Título: Sistema Informático para la Solución a Problemas de Redes a través del Algoritmo de las Margaritas.

Trabajo de Diploma para optar por el título de Ingeniería Informática.

Autor: Adrián Geroy Pérez.

Tutores: Dr. Manuel E. Cortés Cortés.
Ing. Raúl René Alpízar Gamboa.

Cienfuegos, Cuba.
Curso 2011 – 2012.

Agradecimientos.

- *A mi madre que siempre me ha apoyado en todo, me ha sabido comprender y me ha dado fuerzas para seguir adelante, por el gran corazón que tiene.*
- *A mi papá que me aconseja siempre y por ser mi amigo.*
- *A mi viejitos Mirtha y Carlos, por quererme tanto.*
- *A mi tía y mi hermana Betty por aguantarme siempre.*
- *A mi primo Isdel por ser un ejemplo a seguir.*
- *A mi hermano Dairo, por apoyarme.*
- *A mis amigos de cuarto por soportarme durante cinco años.*
- *A Yoelvys, Lázaro, Freddy y Carlos, por la amistad sincera que siempre me han brindado.*
- *A mis tutores el Raure y Manolo, por el tiempo y la paciencia que supieron brindarme.*
- *A Yunieski por brindarme su ayuda con el diseño del sistema*
- *A Lianny, Juan Geisder, el Yasma y a todos los amigos del barrio.*
- *A todas las personas que de una forma u otra contribuyeron al desarrollo de este trabajo.*
- *A mis profesores de la universidad, por brindarme todos sus conocimientos.*

Dedicatoria.

-A mis padres.

-A mis abuelos que ya no se encuentran a mi lado, pero sí en mi corazón.

-A mi familia toda.

Resumen.

El presente trabajo de diploma presenta como título “Sistema Informático para la Solución a Problemas de Redes a través del Algoritmo de las Margaritas” y ha sido realizado con el objetivo de elaborar un producto informático que resuelva el algoritmo de las Margaritas, presente en la teoría de redes.

El producto anteriormente mencionado permite una buena interacción pues posee una interfaz amigable para los usuarios y es de fácil manejo. Posibilita resolver tubularmente dicho algoritmo, analizar los detalles del mismo si son de interés conocerlos por parte del usuario y a la vez constituye un producto informático que posee un algoritmo no incluido en los paquetes informáticos que abordan el tema de la teoría de redes.

Para su elaboración se utilizó el Lenguaje Unificado de Modelado (UML) para el análisis, diseño e implementación de la solución propuesta, siguiendo lo establecido por el Proceso Unificado de Desarrollo de Software (RUP). Para la implementación de la aplicación se utilizó la herramienta Borland C++ Builder6 utilizando el lenguaje de programación orientado a objetos C++.

Abstract.

Índice

Índice de Figuras

Índice de Tablas

Introducción.

La Investigación de Operaciones es una rama de las Matemáticas consistente en el uso de modelos matemáticos, estadística y algoritmos con objeto de realizar un proceso de toma de decisiones prácticas para negocios y otras áreas. A su vez se orienta a la resolución de problemas relacionados con la conducción y coordinación de las operaciones o actividades dentro de una organización. Su ámbito de aplicación es muy amplio, aplicándose a problemas de fabricación, transporte, construcción, telecomunicaciones, planificación y gestión financiera, ciencias de la salud, servicios públicos, etc. En general, puede aplicarse en todos los problemas relacionados con la gestión, la planificación y el diseño [1]. Estos problemas pueden ser, por ejemplo, la repartición de recursos para maximizar ingresos, o agendar actividades para minimizar riesgos. Técnicas propias de la Investigación de Operaciones incluyen programación lineal y otras áreas de optimización, teoría de colas, algoritmos de planificación y análisis de redes. La Investigación de Operaciones también incluye tópicos continuos como procesos de Markov de tiempo continuo, optimización de procesos y martingalas de tiempo continuo [2]. La Investigación de Operaciones incluye un conjunto muy amplio de técnicas orientadas a proporcionar una ayuda cuantitativa a la toma de decisiones. El método empleado es el método científico, y las técnicas que se utilizan son, en buena medida, técnicas matemáticas [1].

La Investigación de Operaciones también comprende la Teoría de Redes, mediante la misma se han resuelto exitosamente muchos problemas administrativos de diseño de sistema de transporte, sistemas de información y programación de proyectos, con ayuda de los modelos de redes y con técnicas de análisis de redes. Entre los adelantos en el campo de la Investigación de Operaciones puede decirse que la teoría de redes está entre las más simples y elegantes que poseen una amplia variedad de aplicaciones. La estructura topológica de las redes puede ser representada por un gráfico con vértices o nodos y ramas o arcos, correspondientes a las estaciones y enlaces respectivamente[3].

Los modelos de redes de actividades sirven para planear, programar y controlar proyectos que constan de numerosos trabajos o tareas separadas que son llevadas a cabo por diversos departamentos, personas, etcétera. Con frecuencia, estos proyectos son tan grandes y/o tan complejos que no es posible que un administrador tenga en mente toda la información relativa al plan, al programa y al avance de su proyecto. En estas situaciones, las técnicas denominadas PERT (Program Evaluation and Review Technique) (o sea, Técnica de evaluación y revisión de programa) Y CPM (Critical Path Method) (o sea, Método de la Ruta Crítica) han demostrado ser extremadamente valiosas para ayudar a los ejecutivos en la toma de decisiones relacionada con los proyectos[3].

TSP.

