UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CONTADURIA Y ADMINISTRACION

APLICACIÓN CLIENTE/SERVIDOR INTELIGENTE ORIENTADA A OBJETOS COMO ESTRATEGIA DE EVACUACIÓN EN CASO DE INCENDIO EN UN EDIFICIO INTELIGENTE

SEMINARIO DE INVESTIGACION INFORMATICA
QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN INFORMATICA

P R E S E N T A:

HUGO JÁUREGUI ESPINOSA

ASESOR DEL SEMINARIO:

C.P. Y M.C.C. MARINA TORIZ GARCÍA

MEXICO, D.F. **1998**

Agradecimientos.

Valga este agradecimiento como un minúsculo homenaje para todos aquellos que han hecho de mi vida y estudios una experiencia gratísima.

Quisiera agradecer en primer lugar a mis padres y a mi familia por su inalterable amor, que en todo momento tuve por seguro.

A la Universidad Nacional Autónoma de México por el regalo más trascendente de mi vida, el ensañarme a ser un universitario digno, algo que jamas podré dejar de ser.

A todos mis profesores por su dedicación y conocimientos, por enseñarme con el ejemplo la nobleza del amor y compromiso con la Universidad y la sociedad a la que ésta y sus egresados se deben.

A mi asesora, por su tiempo y sincero interés.

Al personal de la Dirección de Cómputo para la Administración Académica, por el apoyo y amistad que me brindaron durante toda mi estadía con ellos.

A mis amigos por su afecto y lealtad constante, por ser una segunda familia con la que siempre he contado.

Finalmente, gracias a Lulú por su incondicional amor y confianza que trasciende los límites de la esperanza, por enseñarme que los principios son más un compromiso con el espíritu que con la conciencia.

ÍNDICE.

INTRODUCCIÓN.

La Inteligencia Artificial (IA) ha sido desde sus orígenes un tema que ha causado controversia e interés. El hombre, desde que adquirió conciencia de si mismo y de su entorno, se fue acostumbrando a la posición que la naturaleza le había deparado: en verdad no era el ser más rápido, tampoco era el más fuerte, ni siquiera era el más grande, sin embargo si era el más inteligente, o al menos era el ser cuyo tipo de inteligencia resultaba ser el más eficiente. Gracias a lo que él llamó inteligencia podía realizar abstracciones con una mayor profundidad que el resto de los animales, podía desarrollar una comunicación oral y escrita que le permitió evolucionar en sus procesos de pensamiento, con los cuales pudo tomar conciencia de sí mismo. De esta forma la inteligencia, aun sin haber sido exactamente definida paso a ser la característica por excelencia del ser humano. Tal vez la mayoría de los hombres no sabían a ciencia cierta a que se referían cuando hablaban de inteligencia, pero sabían que ese concepto de tan amplio espectro reflejaba la más grande posesión del ser humano. El hombre no tardaría en considerarse como el único ser que tenía y que podría ser poseedor de un don tan valioso.

Pero el tiempo paso, las teorías evolucionaron y la inteligencia fue objeto de estudio. En un principio, el hombre, luego de un esfuerzo supremo, tuvo que aceptar que los animales también poseían algún grado de inteligencia, no era de la misma clase que la del hombre, ni siquiera parecía que tuviese la posibilidad de evolucionar hasta el grado de evolución a la que había llegado la del hombre, pero era inteligencia. Los animales también conocían y comprendían, también podían hacer abstracciones aunque fuesen rudas y forzadas. El patrimonio del hombre ya no era exclusivamente suyo. Sin embargo, el hombre también terminó adaptándose a esa nueva situación, a fin de cuentas, los animales, por mucha inteligencia que poseyesen jamas lograrían dar alcance al hombre, éste les llevaba toda una evolución de ventaja.

Pero repentinamente llegó al hombre común una nueva noción, una idea que se había empezado a forjar hacía ya bastante tiempo por fin tenía visos de convertirse en una apabullante verdad: no solo los seres vivos podían ser inteligentes. Empezaron a surgir teorías, modelos, conceptos que sentarían las bases de una nueva disciplina, la Inteligencia Artificial.

En unas cuantas décadas el hombre vio aparecer las primeras máquinas que algún sentido podían considerarse que presentaban características de inteligencia. La Inteligencia Artificial hizo su entrada en el mundo del hombre y desde entonces su influencia ha sido cada vez más profunda. El hombre se escudo entonces en la ambigüedad del concepto de inteligencia, amparado por la dispersión de las teorías, conceptos y objetivos de los investigadores de la Inteligencia Artificial. Una máquina jamas podría ser considerada cabalmente inteligente, una máquina esta condenada, y al menos todo parece indicarlo por ahora, a nunca lograr la conciencia sobre sí misma, aunque esto no sea algo que acepten los investigadores de la llamada "Inteligencia Artificial dura".

Sin embargo, más allá de las consideraciones de carácter filosófico y psicológico que envuelven la gran controversia de la Inteligencia Artificial, ésta ha conseguido logros que parecían irrealizables fuera de la imaginación de los escritores de ciencia ficción. Algunos de los productos de los investigadores de la Inteligencia Artificial "blanda", que se ocupa más en obtener resultados que de discusiones que tienen un poco de bizantinas, en verdad nos parece que tienen inteligencia. Tal vez no tengan ni la más remota idea de lo que están haciendo, pero realizan labores que consideramos que requieren siquiera de un mínimo de razonamiento y experiencia. E incluso, algunas de ellas han logrado hacer cosas que la mayoría de nosotros no hemos ni soñado en lograr por exceder nuestras capacidades intelectuales. Hace poco tiempo, una computadora conocida como Deep Blue venció al campeón mundial de ajedrez, el cual es además uno de los mejores jugadores de todos los tiempos, Gary Kasparov. El ajedrez hasta ese día había sido considerado como un juego, que en virtud de su alta exigencia de facultades mentales, era un dominio único

del ser humano; es verdad que ya antes habían perdido ante computadoras grandes maestros, pero nunca un campeón mundial.

Es en ese entorno en el que han surgido los Edificios Inteligentes. A tal grado se ha extendido la utilización de la Inteligencia Artificial en la vida diaria que ahora estamos siendo testigos del inicio de lo que en un futuro no será sino algo completamente natural; edificaciones que puedan controlar de manera óptima y por si mismas sus recursos, de manera tal que ayuden a potenciar el desempeño humano a través de proporcionar un ambiente seguro, ergonómico, confiable y, por que no decirlo, que "comprenda" las necesidades propias y las de sus ocupantes.

El objetivo de esta tesis es demostrar la factibilidad de utilización y modelar algunas de las características de los sistemas inteligentes encargados de apoyar la toma de decisiones en caso de emergencia dentro de un Edificio Inteligente, mediante el diseño e implementación de un Agente Inteligente que utilizando una base de conocimientos y técnicas de inteligencia artificial se encarga de solucionar el problema de encontrar una ruta de evacuación segura para las personas que se encuentren en un Edificio Inteligente en caso de incendio. No se pretende determinar si en verdad el adjetivo inteligente merece ser utilizado o no a programas de software o a construcciones, lo cual es una cuestión cuyo fin no se ve cercano, sino que utilizaremos dicho adjetivo con una connotación lo más formalmente definida que sea posible.

En el primer capítulo se hace una breve introducción a los conceptos que fueron utilizados para el desarrollo e implementación del Agente Inteligente. El segundo capítulo es una introducción a los Edificios Inteligentes. En el tercer capítulo se hace una descripción más detallada de los elementos que conforman los Edificios Inteligentes. Finalmente, en el cuarto capítulo se explica el proceso de diseño e implementación del Agente Inteligente. En el Apéndice A se encuentra el código de generación de la base de datos que se utiliza para modelar al Edificio Inteligente. En el Apéndice B se encuentra el código del Agente Inteligente y de las bibliotecas que éste utiliza. En el Apéndice C esta la descripción de los campos de la base de datos y en el Apéndice D están las pantallas del Agente Inteligente durante una corrida desde Unix.

CAPÍTULO I. FUNDAMENTOS.

1.1. INTELIGENCIA ARTIFICIAL.

1.1.1. Definición.

Los malos entendidos han acompañado a la Inteligencia Artificial, también conocida por su abreviatura IA o AI (por sus siglas en inglés), desde sus orígenes. La Inteligencia Artificial, disciplina de las Ciencias Computacionales, ha arrastrado uno de los nombres más imprecisos en todos los dominios tecnológicos: en efecto, actualmente no se cuenta con definiciones suficientemente precisas de la "inteligencia", y al agregarle el adjetivo "artificial" solamente se aumenta la confusión. Varios años de polémica al respecto no han contribuido a clarificar la cuestión; en cambio, se han delineado varias corrientes de pensamiento o enfoques. No obstante lo anterior, la mayoría de estas corrientes varían en torno a dos dimensiones principales:

- Los que ven a la Inteligencia Artificial como un intento de reproducir los procesos mentales y el razonamiento. Dentro de este enfoque hay trabajos que tratan de modelar la inteligencia tal como se produce en el ser humano (por ejemplo, mediante la interconexión de neuronas) [KANDEL81, ARBIB72].

- Los que utilizan técnicas desarrolladas en el ámbito de la Inteligencia Artificial para reproducir un cierto tipo de conducta. Es decir, se trata de reproducir el comportamiento inteligente, independientemente de los mecanismos que producen dicho comportamiento [Dehn] .

A su vez, ambas se pueden dividir en otro par de enfoques:

- Los que evalúan su desempeño en función de la eficiencia humana.

- Los que lo hacen de conformidad con un concepto de inteligencia ideal, denominado racionalidad. Se considera que un sistema es racional si hace lo correcto.

De acuerdo a lo anterior, es posible construir una matriz que permita delimitar claramente todos los enfoques que ha adoptado la Inteligencia Artificial a lo largo de la historia.

	Eficiencia Humana	*Racionalidad*
Procesos mentales	Sistemas que piensan como humanos.	Sistemas que piensan racionalmente.
Conducta	Sistemas que actúan como humanos.	Sistemas que actúan racionalmente.

Definiciones que se circunscriben en cada uno de estos enfoques son las siguientes:

	Eficiencia Humana	*Racionalidad*
Procesos mentales	La IA es la tarea de lograr que las computadoras piensen, máquinas con mente, en su amplio sentido literal. [Haugeland85] La IA es la automatización de actividades que vinculamos con procesos de pensamiento humano, actividades tales como toma de decisiones, resolución de problemas o aprendizaje. [Bellman78]	La IA es el estudio de las facultades mentales mediante el uso de modelos computacionales. [Charniak86] La IA es el estudio de los cálculos que permiten percibir, razonar y actuar. [Winston92]

	Eficiencia Humana	*Racionalidad*
Conducta	La IA es el arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren de inteligencia. [Kurzweil90] La IA es el estudio de cómo lograr que las computadoras realicen tareas que, por el momento, los humanos hacen mejor. [Rich94]	La IA es un campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales. [Schalkoff90] La IA es la rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente. [Luger93]

Desde luego, existe una tensión entre enfoques centrados en humanos y los centrados en la racionalidad (con esto no se esta diciendo que los humanos sean irracionales o emocionalmente inestables, tan solo que los humanos cometen sistemáticamente errores en el proceso de razonamiento).. El enfoque centrado en el comportamiento humano constituye una ciencia empírica, que entraña el empleo de hipótesis y de la confirmación mediante experimentos. El enfoque racionalista combina matemáticas e ingeniería. Los miembros de estos grupos frecuentemente critican el trabajo realizado por los miembros de otros grupos, si bien todas las orientaciones han hecho valiosas aportaciones.

Igualmente la división no es totalmente nítida, y en muchos trabajos de Inteligencia Artificial confluyen varios enfoques. A continuación se presentará una breve descripción de cada uno de estos enfoques.

1.1.1.1. Actuar como Humano.

El "comportamiento inteligente" fue definido por primera vez por A. Turing en su famoso "juego de la imitación", versión moderna de un juego de salón del siglo XVIII [Turing50] . Dicho juego consiste en que una persona (el árbitro) "conversa" con alguien (el incógnito) situado en otro cuarto, a través de mensajes escritos llevados por un mensajero. Ahora bien, el incógnito puede ser una persona o una computadora, y el arbitro trata de determinar de cual de las dos se trata, únicamente con base en las preguntas y respuestas de los mensajes. El arbitro gana si acierta a adivinar que tipo de interlocutor tiene, y pierde en caso contrario. Se supone que los mensajes están mecanografiados, de manera que el arbitro no se puede guiar por la escritura, sino únicamente por el contenido.

Mediante la Prueba de Turing, se intenta ofrecer una satisfactoria definición operativa de lo que es la inteligencia. Como se puede apreciar, Turing definió una conducta inteligente como la capacidad de lograr eficiencia a nivel humano en todas las actividades de tipo cognoscitivo, suficiente para engañar a un evaluador humano.

Evidentemente, el juego de la imitación supone una gran capacidad por parte de las computadoras para entender el lenguaje natural, así como capacidad de razonamiento con sentido común y una amplia cultura general. A pesar de la gran dificultad para dominar todas estas áreas, recientemente se han elaborado sistemas computacionales que han jugado exitosamente el juego de la imitación, en un concurso llamado "Turing Test" [Epstein92] .

1.1.1.2. Pensar como Humano.

Para poder afirmar que un programa determinado utiliza algún tipo de razonamiento humano, previamente habrá que definir cómo piensan los seres humanos. Habrá que penetrar en le funcionamiento de la mente humana. Hay dos formas de hacer esto: mediante la introspección (para intentar atrapar nuestros propios pensamientos conforme éstos se van dando) o mediante la realización de experimentos psicológicos.

Una vez que se cuente con una teoría bastante precisa de la mente, puede procederse a expresar tal teoría en un programa de computadora. Si los datos de entrada/salida del programa y la duración de tiempo de su comportamiento corresponden a los de la conducta humana, existe evidencia de que algunos de los mecanismos del programa también funcionan en los seres humanos. En el caso de Newell y Simon, creadores de SGP, solucionador general de problemas, no bastó con que su programa resolviera correctamente los problemas propuestos. Lo que les interesaba fundamentalmente era seguir la pista de los pasos de razonamiento y compararla con la ruta seguida por sujetos humanos a los que se propuso los mismos problemas. Esta actitud contrasta fuertemente con la de otros investigadores de la misma época, a quienes lo que les importaba era la obtención de respuestas correctas independientemente de cómo las obtendría un ser humano. En el campo interdisciplinario de la ciencia cognoscitiva concurren modelos computacionales de IA y técnicas experimentales de psicología para intentar elaborar teorías precisas y verificables del funcionamiento de la mente humana.

1.1.1.3. Pensar Racionalmente.

El desarrollo de la lógica formal a fines del siglo XIX y a principios del XX permitió contar con una notación precisa para representar aseveraciones relacionadas con todo lo que existe en el mundo, así como sus relaciones mutuas (a diferencia de lo que se sucede con la notación de la aritmética común, en cuyo caso prácticamente sólo se representan aseveraciones acerca de la igualdad y desigualdad entre números). Ya para 1965 existían programas que , contando con tiempo y memoria suficientes, podían describir un problema en notación lógica y encontrarle solución, siempre y cuando ésta existiese. (De no existir, el programa continuaría sin cesar de buscarla). En la IA, la tradición logicista se esfuerza por elaborar programas como el anterior para crear sistemas inteligentes.

Este enfoque presenta dos obstáculos. En primer lugar, no es fácil recibir un conocimiento informal y expresarlo en los términos formales que exige la notación lógica, especialmente cuando el conocimiento tiene menos del 100 % de certidumbre. En segundo lugar, hay una gran diferencia entre la posibilidad de resolver un problema "en principio", y realmente hacerlo en la práctica. Incluso problemas que entrañan una docena de elementos agotarían la capacidad de cómputo de cualquier computadora a

menos que se cuente con lineamientos sobre los pasos de razonamiento que hay que utilizar primero.

1.1.1.4. Actuar Racionalmente.

Actuar racionalmente implica actuar de manera que se emprenda la mejor acción posible en una situación dada según los objetivos deseados, con base en ciertos supuestos.

En el caso del enfoque de la IA según las "leyes del pensamiento", todo el énfasis se ponía en hacer inferencias correctas. La obtención de estas inferencias a veces forma parte de lo que se considera la actuación racional, puesto que una manera de actuar racionalmente es el razonamiento lógico que nos asegure la obtención de un resultado determinado, con lo que se actuará de conformidad con tal razonamiento. Sin embargo, el efectuar una inferencia correcta no siempre depende de la racionalidad, pues existen situaciones en las que no existe algo que se pueda considerar lo que correctamente debería hacerse, y sin embargo hay que decidirse por un curso de acción. Existen también maneras de actuar racionalmente que de ninguna manera entrañan inferencia alguna.

Todas las "habilidades cognitivas" que se necesitan en la Prueba de Turing permiten emprender acciones racionales. Por lo tanto, es necesario contar con la capacidad para representar el conocimiento y razonar con base en él, pues de esta manera se podrán tomar decisiones correctas en una amplia gama de situaciones. Es necesario ser capaces de generar oraciones comprensibles en lenguaje natural, puesto que la enunciación de tales oraciones nos permiten desenvolvernos en una sociedad compleja. El aprendizaje no se emprende solo por erudición, sino porque el profundizar en el conocimiento de cómo funciona el mundo facilitará concebir mejores estrategias para manejarse en él. La percepción visual no es sólo algo divertido, sino que es algo necesario para darse una mejor idea de lo que una acción determinada puede producir.

Considerar la IA adoptando el enfoque de la Actuación Racional ofrece dos ventajas. Primera, es más general que el enfoque de las "leyes del pensamiento", dado que el efectuar inferencias correctas es sólo un mecanismo útil para garantizar la racionalidad, pero no es un mecanismo necesario. Segunda, es más afín a la manera como

se ha producido el avance científico que los enfoques basados en la conducta o pensamiento humanos, toda vez que se define claramente lo que será la norma de la racionalidad, norma que es de aplicación general. Por el contrario, la conducta humana se adapta bien sólo en un entorno específico y, en parte, es producto de un proceso evolutivo complejo y en gran parte ignoto, cuya perfección todavía se ve distante.

Actualmente, este enfoque práctico en el que lo importante no es la inteligencia en general, sino la forma en que ciertas técnicas computacionales desarrolladas en el ámbito de la Inteligencia Artificial pueden resolver efectivamente problemas reales, se ha vuelto predominante porque ha sido necesario justificar con resultados económicos las grandes inversiones que se han destinado a la investigación en Inteligencia Artificial.

Durante el desarrollo del Agente Inteligente, cuya definición se verá más adelante en este capítulo y cuyo desarrollo se encuentra en el Capítulo IV, se utilizará el enfoque de la Actuación Racional para definir su ámbito de inteligencia.

1.1.2. Áreas de Estudio de la Inteligencia Artificial.

El campo de la Inteligencia Artificial esta compuesto por varias áreas de estudio, de éstas, las áreas fundamentales son:

1. Juegos.
2. Demostración de Teoremas.
3. Sistemas Expertos.
4. Programación Automática.
5. Problemas combinatorios y de Planificación.
6. Percepción.
7. Resolución de problemas.
8. Procesamiento del Lenguaje Natural.
9. Algoritmos genéticos.
10. Redes neuronales.
11. Robótica.

12. Representación del conocimiento.
13. Deducción Automática.
14. Aprendizaje Automático.
15. Incertidumbre.
16. Agentes Inteligentes.

Algunas de ellas representan aplicaciones finales, como los Sistemas Expertos; otras, como el Procesamiento de Lenguaje Natural y la Búsqueda de Soluciones, pueden ser consideradas como bloques o técnicas que son agregados a otros programas para mejorar su desempeño.

1.1.2.1. Juegos.

Hay dos razones para que los juegos sean dominio de exploración de la Inteligencia Artificial:

- Proporcionan una tarea estructurada en la que es muy fácil medir el éxito o el fracaso.
- Requieren una gran cantidad de conocimientos.

Debido a estas razones los "Juegos" necesitan alguna clase de procedimiento de búsqueda heurística, generalmente los procedimientos de búsqueda son procedimientos de generar/comprobar, en donde la comprobación se realiza después de cantidades variables de trabajo realizadas por el generador. En un extremo el generador proporciona soluciones completas propuestas que se comprobarán y evaluarán. En el otro extremo, el generador genera movimientos individuales del espacio de búsqueda.

En un juego los movimientos legales proporcionan la manera de llegar desde el estado inicial hasta el estado - meta.

Mirándolo así, está claro que para mejorar la efectividad de un programa resolutor de problemas basado en la búsqueda, existen dos opciones:

1. mejorar el procedimiento de generación de forma que sólo se generen buenos movimientos o caminos.

2. mejorar el procedimiento de comprobación para que sólo se reconozcan y exploren en primer lugar los mejores movimientos.

En programas de juegos es particularmente importante que se hagan ambas cosas. Puesto que el procedimiento de comprobación deberá atender a cada una de los movimientos legales, el procedimiento de comprobación debe evaluar tantas posibilidades de la forma más rápida. Por lo tanto, probablemente no pueda realizar un buen trabajo.

Con un generador de movimientos más selectivo, el procedimiento de comprobación puede permitirse el lujo de gastar más tiempo en evaluar cada uno de los movimientos que se le proporcionan. Por tanto, puede producir un resultado más fiable. Así pues, al incorporar conocimientos heurísticos tanto en el generador como en el comprobador, se mejora la actuación del sistema global.

La forma ideal de usar un procedimiento de búsqueda para encontrar una solución a un problema es generar movimientos a través del espacio del problema hasta que se encuentra un estado-meta, que es aquel en el cual ganamos.

Gran parte del trabajo realizado en programas de juegos se ha efectuado en el desarrollo de buenas funciones de evaluación estática, que es usar toda la información disponible para evaluar posiciones individuales estimando posibilidades que conduzcan eventualmente a la victoria.

Aunque alguna vez pareció que jugar juegos, como el ajedrez, podría ser fácil para una computadora, hacerlo bien representa enfrentarse a una avasalladora explosión combinatoria que se genera.

1.1.2.2. Demostración de Teoremas.

El encontrar una demostración o refutación de un teorema presentado como conjetura, se considera normalmente como una tarea inteligente. No sólo requiere la capacidad de extraer deducciones a partir de hipótesis, sino que exige destrezas intuitivas tales como la de hacer conjeturas sobre qué lemas deberían ser probados antes para facilitar la demostración del teorema principal, basado en gran cantidad de conocimiento especializado para prever con cierta seguridad qué teoremas previamente demostrados serán útiles en la demostración que se trata de realizar y para fraccionar el problema en subproblemas en los que pueda trabajar independientemente.

1.1.2.3. Sistemas Expertos.

Estos sistemas proporcionan a los usuarios humanos conclusiones técnicas sobre materias especializadas. Un problema clave en el desarrollo de sistemas expertos de consulta es encontrar la forma de representar y usar el conocimiento que los humanos expertos en esas materias poseen y usan. Este problema se hace más difícil por el hecho de que el conocimiento de los expertos es a menudo impreciso, dudoso o anecdótico.

En los Sistemas expertos normalmente se emplean técnicas de deducción basada en reglas de la Inteligencia Artificial. En ellos, el conocimiento experto se representa como un conjunto de numerosas reglas simples, y estas reglas se usan para guiar el diálogo entre el sistema y el usuario y para deducir conclusiones.

1.1.2.4. Programación Automática.

La tarea de escribir un programa está relacionada tanto con la demostración de teoremas como con la Robótica.

La programación automática es como un "super compilador" o programa que podría recibir una descripción a muy alto nivel de lo que el programa buscado debe realizar, y construirlo. La descripción a alto nivel podría ser una sentencia precisa en un

lenguaje formal tal como el cálculo de predicados, o podría ser también una somera descripción, que requeriría ulterior diálogo entre el sistema y el usuario para resolver ambigüedades.

La programación automática se relaciona con la tarea de probar que un programa dado lleve a un determinado resultado. La noción de depuración es una estrategia de resolución de problemas, que insiste en obtener desde el principio una solución exenta por completo de defectos.

1.1.2.5. Problemas Combinatorios y de Planificación.

Una clase interesante de problemas está relacionada con la determinación de planes o combinaciones óptimas.

En la mayor parte de estos problemas, el dominio de las combinaciones o secuencias entre las que debe escogerse la solución es muy grande. Los intentos rutinarios de resolver problemas de estos tipos suelen generar en seguida explosiones combinatorias de posibilidades que agotan la capacidad de los mayores equipos de cómputo.

Pertenecen a una clase que los teóricos de la computación llaman NP-Completos, estos teóricos clasifican la dificultad de los diversos problemas de acuerdo con la forma en que crece el tiempo preciso para su solución.

El tiempo requerido por los mejores métodos conocidos para resolver los problemas NP-Completos crece exponencialmente con el tamaño de los mismos. Ese método puede transformarse en métodos similarmente más rápidos para la resolución de los restantes problemas NP-Completos, tendremos que resolverlos usando métodos con tiempo exponencial.

Se han desarrollado varios métodos para retrasar y moderar la inevitable explosión combinatoria, el conocimiento sobre el dominio del problema es la clave para encontrar métodos de solución más eficaces.

1.1.2.6. Percepción.

El proceso de la percepción tiene datos complejos como entrada por ello requiere "comprensión". Esta "comprensión" necesita una amplia base de conocimiento acerca de los cosas que han de ser percibidas (vistas y oídas).

El propósito del proceso de percepción completo es producir una representación condensada que sustituya los datos de entrada cuya inmensidad y crudeza los hace directamente inmanejables.

La principal dificultad que se encuentra en la percepción de una escena es el enorme número de descripciones posibles que aparecen como candidatos a que el sistema se interese por ellas.

Estas hipótesis se comprueban entonces por detectores especializados en componentes de descripciones, el resultado de esas comprobaciones permite, a su vez, formular mejores hipótesis. Este paradigma de hipótesis y comprobación se aplica en muchos niveles del proceso de percepción.

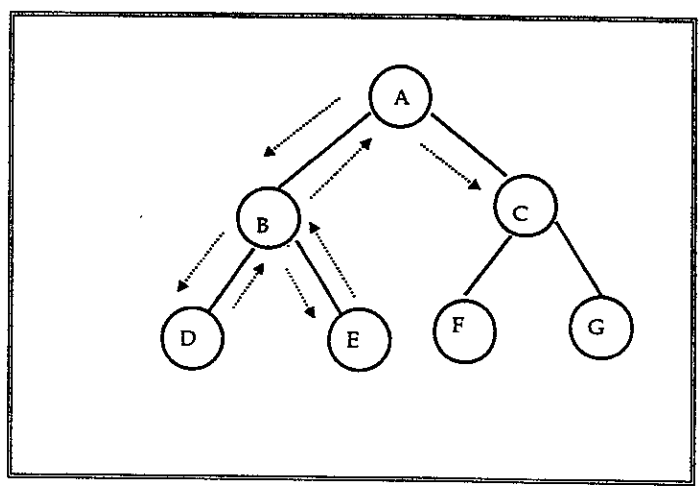
1.1.2.7. Resolución de Problemas o Búsqueda de soluciones.

El término búsqueda, aplicado a la Inteligencia Artificial se refiere a la búsqueda de soluciones de un problema. Solucionar un problema es fundamental para la mayoría de las aplicaciones de Inteligencia Artificial. Existen dos tipos de problemas. El primer tipo puede ser resuelta usando alguna clase de procedimiento determinístico que garantiza tener éxito. Este procedimiento es llamado una Computación. Solucionar un problema por computación normalmente se aplica a aquel tipo de problemas para los cuales dichos procedimientos existen. Es posible el trasladar los métodos que son usados para solucionar estos problemas fácilmente en un algoritmo que pueda ejecutar una computadora. Sin embargo, ya que algunos problemas del mundo real no pueden ser resueltos por medio de soluciones computacionales, deben ser considerados dentro de la segunda categoría, la cual consiste en problemas que son solucionados por medio de una

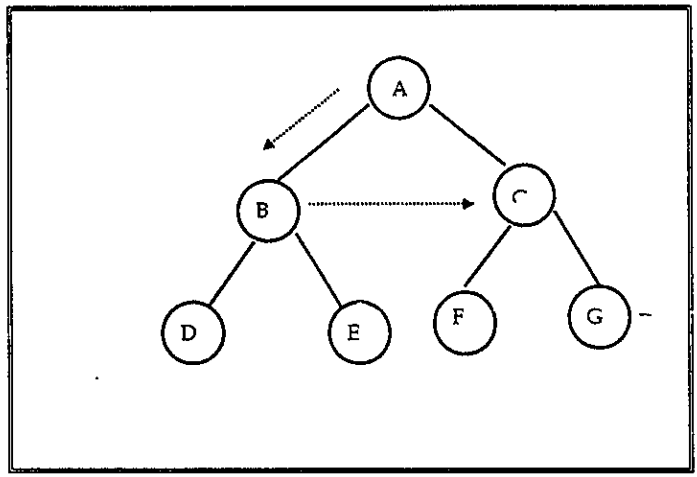
búsqueda de soluciones, el cual es el método involucrado con la solución de problemas en Inteligencia Artificial.

En los métodos de búsqueda de una solución se parte de un problema en que se sabe como generar posibles soluciones al problema. Todo problema puede ser representado por medio de un grafo, con lo cual la búsqueda de soluciones puede ser considerada como la búsqueda de un nodo en un grafo. Las técnicas de búsqueda en un grafo se dividen en:

- Depth-First. Explora cada posible camino hasta su conclusión, o antes si encuentra el nodo meta, antes de intentar otro camino. En el peor de los casos, la búsqueda degenerará en una búsqueda exhaustiva, es decir, examinará todos los nodos. Por ejemplo, si se desea encontrar el nodo C en el siguiente grafo, el camino será el siguiente:



- Breadth-First. Es lo opuesto a la búsqueda Depth-First. Verifica cada nodo en el mismo nivel antes de proceder al siguiente nivel de profundidad. Por ejemplo, si se desea encontrar el nodo C en el siguiente grafo, el camino será el siguiente:



El problema de encontrar el camino más corto entre dos ciudades A y B enlazadas por una red de carreteras, puede servir de ejemplo de un problema que requiere de la inteligencia Artificial para encontrar la solución. Así, para el mapa de la figura 1, una posible ruta de A a B es el camino (A, C, G, D, E, F, G, C, B).

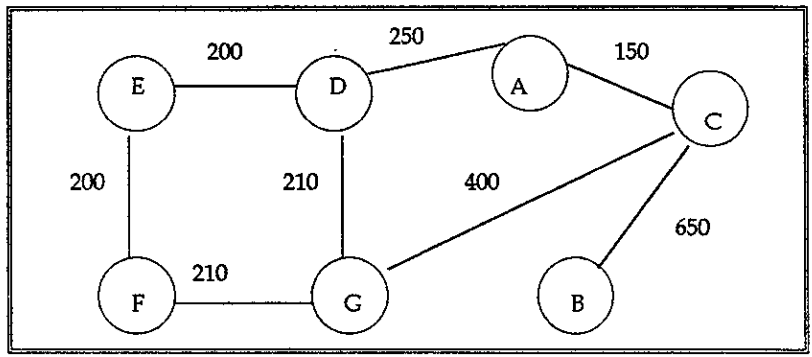


Figura 1.1. Mapa de una red de carreteras

Una solución aparentemente simple consiste en obtener todos los caminos posibles en la gráfica, tomar aquellos que unan A con B, y finalmente seleccionar entre estos al de menor costo. La desventaja de este método es que requiere una gran cantidad de cálculos,

de los cuales la mayor parte se desperdician. A esta explosión de soluciones posibles a examinar se le llama "explosión combinatoria", y es el enemigo a vencer por excelencia en la mayoría de los problemas de Inteligencia Artificial.

Una idea "inteligente" podría ser buscar, a partir de la ciudad A, solamente los caminos que "acerquen" a B, lo cual requiere conocer el mapa para saber si nos acercamos o nos alejamos de B. El defecto de esta solución es que tal vez la solución más corta requiere momentáneamente alejarse de B para después llegar a ella por una mejor ruta (esto ocurre en el mapa de la figura 1.1). Es decir, la idea "inteligente" (que se acostumbra llamar "heurística" en el lenguaje de la Inteligencia Artificial) no garantiza encontrar la mejor solución, pero sí una de las mejores. Para muchos problemas de gran complejidad, es preferible sacrificar el óptimo absoluto con tal de reducir drásticamente la cantidad de posibles soluciones a revisar.

Los tres principales pasos que se requieren para construir un sistema que resuelva un problema particular son:

1. Definir el problema con precisión, debe incluir especificaciones precisas de cuáles serán las situaciones esenciales y de las situaciones finales que constituyen soluciones aceptables del problema.
2. Analizar el problema. Unas pocas características muy importantes pueden tener un impacto inmenso en la adecuación de las diversas técnicas posibles para resolver el problema.
3. Escoger las mejores técnicas y aplicarlas al problema particular.

Pueden describirse fácilmente como un conjunto de reglas formadas por dos partes: una parte izquierda que sirve como un modelo que debemos comprobar que coincida con la realidad y una parte derecha que describe el cambio que debe realizarse.

Sin embargo, el uso de tantas reglas crea dos dificultades prácticas serias:

- Nadie puede proporcionar un conjunto completo de tales reglas. Tomaría demasiado tiempo y, no podrían hacerse sin errores.

- Ningún programa puede manejar fácilmente todas esas reglas.

La misma clase de representación también es útil para problemas peor estructurados que surgen de modo más natural, aunque puede ser necesario el uso de estructuras más complejas de una matriz para describir un estado individual. La representación del espacio de estado forma la base de casi todos los métodos de Inteligencia Artificial. Su estructura corresponde a la estructura de resolución de problemas:

- Permite definir formalmente un problema como la necesidad de convertir una situación dada en una deseada, usando un conjunto de situaciones permisibles.

- Nos permite definir el proceso de solución de un problema concreto como una combinación de técnicas conocidas y de búsqueda, que es la técnica general de explorar el espacio para intentar encontrar una solución.

1.1.2.8. Procesamiento del Lenguaje Natural.

Ha resultado muy difícil desarrollar sistemas informáticos capaces de generar y "entender" incluso simples fragmentos de un lenguaje natural y es que el lenguaje ha evolucionado como medio de comunicación entre seres inteligentes.

En principio sirve para transmitir una porción de "estructura mental" de un cerebro a otro, bajo circunstancias en las que ambos poseen estructuras mentales extensas y muy semejantes, que les proporcionan un contexto común. Además, parte de esas estructuras mentales similares permite a cada uno de los participantes saber que el otro la posee también y que podrá y querrá utilizarla para realizar ciertos procesos durante los "actos" de comunicación.

El generar y comprender un lenguaje encierra un problema de codificación y decodificación de enorme complejidad.

Un sistema informático capaz de comprender un mensaje en un lenguaje natural requerirá tanto un conocimiento del contexto como un proceso para realizar las inferencias supuestas por el generador del mensaje.

Para ese desarrollo son fundamentales algunas ideas de Inteligencia Artificial sobre estructuras para representar el conocimiento contextual y ciertas técnicas para realizar inferencias a partir de ese conocimiento.

1.1.2.9. Algoritmos Genéticos.

El algoritmo genético es una técnica de búsqueda basada en la teoría de la evolución de Darwin, que ha cobrado tremenda popularidad alrededor del mundo durante los últimos años.

La técnica de los algoritmos genéticos se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno. Hoy en día se sabe que estos cambios se efectúan en los *genes* de un individuo (unidad básica de codificación de cada uno de los atributos de un ser vivo), y que sus atributos más deseables (vrg. los que le permiten adaptarse mejor a su entorno) se transmiten a sus descendientes cuando éste se reproduce sexualmente.

Un investigador de la Universidad de Michigan llamado John Holland estaba consciente de la importancia de la selección natural, y a fines de los 60s desarrolló una técnica que permitió incorporarla en un programa de computadora. Su objetivo era lograr que las computadoras aprendieran por sí mismas. A la técnica que inventó Holland se le llamó originalmente "planes reproductivos", pero se hizo popular bajo el nombre "algoritmo genético" tras la publicación de su libro [Holland75] en 1975.

Una definición bastante completa de un algoritmo genético es la propuesta por John Koza [Koza92]:

"Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud".

1.1.2.9.2. FACTIBILIDAD DE UTILIZACIÓN DE LOS ALGORITMOS GENÉTICOS

La aplicación más común de los algoritmos genéticos ha sido la solución de problemas de optimización, en donde han mostrado ser muy eficientes y confiables. Sin embargo, no todos los problemas pudieran ser apropiados para la técnica, y se recomienda en general tomar en cuenta las siguientes características del mismo antes de intentar usarla:

- Su espacio de búsqueda (vgr. sus posibles soluciones) debe estar delimitado dentro de un cierto rango.
- Debe poderse definir una función de aptitud que nos indique qué tan buena o mala es una cierta respuesta.
- Las soluciones deben codificarse de una forma que resulte relativamente fácil de implementar en la computadora.

El primer punto es muy importante, y lo más recomendable es intentar resolver problemas que tengan espacios de búsqueda discretos (aunque éstos sean muy grandes). Sin embargo, también podrá intentarse usar la técnica con espacios de búsqueda continuos, pero preferentemente cuando exista un rango de soluciones relativamente pequeño.

La función de aptitud no es más que la función objetivo de nuestro problema de optimización. El algoritmo genético únicamente maximiza, pero la minimización puede

realizarse fácilmente utilizando el recíproco de la función maximizante (debe cuidarse, por supuesto, que el recíproco de la función no genere una división por cero). Una característica que debe tener esta función es que tiene ser capaz de "castigar" a las malas soluciones, y de "premiar" a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez.

La codificación más común de las soluciones es a través de cadenas binarias, aunque se han utilizado también números reales y letras. El primero de estos esquemas ha gozado de mucha popularidad debido a que es el que propuso originalmente Holland, y además porque resulta muy sencillo de implementar.

1.1.2.9.4. VENTAJAS Y DESVENTAJAS.

Ventajas	*Desventajas*
• No necesitan conocimientos específicos sobre el problema que intentan resolver. • Operan de forma simultánea con varias soluciones, en vez de trabajar de forma secuencial como las técnicas tradicionales. • Cuando se usan para problemas de optimización - maximizar una función objetivo - resultan menos afectados por los máximos locales (falsas soluciones) que las técnicas tradicionales. • Resulta sumamente fácil ejecutarlos en las modernas Arquitecturas masivamente paralelas. • Usan operadores probabilísticos, en vez de los típicos operadores determinísticos de las otras técnicas.	• Pueden tardar mucho en converger, o no converger en absoluto, dependiendo en cierta medida de los parámetros que se utilicen - tamaño de la población, número de generaciones, etc.-. • Pueden converger prematuramente debido a una serie de problemas de diversa índole.

1.1.2.10. Redes Neuronales.

1.1.2.10.1. HISTORIA DE LAS REDES NEURONALES.

Las redes neuronales tuvieron su origen en los diversos intentos de modelar la actividad cerebral usando circuitos electrónicos y teoremas lógicos. De todos ellos el más importante fue el realizado por McCulloch y Pitts (1943), en el que demostraron que una computadora puede emplear los conceptos de la lógica para modelar funciones de entrada/salida utilizando un modelo matemático que denominaron red neuronal. El objetivo inicial de McCulloch y Pitts era simular la actividad cerebral usando lógica binaria, pero encontraron la dificultad de que las neuronas cerebrales no funcionan de forma binaria. En los años 50 se empezó a trabajar en sistemas de reconocimiento de formas en paralelo (sistemas conexionistas) en los que los elementos de proceso ya no eran binarios, sino que tomaban valores continuos en un rango determinado.

Básicamente las redes neuronales son modelos computacionales que tienen unas características comunes como la capacidad de adaptarse o aprender, la capacidad de generalización y la capacidad de auto organizarse, o agrupar la información en clases.

1.1.2.10.2. DEFINICIÓN.

Podemos definir una red neuronal como un conjunto de neuronas interconectadas de una manera concreta, y una neurona como la unidad básica de la red, que recibe múltiples entradas, elabora la información recibida y emite una única salida o respuesta que puede transmitirse a otras neuronas. En su origen, existe una leve inspiración en las neuronas biológicas. Por ejemplo, al igual que en éstas existe un proceso de convergencia de la información de entrada en una única información de salida, y un proceso de divergencia de la información de salida hacia múltiples neuronas (o unidades de salida directamente).

Una sinapsis es la conexión de una neurona j con otra neurona i. En general las sinapsis son unidireccionales (una neurona transmite información a otra, pero la segunda

no transmite nada a la primera), pero también existen modelos de redes en los que el intercambio de información es bidireccional (como en el caso de las redes de Hopfield). Se dice que una sinapsis es inhibitoria cuando el nivel de activación de la neurona que recibe la información decrece como consecuencia de la activación de la neurona origen, y crece cuando se desactiva. En caso contrario, la sinapsis es excitatoria. Cada una de las conexiones entre las neuronas tiene un peso asociado, que determina la contribución de la activación de la neurona emisora a la activación de la neurona receptora, de tal forma que si el peso es negativo, la conexión es inhibitoria y si el peso es positivo la conexión es excitatoria. Cada neurona tiene un estado de activación en cada instante, y una salida que en general depende del estado de activación.

La mayoría de las redes neuronales tienen alguna regla de entrenamiento, en donde la ponderación de las sinapsis se determina de acuerdo con los modelos presentados. Esto significa que las redes neuronales aprenden a través de ejemplos.

1.1.2.11. Robótica.

La investigación sobre robots o Robótica ha conducido a varias técnicas para modelar estados del mundo y para describir los procesos de cambio de uno de estos estados a otro. Ha conducido también a una mejor comprensión de cómo generar planes para secuencias de acciones y cómo dirigir la ejecución de esos planes.

Nos han llevado a desarrollar métodos de planificación con altos niveles de abstracción, ignorando detalles, para después planificar a niveles cada vez inferiores, en los que los detalles se van haciendo importantes.

Los robots han recorrido una larga trayectoria desde el primer robot comercial que fue vendido por Planet en 1959, sin embargo, falta mucho por hacerlos más versátiles, productivos, económicos y ligeros.

La Inteligencia Artificial es el área tecnológica que precisa más desarrollo y pericia para acelerar la evolución de los robots; algunos autores indican que un robot inteligente es capaz de:

- Recibir comunicación.
- Comprender su entorno mediante el uso de modelos.
- Formular planes.
- Ejecutar planes.
- Monitorizar su operación.

1.1.2.11.1 CONCEPTO DE ROBOT

Son principalmente máquinas con manipuladores que pueden programarse fácilmente para hacer una gran variedad de tareas automáticamente. Un robot (para la Robotic Industries Association), es un manipulador multifuncional reprogramable diseñado para mover materiales, piezas o dispositivos especializados mediante movimientos programados para una gran variedad de tareas; un robot consta de:

- Uno o más manipuladores (brazos).
- Efectos finales (manos).
- Un controlador.
- Sensores que proporcionan información sobre el entorno y retro-información sobre la ejecución de la tarea.

Casi todos los robots han sido desarrollados para aplicaciones industriales con el fin de conseguir una mayor productividad, para reducir costos, para suplir mano de obra especializada, para proporcionar flexibilidad en procesos de fabricación por lotes, para mejorar la calidad de los productos y/o para liberar a los seres humanos de tareas repetitivas y pesadas u operaciones en ambientes hostiles.

1.1.2.12. Representación del Conocimiento.

Para actuar "inteligentemente" se requiere tener una representación del ambiente sobre el que se actúa, o por lo menos de los aspectos de él que son relevantes para el problema a resolver. Por ejemplo, el conocimiento necesario sobre la geografía y la red de carreteras que unen a dos ciudades. De toda esta geografía hay muchos detalles que son

irrelevantes para el problema, como el tipo de vegetación predominante a cada región, y podemos restringir nuestra atención a las informaciones que afectan directamente la solución al problema. Podemos representar el conocimiento de este problema como una tabla en la que tenemos las distancias entre las ciudades unidas por la red carretera. Dicha tabla puede ser vista como una función distancia_directa(X,Y) que nos da la distancia de X a Y, que son dos ciudades directamente comunicadas por carretera.

