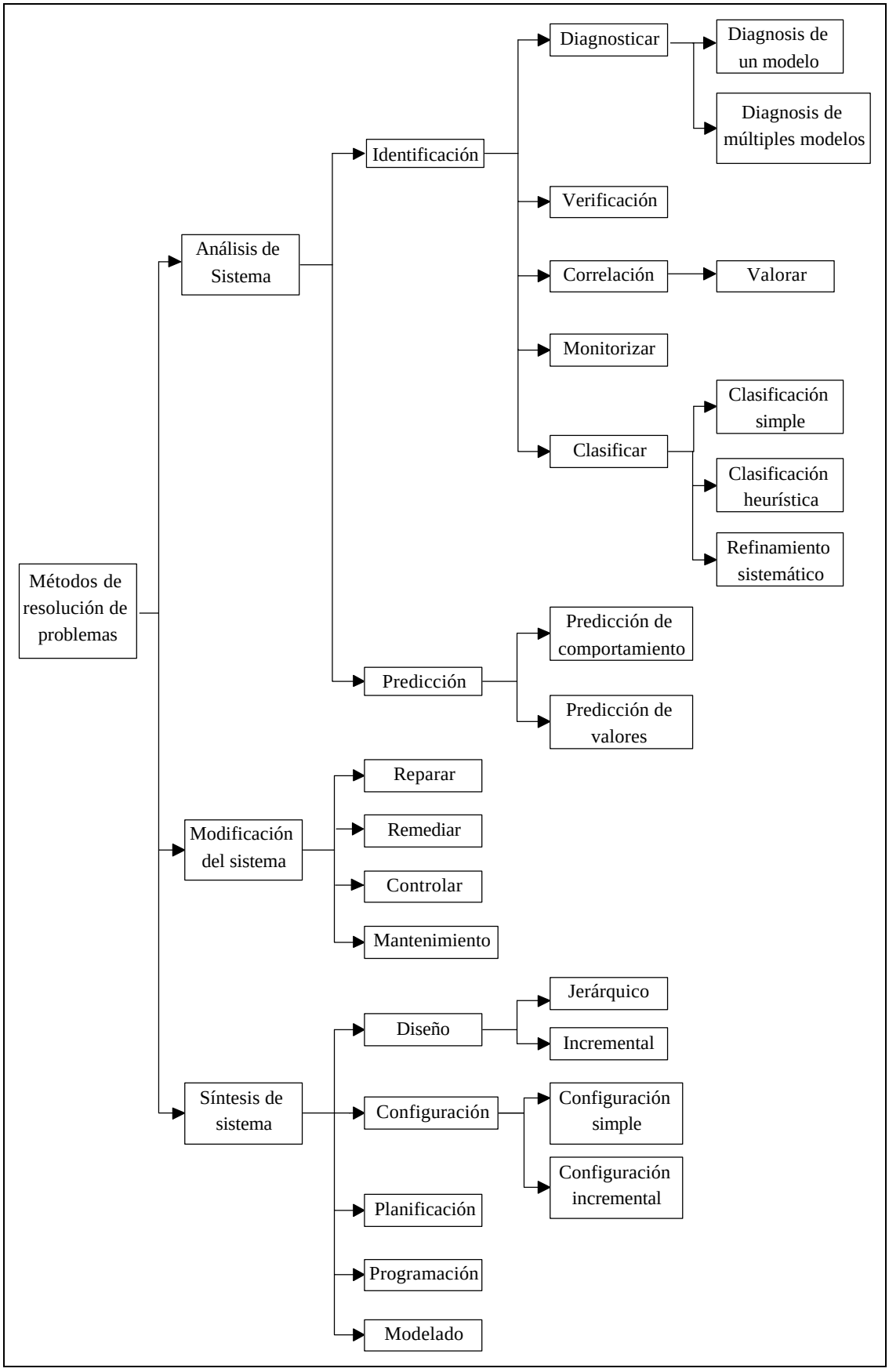


Figura 3.8.- Clasificación de los métodos de resolución de problemas.

La separación del conocimiento de dominio y del conocimiento de control permiten la reutilización del modelo experto [Breuker 1994]. Esto es debido a que todos los expertos humanos realizan un mismo conjunto de tareas cuando resuelven problemas tales como diagnóstico, planificación, ... Independientemente del área en la que trabajen, dado el mismo tipo de problema recogen datos y razonan con información similar. En el modelo experto tenemos definidos meta-modelos del conocimiento de control que se aplicarán ante el mismo problema aunque sea en dominios de aplicación diferentes.

6.2.1.- Toma de Decisiones en la Gestión Hospitalaria

El SBC de gestión hospitalaria debe realizar una distribución de servicios y recursos necesarios, cumpliendo un conjunto de restricciones, que permitan llevar a las variables del hospital al estado deseado. Por ello, el método de resolución de problemas que debemos aplicar es una "Toma de Decisiones", figura 3.9.

La toma de decisiones se descompone en las siguientes tareas [Molina 1996]:

a) Tarea de monitorización: detecta la existencia o no de problemas en el sistema.

b) Tarea de diagnóstico: responde a la pregunta de qué problema existe en el estado actual del sistema.

c) Tarea de predicción: nos indica que ocurrirá en el sistema si se cambian las condiciones de trabajo (ej. redistribución de recursos) suministrándonos una gran variedad de alternativas.

d) Tarea de diseño: determina qué se debería hacer para mejorar el estado actual del sistema, esto es, realiza una elección de la alternativa más adecuada de las propuestas por la tarea de predicción, de acuerdo con una función de costo o índice de calidad.

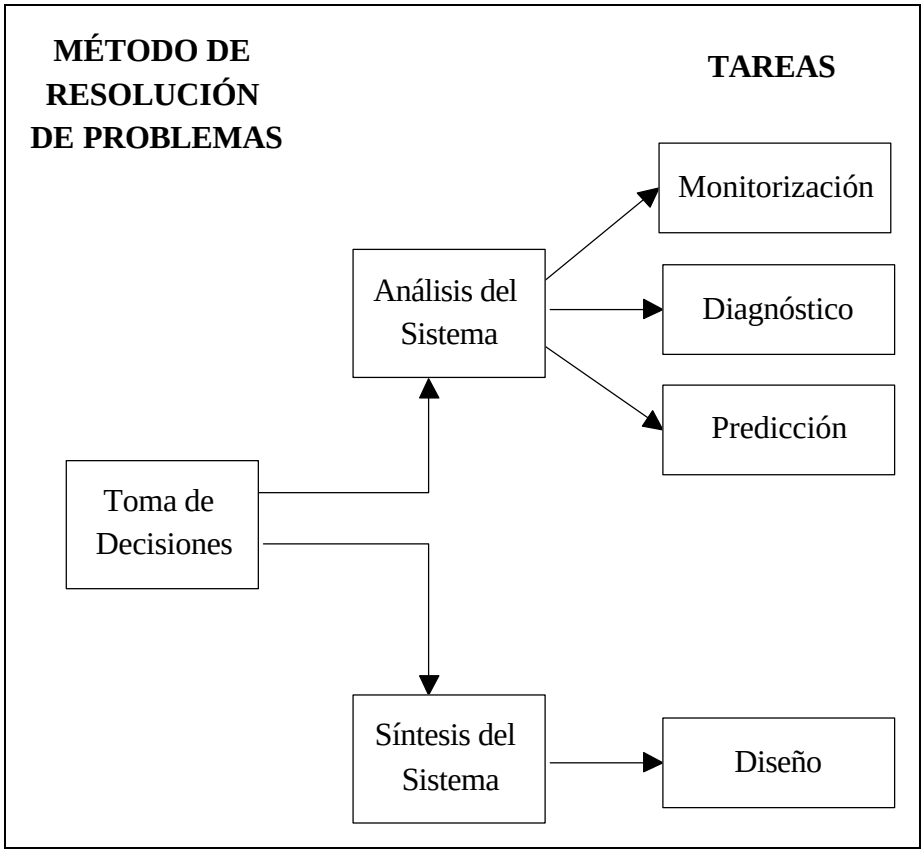


Figura 3.9.- Tareas necesarias para realizar una toma de decisiones

Para realizar una toma de decisiones tenemos que monitorizar a intervalos regulares de tiempo las variables que nos indican el estado del sistema. Si alguna de ellas se sale de los márgenes permitidos, se realiza una tarea de diagnóstico para averiguar cuál es la causa que ha originado cada uno de los problemas existentes. Una vez conocidas todas las causas se lleva a cabo la tarea de predicción de forma que se determinen un conjunto bastante amplio de soluciones locales a cada uno de los problemas existentes. Finalmente se ejecuta una tarea de diseño que analiza las soluciones locales y siguiendo una función de coste o índice de calidad, diseña la mejor solución global como un conjunto de soluciones locales, donde se han seleccionado aquellas que en conjunto reúnen los requisitos exigidos.

La tarea de toma de decisiones esquemáticamente sería como sigue:

```
toma_de_decisiones =
    monitorización (modelo_sistema ⇒ parámetros_erróneos)
    Si parámetros_erróneos ≠ ∅
            diagnóstico (parámetros_erróneos ⇒ problemas_actuales)
            predicción (problemas_actuales ⇒ soluciones_locales_posibles)
            diseño (soluciones_locales_posibles ⇒ solución_global_válida)
    Fin Si
```

A continuación vamos a describir las tareas necesarias en la toma de decisiones. Cada tarea está formada por una serie de inferencias, por lo que el siguiente paso será la definición de todas las inferencias.

El lenguaje utilizado en la definición de las inferencias [Coelho 1995], en este texto, es un lenguaje de emparejamiento de patrones (pattern matching). Un patrón es una descripción de un elemento de conocimiento que se puede utilizar en cualquier parte de la inferencia (:cond, :body, :result) para especificar los elementos con los que se trabaja.

Un patrón consiste en una secuencia ordenada de una o más restricciones. Una restricción intenta reconocer un valor o conjunto de valores, y para ello genera un test en los elementos de conocimiento. El patrón estará activo o instanciado si y sólo si existen piezas de conocimiento que satisfacen todos los tests generados por las restricciones del patrón. Los tipos de restricciones que nos podemos encontrar son:

- Variables libres (?x).- comprueba si existe algún elemento de conocimiento que verifique las restricciones exigidas. Ej: (Hospital ?h), busca los elementos de conocimiento ?h tal que sean un Hospital.

- Restricción operativa.- se aplican uno o más tests. En nuestro caso utilizamos notación prefija y las conexiones de las restricciones se realizan por medio de los operadores:

AND, OR, NOT, <, =, >, >>, <<

La asignación de un valor a una variable libre (?h) se hacer por medio del operador BIND. Ej: (bind ?h Hospital), la variable ?h se le asigna un elemento de

conocimiento con la característica de ser un Hospital.

6.2.1.1.- Tarea de Monitorización

La primera tarea a realizar es la monitorización. Para ello, el SBC toma a intervalos regulares de tiempo (semana) los valores de las variables de estudio del sistema, es decir, aquellas variables que determinan el estado del mismo.

El sistema se divide en componentes y para cada uno de ellos se observa el valor de las variables que los definen. Si estos valores no se encuentran dentro de los márgenes permitidos se le comunica a la tarea de diagnóstico para que determine cuáles son los problemas existentes.

La tarea de monitorización recibe como datos de entrada el sistema a estudiar y produce cómo salidas las variables que tengan valor erróneo. Este tipo de monitorización se denomina dirigida al modelo porque siempre la comienza el SBC, en contraposición con la monitorización dirigida por los datos, que se ejecuta debido a la llegada de un dato de entrada al SBC.

Los pasos elementales para llevar a cabo la tarea de monitorización son los siguientes:

```
monitorización =
    seleccionar_componente (modelo_sistema ⇒ componente)
    Para cada componente
        seleccionar_variable (componente ⇒ variables_estudio)
        Para cada variables ∈ variables_estudio
            especificar (variable ⇒ valor_normal)
            obtener (variable ⇒ valor_observado)
            comparar (diferencia ⇒ valor_normal - valor_observado)
            clasificar_erroneo (diferencia ⇒ variables_valor_erróneo)
```

6.2.1.2.- Inferencias realizadas en la monitorización

La estructura de inferencia [Coelho 1996] de la tarea de monitorización se observa en la figura 3.10.

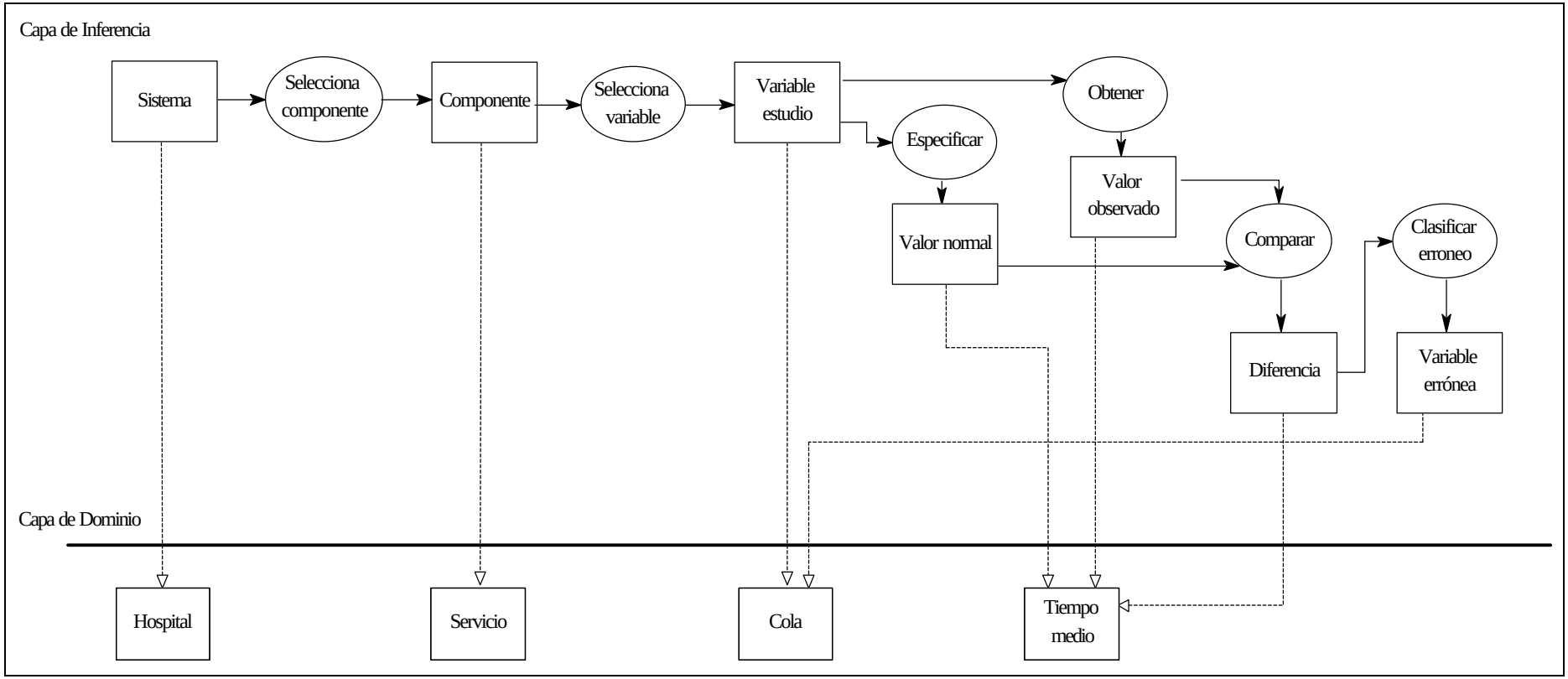


Figura 3.10.- Estructura de inferencia de la tarea de monitorización.

La definición de cada inferencia en término de las piezas de conocimiento de dominio que utiliza es la siguiente:

Inferencia 1 Selecciona-componente (modelo_sistema Þ componente)

"Esta inferencia recibe como entrada el sistema en estudio, en nuestro caso el hospital, y genera como salida todos los componentes del mismo, esto es, los servicios que componen el hospital"

define-inference SELECCIONA-COMPONENTE

"La condición exige que exista un elemento de conocimiento tal que sea un hospital."

"El cuerpo de la inferencia debe seleccionar todas las piezas de conocimiento tal que sean un servicio y que sean un subcomponente del hospital"

"El resultado genera todos los servicios que son componentes del hospital"

:cond (exist ?h

(Hospital ?h))

:body (forall ?s

(Servicio ?s)

(SUBCOMPONENT-OF ?s ?h))

:result (Selecciona-componente ?s)

Inferencia 2 Selecciona-variable (componente Þ variable_estudio)

"Para determinar el estado de un servicio se observa cada una de las actividades que realiza el mismo."

define-inference SELECCIONA-VARIABLE

"La condición se activa para todos los servicios obtenidos en la inferencia selecciona-componente"

"En el cuerpo de la inferencia se seleccionan todas los piezas de conocimiento tal que sean una actividad y que se realice en el servicio"

"En el resultado se generan todas las parejas servicio-actividad posibles"

:cond (forall ?s (Selecciona-componente ?s))

:body (forall ?act

(and

(Actividad ?act)

(REALIZA ?s ?act)

:result (Selecciona-variable (?s ?act))

Inferencia 3 especificar-valor-normal (variable Þ valor_normal)

"Cada actividad tiene asociado una cola, y cada cola tiene un tiempo medio cuyo valor indica si la actividad funciona correctamente. Para conocer las desviaciones y por lo tanto, los errores en las colas se necesita obtener el valor correcto (normal) de cada una de ellas."

define-inference ESPECIFICAR-VALOR-NORMAL

"La condición se activa para cada par servicio-actividad obtenido en la inferencia selecciona-variable"

"En el cuerpo de la inferencia se seleccionan todas las colas de las actividades y se obtiene el tiempo normal de cada cola"

"En el resultado se generan la terna formada por todos los servicios con sus actividades y el tiempo normal de cada una de ellas."

:cond (forall ?s ?act (Selecciona-variable ?s ?act))

:body (and (GENERA ?act ?cola)

(TIEMPO-MEDIO-NORMAL ?cola ?tiempo))

:result (Valor-normal ?s ?act ?tiempo)

Inferencia 4 obtener-valor-observado (variable Þ valor_observado)

"Es necesario conocer el valor que tiene en un momento dado las colas de cada actividad con el objetivo de determinar la desviación que existe con respecto al valor normal"

define-inference OBTENER-VALOR-OBSERVADO

"La condición se activa para cada par servicio-actividad obtenido en la inferencia selecciona-variable"

"En el cuerpo de la inferencia se seleccionan todas las colas de las actividades y se obtiene el tiempo observado de cada cola"

"El resultado genera la terna formada por todos los servicios con sus actividades y el tiempo observado de cada una de ellas."

:cond (forall ?s ?act (Selecciona-variable ?s ?act))

:body (and (GENERA ?act ?cola)

(TIEMPO-MEDIO-OBSERVADO ?cola ?tiempo))

:result (Valor-observado ?s ?act ?tiempo)

Inferencia 5 comparar (diferencia Þ valor_normal - valor_observado)

"Se debe determinar cómo se aleja el valor observado de cada cola del valor normal que debería tener."

define-inference COMPARAR

"La condición se activa para el par de elementos de conocimiento que cumplan que sea un servicio con una actividad, con un valor normal para la actividad y un valor observado"

"El cuerpo de la inferencia asocia a una pieza de conocimiento la diferencia entre el valor normal y el observado de la actividad del servicio"

"El resultado genera la terna formada por cada servicio con sus actividades y la diferencia de tiempo de la colas"

:cond (forall ?s ?act

(and

(Selecciona-variable ?s ?act)

(Valor-normal ?s ?act ?t-normal)

(Valor-observado ?s ?act ?t-observado)))

:body (bind (?diferencia (- (?t-observado ?t-normal))))

:result (Comparar ?s ?act ?diferencia)

Inferencia 6 clasificar-erroneo (diferencia Þ variable_valor_erróneo)

"Si la diferencia es muy grande o muy pequeña significa que el valor observado se aleja del valor normal de la cola y por lo tanto la actividad presenta un problema"

define-inference CLASIFICAR-ERRONEO

"La condición se activa para todos los servicios y sus actividades tales que se conozca la diferencia entre el valor normal y el observado de la cola de la actividad"

"El cuerpo de la inferencia se instancia tanto si diferencia es muy grande o si la diferencia es muy pequeña"

"El resultado considera el servicio con esa actividad con un funcionamiento erróneo"

:cond (forall ?s ?act ?diferencia (Comparar ?s ?act ?diferencia))

:body (or

(>> (?diferencia))

(<< (?diferencia)))

:result (Clasificar-erroneo ?s ?act ?diferencia)

6.2.2.- Tarea de Diagnóstico

Esta tarea analiza el estado actual del sistema para diagnosticar los problemas existentes. Sin embargo, dada la complejidad del sistema no se puede estudiar globalmente el mismo, y debemos analizar por separado sus distintos componentes. El resultado del diagnóstico total es la unión de los diagnósticos locales de los distintos componentes.

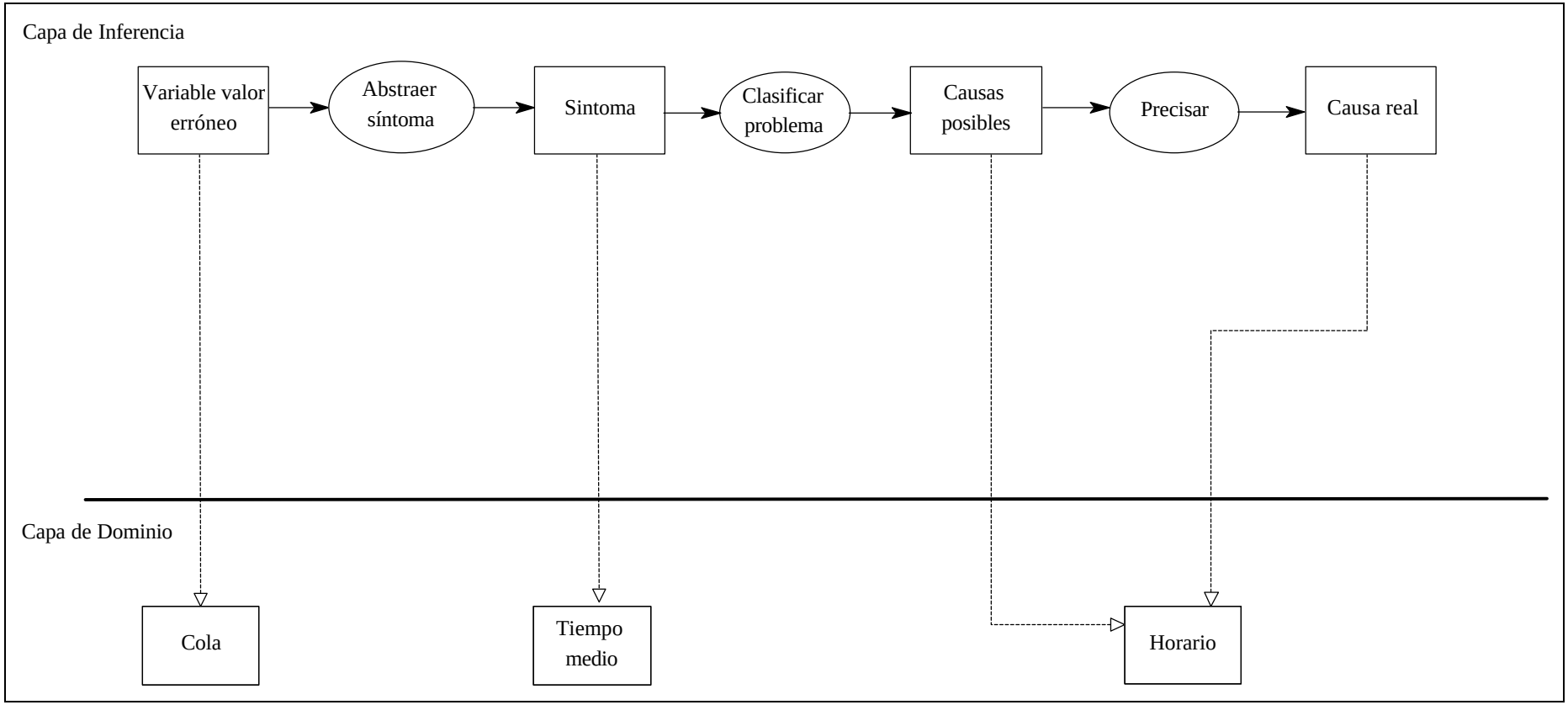


Figura 3.11.- Estructura de inferencia de la tarea de diagnóstico.

El diagnóstico local se realiza a través de un proceso de clasificación heurística en el que debemos:

a) Abstraer el problema:

A partir de los valores no correctos de las variables en estudio se obtienen los síntomas de los problemas.

b) Identificar el mal funcionamiento:

Los síntomas son utilizados para detectar las posibles causas del problema.

c) Precisar el mal funcionamiento:

Una vez identificado el posible problema hay que refinar la clasificación para determinar las características específicas del mismo.

La tarea de diagnóstico recibe como datos de entrada las variables que tienen valores no correctos. Como salida obtiene la causa que en cada componente del sistema ha generado el valor incorrecto en las variables de estudio.

La descripción de la tarea de diagnóstico en sus pasos más elementales es la siguiente:

```
diagnóstico (componentes, variables_valor_erróneo) =
    Para cada variable ∈ variables_valor_erróneo
        abstraer-sintoma (variable_valor_erróneo ⇒ síntoma)
        clasificar-problema (síntoma ⇒ causas_posibles)
        precisar (causas_posibles ⇒ causa_real)
```

6.2.2.1.- Inferencias realizadas en el Diagnóstico

La estructura de inferencia se observa en la figura 3.11.