Una gran cantidad de problemas importantes de optimización no pueden ser resueltos usando métodos exactos, es decir, no es posible encontrar su solución óptima con esfuerzos computacionales aceptables aunque se pueda contar con computadores de alta velocidad operando en paralelo. Un gran problema de la optimización es el fenómeno llamado explosión combinatorial, que significa, que cuando crece el número de variables de decisión del problema, el número de decisiones factibles y el esfuerzo computacional crecen en forma exponencial. Sin embargo, no todos los problemas combinatoriales son tan complejos de resolver; existen algunos para los cuales hay algoritmos que resuelven esos problemas con un esfuerzo computacional que crece de manera polinomial con el tamaño del problema. Enmarcado dentro de estos problemas combinatoriales se encuentra el problema del Viajante Vendedor (TSP). Este problema consiste en encontrar el camino más corto (ruta) para visitar n ciudades bajo la condición de visitar cada ciudad una sola vez, regresando a la ciudad de partida. Desde la década de los 50`s muchos algoritmos han sido desarrollados para encontrar la solución a este problema encontrando buenas soluciones pero no necesariamente las soluciones óptimas[4].

TIC.

Las tecnologías de la información y las comunicaciones (TIC) tienen, día a día, una mayor presencia en todos los aspectos de la vida laboral y personal, ofreciendo un nuevo espacio de innovación en ámbitos como la industria, los servicios, la salud, la administración, el comercio y la educación[5].

El desarrollo que aportan las TIC transforma poderosamente los modos de vida y la actividad laboral y profesional. En el mundo contemporáneo, la utilización de las Tecnologías de la Información y la Comunicación (TIC), ha impulsado de forma acelerada el desarrollo científico-técnico de los países, en la industria, la economía, la salud y la educación, aún cuando estos avances tecnológicos sigan siendo un privilegio de los países del primer mundo o desarrollados. Las TIC potencian la generalización del aprendizaje a lo largo de toda la vida al hacer de la educación un proceso más fácil, individualizado y asequible y contribuyen de manera ostensible a la generalización del conocimiento. Cierto es que han permitido la transformación de los programas de formación de recursos humanos, han diversificado la educación y han surgido nuevos escenarios docentes[6].

Como medios de enseñanza las TIC han servido de apoyo para aumentar la efectividad del trabajo del profesor sin llegar a sustituir la función educativa y humana del maestro, así como para racionalizar la carga de trabajo de los estudiantes y el tiempo necesario para su formación científica y sirven además para elevar la motivación de la enseñanza y el aprendizaje[6].

Los países desarrollados evolucionan aceleradamente hacia economías de información y en los países menos desarrollados se hace imprescindible disponer de una estrategia nacional para la utilización de las TIC como herramientas de desarrollo económico, social y cultural. Para los países subdesarrollados este proceso es más complejo. La evolución hacia una economía de información está sustentada en el desarrollo acelerado de una infraestructura tecnológica, compuesta por los más avanzados recursos informativos y de comunicación, sólo posible en industrias altamente desarrolladas[6].

El avance a mediano y largo plazo que proporciona la casi libre disposición del fondo de conocimiento de la humanidad sin las restricciones del Copyright es un logro inaudito para la historia de la humanidad. Si bien es cierto que el caudal de información es inagotable, que se actualiza cada segundo, diariamente, también lo es que el tiempo no alcanza para procesar toda la información que puede ser obtenida, por lo que se hace imprescindible fomentar mecanismos de recuperación cada vez más eficientes y pertinentes, con el fin de lograr que la información recuperada tenga un alto valor de uso[6].

La integración de las TIC, agiliza los flujos de datos, comunicación y materiales en el seno de la red de empresas, da lugar a mejoras importantes, entre las que podemos citar: la reducción de los niveles de inventario de materias primas, productos en curso y productos terminados, con la consiguiente disminución de necesidades financieras, o los menores tiempos de proceso y de suministro y el superior rendimiento de los equipos y programas de producción, los cuales trascienden más allá de los límites organizativos al acelerar la colocación de nuevos productos en el mercado, mejorar el tiempo y las condiciones de entrega de la mercancía, ofrecer niveles superiores y consistentes de calidad y un mejor servicio a clientes. La superior flexibilidad de los equipos (técnicos y humanos) y, consiguientemente, la mayor capacidad de respuesta de la entidad conducen en definitiva a un aumento de la productividad de las empresas[7].

Las economías modernas se han venido caracterizando en los últimos tiempos por el aumento de las inversiones que han realizado las empresas en las TIC. Se hace evidente que la utilización de las mismas es una piedra angular para lograr el avance y la vida de las organizaciones, pues son un elemento esencial que contribuye a un desarrollo eficaz.

La relación existente entre los Modelos Matemáticos de Redes y los Sistemas Informáticos goza de gran fortaleza, desde el punto de vista de optimización tanto en la representación como en la descripción y muy justificado, por la diversidad de software y paquetes, existentes de este tipo.

Situación Problémica.

La aplicación de problemas TSP, específicamente mediante el método de las margaritas, no se explota debidamente de acuerdo al grado de optimización que presenta, principalmente por la complejidad que presenta el mismo para usuarios que no posean un conocimiento avanzado en Teoría de Redes. El trabajo manual al implementar este algoritmo es muy complejo y engorroso, aun para aquellos que dominen su práctica. En la actualidad no existe ningún producto informático, dada la bibliografía revisada, que desarrolle o incluya el algoritmo de las margaritas.

Problema de Investigación.

Por lo anteriormente expuesto se hace necesario el desarrollo de un producto informático que sea amigable a los usuarios, tanto estudiantes como investigadores, que implemente el algoritmo anteriormente mencionado y que resuelva los problemas de la Investigación de operaciones de una manera más eficiente.

Preguntas de Investigación.

- ¿El diseño de algoritmos matemáticos y computacionales de Teoría de Redes, satisface las necesidades que exige hoy la resolución de problemas de la Disciplina de Investigación de Operaciones?
- ¿Se seleccionará el algoritmo matemático más usual para la implementación del sistema informático en la solución de las margaritas?
- ¿El sistema computacional desarrollado permitirá de una forma más clara la apropiación del conocimiento por parte de los estudiantes e investigadores para dar solución al algoritmo de las margaritas?

Objeto de Estudio.

La Teoría de Redes en la Investigación de Operaciones.

Campo de Acción.

El Algoritmo de las Margaritas en la Teoría de Redes.

Objetivo General.