En general las representaciones del conocimiento usadas en Inteligencia Artificial son colecciones estructuradas de datos simbólicos, como en este ejemplo, aunque hay casos en los que se usan representaciones que incluyen números reales y datos continuos en vez de datos discretos y símbolos.

Las representaciones del conocimiento usadas en la tecnología de los Sistemas Expertos [Hayes-Roth87] han llegado a ser más conocidas que otras representaciones. Entre ellas encontramos principalmente las llamadas "reglas de producción", que son estructuras de la forma "Si la condición C es cierta, se ejecuta la acción A".

Otra representación que se ha abierto paso recientemente es la de los "frames" [Maida87], relacionada con la estructuración orientada a objetos de los sistemas computacionales. Consiste en declarar estructuras de "objetos" que tienen atributos ("slots"); y donde las acciones a ejecutar son lanzadas al momento de acceder a algunos de los atributos.

	A	*B*	*C*	*D*	*E*	*F*	*G*
A	0		150	250			
B		0	650				
C	150	650	0				400
D	250			0	200		210
E				200	0	210	
F					200	0	210
G			400	210		210	0

Tabla 1.1. Distancia entre ciudades conectadas directamente por carreteras

	A	B	C	D	E	F	G
A	0	800	150	250	600	620	460
B	800	0	650	610	410	210	430
C	150	650	0	400	600	620	400
D	250	610	400	0	200	400	210
E	600	410	600	200	0	200	410
F	620	210	620	400	200	0	210
G	460	430	400	210	410	210	0

Tabla 1.2. Distancia mínima entre dos ciudades dadas

Para cada problema en particular se puede buscar una representación "adecuada" del conocimiento existente. La adecuación de la representación elegida depende de como se vaya a explotar el conocimiento. Por ejemplo, en el caso de la red de carreteras, la Tabla 1.1 representa las distancias de las ciudades conectadas directamente por carretera, pero no las distancias entre las ciudades pasando por puntos intermedios. En este caso, como puede haber muchas maneras de llegar de una ciudad a otra, se puede pensar en representar la distancia mínima entre dos ciudades dadas, como en la Tabla 1.2. Si se va a consultar frecuentemente la distancia entre dos ciudades no vecinas, la representación de la Tabla 1.2 es preferible a la de la Tabla 1.1, mientras que si se desea una representación que requiera menos espacio en memoria, puede ser preferible almacenar solamente las distancias directas (como en la Tabla 1.1) y calcular las distancias indirectas en el momento en que se requiera. Por cierto, se puede economizar espacio aprovechando el hecho de que la tabla es simétrica, almacenando solo la mitad inferior izquierda de ella.

Cada representación del conocimiento esta asociada a una manera de explotar dicho conocimiento, y la eficiencia de cualquier representación no puede juzgarse independientemente del mecanismo de explotación. Así, las reglas de producción "Si condición entonces conclusión", están ligadas a los mecanismos estándares de "encadenamiento hacia adelante" y "encadenamiento hacia atrás". Similarmente, las representaciones basadas en objetos están asociadas a mecanismos que explotan la herencia de propiedades. En muchos "shells" comerciales (paquetes computacionales

especializados para el desarrollo y explotación de los sistemas expertos) se ofrecen combinaciones de mecanismos de representación y explotación del conocimiento, buscando aumentar la flexibilidad para el usuario, pero muchas veces reduciendo la solidez formal del sistema resultante.

Podemos distinguir diversas formas de conocimiento, según el problema a atacar y el punto de vista que se adopte, siendo las principales las siguientes:

- Conocimiento general: leyes que se cumplen sobre un conjunto de objetos. Puede presentarse como fórmulas matemáticas o lógicas, o de manera informal, en lenguaje hablado/escrito; sin embargo, la informalidad y la imprecisión obstaculizan la automatización del uso del conocimiento.

- Conocimiento procedural: secuencias de acciones a seguir; se puede representar mediante diagramas de flujo, algoritmos, etcétera.

- Conocimiento factual: hechos, como en el caso de las tablas de distancias.

- Metaconocimiento: conocimiento sobre el conocimiento. Puede ser una forma extremadamente importante de conocimiento, sobre todo en sistemas que aprenden.

1.1.2.13. Deducción Automática.

Una de las capacidades del comportamiento "inteligente" que han sido capturadas más eficazmente por la computadora es la capacidad de efectuar "razonamientos" correctos. Esto puede querer decir en algunos casos obtener conclusiones que se desprenden como consecuencia de datos existentes, o bien, verificar si un dato no contradice los datos existentes.

Por ejemplo en el mapa de carreteras podemos decir que si existe una ruta que me permite llegar de A a B y también puedo llegar de B a C, entonces podemos "concluir" que se puede llegar a A y C por la red de carreteras. Para poder hacer que la computadora

realice tales "razonamientos" es necesario precisar muy detalladamente la manera en que se pueden obtener soluciones a partir de las premisas. También se requiere expresar las informaciones en un lenguaje muy preciso.

Uno de los lenguajes más utilizados para efectuar razonamientos mecánicos es la Lógica [Genesereth87]. Por ejemplo, en lógica podríamos expresar las premisas del ejemplo de ciudades como:

- hay_ruta (A, B).
- hay_ruta (B, C).

En la lógica se puede declarar casi cualquier información; los casos en que la lógica estándar no es suficiente son muy específicos como para verlos aquí.

La forma de extraer conclusiones de las premisas es mediante las Reglas de Inferencia. Una regla de inferencia conocida desde los tiempos de los griegos es la llamada "modus ponens", que tiene la siguiente forma:

Si se sabe que siempre que P es cierto entonces Q es cierto,
y se sabe que actualmente P es cierto
Entonces Q es cierto.

Además de las reglas de inferencia y de las informaciones elementales puede ser necesario contar con unas premisas de uso general, conocidas como Axiomas. En el ejemplo de las carreteras, un axioma podría ser como sigue:

Para cualquier X y Y, si hay_ruta(X,Y) y
hay_ruta (Y,Z) entonces hay_ruta(X,Z).

Es muy sencillo aplicar el axioma de las carreteras y el modus ponens para obtener, a partir de hay_ruta(A,B) y hay_ruta(B,C) la conclusión hay_ruta(A,C); simplemente tenemos que reemplazar los datos reales por las variables, esto es, A por X, B por Y y C por Z.

Puede parecer curioso que con mecanismos tan simples como los que hemos descrito se puedan hacer sistemas de razonamiento mecánico de utilidad práctica. De hecho, el modus ponens es solo un ejemplo de reglas de inferencia; hay muchas reglas de inferencia propuestas y muchas estrategias para hacer eficiente el proceso de razonamiento.

Otros términos relacionados con el razonamiento automático son:

- Inferencia, deducción: Son términos que denotan el proceso de extraer una conclusión a partir de las premisas.

- Cálculo preposicional: Es el lenguaje más primitivo de la lógica, en que los símbolos representan afirmaciones completas; por ejemplo el símbolo PerroAnimal puede significar que es cierto que el perro es un animal. Cada símbolo es atómico (no tiene estructura interna), y solo puede ser cierto o falso (pero no ambos).

- Contradicción: Un conjunto de proposiciones es contradictorio cuando se considera cierta tanto una afirmación P como su negación (indicada como ~P).

- Cálculo de predicados: Es un lenguaje más sofisticado que el cálculo proposicional; se distingue el sujeto y el predicado de las afirmaciones; por ejemplo animal(perro) indicaría que "perro" es un animal.

- Prueba Automática de teoremas: Es una forma de razonamiento mecánico en que la conclusión ya esta dada, y hay que encontrar una secuencia de pasos para llegar a la conclusión. Cada paso es una aplicación de alguna de las reglas de inferencia.

- Prueba por refutación: Para probar que una conclusión C es consecuencia de unas premisas P, se supone falsa la conclusión, y se prueba que P junto con ~C es contradictorio.

Actualmente existen sistemas computacionales que efectúan deducciones automáticas en forma sumamente eficiente[McCune90], y que pueden ser integrados como componentes dentro de otros sistemas.

Además de los sistemas de razonamiento automático de uso general, existen sistemas deductivos para un propósito en particular, como por ejemplo, ingeniería de software [Smith90]. Otros sistemas de deducción automática se especializan en efectuar razonamiento que involucran al tiempo (razonamiento temporal) o al espacio físico (razonamiento espacial); ambas formas encuentran aplicación en la Robótica.

De hecho los Sistemas Expertos utilizan formas restringidas de razonamiento automático, que varían según la representación del conocimiento que haya sido utilizada (reglas, objetos, etc.).

1.1.2.14. Aprendizaje Automático.

Una de las capacidades más importantes en un sistema inteligente es que no dependa exclusivamente de lo que el programador ponga en el, sino que sea capaz de mejorar su comportamiento con base en su propia experiencia; esto es lo que se llama "aprendizaje automático" [Michalski83].

Cabe decir que uno de los mitos de la Inteligencia Artificial es que las máquinas pueden aprender cosas nuevas "por simple observación de su medio". En realidad, para que un sistema computacional sea capaz de "aprender" algo, es necesario establecer un marco restringido en el cual se desarrolle el aprendizaje. Vamos a ilustrar esto con el ejemplo de la red de carreteras.

Supóngase que un robot trata de encontrar, por ensayo y error, la ruta de A a B, y que después de dar algunas vueltas llego al objetivo. Sea (A, C, G, D, E, F, G, C, B) la ruta seguida por el robot. Entonces, si el robot fue memorizando la ruta seguida, puede luego procesarla de manera "inteligente"; lo menos que puede hacer es eliminar de esta los

ciclos. Haciendo esta operación se llega a la ruta (A, C, B), que es lo que el robot "aprendió". Para regresar de B a A ya no tiene que hacer ensayos, y puede seguir la ruta (A, C, B) invertida, esto es, (B, C, A). Este ejemplo, aunque simple, nos muestra una característica esencial del aprendizaje: es necesario memorizar partes de la realidad, pero también hay que olvidar otras. En el ejemplo, el robot "olvida" los ciclos.

Como se ve en el ejemplo, es necesario saber el tipo de informaciones que va a aprender el sistema, y mediante cuales mecanismos va a adquirir esas informaciones. Aun cuando hay investigación en Inteligencia Artificial tendiente a un aprendizaje de conceptos nuevos, de métodos nuevos, etc., estos estudios están aun muy lejos de producir resultados prácticos.

El aprendizaje simbólico no es la única forma posible, y es frecuente encontrar sistemas de Inteligencia Artificial en que el aprendizaje consiste en ajustar los valores numéricos de ciertos parámetros. Un caso muy común en Inteligencia Artificial es el ajuste de coeficientes en las funciones que evalúan que tan "propicia" o favorable es una situación dada. Este problema es importante cuando se quiere examinar que acción, entre varias alternativas posibles, lleva a una situación más favorable. Un caso típico es el de los juegos competitivos (ajedrez, backgammon, etc.). En este sentido es pionero el programa de juego de damas de Samuel [Samuel59], que mejoraba su rendimiento "aprendiendo" de sus mismos juegos, mediante simple ajuste de coeficientes.

1.1.2.15. Incertidumbre.

El mundo real las cosas no son como en lógica en donde solo pueden ser ciertas o falsas, en el mundo real hay muchas áreas grises. Incluso la validez de la mayoría de las observaciones esta determinada por un factor de probabilidad que le esta asociado. Las personas pueden manejar la incertidumbre sin muchos problemas porque están acostumbradas a ello, en cambio, hacer que las computadoras manejen incertidumbre requiere algún esfuerzo.

Intentar resolver situaciones donde cierta información es incierta nos remite a dos importantes áreas de la Inteligencia Artificial. La primera es la Lógica Difusa, la cual se relaciona con la evaluación de expresiones lógicas que puede contener valores desconocidos. La segunda área cubre los Sistemas Probabilísticos, los cuales utilizan la probabilidad de la ocurrencia de varios eventos para llegar a una respuesta.

1.1.2.15.1. LÓGICA DIFUSA..

La Lógica Difusa fue creada por Lotfi Zadeh en 1965. La lógica difusa esta estrechamente relacionada con la lógica booleana, de la cual se deriva, pero que aporta un nuevo concepto, el de los valores desconocidos, es decir, un valor que no es ni totalmente verdadero ni totalmente falso. Su creador se basó en la situación de que las personas frecuentemente toman decisiones basándose únicamente en situaciones imprecisas.

El significado del término Lógica Difusa es obscuro y demasiado amplio ya que ha sido aplicado a muchas situaciones diferentes. Nosotros restringiremos la definición de Lógica Difusa a la evaluación de expresiones lógicas que pueden contener valores desconocidos.

En vista de que la lógica es determinística, se puede pensar que la presencia de valores desconocidos invalida la expresión lógica; sin embargo, en algunos casos, un valor desconocido no significa que la expresión no pueda ser evaluada.

En Lógica Difusa, un valor lógico puede ser cualquiera de los siguientes:

- Verdadero (V).
- Falso (F).
- Desconocido (D).

La tabla de verdad que resulta de la combinación de estos valores es:

P	Q	$P \vee Q$	$P \wedge Q$	$\sim P$
V	V	V	V	F
V	F	F	V	F
V	D	D	V	F
F	F	F	F	V
F	D	F	D	V
D	D	D	D	D

Además de lo anterior, la Lógica Difusa también puede servir para introducir en los sistemas los conceptos de palabras imprecisas o relativas gracias a lo cual es posible utilizar las expresiones muchos, pocos, grande, pequeño.

1.1.2.15.1. SISTEMAS PROBABILÍSTICOS..

Una de las áreas más importantes donde se requiere el procesamiento de incertidumbre es la de los Sistemas Expertos. Para que un Sistema Experto pueda dar un buen consejo, debería también reportar la probabilidad de que dicho consejo sea correcto.

1.1.2.16. Agentes Inteligentes.

1.1.2.16.1. DEFINICIÓN.

Un agente es todo aquello que puede considerarse que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores. En el caso de los agentes de software, sus percepciones y acciones vienen a ser las cadenas de bits codificados.

Un agente racional es aquel que busca obtener el mejor desempeño posible. Existe una gran diferencia entre racionalidad y omnisciencia. Un agente omnisciente es aquel que sabe el resultado real que producirán sus acciones, y su conducta es congruente con

ello; sin embargo, en la realidad no existe la omnisciencia. La racionalidad, en cambio, contempla un éxito esperado, tomando como base lo que se ha percibido. [Russell96]

No se puede culpar a un agente por no haber tomado en cuenta algo que no podía percibir, o por no emprender una acción de la que es incapaz. Si se especifica que un agente inteligente siempre debe hacer lo que es realmente correcto, será imposible diseñar un agente para que cumpla con esa especificación.

El carácter de racionalidad de lo que se hace en un momento dado dependerá de cuatro factores:

- De la medida con la que se evalúa el grado de éxito logrado. Por ejemplo, la medida de desempeño de un conductor automatizado podría ser la de llegar al destino correcto, reducir al mínimo el consumo de combustible, desgaste del vehículo, tiempo de recorrido, su costo, violaciones al reglamento de transito u ofrecer el máximo de seguridad y comodidad al pasajero.
- De todo lo que hasta ese momento haya percibido el agente. A esta historia perceptual completa se le denomina la secuencia de percepciones.
- Del conocimiento que posea el agente acerca del medio.
- De las acciones que el agente pueda emprender.

Lo anterior permite definir un Agente Racional Ideal como un agente que deberá emprender, en todos los casos de posibles secuencias de percepciones, todas aquellas acciones que favorezcan obtener el máximo de su medida de rendimiento, basándose en las evidencias aportadas por la secuencia de percepciones y en todo conocimiento incorporado en tal agente.

Es importante aclarar algunos puntos importantes de la definición anterior, el mapeo de las secuencias de percepciones para acciones y la autonomía.

En vista de que el comportamiento de un agente depende exclusivamente de la secuencia de sus percepciones en un momento dado, se sabe que es posible caracterizar cualquier agente en particular elaborando una tabla de las acciones que éste emprende

como respuesta a cualquier secuencia de percepciones posible. Dicha lista se denomina un mapeo de secuencias de percepciones para acciones. El especificar una función que permita determinar el tipo de acción que deberá emprender un agente como respuesta a una determinada secuencia de percepciones constituye el diseño del agente. Lo anterior no implica, desde luego, que se debe crear una tabla explícita con una entrada por cada posible secuencia de percepciones. En vez de ello, se pude definir una especificación del mapeo sin tener que enumerarlo exhaustivamente.

Igualmente importante es el concepto de autonomía, la cual tiene una muy estrecha relación con el "conocimiento integrado" del agente. Si las acciones que emprende el agente se basan exclusivamente en un conocimiento integrado, con lo que se hace caso omiso de sus percepciones, se dice que el agente no tiene autonomía.

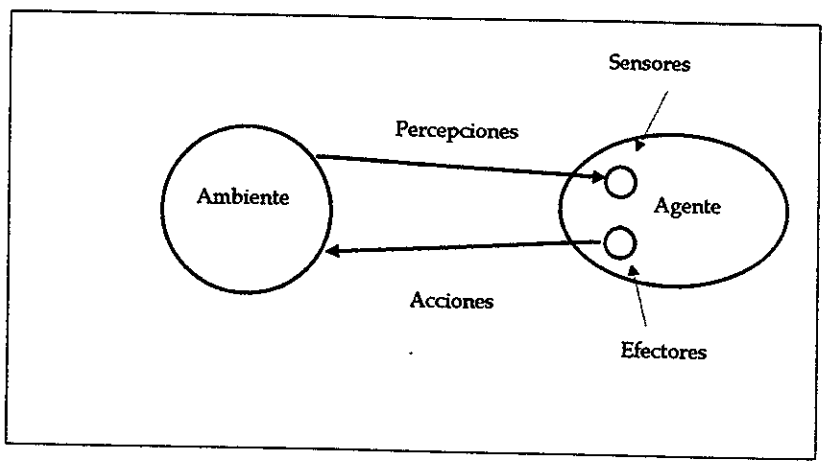


Figura 1.2. Representación de un Agente Inteligente

1.1.2.16.2. ESTRUCTURA..

La función o programa que permite implantar el mapeo del agente para pasar de percepciones a acciones deberá ejecutarse en algún tipo de dispositivo de cómputo, al que se denomina arquitectura. Desde luego, el programa elegido debe ser aquel que la arquitectura acepte y pueda ejecutar. La arquitectura pone al alcance del programa las percepciones obtenidas mediante los sensores, lo ejecuta y alimenta la efector con las

acciones elegidas por el programa conforme éstas se van generando. Así, se puede determinar que un agente es la suma del programa con la arquitectura.

1.1.2.16.3. TIPOS DE AGENTES..

Existen cuatro principales tipos de agentes, a continuación se presenta una breve descripción de cada una de ellas.

1.1.2.16.3.1. Agentes de reflejo simple.

Estos agentes utilizan reglas condición - acción, también conocidas como reglas de situación - acción o reglas si - entonces, que permiten al agente establecer la conexión entre percepciones y acciones. Un ejemplo de este tipo de agentes podría ser un agente que conduciese un automóvil, el cual podría activar una conexión ya definida para el caso de que el carro de enfrente empezara a frenar. La regla quedaría:

Si el carro de adelante está frenando ***entonces*** empiece a frenar.

Si bien agentes como estos se pueden implantar con bastante eficiencia, la extensión del ámbito en donde se pueden aplicar es bastante limitada.

1.1.2.16.3.2. Agentes reflejo con estado interno.

El agente de reflejo simple funcionará sólo si se toma la decisión adecuada con base en la percepción de un momento dado. El problema que este tipo de agentes trata de solucionar es el que se presenta debido a que los sensores no informan acerca del estado total del mundo. En casos así, el agente necesita actualizar algo de información en el estado interno que le permita discernir entre estados del mundo que generan la misma entrada de percepciones pero que, sin embargo, son totalmente distintas.

La actualización de esta información sobre el estado interno conforme va pasando el tiempo exige la codificación de dos tipos de conocimiento en el programa de agente. En primer lugar, se necesita cierta información sobre cómo está evolucionando el mundo,

independientemente del agente. En segundo lugar, se necesita información sobre cómo las acciones del agente mismo afectan al mundo.

1.1.2.16.3.3. Agentes basados en metas.

Para decidir qué hay que hacer no siempre basta con tener información acerca del estado que prevalece en el ambiente. Es decir, además de una descripción del estado prevaleciente, también se requiere de cierto tipo de información sobre la meta, información que detalle las situaciones deseables. El programa de agente puede combinar lo anterior con la información relativa al resultado que producirían las posibles acciones que se emprendan y de esta manera elegir aquellas acciones que permitan alcanzar la meta. En ocasiones esto es muy sencillo, cuando alcanzar la meta depende de responder con una sola acción; otras veces es más complicado, cuando el agente tenga que considerar largas secuencias de giros y vueltas hasta que logre encontrar la vía que le lleve a la meta. La búsqueda es uno de los subcampos de la IA que se ocupan de encontrar las secuencias de acciones que permiten alcanzar las metas de un agente. Este tipo de agentes puede ser menos eficiente que los agentes reflejos, pero en cambio son mucho más flexibles.

1.1.2.16.3.4. Agentes basados en utilidad.

Las metas no bastan por si mismas para generar una conducta de alta calidad. Las metas permiten establecer una tajante distinción entre estados útiles e inútiles, en tanto que mediante una medida de desempeño más general sería posible establecer una comparación entre los diversos estados del mundo.(o secuencia de estados) de acuerdo a cómo exactamente resultarían más útiles al agente en caso de poder lograrlos.

Por lo tanto, la utilidad es una función que correlaciona un estado y un número real mediante el cual se caracteriza el correspondiente grado de satisfacción. La completa especificación de la función de utilidad permite la toma de decisiones racionales en dos tipo de casos en los que las metas se encuentran con problemas. El primero, cuando el logro de algunas metas implica un conflicto, y sólo algunas de ellas se pueden obtener, la función de utilidad definirá cuál es el compromiso adecuado por el que hay que optar.

Segundo, cuando son varias las metas que el agente podría desear obtener, pero no existe la certeza de poder lograr ninguna de ellas, la utilidad es una vía para ponderar la posibilidad de tener éxito considerando la importancia de las diferentes metas.

1.1.2.16.4. AMBIENTES..

Los ambientes en los cuales se puede desenvolver un agente pueden ser divididos en diversos tipos, sin embargo, se debe considerar que cada ambiente puede clasificarse de diferente manera dependiendo de cómo se conceptúe al ambiente y al agente:

- Accesibles y No Accesibles. Si el aparato sensorial de un agente le permite tener acceso al estado total de un ambiente, se dice que es accesible a tal agente. Un ambiente es realmente accesible si los sensores detectan todos los aspectos relevantes a la elección de una acción.
- Deterministas y No Deterministas. Es aquel ambiente en el que el estado siguiente del mismo se determina completamente mediante el estado actual y las acciones escogidas por los agentes.
- Estáticos y Dinámicos. El ambiente se comporta de forma dinámica cuando existe la posibilidad de que el ambiente sufra modificaciones mientras el agente se encuentra deliberando, de lo contrario se considera estático.
- Discretos y Continuos. El ambiente es discreto si existe una cantidad limitada de percepciones y acciones distintas y claramente discernibles.

1.2. PARADIGMA ORIENTADO A OBJETOS.

1.2.1. Definición.

La programación orientada a objetos es un paradigma de la ingeniería de software, cuya filosofía es un modelado de software basado en objetos del mundo real. La programación orientada a objetos puede ser vista también, como una forma diferente de organización de programas que facilita conceptualizar el mundo real mientras se desarrolla una aplicación. Por ello, la programación orientada a objetos puede reducir

significativamente la complejidad del problema en comparación con otras técnicas anteriores como la programación estructurada.

La programación orientada a objetos facilita el mantenimiento de un programa al organizarlo en unidades relativamente independientes. Además, la programación orientada a objetos permite una mejor reusabilidad del código escrito a través de la herencia.

La programación orientada a objetos no tiene la intención de reemplazar las diversas prácticas y buenas costumbres programáticas, tales como las de la programación estructurada, sino al contrario, las usa llevándolas a una concepción más elevada por medio de los lenguajes híbridos.

1.2.2. Principales Conceptos.

1.2.2.1. Objeto.

Un Objeto es un tipo de dato complejo formado por funciones y otros datos. Los datos del objeto representan las características que lo definen, mientras que las funciones que manipulan los datos representan las acciones que puede efectuar dicho objeto. Los datos de un objeto también son conocidos como características, datos miembro o variables miembro; de igual forma, a las funciones de un objeto se les llama acciones, métodos o funciones miembro. Cualquiera de estos nombres tanto para los datos como para las funciones de un objeto puede ser considerado correcto, ya que no existe actualmente consenso alguno que sea generalmente aceptado. Por esta razón, en adelante se usa cualquiera de ellos indistintamente. Al conjunto de datos y funciones de un objeto se les conoce como miembros o propiedades de un objeto.

A los argumentos requeridos por un objeto para ejecutar una acción se les llama mensajes. A la secuencia requerida de mensajes para ejecutar la acción de un objeto se le llama protocolo. Por ello, muchos teóricos afirman que un objeto se comunica con el

exterior mediante un protocolo que es marcado por el paso de mensajes a través de sus acciones.

1.2.2.2. Propiedades.

Ciertas propiedades deben estar presentes de alguna u otra forma en un lenguaje de programación que se declare como orientado a objetos:

- Ocultamiento de la información.
- Construcción.
- Destrucción.
- Herencia .
- Polimorfismo.

El ocultamiento de la información es la propiedad que evita que las características intrínsecas de un objeto puedan ser modificadas directamente por el exterior, de esta forma, un objeto podrá ser utilizado de manera simplificada mediante acciones que son públicas, es decir accesibles a todos. La abstracción de datos es un concepto muy relacionado con el ocultamiento de la información y se refiere a la utilización de tipos complejos de datos sin importar la forma en que ha sido definida su estructura interna, ni la forma como han sido implementadas las funciones. En cierta manera, existe abstracción de datos cuando es posible el ocultamiento de la información, pues al quedar oculta la estructura de un objeto, nos ocupamos entonces de hacer funcionar éste objeto, en lugar de preocuparnos en cómo es que funciona.

La herencia es la definición de un objeto en función de otro u otros objetos. La herencia permite crear una relación jerárquica de padres e hijos entre los objetos, en las que los objetos hijos pueden asumir todas las características y acciones de sus padres. Otro nombre con el que se conoce a la herencia, es el de derivación, por lo que se dice que una clase puede ser derivada de una o varias clases base.

El polimorfismo es la propiedad que permite que un objeto responda diferente a una misma acción dependiendo del mensaje que le es enviado. El polimorfismo es posible

porque pueden existir cualquier cantidad de funciones con el mismo nombre, siempre y cuando se diferencien en el tipo o número de parámetros o por el tipo de dato retornado.

Por estas características al polimorfismo también se le ha conocido como sobrecarga de una función. El polimorfismo puede presentarse desde nivel global, es decir, accesible a cualquier parte de programa, hasta el nivel de operadores, tales como la suma, resta o similares.

Un constructor es un método asociado a un objeto y que es ejecutado al momento de creación del objeto. En los lenguajes orientados a objetos, generalmente existe un constructor por omisión para cada objeto, tal constructor es el encargado de crear los datos usados por el objeto. Regularmente la definición de un constructor no es obligatoria y basta con el constructor por omisión, pero existen otras ocasiones en las que sería deseable inicializar algunas variables o realizar un proceso previo a la utilización del objeto.

Un destructor es un método asociado a un objeto y que es ejecutado al momento de destrucción del objeto. El funcionamiento de un destructor es análogo al del constructor. Existe un destructor por omisión cuyo objetivo es el liberar la memoria ocupada por los datos del objeto. Generalmente la definición de un destructor no es necesaria y basta con el destructor por omisión. Pero existen ocasiones en las que se desearía hacer un proceso previo a la liberación de la memoria ocupada por los datos, como por ejemplo, en objetos que usan memoria dinámica, en tales casos el uso de un destructor puede convertirse en algo casi obligatorio.

1.3. ARQUITECTURA CLIENTE/SERVIDOR.

1.3.1. Definición.

Este enfoque permite diseñar una aplicación de tal forma que los componentes funcionales de una aplicación son repartidos de manera que les permite estar dispersos, y

ser ejecutados en múltiples plataformas de cómputo diversas compartiendo el acceso a uno o más repositorios de datos compartidos.

Según IBM es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo, o a cualquier otro recurso dentro del grupo de trabajo y/o a través de la empresa en plataformas propietarias y no propietarias.

El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por las estaciones de trabajo inteligentes o clientes resultan en un trabajo realizado por otras computadoras llamadas servidores.

Cliente. Es el iniciador de un requerimiento de servicio; el requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través del LAN o del WAN; la ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Servidor. Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de LAN's o WAN's para proveer de múltiples servicios a los clientes, tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

Según Sybase la arquitectura cliente/servidor es la liga de aplicaciones y servicios de hardware y software con un sistema de información. Computadoras de alto rendimiento y servidores mueven información a través de la red y adonde más se necesite, el procesamiento de esta se divide en funciones separadas que comunican al cliente con el servidor para incrementar el tiempo de respuesta y el número de usuarios que pueden accesar a una aplicación o a los datos.

Este tipo de arquitectura balancea las responsabilidades de los usuarios finales con el control del sistema de información. Los usuarios finales pueden escoger cualquier tipo de computadora y cualquier tipo de interfaz para poder accesar a los datos; los departamentos que constituyen las empresas pueden responder a los requerimientos del negocio mucho más rápido que si dependieran solamente de los recursos de un

mainframe. Al mismo tiempo los encargados de sistemas pueden mantener el control centralizado de los datos, aun cuando estos se encuentren distribuidos en varias computadoras.

1.3.2. Importancia.

Hoy en día las empresas tiene que adecuarse con mayor facilidad a los requerimientos del mercado, las tendencias de este y a la vez disminuir sus costos operativos para ser cada vez más competitivas, la arquitectura cliente/servidor se desarrollo con la idea de explotar las capacidades naturales de los equipos y de los usuarios finales, además de proporcionar información veraz y oportuna a aquellas personas encargadas de la toma de decisiones.

Al tener un mejor tiempo de respuesta, las empresas pueden controlar mejor sus aplicaciones; al descentralizar las soluciones aplicativas e implantarlas más cerca del usuario, este puede controlar sus propios requerimientos de información. El área de sistemas dentro de esta arquitectura sufre un cambio radical ya que pasa a ser el proveedor de los servicios de cómputo, en lugar de ser el controlador de los mismos. De esta forma, el dueño de la información es quien la utiliza.

El costo de un sistema de esta clase dependerá del grado de automatización que se tenga; por lo que algunas empresas no ven los resultados de inmediato, y aún peor, algunas ni siguiera se atreven a realizar este gasto. En última instancia, las empresas eficientes serán las que permanezcan en el mercado y sin lugar a dudas estas empresas considerarán a la información como un punto clave de su éxito.

1.3.3. Componentes.

Conceptualmente los componentes de la Arquitectura Cliente/Servidor son: el clientes, el servidor y la red. A continuación se ofrece una breve descripción de cada uno de ellos.

1.3.3.1. Cliente.

Es la entidad por medio de la cual un usuario solicita un servicio, realiza una petición o demanda el uso de recursos. El cliente se encarga de realizar el FRONT END que es la aplicación que interactúa con el usuario, básicamente es la presentación de los datos y/o información al usuario, en un ambiente gráfico.

1.3.3.1.1. CARACTERÍSTICAS DEL CLIENTE.

- Es el medio de enlace y/o comunicación entre el usuario y la computadora.
- Es la entidad que requiere o solicita el servicio.
- Requiere el uso de los recursos de la computadora para realizar cualquier actividad.
- Es el medio por el cual se envía la solicitud y se reciben los resultados o notificaciones del servidor.
- Un cliente puede ser una estación de trabajo UNIX o una computadora personal

1.3.3.2. Servidor.

Es la entidad que provee un servicio, ejecuta el procesamiento de datos, aplicaciones y manejo de la información o recursos. En el servidor se realiza el BACK END que es la parte destinada a recibir las solicitudes del cliente y donde se ejecutan los procesos.

1.3.3.2.1. CARACTERÍSTICAS DEL SERVIDOR.

- Responde a las peticiones de los clientes.
- Tiene gran capacidad de almacenamiento y rapidez.
- Provee el uso de los recursos y servicios a los clientes.
- Es el administrador de los recursos de red.
- Tiene capacidad de procesamiento transaccional.
- Puede fungir como cliente de otros servidores.

- Contiene los datos, programas, aplicaciones, software e información.
- Realiza procesos como accesar, organizar, almacenar, actualizar y manejar datos o recursos compartidos.

1.3.3.3. Red.

Debe haber una liga de comunicación entre el cliente y el servidor por medio de la cual el requerimiento del cliente y la respuesta del servidor se comuniquen. Esta liga de comunicación puede tomar varias formas, lo único importante es la libertad de intercambiar información.

1.3.3.3.1. CARACTERÍSTICAS DE LA RED.

- Conecta las diferentes bases de datos.
- Permite procesar y accesar la información.
- Interactúa con las interfaces del usuario.

CAPÍTULO II. COMPONENTES CONCEPTUALES DEL EDIFICIO INTELIGENTE.

2.1. EVOLUCIÓN HISTÓRICA DE LA TECNOLOGÍA DE EDIFICIOS INTELIGENTES.

Los controles automáticos de temperaturas de tipo termostato para calefacción aparecieron en el mercado hace más de 100 años, desde 1880 este control relativamente simple, estuvo funcionando de forma satisfactoria durante 50 años aproximadamente; mientras que en los grandes edificios comenzaban a incorporarse sistemas de calefacción y ventilación centralizada de aire acondicionado. Después de la segunda guerra mundial, las construcciones con ventanas fijas hicieron que los sistemas de aire acondicionado fueran una absoluta necesidad. Los métodos neumáticos para la medición, transmisión de señales y respuestas de control llegaron a ser un estándar, con tal tecnología era sencillo mantener y operar la climatización de un edificio dentro de límites aceptables.

Los primeros sistemas de control de edificios basados en técnicas electrónicas aparecen en 1960 junto con el aumento en la complejidad de las instalaciones técnicas y del tamaño de los edificios, lo que hizo necesario centralizar la señalización de desperfectos, anomalías y alarmas. Así, surgieron los grandes tableros con esquemas sinópticos que representaban las instalaciones con indicadores luminosos, instrumentos de medidas, interruptores, etcétera, que permitían supervisar, incluso a distancia, los equipos e instalaciones.

La evolución de la Electrónica e informática, así como los conocimientos adquiridos en la automatización de otras áreas (fábricas, centrales eléctricas, etcétera), permitieron desarrollar aplicaciones específicas, mejorar los sistemas de transmisión de señales, buscar sensores especiales y realizar programas encaminados a la optimización

del control de edificios. Surgieron sistemas que han garantizado un servicio de alta fiabilidad y precisión en cuanto al control del funcionamiento de complejos sistemas electromecánicos.

La crisis del petróleo de 1973 y la consiguiente escalada de los precios de la energía propiciaron la necesidad de instalar sistemas capaces de reducir su consumo, esta reducción llegó a ser un objetivo crítico en la operatividad de los edificios existentes y de nueva construcción. El casi simultáneo desarrollo de los microprocesadores proporcionó las herramientas necesarias para la elaboración de sistemas de control con costos relativamente bajos, lo que condujo a que se implantaran cada vez más.

Para 1980 el fin principal fue la conservación de la energía mientras los precios del crudo continuaron creciendo, posteriormente, la contención y disminución de éstos precisó restar protagonismo a tal argumento; en la presente década, el énfasis se ha orientado hacia una eficiente operatividad del edificio, incluyendo controles más consistentes de temperatura y una ventilación y acondicionamiento efectivos. Para cubrir la necesidad de un control de edificios eficiente, los sistemas han pasado de las tecnologías basadas en métodos neumáticos (analógicos) a las de control digital directo.

Desde 1980 se maneja constantemente el concepto de calidad, esto conlleva a la optimización de recursos humanos y materiales. Al desarrollarse tal concepto, se inicia la era del Edificio Inteligente, ya que uno de los factores principales es ahorrar recursos financieros y humanos, esto es, ser eficientes en todas las áreas de la organización.

En la presente década, con la ayuda de la alta tecnología podemos identificar y definir al Edificio Inteligente. Existe toda una efervescencia tecnológica que repercute en el perfeccionamiento de los requisitos para la construcción del mismo. Hoy no se concibe ningún edificio nuevo, destinado a cualquier tipo de servicios, que no incorpore sistema automatizados para el control de su explotación y seguridad. El continuo avance de la microelectrónica y la informática así como la disminución relativa de sus costos, hará que en un futuro estas técnicas se apliquen en forma habitual en las viviendas.

Un Edificio Inteligente no será novedoso para el año 2000, ya que dentro de poco tiempo dejará de ser una moda o un lujo. La empresa que no considere este concepto dentro de su planeación estratégica, para tener una mayor competitividad, estará destinada al fracaso por los altos costos que origina el desarrollarse en una construcción de tipo tradicional.

2.2. CONCEPTO DE EDIFICIO INTELIGENTE.

Los criterios que se utilizan para evaluar cuando hablamos de un Edificio Inteligente son varios y distintos a los considerados hace unos años. El bajo costo en los servicios es un factor de suma importancia sin embargo, se está dando un mayor énfasis al costo de uso de los servicios durante la vida del edificio, es decir, el costo-beneficio es determinante para hacer una consideración de la edificación de un Edificio Inteligente; sin perder de vista la capacidad para satisfacer las necesidades de la organización y su contribución a una mayor productividad del usuario y la empresa, en términos de efectividad organizacional.

Un edificio es inteligente, si cumple con los requerimientos básicos para lograr en conjunto su óptima utilización. No todos los edificios que cuenten con equipos modernos y sofisticados de automatización, podrán ser considerados como inteligentes. En otras palabras son Edificios Inteligentes aquellos en los que el ingenio del hombre crea para los dueños y usuarios, sistemas de aprovechamiento de recursos que les permiten alcanzar sus metas, trazadas dentro de parámetros de eficiencia, comodidad, conveniencia, seguridad, flexibilidad y rentabilidad de acuerdo a sus necesidades. Otros definiciones aceptables son:

- "Un Edificio Inteligente es aquel que cuenta con las características necesarias para optimizar la eficiencia del mismo, permitiendo simultáneamente una administración efectiva de recursos con un costo mínimo en el menor tiempo"[Baker95].

- "Un Edificio Inteligente es aquel que provee un ambiente productivo y eficiente de trabajo a través de la optimización de cuatro de sus elementos básicos: estructura, sistemas, servicios y administración; relacionándolos entre sí para un mejor rendimiento"[IBI90].

- Edificio Inteligente es aquel que optimiza, aprovecha o procesa al máximo los recursos con que cuenta a través de las herramientas necesarias para administrar los sistemas ambientales y su coordinación para brindar la comodidad y seguridad requeridos por los usuarios.

2.3. REQUISITOS QUE DEBEN REUNIR LOS EDIFICIOS PARA SER INTELIGENTES.

Tres factores importantes a integrar en un Edificio Inteligente son:

2.3.1. Flexibilidad del Edificio Inteligente.

2.3.2. Integración de servicios.

2.3.3. Diseño interior y exterior.

2.3.1. Flexibilidad del edificio inteligente.

Para que un edificio pueda considerarse flexible, debe ser concebido de forma dinámica y tener la capacidad de proporcionar a sus ocupantes la suficiente versatilidad. Por ello se debe prever lo siguiente:

Estructura.	Es la que tiene un mayor ciclo de vida, entre 50 y 60 años. Se incluyen todos los elementos comunes (pisos, muros, ventanas, etcétera) pero además, necesita incluir plafones, ductos adicionales para comunicaciones, áreas para los equipos de control y telecomunicaciones, espacio para colocar piso falso, analizar la orientación de la estructura para aprovechar la luz del sol, entre otros.
Servicios.	Abarcan los sistemas que se integran al caparazón y que generalmente son componentes tecnológicos, cuyo ciclo de vida es de entre 15 y 20 años, tales como: sistema eléctrico, aire acondicionado, sistema hidráulico y sanitarios, elevadores y escaleras eléctricas, telecomunicaciones e informática, sistemas de control y seguridad, etcétera.
Acabados.	El ciclo de vida de éstos oscila entre 10 y 15 años. Comprende los elementos de carácter superficial, acabados de pisos, muros, techos, cancelería, etcétera.
Mobiliario.	Debe adaptarse a los cambios requeridos, por lo que es conveniente considerar muebles modulares y desarmables.

Tabla. 2.1. Elementos que denotan la flexibilidad de un Edificio Inteligente.

Un edificio es flexible cuando estos cuatro elementos son independientes cada uno entre sí, es decir, si es necesario realizar un cambio en los servicios no se debe realizar ninguna modificación en la estructura; o con mayor razón, en caso de que se requiera una redistribución de un departamento no se deberá realizar ninguna modificación en los servicios.

La flexibilidad del Edificio Inteligente se refiere a su capacidad para satisfacer necesidades futuras de los usuarios, entre las que destacan la posibilidad de modificar distribuciones físicas de personas y departamentos en una organización, así como posibilitar los cambios de mobiliarios, equipo o funciones por área. Desde luego debe existir una flexibilidad razonable que permita las modificaciones con costos mínimos,

pero que no implique una inversión inicial extraordinaria. En general, los sistemas modulares, el mobiliario del mismo tipo, la iluminación uniforme por zonas, el acondicionamiento uniforme de aire en áreas especiales, y los demás servicios bajo criterios estandarizados permiten la flexibilidad al cambio.

Una de las formas en que se suele subdividir un Edificio Inteligente, es tomar como criterio la distinta duración del ciclo de vida de sus elementos.

Elementos.	*Ciclo de vida en años.*
PC's.	5
Computadora central.	10
Servicio de cableados.	15
Edificación.	20

Tabla 2.2. Ciclo de vida de los elementos del Edificio Inteligente.

2.3.2. Integración de servicios.

El concepto de integración de servicios no es nuevo. Desde hace algunos años ya se hablaba de éste sin ningún éxito, pero a raíz del desarrollo de la tecnología en los campos del control, informática y telecomunicaciones, ha tomado mayor importancia, hasta volverse fundamental dentro de la tecnología de edificios inteligentes.

La integración de servicios permite establecer el momento a partir del cual un edificio puede ser considerado inteligente, así como diferenciar entre distintos grados de inteligencia tecnológica. Los servicios ofrecidos se pueden dividir en cuatro grupos:

2.3.2.1. Automatización del edificio.

2.3.2.2. Automatización de la función informática.

2.3.2.3. Telecomunicaciones.

2.3.2.4. Planificación del espacio.

2.3.2.1. *Automatización del edificio.*

Compuesta por tres sistemas elementales:

2.3.2.1.1. Sistema básico de control.
2.3.2.1.2. Sistema de seguridad.
2.3.2.1.3. Sistema de ahorro de energía.

2.3.2.1.1. *SISTEMA BÁSICO DE CONTROL.*

Es aquel que permite monitorear el estado de las instalaciones y actúa de acuerdo a lo programado, evitando fallas dentro del funcionamiento. Asimismo es el responsable de mantener los distintos grados de comodidad y de llevar las estadísticas de mantenimiento para cada equipo. Se conforma por los siguientes subsistemas:

- Instalaciones de aire acondicionado, calefacción y ventilación.
- Instalación eléctrica.
- Instalación hidrosanitaria.
- Elevadores y escaleras eléctricas.
- Suministros de gas y electricidad.
- Accesos a estacionamientos.

2.3.2.1.2. *SISTEMA DE SEGURIDAD.*

Dentro de la seguridad existen dos aspectos primordiales, la protección de las personas y la del patrimonio. Por ello se debe instalar un sistema que abarque ambos.