La definición de cada inferencia en término de las piezas de conocimiento de dominio que utilizan es la siguiente:

Inferencia 1 abstraer-sintoma (variable_valor_erróneo Þ síntoma)

"Si la diferencia entre el valor medido de la cola de una actividad concreta y el valor normal para la misma es muy grande significa que la cola en esa actividad ha aumentado. Mientras que si la diferencia es muy pequeña significará que la cola ha disminuido."

define-inference ABSTRAE-SINTOMA

"La condición se activa para la terna servicio-actividad-diferencia de tiempo de colas obtenida en la inferencia clasificar-erroneo"

"El cuerpo de la inferencia asocia a síntoma un aumento de cola si la diferencia es muy grande, o si la diferencia es muy pequeña asocia a síntoma una disminución de cola"

"El resultado genera la terna formada por cada servicio con sus actividades y su síntoma"

```
:cond (forall ?s ?act ?diferencia (Clasificar-erroneo ?s ?act ?diferencia))
:body
        (or
                (and
                        (>> (?diferencia))
                        (bind (?sintoma Aumento-cola)))
                (and
                        (<< (?diferencia))
                        (bind (?sintoma Disminucion-cola)))))
:result (Abstrae-sintoma ?s ?act ?sintoma)
```

Inferencia 2 clasificar-problema (síntoma Þ causas_posibles)

"Atendiendo al sintoma que presenta cada actividad se determina el posible problema. Inicialmente se considera que si una actividad tiene un sintoma en el que aumenta la cola el problema sería una escasez de recursos en el servicio donde se realiza esa actividad. Por otro lado, si el síntoma es una disminución de cola el posible problema asociado al servicio de la actividad es una abundancia de recursos."

define-inference CLASIFICAR-PROBLEMA

"La condición se activa para todas las ternas servicio-actividad-sintoma generadas en la inferencia abstrae-sintoma"

"El cuerpo de la inferencia asocia a la pieza de conocimiento posible-problema una escasez de recurso si el síntoma es un aumento de cola, o asocia a posible-problema una abundancia de recurso si el síntoma es disminución de cola"

"El resultado genera la terna formada por servicio-actividad-posible problema"

:cond (forall ?s ?act ?sintoma (Abstrae-sintoma ?s ?act sintoma))

:body (or

(and

(= (?sintoma Aumento-cola))

(bind (?posible-problema Escasez-recurso)))

(and

(= (?sintoma Disminucion-cola))

(bind (?posible-problema Abundancia-recurso))))

:result (Clasificar-Problema ?s ?act ?posible-problema)

Inferencia 3 precisar (causas_posibles Þ causa_real)

"Para determinar el problema exacto se estudia el caso de las actividades con problemas que pertenecen a servicios no centrales. El problema en estos casos puede venir ocasionado por un problema en los servicios centrales que realizan las pruebas clínicas a los pacientes de los servicios no centrales. Por ello si existe un problema de abundancia de recursos en un servicio no central se observa la posibilidad que un servicio central relacionado tenga un problema de escasez de recursos, esto se hace observando si la cola del servicio central es muy grande. El caso contrario es un servicio no central con un problema de escasez de recursos hay que determinar si existe un servicio central relacionado que tenga un problema de abundancia de recursos"

define-inference PRECISAR

"La condición se activa para todas las ternas servicio-actividad-posible problema generada en la inferencia Clasificar-problema y tal que el servicio sea un servicio no central."

"El cuerpo de la inferencia busca una pieza de conocimiento que sea un servicio central, que realiza actividades y que está formado por pruebas clínicas las cuales son necesarias en las patologías de los pacientes del servicio no central. En este caso pueden ocurrir cuatro situaciones:

a) Si el valor observado de las actividades del servicio central es muy superior al valor normal y el posible problema en el servicio no central es una abundancia de recursos se refina el problema como una escasez de recursos en el servicio central y la actividad primera consulta.

b) Si el valor observado de las actividades del servicio central es aproximadamente igual al valor normal y el posible problema en el servicio no central es una abundancia de recursos se verifica que el problema es abundancia de recurso en el servicio no central.

c) Si el valor observado de las actividades del servicio central es muy inferior al valor normal y el posible problema en el servicio no central era una escasez de recursos se refina el problema como una abundancia de recursos en el servicio central y la actividad primera consulta.

d) Si el valor observado de las actividades del servicio central es aproximadamente igual al valor normal y el posible problema en el servicio no central era una escasez de recursos se determina el problema como una escasez de recursos en el servicio no central."

"El resultado de la inferencia es la terna servicio-actividad-problema"

```
:cond (forall ?s ?act ?posible-problema
        (and
                (Clasificar-Problema ?s ?act ?posible-problema)
                (Servicio-no-central ?s)))
:body (exist ?sc
        (and
                (Servicio-Central ?sc)
                (REALIZA ?sc ?actsc)
                (HAS-SUBCOMONENT ?s ?pacientes)
                (HAS-SUBPART-SLOT ?pacientes ?patología)
                (REQUIERE ?patología ?Prueba-Clinica)
                (HAS-SUBCOMPONENT ?sc ?Prueba-Clinica)
                (or
        (and
        (>>  (Valor-observado ?sc ?actsc ?tiempo-observado)
```

```
            (Valor-normal ?sc ?actsc ?tiempo-normal))
        (= (?posible-problema   Abundancia-recurso))
        ( ⇒ (bind (?problema Escasez-recurso))
            (bind (?servicio- problema ?sc))
            (bind (?act ?Primera-consulta))))
        (and
        (≈  (Valor-observado ?sc ?actsc ?tiempo-observado)
            (Valor-normal ?sc ?actsc ?tiempo-normal))
        (= (?posible-problema   Abundancia-recurso))
        ( ⇒ (bind (?problema  ?posible-problema))
            (bind (?servicio- problema ?s))))
        (and
        (<< (Valor-observado ?sc ?actsc ?tiempo-observado)
            (Valor-normal ?sc ?actsc ?tiempo-normal))
        (= (?posible-problema   Escasez-recurso))
        ( ⇒ (bind (?problema Abundancia-recurso))
            (bind (?servicio- problema ?sc))
            (bind (?act ?Primera-consulta))))
        (and
        (≈  (Valor-observado ?sc ?actsc ?tiempo-observado)
            (Valor-normal ?sc ?actsc ?tiempo-normal))
        (= (?posible-problema   Escasesez-recurso))
        ( ⇒ (bind (?problema  ?posible-problema))
            (bind (?servicio-problema ?s))))
            )
        ))
:result (Precisar-Servicio-Problema ?servicio-problema  ?act  ?problema)
```

6.2.3.- Tarea de Predicción

En esta tarea se estudian para un estado inicial dado y un estado final que se quiere alcanzar, diferentes estrategias plausibles que conecten el estado inicial con el final.

Para realizar esta tarea, igual que se hizo en la tarea de diagnóstico, hay que dividir el sistema total en sus componentes y realizar una tarea de predicción local para cada componente.

Para realizar la predicción local utilizamos el Agente Simulación que simula el comportamiento futuro de los distintos componentes. Los pasos a realizar son los siguientes:

1.- Generar las posibles acciones de control.

2.- Simular el comportamiento del sistema bajo las distintas acciones de control, para predecir el efecto de tales acciones [Simon 1995].

3.- Seleccionar el conjunto de las mejores soluciones locales para cada componente.

El método de resolución de problemas utilizado en la predicción local es, como se observa, el genera y testea. Se generan un conjunto de acciones de control que puedan resolver el problema_actual de cada componente, y luego se testea el impacto de cada propuesta mediante la simulación.

El paso de generar es realizado por una tarea básica llamada "genera acciones de control" que usa un modelo de acciones de control para proponer dichas acciones.

La tarea de testeo es realizada por la simulación que hace una predicción del impacto de las acciones propuestas.

La tarea de predicción recibe como datos de entrada las causas que han originado los problemas existentes, y como salida genera el conjunto de todas las posibles soluciones locales para cada componente.

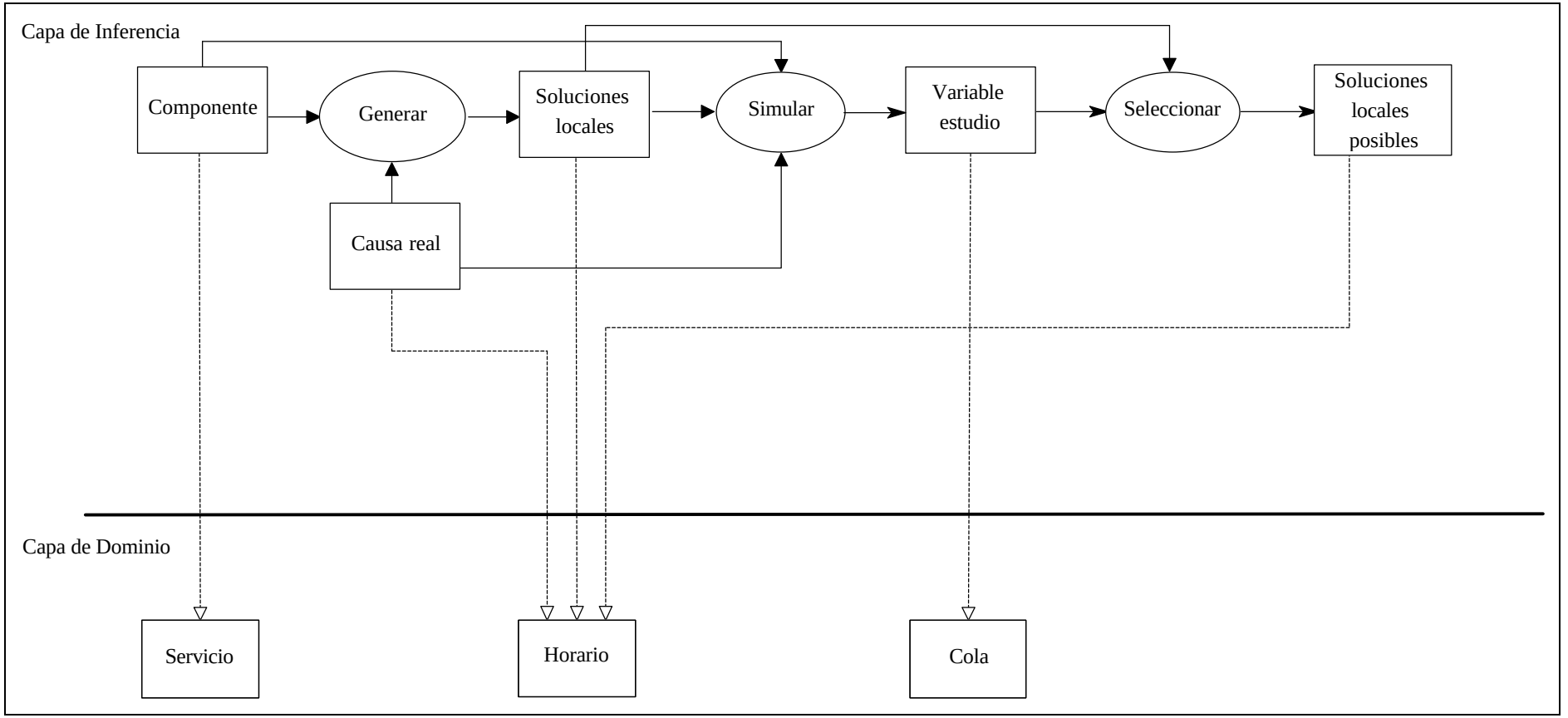


Figura 3.12.- Estructura de inferencia de la tarea de predicción.

La descripción de la tarea de predicción es la siguiente:

```
predicción (componentes, causa_real) =
Para cada causa ∈ causa_real
    generar(componente, causa_real ⇒conjunto_soluciones_locales)
    Para cada solución ∈ conjunto_soluciones_locales
       simular(componente, causa_real, solución ⇒ variable_estudio)
    seleccionar(conjunto_soluciones_locales, variable_estudio ⇒
            soluciones_locales_posibles)
```

6.2.3.1.- Inferencias realizadas en la Predicción

La estructura de inferencia de la tarea de predicción se muestra en la figura 3.12. La definición de cada inferencia en término de las piezas de conocimiento de dominio que utilizan es la siguiente:

Inferencia 1 generar(componente, causa_real Þ conjunto_soluciones_locales)

"Una vez determinado el servicio y la actividad que tiene un problema hay que generar las soluciones. Estas vendrán dadas por un aumento del horario de los recursos si el problema es de escasez de recursos o viceversa (disminución del horario de los recursos si el problema es una abundancia de recursos)."

define-inference GENERAR

"La condición se activa para todas las ternas servicio-actividad-problema generada en la inferencia precisar-servicio-problema"

"En el cuerpo de la inferencia se obtienen todos los recursos que utiliza la actividad con problema y le asocia al horario de ese recurso un nuevo horario generado por el procedimiento modifica-horario."

"La salida de la inferencia es la cuaterna servicio-actividad-recurso-horario"

```
:cond (forall ?servicio ?act ?prob (Precisar-Servicio-Problema ?servicio ?act  ?prob))
:body (forall ?recurso
        (and
              (UTILIZA ?act ?recurso)
```

(HAS-SUBCOMPONENT ?recurso ?horario)

(⇒ (bind (?horario (MODIFICA-HORARIO ?recurso))))

))

:result (Genera ?servicio ?act ?recurso ?horario)

Inferencia 2 simular(componente, causa_real, solución Þ variable_estudio)

"Dada las posibles soluciones hay que determinar el grado de satisfacción que éstas ofrecen. Para ello se simula el servicio con el problema bajo la nueva modificación de horarios de recursos que se ha propuesto como solución a su problema"

define-inference SIMULA-SERVICIO

"La condición se activa para todas las cuaternas servicio-actividad-recurso-horario producidos por la inferencia genera"

"En el cuerpo de la inferencia se realiza la simulación del servicio (mediante un programa que realiza la simulación del hospital) después de la cual se obtiene mediante la inferencia selecciona-variable las actividades del servicio simulado"

"El resultado es la cuaterna servicio-actividad-recurso-horario obtenida después de la simulación del servicio"

:cond (forall ?servicio ?act ?recurso ?horario (Genera ?servicio ?act ?recurso ?horario))

:body(and

(SIMULAR ?servicio)

(Selecciona-variable (?servicio ?act))

:result (Simular ?servicio ?act ?recurso ?horario)

Inferencia 3 seleccionar(conjunto_soluciones_locales, variable_estudio Þ soluciones_locales_posibles)

"Una vez que se han simulado las posibles soluciones se seleccionan aquellas que cumplen con una determinada función de coste."

define-inference SELECCIONA-SOLUCIONES

"La condición se activa para todas las cuaternas servicio-actividad-recurso-horario generados en la inferencia simular"

"En el cuerpo de la inferencia se obtiene el coste asociado a cada recurso y se observa que cumpla que es el menor que el dado en el procedimiento indice-coste"

"El resultado es la cuaterna de servicio-actividad-recurso-horario"

:cond (forall ?servicio ?act ?recurso ?horario (Simular ?servicio ?act ?recurso ?horario))

:body (and

(HAS-SUBPART-SLOT ?horario ?costo)

(< (?costo INDICE-COSTE))

:result (Selecciona-soluciones ?servicio ?act ?recurso ?horario)

6.2.3.2.- Piezas de conocimiento complejas utilizadas en la predicción

Las inferencias describen los pasos primitivos que hay que realizar para la resolución del problema, aunque la implementación de cada inferencia puede ser una transformación inmediata o un proceso más complejo. A continuación vamos a describir las inferencias utilizadas en la tarea de predicción ya que su implementación no es simple.

6.2.3.2.1.- Inferencia modifica-horario

La implementación de esta inferencia se realiza con un programa que genera la acción de control que se debe aplicar en un servicio para eliminar el problema existente. Estas acciones de control consisten en un aumento o una disminución de los recursos asignados al servicio. Esto es, para disminuir (aumentar) el tiempo de espera de los pacientes de un servicio se debe aumentar (disminuir) el horario de los recursos que

necesita el mismo para ser atendido.

La disciplina de control nos brinda medios o mecanismos para lograr el funcionamiento óptimo de sistemas dinámicos, mejorar la calidad y abaratar costos, expandir el ritmo de producción, liberar de la complejidad de muchas rutinas, etc; por ello es utilizada en este caso para determinar la magnitud en la que se debe aumentar/disminuir el horario de los recursos del servicio cuyos valores de cola no están dentro de los márgenes permitidos.

La primera decisión que se ha tomado es realizar acciones de control locales, para luego construir la solución global, en la tarea de diseño, como combinación de soluciones locales compatibles (divide y vencerás). El estudio de las soluciones locales se puede realizar debido a que el hospital está formado por subsistemas que no están fuertemente acoplados, es decir, las interrelaciones existentes entre las partes permiten el estudio de cada una de ellas para luego componer la solución global de todo el sistema. Inicialmente no se busca la solución global debido a que ello implica fijar las prioridades de unos servicios frente a otros, con lo cual se están tomando decisiones sin conocimiento acerca del funcionamiento de los subsistemas y esto puede negar la posibilidad de utilizar soluciones correctas. Para evitar este problema se construye una tabla para cada servicio con problemas que informa sobre el conjunto de soluciones locales en cada caso, y que sirva como elemento de información en la toma de la decisión global.

En cada servicio se realiza un sistema de control en lazo cerrado como el que se muestra en la figura 3.13.

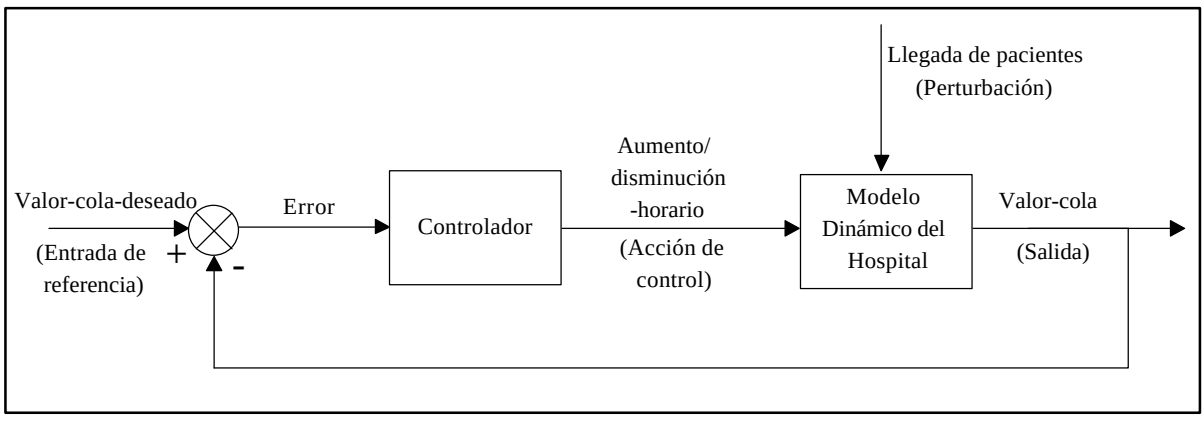


Figura 3.13.- Sistema de control en lazo cerrado producido en cada servicio del hospital.

El sistema de control está formado por los dos elementos siguientes:

a) Un programa que simula la dinámica del hospital, esto es, el flujo de pacientes a través de los distintos servicios del hospital. El tiempo medio de las colas es la variable de salida que debe ser controlada, ya que debido a la escasez de recursos en los distintos servicios se crean colas en las que los pacientes esperan para ser atendidos. El tiempo medio de estas colas debe mantenerse dentro de los márgenes permitidos de forma que el hospital mantenga la eficiencia exigida.

b) El controlador que observa las desviaciones producidas en los valores de las colas de cada servicios y las corrige aumentado o decrementando los recursos asociados al servicio.

El modelo dinámico del servicio informa sobre el valor de las colas en el servicio. El controlador compara este valor con el valor deseado por la gerencia hospitalaria. Si hay una diferencia, la salida del controlador le comunica al sistema el aumento/decremento de los recursos del servicio para alcanzar el valor deseado. Las variaciones en la llegada de pacientes se considera perturbaciones del sistema, por lo que el controlador es un sistema de rechazo de perturbación.

El objetivo sería diseñar un control óptimo [Athans 1966] [Lewis 1986] que minimizando una función de coste permita obtener el conjunto de comandos (acciones de control) que lleven al sistema del estado inicial (con valores de colas no adecuados) al estado final (con los valores de colas deseados).

6.2.3.2.1.1.- Programación Dinámica

Una aproximación inicial al problema fue recurrir a la programación dinámica como método de obtención de la trayectoria óptima que llevara al sistema del estado inicial al estado final deseado [Larson 1978] [Larson 1982].

La programación dinámica de Bellman es un método de resolución de problemas de óptimo pensado para ser resuelto con ordenador, mediante un procedimiento multietapa y que obtiene la trayectoria óptima de la secuencia de control a partir de operaciones elementales de comparación entre magnitudes escalares, no requiriendo, además, hacer ningún tipo de suposición acerca de las propiedades analíticas de la

ecuación de estado y función de coste.

El problema de optimización que se puede resolver mediante la programación dinámica es el siguiente: dado un sistema, cuya ecuación de estados discreta es x(k+1)=g[x(k),u(k),k] donde el vector de estado y control están sometidos a ligaduras, y el estado inicial x(0)=C. Se trata de encontrar una secuencia de control u(0),u(1),u(2),...,u(N) que satisfaciendo las ligaduras minimice la función de costo:

$$J = \sum_{k=0}^{N} L(x(k),u(k),k)$$

La base de la programación dinámica es el principio de optimalidad. Este principio puede ser establecido de la siguiente manera: dada una trayectoria óptima que une los puntos A y C, la parte de la trayectoria que une cualquier punto intermedio B con el punto C debe ser una trayectoria óptima entre los puntos B y C.

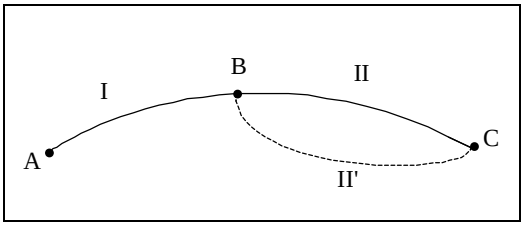


Figura 3.14.- Principio de optimalidad dice que si I-II es el camino óptimo de A-C entonces II es el camino óptimo de B-C.

Supongamos que el camino I-II, que muestra la figura 3.14, es una trayectoria óptima que va de A a C, de acuerdo con el principio de optimalidad el camino II es un camino óptimo que va de B a C. Se puede probar por contradicción: supongamos que otro camino tal como II' es la trayectoria óptima entre B y C. Entonces el camino I-II' tendrá un costo menor que la trayectoria I-II lo cual está en contradicción con la hipótesis.

Bellman estableció el principio diciendo que una política óptima tiene la propiedad que cualquiera que haya sido el estado inicial y la decisión inicial, entonces las restantes decisiones constituyen una política óptima con respecto al estado resultante de la primera decisión.

La programación dinámica exige que el espacio de estados esté cuantizado, así como el conjunto de decisiones o comandos de control. La variable de etapa tomará valores enteros 0,1,2, ... ,N, donde N es el número total de etapas. Esta variable de etapa normalmente es el tiempo, aunque en el caso de que en la función de coste se quiera incluir el tiempo la variable de etapa será cualquier otra variable que cumpla que sea monótona creciente o decreciente.

El procedimiento general de la programación dinámica sería el siguiente:

a) Consideremos un estado cuantizado $x \in X$ en la etapa k. En esta situación aplicamos cada uno de los controles $u \in U$ pertenecientes al conjunto discretizado.

b) Para cada uno de los controles se determina el costo, L(x,u,k), y se suma a la función de costo del siguiente estado de la etapa k+1 mediante la aplicación de la ecuación de estados, g(x,u,k). En el caso en el que un estado particular llegue a un estado que esté fuera del rango de estados admisibles, esta decisión ha de rechazarse como candidata para la secuencia de control.

c) El control óptimo en esta etapa y en este estado es el que minimice la función de costo:

$$I(x,k) = \min_{u \in U} \{ L(x,u,k) + I(g(x,u,k),k+1) \}$$

La estrategia comúnmente utilizada consta de dos pasos: un paso hacia atrás en el que se evalúa en cada etapa y en cada estado el valor del comando y el índice para llegar a la etapa siguiente. En el segundo paso se evalúa la trayectoria óptima a partir del estado inicial y de la información obtenida en el primer paso.

Siguiendo esta estrategia se consigue reducir el espacio de búsqueda porque al partir del estado final se asegura estudiar sólo las trayectorias que llevan desde cualquier estado inicial al final. Por lo tanto, dado el estado final (etapa N), se comienza calculando el comando óptimo que lleva de la etapa N-1 a la etapa N, y así se continua hasta llegar al estado inicial.

En el caso del hospital esta estrategia no se puede seguir porque aunque se conoce el estado final al que deseamos llegar (estado en el cual el valor medio de las colas esté en un valor específico) y cual sería el estado en la etapa N-1, hacer

evolucionar el sistema de la etapa N-1 a la etapa N no es posible debido a que el estado viene dado por el valor medio de las colas de espera (variable macroscópica) y para hacer evolucionar el sistema necesitamos el estado microscópico del mismo, esto es, la situación de cada uno de los pacientes en cada una de las colas. Un valor de tiempo medio de espera puede corresponder a una gran variedad de situaciones de los pacientes en el sistema.

Sin embargo, el primer paso de la Programación Dinámica puede ser también hacia delante. Empezando en el estado inicial se van hallando los controles óptimos que llevan de la etapa 0 a la etapa N. Por lo tanto, esta estrategia es la adecuada en el control del hospital, admitiendo que habrán trayectorias que no consigan llegar al estado final.

El procedimiento a seguir en el control local en cada servicio sería, por tanto, definir la función de costo y luego ir obteniendo la trayectoria óptima que nos lleve de la etapa 0 a la etapa N. La determinación de la función de costo crea un segundo problema porque el gestionador global, que va a diseñar la solución global del hospital, debe disponer de distintas trayectorias óptimas para diferentes funciones de coste, de forma que pueda componer una solución de compromiso entre todas las soluciones locales del hospital. Por lo tanto, para cada posible índice (tiempo mínimo, coste mínimo, variación de comandos mínima) habría que aplicar la programación dinámica y conseguir la trayectoria óptima. Esta tarea ocasiona un gasto computacional excesivo, que no se justifica por el hecho de que el gestor global a posteriori va a realizar una acción de compromiso que no tiene por qué ser la acción optima para cada servicio.

6.2.3.2.1.2.- Solución alternativa

Estas razones nos obligan a implementar otra estrategia en el control de cada servicio del hospital. Este mecanismo está basado en un procedimiento exhaustivo de obtención de trayectorias generadas mediante la aplicación de acciones de control dadas por una función lineal que tendrá en cuenta el error en el sistema, la suma de los errores anteriores y la variación del error por unidad de tiempo (en control estas acciones se denominan acciones PID, proporcional-integral-derivativa). Para el conjunto de trayectorias seleccionadas se calcularán diferentes tipos de índices. Esto es lo que se conoce en la terminología de optimización como procedimiento enumerativo.

Entonces, para cada servicio cuyo valores medios de cola estén fuera del margen permitido se generan un conjunto de trayectorias que nos lleven del estado inicial al estado deseado. Con cada una de las posibles soluciones se generarán una serie de índices (tiempo medio en el estacionario, tiempo de llegada al estado deseado, coste en horas y variaciones de los comando aplicados) que el gestor global, el encargado del diseño de una solución global para el hospital, estudiará para elegir la solución de cada servicio que sea compatible con el resto de soluciones locales existentes.