Elaborar un producto informático que resuelva el algoritmo de las Margaritas.

Objetivos Particulares.

1. Realizar el análisis del algoritmo matemático propuesto.
2. Diseñar el producto informático propuesto.
3. Implementar el producto informático propuesto.
4. Validar el producto informático propuesto.

Tareas de Investigación.

- Elaboración de la propuesta para la solución de problemas de Redes mediante el algoritmo de las Margaritas.
- Estudio de la solución matemática del algoritmo desarrollado.
- Análisis a profundidad de las herramientas que se aplican actualmente en la solución de problemas de Redes.
- Revisión y análisis de la bibliografía contemporánea para caracterizar el estado actual de la problemática planteada.
- Identificación y análisis de la aplicabilidad, ventajas y desventajas de algoritmos existente para la solución de problemas de Redes.
- Definición de los requerimientos funcionales y no funcionales.
- Estudio y análisis de los costos y beneficios que tiene la puesta en marcha de la solución propuesta.
- Aplicación de encuestas a estudiantes con dominio de la Teoría de Redes.
- Procesamiento de las encuestas aplicadas haciendo uso de un programa estadístico.

Idea a defender.

Con la elaboración de un producto informático que implemente el Algoritmo de las Margaritas, se podrán hacer aplicaciones del mismo a problemas de Redes y se contribuirá a la satisfacción de las exigencias actuales de la resolución de problemas en esa temática de la Investigación de Operaciones.

Justificación.

Luego de haberse consultado todos los paquetes de programas que resuelven redes, conocidos por el autor: Qsb, POM, Qmwin2 y TORA, se ha podido identificar que los mismos no presentan una solución en el trabajo con los problemas de transporte mediante el método de las margaritas. El aprendizaje de estos se torna demasiado complejo para realizar operaciones sencillas y de poco grado de dificultad.

Métodos del nivel teórico.

Análisis Histórico – Lógico: Para conocer y analizar el desarrollo de la Investigación de Operaciones y la Teoría de Redes en su decurso histórico.

Análisis – Síntesis: Para analizar toda la información relacionada con los aspectos abordados en la presente investigación y poder realizar el sistema informático.

Hipotético – Deductivo: Para realizar las generalizaciones y obtener el desarrollo de los modelos matemáticos.

Modelación Matemática: Para obtener el Modelo Teórico del algoritmo de las Margaritas.

La presente investigación persigue minimizar la cantidad de rutas maximizando el coste de cada una. **Min (**Cantidad de rutas**) Max (**Coste de cada ruta**)**

Métodos empíricos.

El análisis de documentos, para la revisión de los modelos y métodos aplicables a la temática en cuestión.

El Algoritmo de las Margaritas para la Solución del Problema.

Métodos Estadísticos.

Aplicación de Encuestas sobre la Validación del Producto Informático.

Utilización de paquetes de programas: SSPS 15.0. Para el procesamiento estadístico y el WinQsb para la solución del Algoritmo TSP.

Para el adecuado análisis y entendimiento de este documento, se ha estructurado el mismo de la siguiente forma:

- Resumen.
- Introducción.
- Tres capítulos.
- Conclusiones generales.
- Recomendaciones.
- Referencias bibliográficas.
- Anexos.

- **Capítulo I.-** "Fundamentación teórica":

  En este capítulo se exponen y detallan las características, conceptos básicos y enfoques en el desarrollo del software; se describen los sistemas ya existentes asociados a la Teoría de Redes y se analizan lenguajes y metodologías aplicadas en el desarrollo del sistema.

- **Capítulo II.-** "Descripción y construcción de la solución propuesta":

  En este capítulo se seleccionó como punto de partida el modelo del dominio. También se describe la solución propuesta utilizando algunos de los artefactos que propone la Metodología RUP. Los artefactos referidos son: los requerimientos funcionales y no funcionales, el diagrama de casos de uso, la descripción de cada uno, diagrama de clases del diseño y diagrama de clases persistentes. Se incluyen además los principios de diseño.

- **Capítulo III.-** “Estudio de factibilidad y validación del sistema”:

  En este capítulo se estima el esfuerzo humano, el tiempo de desarrollo y también el costo para la realización del producto informático, determinándose así si es factible o no su desarrollo. También se validará el producto basándose en el análisis de los resultados obtenidos de las encuestas realizadas a los estudiantes que han cursado la asignatura de Investigación de Operaciones II. Se utilizará la prueba de hipótesis para la comparación de los resultados, apreciando así si existe acuerdo o no entre los encuestados.

Capítulo 1. Fundamentación Teórica.

1.1 - Introducción.

En este capítulo se muestran y explican las características, conceptos básicos y enfoques en el desarrollo del software; se describen los sistemas ya existentes asociados a la Teoría de Redes y se analizan lenguajes y metodologías aplicadas en el desarrollo del sistema.

1.2- Descripción del dominio del problema.

1.2.1– Sistema informático.

Un sistema informático es definido como un sistema de información que basa la parte fundamental de su procesamiento, en el empleo de la computación, como cualquier sistema, es un conjunto de funciones interrelacionadas, hardware, software y de Recurso Humano. Un sistema informático normal emplea un sistema que usa dispositivos que se usan para programar y almacenar programas y datos[8]. Un ejemplo clásico de sistema informático lo representa una computadora personal, junto con la persona que la opera y los periféricos que los envuelven.

1.2.2– Teoría de Redes.

En matemáticas y en ciencias de la computación, la teoría de redes (también llamada teoría de grafos) estudia las propiedades de los grafos. Un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas que pueden ser orientados o no[9].

El trabajo sobre el problema de los puentes de Königsberg es considerado el primer resultado de la teoría de grafos, realizado por Leonhard Euler, en 1736.

También un grafo es una terna **G = (V, A, j),** en donde **V** y **A** son conjuntos finitos, y **j** es una aplicación que hace corresponder a cada

elemento de **A** un par de elementos de **V**. Los elementos de **V** y de **A** se llaman, respectivamente, **"vértices"** y **"aristas"** de **G**, y **j** asocia entonces a cada arista con sus dos vértices[9].