Dentro de la protección de las personas destacan:

- Detección de humo y fuego.
- Detección de fugas de gas.
- Detección de fugas de agua.
- Monitoreo de equipo para la extinción de fuego.
- Red de rociadores.
- Absorción automática de humo.
- Señalización de salidas de emergencia.
- Voceo de emergencia.

Por su parte la seguridad patrimonial debe incluir:

- Circuito cerrado de televisión.
- Vigilancia perimetral.
- Control a accesos.
- Control de rondas de vigilancia.
- Intercomunicación de emergencia.
- Seguridad informática.
- Detector de movimientos sísmicos.
- Detectores de presencia.

2.3.2.1.3. SISTEMA DE AHORRO DE ENERGÍA.

Con el sistema básico de control del Edificio Inteligente, ahorrar en el consumo de energía es prácticamente implícito, ya que los equipos serán programados para que operen en situaciones de máximo rendimiento, lo cual se reflejará en la productividad. Cabe enunciar como elementos importantes:

- Zonificación de la climatización.
- Intercambio de calor entre zonas, inclusive con el exterior.
- Uso activo de la energía solar.

- Identificación del consumo.
- Control automático de la iluminación.
- Control de horarios para el funcionamiento de máquinas y equipo.
- Control de ascensores.

2.3.2.2. *Automatización de la función informática.*

La correcta selección de la tecnología involucrada en este aspecto, dará como resultado un incremento en la productividad laboral, proporcionando un importante beneficio en la gestión de la función informática del Edificio Inteligente. Otro factor relevante es la eficiencia en la obtención de la información, reduciéndose el tiempo que transcurre del origen al destino de la misma, lo que permite tomar decisiones con oportunidad. Dentro de los servicios, destacan:

- Acceso a servicios telefónicos de tecnología avanzada.
- Integración de redes de área local (LAN por Local Area Network).
- Estaciones de trabajo integradas.
- Programación de actividades.
- Acceso a bases de datos internas y externas.
- Integración de periféricos: plotters, scanners, impresoras, entre otros.

2.3.2.3. *Telecomunicaciones.*

En nuestro país se requiere un análisis especial en esta área, ya que no existe una infraestructura que permita tener avanzados servicios de telecomunicaciones, sin embargo se debe tratar de contar con:

- Un cableado integral de telecomunicaciones.
- Una central telefónica de conmutación privada.
- Equipos de conexión con redes externas.

Es importante considerar que la integración de cableado estructurado para las telecomunicaciones evitará problemas futuros, en lugar de tener un cableado para voz, datos seguridad y control de forma independiente, se contará con un cableado único, lo que disminuye los costos. Una central telefónica de conmutación privada permitirá que se deje a la compañía proveedora del servicio telefónico únicamente el suministro de éste y no su administración.

Los principales servicios de telecomunicaciones a proporcionar son:

- Telefonía avanzada.
- Transmisión de datos.
- Facsímil, telefax, videotexto.
- Correo electrónico.
- Videoconferencia.
- Comunicación vía satélite.

Un nuevo enfoque es el compartir servicios comunes a todos los usuarios y que para algunos su adquisición representa un alto costo. Tales como:

- Centro de mensajes.
- Correo electrónico.
- Salas de videoconferencia.
- Uso de la red de computadoras.
- Acceso a telepuertos.
- Servicio de CAD.
- Pool de modems, fax, telex e impresoras.

2.3.2.4. *Planificación del espacio.*

Esta área ha cobrado importancia en los últimos años, ya que incide directamente en el bienestar físico del trabajador. El objetivo es lograr un ambiente que facilite el trabajo de los usuarios, para lo cual se debe considerar:

- Posibilidad de zonificar el aire e iluminación.
- Planificación y distribución de las áreas de oficinas.
- Ergonomía en el puesto de trabajo, mobiliario, luz, aislamiento acústico, colores, etcétera.
- Creación de un ambiente seguro, proporcionando eficientes sistemas de seguridad, medios adecuados de evacuación, escaleras de emergencia, etcétera.

2.3.3. Diseño interior y exterior.

En este rubro existen dos grandes ámbitos:

- Diseño interior: relacionado con los acabados interiores, el mobiliario, la ergonomía y planificación del espacio; con el fin de crear ambientes con la mayor comodidad para estimular la actividad intelectual.
- Diseño exterior: relacionado con el diseño arquitectónico, no hay que olvidar que es la proyección de la imagen de la entidad y que debe ser congruente con el diseño interior.

La relación entre el criterio de diseño y los edificios durante las últimas décadas, han sido los siguientes:

- 1960, eficiencia operacional y organizativa.
- 1970, reducción de los costos.
- 1980, búsqueda de la calidad.
- 1990, fomento de la creatividad y del trabajo en equipo.

2.3.4. Grados de inteligencia del edificio inteligente.

Un edificio puede ser denominado inteligente cuando, además dispone de sistemas basados en tecnologías de la información que permiten la oferta de servicios y aplicaciones de automatización de la actividad y de telecomunicaciones a través de los que se genera un importante valor agregado.

Los factores antes mencionados, van perfectamente entrelazados y combinados por un factor de relevancia que es la rentabilidad. Tanto por lo que representan los servicios de automatización como los que dan soporte a la actividad y telecomunicaciones, se pueden distinguir distintos niveles en función de sus características técnicos e integración.

A continuación se clasifican los grados de inteligencia del Edificio Inteligente desde el punto de vista tecnológico.

Grado 1. ***Inteligencia mínima.***	Un sistema básico de automatización del edificio, el cual no está integrado. Existe una automatización de la actividad y servicios de telecomunicaciones, aunque no están integrados.
Grado 2. ***Inteligencia mediana.***	Tiene un sistema de automatización del edificio totalmente integrado. Sistemas de automatización de la actividad, sin una completa integración de las telecomunicaciones.
Grado 3. ***Inteligencia máxima.***	Los sistemas de automatización del edificio, de automatización de la actividad y telecomunicaciones se encuentran totalmente integrados.

Tabla 2.3 . Grados de inteligencia del Edificio Inteligente desde el punto de vista tecnológico

2.4. SISTEMAS DISTRIBUIDOS.

Un sistema de distribución se compone de los cables, adaptadores, y otro equipo de apoyo que permiten conectar teléfonos, terminales de datos, sistemas de energía, seguridad y dispositivos sensores de alarmas, para que tengan comunicación entre ellos. Un sistema de distribución consta de un número variado de productos, muchas veces de diferentes proveedores. Al mismo tiempo, la forma de arreglar todos los elementos descritos anteriormente en forma lógica, coherente y económica, dentro de un Edificio Inteligente o un conjunto de éstos, es dividiéndolo en más subsistemas, de acuerdo a sus necesidades.

El cableado estructurado es un sistema de distribución dentro y fuera del equipo que satisface de forma eficiente las necesidades, presentes y futuras, de servicios para los usuarios.

2.5. CARACTERÍSTICAS DE LOS SISTEMAS DE CONTROL INTELIGENTES.

Dentro de los esquemas de automatización de Edificios Inteligentes existen dos alternativas.

2.5.1. Control tradicional "Sistema cerrado o propietario".

2.5.2. Control distribuido "Sistema de protocolos abiertos".

2.5.1. Control tradicional "Sistema cerrado o propietario".

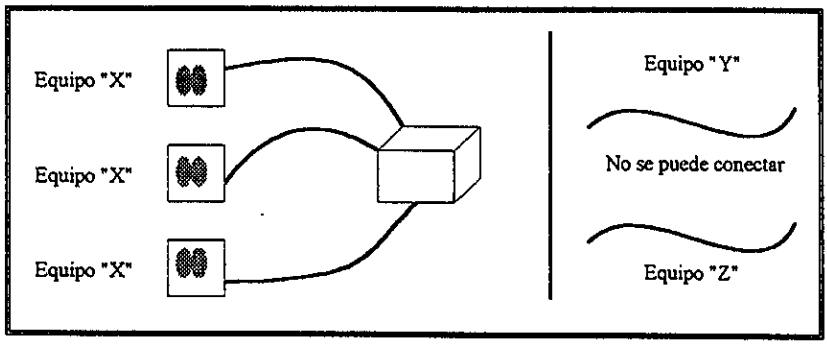


Figura 2.1. Representación esquemática de la incompatibilidad de un sistema cerrado o propietario con equipos que no son del mismo tipo que el propietario.

Basado en la estrategia de un solo fabricante, este método sujeta al usuario a comprar únicamente productos de un fabricante o a tener varios sistemas pero que no trabajan en conjunto, porque los equipos de uno, no se comunican con los del otro.

Es muy difícil que exista un fabricante con equipos óptimos para todos los sistemas, que además operen coordinados y que cumplan con un costo-beneficio razonable. En este caso las modificaciones y expansiones del sistema de automatización están atadas a las nuevas respuestas que solamente el fabricante puede dar. Si para la automatización se decide utilizar diferentes proveedores de equipos propietarios (por ejemplo, un contratista para control de aire acondicionado, uno para iluminación, etcétera) entonces se pagará un alto costo de instalación y estarán muy reducidos los servicios que se esperan del edificio.

2.5.1.1. Características de la instalación del sistema de control tradicional.

- Cableado, materiales de instalación y mano de obra para cada sistema.
- Equipo supervisor o de control individual para cada sistema.
- En el caso que los controles sean centralizados, se requiere llevar cable desde todos los puntos de control hasta donde se encuentre la máquina central.

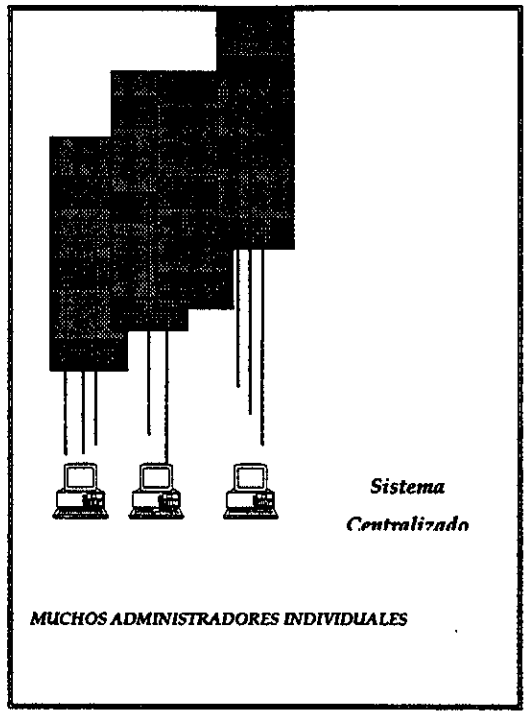


Figura 2. 2. Sistema de control tradicional. Cada sistema tiene un administrador individual.

2.5.2. Control distribuido "Sistemas de protocolos abiertos".

En la automatización de Edificios Inteligentes se aplica la siguiente fórmula:

integrar una red de equipos de diferentes fabricantes que saben comunicarse entre sí y que además tienen la capacidad de autocontrol.

De esta manera se selecciona siempre el mejor producto para cada aplicación, con una variedad de más de 700 compañías a nivel mundial que siguen el mismo esquema.

2.5.2.1. *Características de la instalación del sistema de control distribuido.*

- Se comparte la instalación y el cableado para todos los equipos.
- Trabajan coordinados y en conjunto.
- Frecuentemente se requiere un solo equipo supervisor.
- Flexibilidad para agregar nuevos equipos.

2.5.2.2. *Tecnología abierta.*

Se utiliza la tecnología abierta para la implantación personalizada de redes de coordinación de sistemas ambientales. Esta tecnología de vanguardia permite realizar interfaces con equipos convencionales para:

- Tomar información del medio ambiente (sensores).
- Actuar sobre mecanismos eléctricos o mecánicos.
- Tomar decisiones locales.
- Comunicación en red.
- Diversos canales de comunicación.

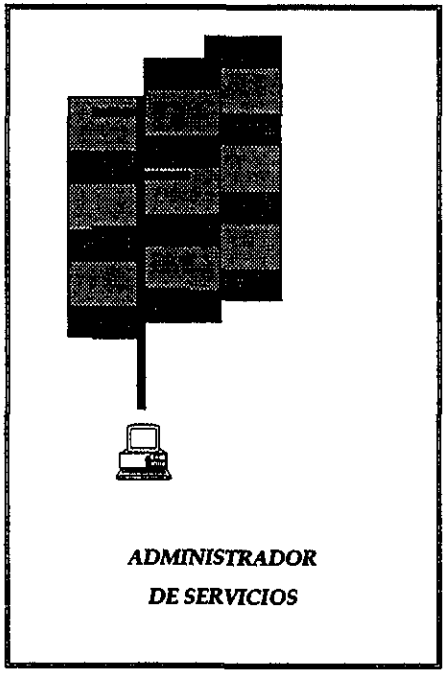


Figura 2.4. Sistema de control distribuido. Un equipo se encarga de comunicar diversos ambientes y coordinar los resultados de los distintos sensores que le reportan.

Esta tecnología es justificable ya que permite realizar control distribuido, es decir, cada zona que requiera control tendrá un módulo inteligente especializado únicamente en sus tareas locales, pero con capacidad de comunicación con otros.

2.6. VENTAJAS DE UN SISTEMA DE CONTROL DISTRIBUIDO CONTRA UNO CENTRALIZADO.

Comparativamente, un sistema de control distribuido contra uno centralizado presenta como ventajas la escalabilidad y expansibilidad de sus componentes, así como baja probabilidad de sufrir fallas y menor costo por modificaciones.

Otras, que por su relevancia se detallan a continuación, son:

2.6.1. Interoperabilidad.
2.6.2. Modularidad.
2.6.3. Flexibilidad.
2.6.4. Interfaces con otros equipos.
2.6.5. Canales de comunicación.
2.6.6. Instalación de la red de control.
2.6.7. Confiabilidad.
2.6.8. Autodiagnóstico.
2.6.9. Mantenimiento y servicio.

2.6.1. Interoperabilidad.

El término interoperabilidad se define como la facilidad de poder trabajar bajo diferentes plataformas, con equipos y proveedores diversos, así como con esquemas de instalación e interconexión con una coordinación capaz de enlazarlos para que respondan como un sistema integrado.

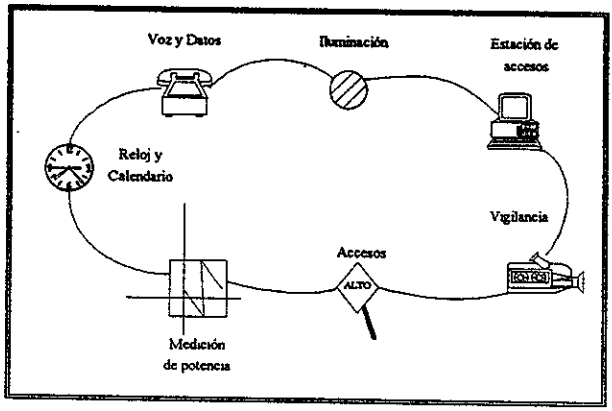


Figura 2. 5. Interoperabilidad. Facilidad para poder trabajar con diferentes plataformas, equipos y proveedores.

El concepto de interoperabilidad entre equipos involucra la interconexión de diversos sistemas mediante un esquema que les permita intercambiar información entre sí, brindando una comunicación confiable y eficiente.

2.6.2. Modularidad.

Dado que la tecnología avanza a pasos agigantados, y a que cada vez más fabricantes de hardware centran su atención sobre el desarrollo de productos bajo esta tecnología de automatización, es un hecho que vendrán módulos de control e interfaz más avanzados. La ventaja de tal tecnología es que crece bajo estándares, por lo cual todos lo avances que se realicen mantendrán compatibilidad con los actuales. La modularidad tiene otra gran ventaja, en caso de falla, la detección del error es casi inmediata y por lo tanto con la sustitución de un módulo de respaldo se evita la pérdida de tiempo. Al modular las áreas también se obtienen beneficios en cuanto a otros componentes como cableado, salidas eléctricas, ductos de aire, etcétera.

2.6.3. Flexibilidad.

El esquema de control distribuido, permite la expansión de la red de control y monitoreo sin problemas. Esta es una gran ventaja comparada contra los sistemas de control centralizados, en los que al momento de crecer, es necesario cambiar el equipo totalmente, además de que una falla en el sistema central repercute en toda la red de control. Este último aspecto tiene poca repercusión en el esquema de control distribuido, ya que los módulos son independientes, y dado el precio de los mismos, se puede justificar en algunos casos inclusive el tener uno de respaldo para un dispositivo de vital importancia.

2.6.4. Interfaces con otros equipos.

La tecnología que se aplica en la automatización se clasifica como arquitectura abierta, pues permite la integración de equipos fabricados por distintos vendedores. Existen alrededor del mundo cientos de empresas de hardware que ofrecen dispositivos compatibles con la tecnología utilizada. Ejemplos de sistemas con los cuales se desea establecer comunicación son: tarjetas microcontroladoras, lectoras de tarjetas, sensores de presencia, humo, humedad, alcalinidad, intensidad luminosa, etcétera.

2.6.5. Canales de comunicación.

La transferencia de información con los módulos de control, puede realizarse con todos los medios de comunicación disponibles: cables de par trenzado, fibra óptica y coaxial. Inclusive, en caso de no querer incurrir en gastos por modificación a las instalaciones de cableado, se tiene la posibilidad de utilizar la misma línea de corriente alterna como medio de transmisión de datos.

Las anteriores facilidades permiten prácticamente la automatización bajo cualquier esquema y en cualquier campo.

2.6.6. Instalación de la red de control.

La variedad en módulos de control, así como los medios de comunicación existentes y la gran variedad de dispositivos diseñados por terceros para realizar las interfaces, permiten que la instalación de la red de control se realice en aproximadamente el 20% del tiempo de instalación normal.

2.6.7. Confiabilidad.

Empresas como Motorola y Toshiba respaldan la tecnología empleada. La red de control fue diseñada para funcionar bajo diferentes esquemas, inclusive para los de trabajo más rudo, como son los de la industria.

2.6.8. Autodiagnóstico.

Dado que los módulos de control utilizados se basan en microprocesadores, y existe comunicación con un Sistema Inteligente, se facilitan esquemas de autodetección de

fallas en la red de control. Por mencionar un esquema básico para detección de falla en un módulo, se solicita información por parte del Sistema Inteligente a cada uno de los módulos por determinado tiempo, y de esta manera se comprueba si todos se encuentran activos.

2.6.9. Mantenimiento y servicio.

Cuando el control se realiza sobre dispositivos que se degradan con respecto al tiempo (lámparas), o cuya vida de operación depende del número de veces que se ha utilizado (contactos, relevadores de estado sólido) se puede llevar un registro de tales parámetros en la estación de supervisión, de esta forma se puede realizar un mantenimiento preventivo y evitar detener las actividades.

2. 7. SISTEMA INTELIGENTE.

El Sistema Inteligente o SI es el que controla y coordina los diferentes recursos de un edificio, se encuentra integrado por lo siguiente:

2.7.1 Módulos de control distribuido.
2.7.2 Estación de supervisión.
2.7.3 Red de recursos ambientales.

2.7.1. Módulos de control distribuido.

El *Sistema Inteligente* está constituido por una red de módulos de control independientes, distribuidos en el Edificio Inteligente, estos módulos poseen programas específicos para coordinar la operación de los sistemas conectados. La información generada y adquirida en cada módulo puede ser transmitida hacia otros módulos de la red para ser compartida y brindar a la red un esquema de operación en base a los reportes de cada sistema. Esta filosofía de distribución de información permite que las decisiones referentes a la administración de recursos sea tomada en función a la operación global de

los sistemas instalados. Es por ello que a un edificio que opera de esta forma se le cataloga como Edificio Inteligente.

El hecho de tener un esquema de control distribuido plantea el requerimiento de un sistema de comunicación, que permita el intercambio de información entre nodos de forma eficiente y confiable. Es aquí donde la tecnología juega un papel importante, proveyendo canales de transmisión de datos tan versátiles como los requerimientos de instalación lo demanden.

Los canales de comunicación basados en la tecnología abierta son los siguientes:

- línea de alimentación de 110V,
- radio frecuencia,
- cables de par trenzado,
- fibra óptica,
- etcétera.

El esquema de operación abierta permite la integración de equipos de diferentes proveedores en el denominado esquema abierto de interoperabilidad, de esta forma los equipos de diferentes fabricantes pueden trabajar en base a información recolectada por otros sistemas. Los conceptos de interoperabilidad y de sistemas abiertos han permitido integrar la información de los equipos instalados, alertar a los sistemas ante situaciones inesperadas, diseñar estrategias para ahorro de energía y programar eventos por horario y calendario mediante aplicaciones informáticas, que pueden ser operadas en forma sencilla por el usuario sin experiencia técnica.

Para el Sistema Inteligente, cada equipo instalado en el Edificio Inteligente es considerado como un objeto con variables de entrada y salida, encapsulando la complejidad de su operación y aislándolo del funcionamiento general de la red.

El Sistema Inteligente actúa como intérprete entre cada equipo y la propia red para que todo funcione como un conjunto armónico.

2.7.2. Estación de supervisión.

Para programar las actividades de los nodos y presentar al usuario (supervisor) lo que ocurre con los diferentes recursos, se instala una estación de supervisión. El equipo anterior es requerido para llevar una historia de los eventos que ocurren en el sistema y los recursos integrados al Sistema Inteligente. La finalidad es presentar al usuario la información respecto a la operación del edificio con una base de datos para evaluar su rendimiento. En caso de que el usuario lo deseara, el equipo anterior se podría expandir para formar una red de monitoreo en los lugares u oficinas en donde se solicitará el servicio para operar el Edificio Inteligente.

La estación de supervisión debe mostrar en forma gráfica el estado de la red de recursos ambientales (sistemas de iluminación, acondicionamiento ambiental, hidráulico, eléctrico, de seguridad, etcétera) del Edificio Inteligente.

Cuenta con una base de datos donde se almacenan todos los eventos ocurridos. Lo anterior permite obtener reportes para la evaluación, y planeación de estrategias, así como gráficas de la operación de algunos equipos para determinar su rendimiento y operación a través de los periodos del año. En este equipo se pueden programar los horarios y calendarios de las luces, fijar los puntos de operación del aire acondicionado, observar las gráficas de consumo diario y mensual, por mencionar sólo algunos puntos. Además, pueden conectarse varias estaciones de supervisión desde las cuales se realicen cambios a la operación del Edificio Inteligente. Una estación de supervisión es una herramienta que ayuda a planear el uso de recursos.

2.7.2.1. Características a considerar:

Las características que deben considerarse para la elección de una estación de supervisión son:

- Operación amigable, fácil de enseñar al personal de operación y mantenimiento.

- Interfase gráfica que permita un fácil acceso a los diferentes recursos y de manera sencilla ver el estado del Edificio Inteligente.

- Almacenamiento en base de datos, de los eventos ocurridos durante el día de los equipos integrados al Sistema Inteligente. El almacenamiento de información debe ser en forma diaria, mensual o anual (dependiendo de las necesidades del usuario).

- Obtención de reportes que ayuden a una buena planeación.

- Monitoreo distribuido que permite desde varios lugares obtener información de cómo está operando el Edificio Inteligente.

- Contraseña para restringir el acceso al Sistema Inteligente.

2.7.3. Red de recursos ambientales.

Está integrado por los sistemas de iluminación, eléctrico, seguridad, acondicionamiento ambiental, hidráulico, entre otros.

CAPÍTULO III. INGENIERÍA DETALLADA DEL EDIFICIO INTELIGENTE.

3.1 COMPONENTES DEL EDIFICIO INTELIGENTE.

El desarrollo tecnológico y las demandas de los usuarios finales han llevado a un cambio en el proceso tradicional de diseño y construcción del Edificio Inteligente. El arquitecto ha dejado de ser el diseñador que define desde su perspectiva todos los criterios, ahora un grupo de especialistas interdisciplinarios, que incluyen al Licenciado en Informática, debe ser el origen de dichos criterios y colaborar en las diversas fases del diseño.

La capacidad de obtener y dar información dentro del Edificio Inteligente está cambiando su operación y mantenimiento. Conforme aparecen nuevas tecnologías su diseño evoluciona y crecen las exigencias de los usuarios. Este cambio se refleja en sus componentes. Los servicios y suministros de los edificios de oficinas, requieren una optimización en los procesos de planeación, instalación, operación y mantenimiento. Por ello, se observan tendencias muy claras en los equipos nuevos: los sistemas de control se integran con sistemas informáticos, se buscan niveles básicos de inteligencia en los componentes de modo que los sensores tomen decisiones a nivel de microproceso sin involucrar a un sistema altamente complejo, se cuenta con un control distribuido pero a la vez de cooperación entre áreas independientes, el monitoreo de sistemas está migrando a una fuente de control distribuido con terminales de monitoreo remoto (dirección general, oficinas corporativas, etc.). Esto ocasiona mejor aceptación de los sistemas automatizados por parte del usuario, pero implica un mayor cuidado en el proceso de diseño.

El Edificio Inteligente debe ser funcional, esto significa que las soluciones aplicadas en su planeación y construcción correspondan con las necesidades de los usuarios y debe ser rentable para permitir su operación. En la práctica los sistemas de tecnología de punta lo son siempre que se les dé el uso adecuado. [EdiInt93]

Algunos de los suministros más importantes a establecer son:

SUMINISTROS.	
Aire acondicionado.	• Porcentaje de fresco. • Volumen de aire, temperatura promedio. • Distribución uniforme o localizada de salidas. • Humedad relativa. • Aportación de calor por otros sistemas.
Sistema eléctrico.	• *Acometidas.* • Plantas de emergencia. • Ubicación de centros de carga. • Distribución de circuitos, considerando cableado y canalizaciones. • Potencia por salida eléctrica. • Sistemas de tierra, pararrayos y de transferencias. • Sistemas de energía ininterrumpible (*UPS* por Uninterrumpible Power Supply).
Redes de cómputo.	• Cableado. • *Protocolos.*

Sistema de seguridad.	• Sistema de Control de Acceso. • Sistema de Detección y Extinción de Incendios. • Sistema de Detección de Intrusos. • *Sensores* y circuito cerrado de T.V.. • Conexión a sistemas externos (bomberos, policía). • Rotación del personal de seguridad. • Control de accesos y códigos de seguridad por áreas.
Hidráulico y sanitarios.	• *Sensores* de flujo y presión. • *Electroniveles.* • *Hidroneumáticos.* • Conexión a las áreas de agua. • Tratamiento de aguas negras. • Depósito de agua y sistemas de purificación.
Edificación.	• Sistema constructivo. • Fachadas, *acabados*, parteluces e imagen. • Decoración: armónica, selección cromática, ubicación de accesos, circulaciones, áreas operativas y *servicios.* • Pisos, techos y muros falsos. • Estacionamientos.
Iluminación.	• Sistemas de control. • Lámparas y *luminarias.* • Niveles de luz por áreas - tiempo. • Aprovechamiento de luz solar. • Ambiente y ubicación.

Tabla 3.1. Suministros a considerar en el Edificio Inteligente.

Por otra parte, uno de los aspectos más importantes en la composición del concepto de Edificio Inteligente es la redundancia en los sistemas. Este no es un concepto nuevo, pero si ha sido modificado en su filosofía de aplicación. Se ha establecido la llamada filosofía "n+1" que propone la necesidad de contar con "n" número de equipos o suministros que operen de manera continua más "1" redundante que operará en situaciones de contingencia.

Finalmente, las necesidades de eficiencia de operación obligan a planear considerando como prioridades los conceptos de calidad y rentabilidad.

3.2. SISTEMA DE ILUMINACIÓN.

3.2.1. Iluminación en oficinas.

Este es un factor importante ya que si se cuenta con una gran cantidad de ventanas, techos demasiado altos, etc., el control del Edificio Inteligente será inestable y en ocasiones deficiente. En cuanto a la iluminación existen 5 métodos básicos [Bernaden88] los cuales son:

3.2.1.1. Directa.
3.2.1.2. Semi - directa.
3.2.1.3. General difusa.
3.2.1.4. Semi - indirecta.
3.2.1.5. Indirecta.

3.2.1.1. Directa.

Es la más utilizada en el ambiente de oficinas y comercial, generalmente es por medio de balaustras y lámparas fijas en el techo.

3.2.1.2. Semi - directa.

Esta dividida en dos zonas, la mitad superior y la mitad inferior. La mitad superior generalmente está cubierta impidiendo el libre paso de la luz, este tipo de alumbrado es el que proveen las lámparas con pantalla.

3.2.1.3. General difusa.

Es la que proporciona un foco de luz de manera general en una habitación, en la cual todos los puntos de la fuente transmiten la misma intensidad de luz.

3.2.1.4. Semi - indirecta.

Es la que proporcionan las lámparas compuestas, en las cuales la parte inferior deja pasar la luz en menor intensidad que la superior, este tipo de alumbrado es frecuentemente utilizado para ambientación, ya que la parte inferior se puede transmitir luz de algún color o tono en especial, razón por la cual no es conveniente para trabajar.

3.2.1.5. Indirecta.

En este método la parte superior es la única que deja pasar libremente la luz, la cual se refleja en el techo por medio de colores claros a toda la habitación de manera uniforme; eliminando gran cantidad de variaciones y parpadeos al ser colocadas varias fuentes indirectas juntas.

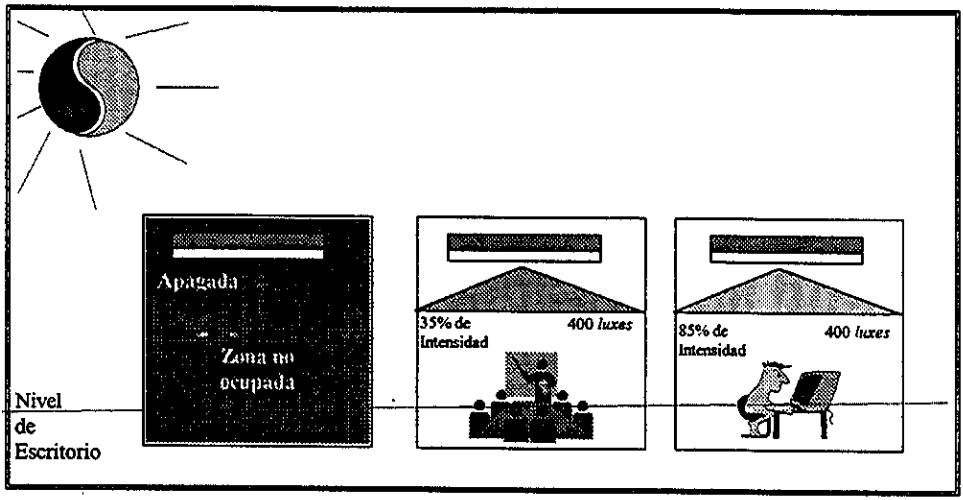


Figura 3.1. Supervisión del sistema de iluminación.. El nivel de iluminación depende del uso que se le da al recinto en cuestión, a la presencia o ausencia de personal en la misma, y a otros factores.

El sistema de control de iluminación de oficinas es útil para que este recurso sea usado óptimamente. Por medio de módulos de control y sensores, la cantidad de luz que proporcionan las luminarias pueden ser ajustadas de acuerdo a la contribución de luz exterior. Además los módulos pueden determinar si la oficina está ocupada; en caso de no haber personal las luminarias son apagadas o su intensidad es disminuida. Con la estación de supervisión del Sistema Inteligente las luces pueden ser programadas en un horario de operación. [Lidec88].

3.2.1.6. Ventajas.

La estrategia de control de iluminación por zona tiene un ahorro de energía eléctrica de hasta el 35% sobre un método convencional. Si se utiliza balaustra electrónica presenta además las siguientes ventajas:

- Tiene una vida útil estimada 3 veces mayor que la de una balaustra magnética; opera a un 20% de la temperatura que el otro tipo, por lo tanto, el equipo de aire acondicionado requiere enfriar menor carga térmica y por ésto hay menos consumo eléctrico.

- Opera a un factor de potencia de 95%, una balaustra convencional trabaja entre un 60 y 80%, por lo que se elimina, en el concepto de iluminación, la multa que cobra la CFE por abundancia de carga inductiva.

- Permite regular la intensidad desde 0 hasta 100%, aprovechando la contribución de luz exterior.

3.2.1.7. Operación.

- El sistema de iluminación es independiente para cada zona y se encarga de encender las luces en respuesta a eventos reportados por sensores.

- Cuando el sistema de detección identifica la salida de un usuario, las luces se apagan considerando que ya no hay ocupantes en el área. La condición de presencia o no de usuarios es reportada a la estación de supervisión del Sistema Inteligente, de manera que en forma remota se puede saber si una oficina está o no ocupada.

- Regula la intensidad luminosa para mantener un nivel adecuado aprovechando la contribución natural de luz exterior. Los niveles de luz ocurridos durante el día son reportados a la estación de supervisión del Sistema Inteligente de manera que puedan ser gratificados.

- La estación de supervisión del Sistema Inteligente podrá obtener reportes de los eventos ocurridos en la oficina y mediante éstos hacer una eficiente planeación por temporadas. Las luces de la oficina podrán ser programadas dentro de un horario de trabajo: cuando la oficina esté en horas fuera de operación, todos los servicios son apagados para evitar el consumo de energía eléctrica. También la iluminación en oficinas podrá encenderse o apagarse en forma remota.

- El usuario podrá apagar las luces a discreción.

- El sistema de iluminación está coordinado con el sistema de acondicionamiento ambiental y el sistema de seguridad. En caso de recibir una señal de alarma, se apagan las luces del circuito normal y se encienden las de emergencia, facilitando la localización de las puertas de salida.

3.2.2. Iluminación en pasillos interiores.

Un sector de continuo alumbrado son los pasillos. Existe la necesidad de mantener un nivel de iluminación constante en ellos ya que nunca se sabe cuando un usuario saldrá o entrará a un corredor, por ello se ha considerado que la luz debe mantenerse encendida; sin embargo, este modelo de operación provoca el consumo de energía todo el día. [Taboada83]

3.2.2.1 Ventajas.

- Reduce el consumo de energía eléctrica significativamente y no sacrifica la comodidad de los usuarios.

- Por medio de controladores y sensores se determina la presencia de personal en una sección del pasillo. Cuando no hay personal circulando, las luces son disminuidas; en el momento en que se reanuda la circulación de personal las luces son restauradas al 100% de su operación.

Figura 3.2.. Iluminación en pasillos interiores. El grado de iluminación depende de la presencia o falta de un ser humano en determinado sector del pasillo.

3.2.2.2 Operación

- El sistema de iluminación en pasillos aprovecha el concepto de ajuste electrónico de nivel de alumbrado, para ello utiliza balaustras electrónicas o lámparas incandescentes cuyo nivel de intensidad puede ajustarse.

- Cuando el sensor de presencia ha detectado movimiento en el pasillo, automáticamente enciende las luces de esa sección al 100% de intensidad, una vez que ya no se detecta presencia, espera algunos minutos y restablece el nivel de iluminación a un 20% para ahorrar energía.

- Si el pasillo tiene contribución de luz exterior, entonces la intensidad de las luminarias son ajustadas al nivel adecuado sin perder calidad de iluminación y comodidad. Los niveles de intensidad luminosa ocurridos durante el día son reportados a la estación de supervisión del Sistema Inteligente de manera que puedan ser gratificados.

- La iluminación en pasillos puede encenderse o apagarse en forma remota desde la estación de supervisión del Sistema Inteligente.

- En caso de recibir una señal de alarma, se apagan las luces del circuito normal y se encienden las de emergencia al 100%, facilitando la localización de las puertas de salida.

3.2.3. Iluminación exterior.

El sistema de iluminación exterior consiste en un controlador y un sensor, los cuales encienden las luces exteriores cuando no hay suficiente luz natural. Al amanecer cuando el nivel de iluminación es adecuado las luces son apagadas.

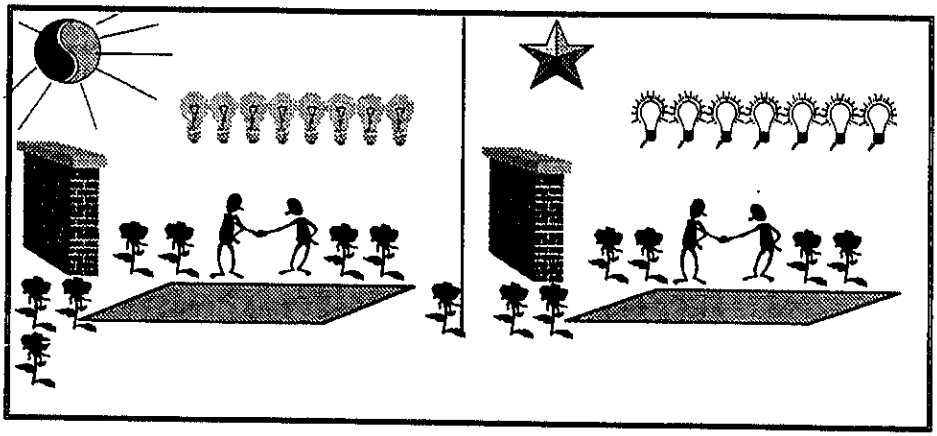


Figura 3.3. Iluminación exterior. La presencia humana en el exterior determina la activación exterior de la iluminación durante los períodos en los que no hay luz solar.

3.2.3.1. Ventajas

Algunas ventajas de un esquema automático de control de iluminación exterior respecto a un sistema no controlado son:

- Ahorro de energía por concepto de luces que se encienden en el momento de recibir una señal que indica que la iluminación natural ya no es satisfactoria y de apagarse cuando lo es.

- Capacidad de mantenimiento preventivo para reportar en forma anticipada la necesidad de reemplazar un componente o dispositivo cuya vida útil está por cumplirse.

3.2.3.2. Operación.

- El sistema de iluminación exterior puede ser operado mediante la estación de supervisión del Sistema Inteligente.

- Los eventos de encendido y apagado son reportados a la estación de supervisión del Sistema Inteligente.

- Los eventos de luz natural suficiente y luz natural no suficiente son también reportados.

- Cuando el módulo local de programación ha detectado un nivel bajo de iluminación natural en el área exterior, el sistema automáticamente enciende las luces.

- En caso de recibir una señal de alarma, el sistema de iluminación exterior apaga las luces del circuito normal y enciende las de emergencia.

3.2.4. Iluminación en estacionamientos.

3.2.4.1. Ventajas.

- Se cuenta con un sistema de fotosensor que detecta la ausencia de luz solar. Este puede combinarse con un horario para lograr un máximo de ahorro de energía. Es decir, se puede programar para encender las luces a cierta hora pero sólo si no es suficiente la luz solar; o se puede evitar encender las luces más allá de cierto horario aún si no hay luz solar.

- Permite el encendido manual y por software mediante la estación de supervisión del Sistema Inteligente.

- Este sistema permite cubrir imprevistos, como la necesidad de encender el alumbrado por causa de algún evento extraordinario o de apagarlas manualmente por la ausencia total de usuarios.

- La función del sistema de iluminación del estacionamiento es encender y apagar las luces de acuerdo a eventos programados y no programados.

3.2.4.2. Operación.

- La principal forma de controlar el sistema de iluminación del estacionamiento es a través de un horario. Este horario servirá como marco de referencia para las estrategias de ahorro de energía.

- El horario se almacena en el módulo de control y es manipulado a través de la estación de supervisión.

- Debido a los cambios en la luminosidad a través de las diferentes estaciones del año, existe la posibilidad de adaptación por medio del Sistema Inteligente lo que ofrece máxima flexibilidad.

- El sistema de iluminación del estacionamiento puede encenderse utilizando la estación de supervisión del Sistema Inteligente, el cual contaría con una estación remota instalada en la cabina de vigilancia. Aclarando que si se cuenta con servicios de red local, está puede estar en cualquier lugar del edificio.

- Opcionalmente se puede utilizar una segmentación del sistema de control del estacionamiento y un sistema detector de presencia. Cada vez que este sistema de detección identifique a un usuario se enciende el sistema de iluminación del estacionamiento por un tiempo mínimo. Transcurrido éste más un período de retraso, si se detecta la ausencia de usuarios en el estacionamiento, entonces se apagan las luces.

- El sistema de encendido manual básico es la última salvaguarda para las ocasiones en que todo el sistema de control falle. Consiste en una llave que se utiliza para sobrepasar todos los parámetros de control por medio de un contacto eléctrico que permite encender o apagar el sistema de iluminación del estacionamiento.

3.3 SISTEMA ELÉCTRICO.

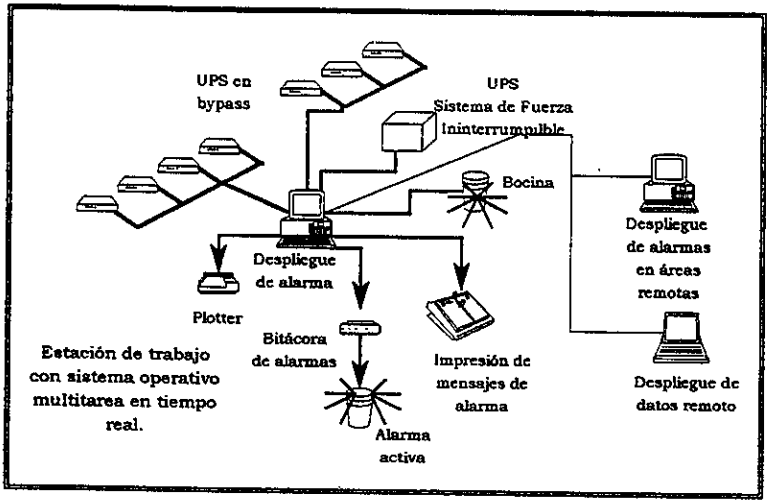


Figura 3.4. Supervisión y control de potencia.

La supervisión y control de potencia sirve para tomar mediciones del consumo de energía eléctrica y para hacer planeación estratégica; también se utiliza para confirmar la cuota que cobra la CFE; por último es un servicio que demuestra la forma en que se aprovechan los energéticos. [RAEEE92]

Cada sección tendrá acceso a su información particular en la base de datos por medio del módulo medidor. También se deberán instalar módulos medidores en áreas comunes y los administradores del Edificio Inteligente tendrán conocimiento de los consumos globales separados por sistemas (por ejemplo, cuanto se gastó en aire acondicionado o cuanto se consumió por piso).

3.3.1. Ventajas

- Evitar consumos que rebasen la demanda contratada e incurrir en multas.
- Llevar un registro anual del consumo eléctrico.
- Comprobar el consumo medido con el comprobante que entrega la CFE.
- Observar el comportamiento del consumo de energía eléctrica en el Edificio Inteligente durante diferentes períodos del año para la planeación estratégica.

3.3.2. Operación

- La estación de supervisión del Sistema Inteligente almacenará todas las muestras de potencia ocurridas durante un año.
- Desde la estación de supervisión del Sistema Inteligente se mostrarán gráficamente los consumos de energía eléctrica en forma diaria, semanal o mensual. Además del registro de Kw/h consumidos.
- Ajuste de la temperatura, el aire acondicionado y el nivel de iluminación cuando se pronostique un consumo que cause una multa.

3.4. SISTEMA DE ACONDICIONAMIENTO AMBIENTAL.

El principal objetivo a cubrir por los sistemas del Edificio Inteligente es brindar mayor comodidad a los usuarios, para mantener un alto nivel de productividad. Se ha comprobado que no sólo la temperatura de una habitación constituye el medio ambiente, sino también otros factores adicionales como son: la iluminación, el ruido y la vibración.

En estudios recientes se ha constatado que el ser humano trabaja de manera más rápida y eficiente en condiciones ambientales óptimas. [EDIFINTEL94]

El sistema de acondicionamiento ambiental es un conjunto integrado de servicios y equipos que se encargan de proveer y mantener las condiciones del medio confortables. Los Edificios Inteligentes son capaces de adaptarse a los cambios reajustando sus parámetros y controles. Los sistemas de acondicionamiento ambiental son capaces de controlar la temperatura, humedad, purificación y distribución del aire. Estos cuatros componentes tienen diferentes efectos en los ocupantes.