Las trayectorias generadas como soluciones locales presentan las dos características siguientes:

a) La variable etapa no puede ser el tiempo puesto que es uno de los factores que se quiere minimizar con el control. Por ello, las etapas vendrán determinadas por cambios en la variable de salida. Así, cuando en el sistema exista un error muy grande se eligirá como etapa los instantes en los que se ha producido un cambio grande (incremento/decremento en 30 días) en el valor de la variable de salida. A medida que el sistema se acerca al valor deseado, el control debe ser más fino, por lo que el cambio en la variable de salida que marca una nueva etapa será menor que en el caso anterior (incremento/decremento en 5 días).

b) La acción de control producida en cada etapa va a ser proporcional al error que presenta la variable de salida. Este tipo de acción hace que la salida del sistema se aproxime al valor deseado, existiendo la posibilidad de que halla un sobrepasamiento de la consigna produciéndose "un regalo de calidad", esto es, el valor de la cola está muy por debajo del valor deseado con lo cual existen un gasto de recursos innecesario. Para evitar este problema se añade además una acción derivativa. Una acción de control de este tipo no asegura llegar al valor de consigna, por lo que se añade una acción integral que modificará el comando a aplicar en función de la historia pasada del error. El controlador resultante es del tipo:

$$u(t) = k_p e(t) + k_d \frac{de(t)}{dt} + k_i \int_0^t e(t)dt$$

Las variaciones en los parámetros kp, kd y ki determinarán distintas estrategias. Aplicando este control a un servicio concreto se consigue los resultados que se muestran

en las siguientes gráficas, figuras 3.15, 3.16, 3.17 y 3.18. La línea roja representa el tiempo medio de espera en el servicio para ser atendido en primera consulta. La línea verde indica el tiempo medio de espera en el mismo servicio para ser atendido en consulta sucesiva. Los asteriscos rojos muestran el número de horas (comando) que se aplica en primera consulta para llevar el valor medio de esta cola al estado deseado. Los asteriscos verdes representan el número de horas (comando) que se necesitan en consulta sucesiva para llevar esta cola al valor adecuado. Para aplicar las acciones de control propuestas hay que realizar determinadas matizaciones en los comandos, esto es debido a que un hospital no es un sistema donde se pueda fácilmente cambiar los horarios. Por ejemplo, en el caso en el que las acciones de control a aplicar en instantes consecutivos de tiempo no se diferencien al menos en una sesión (cada sesión es de 7 horas) la nueva acción de control será igual al comando anteriormente aplicado.

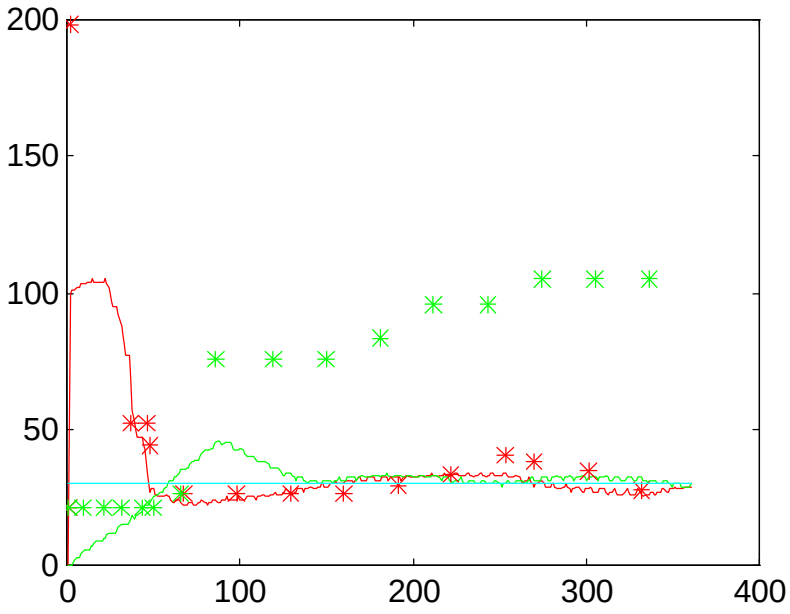


Figura 3.15.- Para controlar la cola de primera consulta (color verde) kp=1.4 y kd=1.0. Para consulta sucesiva (color rojo) kp=5.0 y kd=1.0

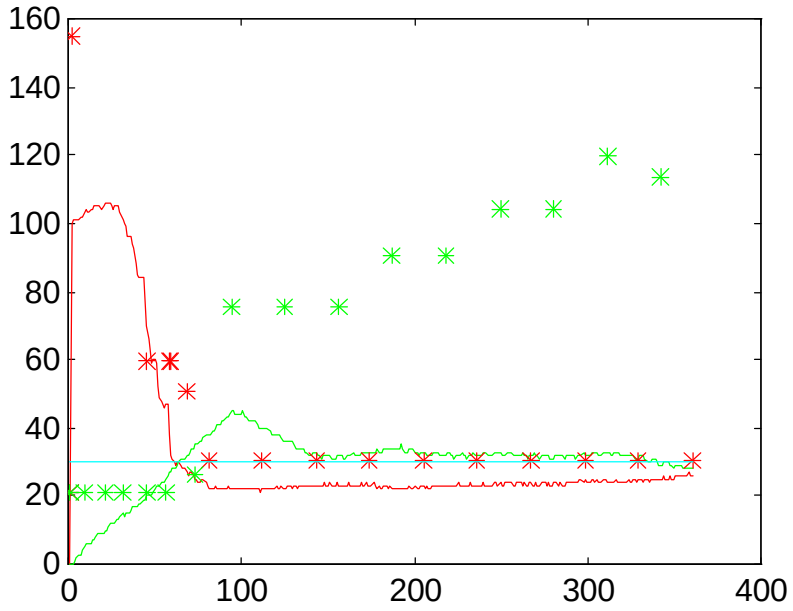


Figura 3.16.- Para controlar la cola de primera consulta (color verde) kp=1.5 y kd=0.5, se observa que se produce un regalo de calidad. Para consulta sucesiva (color rojo) kp=5.0 y kd=0.7

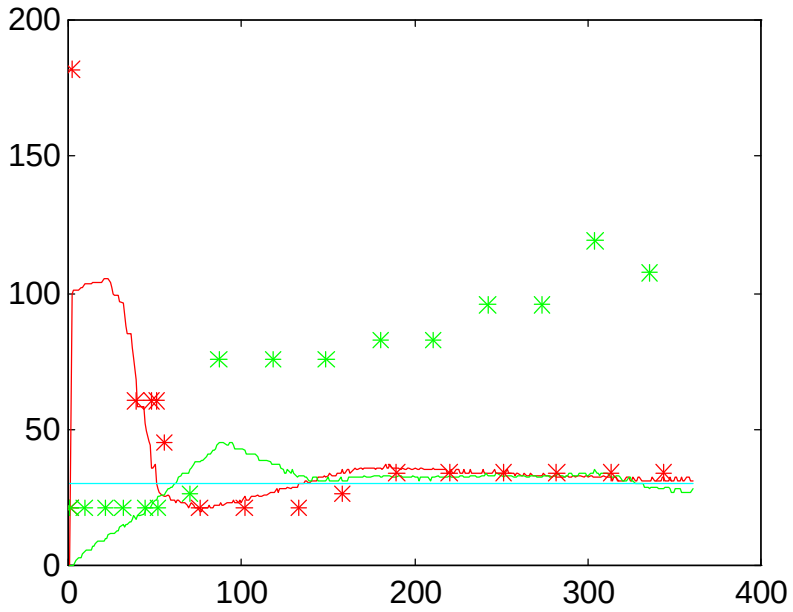


Figura 3.17.- Para controlar la cola de primera consulta (color verde) kp=1.6 y kd=0.7. Para consulta sucesiva (color rojo) kp=5.0 y kd=0.7

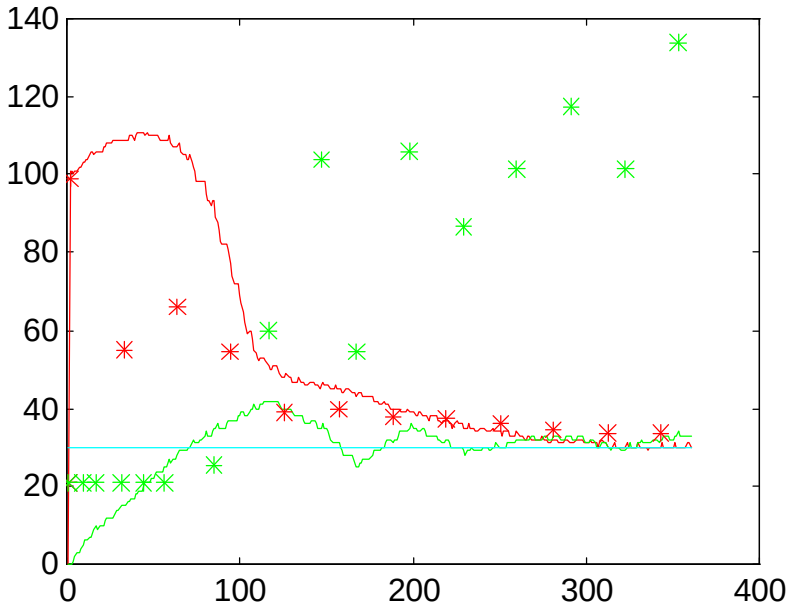


Figura 3.18.- Para controlar la cola de primera consulta (color verde) kp=0.7, kd=0.5, ki=0.005. Para consulta sucesiva (color rojo) kp=5.0, kd=0.05, ki=0.15

6.2.3.2.2.- Inferencia Simular

La implementación de esta inferencia consiste en un programa de simulación de eventos discretos que permite simular la dinámica del hospital a partir de un estado especificado. La ventaja que ofrece es la posibilidad de cambiar el horario de los recursos (acciones de control) y así poder estudiar los efectos que se producen sobre el estado de cada servicio. Más detalles del programa en el capítulo II.

6.2.3.2.3.- Inferencia Indice-Coste

Esta inferencia se implementa mediante un programa que informa sobre la bondad de las acciones de control generadas, para ello se determinan el valor de los índices que vienen marcados por el conjunto de normas establecidas en el plan estratégico que se desarrolla, es decir, se determina el coste de los recursos, tiempo necesario para alcanzar el estado final, dificultad en la obtención de nuevos recursos humanos, evitar excesivos cambios en el hospital, prioridades en las actividades, etc.

Como para cada uno de los servicios con problemas existen un conjunto de soluciones locales posibles, el gestionador de la solución global debe tener información

acerca de las ventajas e inconvenientes de cada una de ellas. Esta información es generada en la inferencia Indice-Coste cuya implementación consiste en determinar para cada solución local, conocido los resultados de la simulación de esa solución local, las siguientes variables:

- Tiempo en días que se tarda en llegar al valor deseado.

- Valor final que tendrá la cola.

- Coste en pesetas de la aplicación de dicha acción.

- Número de cambios que se deben realizar en el horario de los recursos.

Para la obtención del coste en pesetas de cada acción local, se parte del número de horas necesarias para llegar al estado final y se observa de qué tipo de servicio se trata, puesto que para cada uno de ellos se necesitarán un tipo de recurso u otro que tienen un coste diferente. Además hay que tener en cuenta que determinados recursos materiales (quirófanos) y recursos humanos (facultativos) tienen un coste asociado muy grande que impiden su obtención a corto plazo. El estudio del gasto necesario en recursos humanos se realiza a partir de un modelo estático que representa los recursos humanos disponibles en un periodo de tiempo determinado. La estructuración del personal del Hospital Nª Sra de la Candelaria es la que muestra la figura 3.19.

El gerente del hospital está en la cima de la pirámide formada por los recursos humanos del hospital. Sus decisiones se apoyan en el trabajo realizado por el Director de Gestión, el Director de Recursos Humanos y el Director de Enfermería.

El Director de Gestión, como su propio nombre indica, realiza el trabajo de coordinación de las tareas económicas y financieras, no relacionadas directamente con la atención al paciente pero sin las cuales el funcionamiento del hospital sería imposible. Esta dirección se divide en tres subdirecciones:

a) Subdirección de Gestión Económica: realiza las tareas de gestión económicas y financieras, se encarga de los suministros, la facturación y la farmacia.

b) Subdirección de Personal: se encarga de las gestiones administrativas relacionadas con la contratación, las nóminas, etc.

c) Subdirección de Asuntos Generales: lleva a cabo los trabajos administrativos

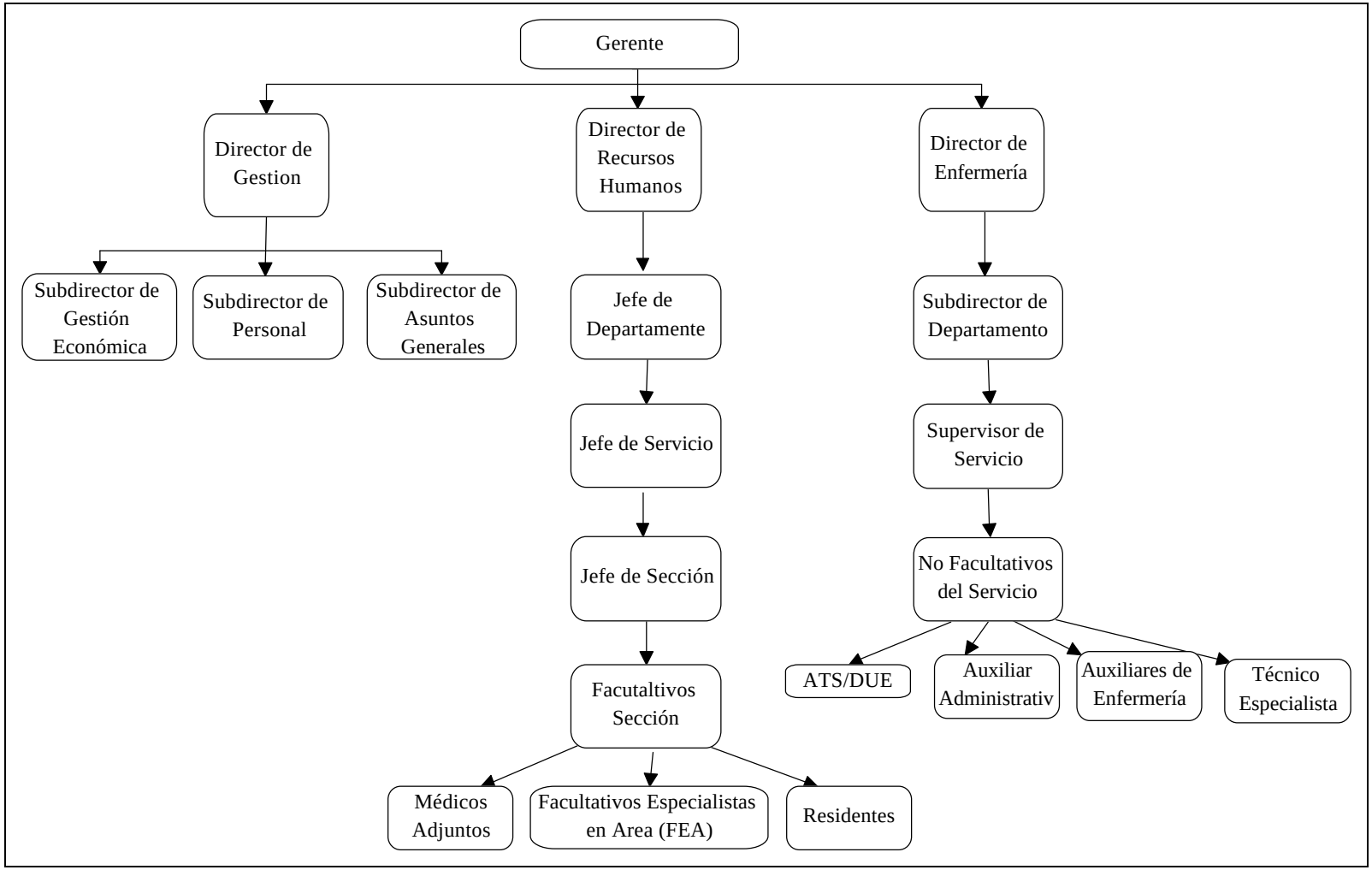


Figura 3.19.- Estructura de los Recursos Humanos del Hospital Nª Sra. de la Candelaria

relacionados con el archivo, la biblioteca, la cocina, etc.

El Director de Recursos Humanos se encarga de la coordinación de los facultativos. Para ello se nombran a los Jefes de Departamentos (planta, consulta) y para cada servicio un Jefe de Servicio. Como los servicios se dividen en secciones o especialidades el Jefe de Sección es el responsable de los facultativos de cada especialidad. Los facultativos son los médicos adjuntos, FEA (facultativo especialista de área) y los residentes.

El Director de Enfermería se encarga de la coordinación del personal no facultativo de los servicios: ATS/DUE, auxiliar de enfermería, técnico especialista y auxiliar administrativo. Para ello dispone de un Subdirector por Departamento y de Supervisores de Servicios en cada servicio del hospital.

Las variaciones dentro de este modelo (incremento, decremento o redistribución) son producidas para mejorar el rendimiento del hospital. La importancia de estos recursos es debida a que en un hospital se realiza trabajo muy especializado y eso dificulta la redistribución del personal. Además, este tipo de recurso es escaso en el mercado laboral y cuando existe suponen un fuerte gasto económico.

La gestión de los recursos humanos implica observar los existente y reubicar aquellos que estén ociosos, siempre que existan labores que puedan realizar. En el caso de que se necesiten más horas de recursos humanos se estudiará la utilización de horas extras con el incremento de precio que esto produce. En determinados casos es preferible la contratación de personal nuevo puesto que sería más económico que las horas extras, a excepción de los médicos puesto que debido a su especialización es muy costosa su contratación y por lo tanto no es una decisión que se pueda tomar a corto plazo. Para poder almacenar, leer y modificar los datos sobre los recursos humanos de cada servicio utilizamos una base de datos que nos permite tener toda esta información de una manera ordenada. Un ejemplo de los datos almacenados para cada servicio se muestra en la figura 3.20.

SERVICIO/SECCIÓN: TOCOGINECOLOGÍA CÓDIGO: OBGI/GIN

RECURSOS BÁSICOS DEL SERVICIO

Recursos Humanos (Personal Sanitario Adscrito)

Facultativos	Efectivos	No Facultativos	Efectivos	% jornada	Comparte con:
Jefe Departamento		Supervisión	5		
Jefe Servicio	1	A.T.S./D.U.E.	29		
Jefe Sección	3	Auxiliar de Enfermería	39		
Adjunto	9	Técnico Especialista			
F.E.A.	18	Auxiliar Administrativo	1		
TOTAL	31	TOTAL	73		
Residentes					
1er año					
2º año					
3 er año					
4º año					
5º año					

Recursos materiales

RECURSOS	NÚMERO	OBSERVACIONES
CAMAS ASIGNADAS	72	
PROMEDIO DE UTILIZADAS	44,7	
CONSULTA (horas/semana) Hospital Área	 175 horas 273 horas	
SESIONES QUIRÚRGICAS (Programa Anestesia General)	7 sesiones/semana	

Figura 3.20.- Tabla que refleja el subsistema 2 con los recursos de los que dispone cada servicio del hospital.

Un ejemplo de la tabla generada por la inferencia Indice-Coste para cada servicio con problemas se muestra en la figura 3.21. En la primera columna aparece el identificador de cada acción de control posible que se caracteriza por tener unos valores específicos en los parámetros del controlador. En la segunda columna se muestra el tiempo en días que se tarda en llevar la cola al valor deseado. En tercer lugar nos encontramos con el valor que va tener la cola en el estacionario, con este dato se determina si la acción de control en estudio produce un regalo de calidad y lleva el valor

de la cola por debajo del valor de consigna. La cuarta columna indica el número de horas totales necesarias para conseguir el estacionario, esta cantidad se eleva al cubo para amplificar las tendencias y poder clasificar mejor las distintas opciones. Por último se presenta un valor que es la relación entre la media aritmética y la media geométrica de los comandos aplicados. Esta magnitud es muy utilizada en procesamiento de señales y nos refleja muy bien las variaciones grandes que queremos evitar en el comando, ya que ofrece una medida relativa de la dispersión de los comandos que se aplican. Para valores próximos a 1 esta dispersión es mínima, condición exigida para el control de los hospital ya que no es conveniente cambiar con frecuencia el horario de los recursos y en especial el de los recursos humanos. A medida que el este índice se aleja del 1 indica una mayor dispersión en el valor de los comandos aplicados.

Conociendo la tablas de todos los servicios con problemas el gestionador global debe elegir aquel conjunto de soluciones locales que, aun no siendo las mejores localmente, son compatibles con las soluciones locales de otros servicios.

Fichero	Días para estacionario	Media estacionario	$(\text{Horas})^3$	Media aritmética/ media geométrica
grad2.mat	256	28.2952	2.691.084.790	1.4132
	103	30.1163	163.206.988	1.1113
grad3.mat	185	33.5625	2.471.102.564	1.4761
	103	30.7326	234.222.628	1.1090
grad6.mat	191	33.0941	2.426.734.907	1.3707
	124	31.9241	180.679.640	1.1335
esqpd1.mat	295	24.8485	2.377.714.062	1.2779
	130	32.0606	180.500.770	1.1590
grad4.mat	92	29.3866	2.629.304.440	1.3838
	123	31.4622	201.042.084	1.1395
esqpi2.mat	342	33.5263	609.475.245	1.0444
	195	32.4277	99.666.119	1.2120
grapi7.mat	284	30.0390	570.357.277	1.0498
	187	32.1494	195.565.740	1.1829
grapi3.mat	229	31.7273	409.470.886	1.1103
	145	31.2361	47.755.652	1.1090
grapi12.mat	173	29.8936	655.290.630	1.1400
	195	31.8614	402.377.898	1.1167
esqpi3.mat	109	30.7460	572.009.720	1.1573
	180	31.7238	324.880.046	1.1101

Figura 3.21.- Tabla generada por la inferencia Indice-Coste para cada problema local.

6.2.4.- Tarea de Diseño

Esta tarea se encarga de encontrar la solución adecuada planteada al hospital en su globalidad.

Como en las tareas anteriores, se divide todo el sistema en subcomponentes. Sin embargo, en este caso, la solución global no puede ser directamente la suma de las soluciones locales, sino que debe ser sintetizada considerando sus interacciones.

El método de resolución de problemas utilizado en esta tarea de diseño global es el propone y revisa. Primero genera una combinación de soluciones locales en el paso de proponer, y luego la combinación es analizada para verificar si satisface las restricciones exigidas. Si se detecta una situación incompatible en el paso de revisar, se modifica la propuesta global usando conocimiento acerca de prioridades entre componentes, repitiendo el proceso hasta que se obtiene un conjunto mínimo de soluciones compatibles.

La tarea de diseño tiene como datos de entrada el sistema en estudio, los componentes del mismo, así como las soluciones locales posibles que han sido generadas en la fase de predicción. Como salida de la tarea de diseño tenemos el conjunto mínimo de soluciones para resolver el problema global del sistema.

La descripción de la tarea de diseño es la siguiente:

```
diseño (modelo_sistema, componentes, soluciones_locales_posibles) =
    propone ( soluciones_locales_posibles  ⇒ solución_global)
    repetir
            verifica (solución_global  ⇒ solución_válida)
            Si solución_válida = ∅
                    remediar (solución_global)
    hasta solución_válida  ≠  ∅
```

6.2.4.1.- Inferencias realizadas en el Diseño

La figura 3.22 muestra la estructura de inferencia de la tarea de diseño:

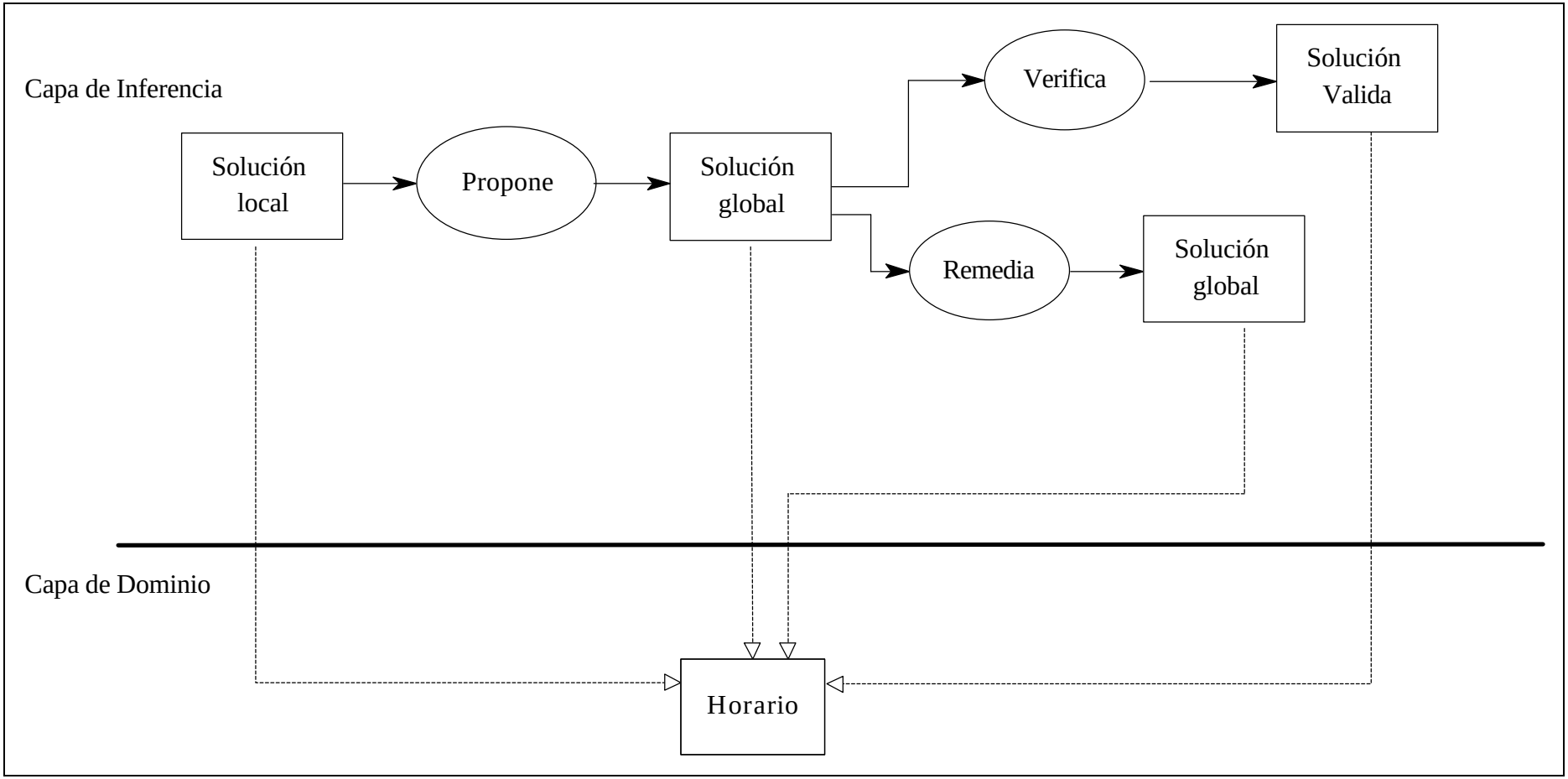


Figura 3.22.- Estructura de inferencia de la tarea de diseño.