Esta definición da lugar a una representación gráfica, en donde cada vértice es un punto del plano, y cada arista es una línea que une a sus dos vértices, como se muestra en la Figura I.

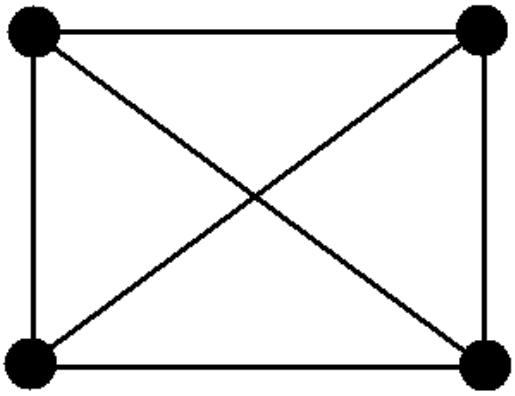


Figura 1. Representación Visual de un Grafo

También se pueden encontrar otras formas de representar un grafo como por ejemplo:

- La representación mediante matriz asociada o de adyacentes, especialmente útil para el tratamiento de problemas de grafos con programas informáticos[10].
- Otras representaciones, como el diccionario de grafo, buscan definir el grafo de una forma más compacta, en términos de posiciones de memoria. Pueden ser útiles para representar grafos de gran tamaño[10].

En la actualidad es escasa la disciplina científica o humanística que no utiliza la teoría de grafos. Como ejemplos se pueden citar la física teórica, que usa los diagramas de Feynmann, la psicología en dinámica de grupos, donde se representan mediante líneas las partículas elementales, los cambios de variable en el cálculo diferencial, el estudio de flujos en redes en programación lineal e investigación operativa, entre otros.

TIPOS DE GRAFOS

Existen dos tipos de grafos los no dirigidos y los dirigidos.

- **NO DIRIGIDOS:** son aquellos en los cuales los lados no están orientados. Cada lado se representa entre paréntesis, separando sus vértices por comas, y teniendo en cuenta **$(V_i, Vj) = (V_j, Vi)$**. Figura II[11].

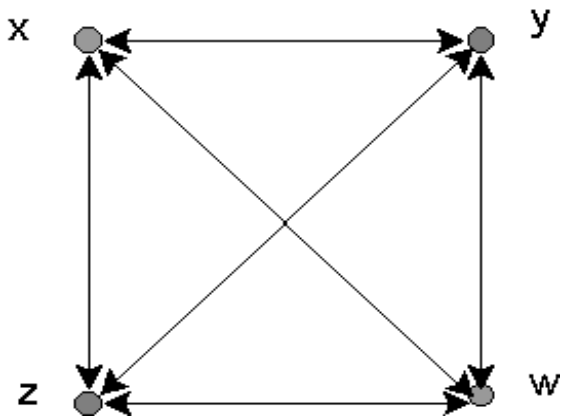


Figura 2. Grafo No Dirigido

- **DIRIGIDOS:** son aquellos en los cuales los lados están orientados. Cada lado se representa entre ángulos, separando sus vértices por comas y teniendo en cuenta **<V_i, V_j>!=<V_j, V_i>**. En grafos dirigidos, para cada lado **<X, Y>**, **X**, el cual es el vértice origen, se conoce como la cola del lado y **B,** el cual es el vértice destino, se conoce como cabeza del lado. Figura III[12].

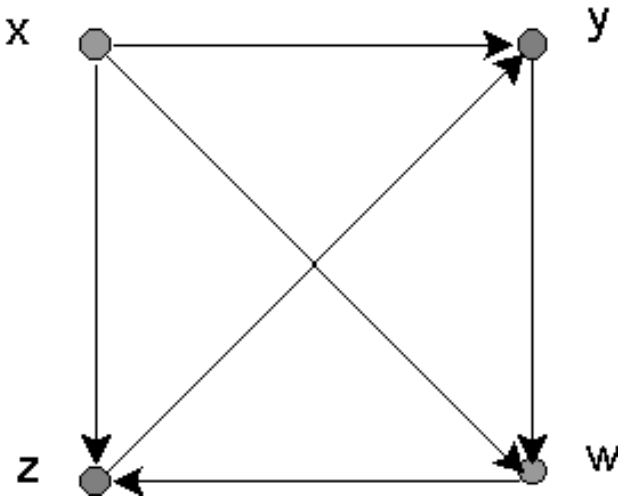


Figura 3. Grafo Dirigido

Terminología de grafos

ADYACENCIA: Para no dirigidos se dice que dos vértices son adyacentes si forman un lado. En dirigidos se extiende el concepto: sea el lado **<V_i, V_j>:** se dice **V_j es adyacente desde V_i** y **V_i es adyacente hacia V_j** [13].

INCIDENCIA: se dice que un lado es incidente sobre el par de vértices que lo conforman[13].

Sea un arco **a** que pertenece a **A** y un vértice **x** que pertenece a **V** de una Red **G = (V, A)**

Se dice que el arco **a** es incidente en el vértice **x** hacia el exterior si **x** es la extremidad inicial de **a** y si la extremidad terminal de **a** es diferente de **x**.

Similarmente se enuncia que el arco **a** es incidente en **x** hacia el interior si la extremidad terminal de **a** es **x** y si la extremidad inicial de **a** es diferente de **x**[14].

GRADO DE UN VÉRTICE (NODO): es el número de lados incidentes sobre él**.**

Para determinar el grado total de un grafo dirigido se suma el **grado entrante** que es el número de lados que llegan al nodo y el grado saliente que es el número de lados que salen del nodo

TRAYECTORIA

Es una secuencia de lados para ir desde $\mathbf{V_i}$ a $\mathbf{V_j}$, en la cual cada pareja consecutiva de vértices debe ser un lado definido para el grafo[13]. Si alguna pareja de vértices consecutivos no está definida en el conjunto de lados del grafo se dice que la trayectoria es **inválida.**

Si en un grafo es posible viajar de cada nodo hacia todos los demás se dice que es un **grafo conectado**, si el grafo es no dirigido. Si el grafo es dirigido se dice que es **fuertemente conectado**.