Las temperaturas elevadas causan sensación de fatiga y a su vez disminuyen el rendimiento. Por otra parte las bajas temperaturas causan una sensación de tensión nerviosa.

La carencia de humedad en el ambiente incrementa la cantidad de carga electrostática y causa resequedad en la piel y ojos. El exceso de humedad produce la sensación de que las áreas son mucho más calientes de lo que la temperatura indica.

Por cuestiones de comodidad, el efecto de movimiento del aire debe ser considerado al momento de diseñar el sistema. Si la distribución del aire se realiza a través del techo, deberá tener una velocidad de por lo menos 20 pies por minuto (fpm por feet per minute) para evitar problemas de malos olores y estancamiento del movimiento del aire. Una velocidad de 25 a 40 pies por minuto (fpm por feet per minute) es normalmente tolerable, sin embargo, las variaciones rápidas en la velocidad de inyección pueden ocasionar molestias y ruidos.

Las áreas ocupadas deberán estar libres de olores irritantes como el polvo, humo de cigarro y sudor. La ventilación involucra el reemplazo del aire de un área cerrada, por aire del exterior; el cual puede ser tratado por equipo de calefacción, enfriamiento, purificación o secado y retornado.

En el código de la ASHRAE (por American Society of Heating, Refrigeration and Air Conditioning Engineers) los estándares se especifican dentro de rangos, los cuales varían de 73°F a 79°F para el verano y de 69°F a 74.5°F en el invierno, contando con una humedad relativa de 30 a 60% durante todo el año; en cuanto a la velocidad del aire para áreas en interiores deberá de estar dentro del rango de 0.15 y 0.25 m/s.

Los estándares de la ISO (International Standards Organization) [IBS88] son más precisos al especificar sus condiciones:

a) Condiciones sugeridas para verano.

- Temperatura de operación de 20°C a 24°C.
- Temperatura de la superficie del piso de 19°C a 26°C. La calefacción a través del suelo deberá estar diseñada para mantener una temperatura de 29°C.
- Diferencia de temperatura vertical del aire de 0.1 m. a 1.1 m. sobre el piso.
- Velocidad promedio del aire de 0.15 m/s.

b) Condiciones durante el invierno.

- Temperatura de operación de 23°C a 26°C.
- Temperatura de la superficie del piso de 22°C a 29°C.
- Diferencia de temperatura vertical del aire de 0.1 m. a 1.1 m. sobre el piso.
- Velocidad promedio del aire de 0.25 m/s.

- Temperatura irradiada de superficies verticales como ventanas, láminas o puertas de menos de 10°C en relación a un pequeño plano vertical de 0.6 m. sobre el piso.

- Temperatura disipada por objetos que generen calor de menos de 5°C en relación a un plano de 0.6 m. sobre el piso.

La diferencia radica en que el estándar dictado por ASHRAE, está realizado pensando en la comodidad del 80% de la población norteamericana, mientras que el estándar ISO está realizado para el 80% de la población mundial.

Estudios realizados en oficinas y ambientes laborales indican que las facilidades y controles ambientales pueden incrementar hasta un 30% la productividad. Por otra parte los sistemas de aire acondicionado y control mal diseñados pueden afectar gravemente la productividad y aumentar el índice de enfermedades. [EDIFINTEL94]

En áreas y espacios muy ocupados como salones, teatros y auditorios la temperatura deberá ser menor a la indicada como temperatura de comodidad, debido a la proximidad en que se encuentran los ocupantes ya que generan más calor. La misma radiación deberá ser tomada en cuenta ya que también existe entre el equipo de cómputo o trabajo y los ocupantes.

En el caso de una inyección y flujo del aire a través del techo, se requerirá de mayor velocidad así como más refrigeración debido a la distancia a recorrer entre el techo y la zona ocupada. Para estas condiciones normalmente se recomienda que el flujo de aire oscile entre 25 y 40 pies por minuto (fpm por feet per minute) sin embargo, las rápidas variaciones, ruidos molestos y las corrientes de aire frío pueden ocasionar dolores en la región del cuello, espalda y rodillas, razón por la cual se recomienda realizar una inyección del aire a través del piso, debido a que no se requiere de gran velocidad del mismo, y de esta manera no se disipa el calor de las lámparas y ventanas antes de que el de las personas. Las capacidades de aire y cantidad de calor a disipar se consideran para determinar el equipo a instalar. Ya que se toman en cuenta la cantidad de calor promedio

generado por cada persona y la cantidad de calor a disipar por máquina se realiza el cálculo para la cantidad de calor o frío que se deberán disipar tanto por el sistema como por las ventanas y puertas. Tomando como referencia el día más frío y el día más cálido registrado en la zona el año anterior.

Existen 4 sistemas de aire acondicionado usados comúnmente en los edificios.

A. Sistema de aire acondicionado con ducto dual.
B. Unidad de inducción de ducto sencillo.
C. Sistema multizona.
D. Sistema de volúmen variable de aire acondicionado por ducto sencillo.

A. Sistema de aire acondicionado con ducto dual. Maneja un volumen constante de aire, requiere de un espacio para mezclarlo debido a que un ducto inyecta aire caliente y otro frío.

B. Unidad de inducción de ducto sencillo. Tiene el efecto de calefactor o refrigerador del aire por medio de columnas de agua bombeada a través de la unidad de inducción según se requiera.

C. Sistema multizona. Contiene reguladores individuales de aire caliente y frío, al ser mezclados desde la unidad central no requiere de espacio adicional.

D. Sistema de volumen variable de aire acondicionado por ducto sencillo. El control de la temperatura se realiza por medio de la variación de aire a proveer por habitación, la cual deberá encontrarse dentro de los límites preestablecidos en el diseño.

Es conveniente complementar estos sistemas por medio de dispositivos como:

- Colocar manejadoras de aire, las cuales ayudarán a proveer de un control más preciso del flujo del mismo.

- Colocar difusores de aire ajustables, cajas de volumen de aire variable y termostatos individuales.

Existen factores que afectan el diseño y operación de los sistemas de acondicionamiento ambiental en los Edificios Inteligentes tales como:

A. Variaciones regionales en cargas de calor y refrigeración.
B. Calor disipado por equipos de oficina y cómputo.
C. Sensibilidad del equipo.

A. Variaciones regionales en cargas de calor y refrigeración. Para ello en los Edificios Inteligentes se incluyen sistemas de aire acondicionado de múltiples zonas, los cuales varían sus condiciones y espacio de afectación.

B. Calor disipado por equipos de oficina y cómputo. La capacidad de los sistemas deberá ser suficiente para acoplarse al incremento de calor generado por los equipos de oficina.

C. Sensibilidad del equipo. El equipo es frecuentemente sensible a la temperatura y la humedad, necesitando estrictos controles ambientales para asegurar su buen funcionamiento.

3.4.1. Control de aire acondicionado.

El aire acondicionado es el sistema que requiere mayor demanda de consumo eléctrico, por lo que es obligatorio tomar las siguientes consideraciones para que este servicio sea eficiente.

- Reducir la carga térmica de otros equipos eléctricos, como las luminarias.

- Coordinar el desempeño del sistema de aire acondicionado con el sistema eléctrico.

- Tener las herramientas para una planeación estratégica.

- Controlar con sistemas informáticos adecuados y no sólo encender o apagar las unidades manejadoras de aire y los enfriadores de agua.

- Aplicar la estrategia de multizonificación por medio de compuertas ajustables.

3.4.1.1. Ventajas.

- Existe un ahorro de energía eléctrica de hasta 35%, si es bueno el funcionamiento del aire acondicionado.

- Aumento en la comodidad del usuario, porque las zonas calientes son enfriadas suficientemente y al mismo tiempo las zonas frías lo son menos.

- La selección de temperatura de comodidad es instruida de modo local por un termostato o remoto por la estación de supervisión del Sistema Inteligente.

- Ajusta la banda de operación por condiciones (coordinación con sensores de presencia o control de acceso) al fijar diferentes temperaturas del usuario, del personal de servicio o de los cuartos vacíos.

- El control regula la presión de aire sobre los ductos para evitar ruidos y usar sólo la energía necesaria en la manejadora.

- El control de flujo de aire, mediante compuertas, permite que una manejadora suministre diferentes temperaturas en una área seccionada.

3.4.1.2. Operación.

- El sistema de aire acondicionado selécciona la temperatura de cada zona por medio de un termostato capaz de fijar puntos de operación. Además desde la estación de supervisión es posible cambiar los puntos de operación de las zonas. El termostato cuenta con una pantalla digital.

- La válvula que da paso al flujo de agua helada o caliente de la manejadora de aire, responde de forma automática a la temperatura programada.

- El sistema de aire acondicionado opera según la temperatura seleccionada por el usuario cuando éste se encuentra dentro de la zona; y cuando no, permite que la temperatura suba para ahorrar energía (Estrategia de ocupación).

- El sistema de aire acondicionado diagnostica si los filtros están sucios y los reporta a la estación de supervisión del Sistema Inteligente. Se registran en una base de datos las estadísticas de temperatura, consumos y operación, para planear estrategias administrativas por temporadas y para implementar un mantenimiento preventivo. Las muestras de temperatura se toman cada cinco minutos de tal manera que se grafique la termografía en forma diaria.

- El sistema responde a las alarmas contra incendio, si ocurre uno, las manejadoras de aire se apagarán para ventilar el humo.

- Desde la estación de supervisión es posible fijar calendarios y horarios para prender y apagar cada una de las manejadoras.

- Cuando el consumo eléctrico rebasa el límite fijado en la estación de supervisión del Sistema Inteligente, la temperatura de las zonas es ajustada para que el consumo disminuya.

3.4.2. Control de la central de aire acondicionado.

En un sistema de aire acondicionado para edificios medianos y grandes, son usados los enfriadores de agua (chillers), bombas para agua fría, bombas para agua caliente y bombas para agua condensada. Estos equipos representan un porcentaje muy alto de la carga total de energía eléctrica, por lo tanto, el uso eficiente de este recurso con estrategias adecuadas es casi obligatorio.

3.4.2.1. Ventajas.

- El enfriado es usado eficientemente, sólo en la cantidad, horario y momento que se requiera. Si el sistema de aire acondicionado está diseñado para válvulas de tres vías, entonces se modula el agua de las manejadoras para que sólo circule en el serpentín la necesaria.

- Si el sistema de aire acondicionado está diseñado para válvulas de dos vías con un pequeño porcentaje de tres vías, entonces presenta las ventajas antes mencionadas y reduce el esfuerzo de bombeo permitiendo que sólo circule el agua necesaria.

- Si el diseño del Edificio Inteligente y las condiciones del lugar obligan a tener, en ciertos períodos del año zonas calientes y zonas frías, entonces se requiere de unidades manejadoras de aire con doble serpentín. La tubería debe ser de cuatro vías.

3.4.2.2. Operación.

- Cuando el agua de retorno en el enfriador tiene una temperatura inferior a la que requiere el Edificio Inteligente, entonces el punto de operación ("set-point") del equipo se cambia para que no se desperdicie energía.

- Permite arrancar o parar los enfriadores desde la estación de supervisión bajo un programa de ahorro de energía.

- Cuando el sistema de supervisión de potencia detecta un sobrepaso en la demanda contratada, entonces los puntos de operación del equipo son cambiados para bajar el consumo y evitar multas.

- Las bombas de agua son alternadas con las de emergencia, para asegurarse de que siempre estén funcionando y que todas las bombas tengan un tiempo de reposo. En un sistema de dos vías las bombas no trabajan a su máxima capacidad por lo que hay un ahorro.

- Permite un monitoreo de los enfriadores y bombas para mantenimiento preventivo.

3.4.3. Control de entalpía.

La entalpía es usar el aire exterior para ayudar a enfriar el Edificio Inteligente. El uso de esta estrategia requiere que la temperatura de aire húmedo exterior promedio durante el año, sea menor a los 21°C. La calidad de aire exterior debe ser el adecuado, en caso de que esto no ocurra es necesario el uso de filtros activos. El controlador de las unidades manejadoras de aire ajusta la compuerta para introducir aire exterior cuando se requiere.

3.4.3.1. Ventajas.

- En los períodos del año donde la temperatura es favorable se obtiene un ahorro alto de energía eléctrica en cuanto a las manejadoras de aire.

- Se inyecta aire limpio y nuevo a las diferentes zonas.

3.4.3.2. Operación.

- Cuando el controlador detecta una temperatura menor a la deseada entonces comienza a modular la entrada de aire externo.

- Si la temperatura es superior a la deseada en una zona, entonces se cierran las compuertas para que sólo se utilice el aire interior.

- Desde la estación de supervisión del Sistema Inteligente se administra la operación del sistema bajo la estrategia de entalpía.

3.5. SISTEMA DE CONTROL DE ACCESO.

Las zonas, oficinas o cuartos además de ofrecer comodidad deben incluir estrategias de seguridad, además de un control de acceso sencillo y seguro. A través de una lectora de proximidad, el usuario únicamente tiene que acercar su tarjeta a la puerta para abrirla.

A través de la estación de supervisión del Sistema Inteligente se puede decidir qué personal o usuarios tienen acceso a una zona, oficina o cuarto. Si la llave se pierde, sencillamente se da de baja el nombre de la persona que la tenía y se da de alta una nueva, de esta manera se evitan accesos no permitidos.

Como medida de seguridad, desde la estación de supervisión se puede saber qué usuario entró además del día y hora en que lo hizo. [EDIFINTEL94]

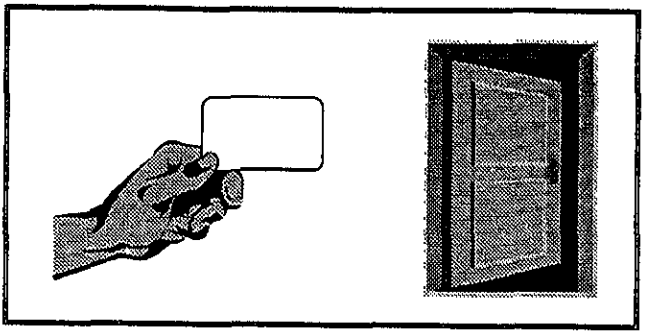


Figura 3.5. Control de acceso, el cual se puede llevar a cabo mediante la utilización de una tarjeta electromagnética.

3.5.1. Ventajas.

- Operación sencilla y rápida.
- Restringe el acceso a las zonas, oficinas o cuartos, permitiendo que sólo aquellos usuarios autorizados puedan entrar.
- No hay necesidad de cambiar las chapas a causa de llaves extraviadas o no devueltas.
- En la estación de supervisión del Sistema Inteligente queda registrado el día, la hora y la persona que entró en determinado lugar.
- En caso de fallas se pueden usar llaves normales.
- Permite fijar una banda de temperatura determinada para cada usuario, si el sistema está integrado con el de acondicionamiento ambiental.
- Permite activar la iluminación, si el sistema está integrado con el control de iluminación.

3.5.2. Operación.

- Por medio de la estación de supervisión del Sistema Inteligente se programan los horarios y privilegios que cada tipo de usuario tiene al usar su tarjeta. Esta información se puede dar de alta, baja o modificar de manera instantánea.

- Cuando el usuario introduzca la tarjeta a la lectora, se encenderá una luz verde indicando que la puerta puede ser abierta.

- En caso de que la tarjeta no esté autorizada para abrir en cierta zona, la lectora hará un "beep" y encenderá una luz roja para indicar que el acceso no está autorizado, la puerta en esta situación no podrá ser abierta. La hora, fecha y usuario serán registrados en la estación de supervisión.

3.6. SISTEMA DE DETECCIÓN Y EXTINCIÓN DE INCENDIOS.

Un elemento básico para la protección de los usuarios es el equipo para detección y extinción de incendios. En caso de siniestro, el sistema avisará oportunamente del problema para que se den las instrucciones para la evacuación y se tomen las medidas necesarias para corregirlo.

Aunque no influye en forma directa en el concepto de costo beneficio, lo hace en forma indirecta al bajar las primas de seguro y al considerar que las vidas de los usuarios son valiosas y que cuando ocurre un siniestro, el costo de la inversión es mucho menor a los daños que se pueden evitar y vidas que se pueden perder.

3.6.1. Ventajas.

- Avisa a los usuarios de un posible siniestro.

- Se dan las instrucciones adecuadas para la evacuación, en caso de que exista una situación de emergencia.

- Protege la inversión del inmueble contra pérdidas irreparables avisando de un posible problema.

- Los usuarios se sientes más protegidos en un lugar seguro.

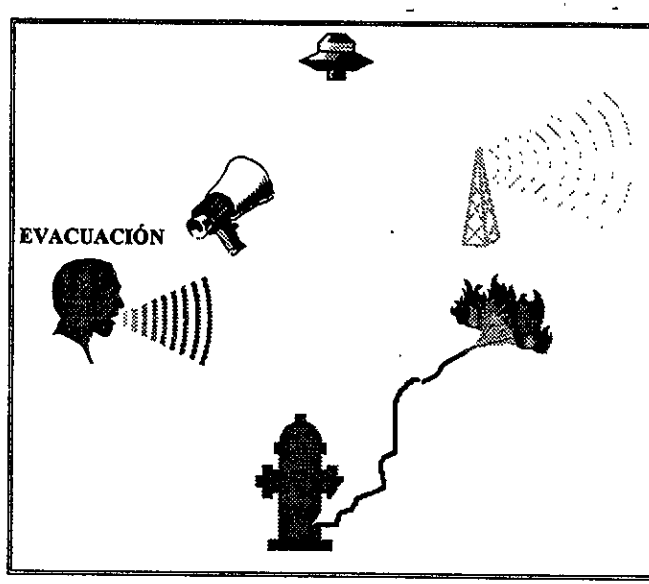


Figura 3.6. Detección y extinción de incendios. Determinación de las medidas de seguridad que se tomarán en caso de que se presente algún siniestro.

- Permite coordinar los equipos del Sistema Inteligente para que tomen las acciones necesarias en caso de una emergencia.

- La estación de supervisión guarda los eventos de alarma reportados por el sistema de detección y extinción de incendios para corroborar el buen funcionamiento del equipo.

- Presenta al operador de manera gráfica, la ubicación de los sensores e indica cual de éstos reporta un problema.

3.6.2. Operación.

- El Sistema Inteligente recibe los eventos de alarma que el sistema de detección y extinción de incendios envía por uno de sus puertos.

- El Sistema Inteligente avisa a los equipos, que estén conectados a la red de recursos ambientales, que hay un evento de alarma. Cada equipo en particular tomará las acciones pertinentes para casos de emergencia.

- La estación de supervisión del Sistema Inteligente guardará todos los eventos que el sistema de detección y extinción de incendios le reporte.

- En caso de un problema, en la estación de supervisión se mostrará la localización (en forma gráfica representando el piso y zona) del sensor que está alertando de un posible siniestro.

3.7. SISTEMA DE DETECCIÓN DE INTRUSOS.

Los sensores determinan si hay zonas ocupadas pero si un cuarto lo está bajo circunstancias anormales, ya sea porque se encuentra fuera de horario o porque no tiene acceso correcto, entonces la estación de supervisión avisa al personal de vigilancia sobre esta situación irregular.

3.7.1 Ventajas.

- Protección contra accesos no autorizados.

- Registro de dónde y cuándo ocurrió un acceso no autorizado.

Figura 3.7. La detección de intrusos y la vigilancia de la normatividad de seguridad son elementos importantes en la prevención de ilícitos.

3.7.2. Operación

- Cuando los sensores de movimiento detectan la presencia de personal o usuarios en determinada zona, oficina o cuarto, entonces se avisa a la estación de supervisión. Por medio de la base de datos de personal o usuarios autorizados y del registro de eventos se determina si la zona, oficina o cuarto debería estar ocupada, si el resultado es negativo la estación de supervisión avisa al vigilante que hay personal no autorizado.

- Las situaciones irregulares son almacenadas en la base de datos de la estación de supervisión para en un futuro hacer referencias cruzadas de los accesos a las diferentes zonas, oficinas o cuartos del Edificio Inteligente.

3.8. SISTEMA HIDRÁULICO.

3.8.1. Control de cisternas.

Si se cuenta con cisternas que deban ser llenadas de un pozo profundo de gran capacidad, entonces a través de módulos de control se puede automatizar esta actividad. Por medio de sensores se determina el nivel de agua de la cisterna, cuando no hay suficiente el sistema enciende las bombas para que la cisterna se llene.

3.8.1.1. Ventajas.

- Las cisternas siempre tendrán el agua necesaria, lista para ser usada.
- No se requiere de personal dedicado a llenar las cisternas.
- Se reducen al mínimo los problemas de cisternas vacías.

3.8.1.2. Operación.

- Cuando se detecte que la cisterna esté media vacía, se enciende la bomba de agua y se abre la válvula selenoide para llenarla.
- Cuando la cisterna esté llena se apaga la bomba de agua y se cierra la válvula selenoide para evitar que el líquido se desborde.
- En caso de que la válvula selenoide se dañe o exista un elemento en el agua que tape la tubería, a través del sensor de presión de agua se detectará esta anomalía y se apagará la bomba para evitar rupturas en la instalación.
- Cuando la cisterna esté completamente vacía, se avisa a las bombas del sistema de riego para que no funcionen, de esta manera se evita que trabajen en seco.
- Los eventos que indican el nivel en que se encuentra la cisterna, en caso de alta presión o presión normal son reportados a la estación de supervisión para registrarlos y corroborar que el control de las mismas esta funcionando adecuadamente o detectar anomalías en su operación.
- A través de la estación de supervisión se puede encender el control de las cisternas para que comience a llenarlas, aún cuando el nivel esté medio lleno.

3.8.2. Control de riego.

El control de riego óptimo ayuda a que los árboles y jardines tengan agua cuando la necesite. El ahorro considerable radica en que se requiere de poco personal para atender los jardines y no se desperdicia agua. El sistema se encarga de encender y apagar las bombas y las válvulas de acuerdo a eventos programados o manuales, y a través de sensores de humedad se corrigen los tiempos asignados de cada sección del jardín para que sean regados sólo lo necesario. La programación se puede realizar a través de la estación de supervisión del Sistema Inteligente o por medio de una sencilla estación manual que puede ser operada por el jardinero sin tener que usar la estación de supervisión.

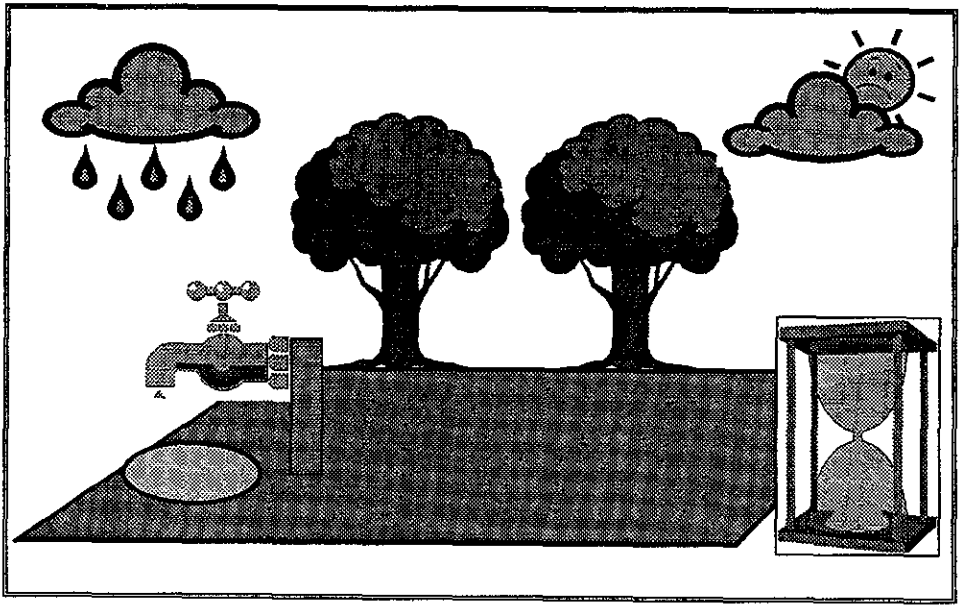


Figura 3.8. El Control de riego depende de varios factores climáticos.

3.8.2.1. Ventajas.

- Operación sencilla desde una estación manual.

- Poco personal para el cuidado de jardines, ya que el sistema se encarga de regar automáticamente y cuando es necesario.

- Los eventos ocurridos con el sistema de riego son reportados a la estación de supervisión.

- Uso adecuado del agua.

3.8.2.2. Operación.

- La función del sistema es encender y apagar las bombas y las válvulas de riego de acuerdo a eventos tanto programados como no programados.

- La estrategia principal de ahorro del sistema de riego sería una programación por medio de horarios. Este horario se puede manipular por medio de la estación de supervisión para adaptar los tiempos de riego a las diferentes temporadas de lluvia y sequía.

- Es posible que en algún momento surja la necesidad de encender las válvulas y bombas de riego para cubrir eventos excepcionales; esto se puede hacer por medio de la estación de supervisión.

- El sistema está dotado de un sensor de humedad el cual acciona el riego automáticamente cuando el suelo está muy seco.

- Para evitar que las bombas trabajen cuando no haya agua, se cuenta con un sensor de nivel en la cisterna, que bloquea el sistema de riego. Este comunica la presencia de anomalías a la estación de supervisión con el fin de que la cisterna sea reparada.

- El sistema cuenta con un sensor de presión, el cual sirve para proteger la tubería de ruptura debido a aspersores bloqueados. Esta anomalía también es reportada a la estación de supervisión.

CAPÍTULO IV. AGENTE INTELIGENTE.

Es condición necesaria para que un edificio pueda ser considerado como inteligente, el que tenga un conjunto de sistemas basados en técnicas de inteligencia artificial que le permitan desempeñar por sí mismo diversas actividades con una efectividad que iguale o mejore el desempeño que podría tener un ser humano. Entre las actividades más importantes que pueden caer dentro de este rubro están:

- Tomar las decisiones necesarias o apoyar la toma de decisiones en caso de emergencia.
- Predecir y, en caso necesario, corregir o ayudar a la corrección de las fallas que ocurran dentro del edificio.
- Controlar las actividades y el funcionamiento dentro del edificio.

En este capítulo se describe el diseño e implementación de un Agente Inteligente desarrollado para modelar algunas de las características de los sistemas inteligentes encargados de apoyar la toma de decisiones en caso de emergencia dentro de un Edificio Inteligente, mediante la utilización de una base de conocimientos y técnicas de inteligencia artificial. Este Agente se encarga de solucionar el problema de encontrar una ruta de evacuación segura para salvar a todas las personas que se encuentren en un Edificio Inteligente en caso de incendio.

Para el funcionamiento del Agente Inteligente se necesitan una serie de dispositivos que funcionen como sus sensores, mediante los cuales pueda percibir su medio ambiente. Estos dispositivos, con los que estamos suponiendo que contamos, están conectados a concentradores, los cuales se encuentran distribuidos en todo el edificio. Los concentradores están dedicados a guardar información sobre el desempeño y el funcionamiento de los dispositivos, ya sea que cada concentrador controle los dispositivos del mismo tipo, o los que se encuentran en un área determinada.

La información es controlada centralmente por un Sistema de Monitoreo, el cual recoge la información sobre los dispositivos contenida en los concentradores y la reporta a una base de datos.

El Agente Inteligente lee esa información de la base de datos y de ser necesario, analiza el caso y toma decisiones para resolver el problema que se haya presentado. Tanto el problema como la solución son presentados al operador. El Agente Inteligente podría controlarse, operarse y tomar las decisiones automáticamente, pero en cuestiones de seguridad humana es preferible que la decisión final sea tomada por el operador, siendo éste apoyado por el Agente Inteligente. No obstante lo anterior, en los casos donde es importante una acción inmediata el Agente Inteligente toma las decisiones que considera necesarias, y se encarga de direccionarlas a los dispositivos en cuestión.

4.1. INFRAESTRUCTURA.

Para el desarrollo del Agente Inteligente se trabajó bajo el supuesto de que se contaba con la infraestructura de un Edificio Inteligente, es decir, se supuso la existencia de un edificio que se encontrase equipado con un extenso sistema de detección humo con sensores localizados en techos y pisos, además de señalizaciones luminosas que indiquen la ruta de evacuación, y de sensores que permitan detectar presencia humana en cualquiera de las áreas del Edificio.

Es evidente, asimismo, que este Edificio Inteligente debe tener la problemática de encontrar una ruta segura para evacuar a sus ocupantes, ya que de otra forma no se podría justificar el desarrollo del sistema.

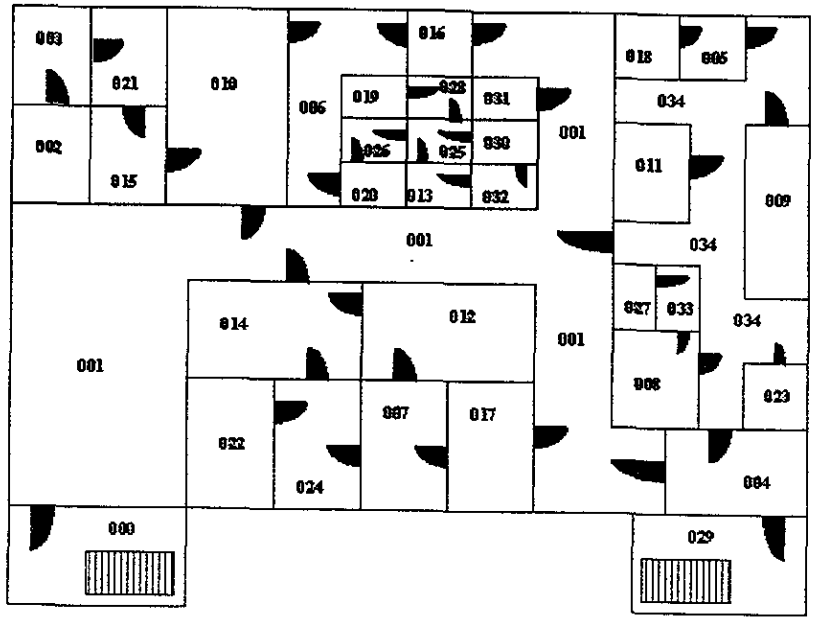
Todos los dispositivos anteriores deben ser vigilados por el Sistema de Monitoreo, a través de los concentradores, el cual reporta el comportamiento de cada uno de los dispositivos a una base de datos, la cual será utilizada para analizar los hechos,

determinando una ruta de evacuación segura para las personas que se encuentren en peligro dentro del edificio.

4.1.1. Diseño Interior.

Se diseñó el plano arquitectónico de un nivel del Edificio Inteligente con el objetivo de que representara alguna problemática al intentar buscar una ruta de evacuación en caso de incendio.

El plano arquitectónico se limitó a un solo nivel ya que esto se consideró suficiente para la demostración de las capacidades de la aplicación de técnicas de Inteligencia Artificial para labores de evacuación en un Edificio Inteligente en caso de incendio. El plano del primer nivel es el siguiente:



Ya que en la mayoría de los casos, las áreas dentro de un edificio tiene dimensiones muy parecidas, en el diseño interior propuesto hemos supuesto que las áreas cumplen con esta característica, por lo cual, en la base de datos todas tienen especificado un tamaño de 10 metros cuadrados.

4.1.2. Dispositivos Sensores.

La prevención de incendios en Edificios Inteligentes es de gran importancia y comienza con la detección temprana de un incendio, además de contar con el equipo necesario para la extinción y con unas instalaciones seguras, diseñadas para disminuir las consecuencias de este tipo de siniestros.

El tipo de medidas y equipo utilizado en cada zona para la prevención de desastres depende de la actividad que en ésta se desarrolle. Los centros de cómputo, cuartos de máquinas y laboratorios requieren de equipo más sofisticado y seguro.

El uso de sensores que no solo detecten el incendio sino que también se conozca la ubicación de éste, permite que se tenga una respuesta más rápida de la brigada contra incendios.

La prevención de incendios consta de tres etapas básicas para su buen funcionamiento:

1. Detección.
2. Extinción.
3. Evacuación.

Para los fines propios del Agente Inteligente solo nos interesan las etapas de Detección y Evacuación, por lo cual serán las únicas que revisaremos.

dos placas cargadas eléctricamente, cuando la densidad de partículas aumenta, se reduce la corriente y, al rebasar el nivel mínimo predeterminado, se activa la señal de alarma.

Frecuentemente se utilizan detectores con dos cámaras de muestreo de aire para compensar la disminución de corriente causada por cambios de humedad y de temperatura. Este método es utilizado donde pueden ocurrir incendios que avanzan muy rápido.

4.1.2.1.1.3. Detectores fotoeléctricos.

Se utilizan especialmente en el inicio de incendios, cuando no hay llamas pero hay demasiado humo; este caso es más frecuente por el uso de materiales inflamables. Este método aprovecha el fenómeno de la disipación de la luz para "ver" el humo. Un tren de pulsos a 10 Hz. de luz infrarroja es disparado dentro de la cámara del detector (conocida como cámara de humo) y, cuando el humo entra, interfiere la señal. La cantidad de luz disipada es monitoreada por un fotodiodo y, cuando se reduce la disipación por la interferencia del humo, se activa la señal de alarma.

La cámara de humo evita el paso de luz natural o del sistema de iluminación, dejando pasar sólo las partículas de humo.

Los detectores fotoeléctricos se utilizan a nivel de piso y techo, y los de ionización sólo en techos, para que el sistema responda más rápido por la tendencia de que las llamas aumentan su altura.

El arreglo de los detectores depende del área donde sean instalados y del tipo del equipo y materiales que ahí se tengan. Puede conectarse en forma cruzada o por sectores. La conexión cruzada se tiene con dos o más circuitos independientes de detectores o aspersores que, en caso de falla de alguno, el otro cubre la zona de siniestro. Por sectores es cuando un sólo circuito cubre una zona.

4.1.2.1.2. DETECCIÓN DE PRESENCIA.

4.1.2.1.2.1. Detectores de Movimientos.

Cuando los detectores de movimiento detectan el más leve movimuento dentro de su área de cobertura, envía señales de radio frecuencia al receptor base, y de ahí al sistema de monitoreo. El área de cobertura es de aproximadamente 100 grados y 15 metros.

4.1.2.1.2.2. Detectores Ópticos.

Son detectores de ultrasonidos que en combinación con las unidades programables de evaluación permiten la detección sensórica de larga distancia (hasta 10 metros). Trabajan a la velocidad de l a luz. Se encuentran disponibles en dos series que se diferencian en el tamaño y en la distancia de detección.

4.1.2.1.2.2.1. FOTOCELULAS CON UNA SALIDA DE CONEXIÓN.

Tamaño de 30 x 30 x 15 mm. Su frecuencia de conexión es de 1000 Hz. y tiene un ajuste de la sensibilidad mediante un potenciometro incorporado.

4.1.2.1.2.2.2. FOTOCELULAS PALPADOR CON SUPRESIÓN DE FONDO.

Proporciona un ajuste adecuado al alcance de los detectores de proximidad ópticos. Tiene un alcance de palpación de hasta 2 metros.

4.1.2.2. Evacuación.

Para la etapa de evacuación se utilizan las señales luminosas y audibles ya establecidas por las autoridades internacionales. Dichas señales están normalizadas y sólo varían en tamaño. Estas reglamentaciones para el uso de las señalizaciones garantizan un fácil y seguro desalojo de las instalaciones.

Algunas señales permanecen encendidas constantemente, pero otras, como las que en este caso estamos suponiendo que tenemos, se activan sólo en caso de emergencia. Todas éstas tienen un bajo consumo de energía y pueden conectarse en operación conjunta o independiente, gracias a su construcción modular.

Los señalamientos deben contar con las siguientes características:

- Mínimo consumo de corriente para reducir el gasto de energía y costo del cableado.
- Alta sonoridad y luminosidad.

En caso de incendio, es recomendable extraer el humo de las áreas a evacuar para que los ocupantes no sufran una intoxicación por el humo inhalado; la extracción del humo se realiza en coordinación con el sistema de aire acondicionado, así como aislar la zona del incendio presurizando el cubo de escaleras. También se envían instrucciones a los diferentes sistemas del edificio para que actúen y ayuden a que se realice la evacuación de los ocupantes de manera ordenada y rápida. Estas labores no fueron contempladas en la implementación del Agente Inteligente por exceder el objetivo que fue establecido para el mismo, que se limita actualmente a la generación de una ruta de evacuación.

Si los detectores de presencia no pueden determinar si la zona está evacuada, el control de acceso tiene la capacidad de enlistar todas las personas que entraron a la zona del incendio y rastrearlas por el edificio, y así determinar si permanece alguien.

4.1.3. Sistema de Monitoreo.

Como ya se mencionó anteriormente, también se ha supuesto que el Edificio Inteligente cuenta con un Sistema de Monitoreo que sea capaz de controlar centralmente la información sobre el desempeño y estado de cualquier dispositivo conectado a la red de comunicaciones, en nuestro caso especial los detectores de incendio y de presencia humana, y

registrar dichos datos en una base de datos relacional en Sybase, en alguna tabla definida para ese fin. El usuario podrá determinar las variables que desea monitorear así como el intervalo de tiempo entre cada monitoreo.

Para efectos de implementación del Agente Inteligente, y considerando que no contamos en realidad ni con los detectores de presencia ni los de incendio, hemos decidido que los datos sobre las áreas donde sea detectada la presencia de seres humanos y aquellas donde se haya identificado un incendio sean introducidas manualmente al Agente Inteligente en tiempo de corrida.

El usuario es responsable de mantener actualizada la base de datos. También es su deber determinar el tipo de información que le interesa saber sobre los dispositivos, el intervalo de monitoreo y la base de datos que desee que se utilice para reportar la información obtenida del monitoreo.

4.2. DISEÑO DEL AGENTE INTELIGENTE.

Se decidió diseñar un Agente Inteligente tomando en consideración las características propias de este tipo de aplicaciones y las del problema de generación de una ruta de evacuación en caso de incendio. Conceptualmente, un Agente Inteligente, como ya fue descrito en el Capítulo I, percibe su ambiente mediante sensores y responde o actúa en tal ambiente por medio de efectores buscando obtener el mejor desempeño posible. En nuestro caso, el Agente Inteligente percibe el estado de un nivel de un Edificio Inteligente mediante sensores para la detección de incendios y de presencia humana y genera una ruta de evacuación segura, la cual hace del conocimiento de las personas que deben ser evacuadas por medio de señales luminosas, las que son consideradas como sus efectores.

El Agente Inteligente ha sido desarrollado bajo el enfoque de la Actuación Racional Limitada. De esta manera, entenderemos que el adjetivo inteligente implica actuar de manera tal que la toma de decisiones sea la más adecuada, considerando que no

El enfoque del modelo cliente/servidor que hemos utilizado es conocido como de "Aplicación Distribuida"[Berson95], y se recomienda para aplicaciones cliente/servidor que son muy complejas, altamente interactivas o que tienen un intenso intercambio de entrada/salida con la base de datos.

El enfoque de "Aplicación Distribuida" considera que los procesos de cómputo relacionados con la base de datos son idealmente puestos en el nodo que contiene al sistema manejador de bases de datos, y la porción de los procesos de cómputo relacionados con las funciones de presentación, o que son ejecutados más económicamente en el cliente[Marion94], deben residir en dicho nodo.

Obviamente como el número de programas que cooperan y de nodos donde residen crecen, la complejidad del diseño de la aplicación y el manejo de la misma también crece. Esta clase de enfoque tiende a ser para propósitos especiales y el cliente podría tomar la forma de un programa escrito en un lenguaje de alto nivel como Cobol o C, en nuestro caso particular, hemos elegido el lenguaje de programación C++ por razones que se detallarán más adelante.

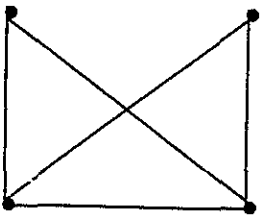
4.2.2. Teoría de grafos.

Es necesario definir brevemente algunos conceptos usados en teoría de grafos ya que en esta teoría se basará el modelado del problema que nos llevará a determinar el algoritmo a utilizar para lograr su solución, al encontrar una ruta de evacuación segura en caso de incendio.

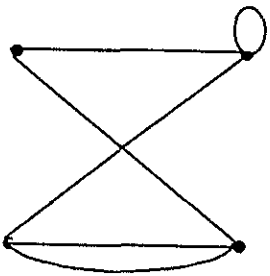
Un grafo consta de dos clases de elementos:

a) Un conjunto N cuyos elementos se llaman nodos, vértices o puntos.
b) Un conjunto S de parejas no ordenadas de nodos diferentes, llamadas segmentos o aristas.

Un multigrafo es un grafo con segmentos múltiples y lazos (los segmentos múltiples son segmentos que conectan a los mismos nodos y lazo es un segmento cuyas terminales son el mismo nodo).



a) Grafo



b) Multigrafo

Un ciclo es un camino (es decir una sucesión alternada de nodos y segmentos de la forma $v_0, s_1, v_1, s_2, v_2, \ldots, s_{n-1}, v_{n-1}, s_n, v_n$) cerrado (es decir, $v_0 = v_n$) tal que todos sus vértices son diferentes excepto v_0 que es igual a v_n.

Un grafo se dice que es conexo si hay una trayectoria entre dos cualesquiera de sus nodos (una trayectoria es un camino en el cual todos los nodos son diferentes).

Un grafo se dice que es acíclico o libre de ciclos si no contiene ciclo. Un árbol es un grafo acíclico conexo.

Un grafo dirigido, también llamado un digrafo, es un multigrafo con una dirección asignada a cada segmento. Llamamos arcos a los segmentos dirigidos.

Un nodo Terminal es un nodo que termina un camino. Meta es el nodo que es el objeto de la búsqueda.

Espacio de búsqueda es el conjunto de todos los nodos del grafo que representa un problema.

Camino Solución es un grafo dirigido de los nodos visitados que lleva a la solución.

Dentro de la problemática que se nos presenta para determinar la ruta de evacuación se ha identificado que los niveles de un edificio pueden ser modelados como un grafo dirigido. De esta forma, cada nodo representará un área y cada arco una conexión entre dos áreas.

4.2.3. Técnicas de búsqueda en Inteligencia Artificial.

La mayoría de los problemas que se incluyen en el ámbito de la Inteligencia Artificial, como el de encontrar la ruta de evacuación en un Edificio Inteligente, son demasiado complejos para ser resueltos mediante técnicas directas; es mejor intentar resolverlos por medio de métodos de búsqueda apropiados usando cualesquiera de las técnicas directas que esté a nuestra disposición para guiar la búsqueda.

Cada proceso de búsqueda puede verse como un corte transversal de un grafo dirigido, en el cual cada nodo representa un estado del problema, y cada arco representa una relación entre los estados representados por los nodos que conecta (En ocasiones el proceso de búsqueda puede describirse con facilidad como la travesía de un árbol, pero es un caso especial de la búsqueda en un grafo). El grafo que requiere ser explorado, debe, en principio estar construido enteramente mediante reglas que definen movimientos permitidos en el espacio del problema.

Una forma simple de realizar una estrategia de búsqueda es atravesando un árbol. Se expande cada nodo del árbol mediante las reglas de producción para generar un conjunto de nodos sucesores, cada uno de los cuales puede expandirse a su vez, continuando hasta que se encuentra un nodo que representa la solución. La realización de dicha contabilidad es sencilla y requiere poca contabilidad. Sin embargo, con frecuencia, este proceso genera el mismo nodo formando parte de diversos caminos y, por tanto, es procesado más de una vez. Esto sucede porque realmente lo que debería explorarse puede ser un grafo dirigido arbitrario en vez de un árbol.

El esfuerzo malgastado al generar el mismo nodo más de una vez puede evitarse al precio de una contabilidad adicional. En vez de atravesar un árbol de búsqueda, atravesamos un grafo dirigido. Este grafo difiere de un árbol en que diversos caminos pueden llegar a un mismo nodo.