La definición de cada inferencia en término de las piezas de conocimiento de dominio que utilizan es la siguiente:

Inferencia 1 propone (soluciones_locales_posibles Þ solución_global)

"Se debe estudiar la solución global a todos los problemas que plantean los distintos componentes del hospital porque debido a las interrelaciones existentes puede haber conflictos y la solución local para el problema de un servicio puede ocasionar problemas en otros. Por ello se crea una solución global que inicialmente está formada por la unión de las soluciones parciales a cada problema"

define-inference PROPONE

"La condición se activa para todas las ternas servicio-actividad-problema generadas en la inferencia precisar-servicio-problema"

"En el cuerpo de la inferencia se buscan todas las piezas de conocimiento recursos, horarios que pertenezcan a la cuaterna servicio-actividad-recurso-horario generada en la inferencia selecciona-soluciones. Esta cuaterna se añade a la solución global y se elimina de la base de conocimiento."

"El resultado es la solución global generada"

:cond (forall ?serv ?act ?problem (Precisar-servicio-problema ?serv ?act ?problem))

:body (exist ?recurso ?horario

(and

(Selecciona-soluciones ?serv ?act ?recurso ?horario)

(⇒ (+ (?solución-global ?serv ?act ?recurso ?horario))

(refract (Selecciona-soluciones ?serv ?act ?recurso ?horario))))

:result (Propone ?solucion-global)

Inferencia 2 verifica (solución_global Þ solución_válida)

"Se debe verificar que la solución global propuesta verifica los requisitos establecidos, es decir, elimina o reduce los problemas existentes. Para ello se simula todo el hospital bajo esa solución global y se determina si existe algún servicio que sigue teniendo problemas"

define-inference VERIFICA

"La condición se activa para cada pieza de conocimiento que sea una solución global generada por la inferencia propone o por la inferencia remedia"

"En el cuerpo de la inferencia se busca una pieza de conocimiento tal que sea un hospital, se realiza la simulación del hospital, después se seleccionan todos los componentes del hospital mediante la inferencia selecciona-componente, se seleccionan las variables de esos componentes mediante la inferencia selecciona-variable. Se compara el tiempo normal y observado de cada variable y se clasifica mediante la inferencia clasificar-erroneo."

"El resultado es la terna servicio-actividad-diferencia tal que haya generado la inferencia clasificar-erróneo"

```
:cond (exist ?solucion-global
        (or
            (Propone ?solucion-global)
            (Remedia ?solucion-global)))
:body (exist ?h
        (and
            (Hospital ?h)
            (⇒ (SIMULAR ?h)
                (Selecciona-componente ?servicio)
                (Selecciona-variable (?servicio ?act))
                (Valor-normal ?servicio ?t-normal)
                (Valor-observado ?servicio ?act ?t-observado)
                (Comparar ?servicio ?act ?diferencia)
                (Clasificar-erroneo ?servicio ?act ?diferencia))))
:result (Verifica ?servicio ?act ?diferencia)
```

Inferencia 3 remediar (solución_global)

"En el caso de que la solución global inicialmente propuesta origine problemas en algún servicio se corrige esta solución global añadiendo otra de las soluciones parciales al servicio con problemas"

define-inference REMEDIA

"La condición se activa para todas las ternas servicio-actividad-diferencia tal que sean generadas por la inferencia verifica"

"En el cuerpo de la inferencia se obtiene las piezas de conocimiento recurso, horario tal que formen parte de la cuaterna servicio-actividad-recurso-horario generado por la inferencia selecciona-soluciones. Esta cuaterna es añadida a la solución global y eliminada de la base de conocimiento"

"El resultado es la solución global generada"

:cond (forall ?servicio ?act ?diferencia (Verifica ?servicio ?act ?diferencia))

:body (exist ?recurso ?horario

(and

(Selecciona-soluciones ?servicio ?act ?recurso ?horario)

(⇒ (+ (?solución-global ?serv ?act ?recurso ?horario))

(refract (Selecciona-soluciones ?serv ?act ?recurso ?horario))))

:result (REMEDIA ?solucion-global)

6.2.4.2.- Piezas de conocimiento complejas utilizadas en el diseño

En esta tarea se debe proponer una solución global, para ello se realiza una tarea de diseño. Formalmente el problema de diseño consiste en la construcción a partir de unas primitivas de la solución a un problema, exigiendo que esta cumpla unos requisitos [Piñeiro 1998]. Para resolver un problema de diseño tenemos que realizar los siguientes pasos:

a) Inicialmente se debe hacer una tarea de síntesis donde se componga una posible solución. El método de proponer dependerá de las primitivas del dominio y sus relaciones. Para el hospital las primitivas van a ser las soluciones locales y las relaciones vienen marcadas por las especificaciones (tiempo mínimo en servicios prioritarios y coste inferior a uno dado).

b) Una vez propuesta una solución se debe realizar una tarea de análisis dónde se determina si esa solución cumple con las especificaciones y satisface los requisitos impuestos. Este proceso se realiza mediante la simulación del hospital.

c) Si la solución no cumple los requisitos necesarios debemos corregirla en la dirección indicada por el método de análisis.

La complejidad en el problema de diseño es que consiste en una búsqueda de soluciones en un espacio de estados [Chandrasekaran 1990]. Para reducir este espacio de estados tenemos que recurrir al conocimiento sobre el dominio.

6.2.4.2.1.- Problema de Búsqueda

El problema de cómo lograr un objetivo es equivalente a determinar qué acciones y estados hay que considerar para llegar a la meta. Cuando existe un conjunto de acciones o estados por los que pasar y se quieren elegir aquellos que son mejores entonces tenemos un problema de búsqueda [Rich 1991] [Russell 1995] en un espacio de soluciones. Para la resolución de este tipo de problema es útil considerar un árbol de búsqueda donde cada nodo representa los estados del sistema y las ramas las acciones para cambiar de estado. El nodo raíz corresponde al estado inicial, y los nodos hojas corresponden a estados que no tienen sucesores en el árbol (un nodo es el antecesor de otro, sucesor, si existe una cadena de una o más ramas del antecesor al sucesor). La

solución al problema se plantea entonces como la búsqueda de una trayectoria que comience en el nodo raíz y termina en un nodo objetivo. Los procedimientos de búsqueda no tienen conocimiento acerca del tamaño o forma final que habrá de tener el árbol de búsqueda completo, todo lo que saben es dónde empezar y cual es la meta. Su trabajo consiste en ir expandiendo los nodos (determinar los hijos de los nodos) comenzando con el nodo raíz, hasta que se logre descubrir un nodo que corresponda con una trayectoria aceptable.

Los árboles de búsqueda se expanden de manera exponencial, esto es, el número total de trayectorias de un árbol con factor de ramificación *b* (el número de hijos de cada nodo que no sea hoja es *b*) y profundidad *d* es b^d. Por lo tanto, se dice que el número de trayectorias se expande exponencialmente a medida que aumenta la profundidad del árbol de búsqueda. En consecuencia se debe intentar desplegar un método de búsqueda que tenga probabilidad de desarrollar el menor número de trayectorias. Ejemplo de estos algoritmos serían la búsqueda en profundidad (se toma uno de los hijos en cada nodo que se visita y se avanza a partir de ese hijo, ignorando otras alternativas del mismo nivel en tanto haya posibilidades de alcanzar la meta mediante la selección original) o búsqueda en amplitud (que revisa todas las trayectorias de una longitud dada antes de avanzar a un trayectoria más larga) [Rich 1991] [Russell 1995] [Winston 1992].

Las estrategias de búsqueda anteriormente citadas encuentran las soluciones a un problema de forma sistemática generando nuevos estados y testeando si son el objetivo. Pero estas estrategias son muy ineficaces en la mayoría de los casos. Para mejorarlas se pueden utilizar estrategias de búsqueda informada que hacen uso del conocimiento sobre el problema concreto para encontrar las soluciones más óptimas de la manera más eficaz.

En un problema de búsqueda el conocimiento acerca del problema específico que se está resolviendo sólo se puede aplicar para determinar el siguiente nodo a expandir. Generalmente, el conocimiento que decide esto, es obtenido de una función de evaluación que mide la bondad de expandir un nodo concreto.

Un algoritmo con estas características es el Primero-el-mejor que expande aquel nodo que tiene una mejor evaluación. Las funciones que se utilizan en la bibliografía son

varias, generalmente cuando se buscan soluciones de coste bajo se suelen utilizar medidas estimadas del coste de la solución e intentan minimizarlo. Por ejemplo, el algoritmo de búsqueda A* intenta obtener la solución más barata conocido el coste hasta el estado actual y una estimación del coste hasta el objetivo. Si esta función heurística es monótona el algoritmo A* es optimo y eficiente, no hay otro que expanda menos nodos que A*.

6.2.4.2.2.-Búsqueda para la obtención de la Solución Global

El problema que se plantea es el siguiente: dadas las soluciones locales a distintos problemas, las cuales tienen un coste asociado, se debe obtener la solución global como combinación de las soluciones locales. Esta solución global debe cumplir los siguientes requisitos: ser la de menor coste y la más óptima (que los problemas locales se resuelvan en el menor tiempo posible sin generar nuevos problemas). Para determinar la solución global tenemos que resolver, por lo tanto, un problema de diseño en el que se propone una solución, se verifica si es adecuada y se remedia en el caso de que no sea la solución buscada. Este problema se puede resolver como un problema de búsqueda informada donde el conocimiento acerca del sistema nos lo proporciona la simulación del mismo. El procedimiento de búsqueda utilizado es: para expandir un nodo a partir del nodo raíz seguir la estrategia "Primero-el-mejor"; a partir de ese momento se realiza una búsqueda en profundidad intentando remediar la solución inicialmente propuesta para que cumpla los requisitos exigidos.

Un ejemplo de cómo evolucionaría el árbol de búsqueda se muestra en la figura 3.23.

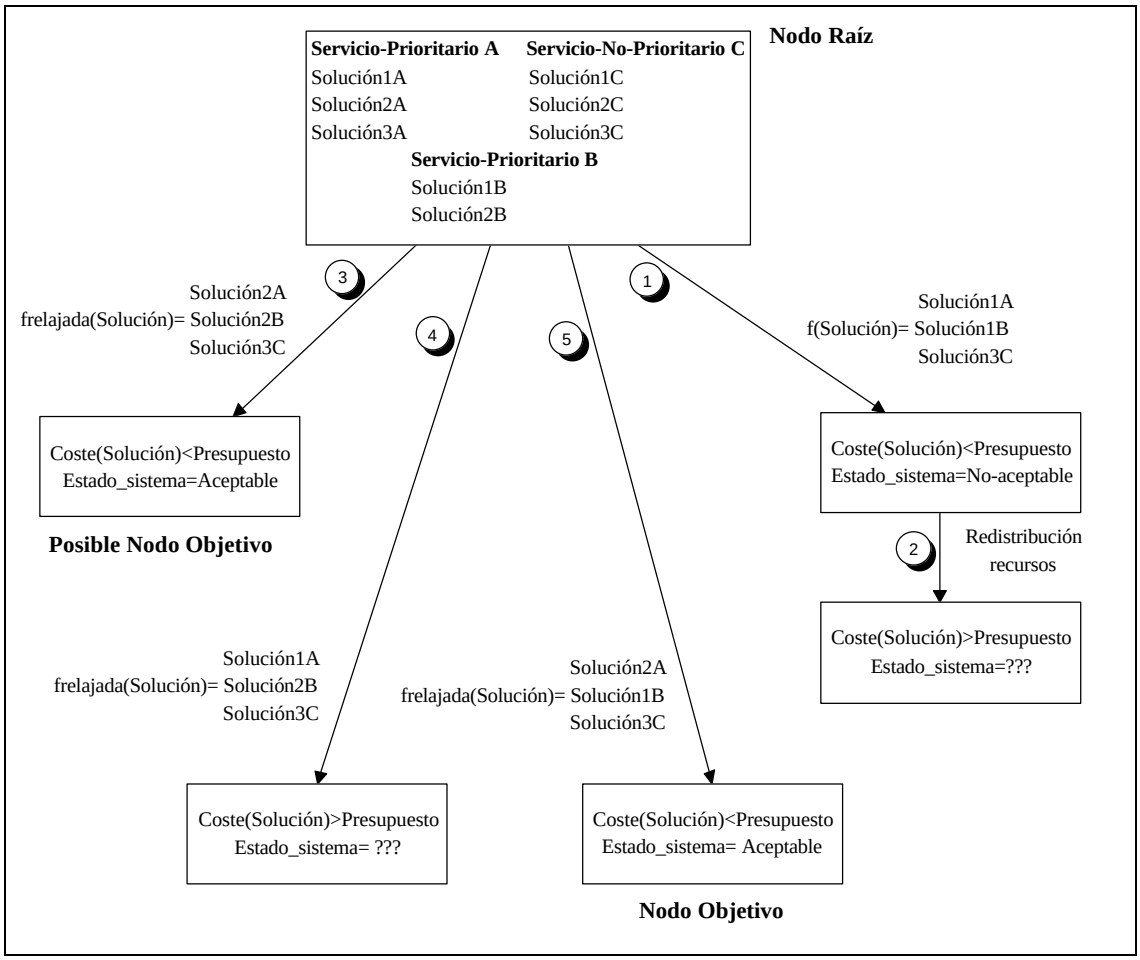


Figura 3.23.- Estados en el árbol de búsqueda de la solución global. La estrategia seguida es expandir primero el mejor y luego hacer una búsqueda en profundidad intentando mejorar la solución inicial.

Partimos por lo tanto de un nodo raíz en el que tenemos un conjunto de soluciones locales que resuelven los problemas de los servicios en el menor tiempo posible pero con un costo grande o en más tiempo con un costo inferior. La expansión del nodo raíz viene como consecuencia de todas las posibles combinaciones de las soluciones locales. Para resolver este problema de búsqueda vamos a utilizar el conocimiento que tenemos acerca del problema y se expandirá el nodo que se considere el mejor según la siguiente función heurística (que ha sido extraída y validada a partir de entrevistas con diferentes expertos en gestión hospitalaria):

$$F = \begin{cases} \text{Solución local de tiempo mínimo} & \text{Si servicio es prioritario} \\ \text{Solución local de coste mínimo} & \text{Si servicio es no prioritario} \end{cases}$$

Esta función heurística indica que el gerente del hospital tiene clasificados los servicios en dos tipos:

- servicios prioritarios: donde es muy importante estabilizar las colas en el menor tiempo posible debido a la gravedad de la patología que tratan o por razones relacionadas con el entorno socio-político.

- servicios no prioritarios: aquello que por la no gravedad de sus patologías no tienen tanta importancia en el momento de repartir recursos.

La mejor elección para el gerente sería la resolución de los problemas de los servicios prioritarios en el menor tiempo posible aunque ello exija un gran gasto; y resolver los problemas de los servicios no prioritarios con el menor costo posible aun en el caso de que hubiera presupuesto suficiente puesto que en un hospital público todos los recursos son escasos y no se puede permitir regalos de calidad.

A partir de esta función se expande el nodo más adecuado. En el ejemplo tendríamos la rama 1, en el que la solución global sería la de menor tiempo para los servicios prioritarios (solución 1A para el A y solución 1B para el B); y la de menor coste para los no prioritarios (solución 3C para el C).

Para determinar si hemos llegado al objetivo se tienen que estudiar dos restricciones: a) que el coste de la transición realizada sea inferior al gasto previsto y b) que el estado alcanzado sea un estado deseable (no se generen problemas nuevos).

La función de coste se define como indica el algoritmo de la figura 3.24. Inicialmente se utilizan todos los recursos ociosos con coste 0 (que generalmente no deben haber muchos), a continuación se utilizan horas extras para abrir nuevas sesiones con coste E si no se llegan a cubrir todas las necesidades se redistribuyen los recursos

```
Dado:
        H: horas necesarias para realizar una acción
        Ri: recursos necesarios para realizar la acción (i:=1 .. j)
        Coste: pesetas necesarias para realizar la acción
Procedure Determina_Coste(IN: H, Ri; OUT: Coste)
var
        N: número de sesiones para cubrir las H horas;
        Ni: número de sesiones del recurso i;
        Ci: coste de contratación de una sesión del recursos i;
begin
        N := H / (horas por sesión);
        Coste := 0;
/* la primera tarea es asignar todos los recursos ociosos */
        For i:=1 to j
                While (Ni > 0) and (Ri tiene sesión ociosa)
                        Ni:= Ni -1;
                End While;
        End For;
/* cuando no quedan recursos ociosos se intentan aumentar las sesiones con horas
extras con coste asociado Ei*/
        For i:=1 to j
                While (Ni > 0) and (Ri puede realizar horas extras)
                        Ni:= Ni -1;
                        Coste:= Coste + Ei
                End While;
        End For;

/* cuando no quedan recursos ociosos se obtienen de otros servicios que no tengan
tanta prioridad con coste asociado Di*/
        For i:=1 to j
                While (Ni > 0) and (Ri tiene sesión disponible en servicio menos
prioritario)
                        Ni:= Ni -1;
                        Coste:= Coste + Di
                End While;
        End For;
/* finalmente se contratan las sesiones que falten por cubrir con coste asociado Ci,
excepto los médicos cuyo coste es muy caro y por ello se utilizan aquellos que
perteneciendo al mismo servicio atienden especialidades diferentes */
        For i:=1 to j
                If Ri<> "médico"
                        Ci:= Ni * Ci;
                        Coste:= Coste + Ci;
                Else
                        While (Ni > 0) and (Ri tiene sesión en especialidad menos
prioritario)
                        Ni:= Ni -1;
                        Coste:= Coste + Di
                End While;
        End For;
End Procedure;
```

que estén disponibles en otros servicios menos prioritarios con coste D, y finalmente se contratan nuevos recursos con coste C , con la excepción de los recursos de tipo facultativo que no se pueden contratar debido al elevado coste que supone (es una decisión a largo plazo) por lo que la solución pasa por redistribuir a los médicos que atienden en otras especialidades del mismo servicio, aunque se le ocasionen problemas a estas especialidades.

Por lo tanto para saber si hemos alcanzado el nodo objetivo se debe hallar el costo de la solución global para comprobar que es inferior al presupuesto disponible. Si esto es así podríamos haber encontrado la solución, para confirmarlo se simula el sistema considerando esta solución global. Si el resultado es que todos los servicios tienen valores de cola dentro de los márgenes permitidos, entonces hemos encontrado la solución global. Si esto no ocurre y algún servicio se ve fuertemente afectado por esta solución global debemos encontrar otra solución mejor. Para ello realizamos una búsqueda en profundidad intentando mejorar la solución de partida. Esta mejora se realiza con la información que aporta la simulación, esto es, se lleva a cabo para la misma solución global otra redistribución de recursos que consiste en conseguir una redistribución de recursos tal que permita realizar las acciones de control propuestas pero sin modificar el personal de aquellos servicios que se habían visto alterados (en el ejemplo la rama 2).

Se continúa esta búsqueda en profundidad hasta alcanzar el objetivo o hasta que el coste de la solución global propuesta sea superior al presupuesto disponible. Si esto último ocurre, debido a que el coste en las sucesivas redistribuciones es monótonamente creciente, ya no se puede seguir mejorando la solución de partida y hay que expandir una nueva rama a partir del nodo raíz (en el ejemplo la rama 3). Para ello se utiliza de nuevo el mecanismo "Primero-el-mejor" pero realizando una relajación del problema [Russell 1995], esto es, se compone una solución global que tenga restricciones más débiles. La función heurística a utilizar sería la siguiente (en el ejemplo la rama 3):

$$F_{relajada} = \begin{cases} \text{Solución local de tiempo mínimo relajada en una cantidad fija respecto a la solución anterior} & \text{Si servicio es prioritario} \\ \text{Solución local de coste mínimo} & \text{Si servicio es no prioritario} \end{cases}$$

El mecanismo utilizado para realizar la búsqueda de manera más eficiente es relajar las restricciones en la misma cantidad en todos los servicios prioritarios, esto es debido a que el experto lo que quiere obtener es que todos los servicios prioritorios consigan los objetivos en el menor tiempo posible (independientemente del gasto asociado). Cuando se produce una relajación del problema, debido a que el coste es superior al presupuesto disponible, se sigue el criterio de buscar soluciones locales cuyo incremento en el tiempo de establecimiento sea el mismo sin considerar por lo tanto que ocasionará mayor gasto en unos servicios que en otros. Por lo tanto, siempre que se quiera obtener una posible solución partiendo del nodo raíz, se debe continuar relajando las restricciones en todos los servicios prioritarios hasta que se llegue a un estado aceptable.

Si se ha obtenido un nodo objetivo que cumple con los requisitos propuestos entonces tenemos una posible solución. Una tarea a realizar ahora es estudiar las combinaciones que hemos obviado al relajar por igual todos los servicios prioritarios, esto es, tenemos acotada la mejor solución pero podemos buscar algunas trayectorias intermedias con las que obtener mejores resultados. Para ello se realiza la búsqueda a partir de relajar únicamente algunos servicios prioritarios de manera que se obtenga la solución más adecuada con las restricciones impuestas (rama 4 y 5 del ejemplo).

Cuando la búsqueda no te ofrece una solución quiere decir que con el presupuesto asignado no se pueden resolver los problemas existentes. En el caso de obtener solución, el gerente tiene la posibilidad de realizar distintas búsquedas considerando otras prioridades en los servicios de forma que manteniendo el costo pueda decidir favorecer a unos servicios con respecto a otros. Otra opción que se le presenta al gerente es bajar el presupuesto para intentar otra solución con menor costo. Por lo tanto el problema se plantea como n-búsquedas informadas en un espacio de

soluciones con el que se obtienen mejores resultados que con el caso de búsqueda exhaustiva debido al gasto computacional que supone tanto una búsqueda de este tipo en un árbol así como la comprobación de haber llegado al nodo objetivo puesto que hay que ejecutar el programa de simulación.

7.- DESARROLLO DE UN PROTOTIPO

El paso de implementación del SBC se realiza sobre un hospital "toys", esto es, un hospital que mantenga todas las características importantes de cualquier hospital general pero donde la información generada sea mucho menor y por lo tanto sea un sistema más fácilmente manejable. La idea es reducir la complejidad a base de disminuir el número de elementos del sistema pero no las interrelaciones que estos tienen, y disponer de una implementación que nos permita presentar la utilidad de la aplicación.

Los pasos a realizar en la construcción del SBC "toys" son los siguientes:

A) Definir la estructura del hospital prototipo, en una base de datos, que tiene los siguientes servicio:

Servicio A de tipo no central con tres especialidades.

Servicio B de tipo no central con dos especialidades.

Servicio C de tipo central que realiza una prueba clínica necesaria en las patologías de el servicio A y el servicio B.

La construcción de un hospital "toys" presenta una doble utilidad: es el sistema en estudio y permite predecir los efectos de las distintas acciones de control. El estudio no se realiza sobre un sistema real porque podría afectar el funcionamiento del mismo asi como perder generalidad en el desarrollo. Por estas razones se necesita del programa de simulación que nos ofrezca el comportamiento del hospital definido anteriormente. Asimismo la predicción sobre el estado futuro del hospital "toys" como consecuencia de la aplicación de las distintas acciones de control también será proporcionado por dicho programa que simulará a partir del estado actual del hospital la dinámica del mismo.

B) Para la implementación de la estructura del SBC se requerirá:

* Seleccionar el software de implementación de forma que cumpla las especificaciones del problema, esto es, que tenga un lenguaje de emparejamiento de patrones, corra bajo Windows y fácil comunicación con otras aplicaciones.

* Implementar las estructuras de conocimiento diseñada en los apartados anteriores.

En cuanto al software de desarrollo, ART-IM y CLIPS son dos herramientas de desarrollo de SBC cuyos fundamentos son prácticamente los mismos. Ambas utilizan el mismo algoritmo de razonamiento hacia delante: RETE. Fue la NASA la que implementó la primera versión de este algoritmo generando la primera versión de CLIPS (C Language Integrated Production System). Dada la eficiencia del mismo y el éxito que tuvo esta herramienta, a la primera versión le han seguido muchas más. A mediados de los ochenta Inference Corporation se aprovecha de que la NASA no posee un copyright exclusivo de la herramienta y desarrolla su propia versión del RETE tomando como referencia el código fuente de CLIPS. ART (Automated Reasoning Tool) se ha convertido en una de las herramientas que más aceptación ha tenido en el mercado, proporcionando un conjunto de paquetes de programación de propósito especial que facilitan el desarrollo y la integración del sistema con otros lenguajes de alto nivel y paquetes como Windows, bases de datos, hojas de cálculo, etc...

ART-IM permite representar el mundo real vía objetos, atributos, relaciones y hechos; se pueden crear dinámicamente y modificar en tiempo de ejecución, sin que se pierda la consistencia del sistema de objetos. Además, posibilita ejecutar programas procedurales. Para clasificar los objetos en jerarquía proporciona mecanismos de herencia.

Las reglas nos permiten expresar como determinadas relaciones entre objetos nos conducen a modificar objetos o a realizar acciones. Independientemente del número de reglas y datos de la base de conocimiento, el sistema mantiene aproximadamente constante el número de reglas que procesa por segundo. Esto es posible debido a que utiliza una versión del algoritmo RETE que realiza una compilación de la base de conocimiento, de forma que permite un acceso rápido a los datos.

La facilidad de acceder a bases de datos es otra de las características en el ART-IM. Así como permitir una fácil integración de la base de conocimiento en una aplicación ya existente.

Y como una de las características más importantes y que necesitamos en nuestro problema es que posea emparejamiento de patrones (*pattern-matching*), esto es, permita trabajar con cada uno de los objetos que cumplen determinadas condiciones.

Como el ART-IM cumple todas las necesidades que pide el problema y dada la experiencia por su uso en la construcción de un "Sistema Experto para la Extracción Automática de las Características de un Potencial Evocado Visual" [Aguilar 1994], es una herramienta adecuada en la implementación del SBC para la ayuda a la toma de decisiones en la gerencia hospitalaria.