El número de lados presentes en una trayectoria se denomina **longitud de la trayectoria**[15].

1.2.2.1 – Aristas.

Son las líneas con las que se unen las aristas de un grafo y con la que se construyen también caminos[16].

Si la arista carece de dirección se denota indistintamente **<X, Y>** o **< Y, X>**, siendo X y Y los vértices que une.

Es importante destacar que las aristas tienen varias clasificaciones, dentro de ellas la más importante es el término de aristas adyacentes el cual se dice que dos aristas son adyacentes si convergen en el mismo vértice[12].

1.2.2.2 – Vértices.

Son los puntos o nodos con los que está conformado un grafo. Se denomina grado de un vértice al número de aristas de las que es extremo. Se dice que un vértice es *par* o *impar* según lo sea su grado[16].

- Vértices Adyacentes: si se tiene un par de vértices de un grafo (U, V) y una arista que los une, entonces U y V son vértices adyacentes y se dice que U es el vértice inicial y V el vértice adyacente.
- Vértice Aislado: Es un vértice de grado cero.
- Vértice Terminal: Es un vértice de grado uno[14].

1.2.2.3 - Caminos.

Sean X, Y dos vértices, se dice que hay un camino en G de *x* a *y* si existe una sucesión finita no vacía de aristas {X, v1}, {v_1, v2},..., {v_n, Y}. En este caso X e Y se llaman los extremos del camino, el número de aristas del camino se llama la longitud del camino. Si los vértices no se repiten el camino se dice propio o simple. Si hay un camino no simple entre 2 vértices, también habrá un camino simple entre ellos. Cuando los dos extremos de un camino son iguales, el camino se llama circuito o camino cerrado. Se le llama ciclo a un circuito simple y un vértice X se dice accesible desde el vértice Y si existe un camino entre ellos[16].

1.2.2.4 – Árbol.

Un grafo que no tiene ciclos y que conecta a todos los puntos, se llama árbol. En un grafo con n vértices, los árboles tienen exactamente n - 1 aristas, y hay n^{n-2} árboles posibles. Su importancia radica en que los árboles son grafos que conectan todos los vértices utilizando el menor número posible de aristas. Un importante campo de aplicación de su estudio se encuentra en el análisis

filogenético, el de la filiación de entidades que derivan unas de otras en un proceso evolutivo, que se aplica sobre todo a la averiguación del parentesco entre especies; aunque se ha usado también, por ejemplo, en el estudio del parentesco entre lenguas[16].

1.2.2.5 – Algoritmo.

Algoritmo de las Margaritas.

Se presentará aquí un método heurístico, debido a B. Lemaire, que permite obtener una solución satisfactoria del problema consistente en determinar el conjunto de rutas o recorridos que deben establecerse para un parque dado de vehículos de transporte, designados para atender a cierto número de clientes.

Este problema puede interpretarse como uno en el cual debe aprovisionarse a un conjunto de clientes desde un centro de distribución, o también como la recogida, a un conjunto de depósitos, de productos que deben ser llevados a un centro de recolección.

Por ejemplo, recogida diaria a productores de leche, productos alimenticios perecederos, ganado, etc., recogida de escolares, envases, etc., distribución a expendedores de bebidas, refrescos, comidas, etc., aprovisionamiento de mayoristas o detallistas de productos diversos, suministro de petróleo o gasolina a estaciones de servicio, etc.

Para realizar la transportación se dispone de un grupo de camiones (sobreentendiéndose un vehículo de transporte cualquiera) y son conocidas sus capacidades y los costos (distancia, tiempo, etc.) de transporte entre clientes y de cada uno de ellos al centro de recolección o distribución. También se conoce la demanda de cada cliente.

El algoritmo posee la suficiente flexibilidad como para que, mediante los análisis que requiera el caso, pueda ser adaptado a situaciones más complejas, donde por ejemplo, deban considerarse demoras en la entrega o recogida (tiempos de carga o descarga, de estacionamiento, horas de apertura y cierre de los clientes, velocidad limitada, etc.), características de los camiones (distintas capacidades, limitaciones de tonelaje o volumen, disponibilidad temporal, etc.), dificultades de tránsito, variedad o exigencias técnicas de la carga, etc.

Teniendo en cuenta lo anteriormente descrito, el objetivo del problema será organizar las rutas de entrega o recogida de acuerdo con las restricciones impuestas y de manera que ello arroje los mejores resultados para el propósito económico o de otra índole, que se desea obtener[14].

1.3– Descripción de los Sistemas Existentes.

Hoy en día la gran evolución de las tecnologías de la informática, ha permitido el desarrollo y creación de infinidad de sistemas que optimizan el trabajo con la Teoría de Redes, mediante la revisión bibliográfica para la realización del presente trabajo se encontraron algunas herramientas que se corresponden en parte con el objeto de estudio del mismo pero no resuelven a totalidad las necesidades del usuario que las consulta pues ninguna presenta el algoritmo antes mencionado. A continuación se muestran los más representativos:

1.3.1- QSB.

QSB es un sistema desarrollado en Pascal, el cual contribuye en la toma de decisiones, contiene herramientas muy útiles para resolver distintos tipos de problemas en el campo de la investigación operativa[17]. Programas específicos para resolver el problema del trasbordo, el problema del transporte, el de asignación, el problema del camino más corto, flujo máximo, árbol generador y problema del agente viajero. Su principal desventaja para el caso

que se analiza en este trabajo es que no posee el Método de las Margaritas para el trabajo con la Teoría de Redes.

1.3.2– Qmwin2.