El tratar el proceso de búsqueda como la búsqueda de un grafo en vez de la búsqueda en un árbol reduce la cantidad de esfuerzo que se invierte explorando el mismo camino varias veces. Pero requiere un esfuerzo adicional, cada vez que se genera un nodo, para ver si se ha generado anteriormente. Si este esfuerzo es justificado depende del problema concreto. Si es muy probable que se genere el mismo nodo en varios caminos, es más conveniente usar un procedimiento de grafo que si tal duplicación aconteciese sólo rara vez.

Sin embargo, la búsqueda puede no ser tan fácil como aparenta, en algunos casos el número de nodos en el espacio de búsqueda es muy grande y, a medida que el espacio de búsqueda crece, asimismo crece el número de diferentes posibles caminos para llegar al nodo meta. El problema es que cada nodo que se agrega al espacio de búsqueda se incrementará más rápidamente con cada nuevo nodo. De esta forma, como el número de posibilidades crece tan rápidamente todas las posibles combinaciones son imposibles de examinar o incluso de enumerar. A esto se le llama Explosión Combinatoria. Para combatir esta clase de problemas se puede utilizar la heurística, la cual ha demostrado ser una técnica útil.

4.2.4. Búsqueda Heurística.

En muchas ocasiones, es posible reducir el costo de la búsqueda con el auxilio de información peculiar de dicha búsqueda. Este tipo de información se suele denominar heurística, y los procedimientos de exploración que la utilizan se llaman métodos de exploración heurística.

Para solucionar eficientemente la mayoría de los problemas difíciles, a menudo es necesario comprometer los requerimientos de movilidad y sistematicidad, y construir una estructura de control que aunque no nos garantice que encontremos la mejor respuesta, por lo menos proporcione una respuesta muy buena. Una técnica heurística es aquella que mejora la eficiencia del proceso de búsqueda, posiblemente a cambio de sacrificar su completitud. Las técnicas heurísticas son como las guías turísticas. Son buenas cuando apuntan a direcciones interesantes; son malas cuando conducen a callejones sin salida. Algunas técnicas heurísticas ayudan a guiar un proceso de búsqueda sin sacrificar ninguna aspiración de completitud que el proceso pueda haber tenido previamente. Otras (de hecho, muchas de las mejores) pueden llevar a que pase inadvertido un camino excelente. Pero, en promedio, mejoran la calidad de los caminos que se exploran. Usando buenas técnicas heurísticas, podemos esperar lograr buenas (incluso óptimas) soluciones a problemas difíciles, en un tiempo menor que el exponencial (Por tiempo exponencial queremos indicar un tiempo proporcional a un número que crece con la potencia N, donde N es el tamaño de la entrada) Hay algunas buenas técnicas heurísticas de propósito general que son útiles en una gran variedad de dominios de problemas. Además, es posible construir técnicas heurísticas para propósitos especiales que exploten el conocimiento específico del dominio para resolver problemas particulares.

Sin técnicas heurísticas, estaríamos atrapados sin esperanza en la explosión combinatoria. Esto por sí solo sería argumento suficiente en favor de su uso, pero existen también otros argumentos:

- En realidad, pocas veces necesitamos la solución óptima; usualmente una buena aproximación nos servirá muy bien. De hecho, alguna evidencia muestra que la gente, al resolver problemas, no suele optimizar sino satisfacer los requerimientos. En otras palabras, busca cualquier solución que satisfaga un conjunto de requerimientos y, tan pronto como encuentra una, abandona la búsqueda.

- Aunque las aproximaciones producidas por las técnicas heurísticas pueden, en el peor de los casos, no ser muy buenas, los peores casos surgen raramente en el mundo real.

Una función heurística es una función que confecciona las medidas de deseabilidad a partir de las descripciones del estado del problema, y las representa usualmente como números. Se escogen los aspectos del estado a considerar y la forma de evaluarlos, así como los pesos asignados a los aspectos individuales, de forma que el valor de la función heurística en un nodo dado del proceso de búsqueda produzca la mejor estimación posible de si el nodo está en el deseado camino a una solución.

Las funciones heurísticas bien diseñadas pueden ser importantes para guiar eficientemente un proceso de búsqueda hacia una solución. Algunas veces, funciones heurísticas muy sencillas pueden proporcionar una estimación bastante acertada de si un camino es bueno o no. Algunas veces un alto valor en la función heurística indica una posición relativamente buena mientras que otras veces un valor bajo indica una situación ventajosa. El propósito de una función heurística es guiar el proceso de búsqueda en la dirección más provechosa, sugiriendo el camino a seguir en primer lugar cuando se dispone de más de uno. Cuanto más precisa sea la estimación que la función heurística hace de los méritos de cada nodo del árbol (o grafo) de búsqueda, más directa será la solución del proceso. En el caso extremo, la función heurística sería tan buena que, en esencia, no se requeriría ninguna búsqueda. El sistema se movería directamente hacia una solución.

Para el desarrollo del Agente Inteligente se ha decidido, en función de la clase de problema que se presenta y de la importancia que presenta el tiempo de respuesta, utilizar una búsqueda heurística, razón por la cual nuestro Agente es de Racionalidad Limitada.

4.2.5. Algoritmo de Hill Climbing.

Como ya se ha mencionado anteriormente, los niveles del Edificio Inteligente serán modelados como un grafo dirigido. En donde, cada nodo representará un área y cada arco una conexión entre dos áreas. De esta manera, la búsqueda de una ruta dentro de dicho grafo puede ser encontrada utilizando alguna de las técnicas de búsqueda en un grafo. Existen las técnicas de búsqueda de inteligencia artificial, de programación entera y de métodos de

investigación de operaciones. Éstas técnicas se diferencian por la estrategia que utilizan para recorrer el grafo buscando una solución al problema. Tanto las técnicas de investigación de operaciones y de programación entera tienden a encontrar la mejor ruta, por su parte, las de inteligencia artificial tienden hacia la búsqueda de una buena ruta, sin que ésta sea la óptima.

Tomando en cuenta que la cantidad de información necesaria para modelar un edificio es significativamente grande, consideramos que lo más eficiente es utilizar un algoritmo de inteligencia artificial en virtud de la extrema importancia que tiene el factor tiempo dentro del dominio del problema que se está tratando (la generación de una ruta de evacuación en caso de incendio).

Como ya se vio en el Capítulo I, las técnicas de búsqueda de Inteligencia Artificial se dividen en Breadth First y Depth First (Anchura Primero y Profundidad Primero).

Hemos decidido utilizar el algoritmo de Hill Climbing, que es de tipo Depth First, ya que nos es útil en vista de que el grafo dirigido que utilizaremos no es de profundidad infinita, además de que como veremos, éste algoritmo nos permite minimizar el número de áreas por los que se habrán de mover las personas en caso de evacuación, lo cual es benéfico tomando en cuenta que la distancia que separa a un cuarto del otro no es muy diferente la mayoría de las veces.

Existen varios problemas en los que es necesario buscar el menor número de conexiones (por ejemplo entre nodos dentro de un grafo) que se deben hacer para recorrer un determinado camino. Un algoritmo de búsqueda que intente encontrar como solución la que minimice el número de conexiones usará la heurística que conciba que una mayor distancia cubierta, mayor es la posibilidad de encontrarse más cerca de su destino; de este modo, reduce el número de conexiones.

Formalmente, el algoritmo de Hill Climbing escoge como próximo paso el nodo que aparezca en el lugar más cercano a la meta. El algoritmo deriva su nombre de la analogía de un excursionista perdido en la oscuridad, a la mitad de una montaña; entonces, incluso en la

oscuridad, el excursionista sabe que cualquier paso hacia arriba lo sitúa en la dirección correcta.

En muchos casos este algoritmo es bastante bueno porque tiende a reducir el número de nodos a visitar antes de encontrar una solución. No obstante, tiene tres grandes inconvenientes:

Problemas	*Soluciones*
Problema de la colina falsa. Es un estado que es mejor que todos sus vecinos pero no es mejor que algunos otros estados más lejanos.	Regresar a algún nodo previo e intentar ir en una dirección diferente. Esto es particularmente razonable si en ese nodo había otra dirección que parecía prometedora o al menos tan prometedora como la que se escogió anteriormente. Para realizar esta estrategia, hay que mantener una lista de caminos posibles y regresar a uno de ellos si el camino que se tomó conduce a un punto muerto.
Mesetas. Es un área plana en el espacio de búsqueda, en la cual todo el conjunto de estados vecinos tiene el mismo valor, así que todos los pasos siguientes parecen igual de buenos o malos. En este caso, el algoritmo de Hill Climbing no es mejor que una búsqueda de Primero en Profundidad.	Hacer un gran salto en alguna dirección para intentar llevar a una nueva sección del espacio de búsqueda. Esto es particularmente apropiado cuando tratamos con mesetas. Si las únicas reglas disponibles describen pequeños pasos elementales, aplicarlas diversas veces en la misma dirección.

Crestas. Es un área del espacio de búsqueda que es más alta que áreas colindantes pero no puede atravesarse mediante movimientos elementales en cualquier dirección. El algoritmo provocará que crucemos la cresta varias veces durante el retroseguimiento.	Aplicar dos o más reglas antes de realizar la comprobación. Esto corresponde a moverse en varias direcciones a la vez. Es una estrategia particularmente buena cuando se esta tratando con crestas.

4.2.5.1. El procedimiento general del Hill-Climbing.

El procedimiento general que se contempla en los programas que utilizan el algoritmo de Hill-Climbing es el siguiente:

1. Generar una solución posible. Es decir, generar un camino desde el estado inicial. Ver si es una solución. Si lo es, terminar. De lo contrario, continuar.

2. Aplicar algunas reglas para generar un nuevo conjunto de soluciones propuestas.

3. Para cada elemento del conjunto hacer lo que sigue:
   3.1. Enviarlo a la función de comprobación. Si es la solución, acabar.
   3.2. Si no lo es, ver si esta más cerca de la solución que cualquiera de los elementos comprobados hasta el momento. Si lo es, recordarla. Si no lo es, olvidarla.

4. Tomar el mejor elemento encontrado anteriormente y usarlo como la siguiente solución propuesta. Este paso corresponde a un movimiento a través del espacio del problema en la dirección que parece conducir más rápidamente hacia la meta.

5. Regresar al paso número 2.

4.2.6. Inteligencia Artificial Orientada a Objetos.

La Inteligencia Artificial Orientada a Objetos (OOAI por sus siglas en inglés) es un ambiente de implementación de técnicas de Inteligencia Artificial que usa el paradigma de diseño y programación orientada a objetos[Tracy97]. Hemos decidido utilizar este enfoque para el desarrollo del Agente Inteligente ya que las técnicas de Inteligencia Artificial, como las que han sido utilizadas para el diseño e implementación del Agente, se ven frecuentemente utilizadas dentro de sistemas de software tradicionales, los cuales, debido a la gran aceptación que actualmente disfruta el paradigma orientado a objetos pueden encontrarse escritos, o estar en proceso de serlo, en algún Lenguaje de Programación Orientado a Objetos (OOPL por sus siglas en inglés).

Por otra parte, la programación orientada a objetos provee una estructura excelente para los programas de Inteligencia Artificial. Las clases permiten lograr una ventajosa representación de Tipos de Datos Abstractos con lo cual dichos programas pueden ser escritos alrededor de los datos que son relevantes al problema, lo cual en nuestro caso facilita la representación y uso de los distintos elementos necesarios para representar nuestro modelo.

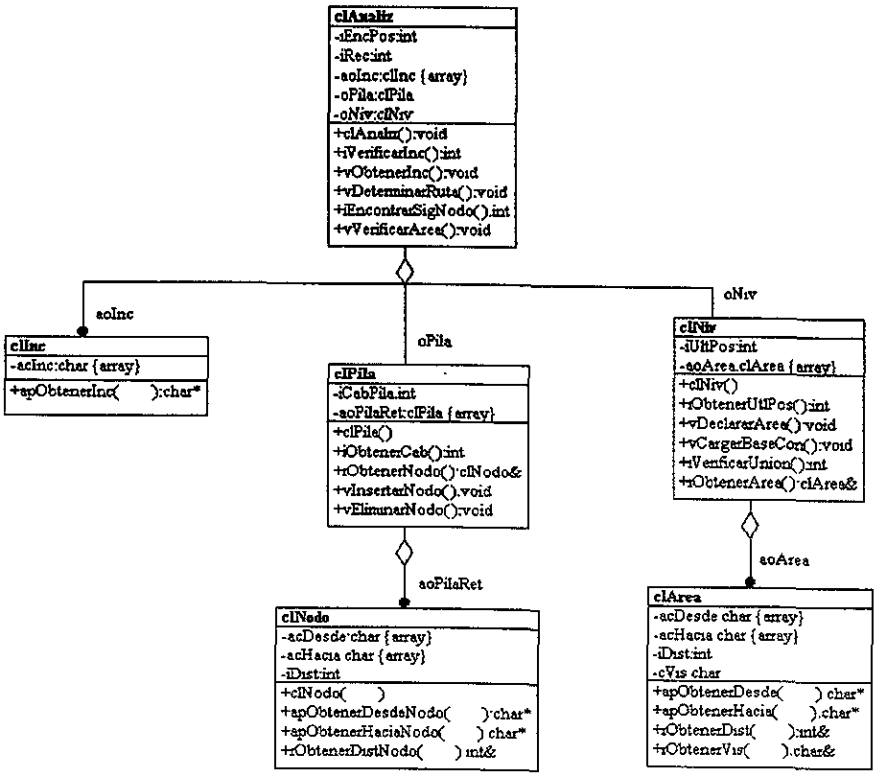
4.2.7. Diseño de Clases.

Para la realización del diagrama de clases, que se utilizará para la implementación del algoritmo de Hill-Climbring con un lenguaje orientado a objetos, hemos utilizado la notación propuesta por la metodología de análisis orientado a objetos OMT (Object Modeling Technique) [Rumbaugh91] por considerarla como una de las más claras y ampliamente difundidas actualmente, además de que posee una riqueza notacional que permite modelar con facilidad todos los elementos del dominio del problema.

El diagrama de clases comprende los siguientes elementos:

- Clases. Representadas por un rectángulo. La parte superior indica el nombre de la clase, la parte intermedia representa los datos miembro, indicando asimismo el tipo de dato a que pertenecen, y la inferior las funciones miembro, identificando el tipo de dato que regresan. Los signos de más o menos indican si se considera un elemento como público o privado respectivamente.
- Las líneas continuas con un rombo definen una relación donde se representa la instanciación de una clase en otra. El nombre de la instancia se pone en la línea de la *relación que sale de la clase instanciada.*

A continuación se presenta el Diagrama de Clases del Agente Inteligente:



4.3. IMPLEMENTACIÓN DEL AGENTE INTELIGENTE.

4.3.1. El lenguaje C++.

Para la implementación del diseño del Agente Inteligente hemos elegido al lenguaje de programación C++ en vista de las múltiples ventajas que nos ofrece. Una gran ventaja de la utilización de C++ para programas de Inteligencia Artificial es que su uso es muy conveniente en sistemas de gran tamaño. Muchos programas de Inteligencia Artificial son parte de sistemas de software más grandes y deben tener una interfaz con ellos. Así, cuando los programas de Inteligencia Artificial están en el mismo lenguaje o en un lenguaje similar al de todo el sistema, el ligado se facilita en extremo. Además, la utilización de C++ puede mejorar el desempeño de los algoritmos de Inteligencia Artificial, los cuales pueden ser muy lentos cuando son implementados en lenguajes como LISP o Prolog. Por otro lado, este lenguaje de programación cuenta con una interfaz con Sybase, conocida como DB-Library, mediante la cual se posibilita el desarrollo de programas escritos en un lenguaje de programación tradicional y en el que se encuentran incrustadas instrucciones SQL. Gracias a lo anterior es posible que el Agente Inteligente haga uso de la información generada por el sistema de monitoreo o por el usuario y almacenada en una base de datos relacional en Sybase.

4.3.1.1. Características del Lenguaje.

Consideramos necesario el presentar alguna de las características del lenguaje de programación C++ a fin de entender mejor sus bondades y limitaciones, lo cual permite contemplar algunas de las ventajas y desventajas de la implementación del Agente Inteligente.

- Es catalogado como un lenguaje de nivel medio, ya que combina las capacidades de un lenguaje de bajo nivel como el ensamblador o lenguaje de máquina y las capacidades de un lenguaje de alto nivel como Pascal, Algol o Modula. Además se le incorporaron las funciones de un lenguaje orientado a

objetos. Igualmente, es catalogado como un lenguaje de programación orientado a objetos (OOPL por sus siglas en inglés) híbrido, ya que permite tanto la programación estructurada como la orientada a objetos, en contraposición con los lenguajes llamados puros, que solo permiten la programación orientada a objetos.

- Al tener las capacidades de bajo nivel permite la manipulación directa de bits, bytes y direcciones de memoria. Esto hace al lenguaje ideal para la programación de sistemas, en sustitución del lenguaje ensamblador que no es portable a otras máquinas y que no es fácil de depurar.

- Al tener las capacidades de alto nivel proporciona las facilidades de control de flujo, subrutinas y estructuras de datos simples, esto le da las características para ser empleado como lenguaje de propósito general.

- Es un lenguaje estructurado porque tiene estructuras de control: while-do, do-while, etc. Además tiene la capacidad de subrutinas parametrizadas. Permite la definición de funciones a manera de subrutinas independientes que soportan el paso de parámetros por valor y referencia, y llamadas recursivas. Sin embargo no se permite el anidamiento de funciones.

- Aunque contempla una amplia variedad de estructuras de datos simples C/C++ es un lenguaje débilmente tipificado, debido a que no existe una comprobación estricta de los tipos de datos usados en el programa. Esta característica hace más flexible y peligroso al lenguaje, ya que si bien el programador puede convertir con facilidad un tipo de datos en otro, puede ser que el programador lo hiciera sin darse cuenta y con ello que se ocasionaran fallas en el programa.

- No existe comprobación al tiempo de ejecución de desbordamiento o rebaso de límites al usar vectores o matrices. Esta característica hace más veloz al lenguaje, sin embargo también lo hace más peligroso, pues el desbordamiento o

rebaso de los límites de un vector es en general un error muy difícil de detectar y que puede ocurrirle con frecuencia a un programador.

- En general la filosofía del lenguaje C/C++ es asumir que el programador es un experto en el lenguaje, que desea código tan pequeño y rápido como sea posible. Debido a esto los compiladores C/C++ evitan las comprobaciones al tiempo de ejecución. C/C++ deja al programador la opción de realizar comprobaciones al tiempo de ejecución.

- Es un lenguaje relativamente compacto pues esta formado por 32 palabras reservadas para el lenguaje C y 16 más para C++ para un total de 48. El Basic de una computadora personal, por ejemplo, está formado por 148.

4.3.1.2. Ventajas y Desventajas.

Las ventajas y desventajas del lenguaje de programación C++ son presentadas a continuación en un cuadro comparativo:

Ventajas	*Desventajas*
• Es un lenguaje estándar. El código es altamente portable, es decir compatible entre computadoras de distintos fabricantes. • El código compilado es más pequeño y rápido, que el mismo código compilado con otros lenguajes. • Es un lenguaje compacto. • Se tiene un poder virtualmente ilimitado sobre la computadora, ya que escribe en cualquier parte de la memoria.	• La flexibilidad del lenguaje en su sintaxis hace que se cometan errores con facilidad. • El programador debe ser extremadamente cuidadoso cuando trabaja con manipulaciones de vectores, matrices, apuntadores a memoria y conversiones de tipo, debido a que no se hacen comprobaciones en el tiempo de ejecución para detectar éstos errores.

4.3.2. DB-Library.

La DB-Library es un conjunto de rutinas y macros de C que permiten a las aplicaciones interactuar con el SQL Server de Sybase. Incluye rutinas que mandan comandos Transact-SQL al SQL Server y otros que procesan los resultados de dichos comandos. Otras rutinas manejan condiciones de error, realizan conversiones de datos, y proveen una variedad de información sobre la interacción con el SQL Server. La DB-Library también contiene varios archivos de cabecera (*.h) que definen estructuras y valores usados por las rutinas. Se han desarrollado versiones de DB-Library para numerosos lenguajes además de C, incluyendo Cobol, Fortran, Ada y Pascal.

La DB-Library permite que la base de datos se convierta en una parte integral de una aplicación. Los enunciados Transact-SQL pueden ser incorporados en la aplicación, permitiendo a la aplicación el ver y actualizar valores de la base de datos. A través de la DB-Library, los valores de la base de datos pueden ser puestos en las variables del programa para ser manipuladas por la aplicación. Asimismo, los valores en las variables del programa pueden ser insertados en la base de datos.

No obstante que la DB-Library contiene un gran número de rutinas, proporcionando a las aplicaciones un gran control sobre su interacción con el SQL Server, la mayoría de las aplicaciones sólo requieren usar unas pocas de ellas.

Programar con DB-Library típicamente involucra los siguiente pasos básicos:

1. Logging en el SQL Server.
2. Colocar los comando Transact-SQL en un buffer y mandarlos al SQL Server.
3. Procesar, si es que existen, los resultados regresados del SQL Server, un comando a la vez y un renglón de resultado a la vez. Los resultados pueden ser puestos en variables de programa, donde pueden ser manipulados por la aplicación.
4. Manejar los errores y mensajes del SQL Server.

5. Cerrar la conexión con el SQL Server.

4.3.3. Estándares de Programación.

Para la programación del Agente Inteligente fueron utilizados una serie de estándares que explicamos a continuación para la mejor comprensión y mantenimiento del código.

4.3.3.1. Notación Húngara.

En la codificación del programa se utiliza, aunque de manera simplificada, el convenio de nomenclatura Húngaro que fue desarrollado por Charles Simonyi a principios de los setentas. Su premisa es que es mas importante dar información al nombrar a los identificadores (de variables, arrays, constantes y funciones) que el poderlos pronunciar fácilmente o que solo tengan significación para el programador que los realizó.

Las reglas simplificadas que se utilizan son:

- Para las variables de cada tipo, se usa la abreviatura del tipo al inicio del nombre de la variable.
- Las abreviaturas de los tipos de datos son las mismas para todo el programa.
- En los casos de los apuntadores, se comienza su nombre con una ap.

El convenio de nomenclatura permite a los programadores añadir a la abreviatura del tipo de variable una o dos palabras descriptivas (cada una debe comenzar con mayúsculas).

Sin embargo, debido a que el propósito de la notación Húngara es incrementar la comprensión, el convenio pone mas énfasis en lo que representan los tipos de datos que en como se declaran. Vgr. si tenemos apuntadores a carácter, para el programador podría ser mas importante que son apuntadores a cadenas acabadas en cero, así, en vez de nombrarlo apcar podríamos nombrarlo str.

Una de las mayores ventajas de usar la notación Húngara es que se hace mas fácil descifrar las expresiones con apuntadores, especialmente los apuntadores a apuntadores.

4.3.4. Código del Agente Inteligente.

El código fuente del Agente Inteligente y sus bibliotecas puede ser consultado en su totalidad en el Apéndice B. La documentación de los códigos, junto con el diagrama de clases permiten lograr una visión integral de su diseño e implementación. No hay ningún texto introductorio a los códigos ya que estos se encuentran profusamente documentados y no se ha considerado necesario entrar en una inútil duplicidad de explicaciones sobre su funcionamiento.

4.3.5. Compilación del Agente Inteligente.

El Agente Inteligente se ejecuta bajo un ambiente Unix, en el mismo servidor en el que esta la base de datos. Para compilar el Agente Inteligente, desde la línea de comando de Unix se ejecuta el siguiente:

```
$ gcc -I. -I/opt/system_X/include nombre_programa.cpp /opt/system_X/lib/libsybdb.a -lm -lnsl -o
  nombre_programa.exe
```

donde:

gcc	Compilador de C, C++ y Objective C de *GNU* para Unix. Gcc es un compilador de alta calidad y muy portable. El compilador esta diseñado para soportar múltiples front-ends y múltiples back-ends mediante la traducción primero en un código intermedio y de ahí en el código ensamblador para la arquitectura objetivo. Nosotros hemos utilizado la versión 2.7.2.1.
-I.	Agrega el directorio donde está el archivo .cpp a la lista de directorios donde serán buscados los archivos de cabecera. Esta opción puede ser usada para substituir un archivo de cabecera del sistema por uno propio, ya que el directorio especificado con esta opción será buscado antes que el directorio donde están los archivos de cabecera del sistema. Si se utiliza más de una vez la opción -I, los directorios serán analizados en un orden de izquierda a derecha.
-I/opt/system_X/include	Agrega el directorio /opt/system_X/include a la lista de directorios donde serán buscados los archivos de cabecera.
Nombre_programa.cpp	Nombre del programa fuente en C++.
/opt/system_X/lib/libsybdb.a	Especifica la biblioteca de funciones a ser utilizada.
-lm	Opción de ligado.
-lnsl	Opción de ligado.
-o nombre_programa.exe	Crea el archivo ejecutable con el nombre de nombre_programa.exe

4.3.6. Ejecución.

Para poner en funcionamiento al Agente Inteligente, desde la línea de comando de Unix se ejecuta el siguiente:

```
$ nombre_programa.exe
```

donde:

nombre_programa.exe	Es el programa ejecutable que se obtuvo de la compilación anteriormente descrita

4.4. DISEÑO DEL SERVIDOR.

Se diseñó una base de datos utilizando el RDBMS (Sistema Administrador de Bases de Datos Relacionales) Sybase para que en ésta se almacenen los datos suministrados por el sistema de monitoreo que esta utilizando nuestro Agente Inteligente. El diseño modela las características generales del Edificio Inteligente poniendo especial énfasis en el modelado de los datos suministrados por el sistema de monitoreo y que son indispensables para el correcto funcionamiento del Agente inteligente, es decir, los datos necesarios para determinar una ruta de evacuación en caso de incendio.

4.4.1. El RDBMS Sybase.

Sybase Inc. fue fundada a fines de 1984, y no fue sino hasta mediados de 1987 que comenzó a comercializar el primer DBMS relacional especialmente diseñado para procesamiento de transacciones en línea.

Sybase fue el primero en desarrollar un DBMS relacional que fue diseñado para repartir el procesamiento entre un front-end que corriera en la estación de trabajo y que tuviera acceso a los datos comunicándose para ello con el motor de la base de datos (engine), el cual corre en el servidor. Esta arquitectura fue bautizada por Sybase con diferentes nombres, "host/servidor", "solicitante/servidor", hasta evolucionar en "cliente/servidor", después de que la compañía Forrest Research publicara un artículo denominado "El nuevo paradigma cliente/servidor" a propósito del anuncio de que Microsoft produciría el SQL Server de Sybase para correr en su sistema operativo.

Sybase ha establecido estándares en el manejo cliente/servidor para DBMSs relacionales, en gran parte debido a que la combinación de la tecnología de Sybase con el nombre y mercadotecnia de Microsoft estimularon la industria de productos cliente/servidor.

La versión conocida como System 10 fue liberada para algunas plataformas a mediados de 1993, y por un periodo de dos años, es decir, hasta 1995, Sybase continuo liberando el producto hasta cubrir todas las plataformas en las que tradicionalmente corren sus productos.

4.4.1.1. Sybase para Unix

El SQL Server de Sybase es el rival más cercano de Oracle y, hasta antes de la liberación de la versión 7.0 del servidor Oracle, ha sido considerado como el líder tecnológico entre los servidores de bases de datos para sistemas Unix.

Una buena referencia sobre Sybase es que muchas de las compañías involucradas en mercados financieros tales como Wall Street han venido usando Sybase por más de cinco años, lo que habla de su alto nivel de desempeño, el cual ha sido confirmado en diferentes benchmarks.

A los comentarios señalados cuando se habló del producto para PC's, podemos agregar lo siguiente.

4.4.1.1.1. CARACTERÍSTICAS

1. Sybase soporta, de manera nativa, sistemas de multiproceso simétrico, gracias a su arquitectura VSA (Virtual Server Arquitecture), lo que permite obtener un incremento en el desempeño del 30 al 50 % sobre las versiones normales de Unix, así como la capacidad de soportar hasta 1,000 usuarios simultáneos de la base de datos.

2. Sybase también pude integrar datos de otros DBMSs con sus propios datos gracias a sus APIs Open Client y Open Server. Open Client soporta acceso de clientes a través de diferentes protocolos de red, incluyendo TCP/IP. Open Server es el gateway a otros servidores de bases de datos, incluyendo Oracle, Informix e INGRES.

3. Portabilidad y Escalabilidad.

4. Soporta más de 30 versiones diferentes de Unix, y es el único entre los servidores relacionales que tiene una versión para estaciones de trabajo NeXT.

5. El lenguaje nativo del SQL Server de Sybase es Transact-SQL, además Sybase ofrece otras herramientas, corriendo bajo Unix, que utilizan el lenguaje APT-SQL, un SQL adicionado con elementos 4GL, para acelerar el desarrollo de aplicaciones.

6. Todas las versiones de SQL Server de Sybase incluyen la utilería de SQL interactivo ISQL (modo carácter).

7. Soporta innumerables front-ends de terceros. Sybase mismo ofrece algunas herramientas: APT Workbench, un ambiente de desarrollo basado en formas y que incluye APT Build, un generador de código y desarrollo de prototipos; así como Data Workbench, una interface de pantalla completa SQL que incluye un generador de reportes.

8. Sybase también soporta desarrollos en lenguajes tales como C, COBOL y Fortran.

4.4.2. Configuración de las variables de ambiente.

Las variables de ambiente necesarias para el buen funcionamiento del Agente Inteligente que utiliza Sybase son:

1. DSQUERY. Esta variable define el nombre del SQL Server al cual los programas cliente intentaran conectarse si no se ha especificado el nombre del SQL Server dentro del mismo programa cliente (por medio de la función dbopen).

2. SYBASE. Esta variable de ambiente define la ruta del directorio donde esta instalado Sybase. Si la variable de ambiente SYBASE no es definida, entonces: sybinit usará la ruta /usr/u/sybase como el directorio donde esta instalado Sybase.

4.4.3. Diagrama Entidad Relación de BDEI

Para la realización del Diagrama Entidad Relación de la base de datos utilizada por el Agente Inteligente, llamada BDEI (Base de Datos del Edificio Inteligente), hemos utilizado el estándar de notación IDEF1X.

La descripción de los campos de las tablas de la base de datos BDEI se encuentran en el Apéndice C.

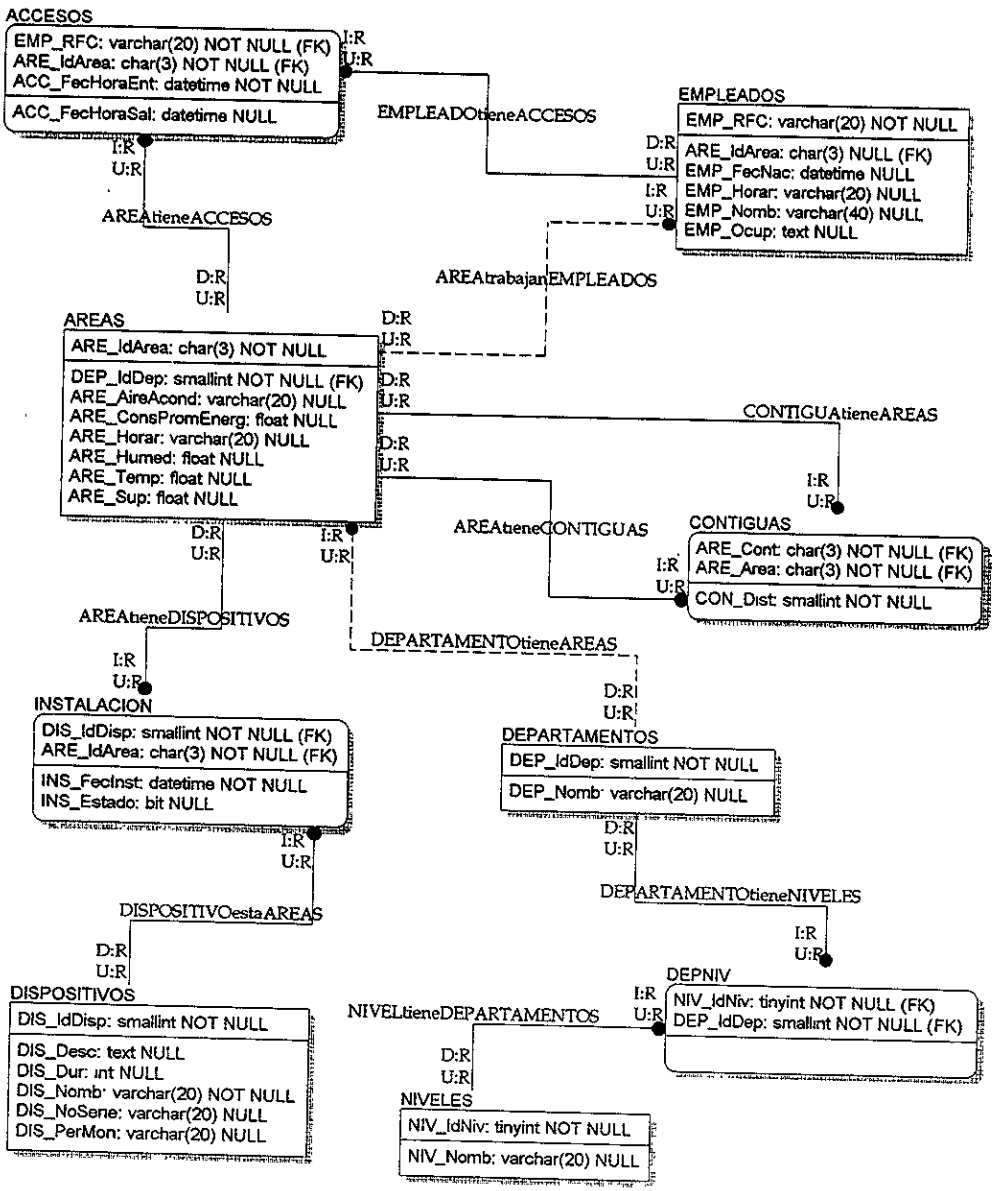
Esta notación comprende los siguientes elementos:

- Entidades independientes. Es una entidad cuyas instancias pueden ser identificadas únicamente sin determinar su relación con otra entidad.

- Entidades dependientes. Es una entidad cuyas instancias no pueden ser identificadas únicamente sin determinar su relación con otra entidad o entidades..
- Relación que identifica. Es aquella relación que contribuye a identificar a la entidad hijo en una relación al migrar la llave primaria del padre en la llave primaria del hijo; y de esta manera haciendo que el hijo dependa del padre para su identidad.

- Relación que no identifica. Es aquella relación que no hace que la entidad hijo dependa del padre. Se indica por medio de una línea punteada con un punto en el hijo. En esta relación, la llave primaria del padre es migrada como atributo no primario a la entidad hijo. Si estos atributos migrados no son requeridos en el hijo, la relación se llama Relación que no identifica opcional, lo cual significa que la instancia de la entidad hijo puede existir sin que ninguna instancia del padre este asociada a ella. Esta relación se identifica por medio de una línea punteada con un punto en el hijo y un diamante en el padre.
- Relación recursiva de red. También es llamada Recursión de tabla doble, y establece una relación de red o web entre el padre y el hijo, donde un padre puede tener cualquier número de hijos, y un hijo puede tener cualquier número de padres. En ambos casos, todas las asociaciones son pares de llaves primarias de la misma tabla, ya que cada una representa un significado diferente. La recursión de red es una situación en la cual una entidad tiene una relación muchos a muchos consigo misma. Cuando este problema aparece, puede ser resuelto creando una entidad intermedia y convirtiendo la relación muchos a muchos en dos relaciones uno a muchos. Se deben asignar los nombres de rol a las llaves foráneas para capturar el significado de la relación recursiva.

A continuación se presenta el Diagrama Entidad Relación de la base de datos BDEI:


ACCESOS
EMP_RFC: varchar(20) NOT NULL (FK)
ARE_IdArea: char(3) NOT NULL (FK)
ACC_FecHoraEnt: datetime NOT NULL
ACC_FecHoraSal: datetime NULL
I:R
U:R
EMPLEADOtieneACCESOS
EMPLEADOS
EMP_RFC: varchar(20) NOT NULL
ARE_IdArea: char(3) NULL (FK)
EMP_FecNac: datetime NULL
EMP_Horar: varchar(20) NULL
EMP_Nomb: varchar(40) NULL
EMP_Ocup: text NULL
D:R
U:R
AREAtieneACCESOS
AREAtrabajanEMPLEADOS
AREAS
ARE_IdArea: char(3) NOT NULL
DEP_IdDep: smallint NOT NULL (FK)
ARE_AireAcond: varchar(20) NULL
ARE_ConsPromEnerg: float NULL
ARE_Horar: varchar(20) NULL
ARE_Humed: float NULL
ARE_Temp: float NULL
ARE_Sup: float NULL
CONTIGUAtieneAREAS
AREAtieneCONTIGUAS
CONTIGUAS
ARE_Cont: char(3) NOT NULL (FK)
ARE_Area: char(3) NOT NULL (FK)
CON_Dist: smallint NOT NULL
AREAtieneDISPOSITIVOS
DEPARTAMENTOtieneAREAS
INSTALACION
DIS_IdDisp: smallint NOT NULL (FK)
ARE_IdArea: char(3) NOT NULL (FK)
INS_FecInst: datetime NOT NULL
INS_Estado: bit NULL
DEPARTAMENTOS
DEP_IdDep: smallint NOT NULL
DEP_Nomb: varchar(20) NULL
DISPOSITIVOestaAREAS
DEPARTAMENTOtieneNIVELES
DISPOSITIVOS
DIS_IdDisp: smallint NOT NULL
DIS_Desc: text NULL
DIS_Dur: int NULL
DIS_Nomb: varchar(20) NOT NULL
DIS_NoSene: varchar(20) NULL
DIS_PerMon: varchar(20) NULL
DEPNIV
NIV_IdNiv: tinyint NOT NULL (FK)
DEP_IdDep: smallint NOT NULL (FK)
NIVELtieneDEPARTAMENTOS
NIVELES
NIV_IdNiv: tinyint NOT NULL
NIV_Nomb: varchar(20) NULL

4.5. IMPLEMENTACIÓN DEL SERVIDOR.

4.5.1. Código de generación de BDEI

El código de generación de la base de datos BDEI se obtuvo automáticamente mediante la utilización de la herramienta CASE de modelado relacional de datos ErWin versión 2.5.01. A ese código se le cambiaron los mensajes mostrados en caso de un intento de infracción a la integridad referencial, ya que éstos por omisión son generados en inglés.

El código de generación de la base de datos puede ser consultado en su totalidad en el Apéndice A.

4.6. PRUEBAS DE FUNCIONAMIENTO.

4.6.1. Casos de incendio usando el Diseño Interior Propuesto.

Los casos de incendio con los que fue probado el desempeño del Agente Inteligente son presentados a continuación mediante un formato tabular:

Áreas donde se detectó presencia humana	*Salida de Emergencia.*	*Áreas donde se detectó un incendio*	*Ruta de evacuación generada por el Agente Inteligente*
025	000	010, 034, 024	025 hacia 013 hacia 025 hacia 026 hacia 020 hacia 006 hacia 016 hacia 001 hacia 000.

Áreas donde se detectó presencia humana	*Salida de Emergencia.*	*Áreas donde se detectó un incendio*	*Ruta de evacuación generada por el Agente Inteligente*
007	000	012, 034, 029	007 hacia 017 hacia 001 hacia 000.
027	029	034, 002, 005	Esta atrapado
018	029	011, 001	018 hacia 005 hacia 034 hacia 004 hacia 029
002	000	025, 032, 034, 024	002 hacia 003 hacia 021 hacia 015 hacia 010 hacia 001 hacia 000
024	029	014, 034, 000	024 hacia 007 hacia 012 hacia 007 hacia 017 hacia 001 hacia 004 hacia 029

Las pantallas del Agente Inteligente pueden ser vistos en el Apéndice D.

CONCLUSIONES

A lo largo de este trabajo de investigación hemos llegado a algunas conclusiones. A continuación las presentamos:

1. Las definiciones del término Inteligencia Artificial, según el objetivo perseguido, pueden identificarse con alguno de los cuatro enfoques principales: Sistemas que piensan como humanos; Sistemas que piensan racionalmente; Sistemas que actúan como humanos; Sistemas que actúan racionalmente.

2. El enfoque que consideramos como el más adecuado para nuestro objetivo, demostrar la factibilidad de utilización y modelar algunas de las características de los sistemas inteligentes encargados de apoyar la toma de decisiones en caso de emergencia dentro de un Edificio Inteligente, es el de Sistemas que actúan racionalmente, ya que este enfoque se aleja de las discusiones de carácter filosófico y psicológico sobre la inteligencia y permite concentrase en la labor de actuar de manera que se emprenda la mejor acción posible en una situación dada según los objetivos deseados, con base en ciertos supuestos.

3. Un agente es todo aquello que *puede considerarse* que percibe su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores. En el caso de los agentes de software, sus percepciones y acciones vienen a ser las cadenas de bits codificados.

4. Un agente racional es aquel que busca obtener el mejor desempeño posible. La racionalidad contempla un éxito esperado, tomando como base lo que se ha percibido.

5. La programación orientada a objetos es un paradigma de la ingeniería de software, cuya filosofía es un modelado de software basado en objetos del mundo real. la programación orientada a objetos puede reducir significativamente la complejidad del problema en comparación con otras técnicas anteriores como la programación estructurada.

6. El enfoque Cliente/Servidor permite diseñar una aplicación de tal forma que los componentes funcionales de una aplicación son repartidos de manera que les permite estar dispersos, y ser ejecutados en múltiples plataformas de cómputo diversas compartiendo el acceso a uno o más repositorios de datos compartidos.

7. El concepto de calidad conlleva a la *optimización* de recursos humanos y materiales. Al desarrollarse tal concepto, se inicia la era del *Edificio Inteligente*, ya que uno de los factores principales es ahorrar recursos financieros y humanos, esto es, ser *eficientes* en todas las áreas de la organización.

8. Un *Edificio Inteligente* es aquel que provee un ambiente productivo y eficiente de trabajo a través de la *optimización* de cuatro de sus elementos básicos: *estructura*, sistemas, *servicios* y administración; relacionándolos entre sí para un mejor rendimiento.

9. Los tres factores a integrar en un Edificio Inteligente son: Flexibilidad del Edificio Inteligente; Integración de servicios y Diseño interior y exterior.

10. El Sistema Inteligente es el que controla y coordina los diferentes recursos de un Edificio Inteligente, se encuentra integrado por : Módulos de control distribuido; Estación de supervisión y una Red de recursos ambientales.

11. Los *servicios* y *suministros* de los edificios de oficinas, requieren una *optimización* en los procesos de planeación, instalación, operación y mantenimiento. Por ello, se observan tendencias muy claras en los equipos nuevos: los sistemas de control se integran con *sistemas informáticos*, se buscan niveles básicos de inteligencia en los componentes de modo que los *sensores* tomen decisiones a nivel de microproceso sin involucrar a un sistema altamente complejo se cuenta con un *control distribuido* pero a la vez de cooperación entre áreas independientes.

12. Los suministros y servicios más importantes a establecer son: Aire Acondicionado; Sistema eléctrico; Redes de Cómputo; Sistema de Seguridad; Hidráulico y Sanitarios; Edificación e Iluminación.

13. Aunque el Sistema de detección y Extinción de Incendios no influye en forma directa en el concepto de costo beneficio, lo hace en forma indirecta al bajar las primas de seguro y al considerar que las vidas de los usuarios son valiosas y que cuando ocurre un siniestro, el costo de la inversión es mucho menor a los daños que se pueden evitar y vidas que se pueden perder.

14. Es condición necesaria para que un edificio pueda ser considerado como inteligente, el que tenga un conjunto de sistemas basados en técnicas de inteligencia artificial que le permitan desempeñar por sí mismo diversas actividades con una efectividad que iguale o mejore el desempeño que podría tener un ser humano. Entre las actividades más importantes que pueden caer dentro de este rubro está el tomar las decisiones necesarias o apoyar la toma de decisiones en caso de emergencia. A esta actividad es a la cual esta enfocada el Agente Inteligente que se desarrollo durante esa investigación.