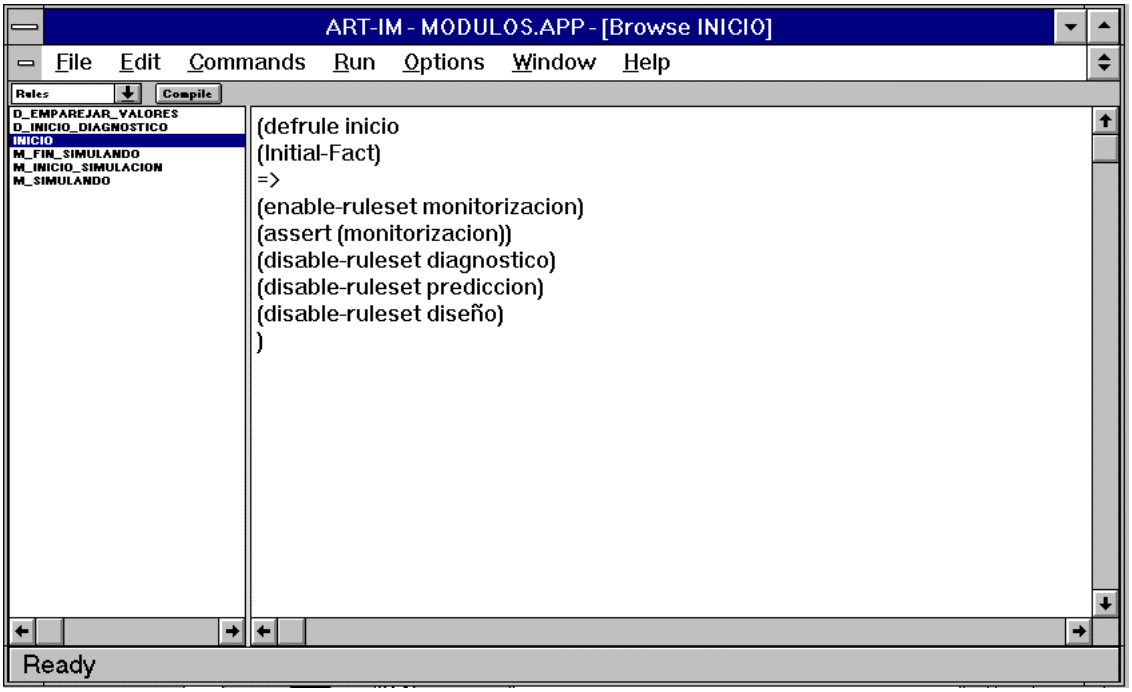


Figura 3.25.- Entorno de edición de reglas en ART-IM

La construcción de las estructuras de conocimiento se divide en cuatro módulos que corresponden con las cuatro tareas definidas en el diseño del SBC, monitorización, diagnóstico, predicción y diseño. Para ello se construyen islas de conocimiento que estarán formadas por el conjunto de reglas que realizan cada tarea, la principal característica de utilizar conjuntos de reglas es que se pueden habilitar o deshabilitar

según las necesidades de la aplicación. Con lo cual, cuando se ejecuta la tarea de monitorización los conjuntos de reglas pertenecientes a las tareas de diagnóstico, predicción y diseño, estarán deshabilitadas y así la implementación sigue la misma estructuración que el diseño de la base de conocimiento. Cada módulo realizará las inferencias asociadas con la tarea que resuelven:

a) Monitorización.- Al no trabajarse con el sistema real, la monitorización debe realizarse sobre el hospital "toys" por lo que se necesita ejecutar el programa de simulación que nos proporcionará la dinámica del mismo. Para monitorizar sobre la simulación se define una función en ART que ejecutará cualquier programa en Windows, figura 3.26, figura 3.27. El conjunto de reglas pertenecientes a la monitorización acceden periódicamente a la base de datos de Access donde se almacenan los resultados de la simulación. El acceso del SBC a la base de datos para conocer el estado actual y el deseado del hospital se realiza por medio de librerías de enlace dinámico (dll).

```
(DEF-ART-FUN WIN-EXEC
   (?SZCOMMAND ?IPARAM)
   (BIND ?WIN-EXEC
      (DLL-LOCATE "KERNEL" "WINEXEC" "ICI"))
   (BIND ?SUCCESS
      (DLL-CALL ?WIN-EXEC ?SZCOMMAND ?IPARAM))
   (DLL-FREE ?WIN-EXEC))
```

Figura 3.26 .- Función definida en ART-IM que permite la ejecución de programas bajo windows.

```
(defrule M_Inicio_Simulacion
     (declare (ruleset monitorizacion))
               (NOT (Fin_Simulacion))
               (NOT (Simulando))
     =>
               (WIN-EXEC "gine2.exe" 1)
               (set-asynch-fun fin)
               (assert (Simulando))
)
```

Figura 3.27.- Regla para ejecutar el programa de simulación "gine2.exe".

b) Diagnóstico.- Cuando la monitorización detecta alguna variable fuera de rango deshabilita las reglas asociadas con su tarea y habilita las reglas asociadas con el módulo de diagnóstico. Estas determinan en que servicio existe un problema y de qué problema se trata. Para realizar su trabajo necesitan conocer la interrelaciones entre los servicios, para ello tienen que acceder a la base de datos donde se guarda la estructura del hospital ejemplo.-

c) Predicción.- Para cada problema diagnosticado se ejecuta el conjunto de reglas del modulo de predicción. Estas llaman a un programa de control (realizado en lenguaje C) que genera las acciones de control locales (control tipo PID). Para cada servicio con problema se genera una tabla con todas las acciones de control propuestas. Estos resultados son almacenados en la base de datos, con la característica de ser conocimiento temporal cuya validez caduca al ser diseñado en el siguiente módulo la solución global.

d) Diseño.- El módulo de diseño dispara un programa que realiza la búsqueda de la solución global siguiendo una estrategia de primero-el-mejor, para luego seguir con una búsqueda en profundidad mejorando la solución de partida. El programa de búsqueda comunica cada posible nodo objetivo al módulo de diseño y éste comprueba si es el resultado deseado realizando la simulación de esa solución. Si no se obtiene el estado deseado, ART marca como no utilizable los recursos que han ocasionado fallo de la solución y le indica al programa de búsqueda que continúe la búsqueda.

Este prototipo, en fase de realización, muestra de forma tangible la cooperación entre la Simulación y la IA. Dicho prototipo nos está permitiendo validar la metodología propuesta sin necesidad de recurrir a hospitales reales que complicaría la implementación innecesariamente. Además, esta aplicación nos permitirá realizar presentaciones del modelo propuesto en diferentes organizaciones sanitarias sin hacer referencia a ningún hospital concreto.

CONCLUSIONES

CONCLUSIONES

Se ha presentado en este trabajo una metodología basada en Simulación e Inteligencia Artificial que nos ha permitido aportar un modelo dinámico a la toma de decisiones en la gerencia hospitalaria, y el cual es susceptible de generalizar a cualquier organización.

Esta metodología se desarrolla a partir del:

- análisis de la organización.

- automatización de la gestión de la información obtenida del análisis anterior.

Se realiza un estudio exhaustivo de la organización, que en este trabajo ha sido de tipo sanitario. Para realizar esta primera tarea presentamos como elemento imprescindible la simulación de la dinámica de la organización (hospital) que fue modelizado como un sistema de eventos discretos. La ventaja principal que nos ofrece una herramienta de estas características es la facilidad en la adquisición del conocimiento. Debido a que el objetivo final es una adecuada gestión de la organización se realiza una microsimulación, esto es, una simulación en la que tengamos total conocimiento de todas las transiciones que tienen lugar en el sistema. Inicialmente se realizó una simulación orientada al evento utilizando las Redes de Petri (RdP) por ser

una herramienta matemática que facilitaba el modelado y depuración del sistema. Pero debido a la envergadura y flexibilidad necesaria en un simulación de este tipo, las RdP no resultaron el mecanismo más adecuado. Por esto se realizó una simulación orientada al proceso, esto es, se definieron las reglas de comportamiento de cada elemento que interviene en la organización. Este tipo de simulación ofrece la ventaja añadida de estar orientado al usuario, cuestión necesaria en toda gestión moderna e innovadora.

El programa resultante sirve para realizar distintos experimentos con hospitales virtuales permitiendo al personal de gerencia adquirir un completo conocimiento de cómo es su hospital y cómo responderá frente a diferentes situaciones. Si los experimentos se realizan con organizaciones muy complejas puede resultar un tiempo de ejecución excesivo. La solución adoptada en este trabajo fue focalizar el problema en estudio y reemplazar la simulación microscópica del resto de la organización por una simulación macroscópica (tiempos medios), con ello se obtienen tanto las ventajas de la microsimulación (total conocimiento de cada elemento que interviene en el problema) como las de la macrosimulación (reducción del tiempo de ejecución). Una línea abierta de este trabajo es la paralelización del programa de simulación que permite abordar la simulación totalmente microscópica.

La segunda fase de la metodología ha consistido en el tratamiento automático de la información obtenida en la fase de análisis, para inferir las medidas que se deben adoptar y alcanzar los objetivos de calidad y eficiencia exigido en la organización.

La información recogida es muy abundante, presenta muchas interrelaciones y se necesita conocimiento heurístico para un estudio eficiente de la misma. Por estas razones se presenta el uso de técnicas de Inteligencia Artificial como el complemento adecuado para realizar este trabajo.

Se propone un Sistema Basado en Conocimiento como mecanismo necesario para completar el estudio de la gestión de las organizaciones. El desarrollo de este sistema se plantea de forma genérica para que pueda ser utilizado en cualquier organización sanitaria y con pequeñas modificaciones en cualquier organización de características similares. Con esta perspectiva y para simplificar el proceso de creación y depuración del SBC se sigue una metodología de estructuración del conocimiento llamada KADS. Esto nos ha permitido realizar el diseño desde distintos puntos de vista.

Inicialmente, se define el modelo organizativo que sitúa el SBC dentro de la organización en la que trabajará. A continuación, el modelo de tareas especifica las funciones del SBC, y el modelo de agentes indica con que otros componentes (usuarios, software) colaborará.

El elemento más importante en el diseño del SBC fue el modelo experto donde se define todo el conocimiento que debe tener para su correcto funcionamiento. La forma en la que se expresa este conocimiento es importante debido a que permite su reutilización en otras organizaciones. Para ello, se siguió en este trabajo la metodología KADS que divide el conocimiento en dominio y control.

El conocimiento de dominio determina aquellos conceptos y relaciones que debe conocer el SBC para resolver el problema que se plantea, es decir, para realizar una gestión hospitalaria. Para definir esta teoría se utilizó el lenguaje Ontolingua que permite el uso del KIF (lenguaje de intercambio de conocimiento), así como la reutilización de piezas de conocimiento definidas en otras teorías. Como resultado se presenta el desarrollo de una Ontología de Gerencia Hospitalaria que puede ser reutilizada en la construcción de cualquier SBC para la ayuda a la toma de decisiones en la gestión hospitalaria, y que puede servir como ontología base sobre la que definir la gerencia en otra organización de tipo no sanitario.

El conocimiento de control describe el proceso de razonamiento necesario en la resolución del problema. Debido a que todos los expertos resuelven los problemas de la misma forma en distintos campos del saber, se recurre a métodos de resolución de problemas utilizados en procesos de razonamiento similares. Un gerente de cualquier organización tiene que realizar una toma de decisiones que es un proceso de razonamiento frecuentemente utilizado en distintas áreas de conocimiento. Para describir en detalle este proceso se divide en las tareas que lo componen:

* monitorización: detectar los valores no correctos en la organización.

* diagnóstico: determinar los problemas existentes.

* predicción: encontrar las posibles soluciones a los problemas. Una aportación de nuestro trabajo en esta tarea es el uso del programa de simulación de la dinámica de las organizaciones como elemento imprescindible para conocer las consecuencias de las

distintas decisiones y por lo tanto saber cual es la más conveniente. Además, se consideró el estudio de soluciones locales a los distintos componentes de la organización (los servicios en los hospitales) para conseguir su correcto funcionamiento. Se utilizó en esta metodología decisiones del tipo PID (proporcional-integral-derivativa), tomadas de la teoría de control. La explicación del uso de este tipo de acción fue que el gerente frente a un problema realiza una corrección que es proporcional al error detectado (acción proporcional). Debido a que con este tipo de acción no se obtiene un error nulo, se debe tener en cuenta la historia pasada del error para conseguir los valores deseados (acción integral). Finalmente para evitar realizar correcciones de magnitud superior a la necesaria se tiene que consideran las variaciones del error (acción derivativo).

* diseño: construir, a partir de las primitivas dadas en la tarea de predicción, la solución que resuelva todos los problemas existentes en el sistema. Esta tarea se planteó como una búsqueda en un espacio de soluciones donde el uso de funciones heurísticas permitieron la poda de ramas.

Para resolver cada una de estas tareas se dividieron en pasos más elementales o inferencias. La descripción de cada uno de estos pasos sirve para completar el diseño de un SBC para la toma de decisiones. El siguiente trabajo consistió en que la toma de decisiones se realizara en la gerencia hospitalaria, esto es, se asoció el conocimiento de dominio con el de control. Para ello se presentó las estructuras de inferencias que muestran esta relación.

Finalmente se ha construido un prototipo con la herramienta de desarrollo ART que presenta los requisitos impuestos por el SBC: lenguaje de emparejamiento de patrones, comunicación con otras aplicaciones, facilidad de depuración y tiempo de ejecución adecuado.

Con esta metodología propuesta se ha comprobado como la Simulación y la Inteligencia Artificial son mecanismos que se complementan en el estudio de sistemas complejos, permitiendo la construcción de sistemas de gestión de la información que además de almacenar y manipular datos ofrezcan la posibilidad de conocer el comportamiento dinámico del sistema en estudio, predecir las consecuencias de las actuaciones, así como poder inferir por qué una toma de decisiones es más adecuada

que otra.

APÉNDICE A

APÉNDICE A

A continuación se presenta el código del programa de simulación realizado en Modsim II. El programa se divide en cuatro módulos cada uno de ellos formado por dos ficheros, uno en el que se definen los datos que intervienen en el problema y otro donde se realiza la implementación de la simulación.

A.1.- Definición del módulo Servicio

```
DEFINITION MODULE Servicio;

FROM GrpMod IMPORT QueueObj,RankedObj;
FROM ResMod IMPORT ResourceObj;
FROM SimMod IMPORT TriggerObj;

CONST DIA = 1440.0;  { minutos que tiene un dia }
      HORA = 60.0;  {minutos que tiene una hora}

VAR
      TIEMPOSIM: REAL; { = 100.0 * DIA;} { siete dias }
      DIASSIM: INTEGER; { dias de simulacion }
      salvarestado : BOOLEAN;

TYPE
       diario = [1..24];
       {un turno en un horario viene definido por un par de horas}

       tipolistaturno = RECORD
                        hinicio:  diario;
                        hfin: diario;
                        ocupacion: ActividadObj;
                        sigturno: tipolistaturno; {puntero al siguiente tur.}
```

```
                    END RECORD;
        tipolistareales = RECORD
                        valor: REAL;
                        ocupacion: ActividadObj;
                        next: tipolistareales;
                       END RECORD;
       tipohorario = ARRAY [1..7] OF tipolistaturno;
       tipoarrayact = ARRAY INTEGER,INTEGER OF ActividadObj;
       tipoarraycola = ARRAY INTEGER OF QueueObj;

{ ------------------------------------------------------------- }

       ActividadObj = OBJECT

              id: INTEGER;
              recurso: ResourceObj;
              colrec: QueueObj; {cola de RecursoObj}
              nombre: STRING;

              ASK METHOD InitActividad(IN r: ResourceObj; IN q: QueueObj; IN n: STRING;
IN ident:INTEGER);

       END OBJECT;


{ ------------------------------------------------------------- }
       PruebaSObj = OBJECT
              nombre: STRING; { nombre de la prueba }
              duracion: REAL; { tiempo que tarda la prueba }
              colact: QueueObj; { cola de Actividad }
              pendientes: ARRAY INTEGER OF INTEGER;
              totalpendientes: ARRAY INTEGER OF INTEGER;

              ASK METHOD InitPrueba(IN n: STRING; IN d: REAL; IN q: QueueObj);
              ASK METHOD AddActividad(IN a: ActividadObj);
              ASK METHOD TeSolicito(IN dia:INTEGER);
              ASK METHOD TeConsegui(IN dia:INTEGER);
              ASK METHOD CalculaTotal(IN dia:INTEGER);
       END OBJECT;

{ -------------------------------------------------------------}

{ un recurso aparte de las propiedades heredadas, debe tener un horario }
   RecursoObj = OBJECT(ResourceObj)
       horario: tipohorario;
       numrec : INTEGER; {numero de instancias que tiene}
       conocupa: BOOLEAN; { flag que indicara si es un recurso
                            que dependiendo del turno esta en una ocupacion
                           u otra }
       objeto: ANYOBJ; {objeto que lo tiene adquirido en ese momento}
       metodo: ACTID; {metodo que esta ejecutando quien lo tiene}
       tipo: STRING; {identificador del tipo de objeto}

       ASK METHOD InitRecurso(IN h: tipohorario;IN t:STRING);
       ASK METHOD AdquiridoPor(IN obj:ANYOBJ; IN met:ACTID);
       ASK METHOD PonteDisponible(IN act: ActividadObj);
       TELL METHOD Adquierete(IN act: ActividadObj; IN aid:ACTID);
```

```
        TELL METHOD GestionHorario;

    END OBJECT;


{ ------------------------------------------------------------- }
    ConsultaObj = OBJECT(RecursoObj) { Objeto que representa un consultorio }
        id : INTEGER; { identificador de la consulta }
        diaprim: INTEGER; {ultimo dia que tiene ya reservas prim. cons.}
        minutosprim: INTEGER; {minutos que le quedan libres del dia a prim.cons}
        diasuc:INTEGER; {idem de diaprim para consultas sucesivas}
        minutossuc: INTEGER; {iden de minutosprim para consultas suc. }
        actividad: ActividadObj; {referencia a la actividad a a la que se dedica }

        durconsp,durconss: INTEGER; {duraciond e consulta primera y sucesivas en minutos
}

        ASK METHOD InitConsulta(IN h:tipohorario;IN ppr: REAL; IN idcons: INTEGER; IN ac:
ActividadObj
                               ; IN dcp,dcs:INTEGER);
        ASK METHOD DaCita(IN primera: BOOLEAN;IN pprim:REAL):REAL;

    END OBJECT;

{ ------------------------------------------------------------------}
    PatologiaObj = OBJECT {objeto que representa una patologia}
        nombre: STRING;
        servicio: ServicioObj;
        numconsuc: INTEGER; { numero de consultas sucesivas asociada a pat.}
        porcpac: REAL; {tanto por uno del total de pacientes que llegan
                       que son de esta determinada patologia}
        porcprim: REAL; {tanto por uno del total de horas reservado para
                         primeras consultas }
        tmprequir:   INTEGER; {tiempo medio prequirurgico en unidades de
                              tiempo de simulacion }
        tmpostquir:  INTEGER; {tiempo medio prequirurgico en unidades de
                              tiempo de simulacion }
        tmquirofano: INTEGER; {tiempo medio prequirurgico en unidades de
                              tiempo de simulacion }
        consultas: QueueObj; {es una cola de consultas}
        colapruebas: QueueObj; {cola de pruebas que se puede hacer (se guarda el nombre)
}
        { la fila 0 es para la primera consulta y la uno para la sucesiva }
        pendientes: ARRAY [0..1],INTEGER OF INTEGER;
        totalpendientes: ARRAY [0..1],INTEGER OF INTEGER;

        ASK  METHOD  InitPatologia(IN  nom:STRING;  IN  pp,ppr:REAL;IN  tpre,tpost,tqui:
INTEGER;
                                  IN ser: ServicioObj; IN ncs: INTEGER; IN cp: QueueObj);
        ASK METHOD MarcaPruebas(IN nc: INTEGER):QueueObj;
        ASK METHOD DaConsulta(IN primera: BOOLEAN;OUT pri:REAL):ConsultaObj;

        ASK METHOD CuantasSucesivas():INTEGER;
        TELL METHOD ArrancaConsultas;
        ASK METHOD TeSolicito(IN dia:INTEGER;IN tipoconsulta:INTEGER);
        ASK METHOD TeConsegui(IN dia:INTEGER;IN tipoconsulta:INTEGER);
        ASK METHOD CalculaTotal(IN dia:INTEGER);
```

```
   END OBJECT;

{ ----------------------------------------------------------------}

   ServicioObj = OBJECT {objeto servicio}
           nombre: STRING;
           lispat: QueueObj;
           lispac: QueueObj; { Lista de pacientes que esta en el sistema }
           npc: INTEGER; {numero de pacientes diarios que llegan al servicio }
           numrecursos: INTEGER; { numero de recursos diferentes que hay en el servicio
}
           listarec: tipoarraycola;
           listaact: tipoarrayact;

              ASK METHOD InitServicio(IN numpac:INTEGER; IN name: STRING; IN colpat:
QueueObj;
                     IN ar: tipoarraycola; IN aa: tipoarrayact;IN nrec:INTEGER);
                  ASK METHOD ActividadesConsulta(IN cons: ConsultaObj;IN cadac:
STRING):QueueObj;
           ASK METHOD LanzarRecursos;
           TELL METHOD GenerarPacientes(IN state:INTEGER);
           TELL METHOD SaveState;
   END OBJECT;

{ ----------------------------------------------------------------}
{ El objeto servicio central hereda las características del objeto Servicio,
  cambia la lista de patologias por un Array de pruebas que tiene que realizarse }

   SerCentralObj = OBJECT(ServicioObj);

        colapruebas: QueueObj; {cola de pruebas de la que dispone el s.c.}
        { hereda numrecursos, listarec, listaact y el metodo Lanzar Recursos }

        ASK METHOD InitServicioCentral(IN colpru: QueueObj; IN ar: tipoarraycola;
                     IN aa: tipoarrayact;IN nrec:INTEGER);
        TELL METHOD GestionaPruebas(IN state:INTEGER);

    END OBJECT;

{ ----------------------------------------------------------------}
   PruebaObj = OBJECT(ResourceObj);

        nombre: STRING;
        tmr: REAL; { tiempo medio que se tarda en realizar la prueba }
        ASK METHOD InitPrueba(IN n:STRING; IN t:REAL; IN instancias: INTEGER);

   END OBJECT;

{ ----------------------------------------------------------------}

END MODULE.
```