Qmwin2 es un sistema desarrollado por la Universidad de Guayaquil, su objetivo fundamental radica en la resolución y descripción de algunos algoritmos en la Teoría de Redes [18]. Este software describe en su desarrollo la resolución del problema Su principal desventaja para el caso que se analiza en este trabajo es que no posee el Método de las Margaritas para el trabajo con la Teoría de Redes.

1.3.3 – POM.

POM es un sistema desarrollado en una Universidad de Estados Unidos, en su contenido, sobre el tema Teoría de Redes, posee solamente un módulo de transporte, en el que tanto la entrada como la representación de los datos se realiza de manera literal y de muy alta complejidad para un usuario de nivel básico en el tema. Su principal desventaja para el caso que se analiza en este trabajo es que no posee el Método de las Margaritas para el trabajo con la Teoría de Redes[19].

1.4 - Tendencias, metodologías, lenguaje, herramientas y tecnologías actuales.

Las fases en el desarrollo de una herramienta de software deben estar soportadas por las indicaciones de algunas metodologías para poder garantizar su calidad y eficiencia. Unido a esto se debe tener presente que antes de llevarse a cabo cualquier aplicación es necesario realizar un análisis de las tecnologías actuales, con el objetivo de tomar la más indicada según los requerimientos que se deben cumplir para el desarrollo de la nueva propuesta.

Además, es importante analizar el estado de las herramientas afines a escala global.

1.4.1 - Tendencias actuales a considerar.
Programación Orientada a Objetos (POO).

La programación orientada a objetos es una de las formas más populares de programar y ha tenido gran acogida en el desarrollo de proyectos de software desde los últimos años. Esta aceptación entre los desarrolladores se debe a sus grandes capacidades y ventajas frente a las antiguas formas de programar. Tradicionalmente, la programación fue hecha en una manera secuencial o lineal, es decir una serie de pasos consecutivos con estructuras consecutivas y bifurcaciones.

Los lenguajes basados en esta forma de programación ofrecían ventajas al principio, pero el problema ocurre cuando los sistemas se vuelven complejos. Estos programas escritos basados en la programación lineal no ofrecen flexibilidad y el mantener una gran cantidad de líneas de código en sólo bloque se vuelve una tarea complicada.

Frente a esta dificultad aparecieron los lenguajes basados en la programación estructurada. La idea principal de esta forma de programación es separar las partes complejas del programa en módulos o segmentos que sean ejecutados conforme se requieran. De esta manera se tiene un diseño modular, compuesto por módulos independientes que puedan comunicarse entre sí. Poco a poco este estilo de programación fue reemplazando al estilo “espaguetti” impuesto por la programación lineal.

La creciente tendencia de crear programas cada vez más grandes y complejos llevó a los desarrolladores a crear una nueva forma de programar que les permita crear sistemas de niveles empresariales y con reglas de negocios muy complejas. Para estas necesidades ya no bastaba la programación estructurada ni mucho menos la programación lineal. Es así como aparece la programación orientada a objetos (POO).

La POO viene de la evolución de la programación estructurada; básicamente simplifica la programación con la nueva filosofía y nuevos conceptos que tiene. Este tipo de programación por su parte, fue creado precisamente para superar las insuficiencias de la programación estructurada, aprovechando sus mejores ideas. Su principal preocupación durante el desarrollo de programas es determinar los objetos que representarán, de una forma más adecuada, los elementos presentes en un problema, una vez determinados estos, se pasa a determinar cuáles son sus características o atributos principales y cuáles son las acciones (procedimientos) que son realizados con tales atributos y que por tanto caracterizan su comportamiento. La cuestión más interesante de la POO es que proporciona conceptos y herramientas con las cuales se modela y representa el mundo real tan fielmente como sea posible.

Ventajas de un lenguaje orientado a objetos.

* Fomenta la reutilización y extensión del código.
* Permite crear sistemas más complejos.
* Relacionar el sistema al mundo real.
* Facilita la creación de programas visuales.
* Construcción de prototipos.
* Agiliza el desarrollo de software.
* Facilita el trabajo en equipo.
* Facilita el mantenimiento del software[20].