15. En este trabajo de investigación se desarrolló un Agente Inteligente encargado de apoyar la toma de decisiones en caso de emergencia dentro de un Edificio Inteligente, mediante la utilización de una base de conocimientos y técnicas de inteligencia artificial. Este Agente se encarga de solucionar el problema de encontrar una ruta de evacuación segura para evacuar a todas las personas que se encuentren en un Edificio Inteligente en caso de incendio.

16. Para el desarrollo del Agente Inteligente se utilizaron el paradigma de la orientación a objetos aplicado a la Inteligencia Artificial y la arquitectura Cliente/Servidor por las ventajas que ofrecen al ser de gran aceptación y actualidad. Asimismo, se eligió utilizar el lenguaje de programación C++ para la codificación por ser un lenguaje adecuado para la implementación de algoritmos de inteligencia artificial y tener una interfaz con el RDBMS Sybase, el cual fue elegido por tener la capacidad suficiente para soportar el modelado de todo un edificio.

Apéndice A. Código de Generación de la Base de Datos BDEI.

El código de generación de la base de datos BDEI se obtuvo automáticamente mediante la utilización de la herramienta CASE de modelado relacional de datos ErWin versión 2.5.01. A ese código se le cambiaron los mensajes mostrados en caso de un intento de infracción a la integridad referencial, ya que éstos por omisión son generados en inglés.

```
CREATE TABLE ACCESOS (
     EMP_RFC              varchar(20) NOT NULL,
     ARE_IdArea           char(3) NOT NULL,
     ACC_FecHoraEnt       datetime NOT NULL,
     ACC_FecHoraSal       datetime NULL
)
go


ALTER TABLE ACCESOS
     ADD  PRIMARY KEY (EMP_RFC, ARE_IdArea, ACC_FecHoraEnt)
go

exec sp_primarykey ACCESOS,
     EMP_RFC,
     ARE_IdArea,
     ACC_FecHoraEnt
go

CREATE TABLE AREAS (
     ARE_IdArea           char(3) NOT NULL,
     DEP_IdDep            smallint NOT NULL,
     ARE_AireAcond        varchar(20) NULL,
     ARE_ConsPromEnerg    float NULL,
     ARE_Horar            varchar(20) NULL,
     ARE_Humed            float NULL,
     ARE_Temp             float NULL,
     ARE_Sup              float NULL
)
go


ALTER TABLE AREAS
     ADD  PRIMARY KEY (ARE_IdArea)
go

exec sp_primarykey AREAS,
     ARE_IdArea
go
```

```
CREATE TABLE CONTIGUAS (
    ARE_Cont        char(3) NOT NULL,
    ARE_Area        char(3) NOT NULL,
    CON_Dist        smallint NOT NULL
)
go


ALTER TABLE CONTIGUAS
    ADD  PRIMARY KEY (ARE_Cont, ARE_Area)
go

exec sp_primarykey CONTIGUAS,
    ARE_Cont,
    ARE_Area
go

CREATE TABLE DEPARTAMENTOS (
    DEP_IdDep       smallint NOT NULL,
    DEP_Nomb        varchar(20) NULL
)
go


ALTER TABLE DEPARTAMENTOS
    ADD  PRIMARY KEY (DEP_IdDep)
go

exec sp_primarykey DEPARTAMENTOS,
    DEP_IdDep
go

CREATE TABLE DEPNIV (
    NIV_IdNiv       tinyint NOT NULL,
    DEP_IdDep       smallint NOT NULL
)
go


ALTER TABLE DEPNIV
    ADD  PRIMARY KEY (NIV_IdNiv, DEP_IdDep)
go

exec sp_primarykey DEPNIV,
    NIV_IdNiv,
    DEP_IdDep
go
```

```
CREATE TABLE DISPOSITIVOS (
    DIS_IdDisp        smallint NOT NULL,
    DIS_Desc          text NULL,
    DIS_Dur           int NULL,
    DIS_Nomb          varchar(20) NOT NULL,
    DIS_NoSerie       varchar(20) NULL,
    DIS_PerMon        varchar(20) NULL
)
go


ALTER TABLE DISPOSITIVOS
    ADD  PRIMARY KEY (DIS_IdDisp)
go

exec sp_primarykey DISPOSITIVOS,
   DIS_IdDisp
go

CREATE TABLE EMPLEADOS (
    EMP_RFC           varchar(20) NOT NULL,
    ARE_IdArea        char(3) NULL,
    EMP_FecNac        datetime NULL,
    EMP_Horar         varchar(20) NULL,
    EMP_Nomb          varchar(40) NULL,
    EMP_Ocup          text NULL
)
go


ALTER TABLE EMPLEADOS
    ADD  PRIMARY KEY (EMP_RFC)
go

exec sp_primarykey EMPLEADOS,
   EMP_RFC
go

CREATE TABLE INSTALACION (
    DIS_IdDisp        smallint NOT NULL,
    ARE_IdArea        char(3) NOT NULL,
    INS_FecInst       datetime NOT NULL,
    INS_Estado        bit
)
go


ALTER TABLE INSTALACION
    ADD  PRIMARY KEY (DIS_IdDisp, ARE_IdArea)
go



exec sp_primarykey INSTALACION,
   DIS_IdDisp,
   ARE_IdArea
```

```
go

CREATE TABLE NIVELES (
    NIV_IdNiv       tinyint NOT NULL,
    NIV_Nomb        varchar(20) NULL
)
go


ALTER TABLE NIVELES
    ADD PRIMARY KEY (NIV_IdNiv)
go

exec sp_primarykey NIVELES,
    NIV_IdNiv
go

ALTER TABLE ACCESOS
    ADD FOREIGN KEY (EMP_RFC)
        REFERENCES EMPLEADOS
go


ALTER TABLE ACCESOS
    ADD FOREIGN KEY (ARE_IdArea)
        REFERENCES AREAS
go


exec sp_foreignkey ACCESOS, EMPLEADOS,
    EMP_RFC
go

exec sp_foreignkey ACCESOS, AREAS,
    ARE_IdArea
go

ALTER TABLE AREAS
    ADD FOREIGN KEY (DEP_IdDep)
        REFERENCES DEPARTAMENTOS
go


exec sp_foreignkey AREAS, DEPARTAMENTOS,
    DEP_IdDep
go




ALTER TABLE CONTIGUAS
    ADD FOREIGN KEY (ARE_Cont)
        REFERENCES AREAS
go
```

```
ALTER TABLE CONTIGUAS
    ADD FOREIGN KEY (ARE_Area)
        REFERENCES AREAS
go

exec sp_foreignkey CONTIGUAS, AREAS,
    ARE_Cont
go

exec sp_foreignkey CONTIGUAS, AREAS,
    ARE_Area
go

ALTER TABLE DEPNIV
    ADD FOREIGN KEY (DEP_IdDep)
        REFERENCES DEPARTAMENTOS
go

ALTER TABLE DEPNIV
    ADD FOREIGN KEY (NIV_IdNiv)
        REFERENCES NIVELES
go

exec sp_foreignkey DEPNIV, DEPARTAMENTOS,
    DEP_IdDep
go

exec sp_foreignkey DEPNIV, NIVELES,
    NIV_IdNiv
go

ALTER TABLE EMPLEADOS
    ADD FOREIGN KEY (ARE_IdArea)
        REFERENCES AREAS
go

exec sp_foreignkey EMPLEADOS, AREAS,
    ARE_IdArea
go

ALTER TABLE INSTALACION
    ADD FOREIGN KEY (DIS_IdDisp)
        REFERENCES DISPOSITIVOS
go

ALTER TABLE INSTALACION
    ADD FOREIGN KEY (ARE_IdArea)
```

```
            REFERENCES AREAS
go


 exec sp_foreignkey INSTALACION, DISPOSITIVOS,
     DIS_IdDisp
go

 exec sp_foreignkey INSTALACION, AREAS,
     ARE_IdArea
go


 create trigger tI_ACCESOS on ACCESOS for INSERT as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* INSERT trigger on ACCESOS */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* EMPLEADOS Un Empleado tiene  Accesos no autorizados ACCESOS ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(EMP_RFC)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,EMPLEADOS
        where
          /* %JoinFKPK(inserted,EMPLEADOS) */
          inserted.EMP_RFC = EMPLEADOS.EMP_RFC
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'No es posible INSERTAR "ACCESOS" porque "EMPLEADOS" no existe.'
      goto error
    end
  end

  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* AREAS Un Area tiene  Accesos no autorizados ACCESOS ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(ARE_IdArea)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,AREAS
```

```
      where
        /* %JoinFKPK(inserted,AREAS) */
        inserted.ARE_IdArea = AREAS.ARE_IdArea
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
           @errmsg = 'No es posible INSERTAR "ACCESOS" porque "AREAS" no existe.'
      goto error
    end
  end


  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

create trigger tU_ACCESOS on ACCESOS for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* UPDATE trigger on ACCESOS */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insEMP_RFC varchar(20),
           @insARE_IdArea char(3),
           @insACC_FecHoraEnt datetime,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* EMPLEADOS Un Empleado tiene  Accesos no autorizados ACCESOS ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(EMP_RFC)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,EMPLEADOS
        where
          /* %JoinFKPK(inserted,EMPLEADOS) */
          inserted.EMP_RFC = EMPLEADOS.EMP_RFC
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
           @errmsg = 'No es posible ACTUALIZAR "ACCESOS" porque "EMPLEADOS" no existe.'
      goto error
```

```
  end
 end

 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* AREAS Un Area tiene  Accesos no autorizados ACCESOS ON CHILD UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(ARE_IdArea)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,AREAS
     where
      /* %JoinFKPK(inserted,AREAS) */
      inserted.ARE_IdArea = AREAS.ARE_IdArea
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
        @errmsg = 'No es posible ACTUALIZAR "ACCESOS" porque "AREAS" no existe.'
   goto error
  end
 end


 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go

create trigger tD_AREAS on AREAS for DELETE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* DELETE trigger on AREAS */
begin
 declare  @errno  int,
       @errmsg  varchar(255)
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* AREAS Un Area tiene Dispositivos INSTALACION ON PARENT DELETE RESTRICT */
  if exists (
   select * from deleted,INSTALACION
   where
    /*  %JoinFKPK(INSTALACION,deleted," = "," and") */
    INSTALACION.ARE_IdArea = deleted.ARE_IdArea
  )
  begin
   select @errno  = 30001,
        @errmsg = 'No es posible BORRAR "AREAS" porque "INSTALACION" existe.'
   goto error
  end

  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
```

```
   /* AREAS Una Contigua tiene al lado unas Areas CONTIGUAS ON PARENT DELETE
RESTRICT */
   if exists (
    select * from deleted,CONTIGUAS
    where
     /*  %JoinFKPK(CONTIGUAS,deleted," = "," and") */
     CONTIGUAS.ARE_Cont = deleted.ARE_IdArea
   )
   begin
    select @errno  = 30001,
        @errmsg = 'No es posible BORRAR "AREAS" porque "CONTIGUAS" existe.'
    goto error
   end

   /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
   /* AREAS Un Area tiene al lado unas Contiguas CONTIGUAS ON PARENT DELETE
RESTRICT */
   if exists (
    select * from deleted,CONTIGUAS
    where
     /*  %JoinFKPK(CONTIGUAS,deleted," = "," and") */
     CONTIGUAS.ARE_Area = deleted.ARE_IdArea
   )
   begin
    select @errno  = 30001,
        @errmsg = 'No es posible BORRAR "AREAS" porque "CONTIGUAS" existe.'
    goto error
   end

   /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
   /* AREAS Un Area tiene  Accesos no autorizados ACCESOS ON PARENT DELETE RESTRICT
*/
   if exists (
    select * from deleted,ACCESOS
    where
     /*  %JoinFKPK(ACCESOS,deleted," = "," and") */
     ACCESOS.ARE_IdArea = deleted.ARE_IdArea
   )
   begin
    select @errno  = 30001,
        @errmsg = 'No es posible BORRAR "AREAS" porque "ACCESOS" existe.'
    goto error
   end

   /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
   /* AREAS En Area trabajan Empleados EMPLEADOS ON PARENT DELETE RESTRICT */
   if exists (
    select * from deleted,EMPLEADOS
    where
     /*  %JoinFKPK(EMPLEADOS,deleted," = "," and") */
     EMPLEADOS.ARE_IdArea = deleted.ARE_IdArea
   )
   begin
    select @errno  = 30001,
        @errmsg = 'No es posible BORRAR "AREAS" porque "EMPLEADOS" existe.'
    goto error
```

```
    end


    /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

create trigger tI_AREAS on AREAS for INSERT as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* INSERT trigger on AREAS */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* DEPARTAMENTOS Un Departamento tiene asignadas Areas AREAS ON CHILD INSERT
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(DEP_IdDep)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,DEPARTAMENTOS
        where
          /* %JoinFKPK(inserted,DEPARTAMENTOS) */
          inserted.DEP_IdDep = DEPARTAMENTOS.DEP_IdDep
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'No es posible INSERTAR "AREAS" porque "DEPARTAMENTOS" no existe.'
      goto error
    end
  end


  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

create trigger tU_AREAS on AREAS for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
```

```
/* UPDATE trigger on AREAS */
begin
 declare @numrows int,
        @nulicnt int,
        @validcnt int,
        @insARE_IdArea char(3),
        @ermo  int,
        @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* AREAS Un Area tiene Dispositivos INSTALACION ON PARENT UPDATE RESTRICT */
 if
  /* %ParentPK(" or",update) */
  update(ARE_IdArea)
 begin
  if exists (
   select * from deleted,INSTALACION
   where
    /*  %JoinFKPK(INSTALACION,deleted," = "," and") */
    INSTALACION.ARE_IdArea = deleted.ARE_IdArea
  )
  begin
   select @ermo  = 30005,
        @errmsg = 'No es posible ACTUALIZAR "AREAS" porque "INSTALACION" existe.'
   goto error
  end
 end

 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* AREAS Una Contigua tiene al lado unas Areas CONTIGUAS ON PARENT UPDATE
RESTRICT */
 if
  /* %ParentPK(" or",update) */
  update(ARE_IdArea)
 begin
  if exists (
   select * from deleted,CONTIGUAS
   where
    /*  %JoinFKPK(CONTIGUAS,deleted," = "," and") */
    CONTIGUAS.ARE_Cont = deleted.ARE_IdArea
  )
  begin
   select @ermo  = 30005,
        @errmsg = 'No es posible ACTUALIZAR "AREAS" porque "CONTIGUAS" existe.'
   goto error
  end
 end

 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* AREAS Un Area tiene al lado unas Contiguas CONTIGUAS ON PARENT UPDATE RESTRICT
*/
 if
  /* %ParentPK(" or",update) */
  update(ARE_IdArea)
 begin
```

```
  if exists (
    select * from deleted,CONTIGUAS
    where
      /*  %JoinFKPK(CONTIGUAS,deleted," = "," and") */
      CONTIGUAS.ARE_Area = deleted.ARE_IdArea
  )
  begin
    select @errno  = 30005,
          @errmsg = 'No es posible ACTUALIZAR "AREAS" porque "CONTIGUAS" existe.'
    goto error
  end
end

/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* AREAS Un Area tiene  Accesos no autorizados ACCESOS ON PARENT UPDATE RESTRICT
*/
if
  /* %ParentPK(" or",update) */
  update(ARE_IdArea)
begin
  if exists (
    select * from deleted,ACCESOS
    where
      /*  %JoinFKPK(ACCESOS,deleted," = "," and") */
      ACCESOS.ARE_IdArea = deleted.ARE_IdArea
  )
  begin
    select @errno  = 30005,
          @errmsg = 'No es posible ACTUALIZAR "AREAS" porque "ACCESOS" existe.'
    goto error
  end
end



/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* AREAS En Area trabajan Empleados EMPLEADOS ON PARENT UPDATE RESTRICT */
if
  /* %ParentPK(" or",update) */
  update(ARE_IdArea)
begin
  if exists (
    select * from deleted,EMPLEADOS
    where
      /*  %JoinFKPK(EMPLEADOS,deleted," = "," and") */
      EMPLEADOS.ARE_IdArea = deleted.ARE_IdArea
  )
  begin
    select @errno  = 30005,
          @errmsg = 'No es posible ACTUALIZAR "AREAS" porque "EMPLEADOS" existe.'
    goto error
  end
end
```

```
 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* DEPARTAMENTOS Un Departamento tiene asignadas Areas AREAS ON CHILD UPDATE
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(DEP_IdDep)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,DEPARTAMENTOS
    where
     /* %JoinFKPK(inserted,DEPARTAMENTOS) */
     inserted.DEP_IdDep = DEPARTAMENTOS.DEP_IdDep
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
        @errmsg = 'No es posible ACTUALIZAR "AREAS" porque "DEPARTAMENTOS" no
existe.'
    goto error
  end
 end


 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
go



create trigger tI_CONTIGUAS on CONTIGUAS for INSERT as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* INSERT trigger on CONTIGUAS */
begin
 declare  @numrows int,
        @nullcnt int,
        @validcnt int,
        @errno   int,
        @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* AREAS Una Contigua tiene al lado unas Areas CONTIGUAS ON CHILD INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(ARE_Cont)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,AREAS
```

```
      where
        /* %JoinFKPK(inserted,AREAS) */
        inserted.ARE_Cont = AREAS.ARE_IdArea
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */
    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
           @errmsg = 'No es posible INSERTAR "CONTIGUAS" porque "AREAS" no existe.'
      goto error
    end
  end

  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* AREAS Un Area tiene al lado unas Contiguas CONTIGUAS ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(ARE_Area)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,AREAS
        where
          /* %JoinFKPK(inserted,AREAS) */
          inserted.ARE_Area = AREAS.ARE_IdArea
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
           @errmsg = 'No es posible INSERTAR "CONTIGUAS" porque "AREAS" no existe.'
      goto error
    end
  end

  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

create trigger tU_CONTIGUAS on CONTIGUAS for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* UPDATE trigger on CONTIGUAS */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insARE_Cont char(3),
           @insARE_Area char(3),
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
```

```
 /* AREAS Una Contigua tiene al lado unas Areas CONTIGUAS ON CHILD UPDATE RESTRICT
*/
 if
  /* %ChildFK(" or",update) */
  update(ARE_Cont)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,AREAS
    where
     /* %JoinFKPK(inserted,AREAS) */
     inserted.ARE_Cont = AREAS.ARE_IdArea
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'No es posible ACTUALIZAR "CONTIGUAS" porque "AREAS" no existe.'
   goto error
  end
 end

 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* AREAS Un Area tiene al lado unas Contiguas CONTIGUAS ON CHILD UPDATE RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(ARE_Area)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,AREAS
    where
     /* %JoinFKPK(inserted,AREAS) */
     inserted.ARE_Area = AREAS.ARE_IdArea
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */
  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30007,
       @errmsg = 'No es posible ACTUALIZAR "CONTIGUAS" porque "AREAS" no existe.'
   goto error
  end
 end

 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go

create trigger tD_DEPARTAMENTOS on DEPARTAMENTOS for DELETE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* DELETE trigger on DEPARTAMENTOS */
begin
 declare  @errno   int,
```

```
        @errmsg  varchar(255)
    /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
    /* DEPARTAMENTOS Un Departamento esta en Niveles DEPNIV ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,DEPNIV
      where
        /*  %JoinFKPK(DEPNIV,deleted," = "," and") */
        DEPNIV.DEP_IdDep = deleted.DEP_IdDep
    )
    begin
      select @errno  = 30001,
            @errmsg = 'No es posible BORRAR "DEPARTAMENTOS" porque "DEPNIV" existe.'
      goto error
    end

    /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
    /* DEPARTAMENTOS Un Departamento tiene asignadas Areas AREAS ON PARENT DELETE
RESTRICT */
    if exists (
      select * from deleted,AREAS
      where
        /*  %JoinFKPK(AREAS,deleted," = "," and") */
        AREAS.DEP_IdDep = deleted.DEP_IdDep
    )
    begin
      select @errno  = 30001,
            @errmsg = 'No es posible BORRAR "DEPARTAMENTOS" porque "AREAS" existe.'
      goto error
    end

    /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

create trigger tU_DEPARTAMENTOS on DEPARTAMENTOS for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* UPDATE trigger on DEPARTAMENTOS */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insDEP_IdDep smallint,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* DEPARTAMENTOS Un Departamento esta en Niveles DEPNIV ON PARENT UPDATE
RESTRICT */
  if
    /* %ParentPK(" or",update) */
```

```
    update(DEP_IdDep)
   begin
    if exists (
     select * from deleted,DEPNIV
     where
      /*  %JoinFKPK(DEPNIV,deleted," = "," and") */
      DEPNIV.DEP_IdDep = deleted.DEP_IdDep
    )
    begin
     select @errno  = 30005,
          @errmsg = 'No es posible ACTUALIZAR "DEPARTAMENTOS" porque "DEPNIV" existe.'
     goto error
    end
   end

  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* DEPARTAMENTOS Un Departamento tiene asignadas Areas AREAS ON PARENT UPDATE
 RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(DEP_IdDep)
   begin
    if exists (
     select * from deleted,AREAS
     where
      /*  %JoinFKPK(AREAS,deleted," = "," and") */
      AREAS.DEP_IdDep = deleted.DEP_IdDep
    )
    begin
     select @errno  = 30005,
          @errmsg = 'No es posible ACTUALIZAR "DEPARTAMENTOS" porque "AREAS" existe.'
     goto error
    end
   end


  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
 error:
    raiserror @errno @errmsg
    rollback transaction
 end
 go

 create trigger tI_DEPNIV on DEPNIV for INSERT as
 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* INSERT trigger on DEPNIV */
 begin
  declare  @numrows int,
         @nullcnt int,
         @validcnt int,
         @errno   int,
         @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
```

```
 /* DEPARTAMENTOS Un Departamento esta en Niveles DEPNIV ON CHILD INSERT
RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(DEP_IdDep)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,DEPARTAMENTOS
    where
     /* %JoinFKPK(inserted,DEPARTAMENTOS) */
     inserted.DEP_IdDep = DEPARTAMENTOS.DEP_IdDep
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
       @errmsg = 'No es posible INSERTAR "DEPNIV" porque "DEPARTAMENTOS" no existe.'
   goto error
  end
 end




 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* NIVELES Un Nivel tiene Departamentos DEPNIV ON CHILD INSERT RESTRICT */
 if
  /* %ChildFK(" or",update) */
  update(NIV_IdNiv)
 begin
  select @nullcnt = 0
  select @validcnt = count(*)
   from inserted,NIVELES
    where
     /* %JoinFKPK(inserted,NIVELES) */
     inserted.NIV_IdNiv = NIVELES.NIV_IdNiv
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
   select @errno  = 30002,
       @errmsg = 'No es posible INSERTAR "DEPNIV" porque "NIVELES" no existe.'
   goto error
  end
 end


 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 return
error:
  raiserror @errno @errmsg
  rollback transaction
end
```

```
go

create trigger tU_DEPNIV on DEPNIV for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* UPDATE trigger on DEPNIV */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insNIV_IdNiv tinyint,
           @insDEP_IdDep smallint,
           @errno  int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* DEPARTAMENTOS Un Departamento esta en Niveles DEPNIV ON CHILD UPDATE
RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(DEP_IdDep)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,DEPARTAMENTOS
        where
          /* %JoinFKPK(inserted,DEPARTAMENTOS) */
          inserted.DEP_IdDep = DEPARTAMENTOS.DEP_IdDep
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'No es posible ACTUALIZAR "DEPNIV" porque"DEPARTAMENTOS" no
existe.'
      goto error
    end
  end

  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* NIVELES Un Nivel tiene Departamentos DEPNIV ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(NIV_IdNiv)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,NIVELES
        where
          /* %JoinFKPK(inserted,NIVELES) */
          inserted.NIV_IdNiv = NIVELES.NIV_IdNiv
  /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

  if @validcnt + @nullcnt != @numrows
  begin
    select @errno  = 30007,
```

```
        @errmsg = 'No es posible ACTUALIZAR "DEPNIV" porque "NIVELES" no existe.'
    goto error
   end
  end


 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go

create trigger tD_DISPOSITIVOS on DISPOSITIVOS for DELETE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* DELETE trigger on DISPOSITIVOS */
begin
 declare  @errno   int,
          @errmsg  varchar(255)
   /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
   /* DISPOSITIVOS Un Dispositivo esta en Areas INSTALACION ON PARENT DELETE
RESTRICT */
   if exists (
    select * from deleted,INSTALACION
    where
     /*  %JoinFKPK(INSTALACION,deleted," = "," and") */
     INSTALACION.DIS_IdDisp = deleted.DIS_IdDisp
   )
   begin
    select @errno  = 30001,
          @errmsg = 'No es posible BORRAR "DISPOSITIVOS" porque "INSTALACION" existe.'
    goto error
   end


   /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
   return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go

create trigger tU_DISPOSITIVOS on DISPOSITIVOS for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* UPDATE trigger on DISPOSITIVOS */
begin
 declare  @numrows int,
          @nullcnt int,
          @validcnt int,
          @insDIS_IdDisp smallint,
          @errno   int,
          @errmsg  varchar(255)

 select @numrows = @@rowcount
```

```
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* DISPOSITIVOS Un Dispositivo esta en Areas INSTALACION ON PARENT UPDATE
 RESTRICT */
  if
   /* %ParentPK(" or",update) */
   update(DIS_IdDisp)
  begin
   if exists (
    select * from deleted,INSTALACION
    where
     /*  %JoinFKPK(INSTALACION,deleted," = "," and") */
     INSTALACION.DIS_IdDisp = deleted.DIS_IdDisp
   )
   begin
    select @errno  = 30005,
          @errmsg = 'No es posible ACTUALIZAR "DISPOSITIVOS" porque "INSTALACION"
 existe.'
    goto error
   end
  end



  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go

create trigger tD_EMPLEADOS on EMPLEADOS for DELETE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* DELETE trigger on EMPLEADOS */
begin
 declare  @errno  int,
         @errmsg  varchar(255)
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* EMPLEADOS Un Empleado tiene  Accesos no autorizados ACCESOS ON PARENT DELETE
RESTRICT */
  if exists (
   select * from deleted,ACCESOS
   where
    /*  %JoinFKPK(ACCESOS,deleted," = "," and") */
    ACCESOS.EMP_RFC = deleted.EMP_RFC
  )
   begin
    select @errno  = 30001,
          @errmsg = 'No es posible BORRAR "EMPLEADOS" porque "ACCESOS" existe.'
   goto error
  end


  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
```

```
   raiserror @errno @errmsg
   rollback transaction
end
go

create trigger tI_EMPLEADOS on EMPLEADOS for INSERT as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* INSERT trigger on EMPLEADOS */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* AREAS En Area trabajan Empleados EMPLEADOS ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(ARE_IdArea)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,AREAS
        where
          /* %JoinFKPK(inserted,AREAS) */
          inserted.ARE_IdArea = AREAS.ARE_IdArea
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */
    select @nullcnt = count(*) from inserted where
      inserted.ARE_IdArea is null
    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
             @errmsg = 'No es posible INSERTAR "EMPLEADOS" porque "AREAS" no existe.'
      goto error
    end
  end


  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go

create trigger tU_EMPLEADOS on EMPLEADOS for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* UPDATE trigger on EMPLEADOS */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insEMP_RFC varchar(20),
```

```
        @errno  int,
        @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* EMPLEADOS Un Empleado tiene  Accesos no autorizados ACCESOS ON PARENT UPDATE
RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(EMP_RFC)
  begin
    if exists (
      select * from deleted,ACCESOS
      where
        /*  %JoinFKPK(ACCESOS,deleted," = "," and") */
        ACCESOS.EMP_RFC = deleted.EMP_RFC
    )
    begin
      select @errno  = 30005,
            @errmsg = 'No es posible ACTUALIZAR "EMPLEADOS" porque "ACCESOS" existe.'
      goto error
    end
  end

  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* AREAS En Area trabajan Empleados EMPLEADOS ON CHILD UPDATE RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(ARE_IdArea)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,AREAS
        where
          /* %JoinFKPK(inserted,AREAS) */
          inserted.ARE_IdArea = AREAS.ARE_IdArea
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */
    select @nullcnt = count(*) from inserted where
      inserted.ARE_IdArea is null
    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'No es posible ACTUALIZAR "EMPLEADOS" porque "AREAS" no existe.'
      goto error
    end
  end


  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end
go
```

```
create trigger tI_INSTALACION on INSTALACION for INSERT as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* INSERT trigger on INSTALACION */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* DISPOSITIVOS Un Dispositivo esta en Areas INSTALACION ON CHILD INSERT RESTRICT
*/
  if
    /* %ChildFK(" or",update) */
    update(DIS_IdDisp)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,DISPOSITIVOS
        where
          /* %JoinFKPK(inserted,DISPOSITIVOS) */
          inserted.DIS_IdDisp = DISPOSITIVOS.DIS_IdDisp
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
           @errmsg = 'No es posible INSERTAR "INSTALACION" porque "DISPOSITIVOS" no
existe.'
      goto error
    end
  end

  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* AREAS Un Area tiene Dispositivos INSTALACION ON CHILD INSERT RESTRICT */
  if
    /* %ChildFK(" or",update) */
    update(ARE_IdArea)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,AREAS
        where
          /* %JoinFKPK(inserted,AREAS) */
          inserted.ARE_IdArea = AREAS.ARE_IdArea
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30002,
           @errmsg = 'No es posible INSERTAR "INSTALACION" porque "AREAS" no existe.'
      goto error
    end
  end
```

```
 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 return
error:
   raiserror @errno @errmsg
   rollback transaction
end
go



create trigger tU_INSTALACION on INSTALACION for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* UPDATE trigger on INSTALACION */
begin
 declare  @numrows int,
          @nullcnt int,
          @validcnt int,
          @insDIS_IdDisp smallint,
          @insARE_IdArea char(3),
          @errno   int,
          @errmsg  varchar(255)

 select @numrows = @@rowcount
 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* DISPOSITIVOS Un Dispositivo esta en Areas INSTALACION ON CHILD UPDATE RESTRICT
*/
 if
   /* %ChildFK(" or",update) */
   update(DIS_IdDisp)
 begin
   select @nullcnt = 0
   select @validcnt = count(*)
     from inserted,DISPOSITIVOS
       where
         /* %JoinFKPK(inserted,DISPOSITIVOS) */
         inserted.DIS_IdDisp = DISPOSITIVOS.DIS_IdDisp
   /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

   if @validcnt + @nullcnt != @numrows
   begin
     select @errno  = 30007,
          @errmsg = 'No es posible ACTUALIZAR"INSTALACION"porque"DISPOSITIVOS"no
existe.'
     goto error
   end
 end

 /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
 /* AREAS Un Area tiene Dispositivos INSTALACION ON CHILD UPDATE RESTRICT */
```

```
  if
    /* %ChildFK(" or",update) */
    update(ARE_IdArea)
  begin
    select @nullcnt = 0
    select @validcnt = count(*)
      from inserted,AREAS
        where
          /* %JoinFKPK(inserted,AREAS) */
          inserted.ARE_IdArea = AREAS.ARE_IdArea
    /* %NotnullFK(inserted," is null","select @nullcnt = count(*) from inserted where"," and") */

    if @validcnt + @nullcnt != @numrows
    begin
      select @errno  = 30007,
            @errmsg = 'No es posible ACTUALIZAR "INSTALACION" porque "AREAS" no existe.'
      goto error
    end
  end


  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end
go

create trigger tD_NIVELES on NIVELES for DELETE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* DELETE trigger on NIVELES */
begin
  declare  @errno   int,
           @errmsg  varchar(255)
    /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
    /* NIVELES Un Nivel tiene Departamentos DEPNIV ON PARENT DELETE RESTRICT */
    if exists (
      select * from deleted,DEPNIV
      where
        /*  %JoinFKPK(DEPNIV,deleted," = "," and") */
        DEPNIV.NIV_IdNiv = deleted.NIV_IdNiv
    )
    begin
      select @errno  = 30001,
            @errmsg = 'No es posible BORRAR "NIVELES" porque "DEPNIV" existe.'
      goto error
    end


    /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
    return
error:
    raiserror @errno @errmsg
    rollback transaction
end
```

```
go

create trigger tU_NIVELES on NIVELES for UPDATE as
/* ERwin Builtin Sun Jun 21 09:47:58 1998 */
/* UPDATE trigger on NIVELES */
begin
  declare  @numrows int,
           @nullcnt int,
           @validcnt int,
           @insNIV_IdNiv tinyint,
           @errno   int,
           @errmsg  varchar(255)

  select @numrows = @@rowcount
  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  /* NIVELES Un Nivel tiene Departamentos DEPNIV ON PARENT UPDATE RESTRICT */
  if
    /* %ParentPK(" or",update) */
    update(NIV_IdNiv)
  begin
    if exists (
      select * from deleted,DEPNIV
      where
        /*  %JoinFKPK(DEPNIV,deleted," = "," and") */
        DEPNIV.NIV_IdNiv = deleted.NIV_IdNiv
    )
    begin
      select @errno  = 30005,
             @errmsg = 'No es posible ACTUALIZAR "NIVELES" porque "DEPNIV" existe.'
      goto error
    end
  end


  /* ERwin Builtin Sun Jun 21 09:47:58 1998 */
  return
error:
    raiserror @errno @errmsg
    rollback transaction
end
go
```

Apéndice B. Código del Agente Inteligente y Bibliotecas.

Se presentan a continuación los código fuente del Agente Inteligente y sus bibliotecas. La documentación de los mismos, junto con el diagrama de clases permiten lograr una visión integral de su diseño e implementación. No hay ningún texto introductorio a los códigos ya que estos se encuentran profusamente documentados y no se ha considerado necesario entrar en una inútil duplicidad de explicaciones sobre su funcionamiento.

4.3.6.1. Agente.cpp

```
/*-------------------------------------------------------------------*/
/*                          Agente.cpp                               */
/*-------------------------------------------------------------------*/
/*
  Autor:
        Jauregui Espinosa Hugo,                (11/Junio/98)
  Compilador:
        Borland C++ Version 4.5
  Descripcion:
        Obtiene los datos que deben ser proporcionados por los sensores del edificio
        sobre las areas en las que se encuentran personas, las salidas y el
        lugar en que se origina el incendio.
  Librerias:
        analiz.h
  Indice:
        Ninguna
  Funciones dependientes del sistema operativo:
        Ninguna
  Definiciones:
        Ninguna
  Pendientes:
        Conexion con los sensores.
*/

# include <clAnaliz.h>

void main(void){
 char acDesde[20], acHacia[20];

 printf("\nCargando la Base de Conocimientos.");

 clAnaliz oAnaliz;


 printf("\nBase de Conocimientos Cargada.\n\n");
 printf("\n     Agente Inteligente para la Generacion de Rutas");
```

```
printf("\n        de Evacuacion en Caso de Incendio en un");
printf("\n              Edificio Inteligente\n\n\n");

printf("Area donde se detecto presencia humana? ");
gets(acDesde);
printf("Salida de Emergencia? ");
gets(acHacia);
oAnaliz.vObtenerInc();

oAnaliz.vVerificarArea(acDesde,acHacia);

printf("\n\nLa Ruta de Evacuacion es:\n");

oAnaliz.vDeterminarRuta(acHacia,0);

} /*** Agente.cpp ***/
```

4.3.6.2. *clArea.h*

```
/*-----------------------------------------------------------------*/
/*                          clArea.h                               */
/*-----------------------------------------------------------------*/
/*
 Autor:
        Jauregui Espinosa Hugo,                 (15/Junio/98)
 Compilador:
        GNU gcc  Version 2.7.2.1
 Descripcion:
        Descripcion de la estructura y rutinas de manejo de las areas. Se
        definene rutinas de acceso a los datos miembros de la clase.
 Librerias:
        Ninguna
 Indice:
        apObtenerDesde : Regresa el dato miembro acDesde
        apObtenerHacia : Regresa el dato miembro acHacia
        rObtenerDist : Regresa una referencia al dato miembro iDist
        rObtenerVis : Regresa una referencia al dato miembro cVis
 Funciones dependientes del sistema operativo:
        Ninguna
 Definiciones:
        _clArea    : Identifica la clase encargada del manejo de las areas
 Pendientes:
        Ninguno
*/

#if !defined (_clArea)
#define _clArea

/*-----------------------------------------------------------------*/
/*              Clase para el manejo de las areas                  */
/*-----------------------------------------------------------------*/
class clArea {

 char acDesde[20];
 char acHacia[20];
 int iDist;
 char cVis;

 public:

/*-----------------------------------------------------------------*/
/*              Regresa el dato miembro acDesde                    */
/*-----------------------------------------------------------------*/
 char * apObtenerDesde (void)
/*
 Autor:
        Jauregui Espinosa Hugo,                 (15/Junio/98)
 Descripcion:
        Regresa el dato miembro acDesde, el cual, siendo un arreglo, tambien puede

        ser manejado como un apuntador al inicio de si mismo
 Librerias:
```

```
        Ninguna
  Parametros:
        Ninguno
  Retoma:
        char * que representa un apuntador al inicio del arreglo acDesde
  Ejemplo:
        //Asigna al dato miembro acDesde lo que se introduzca desde la entrada
        //estandar
        gets(apObtenerDesde());
*/
  {
   return acDesde;
  } /*** apObtenerDesde () ***/


/*-------------------------------------------------------------*/
/*              Regresa el dato miembro acHacia                */
/*-------------------------------------------------------------*/
  char * apObtenerHacia (void)
/*
  Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
  Descripcion:
        Regresa el dato miembro acHacia, el cual, siendo un arreglo, tambien puede
        ser manejado como un apuntador al inicio del arreglo
  Librerias:
        Ninguna
  Parametros:
        Ninguno
  Retoma:
        char * que representa un apuntador al inicio del arreglo acHacia
  Ejemplo:
        //Asigna al dato miembro acHacia lo que se introduzca desde la entrada
        //estandar
        gets(apObtenerHacia());
*/
  {
   return acHacia;
  } /*** apObtenerHacia () ***/


/*-------------------------------------------------------------*/
/*          Regresa una referencia al dato miembro iDist       */
/*-------------------------------------------------------------*/
  int & rObtenerDist (void)
/*
  Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
  Descripcion:

        Regresa el dato miembro iDist como una referencia
  Librerias:
        Ninguna
  Parametros:
```

```
        Ninguno
  Retoma:
        int & que representa una referencia al dato miembro iDist
  Ejemplo:
        //Asigna al dato miembro iDist un valor de 10
        rObtenerDist()=10;
*/
  {
   return iDist;
  } /*** rObtenerDist () ***/


/*-------------------------------------------------------------------*/
/*            Regresa una referencia al dato miembro cVis    */
/*-------------------------------------------------------------------*/
  char & rObtenerVis (void)
/*
  Autor:
        Jauregui Espinosa Hugo,                (15/Junio/98)
  Descripcion:
        Regresa el dato miembro cVis como una referencia
  Librerias:
        Ninguna
  Parametros:
        Ninguno
  Retoma:
        char & que representa una referencia al dato miembro cVis
  Ejemplo:
        //Asigna al dato miembro cVis un valor de 'F'
        rObtenerVis()='F';
*/
  {
   return cVis;
  } /*** rObtenerVis () ***/

}; /*** clArea ***/

#endif  /*** #if () ***/
```

4.3.6.3. clNiv.h

```
/*-------------------------------------------------------------*/
/*                        clNivel.h                            */
/*-------------------------------------------------------------*/
/*
 Autor:
        Jauregui Espinosa Hugo,                 (15/Junio/98)
 Compilador:
        GNU gcc  Version 2.7.2.1
 Descripcion:
        Descripcion de la estructura y rutinas de manejo de las areas que
        conforman un nivel o piso dentro del Edificio Inteligente. Esta clase
        asimismo, es la encargada de realizar la conexion con el servidor de Sybase
        y obtener los datos necesarios para llenar la Base de Conocimiento necesaria
        para generar la ruta de evacuacion en caso de incendio.
 Librerias:
        clArea.h
        stdio.h
        string.h
        sybdb.h
        syberror.h
        sybfront.h
 Indice:
        clNiv : Inicializa las variables de la clase
        iObtenerUltPos : Regresa el valor de la variable iUltPos
        rObtenerArea : Obtiene una referencia a un determinado objeto de aoArea
        vDeclararArea : Declara un area dentro del dato miembro aoArea
        vCargarBaseCon : Carga la base de conocimientos desde Sybase
        iVerificarUnion : Verifica si existe una salida entre dos areas determinadas
Funciones dependientes del sistema operativo:
        Ninguna
 Definiciones:
        _clNiv   : Identifica la clase encargada del manejo de los niveles o pisos
        _Falso   : Representa el valor 0
        _Verdadero : Representa el valor 1
 Pendientes:
        Ninguno
*/

#if !defined (_clNiv)
#define _clNiv

# include <clArea.h>
# include <stdio.h>
# include <string.h>
# include <sybfront.h>
# include <sybdb.h>
# include <syberror.h>



# define Falso 0
# define Verdadero 1
```

```
/*-------------------------------------------------------------------*/
/*  Clase para el manejo de las areas en un Nivel o Piso  */
/*-------------------------------------------------------------------*/
class clNiv
{
 struct clArea aoArea[MAX];

 int iUltPos;

 public:

/*-------------------------------------------------------------------*/
/*              Inicializa las variables de la clase                 */
/*-------------------------------------------------------------------*/
  clNiv (void)
/*
  Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
  Descripcion:
        Inicializa los datos miembro para evitar que contengan basura.
  Librerias:
        Ninguna
  Parametros:
        Ninguno
  Retorna:
        Nada
  Ejemplo:
        //Construye el nivel oNiv
        clNiv oNiv;
*/
  {
   iUltPos=0;
  }


/*-------------------------------------------------------------------*/
/*          Regresa el valor de la variable iUltPos                  */
/*-------------------------------------------------------------------*/
  int iObtenerUltPos(void)
/*
  Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
  Descripcion:
        Regresa el valor de la variable iUtlPos, que representa la ultima posicion.
  Librerias:
        Ninguna
  Parametros:
        Ninguno

  Retorna:
        int que representa el valor de iUltPos
  Ejemplo:
        //Se asigna a la variable iNuevaPos el valor de la variable iUltPos
```

```
        int iNuevaPos = iObtenerUtlPos();
*/
 {
  return iUltPos;
 } /*** iObtenerUtlPos () ***/


/*-------------------------------------------------------------------*/
/*Obtiene una referencia a un determinado objeto de aoArea*/
/*-------------------------------------------------------------------*/
 clArea & rObtenerArea(int iInd)
/*
 Autor:
        Jauregui Espinosa Hugo,                 (15/Junio/98)
 Descripcion:
        Regresa una referencia a un elemento determinado por el parametro iInd del
        arreglo de objetos aoArea.
 Librerias:
        Ninguna
 Parametros:
        iInd  : Representa el indice que sera utilizado para determinar el elemento
                        que se debe regresar como referencia del arreglo aoArea
 Retoma:
        clArea & que es una referencia a un objeto de clase clArea
 Ejemplo:
        //Se asigna una distancia de 10 al objeto de clase clArea que esta como
        //tercer elemento dentro del arreglo aoArea
        rObtenerArea(3).rObtenerDist()=10;
*/
 {
  return aoArea[iInd];
 } /*** rObtenerArea () ***/


/*-------------------------------------------------------------------*/
/*   Declara un area dentro del dato miembro aoHabit        */
/*-------------------------------------------------------------------*/
 void vDeclararArea(char *apDesde,char *apHacia,int iDist)
/*
 Autor:
        Jauregui Espinosa Hugo,                 (15/Junio/98)
 Descripcion:
        Si la ultima posicion es menor al maximo de posiciones definido, se
        introducen los datos apDesde, apHacia e iDist que son proporcionados como
        parametros al nuevo objeto dentro del arreglo. Se determina como Falso la
        variable para definir si ha sido visitado dicho objeto y se incrementa en
        uno el valor de la variable iUtlPos. Si la ultima posicion es mayor o igual

        al maximo de posiciones definido para el arreglo, entonces se manda un
        mensaje de error indicando que la Base de conocimientos esta llena.
 Librerias:
        clArea.h
 Parametros:
        apDesde : Representa el area de origen
```

```
        apHacia : Representa el area de destino
        iDist   : Representa la distancia que existe desde el area origen
                                hasta el area destino
 Retoma:
        Nada
 Ejemplo:
        //Declara que el area 001 tiene una salida hacia el area 008 y
        //que existe una distancia entre ambas de 39
        vDeclararArea("001","008",39);
*/
 {
  if (iUltPos<MAX){
   strcpy(aoArea[iUltPos].apObtenerDesde(), apDesde);
   strcpy(aoArea[iUltPos].apObtenerHacia(), apHacia);
   aoArea[iUltPos].rObtenerDist()=iDist;
   aoArea[iUltPos].rObtenerVis()=False;
   iUltPos++;
  }
  else printf("Base de conocimientos llena.\n");
 } /*** vDeclararArea () ***/



/*-------------------------------------------------------------------*/
/*          Carga la base de conocimientos desde Sybase              */
/*-------------------------------------------------------------------*/
 void vCargarBaseCon(void)
/*
 Autor:
        Jauregui Espinosa Hugo,                 (15/Junio/98)
 Descripcion:
        Se declaran las variables que serviran para contener los resultados del query
        al servidor SQL Server, los tipos de datos de las tres primeras variables
        corresponden con los que se encuentran en la tabla CONTIGUAS. Esta table
        contiene la informacion con la cual se puede formar el grafo sobre el que se
        trabajara. La variable ret_code se utiliza para recibir el codigo de
        resultado que regresa la funcion dbresults al intentar ejecutar un comando,
        puede regresar SUCCEED, FAIL o NO_MORE_RESULTS. Asimismo se declaran los
        apuntadores login y dbproc, el primero es utilizado para que por medio de la
        funcion dblogin se aloje en memoria una estructura LOGINREC, por medio de la
        cual se puede definir el nombre del usuario de la base de datos y su
        respectivo password, lo cual servira posteriormente para que sea usada por la
        funcion dbopen; por su parte, la variable dbproc sirve para alojar la
        direccion de una estructura DBPROCESS que ya ha sido alojada en memoria
        e inicializada. Luego de la declaracion de variables, se aloja en memoria la
        variable login por medio de dblogin() y se establece en la misma variable el
        nombre del usuario, que en este caso es "rutas" y su password "jauss98".

        Posteriormente, se aloja en memoria y se inicializa la variable dbproc,
        usando para ello a la variable login, que previamente fue alojada en memoria
        e inicializada, y el nombre del SQL Server, que en este caso es SYBASE_10.
        Luego se verifica si dbproc es igual a NULL, lo cual indicaria que la
        estructura DBPROCESS no pudo ser creada o inicializada o que fallo el login
        para el SQL Server, de ser asi, se manda un mensaje de error. De lo
        contrario, se agrega un comando SQL para determinar la base de datos a
        utilizar al buffer de comandos de la estructura DBPROCESS, que es dbproc.
```

Despues se envian los comandos que estan en dbproc al SQL Server, por medio de la funcion dbsqlexec(), y se verifica si ha regresado FAIL, lo cual indicaria un error de sintaxis o de semantica en la sentencia SQL, de ser asi, se invoca a la funcion dbexit(), la cual cierra y borra de memoria la estructura DBPROCESS y luego se manda un mensaje que indica que hubo un error al usar la base de datos. En cambio, si se ejecuta normalmente el comando SQL, se agrega otro comando SQL con la consulta a la table CONTIGUAS al buffer de comandos de la estructura DBPROCESS, dbproc. Luego se envia el comando que esta en dbproc al SQL Server, por medio de la funcion dbsqlexec(), y se verifica si ha regresado FAIL, lo cual indicaria un error de sintaxis o de semantica en la consulta, de ser asi, se invoca a la funcion dbexit y luego se manda un mensaje de error que indica que hubo un error al hacer el select. Si se ejecuta correctamente el comando SQL, mientras que el resultado de dbresults(), que prepara para procesar los resultados del query, y que se iguala con ret_code es diferente a NO_MORE_RESULTS, es decir, es igual a SUCCEED, se verifica si el resultado de dbresults es igual a FAIL, y de ser asi se manda un mensaje de error y se rompe el ciclo while; de no ser igual a FAIL, se verifica si el numero de columnas del resultado es diferente de 3, que es el numero de columnas que se deben obtener, en cuyo caso se manda un mensaje de error de consistencia y se rompe el ciclo while; si no hay error de consistencia, se ligan las columnas del resultado con la respectiva variable del programa y mientras que ya no haya mas renglones, es decir tuplas, se invoca a la funcion vDeclararArea() pasandole como parametros las variables que fueron ligadas avanzando renglon por renglon por medio de la funcion dbnextrow(). Finalmente se invoca a dbexit(), la cual cierra y borra de memoria la estructura DBPROCESS.