A.2.- Implementación del módulo Servicio

```
IMPLEMENTATION MODULE Servicio;

FROM SimMod IMPORT SimTime,ResetSimTime,InterruptAll, InterruptMethod, Timescale;
FROM GrpMod IMPORT QueueObj;
FROM IOMod IMPORT ReadKey;
FROM ResMod IMPORT ResourceObj, EntryObj;
FROM IOMod IMPORT ReadKey;
FROM Paciente IMPORT PacienteObj;
FROM                                CAccess                                IMPORT
InicializaConexion,CierreConexion,ExecSQL,ReadRecord,ReadField,WriteSQL;
FROM Proced IMPORT DiaSimulacion,DiaHoy,EscribeHorario,SelectPatologia,SelectPrueba,
                   NumHorasTurno,ImprimeColaPacientes,Cierra;

OBJECT RecursoObj;

   ASK METHOD InitRecurso(IN h: tipohorario; IN t:STRING);
   BEGIN
      horario := h;
      tipo := t;
      numrec := 1;
   END METHOD;

   ASK METHOD AdquiridoPor(IN obj:ANYOBJ; IN met:ACTID);
   BEGIN
       objeto := obj;
       metodo := met;
   END METHOD;

   ASK METHOD PonteDisponible(IN act: ActividadObj);
   BEGIN
       { si es realmente una actividad se mete en la cola de actividades }
       IF (ASK act colrec) <> NILOBJ
              ASK (ASK act colrec) TO Add(SELF);
       END IF;
       { pone disponible tantos recursos como contenga }
       ASK (ASK act recurso) TO TakeBack(SELF,numrec);
       AdquiridoPor(NILOBJ,NILREC);
   END METHOD;

   TELL METHOD Adquierete(IN act: ActividadObj; IN aid: ACTID);
   VAR q: QueueObj;
   BEGIN
       q := ASK act colrec;

       WAIT FOR (ASK act recurso) TO PriorityGive(SELF,numrec,TIEMPOSIM);
          IF q <> NILOBJ
            IF ASK q TO Includes(SELF)
              ASK q TO RemoveThis(SELF);
            END IF;
          END IF;
          AdquiridoPor(SELF,aid);
        ON INTERRUPT
```

```
        IF (SimTime() >= TIEMPOSIM )
         TERMINATE;
        END IF;
      END WAIT;
   END METHOD;

   TELL METHOD GestionHorario;
   VAR i: INTEGER;
       t,resto: REAL;
       head,ant,r: tipolistareales;
       p :tipolistaturno;
       flag,primerdia : BOOLEAN;
       res : ResourceObj;
       colaact : QueueObj; { cola de todas las actividades que realiza }
       pac,pak: PacienteObj;
       obj2,pac2: ANYOBJ;
       obj3,cualquier: EntryObj;
       aid : ACTID;


   BEGIN

        NEW(colaact); { creo la cola de actividades }

        head := NILREC;
       ant := NILREC;
       r := NILREC;
       t := 0.0;

       { voy a crear una lista circular. Cada nodo de la lista esta formado por un valor
real y un puntero a otro nodo. Cada uno de estos valor es reales corresponde a un tiempo
en minutos que se debe de esperar hasta realizar una  accion. Se espera el tiempo y
despues se ejecuta la accion. Hay dos tipos de acciones que se van a dar de forma alter
nada: (liberar recursos y coger recursos con maxima prioridad) se comienza por liberar
recursos, ya que se supone que se comienza la simulacion con los recursos reservados }

       FOR i := 1 TO 7 { recorro el horario para cada dia de la semana}

         p := horario[i];
         IF p = NILREC
               t := t + 24.0 * HORA;
        ELSE

            WHILE p <> NILREC { recorro los turnos de cada dia }

                 NEW(r); {voy creando la lista de tiempos}
                 IF head=NILREC
                       head:= r; {establezco la cabeza}
                 ELSE
                       ant.next := r; { enlazo con el anterior }
                 END IF;
                 t := FLOAT(p.hinicio) * HORA + t;
                 r.valor := t;
                 r.ocupacion := p.ocupacion;
                 { voy metiendo las ocupaciones en una cola para despues
                   adquirirlas en un principio }
                 IF (NOT ASK colaact TO Includes(ASK r.ocupacion recurso))
```

```
                    ASK colaact TO Add(ASK r.ocupacion recurso);
                END IF;
                IF ant<>NILREC ant.next := r; END IF;
                ant := r;
                NEW(r);
                r.valor := FLOAT(p.hfin - p.hinicio) * HORA;


                r.ocupacion := p.ocupacion;
                { voy metiendo las ocupaciones en una cola para despues
                  adquirirlas en un principio }
                IF (NOT ASK colaact TO Includes(ASK r.ocupacion recurso))
                    ASK colaact TO Add(ASK r.ocupacion recurso);

                END IF;
                IF p.sigturno = NILREC
                   t := FLOAT(24 - p.hfin) * HORA; {sumamos lo que queda del dia}
                ELSE
                   t := -(FLOAT(p.hfin)*HORA);
                END IF;
                p := p.sigturno;
                ant.next := r;
                ant := r;
            END WHILE;
          END IF;
       END FOR;


       ant.next := head; { ya que se trata de una lista circular }
       resto := t; { lo que quedo por esperar hasta el final de la semana}
       r := head;
       ant := head;

       aid := THISMETHOD;
        { si tiene varias ocupaciones, aqui tiene que recogerlas todas }
       WHILE (ASK colaact First() <> ASK colaact Last())
         res := ASK colaact TO Remove();
         WAIT FOR res TO PriorityGive(SELF,numrec,TIEMPOSIM);
            AdquiridoPor(SELF,aid);
                       OUTPUT(tipo,"   adquiere   recursos   con   maxima   prioridad,
",DiaSimulacion(SimTime()));
          ON INTERRUPT
             IF (SimTime() >= TIEMPOSIM )
               TERMINATE;
             END IF;
          END WAIT;
       END WHILE;
       res := ASK colaact TO Remove();
        WAIT FOR res TO PriorityGive(SELF,numrec,TIEMPOSIM);
            AdquiridoPor(SELF,aid);
                       OUTPUT(tipo,"   adquiere   recursos   con   maxima   prioridad,
",DiaSimulacion(SimTime()));
        ON INTERRUPT
             IF (SimTime() >= TIEMPOSIM )
               TERMINATE;
             END IF;
        END WAIT;
```

```
        DISPOSE(colaact);


        t := SimTime();
        flag := TRUE;
        primerdia := TRUE;
        WHILE ((t + r.valor) < TIEMPOSIM)
                WAIT DURATION r.valor;
                ON INTERRUPT
                  IF (SimTime() >= TIEMPOSIM )
                    TERMINATE;
                   END IF;
                END WAIT;
                IF flag
                        OUTPUT(tipo,"                    Pone                    Recursos
disponibles,",DiaSimulacion(SimTime()));
                        ASK SELF TO PonteDisponible(r.ocupacion);
                ELSE
                        {  obj3  :=  ASK  (ASK  (ASK  r.ocupacion  recurso)  AllocationList)
First(); }
                        pac := objeto;
                        { IF obj3 <> NILOBJ
                                obj2 := ASK obj3 Object;
                                pak := obj2;
                                OUTPUT("quien tiene el rec. que quiere ",tipo," es el pac
id ",ASK pak id);
                        END IF;}

                        IF pac <> NILOBJ
                                OUTPUT(tipo," va a interrumpir al paciente ",ASK pac id,"
en ",DiaSimulacion(SimTime()));

                                InterruptMethod(metodo);
                                { No se pone el wait duration porque si no inmediatamente
despues de interrumpir entra el paciente en el on interrupt devuelve el recurso y lo
coge el siguiente antes de que se ejecute el adquierete. Sin embargo, si no lo ponemos
se produce en el evento siguiente, y ya el esta en el adquierete y recoge el recurso }
                                 {WAIT DURATION 0.0
                                 ON INTERRUPT
                                        TERMINATE;
                                 END WAIT;}

                        END IF;
                        WAIT FOR SELF TO Adquierete(r.ocupacion,aid);
                        ON INTERRUPT
                                IF (SimTime() >= TIEMPOSIM )
                                        TERMINATE;
                                END IF;
                        END WAIT;
                        OUTPUT(tipo," Adquirio los Recursos, ",DiaSimulacion(SimTime()));
                END IF;
                flag := NOT flag; {las acciones son alternadas}
                t := SimTime();
                r := r.next; { realizamos la siguiente accion }
                IF primerdia { una vez transcurrido el inicio hay que sumar
                                el resto hasta el final de la semana }
                        primerdia := FALSE;
```

```
                head.valor := head.valor + resto;
            END IF;
        END WHILE;
        r := head.next;
        head.next :=  NILREC;
        REPEAT
            ant := r;
            DISPOSE(ant);
            r := r.next
        UNTIL r=NILREC;

    END METHOD;
END OBJECT;



OBJECT PatologiaObj;
        ASK METHOD InitPatologia(IN nom:STRING; IN pp,ppr:REAL;
                                 IN tpre,tpost,tqui: INTEGER; IN ser: ServicioObj;
                                  IN ncs: INTEGER; IN cp: QueueObj);
        VAR i,j:INTEGER;
        BEGIN
            nombre := nom;
            porcpac := pp;
            porcprim := ppr;
            tmprequir := tpre;
            tmpostquir := tpost;
            tmquirofano := tqui;
            servicio := ser;
            numconsuc := ncs;
            colapruebas := cp;
            NEW(consultas);
            { inicilizamos los arrays para las estadisticas }
            NEW(pendientes,0..1);
            NEW(pendientes[0],1..DIASSIM);
            NEW(pendientes[1],1..DIASSIM);
            NEW(totalpendientes,0..1);
            NEW(totalpendientes[0],1..DIASSIM);
            NEW(totalpendientes[1],1..DIASSIM);
            FOR j := 0 TO 1
              FOR i := 1 TO DIASSIM
                   pendientes[j,i] := 0;
                   totalpendientes[j,i] := 0;
              END FOR;
            END FOR;
        END METHOD;

        ASK METHOD DaConsulta(IN primera: BOOLEAN; OUT pri:REAL):ConsultaObj;
         VAR  cons,mincons : ConsultaObj;
              min,dia : INTEGER;
        BEGIN

            { aqui se mira que consulta es la que tiene menor el campo
              diaprim si es de primera consulta o diasuc si es de
              sucesivas }

            cons := ASK consultas TO First();
```

```
            mincons := cons;
            IF primera
                   min := ASK cons TO diaprim;
                   WHILE cons <> ASK consultas TO Last();
                      cons := ASK consultas TO Next(cons);
                      dia := ASK cons TO diaprim;
                      IF dia < min
                           min := dia;
                            mincons := cons;
                      END IF;
                   END WHILE;

            ELSE
                   min := ASK cons TO diasuc;
                   WHILE cons <> ASK consultas TO Last();
                      cons := ASK consultas TO Next(cons);
                      dia := ASK cons TO diasuc;
                      IF dia < min
                           min := dia;
                            mincons := cons;
                      END IF;
                   END WHILE;
            END IF;
            { ahora se le pida prioridad para esa consulta }
            pri := ASK mincons TO DaCita(primera,porcprim);
            RETURN mincons;
      END METHOD;

      ASK METHOD CuantasSucesivas():INTEGER;
      BEGIN
            RETURN(numconsuc);
      END METHOD;

      { Este metodo es invocado por los pacientes para que se le marquen las pruebas a
realizarse en funcion del numero de consultas que lleva realizadas. Lo que se devuelve
es una cola de las pruebas que tiene que realizarse }
      ASK METHOD MarcaPruebas(IN nc: INTEGER):QueueObj;
      VAR npru,resta,tpru,i : INTEGER;
          q : QueueObj; {cola de pruebas que se le va a mandar }
          p: PruebaSObj;
      BEGIN

          IF colapruebas <> NILOBJ
             npru := ASK colapruebas numberIn;
             { regla de tres para saber cuantas pruebas se tiene que hacer en funcion
               de la consulta sucesiva que sea }
             tpru := ( (numconsuc+1-nc) * npru) DIV numconsuc;
             IF tpru > 0
                   NEW(q);
                   p := ASK  colapruebas TO First();
                   FOR i := 1 TO tpru
                      ASK q TO Add(p);
                      p := ASK colapruebas TO Next(p);
                   END FOR;
                   RETURN(q);
             ELSE
                   RETURN(NILOBJ);
```

```
            END IF;
        ELSE
            RETURN(NILOBJ);
        END IF;
    END METHOD;

    TELL METHOD ArrancaConsultas;
    VAR i,n: INTEGER;
        cons : ConsultaObj;
    BEGIN
          IF consultas<>NILOBJ
           n := ASK consultas numberIn;
           FOR i:=1 TO n
                  IF i=1
                     cons := ASK consultas First();
                  ELSE
                     cons := ASK consultas Next(cons);
                   END IF;
                  TELL cons GestionHorario;
           END FOR;
          END IF;
    END METHOD;

    ASK METHOD TeSolicito(IN dia:INTEGER;IN tipoconsulta:INTEGER);
    BEGIN
           INC(pendientes[tipoconsulta,dia]);
    END METHOD;
    ASK METHOD TeConsegui(IN dia:INTEGER;IN tipoconsulta:INTEGER);
    BEGIN
           DEC(pendientes[tipoconsulta,dia]);
    END METHOD;
    ASK METHOD CalculaTotal(IN dia:INTEGER);
    VAR i:INTEGER;
    BEGIN
           FOR i:=1 TO dia
               totalpendientes[0,dia] := totalpendientes[0,dia]+pendientes[0,i];
               totalpendientes[1,dia] := totalpendientes[1,dia]+pendientes[1,i];

           END FOR;
    END METHOD;


END OBJECT;



OBJECT ConsultaObj;

    ASK METHOD InitConsulta(IN h:tipohorario;IN pprim: REAL; IN idcons: INTEGER; IN
ac:ActividadObj;
                            IN dcp,dcs:INTEGER);
    VAR i: INTEGER;
        aux : REAL;
    BEGIN
           horario := h;
           id := idcons;
           durconsp := dcp;
```

```
            durconss := dcs;
            numrec := 1; {numero de recursos de este tipo que hay}
            tipo := "consulta";
            actividad := ac;
            conocupa := FALSE;
            ASK SELF TO Create(numrec);
            i := 1;
            WHILE (h[i]=NILREC)
                   i := i + 1;
            END WHILE;
            diaprim := i;
            diasuc := i;
            aux := FLOAT(NumHorasTurno(h,i)) * HORA;

            minutosprim := ROUND(aux * pprim);
            minutossuc := ROUND(aux) - minutosprim;

      END METHOD;

      { Este método DaCita lo que hace es devolver una prioridad con la que el paciente
se pondrá en cola de esa consulta. La forma en la que se calcula esta prioridad es la
siguiente:
         prioridad = TOTALDIASSIMULACION - (dia_hoy + incremento )

         incremento = diaprim o diasuc - dia_hoy

         por lo tanto:
         si es de primera consulta
           prioridad = TOTALDIASSIMULACION - diaprim
         sino
           prioridad = TOTALDIASSIMULACION - diasuc
         fin si
      }

      ASK METHOD DaCita(IN primera: BOOLEAN;IN pprim: REAL):REAL;
      VAR ind: INTEGER; {indice del horario del dia correspondiente}
          horas: INTEGER; {horas que esta abierta la consulta}
          hoy,h,min,ultimahora : INTEGER; {para saber que dia es hoy}
      BEGIN
            DiaHoy(SimTime(),hoy,h,min);
            ind := (hoy MOD 7);
            IF ind=0 ind:= 7; END IF;
            ultimahora := Cierra(horario,ind); { hora a la que cierra hoy}
            IF primera
                   { Lo primero es comprobar si es menor que hoy}
                   { o si hoy ya cerro la consulta }
                   { IF (diaprim < hoy)
                          diaprim := hoy - 1;
                          minutosprim := 0;
                   ELSE
                       IF ((diaprim = hoy) AND (ultimahora <= h))
                          minutosprim := 0;
                       END IF;
                   END IF;}

                   { supone una media de DURCONS min. por consulta }
                   WHILE ((ASK SELF minutosprim < durconsp) AND (FLOAT(diaprim) <
```

```
TIEMPOSIM - 2.0))
                              diaprim := diaprim + 1; {avanzo el dia}
                              ind := (diaprim MOD 7);
                              IF ind=0 ind:= 7; END IF;
                              horas := NumHorasTurno(horario,ind);
                              { avanzo hasta el dia que haya consulta }
                              WHILE (horas <= 0)
                                 diaprim := diaprim + 1;
                                 ind := (diaprim MOD 7);
                                 IF ind=0 ind:= 7; END IF;
                                 horas := NumHorasTurno(horario,ind);
                              END WHILE;
                              { minutos de ese dia dedicacos a primera cons}

                              minutosprim := ROUND(FLOAT(horas) * HORA * pprim);
                       END WHILE;
                       minutosprim := minutosprim - durconsp; {quito la consulta}
                       RETURN( TIEMPOSIM - 1.0 - FLOAT(diaprim));
                ELSE
                       { Lo primero es comprobar si es menor que hoy}
                       { o si hoy ya cerro la consulta }
                       { IF (diasuc < hoy)
                              diasuc := hoy - 1;
                              minutossuc := 0;
                       ELSE
                           IF ((diasuc = hoy) AND (ultimahora <= h))
                              minutossuc := 0;
                           END IF;
                       END IF; }

                       { supone una media de DURCONS min. por consulta }
                       WHILE ((ASK SELF minutossuc < durconss) AND (FLOAT(diasuc) <
TIEMPOSIM - 2.0))
                              diasuc := diasuc + 1;
                              ind := (diasuc MOD 7);
                              IF ind=0 ind:= 7; END IF;
                              horas := NumHorasTurno(horario,ind);
                              { avanzo hasta el dia que haya consulta }
                              WHILE (horas <= 0)
                                 diasuc := diasuc + 1;
                                 ind := (diasuc MOD 7);
                                 IF ind=0 ind:= 7; END IF;
                                 horas := NumHorasTurno(horario,ind);
                              END WHILE;
                              { minutos de ese dia dedicacos a cons suc.}

                              minutossuc := ROUND(FLOAT(horas) * HORA * (1.0 - pprim));

                       END WHILE;
                       minutossuc := minutossuc - durconss; {quito la consulta}

                       RETURN ( TIEMPOSIM - 1.0 - FLOAT(diasuc));
                END IF;
        END METHOD;

END OBJECT;
```

```
OBJECT PruebaObj;
       ASK METHOD InitPrueba(IN n:STRING; IN t:REAL; IN instancias: INTEGER);
       BEGIN
              nombre := n;
              tmr:= t;
              ASK SELF TO Create(instancias); {creamos recurso }

       END METHOD;
END OBJECT;
OBJECT ServicioObj;

               ASK METHOD InitServicio(IN numpac:INTEGER; IN name: STRING; IN colpat:
QueueObj;
                     IN ar: tipoarraycola; IN aa: tipoarrayact;IN nrec:INTEGER);
               BEGIN
                     nombre := name;
                     npc := numpac;
                     lispat := colpat;
                     numrecursos := nrec;
                     listarec := ar;
                     listaact := aa;
                     NEW(lispac);
               END METHOD;

              { Este procedimiento va a devolver una lista de actividades que le hacen
falta al paciente para poder pasar la consulta; se le pasa como argumentos la consulta
que se le ha asignado segun su patologia, y una cadena de caracteres cadac que contiene
una clave que identifica que esa actividad esta relacionada con la actividad de pasar
consulta }

              ASK   METHOD   ActividadesConsulta(IN   cons:   ConsultaObj;IN   cadac:
STRING):QueueObj;
              VAR q,qaux,colr: QueueObj;
                  r : RecursoObj;
                  i,j: INTEGER;
                  acti: ActividadObj;
              BEGIN
                     NEW(q); {creo la cola de actividades}
                     { meto la actividad de la consulta que me han pasado }
                     acti := ASK cons actividad;
                     ASK q TO Add(acti);


                     { ahora tengo que meter todas las actividades que tengan la
palabra clave que estoy buscando cadac }
                     FOR i := 1 TO numrecursos
                          { para cada recurso miro si alguna de sus actividades
corresponde con la cadena }
                        FOR j := LOW(listaact[i]) TO HIGH(listaact[i])
                            IF ((ASK listaact[i,j] nombre) = cadac)

                                   ASK q TO Add(listaact[i,j]) ;
                            END IF;
                        END FOR;
                     END FOR;
                     RETURN(q);
```

```
END METHOD;


ASK METHOD LanzarRecursos;
VAR i,j,n: INTEGER;
    q: QueueObj;
    r: RecursoObj;
    p : PatologiaObj;
BEGIN

       IF lispat<>NILOBJ
              n := ASK lispat numberIn;
              FOR i:=1 TO n
                  IF i=1
                     p := ASK lispat First();
                  ELSE
                     p := ASK lispat Next(p);
                  END IF;
                  TELL p ArrancaConsultas;
              END FOR;
       END IF;


       FOR i:= 1 TO numrecursos
          q := listarec[i];
          r := ASK q TO First();
          TELL r TO GestionHorario;
          n := ( ASK q numberIn ) - 1;
          FOR j := 1 TO n
               { ponemos a ejecutar a cada recurso }
               r := ASK q TO Next(r);
               TELL r TO GestionHorario;

          END FOR;
       END FOR;
END METHOD;

{ tiene como parametro de entrada el estado en el que quiere comenzar. Si
se le pasa un
   numero negativo es que es una simulacion que empieza en el estado
origial }
TELL METHOD GenerarPacientes(IN state: INTEGER);
VAR n,npac,i,j,d,h,m: INTEGER;
    ts: REAL;
    pac : PacienteObj;
    patol: PatologiaObj;
    idpac,resultado,quecons,numpac,ident,npru : INTEGER;
    nompru,nompat,nomser,qry,qry2,consulta,linea,aux: STRING;
    colpru : QueueObj;
    prueba: PruebaSObj;
BEGIN
   WAIT DURATION 1.0;
       ON INTERRUPT
   END WAIT;

   { espero una unidad para que los pacientes no empiecen
     a pedir consulta antes de que esta sea reservada por
```

```
                    ellas mismas }
                IF ( state >= 0 ) { estado no inicial }
                    nomser := ASK SELF nombre;
                    consulta := "SELECT idpac, quecons, patologia FROM PACIENTES WHERE
idestado="+INTTOSTR(state)+" AND servicio='"+nomser+"' ORDER BY idpac;";
                    numpac := ExecSQL(consulta,qry);
                    FOR i := 1 TO numpac
                       resultado := ReadRecord(qry,i,linea);
                       ReadField(linea,1,aux);
                       idpac := STRTOINT(aux); { identificador del paciente }
                       ReadField(linea,2,aux);
                       quecons := STRTOINT(aux); { consulta en la que se encuentra }
                        ReadField(linea,3,nompat); { nombre de la patologia que padece
}
                         patol := SelectPatologia(ASK SELF lispat,nompat); { patologia
que tiene }
                           consulta := "SELECT prueba FROM PRUEBAPENDIENTE WHERE
idpac="+INTTOSTR(idpac)+";";
                       npru := ExecSQL(consulta,qry2);
                       IF npru > 0
                            NEW(colpru);
                       ELSE
                            colpru := NILOBJ;
                       END IF;
                       FOR j := 1 TO npru { anado las pruebas que se tiene que hacer }
                            resultado := ReadRecord(qry2,j,aux); { en nompru tenemos el
nombre prueba}
                            ReadField(aux,1,nompru);
                            prueba := SelectPrueba(ASK patol colapruebas,nompru);
                             IF prueba <> NILOBJ
                                ASK colpru TO Add(prueba);
                            END IF;
                       END FOR;
                       NEW(pac);
                       ASK lispac TO Add(pac); { lo metemos en la lista de pacientes }
                       ASK pac TO InitPaciente(idpac,patol,colpru,quecons);
                       WAIT DURATION 0.0
                            TELL pac TO EjecucionPaciente;
                       ON INTERRUPT
                       END WAIT;
                     END FOR;
                ELSE
                    idpac := 0;
                END IF;

                ident := idpac;
                ts := SimTime();
                WHILE ts <= TIEMPOSIM
                    { presupone que el campo de porcentajes de las
                      patologias esta bien }
                    patol := ASK lispat TO First();
                    WHILE patol <> ASK lispat TO Last()
                       { recorro las patologias generando los pacientes y calculando
los totales del dia anterior }
                      DiaHoy(SimTime(),d,h,m);
                      d := d - 1;
                      IF d > 0
```

```
                        ASK patol CalculaTotal(d);
                    END IF;
                    npac := ROUND(FLOAT(npc)*(ASK patol porcpac));
                    FOR i := 1 TO npac
                        WAIT DURATION FLOAT(i);
                        ON INTERRUPT
                            TERMINATE;
                        END WAIT;
                        INC(ident);
                        NEW(pac);
                        ASK lispac TO Add(pac); { anadimos el paciente a la lista }
                        ASK pac TO InitPaciente(ident,patol,NILOBJ,0);
                        TELL pac TO EjecucionPaciente;
                    END FOR;
                    patol := ASK lispat TO Next(patol);
                END WHILE;
                DiaHoy(SimTime(),d,h,m);
                d := d - 1;
                IF d > 0
                    ASK patol CalculaTotal(d);
                END IF;
                npac := ROUND(FLOAT(npc)*(ASK patol porcpac));
                FOR i := 1 TO npac
                   INC(ident);
                   NEW(pac);
                   ASK lispac TO Add(pac); { anadimos el paciente a la lista }
                   ASK pac TO InitPaciente(ident,patol,NILOBJ,0);
                   TELL pac TO EjecucionPaciente;
                END FOR;
                { espero hasta el siguiente dia }
                WAIT DURATION DIA;
                ON INTERRUPT
                    TERMINATE;
                END WAIT;

                ts := SimTime();
            END WHILE;
                    consulta := "DELETE * FROM SALIDA WHERE
idestado="+INTTOSTR(state)+";";
            WriteSQL(consulta);
            n := ASK lispat TO numberIn;
            FOR i := 1 TO n
                IF i = 1
                    patol := ASK lispat First();
                 ELSE
                    patol := ASK lispat TO Next(patol);
                END IF;
                FOR j := 1 TO DIASSIM
                    consulta := "INSERT INTO SALIDA (dia, resultado, nomser,
nompat, tipo, idestado ) VALUES ("+INTTOSTR(j)+", "+INTTOSTR(ASK patol
totalpendientes[0,j])+", '"+(ASK SELF nombre)+"', '"+(ASK patol nombre)+"', 'primera'
,"+INTTOSTR(state)+");";
                    WriteSQL(consulta);
                    consulta := "INSERT INTO SALIDA (dia, resultado, nomser,
nompat, tipo, idestado ) VALUES ("+INTTOSTR(j)+", "+INTTOSTR(ASK patol
totalpendientes[0,j])+", '"+(ASK SELF nombre)+"', '"+(ASK patol nombre)+"', 'sucesiva'
,"+INTTOSTR(state)+");";
```

```
                        WriteSQL(consulta);
                    END FOR;
                END FOR;
                { Salva el estado del sistema }
                IF salvarestado
                  TELL SELF TO SaveState;
                END IF;
            END METHOD;

            TELL METHOD SaveState;
            VAR p: PacienteObj;
                pat: PatologiaObj;
                s: ServicioObj;
                bucle2,bucle,resultado,ncons,idpac,i,t,j,npru: INTEGER;
                qry,prueba,patologia,servicio: STRING;
                pru: PruebaSObj;
                colap : QueueObj;
                c: CHAR;
                consulta : STRING;
            BEGIN
                 { InicializaConexion(); }
                 t := ASK lispac TO numberIn;
                 FOR i := 1 TO t
                    p := ASK lispac TO Remove();
                    { salvamos los estados a la tabla }
                    idpac := ASK p id;
                    ncons := ASK p quecons;
                    colap := ASK p colapru;
                    pat := ASK p especialidad;
                    patologia := ASK pat nombre;
                    s := ASK pat TO servicio;
                    servicio := ASK s nombre;
                    consulta := "INSERT INTO PACIENTES (idpac, quecons, patologia,
servicio, idestado, descestado) VALUES ("+INTTOSTR(idpac)+", "+INTTOSTR(ncons)+",
'"+patologia+"', '"+servicio+"', "+INTTOSTR(1)+", "+"'Prueba'"+");";
                    WriteSQL(consulta);
                    IF colap <> NILOBJ
                      npru := ASK colap TO numberIn;
                      FOR j := 1 TO npru
                         pru := ASK colap TO Remove();
                         prueba := ASK pru nombre;
                            consulta := "INSERT INTO PRUEBAPENDIENTE (idpac, prueba,
idestado) VALUES ("+INTTOSTR(idpac)+", '"+prueba+"', "+INTTOSTR(1)+");";

                         WriteSQL(consulta);
                      END FOR;
                      DISPOSE(colap);
                    ELSE
                    END IF;

                    InterruptAll(p);
                    WAIT DURATION 0.0
                           DISPOSE(p);
                    ON INTERRUPT
                           TERMINATE;
                    END WAIT;
                 END FOR;
```

```
                DISPOSE(lispac);

            END METHOD;

END OBJECT;


{ ------------------------------------------------------------- }

OBJECT SerCentralObj;


    ASK METHOD InitServicioCentral(IN colpru: QueueObj; IN ar: tipoarraycola;
                    IN aa: tipoarrayact;IN nrec:INTEGER);
    BEGIN
       colapruebas := colpru;
       numrecursos := nrec;
       listarec := ar;
       listaact := aa;
    END METHOD;


    TELL METHOD GestionaPruebas(IN state:INTEGER);
    VAR pru : PruebaSObj;
       j,i,d,m,h,n: INTEGER;
       ts: REAL;
       consulta: STRING;
    BEGIN
        WAIT DURATION DIA
           ON INTERRUPT
              TERMINATE
        END WAIT;
        DiaHoy(ts,d,h,m);

       ts := SimTime();
       WHILE ts <= TIEMPOSIM
           IF d <= DIASSIM
                  n := ASK colapruebas numberIn;
                 FOR i := 1 TO n
                   IF i = 1
                      pru := ASK colapruebas First();
                   ELSE
                      pru := ASK colapruebas Next(pru);
                   END IF;
                   ASK pru CalculaTotal(d);
                 END FOR;
            END IF;
           WAIT DURATION DIA
           ON INTERRUPT
              TERMINATE
           END WAIT;
           ts := SimTime();
           INC(d);
       END WHILE;
       n := ASK colapruebas numberIn;
       consulta := "DELETE * FROM SALIDASC WHERE idestado="+INTTOSTR(state)+";";
       WriteSQL(consulta);
```

```
        FOR i := 1 TO n
          IF i = 1
               pru := ASK colapruebas First();
          ELSE
               pru := ASK colapruebas Next(pru);
          END IF;
          FOR j := 1 TO DIASSIM
               consulta := "INSERT INTO SALIDASC (dia, resultado, nomser, nompat,
idestado ) VALUES ("+INTTOSTR(j)+", "+INTTOSTR(ASK pru totalpendientes[j])+", 'Servicio
Central', '"+(ASK pru nombre)+"' ,"+INTTOSTR(state)+");";
               WriteSQL(consulta);
          END FOR;
        END FOR;
      END METHOD;


END OBJECT;
{ ------------------------------------------------------------------ }
   OBJECT PruebaSObj;

        ASK METHOD InitPrueba(IN n:STRING; IN d:REAL; IN q: QueueObj);
        VAR i: INTEGER;
        BEGIN
               nombre:= n; duracion := d; colact := q;
               { inicializamos los arrays de estadísticas }
               NEW(pendientes,1..DIASSIM);
               NEW(totalpendientes,1..DIASSIM);
               FOR i := 1 TO DIASSIM
                      pendientes[i] := 0;
                      totalpendientes[i] := 0;
               END FOR;
        END METHOD;

        ASK METHOD AddActividad(IN a: ActividadObj);
        BEGIN
               ASK colact TO Add(a);
        END METHOD;

        ASK METHOD TeSolicito(IN dia:INTEGER);
        BEGIN
               INC(pendientes[dia]);
        END METHOD;
        ASK METHOD TeConsegui(IN dia:INTEGER);
        BEGIN
               DEC(pendientes[dia]);
        END METHOD;
        ASK METHOD CalculaTotal(IN dia:INTEGER);
        VAR i:INTEGER;
        BEGIN
               FOR i:=1 TO dia
                   totalpendientes[dia] := totalpendientes[dia]+pendientes[i];
               END FOR;
        END METHOD;
   END OBJECT;
{ ------------------------------------------------------------------ }
   OBJECT ActividadObj;
```

```
       ASK METHOD InitActividad(IN r:ResourceObj; IN q: QueueObj; IN n:STRING; IN
ident:INTEGER);
       BEGIN
              recurso := r; colrec := q; nombre := n; id := ident;
              { REVISAR seria que se crearan tantos como recursos tienen esa actividad
              ASK recurso TO Create(1); }
       END METHOD;
   END OBJECT;
{ ---------------------------------------------------------------- }

END MODULE.
```