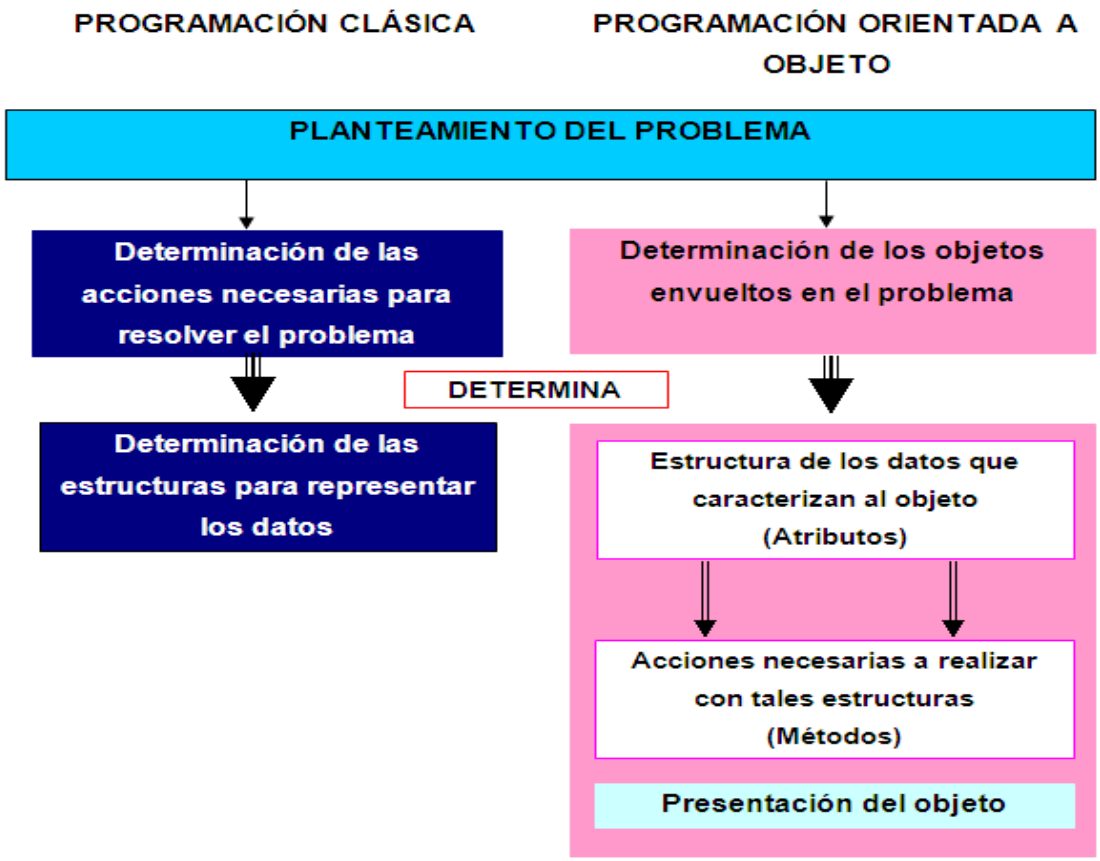


Figura 4. Características de la Programación Clásica y la POO.

Por ello la programación orientada a objeto tiene como objetivos facilitar el desarrollo y comprensión de programas de gran porte y posibilitar la reutilización de código. Las principales características que identifican a un lenguaje orientado a objeto son los conceptos de clases, herencia y polimorfismo.

A diferencia de otros paradigmas de programación, el paradigma orientado a objetos es tanto un estilo de programación como una metodología, que cuenta como principales características:

1. Flexibilidad y adaptabilidad.
2. Reutilización del software.
3. Notación e integración consistente de las fases del diseño.

En la POO estas características también deben estar presentes y esta forma de programar por sus características y mecanismos ayuda a conservarlas y las afianza. Estas características están muy interrelacionadas con las cuatro

propiedades básicas del modelo de objetos, que son la abstracción, encapsulación, modularidad y jerarquía, tal y como se presenta a continuación.

- Claridad

La claridad del código fuente de los programas ayuda a la comprensión no sólo de quienes los confeccionan, sino también de otras personas que le pueden dar mantenimiento. Muchas veces un programa se ve limitado porque no hay quien lo entienda y pueda mejorarlo o hacer otras versiones y otras veces no se puede reusar su código por el mismo motivo. A la claridad ayuda, poner nombres adecuados a las rutinas y variables, ni muy largos ni muy cortos y carentes de sentido, también ayuda la modularidad, la abstracción y el encapsulamiento.

- Modularidad

La modularidad, la abstracción, y el encapsulamiento están estrechamente relacionadas entre sí.

La **modularidad** consiste en dividir las tareas del programa de una forma y lógica razonable, generalmente siguiendo un diseño t*op-down*, desde lo más general a lo más específico. Un programa con modularidad puede dividirse en módulos separados a los cuales se les puede dar mantenimiento también por separado.

La **abstracción** es la propiedad que permite representar las características esenciales de un objeto, sin preocuparse de las restantes características (no esenciales).

Una abstracción se centra en la vista externa de un objeto, de modo que sirva para separar el comportamiento esencial de un objeto de su implementación. Definir una abstracción significa describir una entidad del mundo real, no importa lo compleja que pueda ser, y a continuación utilizar esta descripción en un programa.

El elemento clave de la programación orientada a objetos es la **clase**. Una **clase** se puede definir como una descripción abstracta de un grupo de objetos, cada uno de los cuales se diferencia por su *estado* específico y por la posibilidad de realizar una serie de *operaciones.*

La idea de escribir programas definiendo una serie de abstracciones no es nueva, pero el uso de clases para gestionar dichas abstracciones en lenguajes de programación ha facilitado considerablemente su aplicación.

La **encapsulación** *o* **encapsulamiento** es la propiedad que permite asegurar qué contenido de la información de un objeto está oculta al mundo exterior: El objeto A no conoce lo que hace el objeto B, y viceversa. La encapsulación (también se conoce como *ocultamiento de la información),* en esencia, es el proceso de ocultar todos los secretos de un objeto que no contribuyen a sus características esenciales.

La encapsulación permite la división de un programa en módulos. Estos módulos se implementan mediante clases, de forma que una clase representa la encapsulación de una abstracción. En la práctica, esto significa que cada clase debe tener dos partes: una interface y una implementación. La *interface* de una clase captura sólo su vista externa y la *implementación* contiene la representación de la abstracción, así como los mecanismos que realizan su comportamiento.

- Protección

La protección consiste en cuidar de las manos inexpertas las partes de los programas más sensibles o que no se desean que se tenga acceso. En C esto se logra colocando en los ficheros de encabezamiento (.h) declaraciones de funciones y de los datos y ofrecer los módulos compilados (obj.) donde están las definiciones de las funciones y los datos.

- Reusabilidad

La reusabilidad consiste en hacer los programas de tal manera que sus módulos, estructuras de datos y funciones, puedan usarse en otros programas

con un mínimo de cambios. A esto ayudan las antes mencionadas: modularidad, abstracción, y encapsulamiento.

- Extensibilidad

La extensibilidad es parecida a la reusabilidad, pero persigue otro objetivo. Cuando se diseña un programa este debe ser fácilmente modificable de manera que se le puedan agregar nuevas funcionalidades, es decir, extender el programa, con un mínimo de cambios en el código fuente[21].

1.4.2 – Metodología de Desarrollo de Software.

1.4.2.1 - Proceso Unificado de Desarrollo (RUP).

En la actualidad existe una tendencia a la creación de un software más grande y complejo, esto impulsado sin duda alguna por el vertiginoso avance de la tecnología que pone al alcance de todos máquinas más potentes con una capacidad de procesamiento y almacenamiento de información que crece casi sin límites. Hoy en día se necesita un software más complejo que satisfaga la demanda de los usuarios, que cada día son mayores. Ante estos retos los desarrolladores de software necesitan nuevos métodos para el desarrollo de aplicaciones informáticas.