Librerias:
sybdb.h
syberror.h
sybfront.h

Parametros:
Ninguno

Retorna:
Nada

Ejemplo:
//Carga la base de conocimientos desde el servidor de Sybase
vCargarBaseCon();

```
*/
 {
  DBCHAR      DBOrigen[3];
  DBCHAR      DBDestino[3];
  DBSMALLINT  DBDistancia;

  RETCODE ret_code;


  {
  LOGINREC  *login;
  DBPROCESS  *dbproc;
  login=dblogin();
  DBSETLUSER (login,"rutas");
  DBSETLPWD (login, "jauss98");
  dbproc = dbopen (login, "SYBASE_10");

  if(dbproc==NULL)
  {
```

```
    printf("Existe un error al abrir.\n");
   }

   dbcmd (dbproc, "use rutas");
   if(dbsqlexec (dbproc) == FAIL)
   {
    dbexit();
    printf("Hubo un error al usar la base de datos rutas.");
   }
   while((ret_code=dbresults (dbproc))!=NO_MORE_RESULTS)
   {
    if(ret_code==FAIL)
    {
     break;
     printf("No hay datos para cargar la base de conocimientos.");
    }
   }

  dbcmd (dbproc, "select ARE_Area, ARE_Cont, CON_Dist from CONTIGUAS");
  if(dbsqlexec (dbproc) == FAIL)
  {
   printf("Error al intentar hacer el select para cargar las areas.");
   dbexit();
  }
  while((ret_code=dbresults (dbproc)) != NO_MORE_RESULTS)
  {
   if(ret_code==FAIL)
   {
    printf("Error en resultados de select.");
    break;
   }

   if(dbnumcols(dbproc)!=3)
   {
    printf("Error de consistencia al cargar la base de conocimientos.");
    break;
   }
   dbbind (dbproc, 1, NTBSTRINGBIND, 0, DBOrigen);
   dbbind (dbproc, 2, NTBSTRINGBIND, 0, DBDestino);
   dbbind (dbproc, 3, SMALLBIND, 0, (BYTE *) &DBDistancia);

   while (dbnextrow(dbproc)!=NO_MORE_ROWS)
   {

    vDeclararArea(DBOrigen,DBDestino,DBDistancia);
    printf ("%s %s %d\n", DBOrigen,DBDestino,DBDistancia);
   }
   dbexit ();
  }
 }

} /*** vCargarBaseCon () ***/


/*-------------------------------------------------------------------*/
```

```
/*Verifica si existe una salida entre dos areas determinadas*/
/*-------------------------------------------------------------*/
 int iVerificarUnion(char *apDesde,char *apHacia)
/*
 Autor:
        Jauregui Espinosa Hugo,                (15/Junio/98)
 Descripcion:
        Se declara una variable que sirve como indice en el ciclo que recorre todo
        el arreglo aoArea. Cada vez que se avanza en el arreglo se verifica si los
        datos miembro apDesde y apHacia, que representan una conexion entre dos
        areas, del objeto de clase clArea que se encuentran como miembros
        del arreglo aoArea concuerdan ambos con los parametros apDesde y apHacia de
        esta funcion. En caso de que concuerden, se regresa una referencia a la
        distancia que existe entre ambas areas, de lo contrario se regresa
 Falso, es decir un valor de 0.
 Librerias:
        string.h
 Parametros:
        apDesde : Representa el area de origen
        apHacia : Representa el area de destino
 Retorna:
        int que representa la afirmacion o negacion de la union entre las dos
        areas
 Ejemplo:
        //Verifica si el area 018 y 045 estan unidas, en cuyo caso se muestra
        //en la salida estandar el mensaje "Estan unidas"
        if(iVerificarUnion("018","045")) printf("Estan unidas");
*/
 {
  register int riInd;

  for (riInd=iUltPos-1;riInd>-1;riInd--)
   if(!strcmp(aoArea[riInd].apObtenerDesde(), apDesde) &&
!strcmp(aoArea[riInd].apObtenerHacia(), apHacia))
    return aoArea[riInd].rObtenerDist();
  return Falso;
 } /*** iVerificarUnion () ***/

}; /*** clNiv ***/

#endif  /*** #if () ***/
```

4.3.6.4. clNodo.h

```
/*-------------------------------------------------------------*/
/*                        clNodo.h                             */
/*-------------------------------------------------------------*/
/*
 Autor:
        Hugo Jauregui Espinosa, (24/Marzo/98).
 Compilador:
        GNU gcc  Version 2.7.2.1
 Descripcion:
        Descripcion de la estructura y rutinas de manejo de nodos utilizados por la
        pila secuencial. Se definene rutinas de acceso y construccion de este
        tipo de nodos. Igualmente se define que la clase tiene como datos miembros
        dos apuntadores que determinan la orientacion del acceso a las habitaciones y
        un entero que define la distancia que existe entre una habitacion y otra.
 Librerias:
        Ninguna
 Indice:
        clNodo :  Inicializa las variables de la clase
        apObtenerDesdeNodo : Obtiene un apuntador al dato miembro acDesde
        apObtenerHaciaNodo : Obtiene un apuntador al dato miembro acHacia
        rObtenerDistNodo : Obtiene una referencia al dato miembro iDist
 Funciones dependientes del sistema operativo:
        Ninguna
 Definiciones:
        _clNodo    : Identifica la clase encargada del manejo de los nodos
 Pendientes:
        Ninguno
*/

#if !defined (_clNodo)
#define _clNodo

/*-------------------------------------------------------------*/
/*            Clase para el manejo de nodos                    */
/*-------------------------------------------------------------*/
class clNodo
{
 char acDesde[20];
 char acHacia[20];
 int iDist;

public:

/*-------------------------------------------------------------*/
/*            Inicializa las variables de la clase             */
/*-------------------------------------------------------------*/
 clNodo (void)
/*
 Autor:
        Hugo Jauregui Espinosa, (24/Marzo/98).

 Descripcion:
        Inicializa los datos miembro para evitar que contengan basura.
```

```
  Librerias:
        Ninguna
  Parametros:
        Ninguno
  Retorna:
        Nada
  Ejemplo:
        //Construye el nodo oNodo
        clNodo oNodo;
*/
  {
   iDist = 0;
  }

/*-------------------------------------------------------------*/
/*          Obtiene un apuntador al dato miembro acDesde       */
/*-------------------------------------------------------------*/
  char * apObtenerDesdeNodo(void)
/*
  Autor:
        Hugo Jauregui Espinosa, (24/Marzo/98).
  Descripcion:
        Regresa el valor de acDesde, que es un arreglo a caracter.
  Librerias:
        Ninguna
  Parametros:
        Ninguno
  Retorna:
        char * que es la direccion del primer elemento del arreglo.
  Ejemplo:
        //Copia el contenido del dato miembro acDesde del objeto oNodo hacia el
        //apuntador apStr que tuvo que ser definido anteriormente
        strcpy(apStr,oNodo.apObtenerDesdeNodo());
*/
  {
   return acDesde;
  } /*** apObtenerDesdeNodo() ***/


/*-------------------------------------------------------------*/
/*          Obtiene un apuntador al dato miembro acHacia       */
/*-------------------------------------------------------------*/
  char * apObtenerHaciaNodo(void)
/*
  Autor:
        Hugo Jauregui Espinosa, (24/Marzo/98).
  Descripcion:
        Regresa el valor del dato miembro acHacia, que es un arreglo a caracter.
  Librerias:
        Ninguna

  Parametros:
        Ninguno
  Retorna:
        char * que es la direccion del primer elemento del arreglo.
```

```
  Ejemplo:
        //Copia el contenido del dato miembro acHacia del objeto oNodo hacia el
        //apuntador apStr que tuvo que ser definido anteriormente
        strcpy(apStr,oNodo.apObtenerHaciaNodo());
*/
  {
   return acHacia;
  } /*** apObtenerHaciaNodo() ***/


/*-------------------------------------------------------------------*/
/*          Obtiene una referencia al dato miembro iDist             */
/*-------------------------------------------------------------------*/
  int & rObtenerDistNodo(void)
/*
  Autor:
        Hugo Jauregui Espinosa, (24/Marzo/98).
  Descripcion:
        Regresa el valor del dato miembro iDist.
  Librerias:
        Ninguna
  Parametros:
        Ninguno
  Retorna:
        int & que es una referencia al dato miembro iDist
  Ejemplo:
        //Asigna al dato miembro iDist del objeto oNodo la suma de si mismo con el
        //valor del dato miembro iDist del objeto oNodo1
        oNodo.rObtenerDistNodo() += oNodo1.rObtenerDistNodo();
*/
  {
   return iDist;
  } /*** rObtenerDistNodo() ***/

}; /*** clNodo ***/

#endif  /*** #if () ***/
```

4.3.6.5. clPila.h

```
/*-------------------------------------------------------------*/
/*                        clPila.h                             */
/*-------------------------------------------------------------*/
/*
 Autor:
        Jauregui Espinosa Hugo,                (15/Junio/98)
 Compilador:
        GNU gcc  Version 2.7.2.1
 Descripcion:
        Descripcion de la estructura y rutinas de manejo de la pila de retroceso. Se
        definen funciones de acceso a los nodos de la pila, asi como funciones para
        insertar y eliminar nodos. Esta pila es una pila secuencial, ya que no
        utiliza memoria dinamica sino memoria estatica para la construccion de los
        nodos.
 Librerias:
        stdio.h
        string.h
        clNodo.h
 Indice:
        clPila : Inicializa las variables de la clase
        iObtenerCab : Regresa el valor del dato miembro iCabPila
        rObtenerNodo : Regresa una referencia a un nodo determinado
        vInsertarNodo : Inserta un nuevo nodo dentro de la pila
        vEliminarNodo : Elimina un nodo de la pila
 Funciones dependientes del sistema operativo:
        Ninguna
 Definiciones:
        _clPila  : Identifica la clase encargada del manejo de la Pila de Retroceso
        _MAX     : Numero maximo de elementos de la pila
 Pendientes:
        Ninguno
*/

#if !defined (_clPila)
#define _clPila

# include <stdio.h>
# include <string.h>
# include <clNodo.h>

# define MAX 100

/*-------------------------------------------------------------*/
/*          Clase para el manejo de la Pila de Retroceso       */
/*-------------------------------------------------------------*/
class clPila
{
```

```
int iCabPila;

struct clNodo aoPilaRet[MAX];

public:

/*-------------------------------------------------------------*/
/*          Inicializa las variables de la clase               */
/*-------------------------------------------------------------*/
 clPila (void)
/*
 Autor:
       Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
       Inicializa los datos miembro para evitar que contengan basura.
 Librerias:
       Ninguna
 Parametros:
       Ninguno
 Retorna:
       Nada
 Ejemplo:
       //Construye el objeto oPila
       clPila oPila;
*/
 {
 iCabPila=0;
 } /*** clPila () ***/


/*-------------------------------------------------------------*/
/*          Regresa el valor del dato miembro iCabPila         */
/*-------------------------------------------------------------*/
 int iObtenerCab(void)
/*
 Autor:
       Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
       Regresa el valor del dato miembro iCabPila, que representa la cabeza de
       la pila.
 Librerias:
       Ninguna
 Parametros:
       Ninguno
 Retorna:
       int que representa un indice a la cabeza de la pila
 Ejemplo:
       //Asigna al dato miembro iDist del nodo que esta a la cabeza de la pila un
       //valor de 10
       aoPilaRet[iObtenerCab].rObtenerDist()=10;
*/
 {
```

```
 return iCabPila;
} /*** iObtenerCab () ***/


/*-------------------------------------------------------------*/
/*          Regresa una referencia a un nodo determinado       */
/*-------------------------------------------------------------*/
 clNodo & rObtenerNodo(int iInd)
/*
 Autor:
       Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
       Regresa el nodo de la pila aoPilaRet determinado por el parametro iInd al
       utlizarlo como indice
 Librerias:
       Ninguna
 Parametros:
       iInd : Representa el indice que sera utilizado para determinar el nodo que
                    sera regresado de la pila secuencial
 Retorna:
       clNodo & que representa el nodo que sera regresado
 Ejemplo:
       //Asigna al elemento tres de la pila, en su dato miembro acDesde la que se
       //accese desde la entrada estandar
       gets(rObtenerNodo(2).apObtenerDesdeNodo());
*/
 {
 return aoPilaRet[iInd];
} /*** rObtenerNodo () ***/


/*-------------------------------------------------------------*/
/*          Inserta un nuevo nodo dentro de la pila            */
/*-------------------------------------------------------------*/
 void vInsertarNodo(char *apDesde,char *apHacia,int iDist)
/*
 Autor:
       Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
       Si la cabeza de la pila no ha superado el numero maximo de nodos permitido
       se copian los parametros que le fueron enviados a la funcion en el nuevo
       nodo, definido utilizando la cabeza de la pila como indice. Finalmente se
       incrementa la cabeza de la pila en uno para cuando se quiera agregar un
       nuevo nodo. En caso de que la cabeza de la pila hubiese sobrepasado el
       limite permitido durante la primera comprobacion, enconces solo se manda
       un mensaje de "Pila llena".
 Librerias:
       clNodo.h
 Parametros:
       apDesde : Representa la habitacion de origen
       apHacia : Representa la habitacion hacia donde hay una salida desde la
```

```
                                habitacion origen
        iDist : Representa la distancia desde la habitacion de origen hasta la
                                habitacion destino

 Retorna:
        Nada
 Ejemplo:
        //Inserta un nuevo nodo en la pila de retroceso que define que la habitacion
        //052 tiene una salida hacia la habitacion 065 y la distancia entre ambas
        //esde 45
        vInsertarNodo("052","065",45);
*/
 {
 if(iCabPila<MAX){
  strcpy(aoPilaRet[iCabPila].apObtenerDesdeNodo(), apDesde);
  strcpy(aoPilaRet[iCabPila].apObtenerHaciaNodo(), apHacia);
  aoPilaRet[iCabPila].rObtenerDistNodo()=iDist;
  iCabPila++;
 }
 else printf("Pila llena \n");
 } /*** vInsertarNodo () ***/



/*-------------------------------------------------------------------*/
/*                  Elimina un nodo de la pila                       */
/*-------------------------------------------------------------------*/
 void vEliminarNodo(char *apDesde,char *apHacia,int *iDist)
/*
 Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
        Verifica si la pila aun tiene elementos, de ser asi, se disminuye la cabeza
        de la pila en uno, con esto ultimo el ultimo elemento ha sido borrado. Luego
        se copian a los parametros, que son apuntadores, los valores del nodo que se
        esta borrando. Si la pila no tenia elementos desde un principio, entonces
        se manda un mensaje de "Pila vacia".
 Librerias:
        clNodo.h
 Parametros:
        apDesde : Representa la habitacion de origen
        apHacia : Representa la habitacion hacia donde hay una salida desde la
                                habitacion origen
        iDist : Representa la distancia desde la habitacion de origen hasta la
                                habitacion destino
 Retorna:
        Nada
 Ejemplo:
        //Elimina el ultimo nodo de la pila y copia el valor de los datos miembro
        //de ese nodo a las varibles apDesde, apHacia e iDist
        vEliminarNodo(apDesde,apHacia,iDist);
*/
```

```
  {
   if(iCabPila>0){
    iCabPila--;
    strcpy(apDesde,aoPilaRet[iCabPila].apObtenerDesdeNodo());

    strcpy(apHacia,aoPilaRet[iCabPila].apObtenerHaciaNodo());
    *iDist=aoPilaRet[iCabPila].rObtenerDistNodo();
   }
   else printf("Pila vacia \n");
  } /*** vEliminarNodo () ***/

}; /*** clPila ***/

#endif  /*** #if () ***/
```

4.3.6.6. clInc.h

```
/*-------------------------------------------------------------*/
/*                        clInc.h                              */
/*-------------------------------------------------------------*/
/*
 Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
 Compilador:
        GNU gcc  Version 2.7.2.1
 Descripcion:
        Descripcion de la estructura y rutinas de manejo de los incendios. Se
        definene una funcion para acceder al valor del objeto.
 Librerias:
        Ninguna
 Indice:
        apObtenerInc : Proporciona un apuntador al arreglo acInc.
Funciones dependientes del sistema operativo:
        Ninguna
 Definiciones:
        _clInc    : Identifica la clase encargada del manejo de los incendios
 Pendientes:
        Ninguno
*/

#if !defined (_clInc)
#define _clInc

/*-------------------------------------------------------------*/
/*    Clase para el manejo de incendios en habitaciones        */
/*-------------------------------------------------------------*/
class clInc{

 char acInc[20];

 public:

/*-------------------------------------------------------------*/
/*          Proporciona un apuntador al arreglo acInc          */
/*-------------------------------------------------------------*/
 char * apObtenerInc(void)
/*
 Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
        Regresa el primer elemento de la variable acInc, la cual al ser un arreglo
        puede ser utilizada como apuntador.
 Librerias:
        Ninguna
 Parametros:
```

```
        Ninguno

  Retorna:
        Un apuntador a caracter que es el primer elemento del arreglo acInc.
  Ejemplo:
        //Se asigna el valor del incendio desde la entrada estandar.
        scanf("%s",apObtenerInc())
*/
  {
   return acInc;
  } /*** apObtenerInc () ***/

}; /*** clInc ***/

#endif /*** #if () ***/
```

4.3.6.7. clAnaliz.h

```
/*-------------------------------------------------------------*/
/*                      clAnaliz.h                             */
/*-------------------------------------------------------------*/
/*
 Autor:
        Jauregui Espinosa Hugo,          (15/Junio/98)
 Compilador:
        GNU gcc  Version 2.7.2.1
 Descripcion:
        Verifica el numero minimo de interconexiones para llegar a la salida,
        determinando si se ha quedado atrapado por el fuego.
 Librerias:
        stdio.h
        string.h
        clInc.h
        clPila.h
        clNivel.h
 Indice:
        clAnaliz : Inicializa las variables de la clase
        vObtenerInc : Determina todas las areas en donde se presenta un incendio
        vDeterminarRuta : Determina la ruta de evacuacion conciderando la ubicacion de los
incendios
        iVerificarInc : Verifica la existencia de incendio en algun area
        iEncontrarSigNodo : Encuentra el siguiente nodo a visitar
        vVerificarArea : Verifica si existe union entre las areas que llegan como parametros
 Funciones dependientes del sistema operativo:
        Ninguna
 Definiciones:
        _clAnaliz   : Identifica la clase encargada del analisis
 Pendientes:
        Ninguno
*/

#if !defined (_clAnaliz)
#define _clAnaliz


# include <stdio.h>
# include <string.h>
# include <clInc.h>
# include <clPila.h>
# include <clNivel.h>


/*-------------------------------------------------------------*/
/* Clase encargada del analisis de las condiciones de evacuacion */
/*-------------------------------------------------------------*/
class clAnaliz
```

```
{

 int iEncPos;
 int iRec;

 clInc aoInc[20];

 clPila oPila;
 clNiv oNiv;

 public:

/*-------------------------------------------------------*/
/*          Inicializa las variables de la clase         */
/*-------------------------------------------------------*/
 clAnaliz (void)
/*
 Autor:
       Jauregui Espinosa Hugo,          (15/Junio/98)
 Descripcion:
       Inicializa los datos miembro para evitar que contengan basura.
 Librerias:
       clNivel.h
 Parametros:
       Ninguno
 Retorna:
       Nada
 Ejemplo:
       //Construye el objeto oAnaliza
       clAnaliz oAnaliza;
*/
 {
 iEncPos=0;
 iRec=0;
 oNiv.vCargarBaseCon();
 } /*** clAnaliz () ***/


/*-------------------------------------------------------*/
/* Determina todas las areas en donde se presenta un incendio */
/*-------------------------------------------------------*/
 void vObtenerInc (void)
/*
 Autor:
       Jauregui Espinosa Hugo,          (15/Junio/98)
 Descripcion:
       Se inicializa la variable iRes a 1 y que servira de bandera, luego mientras
       sea diferente de 0, lo cual querria decir que ya no hay algun otro incedio
       que dar de alta, se pregunta por el area o areas en donde han
       aparecido incendios. Cada uno de ellos es almacenado en un arreglo, creando
       una instancia mas de la clase clInc.
```

```
 Librerias:
        clInc.h

        stdio.h
 Parametros:
        Ninguno
 Retorna:
        Nada
 Ejemplo:
        //Da de alta todos los incendios que han aparecido
        vObtenerInc();
*/
 {
 int iRes=1;

 while (iRes!=0)
 {
  printf("Area en donde se detecto el incendio? ");
  scanf("%s",aoInc[iRec].apObtenerInc());
  iRec++;
  printf("Existe otro incendio (0) No (1)Si ? ");
  scanf("%d", &iRes);
 }
} /*** vObtenerInc () ***/



/*-------------------------------------------------------------------------*/
/* Determina la ruta de evacuacion conciderando la ubicacion de los incendios */
/*-------------------------------------------------------------------------*/
 void vDeterminarRuta(char *apHacia,int iPosInic)
/*
 Autor:
        Jauregui Espinosa Hugo,                 (15/Junio/98)
 Descripcion:
        Se iguala iDist, que sirve como variable acumulable, a 0 e iTemp, que sirve
        como variable temporal para controlar el ciclo while, con el valor del
        parametro iPosInic. Si la cabeza de la pila es menor a 1, es decir que solo
        tiene el elemento de partida, entonces quiere decir que no se ha encontrado
        ninguna ruta de evacuacion, por lo cual se manda un mensaje de "Esta
        atrapado" y termina la funcion. De lo contrario, es decir, si se ha
        encontrado una ruta de evacuacion, y que esta se encuentra en la pila de
        retroceso, entonces mientras que la variable iTemp sea menor a la cabeza de
        la pila de retroceso, entoces se imprime el area hacia la que se deben
        mover, se suma la distancia a la variable iDist y se incrementa en uno a
        iTemp. Luego se muestra el ultimo destino, que es la salida hacia la que
        se debe dirigir la ruta y la distancia      todal recorrida hacia dicha salida.
Librerias:
        clPila.h
        clNodo.h
Parametros:
        apHacia : Destiono final de la ruta, que es la salida de emergencia
```

```
        iPosInic : Posicion desde la cual se va a empezar a mostrar el contenido de
                                la pila de retroceso
 Retorna:

        Nada
 Ejemplo:
        //Muestra la ruta de evacuacion hacia la salida 055, mostrando la pila de
        //retroceso desde su segundo elemento
        vDeterminarRuta("055",1);
*/
 {
  int iDist, iTemp;

  iDist=0;
  iTemp=iPosInic;

  if (oPila.iObtenerCab()<1){
   printf("Esta atrapado");
   return;
  }

  while(iTemp<oPila.iObtenerCab()){
   printf("%s hacia ",oPila.rObtenerNodo(iTemp).apObtenerDesdeNodo());
   iDist+=oPila.rObtenerNodo(iTemp).rObtenerDistNodo();
   iTemp++;
  }
  printf("%s \n",apHacia);
  printf("La distancia es %d \n", iDist);
 } /*** vDeterminarRuta () ***/



/*-------------------------------------------------------------*/
/*      Verifica la existencia de incendio en algun area       */
/*-------------------------------------------------------------*/
 int iVerificarInc (void)
/*
 Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
        Iguala la variable iAvance, que servira como control para el ciclo while, con
        el dato miembro iRec menos uno, con lo cual se recorrera todo el arreglo de
        incendios. Luego, mientras que iAvance sea mayor que cero, disminuyendose de
        uno en uno, se verifica si el area, determinada por el dato miembro
        iEncPos que se usa como indice para accesar al arreglo de areas del
        objeto oNiv, es igual a algun area en donde se ha detectado incendio.
        En caso de que sea cierta alguna de las comparaciones, la funcion regresa 0;
        de lo contrario se regresa 1.
 Librerias:
        string.h
        clNivel.h
        clArea.h
```

```
        clInc.h
 Parametros:
        Ninguno
 Retorna:

        int que representa si el area determinada por el parametro iEncPos es
        una de las que han sido declaradas como con incendio. 1 si es cierto, 0 si
         es falso.
 Ejemplo:
        //Verifica si el area determinada por el parametro iEncPos al
        //usarlo como indice del arreglo de areas, esta incendiada, en
        //cuyo caso despliega el mensaje "Esta incendiada esta area"
        if(!iVerificarInc()) printf("Esta incendiada esta area");
*/
 {

 int iAvance=iRec-1;
 do
 {
 if (!strcmp(oNiv.rObtenerArea(iEncPos).apObtenerHacia(),aoInc[iAvance].apObtenerInc()))
 {
  return 0;
 }
 }while(iAvance--);
 return 1;
} /*** iVerificarInc () ***/



/*-------------------------------------------------------------*/
/*              Encuentra el siguiente nodo a visitar          */
/*-------------------------------------------------------------*/
 int iEncontrarSigNodo(char *apDesde,char *apCualqLug)
/*
 Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
        Se inicializan las variables iPos, que sirve para almacenar la posicion del
        elemento cuyos datos seran utilizados comos siguiente nodos revisado,
        iEncPos, que sirve como variable temporal para contener la posicion del nodo
        que sera pasado a iPos, e iDist, que sirve como variable temporal para
        almacenar la distancia mas grande recorrida hasta entonces, a 0 e iPrimReg,
        que sirve como bandera para indicar si se esta revisando el primer elemento
        del arreglo, como Falso. Mientras que iEncPos es menor que el dato miembro de
        la clase clNivel que representa la ultima posicon del arreglo de las
        areas, se verifica si existe algun area que no haya sido
        visitada y que no este incendiada y si la distancia que se recorre desde esa
        area hacia cualquier otra area es mayor que la variable iDist, de
        ser cierto, si iPos es igual a 0 entonces se asigna a la variable iPrimReg el
        valor de Verdadero, luego pos se iguala con iEncPos y iDist se iguala  a la
        distancia que existe desde el area que se esta tomando como origen
        hacia cualquier otra area cuya distancia haya sido mayor a la variable
```

```
        iDist. Luego se incrementa en uno iEncPos. Despues, si iPos es diferente
        de 0 o iPrimReg es diferente de 0 se copia del arreglo de areas cuyo
        indice es iPos el area de destino a la variable que llego como
        parametro apCualqLug, se le asigna a la bandera de visitado del area
        un valor de Verdadero y se regresa la distancia que hay de esa area a

        aquella cuya distancia es mayor que iDist. Si iPos era igual a 0 o iPrimReg
        era Falso, se regresa 0.
  Librerias:
        string.h
        clNivel.h
        clArea.h
        clInc.h
  Parametros:
        apDesde : Area que se toma como origen
        apCualqLug :
  Retorna:
        int que representa la distancia que hay de un area a otra o 0 de
        no encontrarse ninguna adecuada
  Ejemplo:
        //Verifica si existe un area que cumpla con los requisitos
        if(iEncontrarSigNodo()) printf("Existe un area");
*/
 {
  int iPos, iDist, iPrimReg;

  iPos=iDist=0;
  iEncPos=0;
  iPrimReg=Falso;

  while(iEncPos<oNiv.iObtenerUltPos()){
   if(!strcmp(oNiv.rObtenerArea(iEncPos).apObtenerDesde(), apDesde)&&
!oNiv.rObtenerArea(iEncPos).rObtenerVis() && iVerificarInc()){
    if(oNiv.rObtenerArea(iEncPos).rObtenerDist()>iDist){
     if(!iPos)
     {
      iPrimReg=Verdadero;
     }
     iPos=iEncPos;
     iDist=oNiv.rObtenerArea(iEncPos).rObtenerDist();
    }
   }
   iEncPos++;
  }
  if (iPos || iPrimReg){
  strcpy(apCualqLug, oNiv.rObtenerArea(iPos).apObtenerHacia());
  oNiv.rObtenerArea(iPos).rObtenerVis()=Verdadero;
  return oNiv.rObtenerArea(iPos).rObtenerDist();
  }
  return 0;
 } /*** iEncontrarSigNodo () ***/
```

```
/*-------------------------------------------------------------------------*/
/*  Verifica si existe union entre las areas que llegan como parametros    */
/*-------------------------------------------------------------------------*/
 void vVerificarArea(char *apDesde,char *apHacia)

/*
 Autor:
        Jauregui Espinosa Hugo,              (15/Junio/98)
 Descripcion:
        Se declaran las variables iTemp, que sirve para guardar la distancia que
        existe entre las areas representadas por apDesde y apHacia por medio
        de la utilizacion de la funcion iVerificarUnion; e iDist, que se utiliza para
        guardar la distancia maxima a la que se puede avanzar utiiizando la funcion
        iEncontrarSigNodo, asi como acCualqLug, que sirve para almacenar el
        area que saliendo hacia la cual se avanza mayor distancia. Se iguala la
        variable iTemp con la verificacion de si hay union entre las areas que
        llegaron como parametro de la funcion; de existir union, se inserta un
        nuevo nodo a la pila de retroceso con la informacion del area de
        origen y destino, asi como la distancia entre ambas y se termina la funcion.
        Si no existe union, entonces se iguala la variable iDist con el resultado de
        la invocacion a la funcion iEncontrarSigNodo, pasandole como parametros el
        parametro que llego apDesde y la variable acCualqLug.
 Librerias:
        string.h
        clNivel.h
        clArea.h
        clInc.h
 Parametros:
        apDesde :
        apHacia :
 Retorna:
        Nada
 Ejemplo:
        //Verifica si existe un area que cumpla con los requisitos
        if(iEncontrarSigNodo()) printf("Existe un area");
*/
 {
 int iTemp, iDist;
 char acCualqLug[20];

 if (iTemp=oNiv.iVerificarUnion(apDesde, apHacia)){
  oPila.vInsertarNodo(apDesde, apHacia,iTemp);
  return;
 }

 if(iDist=iEncontrarSigNodo(apDesde,acCualqLug)){
  oPila.vInsertarNodo(apDesde,apHacia,iDist);
  vVerificarArea(acCualqLug, apHacia);
 }
```

```
  else if (oPila.iObtenerCab()>0){
   oPila.vEliminarNodo(apDesde,apHacia, &iDist);
   vVerificarArea(apDesde,apHacia);
  }
 } /*** vVerificarArea () ***/


}; /*** clAnaliz ***/

#endif  /*** #if () ***/
```

Apéndice C. Descripción de los Campos de la Base de Datos BDEI.

A continuación se presentan las tablas que forman la base de datos BDEI y la descripción de sus campos respectivos.

ACCESOS			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
EMP_RFC	PK, FK	varchar(20)	Registro Federal de Contribuyentes del empleado.
ARE_IdArea	PK	char(3)	Identificador del área a la que accesó el empleado.
ACC_FecHoraEnt	PK	datetime	Fecha y hora de la entrada al área.
ACC_FecHoraSal		datetime	Fecha y hora de la salida del área.

EMPLEADOS			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
EMP_RFC	PK	varchar(20)	Registro Federal de Contribuyentes del empleado.
ARE_IdArea	FK	char(3)	Identificador del área en donde trabaja el empleado.
EMP_FecNac		datetime	Fecha de nacimiento del empleado.
EMP_Horar		varchar(20)	Horario de trabajo del empleado.
EMP_Nomb		varchar(40)	Nombre completo del empleado.
EMP_Ocup		text	Ocupación del empleado.

AREAS			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
ARE_IdArea	PK	char(3)	Identificador del área.
DEP_IdDep	FK	smallint	Identificador del departamento.
ARE_AireAcond		varchar(20)	Velocidad del Aire Acondicionado.
ARE_ConsPromEnerg		float	Consumo promedio de energía.
ARE_Horar		varchar(20)	Horario de trabajo en el área.
ARE_Humed		float	Humedad del área.
ARE_Temp		float	Temperatura del área.
ARE_Sup		float	Superficie del área.

CONTIGUAS			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
ARE_Cont	PK	char(3)	Identificador del área contigua.
ARE_Area	PK	char(3)	Identificador del área.
CON_Dist		smallint	Distancia entre un área y otra.

INSTALACION			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
DIS_IdDisp	PK, FK	smallint	Identificador del dispositivo.
ARE_IdArea	PK, FK	char(3)	Identificador del área.
INS_FecInst		datetime	Fecha de instalación del dispositivo.
INS_Estado		bit	Estado de funcionamiento del dispositivo.

DEPARTAMENTOS			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
DEP_IdDep	PK	smallint	Identificador del departamento.
DEP_Nomb		varchar(20)	Nombre del departamento.

DEPNIV			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
NIV_IdNiv	PK, FK	tinyint	Identificador del nivel.
DEP_IdDep	PK, FK	smallint	Identificador del departamento.

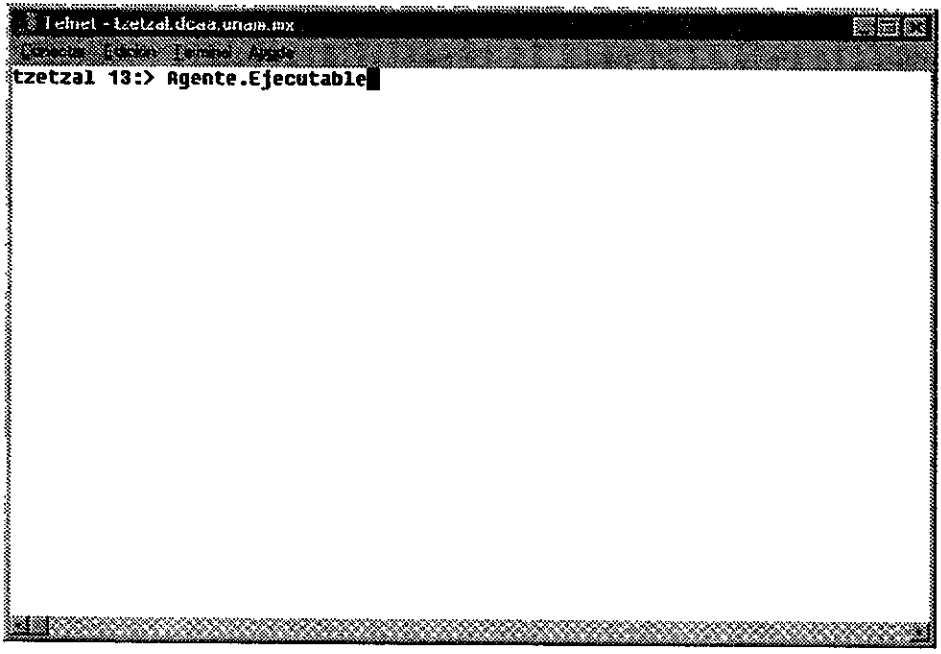
DISPOSITIVOS			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
DIS_IdDisp	PK	smallint	Identificador del dispositivo.
DIS_Desc		text	Descripción del dispositivo.
DIS_Dur		int	Duración promedio del dispositivo.
DIS_Nomb		varchar(20)	Nombre del dispositivo.
DIS_NoSerie		varchar(20)	Número de serie del dispositivo.
DIS_PerMon		varchar(20)	Periodo de monitoreo.

NIVELES			
Campo	*Tipo*	*Tipo y Extensión*	*Descripción*
NIV_IdNiv	PK	tinyint	Identificador del nivel.
NIV_Nomb		varchar(20)	Nombre del nivel.

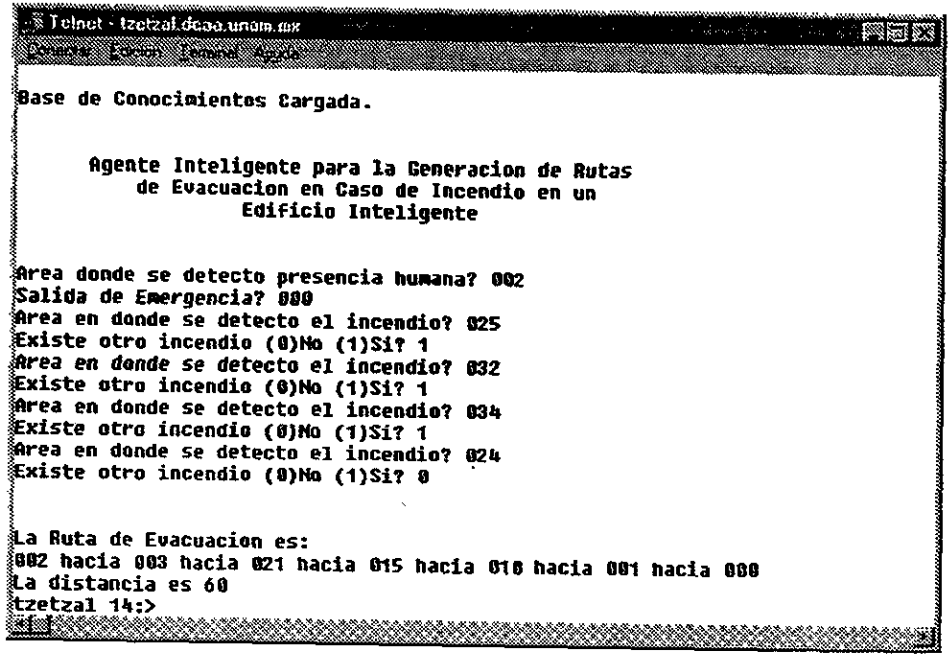
Apéndice D. Pantallas del Agente Inteligente.

A continuación se presentan las pantallas del Agente Inteligente. El Agente corre desde un servidor Unix, el cual es el mismo donde esta la base de datos.

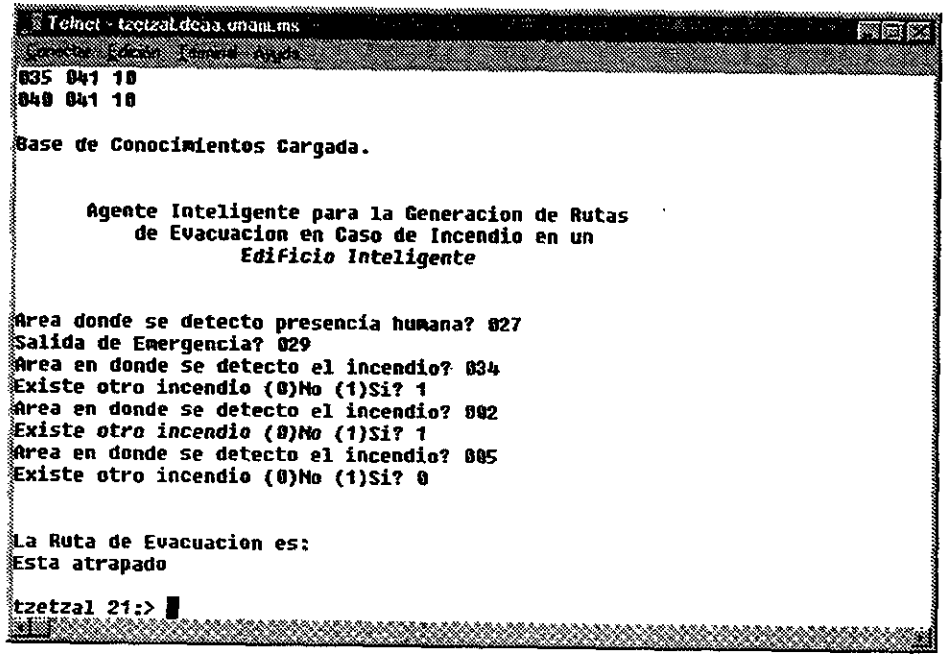
El Agente Inteligente se ejecuta con la invocación a su archivo ejecutable, en este caso el archivo ejecutable fue llamado Agente.Ejecutable:



Al introducir uno de los casos de prueba, donde existía una ruta de evacuación, se presenta la sucesión de áreas por las que se debe pasar para llegar al área de salida, además de la distancia recorrida durante el recorrido:



Al introducir uno de los casos de prueba, donde no existía una ruta de evacuación, se presenta un mensaje con la leyenda "Esta atrapado" para que se tomen otras medidas con el objetivo de evacuar a esa o esas personas atrapadas, evidentemente no se presenta ninguna distancia recorrida pues no existe una ruta de evacuación segura:



GLOSARIO

1. *Acabados.* Cualquiera de las operaciones destinadas a mejorar el aspecto final de una pieza o a proferir a su superficie alguna propiedad particular.