A.3.- Definición del módulo Paciente

```
DEFINITION MODULE Paciente;

FROM Servicio IMPORT PatologiaObj,ConsultaObj, PruebaSObj, ActividadObj;
FROM ResMod IMPORT ResourceObj;
FROM ListMod IMPORT QueueList;
FROM GrpMod IMPORT QueueObj;
FROM SimMod IMPORT TriggerObj;

TYPE
    SetRecursos = RECORD;
        idset: INTEGER;
        motivo: STRING;
        reccogidos: INTEGER;
        cola: QueueObj;
        colaids: QueueList; { cola de identificadores de actividad }
    END RECORD;

    PacienteObj = OBJECT
       id: INTEGER; {identificador de paciente}
       especialidad : PatologiaObj;
       interrumpido : BOOLEAN;
       colapru: QueueObj; { cola de pruebas que se ha de hacer }
       pruebashechas: INTEGER; { numero de pruebas que ya tiene hechas }
       colaset: QueueList; { para guardar los sets }
       nextset: INTEGER;
       quecons: INTEGER; {numero de la consulta en la que se encuentra:
                         0: en la primera consulta
                         >0: consulta sucesiva }


       ASK METHOD  InitPaciente(IN ident: INTEGER;IN esp: PatologiaObj; IN cp: QueueObj;
IN qc: INTEGER);
       ASK METHOD  QuitateDeLaCola(IN m: ResourceObj);
       TELL METHOD SolicitaNumero(IN primconsulta: BOOLEAN);
       TELL METHOD EjecucionPaciente;
       TELL METHOD EsperaPruebas(IN cp: QueueObj);
       TELL METHOD EsperaPor(IN p: PruebaSObj; IN sync: TriggerObj; IN npru: INTEGER);
       TELL METHOD GetRecurso(IN t:TriggerObj; IN act: ActividadObj; IN total:INTEGER;
IN set:SetRecursos;IN aid: ACTID;IN pri:REAL);
       ASK METHOD DevuelveCogidos(IN colaact: QueueObj);
```

```
        TELL METHOD Consigue(IN colaact: QueueObj; IN duracion: REAL; IN pri:REAL);
        ASK METHOD ObjTerminate;

    END OBJECT;

    PROCEDURE DestruyeCola(INOUT cola: QueueObj);

VAR
        atendidosp, atendidoss: INTEGER;

END MODULE.
```

A.4.- Implementación del módulo Paciente

```
IMPLEMENTATION MODULE Paciente;

FROM Servicio IMPORT PatologiaObj,ConsultaObj, ServicioObj,ActividadObj,
                     PruebaSObj,DIA,RecursoObj,TIEMPOSIM,DIASSIM;
FROM SimMod IMPORT SimTime,InterruptAll, Timescale, InterruptMethod;
FROM ResMod IMPORT ResourceObj,EntryObj;
FROM GrpMod IMPORT QueueObj;
FROM SimMod IMPORT TriggerObj;
FROM Proced IMPORT DiaSimulacion,DiaHoy;

{ ----------------- objeto paciente --------------------------- }
OBJECT PacienteObj;

        ASK METHOD  InitPaciente(IN ident: INTEGER;IN esp: PatologiaObj; IN cp: QueueObj;
IN qc: INTEGER);
        BEGIN
                id := ident;
                especialidad := esp;
                colapru := cp;
                quecons := qc;
                nextset := 0;
                pruebashechas := 0;
                interrumpido := FALSE;
                NEW(colaset);
        END METHOD;

        ASK METHOD QuitateDeLaCola(IN m: ResourceObj);
        VAR cualquier: EntryObj;
                i: INTEGER;
                p : PacienteObj;
        BEGIN
           IF ASK m.PendingList numberIn > 0
                cualquier := ASK m.PendingList TO First();
                i := 1;
                p := cualquier.Object;
                WHILE ( ((ASK p id) <> id) AND (i < ASK m.PendingList TO numberIn))
                     cualquier := ASK m.PendingList TO Next(cualquier);
                     p := cualquier.Object;
                     i := i + 1;
                END WHILE;
```

```
        IF ((ASK p id) = id)
           ASK m.PendingList TO RemoveThis(cualquier);
        END IF;
   END IF;
END METHOD;


TELL METHOD EsperaPruebas(IN cp: QueueObj);
VAR p: PruebaSObj;
    sync: TriggerObj;
    i,npru : INTEGER; {numero de pruebas que se tiene que hacer }
BEGIN
        NEW(sync); { creamos el sincronizador }
        pruebashechas := 0;
        npru := ASK cp numberIn;
        IF npru > 0
                FOR i := 1 TO npru
                        p := ASK cp TO Remove();
                        TELL SELF TO EsperaPor(p,sync,npru);
                END FOR;
                WAIT FOR sync TO Fire;
                ON INTERRUPT
                     IF ( SimTime() >= TIEMPOSIM)
                       TERMINATE;
                     END IF;
                END WAIT;
        END IF;
        DISPOSE(sync);
END METHOD;



TELL METHOD EsperaPor(IN p: PruebaSObj; IN sync: TriggerObj; IN npru: INTEGER);
VAR
        d,h,m: INTEGER;
        t : REAL;
BEGIN
        t := SimTime();
        DiaHoy(t,d,h,m);
        IF (d <= DIASSIM)
                OUTPUT(">>> Pac ",id," solicita prueba ",ASK p nombre," d= ",d);
                interrumpido := FALSE;
                ASK p TeSolicito(d);
        END IF;
        WAIT FOR SELF TO Consigue(ASK p colact,ASK p duracion,1.0);
                pruebashechas := pruebashechas + 1;
                IF pruebashechas = npru
                        TELL sync TO Trigger;
                END IF;
                IF (d <= DIASSIM )
                  ASK p TeConsegui(d);
                END IF;
        ON INTERRUPT
                IF ( SimTime() >= TIEMPOSIM)
                        TERMINATE;
                END IF;
        END WAIT;
```

```
        END METHOD;



        TELL METHOD SolicitaNumero(IN primconsulta: BOOLEAN);
        VAR prioridad: REAL ;
            consulta : ConsultaObj;
            ser : ServicioObj;
            medcons: ResourceObj;
            cadena : STRING;
            colmed,colaactividades: QueueObj;
            act: ActividadObj;
            dc,d,h,m,tipocons: INTEGER;
        BEGIN
               ser := ASK especialidad servicio;
               consulta := ASK especialidad TO DaConsulta(primconsulta,prioridad);
               { OUTPUT("El pac ",id," solicitó numero con prioridad ",prioridad);}
               colaactividades            :=            ASK            ser            TO
ActividadesConsulta(consulta,"consultorio");
               interrumpido := TRUE;
               IF primconsulta
                      cadena := " (primera) ";
                      tipocons := 0;
                      dc := ASK consulta durconsp;
               ELSE
                      cadena := " (sucesiva) ";
                      tipocons := 1;
                      dc := ASK consulta durconss;
               END IF;
               WHILE interrumpido
                      interrumpido := FALSE;
                      DiaHoy(SimTime(),d,h,m);
                      ASK especialidad TeSolicito(d,tipocons);
                      WAIT FOR SELF TO Consigue(colaactividades,FLOAT(dc),prioridad);
                           { ojo, entra por aqui tanto si se interrumpe }
                           { en consulta como si no }
                           ASK especialidad TeConsegui(d,tipocons);
                      ON INTERRUPT
                           IF ( SimTime() >= TIEMPOSIM)
                             TERMINATE;
                           END IF;
                      END WAIT;
               END WHILE;
               DestruyeCola(colaactividades);
        END METHOD;


        ASK METHOD ObjTerminate;
        BEGIN
           { mensaje de salida de la ejecucion del paciente }
            { OUTPUT("Finaliza pac id=",id," Esp = ",ASK especialidad nombre," Tiempo
",DiaSimulacion(SimTime()));}
        END METHOD;

{ ------------------------------------------------------------- }

        TELL METHOD EjecucionPaciente;
```

```
       VAR numsuc,nc: INTEGER; { numero de consultas sucesivas que se ha de hacer}
           t : REAL;
           colapruebas: QueueObj;
           esprimera,prupendientes : BOOLEAN;
       BEGIN
          { calculo de cuantas sucesivas quedan }
          numsuc := ASK especialidad CuantasSucesivas();
          WHILE  quecons <= numsuc
                esprimera := (quecons = 0);
                prupendientes := (colapru <> NILOBJ);
                IF prupendientes
                   colapruebas := CLONE(colapru);
                   interrumpido := FALSE;
                   WAIT FOR SELF TO EsperaPruebas(colapruebas);
                      prupendientes := FALSE;
                   ON INTERRUPT
                      IF ( SimTime() >= TIEMPOSIM)
                             TERMINATE;
                      END IF;
                   END WAIT;
                END IF;
                { va a entrar por la consulta que corresponda }
                WAIT FOR SELF TO SolicitaNumero(esprimera);
                           OUTPUT("El  ",id,"  con  quecons  =  ",quecons,"  sale  en
",DiaSimulacion(SimTime()));
                    INC(quecons); { incrementa el numero de la consulta }
                    IF esprimera
                      INC(atendidosp);
                    ELSE
                      INC(atendidoss);
                    END IF;
                ON INTERRUPT
                    { reseteo el flag de interrumpido }
                    interrumpido := FALSE;
                    IF ( SimTime() >= TIEMPOSIM)
                             TERMINATE;
                    END IF;
                END WAIT;
               { Espera al dia siguiente para voler a solicitar numero }
               t := SimTime();
               t := DIA *( 1.0 - ((t/DIA) - FLOAT(TRUNC(t/DIA ))));
               WAIT DURATION t;
               ON INTERRUPT
                      IF ( SimTime() >= TIEMPOSIM)
                             TERMINATE;
                      END IF;
               END WAIT;
               { limpio las colas para volver a mandar mas pruebas}
               IF colapru <> NILOBJ
                    DestruyeCola(colapru);
               END IF;
               IF colapruebas <> NILOBJ
                    DestruyeCola(colapruebas);
               END IF;
               { se mandan mas pruebas en funcion de la consulta en la que este }
               colapru := ASK especialidad TO MarcaPruebas(quecons);
```

```
          END WHILE;
          IF colapruebas <> NILOBJ
               DestruyeCola(colapruebas);
          END IF;
          { se quita de la lista de pacientes del sistema ya que finalizo su ejecucion }
          ASK (ASK (ASK especialidad servicio) lispac) TO RemoveThis(SELF);
       END METHOD;


{ ------------------------------------------------------------ }


{ se le pasa el disparador, la actividad que tiene que hacer y el numero total de
recursos que necesita. Si lo consigue disparara el trigger }
TELL METHOD GetRecurso(IN t:TriggerObj; IN act: ActividadObj; IN total:INTEGER; IN
set:SetRecursos;
                       IN aid: ACTID;IN pri: REAL);
VAR q,q2: QueueObj;
    r: ResourceObj;
    rec: RecursoObj;
    aux: ANYOBJ;
    aid2: ACTID;

BEGIN
       aid2 := THISMETHOD;
       r :=  ASK act recurso;
       WAIT FOR r TO PriorityGive(SELF,1,pri);
          q :=  ASK act colrec;
           { hasta el wait sirve para tener en rec el RecursoObj al que le tengo que
notificar
          que lo cogi }
          IF q <> NILOBJ { la actividad tiene cola asignada }
               rec := ASK q TO Remove();
          ELSE
               aux := r;
               rec := aux;
          END IF;

          ASK rec AdquiridoPor(SELF,aid); { para que pueda ser interrumpido}
          INC(set.reccogidos);
          ASK set.cola TO Add(act);
          ASK set.cola TO Add(rec);
          IF (ASK set.colaids TO Includes(aid2))
                 ASK set.colaids TO RemoveThis(aid2);
          END IF;
          IF set.reccogidos = total
              TELL t TO Trigger;
          END IF;
       ON INTERRUPT
          interrumpido := TRUE;
          IF ( SimTime() >= TIEMPOSIM)
               TERMINATE;
          END IF;
          { no tengo que devolverlo ya que se supone que no lo he cogido }
       END WAIT;
END METHOD;
```

```
ASK METHOD DevuelveCogidos(IN colaact: QueueObj);
VAR q,newcolaact: QueueObj;
    n,i,alloc: INTEGER;
    r: ResourceObj;
    rec: RecursoObj;
    aux: ANYOBJ;
    act: ActividadObj;
BEGIN

       IF colaact <> NILOBJ
              newcolaact := CLONE(colaact);
              n := ASK newcolaact numberIn;
              n := n DIV 2;
              FOR i := 1 TO n
               act := ASK newcolaact TO Remove();
                rec := ASK newcolaact TO Remove();
               r :=  ASK act recurso;
               q :=  ASK act colrec;
               alloc := ASK r NumberAllocatedTo(SELF);
               IF  alloc > 0
                     ASK r TakeBack(SELF,alloc);
                     ASK rec TO AdquiridoPor(NILOBJ,NILREC);
                     IF q <> NILOBJ { la actividad tiene cola asignada }
                            ASK q TO Add(rec);
                     END IF;
               END IF;
               END FOR;
               DISPOSE(newcolaact);
       END IF;
END METHOD;


TELL METHOD Consigue(IN colaact: QueueObj; IN duracion: REAL; IN pri:REAL);
VAR
    act: ActividadObj;
    q2,qu,newcolaact: QueueObj;
    rcc,id1,i,n: INTEGER;
    sync: TriggerObj;
    set: SetRecursos;
    aid,aid2: ACTID;


BEGIN

       NEW(set);
       INC(nextset);
       set.idset := nextset;
       NEW(set.cola);
       NEW(set.colaids);
       set.reccogidos := 0;
       ASK colaset TO Add(set);
       interrumpido := FALSE;
       newcolaact := CLONE(colaact);
       n := ASK newcolaact numberIn;
       aid := THISMETHOD;
       NEW(sync); { creo el disparador }
       { lanzo tantos metodos como recursos tengo que coger }
```

```
        FOR i := 1 TO n
          act := ASK newcolaact TO Remove();
          aid2 := TELL SELF TO GetRecurso(sync,act,n,set,aid,pri);
          ASK set.colaids TO Add(aid2); { meto el identificador de aid en la cola }
        END FOR;
        DISPOSE(newcolaact);


        { Aqui espero por el disparador }
        WAIT FOR sync TO Fire;
          { El paciente ya tiene todos los recursos necesarios }
          WAIT DURATION duracion;
              interrumpido := FALSE;
              { Aqui tengo que devolver todos los recursos que he cogido }
              ASK SELF TO DevuelveCogidos(set.cola);
          ON INTERRUPT
              interrumpido := TRUE;
              { Aqui tengo que devolver todos los recursos que he cogido }
              n := ASK set.colaids TO numberIn; {interrumpo los recursos cogidos}
              FOR i := 1 TO n
                aid2 := ASK set.colaids TO Remove();
                InterruptMethod(aid2);
                 WAIT DURATION 0.0
                 ON INTERRUPT
                    TERMINATE;
                 END WAIT;
              END FOR;
              ASK SELF TO DevuelveCogidos(set.cola);
              { nuevo }
              IF ( SimTime() >= TIEMPOSIM)
                TERMINATE;
              END IF;
          END WAIT;
        ON INTERRUPT
          { Le digo que termine todos los procedimientos que esperan por un recurso }
          n := ASK set.colaids TO numberIn; {interrumpo los recursos cogidos}
          FOR i := 1 TO n
                aid2 := ASK set.colaids TO Remove();
                InterruptMethod(aid2);
                 WAIT DURATION 0.0
                 ON INTERRUPT
                    TERMINATE;
                 END WAIT;
           END FOR;
          interrumpido := TRUE;
          { Aqui tengo que devolver todos los recursos que he cogido }
          ASK SELF TO DevuelveCogidos(set.cola);
          { nuevo }
          IF ( SimTime() >= TIEMPOSIM)
                TERMINATE;
           END IF;
        END WAIT;
        DISPOSE(sync);
        ASK colaset TO RemoveThis(set);
        DestruyeCola(set.cola);
        DISPOSE(set);
END METHOD;
```

```
END OBJECT;

{ ------------------------------------------------------------- }


PROCEDURE DestruyeCola(INOUT cola: QueueObj);
VAR a: ANYOBJ;
    i,n: INTEGER;
BEGIN
    IF cola <> NILOBJ
       n := ASK cola numberIn;
       FOR i := 1 TO n
              a := ASK cola Remove();
               { DISPOSE(a); }
       END FOR;
       DISPOSE(cola);
    END IF;
END PROCEDURE;

END MODULE.
```

A.5.- Definición del módulo con los procedimientos para Access.

```
DEFINITION MODULE CAccess;

       PROCEDURE InicializaConexion(); NONMODSIM;
       PROCEDURE CierreConexion(); NONMODSIM;

       PROCEDURE  SQL(IN  consulta:STRING;IN  linea:INTEGER;  IN  columna:INTEGER;INOUT
dato:STRING):INTEGER;

       { este procedimiento ejecuta la consulta y devuelve el resultado.
         lo que devuelve la función es el número de líneas de lan consulta }
       PROCEDURE ExecSQL(IN consulta:STRING; OUT resultado:STRING):INTEGER;
       PROCEDURE WriteSQL(IN consulta:STRING);


       { esta funcion lee el registro nrec de texto en record y el numero de campos
         de ese registro es lo que devuelve. Si no existe ese registro devuelve -1 }
       PROCEDURE ReadRecord(IN texto:STRING;IN nrec:INTEGER;OUT record:STRING):INTEGER;


       { Esta funcion lee el campo nfield de un registro.}
       PROCEDURE ReadField(IN record:STRING; IN nfield: INTEGER; OUT field:STRING);

END MODULE.
```

A.6.- Implementación del módulo con los procedimientos para Access.

```
IMPLEMENTATION MODULE CAccess;
TYPE cadena = FIXED ARRAY [0..10000] OF CHAR;

      PROCEDURE LeeSql(IN nombre:STRING;OUT s:cadena); NONMODSIM;
       PROCEDURE EscribeSql(IN nombre:STRING);NONMODSIM;
```

```
	PROCEDURE LeeLinea(IN texto:STRING; IN linea:INTEGER; OUT s:cadena):INTEGER;
NONMODSIM;
	PROCEDURE LeeCampo(IN texto:STRING; IN campo:INTEGER; OUT s:cadena):INTEGER;
NONMODSIM;
	PROCEDURE NumeroLineas(IN texto:STRING):INTEGER; NONMODSIM;
	PROCEDURE SQL(IN consulta:STRING;IN linea:INTEGER;
			IN columna:INTEGER;INOUT dato:STRING):INTEGER;
	VAR res:INTEGER;
	    s1:cadena;
	    c: STRING;

	BEGIN
		LeeSql(consulta,s1);
		c := CHARTOSTR(s1);
		res := LeeLinea(c,linea,s1);
		 IF (res < 0)
			RETURN res;
		END IF;
		c := CHARTOSTR(s1);
		res := LeeCampo(c,columna,s1);
		dato := CHARTOSTR(s1);
		RETURN res;

	END PROCEDURE;

	{ este procedimiento ejecuta la consulta y devuelve el resultado.
	  lo que devuelve la función es el número de líneas de lan consulta }
	PROCEDURE ExecSQL(IN consulta:STRING; OUT resultado:STRING):INTEGER;
	VAR s1: cadena;
	    c : STRING;
	    n : INTEGER;
	 BEGIN
		LeeSql(consulta,s1);
		c := CHARTOSTR(s1);
		resultado := CHARTOSTR(s1);
		n := NumeroLineas(resultado);
		RETURN(n);
	END PROCEDURE;

	{ este procedimiento ejecuta la consulta y devuelve el resultado.
	  lo que devuelve la función es el número de líneas de lan consulta }
	PROCEDURE WriteSQL(IN consulta:STRING);
	 BEGIN
		EscribeSql(consulta);
	END PROCEDURE;
	{ esta funcion lee el registro nrec de texto en record y el numero de campos de
ese registro es lo que devuelve. Si no existe ese registro devuelve -1 }

	PROCEDURE ReadRecord(IN texto:STRING;IN nrec:INTEGER;OUT record:STRING):INTEGER;
	VAR s1: cadena;
	    n: INTEGER;
	BEGIN
		n := LeeLinea(texto,nrec,s1);
		IF n <> -1
			record := CHARTOSTR(s1);
		END IF;
		RETURN(n);
```

```
        END PROCEDURE;

        PROCEDURE ReadField(IN record:STRING; IN nfield: INTEGER; OUT field:STRING);
        VAR s1: cadena;
            n,i: INTEGER;

        BEGIN
               n := LeeCampo(record,nfield,s1);

               IF n <> -1
                      WHILE s1[i] <> CHR(0)
                             IF s1[i] = CHR(13)
                                    s1[i] := CHR(0);
                             END IF;
                             INC(i);
                      END WHILE;
                      field := CHARTOSTR(s1);
               ELSE
                      field := "";
               END IF;
        END PROCEDURE;


END MODULE.
```

A.7.- Definición del módulo con procedimientos varios.

```
DEFINITION MODULE Proced;

FROM GrpMod IMPORT QueueObj;
FROM ResMod IMPORT ResourceObj;
FROM Servicio IMPORT tipohorario, PatologiaObj,PruebaSObj;

        PROCEDURE DiaSimulacion(IN t:REAL):STRING;
        PROCEDURE DiaHoy(IN t:REAL; INOUT d,h,m:INTEGER);
        PROCEDURE EscribeHorario(IN h: tipohorario);
        PROCEDURE SelectPatologia(IN q: QueueObj; IN nombre: STRING):PatologiaObj;
        PROCEDURE SelectPrueba(IN q: QueueObj;IN name:STRING):PruebaSObj;
        PROCEDURE NumHorasTurno(IN h:tipohorario;IN dia:INTEGER):INTEGER;
        PROCEDURE ImprimeColaPacientes(IN m: ResourceObj);
        PROCEDURE Cierra(IN h: tipohorario; IN i:INTEGER):INTEGER;


END MODULE.
```

A.8.- Implementación del módulo con procedimientos varios.

```
IMPLEMENTATION MODULE Proced;

FROM IOMod IMPORT ReadKey;
FROM ResMod IMPORT ResourceObj, EntryObj;
FROM IOMod IMPORT ReadKey;
FROM Paciente IMPORT PacienteObj;
FROM GrpMod IMPORT QueueObj;
```

```
FROM Servicio IMPORT tipohorario, PatologiaObj,PruebaSObj,DIA,tipolistaturno;


{ este procedimiento calcula el maximo }
PROCEDURE Max(IN a,b:INTEGER):INTEGER;
BEGIN
       IF a > b
              RETURN a
       ELSE
              RETURN b
       END IF;
END PROCEDURE;

{ este procedimiento se le pasa el tiempo de simulacion
  y calcula a que dia de simulacion corresponde }
PROCEDURE DiaSimulacion(IN t:REAL):STRING;
VAR aux,aux1,aux2: REAL;
    d,h,m : INTEGER;
    s: STRING;
BEGIN
       aux := t / DIA; { dias }
       d := TRUNC(aux);
       aux1 := (t - FLOAT(d)*DIA) / 60.0; {horas}
       h := TRUNC(aux1);
       aux2 := t - FLOAT(d)*DIA - FLOAT(h)*60.0;
       m := TRUNC(aux2);

       d := d + 1;
       s := SPRINT (d,h,m) WITH "Dia *****,**:**";
       RETURN(s);
END PROCEDURE;

{ este procedimiento se le pasa el tiempo de simulacion
  y calcula a que dia de simulacion corresponde }
PROCEDURE DiaHoy(IN t:REAL;INOUT d,h,m:INTEGER);
VAR aux,aux1,aux2: REAL;
BEGIN
       aux := t / DIA; { dias }
       d := TRUNC(aux);
       aux1 := (t - FLOAT(d)*DIA) / 60.0; {horas}
       h := TRUNC(aux1);
       aux2 := t - FLOAT(d)*DIA - FLOAT(h)*60.0;
       m := TRUNC(aux2);
       d := d + 1;
END PROCEDURE;

{ este procedimiento dice a que hora cierra definitivamente la consulta especificada
para este índice }
PROCEDURE Cierra(IN h: tipohorario; IN i:INTEGER):INTEGER;
VAR
    p: tipolistaturno;
    last:INTEGER;
BEGIN
       p := h[i];
       WHILE p<> NILREC
           last := p.hfin;
           p := p.sigturno;
```

```
       END WHILE;
       RETURN(last);
END PROCEDURE;


PROCEDURE Salir():BOOLEAN;
VAR c: CHAR;
BEGIN
       REPEAT
            c := ReadKey();
       UNTIL ((c='S') OR (c='s')) OR ((c='n') OR (c='N'));
       IF ((c='S') OR (c='s'))
              RETURN(FALSE);
       ELSE
              RETURN(TRUE);
       END IF;
END PROCEDURE;



PROCEDURE EscribeHorario(IN h: tipohorario);
VAR i: INTEGER;
    p: tipolistaturno;
BEGIN
       OUTPUT("Escribiendo Horario ... ");
       FOR i := 1 TO 7
          OUTPUT(" ----dia---- ",i);
          p := h[i];
          WHILE p<> NILREC
               OUTPUT("[",p.hinicio,",",p.hfin,"]");
                p := p.sigturno;
          END WHILE
       END FOR;
END PROCEDURE;

{ este procedimiento pasado un dia [1..7] dice el numero de horas laborables que tiene
ese dia }
PROCEDURE NumHorasTurno(IN h:tipohorario;IN dia:INTEGER):INTEGER;
VAR total: INTEGER;
    p : tipolistaturno;
BEGIN
       total := 0;
       p := h[dia];
       WHILE  p <> NILREC
              total := total + p.hfin - p.hinicio;
              p := p.sigturno;
       END WHILE;
       RETURN total;
END PROCEDURE;



PROCEDURE ImprimeColaPacientes(IN m: ResourceObj);
VAR cualquier: EntryObj;
    p : PacienteObj;
    i : INTEGER;
    tecla: CHAR;
```

```
BEGIN
        cualquier := NILOBJ;
        FOR i := 1 TO (ASK m.PendingList numberIn)
          IF cualquier = NILOBJ
              cualquier := ASK m.PendingList TO First();
          ELSE
              cualquier := ASK m.PendingList TO Next(cualquier);
          END IF;
          p := cualquier.Object;
          OUTPUT("[[",ASK p TO id,"]]");
        END FOR;
        tecla := ReadKey;
END PROCEDURE;

        { Procedimiento que selecciona una patologia de una cola de ellas }
        PROCEDURE SelectPatologia(IN q: QueueObj; IN nombre: STRING):PatologiaObj;
        VAR p,r: PatologiaObj;
            i,sel: INTEGER;
        BEGIN
               r := NILOBJ;
               FOR i := 1 TO (ASK q numberIn)
                  IF i <> 1
                   p := ASK q TO Next(p);
                   ELSE
                     p := ASK q TO First();
                  END IF;
                  IF (ASK p nombre) = nombre
                      r := p;
                  END IF;
               END FOR;
               RETURN(r);
        END PROCEDURE;

{ ------------------------------------------------------------------ }

        { Procedimiento que devuelve un objeto prueba que tenga el nombre que se le pasa
}
        PROCEDURE SelectPrueba(IN q: QueueObj;IN name:STRING):PruebaSObj;
        VAR i,n: INTEGER;
            p : PruebaSObj;
            found : BOOLEAN;
        BEGIN
               n := ASK q TO numberIn;
               found := FALSE;
               i := 1;
               WHILE ( (NOT found) AND (i <= n))
                    IF i = 1
                      p := ASK q TO First();
                    ELSE
                      p := ASK q TO Next(p);
                    END IF;
                    IF (ASK p nombre = name)
                              found := TRUE;
                    END IF;
                    INC(i);
               END WHILE;
               IF NOT found
```

```
                    p := NILOBJ;
                END IF;
                RETURN p;
        END PROCEDURE;

END MODULE;
```

APÉNDICE B

APÉNDICE B

A continuación se presenta diferentes procedimientos realizados en Matlab que han sido necesarios durante todo el trabajo.