2. *Acometidas.* Cada una de los ramales que desembocan en una alcantarilla o sumidero.

3. *Adaptador(es).* Dispositivo que convierte bits de información dispuestos en serie a su disposición en paralelo, con el fin de emplearlos en la memoria de exploración.

4. *Algoritmo.* Conjunto finito de reglas que proporcionan una secuencia de operaciones para resolver un tipo específico de problemas. Una secuencia finita de pasos, dirigidos a realizar una tarea específica.

5. *Algoritmos Genéticos.* Es un *algoritmo* matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud.

6. *Analógico(s).* Información presentada de manera secuencial y continua. Análogo implica operación continua y se contrapone con digital, que es desmenuzar en números.

7. *Aprendizaje simbólico.* Engloba todas aquellas investigaciones que se orientan hacia la búsqueda de los principios formales de la inteligencia. El principio básico de ésta corriente se condensa en la hipótesis de los símbolos físicos.

8. *Aprendizaje.* Proceso mediante el cual el ser humano adquiere sus hábitos y asume la cultura de su entorno.

9. *Arquitecturas masivamente paralelas.* Construcción de un tipo de ordenador que posea múltiples unidades de proceso. Los computadores masivamente paralelos se han convertido en la alternativa más prometedora para conseguir grandes potencias de cálculo. Los multiprocesadores con memoria distribuída, también denominados multicomputadores, han sufrido una evolución muy rápida en la última década gracias a su excelente escalabilidad. Una máquina paralela es síncrona si en cada etapa, todos los procesadores ejecutan una instrucción (no necesariamenete la misma) simultáneamente; si no, se dice que es asíncrona. Objetivo principal del paralelismo: mejorar tiempos de ejecución. Un algoritmo paralelo puede ser visto como una colección de módulos independientes, que pueden ser ejecutados simultáneamente. Los módulos se comunican entre ellos durante la ejecución de un algoritmo.

10. *ASHRAE.* American Society of Heating, Refrigeration and Air Conditioning Engineers. Sociedad Americana de Ingenieros en Calentadores, Refrigeración y Aire Acondicionado.

11. *Aspersores.* Rociador, aparato para asperjar o rociar.

12. *Automático.* Dícese de lo que obra por medio de mecanismos, sin necesitar la intervención de operadores.

13. *Automatización.* Reemplazo de las operaciones manuales por métodos informatizados.

14. *Balaustra(s).* De balaustre. Cada una de las columnas pequeñas adornadas con molduras que se disponen en serie a modo de cerramiento.

15. *Base de datos.* Un conjunto de archivos interrelacionados que es creado y manejado por un sistema de gestión o de administración de bases de datos (DBMS).

16. *Beep.* Señal auditiva que emite una máquina.

17. *Búsqueda heurística de soluciones.* Métodos de búsqueda que utilizan conocimientos heurísticos sobre el campo para ayudar a centrar la búsqueda. Operan generando y comprobando estados intermedios a lo largo de rutas potenciales de solución.

18. C ++. Lenguaje de programación híbrido (programación orientada a objetos y estructurada), de nivel intermedio, en el cual se programan la mayoría de las aplicaciones de alta complejidad. C++ es una versión de C orientada a objetos creada por Bjarne Stroustrup.

19. *Cable coaxial*. Un cable de alta capacidad utilizado en comunicaciones y vídeo. Contiene un alambre aislado, sólido o multifilamento, que está rodeado por una pantalla sólida o de malla trenzada, bajo una cubierta exterior. El revestimiento exterior de teflón para protección contra incendios es opcional. Los cables *coaxiales* proveen un ancho de banda muy superior al de los pares trenzados. Un tipo de cable eléctrico en el cual un alambre sólido de metal es cubierto por un aislante, todo lo cual es protegido por una malla de metal cuyo eje de curvatura coincide con el del alambre, de ahí el nombre de coaxial (eje común).

20. *Centralizados*. De central. Reunir en un centro común. Hacer depender todas las actividades políticas y administrativas de un poder central único.

21. *CFE*. Compañía Federal de Electricidad.

22. *Cisternas*. Depósito subterráneo para almacenar y conservar el agua.

23. *Coeficiente(s)*. Valor numérico o factor que caracteriza una propiedad específica de una materia dada y que es constante si las condiciones son las mismas que han servido para calcularlo.

24. *Combinatoria(s)*. Parte de las matemáticas que trata de los grupos que pueden formarse con varios elementos combinándolos de diversas formas en el interior de cada grupo.

25. *Comprensión*. Entender, penetrar, alcanzar.

26. *Compuertas*. Tablero o plancha que se desliza encajado entre ranuras, para cortar o graduar el paso del agua en canales y presas. Puerta que se desliza verticalmente entre dos ranuras y sirve para detener las aguas, dejarlas correr o regular su gasto en los canales, presas y otras instalaciones hidráulicas. Es el dispositivo que conecta y maneja el tráfico entre redes de distintas topologías y distintos protocolos.

27. *Conjetura(s)*. Opinión basada en indicios, no en pruebas.

28. *Conmutación.* Técnica para manejar altos volúmenes de tráfico en una red descomponiendo los mensajes en paquetes de longitud fija que son transmitidos a su destino a través de la ruta más oportuna.

29. *Control distribuido.* (Distributed Control System) Sistema de *Control distribuido.* Sistema de control en el proceso que utiliza computadoras desembolsadas en la línea de fabricación para su control.

30. *Control.* Es la medición y corrección del desempeño, en las actividades de los subordinados para asegurarse de que los objetivos y los planes se están llevando a cabo con eficiencia, eficacia y congruencia.

31. *Controlador(es).* En la computadora, unidad de control, o *controlador,* es un hardware que controla las actividades de los periféricos, tales como un disco o una pantalla de presentación. A partir de señales que recibe la CPU, ejecuta las transferencias físicas de datos entre la memoria y el dispositivo periférico.

32. *Converger.* Dirigirse varias cosas a un punto de unión, llegando o no a juntarse.

33. *Chiller.* Enfriador de agua.

34. *Deducción(es).* Método de razonamiento por el que, partiendo de un principio general, se llega a un principio particular. Un proceso de razonamiento en el que la conclusión sigue a las premisas dadas.

35. *Depuración.* Es el proceso de encontrar y corregir errores.

36. *Detector(es).* En física, aparato que sirve para detectar.

37. *Diagnosis.* Identificación de una enfermedad basándose en sus síntomas.

38. *Dimmer.* Grado de oscuridad.

39. *Diverger.* Irse separando progresivamente más de otras dos o más líneas o superficies. Acción de separarse progresivamente más de otras a medida que se alejan de sus orígenes comunes.

40. *Ducto(s).* Cualquier vía, generalmente cerrada, para que circule por ella alguna cosa (agua).

41. *Edificio Inteligente.* Edificio que optimiza, aprovecha o procesa al máximo los recursos con que cuenta a través de las herramientas necesarias para administrar los sistemas ambientales y su coordinación para brindar la comodidad y seguridad requeridos por los usuarios para que éstos tengan un mejor desempeño.

42. *Electromecánicos.* Dícese de los aparatos de gobierno o regulación especialmente los que obran a distancia, constituidos por órganos mecánicos y eléctricos.

43. *Electrónica(s).* Ciencia que estudia los fenómenos originados por el paso de partículas atómicas electrizadas a través de espacios vacíos o de gases más o menos enrarecidos, y técnica que aplica estos conocimientos a la industria.

44. *Encapsulando.* Ocultamiento de información. Es la propiedad que permite que las variables de una clase X solo pueden ser accesadas por las funciones miembro de X. Es la propiedad que evita que las características de un dato miembro puedan ser modificadas por el exterior, de esta forma un objeto podrá ser usado mediante sus acciones públicas, es decir, accesibles a todos..

45. *Enfoque del modelo cliente/servidor.* Existen cinco enfoques del modelo cliente/servidor que son: a) Presentación remota, b) Presentación distribuida, c) Aplicación distribuida, d) Manejo remoto de datos y e) Manejo distribuido de datos.

46. *Entalpía.* Energía calorífica de un sistema termodinámico, cuya magnitud depende de los estados inicial y final del mismo.

47. *Entender.* Tener idea clara de las cosas, comprenderlas. Conocer, penetrar, discurrir, inferir, deducir, creer, pensar, juzgar.

48. *Ergonomía.* Un producto diseñado ergonómicamente implica que el dispositivo combina suavemente con el cuerpo o acciones de una persona.

49. *Escalabilidad.* Capacidad de ampliar. Implica un cambio mínimo en procedimientos actuales para acomodar el crecimiento.

50. *Estandarizados.* Normalizados. Estándard especificación que debe utilizar un sistema para cumplir con las características que exige el mercado si es que quiere ser compatible y lograr la comunicación.

51. *Estructura.* Incluye todos los elementos comunes (pisos, muros, ventanas, etcétera) pero además, necesita incluir plafones, *ductos* adicionales para comunicaciones, áreas para los equipos de control y telecomunicaciones, espacio para colocar piso falso, analizar la orientación de la *estructura* para aprovechar la luz del sol, entre otros. Modo de que son dispuestas las partes constituyenies de un todo.

52. *Excitatoria.* Conexión de intensidad positiva que afecta a la neurona que la recibe, activándola o disparándola. Dícese del elemento que puede provocar, intensificar alguna acción.

53. *Expansibilidad.* Dilatación de los cuerpos, aumento de su volumen o extensión de su superficie.

54. *Facsímil.* Copia, reproducción o imitación exacta de un documento. Reproducción a distancia, por ondas hertzianas, de documentos en blanco y negro. Originalmente llamado telecopying (telecopia), es la comunicación de una página impresa entre lugares lejanos.

55. *Fiabilidad.* Es el grado de confianza que puede concederse a un elemento, atendiéndose a la calidad de los materiales empleados, la perfección con que ha sido labrado, y la multiplicidad y cuidado de los controles y pruebas a que ha sido sometido.

56. *Fibra óptica.* Sistema de transmisión que utiliza fibra de vidrio como conductor de frecuencias de luz visible o infrarrojas. Este tipo de transmisión tiene la ventaja de que no se pierde casi energía pese a la distancia (la señal no se debilita) y que no le afectan las posibles interferencias electromagnéticas que sí afectan a la tecnología de cable de cobre clásica.

57. *Fotosensor.* Un dispositivo sensible a la luz que se usa en el mecanismo de exploración óptica.

58. *fpm.* Feet per minute. Pies por minuto. Un pie es una medida de longitud que equivale a 28 cm..

59. *Frame(s).* Trama o marca de referencia. Estructura. Secuencia de bits. Es una estructura de datos para representar situaciones y objetos estereotipados. En gráficos por computadora, el contenido de una pantalla de datos a su espacio equivalente de almacenamiento. En comunicaciones, un grupo de bits que conforman un bloque elemental de datos para su transmisión por ciertos protocolos. En inteligencia artificial, *estructura* de datos que contiene una descripción general de un objeto. La descripción proviene de conceptos básicos y de la experiencia.

60. *Funcional.* Que responde a una función determinada.

61. *Generador.* Aparato productor de energía eléctrica por transformación de otra forma de energía. Circuito o dispositivo que engendra señales o corrientes.

62. *Genes.* Segmento de ADN que constituye la unidad hereditaria de los seres vivos, y que, en número constante para cada especie, forma parte de los cromosomas de las células.

63. *GNU.* Proyecto de la Free Software Foundation para proveer reemplazos distribuidos gratuita y libremente para Unix.

64. Grados centígrados. Escala que asigna de forma arbitraria el número 0 al punto fijo inferior y el número 100 al punto superior, las cien unidades representan temperaturas comprendidas entre el punto de congelación y el punto de ebullición del agua.

65. Grados Farenheit. Escala de medición de temperaturas que se basa en la relación de diferentes puntos fijos. La temperatura de una solución congelada de agua de sal como su punto inferior o 0. El punto superior lo escogió tomando como base la temperatura del cuerpo humano asignándole el 96°F.

66. *Grafo.* Un grafo consta de dos clases de elementos: a) un conjunto N cuyos elementos se llaman nodos, vértices o puntos; b) un conjunto S de parejas no ordenadas de nodos diferentes, llamadas segmentos o aristas.

67. *Heurística.* Un método de resolver problemas utilizando exploración y métodos de prueba y error. El *diseño* heurístico de programas provee de un marco para resolver el problema en contraposición con un conjunto fijo de reglas (*algoritmo*) que no puede variar.

68. *Hidroneumáticos.* Dícese de los dispositivos en cuyo funcionamiento intervienen a la vez un líquido y un gas comprimido.

69. *Hipótesis.* Suposición o conjetura que se hace sobre algo y de la cual se infiere una consecuencia. En matemáticas, parte de un teorema que suponemos que se cumple.

70. *Incandescente(s).* Dícese de la materia cuando, por efecto del calor, toma color rojo o blanco luminoso.

71. *Incertidumbre.* Ausencia de certidumbre, inseguridad. Desasosiego causado por la duda.

72. *Inferencia(s).* Deducir una conclusión en base a los hechos y reglas contenidas en una base de conocimientos, utilizando diversas técnicas de la inteligencia artificial.

73. *Informática.* Campo del conocimiento que se dedica a desarrollar conocimientos y técnicas para el procesamiento de información con auxilio de dispositivos y máquinas. La Informática no utiliza solamente computadoras, aunque ellas son el auxiliar más poderoso de que dispone.

74. *Inhibitoria.* Conexión de intensidad negativa que afecta a la neurona que la recibe impidiéndole que se dispare.

75. *Inteligencia Artificial.* Es la parte de la ciencia computacional que se encarga del *diseño* de sistemas computacionales inteligentes, dicho de otra manera, sistemas que exhiben las características que asociamos con la inteligencia humana (su ambiente, entendimiento del lenguaje, *aprendizaje*, razonamiento, solución de problemas, etc.).

76. *Inteligente(s).* Grado en que un individuo puede resolver satisfactoriamente a una nueva situación o problema. Está basado en el nivel de conocimientos individuales y en la habilidad para manipular apropiadamente y reformular dichos conocimientos y datos de entrada en función de los requerimientos de la situación o problema.

77. *Interfase.* El sistema mediante el cual el usuario interacciona con el ordenador. En general la unión entre dos componentes. Una conexión e interacción entre hardware, software y usuario. Las interfaces de hardware son los conectores, zócales y cables que transportan las señales eléctricas

en un orden prescrito. Las interfaces de software son los lenguajes, códigos y mensajes que utilizan los programas para comunicarse unos con otros, tal como entre un programa de aplicación y el sistema operativo. Las interfaces de usuario son los teclados, ratones, diálogos, lenguajes de comando y menúes empleados para la comunicación entre el usuario y la computadora.

78. *Interfaz.* En electrónica, dispositivo que conecta dos sistemas o circuitos.

79. *Interoperabilidad.* Proceso donde las computadoras pueden operar interactuando con otras a través de una red sin conversión de datos o intervención humana. Interoperable. Capacidad de un sistema para funcionar de forma adecuada sin otro sistema.

80. ISO. En inglés, sigla de International Standards Organization (Organización Internacional de Normas).. Es una organización de carácter voluntario fundada en 1946 responsable de la creación de normas internacionales en muchas áreas, entre las que se incluyen la informática y las comunicaciones. Está constituida por las organizaciones de normalización de sus 89 países miembros.

81. *Kw/h.* Kilowatt por hora.

82. *Lenguaje natural.* Lenguaje cuya sintaxis y semántica son muy complejas y están a menudo interrelacionadas, en ellos es posible escribir expresiones que sean sintácticamente correctas y sin embargo carentes de significado.

83. *LISP.* LISt Proccesing. Procesamiento de listas. Lenguaje utilizado para desarrollos de inteligencia artificial. Un lenguaje de programación de alto nivel utilizado extensamente en programación no numérica, en la cual se manipulan objetos, más que números

84. *Lógica.* Ciencia de las formas y leyes generales del pensamiento humano. Método, correspondencia con lo razonable.

85. *Luminarias.* Cualquier aparato, provisto de lámparas, propio para alumbrar locales o edificios.

86. *Lux(es).* Unidad de intensidad de iluminación, cuyo símbolo es lx, equivalente a la iluminación uniforme de una superficie que recibe un flujo luminoso de un lumen por metro cuadrado.

87. *Métodos neumáticos.* Métodos *Analógicos.*

88. *Microelectrónica.* La miniaturización de circuitos electrónicos

89. *Microprocesadores.* Circuito integrado que puede realizar las funciones de una unidad central de un ordenador. Una CPU en un solo chip. Para funcionar como una computadora, requiere suministro de potencia, reloj y memoria. La primera generación de microprocesadores fueron los 8080 de Intel, Z80 de Zilog, 6800 de Motorola y 6502 de Rockwell Internacional. El primer microprocesador fue creado por Intel.

90. *Modelo matemático.* Es una representación cuantitativa de la realidad, se aplican en los sistemas para apoyar las funciones del procesamiento de transacciones de una organización y para proporcionar la información para el apoyo de las actividades vitales de planeación y control. Consta de una lista de variables (P y t) que describen la situación dada, junto con una o más ecuaciones que relacionan esas variables ($dP / dt = KP$, $P(o) = Po$) que son conocidas o que se supone que tienen validez.

91. *Modular(es).* Proceso que realizan los módems para adaptar la información digital a las características de las líneas telefónicas analógicas. Mezclar una voz o señal de datos con una portadora para transmisión en una red de comunicaciones. Los datos se modulan sobre la portadora por varios métodos, incluyendo modulación de amplitud, en la cual la altura de la onda se cambia; o modulación de frecuencia, en lo cual la frecuencia se cambia; o modulación de fase, en la cual la fase (polaridad) de la onda se cambia.

92. *Modularidad.* Este criterio de desempeño general significa que se pueden agregar o modificar fácilmente componentes de hardware o módulos de software. El componente estructural de la tecnología puede crecer y cambiar para satisfacer las necesidades de un sistema de información que crece y cambia.

93. *Módulo(s).* Un componente autónomo hardware o software que interactúa con un sistema mayor. Los módulos del hardware se hacen a menudo para enchufar al sistema principal. Los módulos de programas se diseñan para manejar una tarea específica dentro de un programa mayor.

94. *Neurona.* Célula nerviosa de forma generalmente irregular; consta de un cuerpo celular donde se encuentra el núcleo, del que parten numerosas ramificaciones, llamadas dendritas, receptoras de los estímulos; además, del cuerpo celular sale una prolongación única y más larga, el axón o neurita, que se encarga de enviar los estímulos nerviosos.

95. *Nodo(s).* Por definición punto donde convergen más de dos líneas. A veces se refiere a una única máquina en Internet. Normalmente se refiere a un punto de confluencia en una red. Punto en que se produce la interferencia de dos ondas en el movimiento vibratorio. Es el punto de unión entre varias redes. Tiene especial importancia para la rapidez de las conexiones que el ordenador gestor de este punto sea una máquina potente y capaz de soportar un tráfico de comunicaciones intenso puesto que, de lo contrario, se produciría un "cuello de botella" en el mismo y, como consecuencia, importantes demoras. Otra forma de denominar a un dispositivo que tiene acceso a Internet. Es un equipo conectado a la red para ser utilizado por un usuario final. Este término se utiliza generalmente para referirse a una estación de trabajo dentro de una red. Punto computacional dentro de una red de comunicaciones.

96. *Objeto.* En la programación orientada a objetos, un objeto es un conjunto formado por un conjunto posiblemente vacío llamado datos miembro formado a su vez por un conjunto posiblemente vacío de variables y un conjunto posiblemente vacío de símbolos de constante y por un segundo conjunto posiblemente vació de funciones computables llamado funciones miembro.

97. *Par trenzado.* Cables conteniendo desde uno hasta varios cientos de pares trenzados se usan en miles de interconexiones *electrónicas* y telefónicas. Los cables de par trenzado tienen anchos de banda limitados en comparación con los *coaxiales* y la *fibra óptica.* Cable que se forma de dos alambres aislados, que se tuercen entre si. Existen dos variantes básicas blindado y no-blindado.

98. *Patrimonio.* Bienes heredados de los ascendientes. Conjunto de bienes, propiedades, derechos y obligaciones, pertenecientes a una misma persona natural o jurídica.

99. *Percepción.* Un proceso activo en el que se hacen hipótesis acerca de la naturaleza del entorno, o se contempla información sensorial para confirmar o refutar hipótesis.

100.*Perceptrón.* Es un ejemplo clásico de red neural. En este modelo una neurona tiene n entradas, cada una de ellas ponderada con pesos w i ; i = 1; : : : ; n, además de un término adicional llamado de predisposición.

101.*Plafón(es).* Plano inferior del saliente de una cornisa. Adorno en el centro del techo de una habitación donde está el soporte de la lámpara. Lámpara traslúcida y plana que se coloca pegada al techo o a una pared para ocultar las bombillas. Placa lisa que cubre una cosa o forma un falso techo. Cielo raso. Tablero rehundido.

102.*Plataformas.* La arquitectura del hardware de un modelo particular o familia de computadoras. La plataforma es el estándar con que los diseñadores de software escriben sus programas. El término a menudo se refiere al sistema operativo incluido con el hardware.

103.*Plotter.* Una impresora gráfica que dibuja imágenes con plumas de tinta. Los trazados requieren datos en formato de gráficos vectoriales, de manera que una imagen se compone de una serie de líneas de punto a punto.

104.*Pool.* El término viene del inglés "poll": sondeo. Es una forma de control en redes de comunicaciones del tipo LAN, según la cual la unidad central de proceso pide, de acuerdo con una programación determinada a cada puesto de trabajo conectado a la red, si ha de enviar alguno información. Literalmente, encuestamiento. Bajo esta técnica, un dispositivo atiende a varios a través de ir "revisando" cada uno de ellos, y verificar si tiene algo que recibir o transmitir.

105.*Premisa(s).* Una primera proposición sobre la que descansan los razonamientos subsiguientes. Cada una de las proposiciones del silogismo de donde se infiere la conclusión y que ya contiene en su interior los elementos necesarios para no caer en tautología. Indicio. Hecho del que se parte para llegar a una conclusión o valoración; supuesto, base de argumentación.

106.*Problemas NP-Completos.* El conjunto de problemas que tienen un veloz algoritmo de prueba, se denomina NP (el nombre NP proviene de una definición matemática del conjunto más precisa). Cualquier problema que sea uno de los más difíciles en NP se denomina NP-completo. Los problemas factibles tienen un algoritmo veloz para determinar sus soluciones. Los problemas en

NP pueden o no tener tal algoritmo pero al menos las soluciones propuestas son fáciles de verificar. Los problemas completos en NP son los problemas más difíciles en NP.

107. *Procesamiento de Lenguaje Natural.* Sistemas informáticos capaces de generar y *"entender"* un lenguaje natural.

108. *Protocolos.* En comunicaciones, un conjunto de normas y regulaciones que gobiernan la transmisión y recepción de datos. Conjunto de interconexiones, tanto de software como de hardware, que permiten establecer una red de comunicaciones en terminales y computadoras. Se denomina protocolo a un conjunto de normas y/o procedimientos para la transmisión de datos que ha de ser observado por los dos extremos de un proceso comunicacional (emisor y receptor). Estos protocolos "gobiernan" formatos, modos de acceso, secuencias temporales, etc. Son las normas que deben cumplir dos o más ordenadores para intercambiar mensajes entre sí. El protocolo describe tanto el formato de los mensajes como la forma de respuesta a cada uno de ellos. Descripción formal de formatos de mensaje y de reglas que dos ordenadores deben seguir para intercambiar dichos mensajes. Un protocolo puede describir detalles de bajo nivel de las interfaces máquina-a-máquina o intercambios de alto nivel entre programas de asignación de recursos. Conjunto de reglas convencionales, utilizado para comunicar dos dispositivos de la misma naturaleza.

109. *Radio frecuencia.* Frecuencia de radio. Rango de frecuencias electromagnéticas superior al de audio e inferior al de la luz visible. Todas las transmisiones radiales, desde las de radio AM hasta las de satélites, entran en este rango, que va desde los 30 KHz hasta los 300 GHz.

110. *Red(es).* Equipos de comunicación interconectados, a través de uno o varios caminos o medios de transmisión. Equipos de comunicación llamados nodos conocidos también como vértices o puntos que están interconectados a través de uno o varios medios de comunicación llamados segmentos, trayectorias o aristas. Una disposición de objetos que están interconectados.

111. *Redes neuronales.* Redes neurales. Conjunto de neuronas interconectadas de una manera concreta, y una neurona como la unidad básica de la red, que recibe múltiples entradas, elabora la información recibida y emite una única salida o respuesta que puede transmitirse a otras neuronas. En su origen, existe una leve inspiración en las neuronas biológicas.

112.*Redundancia.* Repetición innecesaria o inútil de un concepto. Exceso, abundancia excesiva.

113.*Refutación.* Un intento de probar la imposibilidad de una conclusión hipotetizada u objetivo. Argüir contra las razones de otro. Rechazar.

114.*Reglas de producción.* Una estructura modular de conocimientos que representa una parte sustancial de conocimientos, usualmente en forma de "si-entonces" (if-then) o en forma de "antecedente-consecuente". Expresión muy utilizada en sistemas expertos.

115.*Rentable. Rentabilidad.* Dícese de las inversiones de fondos que producen renta o beneficio suficiente. Dícese de todo lo que puede ser beneficioso.

116.*Representación del conocimiento.* La forma de la estructura de datos usada para organizar los conocimientos requeridos para un problema.

117.*Robot(s).* Son principalmente máquinas con manipuladores que pueden programarse fácilmente para hacer una gran variedad de tareas automáticamente.

118.Robótica. Arte y ciencia de la creación y empleo de robots.

119.*Scanner.* Dispositivo que lee texto, imágenes y códigos de barras. Los exploradores de texto y de código de barras reconocen las letras impresas y los códigos de barras y los convierten en código digital, tal como el ASCII. Los exploradores gráficos convierten una imagen impresa en una de video (gráficos por trama) sin reconocer el contenido real del texto.

120.*Secuenciamiento* de operaciones. Proceso de dividir un mensaje de datos, en piezas más pequeñas, para su transmisión.

121.*Sensor(es).* Dispositivo que mide o detecta una condición del mundo real, tal como el movimiento, el calor o la luz, y convierte la condición en una representación analógica o digital de la misma. Un sensor óptico detecta la intensidad o brillo de la luz, o la intensidad de rojo, verde y azul para sistemas en color.

122.*Serpentín.* Tubo de diámetro pequeño respecto a su longitud, curvado varias veces sobre sí mismo, que ocupa poco espacio y presenta una gran superficie de irradiación. Grabación en serpentina. Formato de grabación de cintas de pistas paralelas en el que los datos "reptan" de un lado a otro de pista a pista. Tubo en espiral, en hélice o acodado para facilitar el enfriamiento de la destilación en los alambiques. Antigua pieza de artillería. Serpentina, roca.

123.*Servicios.* Sistemas que se integran al caparazón y que generalmente son componentes tecnológicos, cuyo ciclo de vida es de entre 15 y 20 años, tales como: sistema eléctrico, aire acondicionado, sistema hidráulico y sanitarios, elevadores y escaleras eléctricas, telecomunicaciones e *informática,* sistemas de control y seguridad, etcétera.

124.*Servo.* Elemento que entra a formar parte de palabras propias de la mecánica y que designa sistemas auxiliares: Servofreno, servomotor. Dispositivo electromecánico que emplea retroalimentación para proveer señales precisas de arranque y detención para funciones tales como las de los motores de una unidad de cinta o el movimiento de un brazo de acceso sobre un disco.

125.*Sinapsis.* Es el lugar donde se produce la unión dando lugar a la diferenciación entre la célula presináptica que emite la información y la célula postsináptica o de salida.

126.*Sistema Experto.* Sistemas que proporcionan a los usuarios humanos conclusiones técnicas sobre materias especializadas.

127.*Sistema Inteligente (SI).* Controla y coordina los diferentes recursos de un edificio. Conjunto de elementos interrelacionados, en donde se combina la tecnología, los conocimientos y métodos de inferencia propios de la Inteligencia Artificial que tienen como objetivo el realizar acciones que implican un *comportamiento inteligente.*

128.*Sistemas abiertos.* Un sistema independiente del fabricante que está diseñado para interconectarse con una variedad de productos comúnmente disponibles. Implica que los estándares para tal sistema están determinados a partir de un consenso de las partes interesadas, más que de uno o dos fabricantes solamente.

129. *Sistemas conexionistas.* Sistemas que se orientan a la construcción de sistemas inteligentes simulando por medio de redes neuronales la actividad del cerebro humano, máquinas con capacidad de aprendizaje, intenta implementar la capacidad de conocimeintos inciertos distribuidos globalmente, controlado de modo difuso y con capacidad de aprendizaje. Procesamiento de la información globalmente distribuida e interconectada.

130. *Sistemas Distribuidos.* Un sistema de computación diseñado para múltiples usuarios que proporciona a cada uno una computadora funcionalmente completa.

131. *Sistemas informáticos.* Un sistema formado por una CPU, todos los dispositivos periféricos conectados a ella y el sistema operativo. Los sistemas informáticos pueden englobarse en categorías llamadas microcomputadoras (computadoras personales), minicomputadoras y macrocomputadoras, es decir (aproximadamente) pequeñas, medianas y grandes.

132. *Sistemas inteligentes.* Conjunto de elementos interrelacionados, en donde se combina la *tecnología,* los conocimientos y métodos de *inferencia* propios de la *Inteligencia Artificial* que tienen como objetivo el realizar acciones que implican un *comportamiento inteligente.*

133. *Telecomunicaciones.* Comunicación de todas las formas de información, incluidas la voz, (el audio) y el video. La transferencia *electrónica* de información de un lugar a otro. Las comunicaciones de datos se refieren a las transmisiones digitales, y las telecomunicaciones se refieren a todas las formas de transmisión, incluyendo voz y video *analógico.*

134. *Telefonía avanzada.* Es telefonía digital. Posteriormente se ofrecerían servicios como: telefonía avanzada, música digital, información y teletramitación, pago por visión (PPV), videojuegos a distancia, telecompra, telebanca, TV a la carta, vídeo bajo demanda (VOD), etc...

135. *Telepuertos.* Tiene como objetivo principal el brindar comunicación satelital conjuntar las mejores características del satélite y la fibra óptica. Para poner las comunicaciones satelitales al alcance de todas las empresas ubicadas en las principales ciudades del país, se emplazan los telepuertos

136. *Telex.* Telegrafía por teletipo cuando se transmiten las señales aprovechando las corrientes portadoras de las líneas telefónicas.

137.*Teorema(s)*. Proposición que afirma una verdad que se puede demostrar.

138.*Termostato(s)*. Aparato conectado a una fuente de calor que impide que la temperatura suba o baje del grado conveniente. Regulador que permite conservar una temperatura sensiblemente constante en el interior de un recinto.

139.*Tipos de Datos Abstractos*. Unidad de programación que incluye un tipo de dato y un conjunto asociado de operaciones.

140.*Transact - SQL*. Es la versión mejorada de Sybase del lenguaje de base de datos SQL. Las aplicaciones lo usan para comunicarse con el SQL Server. Provee comandos para crear y manipular objetos de la base de datos, así como para insertar, actualizar y consultar los datos. Las mejoras de Sybase incluyen aspectos de integridad de datos y procedimientos almacenados.

141.*UPS*. Uninterrumpible Power Supply. Suministro ininterrumpible de energía. Fuente de alimentación ininterrumpible. Energía de seguridad para un sistema de computación cuando la energía eléctrica de la línea se interrumpe o baja a un nivel de tensión inaceptable. Los pequeños sistemas UPS proveen energía de baterías por sólo unos pocos minutos; los necesarios para apagar la computadora de manera ordenada. Los sistemas más sofisticados están conectados a *generadores* eléctricos y pueden proveer energía durante días enteros.

142.*Vía(s)*. Cada uno de los carriles del ferrocarril o conjunto de los dos y terreno en que se asientan. Camino; hoy se utiliza con el sentido de ruta o pasando por... Conducto del cuerpo humano. Procedimiento, modo.

143.*Videoconferencia*. Una video teleconferencia es una video conferencia realizada entre varios usuarios mediante cámaras y monitores de videos ubicados en las instalaciones del cliente o en un centro de conferencias público. Una video teleconferencia requiere una red de comunicaciones propia, que emplea *cable coaxial*, fibras ópticas, transmisiones por microondas o satélites, ya que las redes informáticas convencionales no pueden manipular señales de video. La utilización de infraestructura de conferencias remotas que incluyen señales de video.

144. *Videotexto.* Una tecnología de información interactiva que incluye compras, bancos, noticias, meteorología y servicios de correo electrónico. Puede además proporcionar una puerta de acceso a otros servicios de tiempo compartido y de información. Se transmite sobre una línea telefónica a un decodificador, que contiene un teclado adosado al T.V. del abonado.

145. *Visión.* Acción y efecto de ver. Contemplación inmediata y directa de los objetos. Representación fantástica o ilusoria que se toma por real.

BIBLIOGRAFÍA

[Alcalde88] Alcalde E. y M. García; Informática Básica; Ed. McGraw-Hill; México 1988.

[Anderson89] Anderson David; Artificial Intelligence and Intelligent Systems; Ed. Ellis Horwood Limited; England 1989.

[Arbib72] Arbib M.; "Toward an automata theory of brains"; CACM; Julio 1972.

[Baker95] Baker Brad y Setrag Khofhafian; "Edificios inteligentes"; Madrid 1995.

[Barr89] Barr Avron y Edward A Feigenbaum; The handbook of Artificial Intelligence Volume 1; Ed. Addison-Wesley; USA 1989.

[Bellman78] Bellman, R.E.; An introduction to artificial Intelligence: Can computers Think?; Ed. Boyd & Fraser Publishing Co.; 1st ed.; San Francisco; 1978.

[Bernaden88] Bernaden, John y Neubauer Richard; The intelligent building sourcebook; Ed. The Fairmont Press; 1ª. ed. ; New York; 1988.

[Berson95] Berson Alex y George Anderson; Sybase and Client/Server Computing; Ed. McGraw-Hill; 1ª ed.; New York; 1995

[Booker89] Booker, L.B.; D.E. Goldberg y J.H. Holland; "Classifier Systems and Genetic Algorithms"; Ed. Artificial Intelligence; 1989.

[Buckles92] Buckles, B.P. y F.E. Petry; "Genetic Algorithms"; IEEE Computer Society Press; 1992; 109 p.

[Charniak86] Charniak Eugene y Drew McDermott; Introduction to Artificial Intelligence; Ed. Addison-Wesley; ; 1st ed.; Massachusetts; 1985.

[Davis91] Davis, L.; "Handbook of Genetic Algorithms"; Ed. Van Nostrand Reinhold; 1991, 385 p.

[DBLIB90] Open Client DB-Library/C Reference Manual; Ed. Sybase Inc.; 1990

[Dehn] Dehn N. y R. Shank; "Artificial and Human intelligence"; en Handbook of Human Intelligence; R. Stenberg (editor); Cambridge University Press.

[EDIFINTEL94] Edifintel 1994: Conferencias y exposición; Instituto Mexicano del Edificio Inteligente A.C.; México; 1994.

[EdiInt93] Edificios Inteligentes: Altas tecnologías; Ciclo de conferencias en la Facultad de Arquitectura de la UNAM; México; 1993.

[Freeman91] Freeman J y Skapura D; "Neural Networks"; Addison-Wesley; 1991.

[Genesereth87] Genesereth M. y N. Nilsson; "Logical Foundations of Artificial Intelligence"; Morgan Kaufmann; 1987.

[Gevantes] Gevartes William M.; Máquinas Inteligentes; Ed. Ediciones Díaz De Santos S.A.

[Goldberg89] Goldberg, D.E.; "Genetic Algorithms in Search, Optimization and Machine Learning"; Ed. Addison - Wesley Publishing Company; 1989.

[Hazlehurst96] Hazlehurst, Peter; Using sybase system XI; 1ª ed.; Indianapolis; 1996.

[Haugeland85] Haugeland, J.; Artificial intelligence: the very idea; Ed. MIT Press; 1st ed.; Cambridge; 1985.

[Holland75] Holland, J.H.; Adaptation in Natural and Artificial Systems"; Ed. University of Michigan Press; 1975.

[IBI90] Intelligent Buildings Institute; Washington D.C. 1990

[IBS88] The Ingelligent Building Sourcebook; Ed. Prentice - Hall; New Jersey; 1988.

[Kandel81] Kandel E. y Schwartz J. ; Principles of Neuronal Science; Ed. Elsevier North Holland; 1981.

[Kaufmann77] Kaufmann, A. y R. Faure; Invitación a la Investigación de Operaciones; Ed. CECSA; 2ª ed.; México; 1977.

[Kim90] Kim Steven H.; Designing Intelligence; Ed. Oxford University Press; USA 1990

[Koza92] Koza, J.R.; Genetic Programming. On the Programming of Computers by Means of Natural Selection; The MIT Press; 1992.

[Koza92a] Koza, J.; Genetic Programming II : Automatic Discovery of Reusable Programs; The MIT Press; 1992.

[Kurzweil90] Kurzwil, R.; The age of intelligent machines; Ed. MIT Press; 1st ed.; Cambridge; 1990.

[Lidec88] Lidec, Philips; Manual de Alumbrado; Ed. Paraninfo; 1ª ed. ; Madrid; 1988

[Lipschutz83] Lipschutz, Seymour; Matemáticas para computación; Ed. McGraw-Hill; 1ª ed.; México; 1983.

[Luger93] Luger, G.F.; Artificial Intelligence: Structures and Strategies for complex problem solving; Ed. Benjamin - Cummings; 2nd ed.; Redwood City; 1993.

[Marion94] Marion, William; Client/Server Strategies; Ed. McGraw-Hill; 1st ed.; NewYork; 1994.

[McCune90] McCune W; "Otter 2.0 Users Guide", Argonne National Laboratory; 1990.

[Michalski83] Michalski R, Carbonell J, Mitchell T. ; Machine Learning: An Artificial Intelligence Approach; Ed. Morgan Kaufmann; 1983.

[Ñeco97] Ñeco García Ramón P.; Uso de redes neurales recurrentes para la inferencia de traductores formales ; Ed. Universidad de Alicante; 1997.

[Nilsson87] Nils J. Nilsson; Principios de Inteligencia Artificial; Ed. Díaz de Santos; Madrid 1987.

[Rawlins91] Rawlins, G.J.E.; Foundations of Genetic Algorithms; Ed. Morgan Kaufmann Publishers; 1991.

[RAEEE92] Recomendaciones para el ahorro de energía eléctrica en edificios; Fideicomiso de apoyo al programa de energía del sector eléctrico; México; 1992.

[Rich94] Rich, E. y Kevin Knight; Inteligencia Artificial; Ed. McGraw - Hill; 2ª ed.; Madrid; 1994.

[Russell96] Russell, Stuart y Peter Norving; Inteligencia Artificial: un enfoque moderno; Ed. Prentice-Hall; 1ª ed.; México; 1996.

[Schildt90] Schildt, Herbert; Utilización de C en Inteligencia Artificial; Ed. McGraw-Hill; 1ª ed.; México; 1990.

[Schalkoff90] Schalkoff, R.J.; Artificial Intelligence: an engineering approach; Ed. McGraw - Hill; 1st ed. ; New York; 1990.

[SHL94] SHL SYSTEMHOUSE. Glosario de Términos. Conectividad y Arquitectura Cliente-Servidor. Edición 1994.

[Sterling86] Sterling L, Shapiro E; The Art of Prolog; Ed. MIT press; 1986.

[Stroustrup93] Stroustrup, Bjarne; El lenguaje de programación C++; Ed. Addison-Wesley; 2ª ed.; México; 1993.

[Taboada83] Taboada, J.A.; Manual de Luminotecnia; Ed. Dossat; 1ª ed.; Madrid; 1983.

[Tracy97] Tracy Kim y Peter Bouthoorn; Object-Oriented Artificial Intelligence Using C++; Ed. Computer Science Press; 1st ed.; New York, 1997.

[Vaughn94] Vaughn, Larry T.; Client/Server System Design & Implementation; Ed. McGraw-Hill; 1st ed.; New York; 1994.

[Whitley93] Whitley, L.D.; Foundations of Genetic Algorithms 2; Morgan Kaufmann Publishers; 1993.

[Winston84] Winston P; Artificial Intelligence; Addison-Wesley; 1984

[Winston92] Winston, P. H.; Artificial Intelligence; Ed.Addison - Wesley; 3rd ed.; Massachusetts; 1992.

HEMEROGRAFÍA

[Zozaya94] Zozaya, C.; "Arquitecturas de Sistemas Expertos"; Soluciones Avanzadas; Año 2, No. 13; Septiembre 1994; pp.41-47.

[Forrest93] Forrest, S.; "Genetic Algorithms: Principles of Natural Selection Applied to Computation" ; Science; Vol. 261; No. 5123; August 13, 1993; pp. 872-878.

[Epstein92] Epstein R; "Can Machines Think?"; AI Magazine; Vol.13; No.2; 1992.

[Hayes-Roth87] Hayes-Roth F; "Expert Systems"; Encyclopedia of artificial intelligence; S. Shapiro; Vol. 1; John Wiley; 1987.

[Hudak89] Hudak P; "Conception, Evolution, and Application of Functional Programming Languages"; ACM Computing Surveys; Vol.21; No.3; 1989.

[Maida87] Maida A; "Frame Theory"; Encyclopedia of artificial intelligence; S. Shapiro ; Vol. 1; John Wiley, 1987.

[McCarthy60] McCarthy J; "Recursive Functions of Symbolic Expressions and their computation by Machin, part I"; CACM; Vol.3; No.4; 1960.

[Porter88] Porter, K.; "Handling Huge Arrays"; Dr. Dobb's Journal of Software Tools for the Professional Programmer; Vol. 3; No. 3; 1988;pp. 60-3.

[Ribeiro94] Ribeiro Filho, J.L.; Treleaven, Ph.C., and Alippi, C.; "Genetic-Algorithm Programming Environments"; IEEE Computer; June 1994; pp. 28-43.

[Samuel59] Samuel A; "Some Studies in Machine Learning Using the Gane of Checkers"; IBM Journal of Research and Development; No.3; 1959.

[Smith90] Smith D; "KIDS: A Semi-Automatic Program Development System"; IEEE Transactions on Software Engineering: Special Issue on Formal Methods; September 1990.

[Smith91] Smith, R.E.; D.E. Goldberg y J.A. Earickson; "SGA-C : A C-language Implementation of a Simple Genetic Algorithm"; TCGA Report No. 91002; The Clearinghouse for Genetic Algorithms; The University of Alabama; May 14, 1991.

[Srinivas94] Srinivas, M., and Patnaik, L.M.; "Genetic Algorithms : A Survey"; IEEE Computer; June 1994; pp. 17-26.

[Turing50] Turing A; "Computing Machinery and Intelligence"; Mind Journal; Vol.LIX; No.236; October 1950.