B.1.- Controlpi.m

```
%function sesion=control(e1,e2,c)
%sesion=control(error1,error2,consigna)
l=size(error1);
l2=size(vecdiacontrol1);
horas1=round(kp*(error1(:,l(2))));
if(l2(2)>1)
horas1=horas1+kd*((error1(:,l(2))-error1(:,vecdiacontrol1(:,l2(2))))
/(vecdiacontrol1(:,l2(2))-vecdiacontrol1(:,l2(2)-1)));
elseif (l2(2)==1)
horas1=horas1+kd*(error1(:,l(2))-error1(:,vecdiacontrol1(:,l2(2))) );
end;
if horas1(1)<0
  horas1(1)=21;
end;

l=size(error2);
l2=size(vecdiacontrol2);
horas2=round(kp2*(error2(:,l(2))));
if(l2(2)>1)
horas2=horas2+kd*((error2(:,l(2))-error2(:,vecdiacontrol2(:,l2(2))))
/(vecdiacontrol2(:,l2(2))-vecdiacontrol2(:,l2(2)-1)));
elseif (l2(2)==1)
horas2=horas2+kd*(error2(:,l(2))-error2(:,vecdiacontrol2(:,l2(2))) );
end;
if horas2(1)<0
  horas2(1)=21;
end;


l=size(vechoras1);
```

```
vechoras1(:,l(2)+1)=horas1;

l=size(vechoras2);
vechoras2(:,l(2)+1)=horas2;
```

B.2.- Controlpid.m

```
%function sesion=control(e1,e2,c)
%sesion=control(error1,error2,consigna)
  l=size(error2);
  ld=size(vecdia);
  ls=size(sal2);
  estant2=sal2(1,ls(2));
  l2=size(vecdiacontrol2);
  vecdiacontrol2(:,l2(2)+1)=vecdia(ld(2));
  horas2=round(kp2*(error2(:,vecdia(ld(2)))));
  if(l2(2)>1)
   paso=(vecdia(ld(2))-vecdiacontrol2(:,l2(2)));
   if paso>0
    horas2=horas2+kd2*((sal2(:,vecdia(ld(2)))-sal2(:,vecdiacontrol2(:,l2(2)))) /paso);
   end;
  elseif (l2(2)==1)
    horas2=horas2+kd2*(sal2(:,vecdia(ld(2)))-sal2(:,vecdiacontrol2(:,l2(2))) );
  end;
  if integral2==1
    horas2=horas2+ki2*suma2;
  end;
  lh=size(horas2);
  if(lh(2)>0)
   if horas2(1)<0
     horas2(1)=21;
   end;
  end;

  l=size(vechoras2);
  vechoras2(:,l(2)+1)=horas2;
```

B.3.- Control.m

```
%function sesion=control(e1,e2,c)
%sesion=control(error1,error2,consigna)

l=size(error2);
l2=size(vecdiacontrol2);
horas2=round(kp2*(error2(:,diacontrol2)));
if(l2(2)>1)
horas2=horas2+kd*((error2(:,diacontrol2)-error2(:,diacontrol2))           /(diacontrol2-
vecdiacontrol2(:,l2(2))));
elseif (l2(2)==1)
horas2=horas2+kd*(error2(:,diacontrol2)-error2(:,diacontrol2) );
end;
if horas2(1)<0
  horas2(1)=21;
end;
```

B.4.- Tipo1.m

```
l=size(sal1);
h1=1;
 if (sal1(1,l(2))>60)
   contpdi1;
   variacion1=30;
 elseif (sal1(1,l(2))>40)
   contpdi1;
   variacion1=20;
 elseif (sal1(1,l(2))>20)
    contpdi1;
   %contri1;
   variacion1=5;
 else
   contpdi1;
   variacion1=5;
 end;
```

B.5.- Tipo2.m

```
l=size(sal2);
h2=1;
 if (sal2(1,l(2))>40)
   integral2=0;
   contpdi2;
   variacion2=15;
 elseif (sal2(1,l(2))>30)
   contpdi2;
   variacion2=10;
 elseif (sal2(1,l(2))>20)
    integral2=1;
    contpdi2;
%   contri2;
   variacion2=10;
 else
   integral2=0;
   contpdi2;
   variacion2=5;
 end;
```

B.6.- Funnac.m

```
%vamos a hallar para cada acción de control que se entra en el fichero nombre
%una funcio = media aritmética/media geométrica, que proporciona una medida relativa
%de como se diferencian los valores

nombre;
ln=size(n);
vfun1a=zeros(1,ln(1));
vfun1b=zeros(1,ln(1));
vfun1c=zeros(1,ln(1));
vfun2a=zeros(1,ln(1));
vfun2b=zeros(1,ln(1));
```

```
vfun2c=zeros(1,ln(1));
for i=1:ln(1)
  v=['load ' n(i,:)];
  eval(v);
  indic=indic+1

estfin1;
l=size(vecdiacontrol1);
c=0;
for indcont=1:l(2)
 if (vecdiacontrol1(indcont) < diaesta1)
        c=c+1;
        if(vechoras1(1,indcont)<(1*commedio1))
            vfun1a(indic)=vfun1a(indic)+1;
       elseif (vechoras1(1,indcont)<(3*commedio1))
             vfun1b(indic)=vfun1b(indic)+1;
        else
             vfun1c(indic)=vfun1c(indic)+1;
        end;
 end
end
       vfun1a(indic)=vfun1a(indic)/c;
       vfun1b(indic)=vfun1b(indic)/c;
       vfun1c(indic)=vfun1c(indic)/c;


estfin2;
l=size(vecdiacontrol2);
c=0;
for indcont=1:l(2)
 if (vecdiacontrol2(indcont) < diaesta2)
        c=c+1;
        if(vechoras2(1,indcont)<(1*commedio2))
            vfun2a(indic)=vfun2a(indic)+1;
       elseif (vechoras2(1,indcont)<(3*commedio2))
             vfun2b(indic)=vfun2b(indic)+1;
        else
             vfun2c(indic)=vfun2c(indic)+1;
        end;
 end
end
        vfun2a(indic)=vfun2a(indic)/c;
        vfun2b(indic)=vfun2b(indic)/c;
        vfun2c(indic)=vfun2c(indic)/c;

end;
```

BIBLIOGRAFÍA

[Aben 1993] Manfred Aben, *CommonKADS Inferences*, Esprit P5248 KADS-II/M2/TR/UvA/041/1.0, University of Amsterdam, 1993.

[Aben 1995] M. Aben, *Formal Methods in Knowledge Engineering*, PhD thesis, SWI,Universidad de Amsterdam, 1995.

[Aguilar 1994] R. M. Aguilar, *Sistema Experto para la Extracción Automática de Características del Potencial Evocado Visual.* Memoria de Licenciatura. Universidad de La Laguna. 1994.

[Aracil 1986a] J. Aracil, *Introducción a la dinámica de sistemas*, 3ª edición, Alianza Universidad Textos, 1986.

[Aracil 1986b] J. Aracil, *Máquinas, Sistemas y Modelos*, Tecnos, 1986.

[Athans 1966] M. Athans, P.L. Falb, *Optimal Control: An Introduction to the Theory and Its Applications*. McGraw-Hill. 1966

[Azoff 1994] E. M. Azoff, *Neural Network. Time Series Forecasting of Finacial Markets*, John Wiley & Sons, 1994.

[Beltrán 1997] J.L. Beltrán Aguirre, *Legislación sobre Sanidad*, 2ª edición, Editorial Tecnos, 1997.

[Benjamins 1995] V.R. Benjamins, Problem-solving methods for diagnosis and their role in knowledge acquisition. *International Journal of Expert Systems: Research and Applications*, vol 2, nº 8, pp. 93-120, 1995

[Boy 1991] G. A. Boy, *Intelligent Assistant Systems*, Academic Press. 1991.

[Brams 1986a] G.W. Brams, *Las Redes de Petri. Teoría y Práctica. Teoría y Análisis.* Tomo 1, Masson, 1986

[Brams 1986b] G.W. Brams, *Las Redes de Petri. Teoría y Práctica. Modelización y Aplicaciones*. Tomo 2, Masson, 1986

[Breuker 1994] J. Breuker, W. Van de Velde, *CommonKADS Library for Expertise Modeling*, IOS Press, 1994.

[Bylander 1988] T. Bylander, B. Chandrasekaran, Generic Task in Knowledge-Based Reasoning. The Right Level of Abstraction for Knowledge Acquisition. *Knowledge Acquisition for Knowledge-Based Systems*, vol I, pp. 65-77, Academic Press, 1988.

[CACI 1995a] *Modsim II. The Language for Object-Oriented Programming. Reference Manual,* CACI Products

Company, 1995.

[CACI 1995b] *Simgraphics II. User's Manual for Modsim II*, CACI Products Company, 1995.

[Cañamero 1996] D. Cañamero, Plan Recognition for Decision Support, *IEEE Expert*, Vol. 11, nº 3, pp. 54-63, 1994.

[Coelho 1995] E. Coelho, G. Lapalme, Extending Ontolingua for Representing Control Knowledge. In *IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[Coelho 1996a] E. Coelho, G. Lapalme, V. Patel, From KADS Models to Operational Problem-Solving Methods: Perspectivaes in the Domain View, *6th Workshop on Knowledge Engineering: Methods & Languages*, 1995.

[Coelho 1996b] E. Coelho, G. Lapalme, Describing reusable problem-solving methods with a method ontology. *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp. 3.1-3.20, SRDG Publications, University of Calgary, 1996.

[Chandrasekaran 1983] B. Chandrasekaran, Towards a Taxonomy of Problem Solving Types, *AI Magazine*, Vol. 1, nº4, pp. 9-17, 1983.

[Chandrasekaran 1990] B. Chandrasekaran , Desing problem solving: a task analysis, *AI Magazine*, p.p. 59-71, 1990.

[Davis 1982] R. Davis, D.B. Lenat, TEIRESIAS: Applications of meta-level knowledge, (Ed) *Knowledge-Based System in Artificial Intelligence*, McGraw-Hill, pp. 229-461, 1982.

[Dekkers 1995] G.J.M. Dekkers, Including Macro econometric Models in Microsimulation Models: the case of NEDYMAS, *Nordic Seminar on Microsimulation Models*, 1995.

[Delaney 1989] W. Delaney, E. Vaccari, *Dynamic Models and Discrete Event Simulation*, Marcel Dekker, 1989.

[Durkin 1996] J. Durkin, Expert Systems: A View of the Field, *IEEE Expert*, Vol. 11, nº 2, pp. 56-63, 1996.

[Fishwick 1995] P. A. Fishwick, *Simulation Model Design and Execution. Building Digital Worlds*, Prentice Hall, 1995.

[Fortmann 1977] T.E. Fortmann, K L. Hitz, *An Introduction to Linear Control Systems*, Marcel Dekker, 1977.

[Fowler 1997] M. Fowler, *Analysis Patterns: Reusable Object Models*, Addison Wesley, 1997.

[Franklin 1990] G.F. Franklin, J.D. Powell, M.L. *Workman: Digital Control of Dynamic Systems*. 2nd Edition. Addison-Wesley. 1990.

[Freeman 1991] J. A. Freeman, D. M. Skapura, *Redes Neuronales. Algoritmos, aplicaciones y técnicas de programación*, Addison-Wesley/Díaz de Santos, 1991.

[Fu 1988] K.S. Fu, R.C. González, C.S. Lee, *Robótica: Control, detección, visión e inteligencia*, McGraw-Hill, 1988

[Garbe 1991] W.Garbe, *Visual Simnet 1.32*, 1991.

[Genesereth 1992] M. Genesereth, R. Fikes, *Knowledge Interchange Format, versión 3.0*, Reference Manual, Computer Science Department, Stanford University, 1992.

[Gilbert 1993] G.N. Gilbert, J. Doran, *Simulating societies: the computer simulation of social processes*, UCL Press, 1993.

[Gordon 1978] G. Gordon, *System Simulation*, 2nd edition, Prentice Hall, 1978.

[Gregory 1995] R.L. Gregory (Editor), *Diccionario Oxford de la Mente*, *Alianza Editorial*, 1995.

[Gruber 1992] T. Gruber, *Ontolingua: A mechanism to support portable ontologies*. Technical Report KSL-91-66, Stanford University, 1992.

[Gruber 1993a] T. Gruber, *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, Technical Report KSL-93-04, Stanford University, 1993.

[Gruber 1993b] T. Gruber, A Translation Approach to Portable Ontology Specifications, *Knowledge Adquisition*, vol. 5, nº2, pp. 199-220, 1993.

[Gruber 1994] T. Gruber, G. Olsen, *Theory Component-assemblies*. Available in ftp://ksl.stanford.edu/pub/knowledge-sharing/ontologies/

[Guadalajara 1994] N. Guadalajara, *Análisis de Costes en los Hospitales*, M/C/Q Ediciones, 1994.

[Hama 1992] T. Hama, M. Hori, Y. Nakamura, Task-Specific Language Constructs for Describing Constrints in Job Assignment Problems. IBM Research Report RT0084. Available in ftp://ksl.stanford.edu/pub/knowledge-sharing/ontologies/

[Hammer 1993] M. Hammer, J. Champy, *Reengineering the Corporation. A Manifiesto for Business Revolution*, HarperCollins Publisher, 1993.

[Harding 1990] A. Harding, *Dynamic microsimulation models:problems and prospects*, London School of Economics, 1990.

[Harmon 1988] P. Harmon, D. King, *Sistemas Expertos. Aplicaciones de la inteligencia artificial en la actividad empresarial*, Ediciones Díaz de Santos, 1988.

[Hoog 1994] R. Hoog, R. Martil, B. Wielinga, R. Taylor, C. Bright, W. Van de Velde, *The Common KADS model set*, Esprit P5248 KADS-II/M1/DM1.1b/UvA/018/6.0/FINAL, University of Amsterdam, 1994.

[Jacobson 1992] I. Jacobson, M. Christerson, P. Jonsson, G.Övergaard, *Object-Oriented Software Engineering. A Use Case Driven Approach*, Addison-Wesley, 1992.

[Jamshidi 1983] M. Jamshidi, *Large-Scale System. Modeling and Control*, North-Holland, 1983.

[Kline 1989] P. Kline, S. Dolins, *Designing Expert System. A Guide to Selecting Implementation Techniques*, John Wiley & Sons, 1989.

[Langton 1995] Langton, Christopher, N.Minar, R.Burkhart, *The Swarm Simulation System: A Tool for Studying Complex Systems*, Santa Fe Institute, www.santafe.edu/projects/swarm

[Larson 1978] R.E. Larson, J.L. Casti, *Principles of Dynamic Programming. Part I. Basic Analytic and Computational Methods*. Marcel Dekker. 1978.

[Larson 1982] R.E. Larson, J.L. Cast, *Principles of Dynamic Programming. Part II. Advanced Theory and Applications*. Marcel Dekker. 1982.

[Lewis 1986] F.L. Lewis, *Optimal Control.* John Wiley & Sons. 1986.

[Luenberger 1979] D.G. Luenberger, *Introduction to Dynamic Systems. Theory, Models and Applications*. John Wiley & Sons. 1979.

[Maes 1988] P. Maes, D. Nardi, *Meta Level Architectures and Reflection*, (Editor), 1988.

[Matko 1992] D. Matko, B. Zupancic, R. Karba, *Simulation and Modelling of Continuous Systems. A Case Study Approach*, Prentice Hall, 1992.

[McCorduck 1979] P. McCorduck, *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*, Freeman & Co, 1979.

[McHaney 1991] R. McHaney, *Computer Simulation. A Practical Perspective*, Academic Press, 1991.

[Molina 1996] M. Molina, Y. Shahar, J. Cuena, M.A. Musen, A structure of problem-solving methods for real-time decision support: Modeling approaches using protégé-ii and ksm. *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, pp. 8.1-8.20, SRDG

Publications, University of Calgary, 1996.

[Moreno 1995] L. Moreno, J.D. Piñeiro, R.M. Aguilar, J.L. Sánchez, S. Mañas, J. J. Merino, J. I.Estévez, Knowledge based System for Diagnosis of Pathologies observable in Brain Signals, *IEEE International Conference on Systems, Man and Cybernetics*,vol 3, pp.2840-2843, Conference proceedings, 1995.

[Moreno 1996a] L.Moreno, J.D.Piñeiro, R.M.Aguilar, C.A.Martín., J.L.Sánchez, V.I.Jiménez, M.A.González**,** B.González, S.Rodríguez, Simulation as a Knowledge Acquisition Tool in Hospital Management, *IASTED International Conference Modelling, Simulation and Optimization*, , Conference proceedings, 1996.

[Moreno 1996b] L.Moreno, V.I.Jiménez, R.M.Aguilar, C.A.Martín., M.A.González**,** J.D.Piñeiro, B.González, S.Rodríguez, J.L.Sánchez, Modelling and Simulation in Hospital Management, *International Conference on Information Systems. Analysis and Synthesis*, Proceedings, pp.446-452, 1996.

[Moreno 1997a] L.Moreno, R.M.Aguilar, C.A.Martín., J.D.Piñeiro, Jiménez V.I., M.A.González**,** B.González, S.Rodríguez, A Knowledge-Based System To Aid Hospital Management, *Systemics, Cibernetics and Informatics*,vol 3, pp.438-445, Conference proceedings, 1997.

[Moreno 1997b] L.Moreno, R.M.Aguilar, C.A.Martín., J.D.Piñeiro, J.L.Sánchez, V.I.Jiménez, M.A.González**,** B.González, S.Rodríguez, Modelling and Simulation in management decision making, *Simulation in the Medical Sciences Conference*, The Society for Computer Simulation International, pp.43-48, 1997.

[Moreno 1998] L. Moreno, R.M. Aguilar, C.A. Martín ,J.D. Piñeiro, J.F. Sigut, A.F. Hamilton, J.L. Estévez, An Automated System to Aid Decision Making in the Hospital Management, *Proceedings of the International ICSC Symposium on Engineering of Intelligent System*, vol. 3, pp. 203-209, ICSC Academic Press, University of La Laguna, 1998.

[Moreno 1998b] L. Moreno, R.M. Aguilar, C.A. Martín ,J.D. Piñeiro, J.F. Sigut, A.F. Hamilton, J.L. Estévez, J.L. Sánchez, V.I. Jiménez, Patient-Centered Computer Simulation in Hospital Management, *Journal of Network and Computer Applications*, Academic Press, en segunda revisión.

[Musen 1994] M. Musen, J. Gennari, H. Eriksson, S. Tu, A. Puerta, *PROTÉGÉ-II: Computer Support for Development of Intelligent Systems From Libraries of Components.* Technical Report KSL-94-60, Stanfor University, 1994.

[Newell 1982] A. Newell, The Knowledge Level, *Artificial Intelligence*, 18, pp. 82-127, 1982.

[Oñorbe 1997] M. Oñorbe, Salud pública hoy y mañana, *Revista de Administración Sanitaria*, Vol. 1, nº2, 1997.

[Penrose 1989] R. Penrose, *The Emperor's New Mind*, Oxford University Press. 1989. Edición en castellano en *La Nueva Mente del Emperador*, Mondadori España, 1991.

[Pigford 1990] D.V. Pigford, G. Baur, *Expert Systems for Business. Concepts and Applications*, Boyd & Fraser Publishing Company, 1990.

[Pineault 1989] R. Pineault, C. Daveluy, *La Planificación Sanitaria. Conceptos, Métodos, Estrategias*; 2nd edición, Masson, 1989

[Piñeiro 1995] J.D. Piñeiro, J.L. Sánchez, S. Mañas, R.M. Aguilar, J. I. Estévez, L. Moreno, Sistema Basado en el Conocimiento para el Estudio de Señales Cerebrales, *I Jornadas de Informática*, Actas, Asociación Española de Informática y Automática, pp. 251-255, 1995.

[Piñeiro 1996] J. D. Piñeiro, *Hacia un Sistema Basado en el Conocimiento para el Diagnóstico de Patologías en Señales Cerebrales, Tesis Doctoral*, Universidad de La Laguna, 1996.

[Piñeiro 1998] Piñeiro J.D., Sigut J.F., Moreno L., Estévez J.I., Aguilar R.M., Sánchez J.L., Mañas S., Merino J.J., A knowledge based system for aiding in the diagnosis in Neurophysiology, *Proceedings of the International ICSC Symposium on Engineering of Intelligent System*, vol. 3, pp. 198-202, ICSC Academic Press, University of La Laguna, 1998.

[Post 1997] W. Post, B. Wielinga, R. Hoog, G. Schreiber, Organizational Modeling in CommonKADS: The Emergency Medical Service, *IEEE Expert*, Vol. 12, nº 6, pp. 46-52, 1997.

[Pressman 1988] R. S. Pressman, *Ingenieria del Software: Un enfoque práctico*, McGraw-Hill, 1988.

[Rich 1991] E. Rich, K. Knight, *Artificial Intelligence*, 2nd Edition, McGraw-Hill, 1991.

[Romay 1997] J.M. Romay, La Sanidad Española, presente y futuro, *Revista de Administración Sanitaria*, Vol. 1, nº1, 1997.

[Russell 1995] S.J. Russell, P. Norvig, *Artificial Intelligence*, Prentice Hall, 1995.

[Sánchez 1998] Sánchez J.L., Estévez J.I., Moreno L., Mañas S., Merino J.J., Sigut J.F., Piñeiro J.D., Aguilar R.M., Design of a system based on fuzzy logic for the detection of EEG Morphologies, *Proceedings of the International ICSC Symposium on Engineering of Intelligent System*, vol. 1, pp. 127-133, ICSC Academic Press, University of La

Laguna, 1998.

[Schreiber 1993] G. Schreiber, B. Wielinga, J. Breuker, *KADS. A Principled Approach to Knowledge-Based System Development*, Academic Press, 1993.

[Schreiber 1994] G. Schreiber, B. Wielinga, R. Hoog, CommonKads: A Comprehensive Methodology for KBS Development, *IEEE Expert*, Vol. 9, nº 6, pp. 28-37, 1994.

[Shortliffe 1976] E. Shortliffe, *Computer Based Medical Consultations: Mycin*, 1976.

[Sierra 1995] G.J. Sierra, E. Bonsón, C.Nuñez, M. Orta, *Sistemas Expertos en Contabilidad y Administración de Empresa. Desarrollo de aplicaciones usando Crystal*, Ra-ma, 1995.

[Silva 1985] M. Silva, *Las Redes de Petri: en la Automática y la Informática*, Ed. AC, 1985.

[Silva 1998] D. Silva, J. Garrido, L.A. Oteo, Bases Conceptuales en la empresa moderna. Experiencia de innovación en el sector sanitario, *Revista de Administración Sanitaria*, Vol. 2, nº5, 1998.

[Simon 1995] H. Simon, Artificial intelligence: an empirical science, *Artificial Intelligence*, 77, pp. 95-127, 1995

[Turing 1950] A.M. Turing, *Computing Machinery and Intelligence Mind*, 59, pp. 433-460, 1950.

[Unger 1993] B.W. Unger, J.G. Cleary, Practical Parallel Discrete Event Simulation, *ORSA Journal on Computing*, vol. 3, nº5, pp. 242-244, 1993.

[Valette 1994] R. Valette (Editor), *Lecture Notes in Computer Science. Application and Theory of Petri Nets*, Spring-Verlag, 1994.

[Waldrop 1992] M.Waldrop, *Complexity: the emerging science at the edge of chaos*, Simon&Schuster, 1992.

[Wasserman 1989] P.D. Wasserman, *Neural Computing. Theory and Practice*, Van Nostrand Reinhold, 1989

[Weiss 1984] S. Weiss, C. Kulikowski, *A Practical Guide to Designing Expert Systems*, Rowman & Allanhed, 1984.

[Wielinga 1994] B. Wielinga, H. Akkermas, H. Hassan, O. Olsson, K. Orsvärn, G. Schreiber, P. Terpstra, W. Van de Velde, S. Wells, *Expertise model definition document*, Esprit P5248 KADS-II/M2/UvA/026/5.0, University of Amsterdam, 1994.

[Wiener 1965] N.Wiener, *Cybernetics or Control and Communication in the Animal*, 2nd edition, MIT Press, 1965.

[Winston 1992] P.H. Winston, *Artificial Intelligence*, Addison-Wesley, 1992.