La comunidad de desarrolladores de software necesita una forma coordinada de trabajar. Necesitan un proceso que integre las múltiples facetas de desarrollo. Este proceso debe[22]:

- Proporcionar una guía para ordenar las actividades de un equipo.
- Dirigir las tareas de cada desarrollador por separado y del equipo como un todo.
- Ofrecer criterios para el control y la medición de los productos y actividades del proyecto.

La actual solución al problema del software lo constituye el Proceso Unificado de Desarrollo (RUP, siglas en ingles de Rational Unified Process). El proceso

Unificado surge en 1998 y se identifica por ser un proceso de desarrollo de software definido por tres fases claves.

Dirigido por Casos de Uso: Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Estos guían el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema. “Los casos de uso no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor. Dirigido por casos de uso quiere decir que el proceso de desarrollo sigue un hilo – avanza a través de una serie de flujos de trabajos que parten de los casos de uso”[22].

Centrado en la Arquitectura: La arquitectura invoca los elementos más significativos del sistema y está influenciada entre otros por plataformas de software, sistemas operativos, gestores de base de datos, protocolos, consideraciones generales y requisitos no funcionales. La arquitectura abarca decisiones importantes sobre la organización del sistema, los elementos estructurales, sus interfaces y comportamientos. Es de resaltar la interacción que debe existir entre arquitectura y caso de uso, estos dos aspectos deben evolucionar en paralelo[22].

Iterativo e Incremental: para hacer más manejable un proyecto se recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un mini - proyecto, cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo[22].

Se puede decir que el Proceso unificado atraviesa por una serie de ciclos, los cuales se dividen en las fases:

Inicio: En esta fase inicial se desarrolla una descripción del producto final partiendo de una buena idea. Es en esta fase donde se definen las principales funciones del sistema para sus usuarios, se obtiene una vista preliminar de la arquitectura del sistema y se define cuál es el plan del proyecto y su costo.

Elaboración: En esta fase se especifican los casos de uso y se diseña la arquitectura del sistema, la cual se expresa en forma de vista de todos los modelos, los que en su totalidad conforman el sistema entero.

Construcción: Es una fase de desarrollo donde se emplean la mayor cantidad de recursos para poder entregar a los usuarios un producto.

Transición: En esta fase el producto es probado por un grupo pequeño de usuarios con experiencias para informar las deficiencias y que estas son corregidas. En esta fase donde se forma al cliente, se proporciona una línea de ayuda y asistencia y se corrigen los defectos encontrados tras la entrega.
Cada fase se divide en iteraciones o mini – proyectos los cuales atraviesan por los flujos de trabajo: requisitos, análisis, diseño, implementación y prueba.
Es acertado decir que el Proceso Unificado guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo de desarrollo y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos y el establecimiento de una guía arquitectónica, para diseñar y probar el sistema. Además el Proceso Unificado es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas. Estas razones hacen del RUP la metodología seleccionada para el desarrollo de numerosas aplicaciones.

1.4.3 - Lenguajes.

1.4.3.1 – UML.

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (Unified Modeling Lenguaje, UML) para preparar todos los esquemas de un sistema de software[23]. El Lenguaje UML fue creado por un grupo de estudiosos de la Ingeniería del Software en el año 1995 y es un lenguaje gráfico de modelado orientado a objetos. Este lenguaje tiene una sintaxis y una semántica bien definidas, sirviendo además para todas las etapas de desarrollo. UML ofrece un estándar para describir un plano del sistema (modelo), incluyendo aspectos

concretos como expresiones de lenguajes de programación, esquemas de base de datos y componentes de software reutilizables.

Estereotipo UML

Los estereotipos son el mecanismo de extensibilidad incorporado más utilizado dentro de UML. Un estereotipo representa una distinción de uso. Puede ser aplicado a cualquier elemento de modelado, incluyendo clases, paquetes, relaciones de herencia, etc. Por ejemplo, una clase con estereotipo \' actor \' es una clase usada como un agente externo en el modelado de negocio. Una clase patrón es modelada como una clase con estereotipo parametrizado, lo que significa que puede contener parámetros.

Importancia del UML.

Hoy en día, UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código.

En otros términos, así como en la construcción de un edificio se realizan planos previo a su construcción, en Software se deben realizar diseños en UML previa codificación de un sistema, ahora bien, aunque UML es un lenguaje, éste posee más características visuales que programáticas, las mismas facilitan a integrantes de un equipo multidisciplinario participar e intercomunicarse fácilmente, estos integrantes siendo los analistas, diseñadores, especialistas de área y desde luego los programadores[24].

Complejidad / Objetos UML

Entre más complejo es el sistema que se desea crear más beneficios presenta el uso de UML, las razones de esto son evidentes, sin embargo, existen dos puntos claves: El primero se debe a que mediante un plano/visión global resulta más fácil detectar las dependencias y dificultades implícitas del sistema, y la segunda razón radica en que los cambios en una etapa inicial (Análisis) resultan más fáciles de realizar que en una etapa final de un sistema como lo sería la fase intensiva de codificación.

Puesto que UML es empleado en el análisis para sistemas de mediana y alta complejidad, era de esperarse que su base radique en otro paradigma empleado en diseños de sistemas de alto nivel que es la orientación a objetos, por lo que para trabajar en UML puede ser considerado un pre-requisito tener experiencia en un lenguaje orientado a objetos[24].

1.4.3.2 – C++.

El lenguaje C++ se comenzó a desarrollar en 1980. Su autor fue B. Stroustrup, formado en la ATT. Al comienzo era una extensión del lenguaje C que fue denominada C with classes. Este nuevo lenguaje comenzó a ser utilizado fuera de la ATT en 1983. El nombre C++ es también de ese año, y hace referencia al carácter del operador incremento de C (++). Ante la gran difusión y éxito que iba obteniendo en el mundo de los programadores, la ATT comenzó a estandarizarlo internamente en 1987. En 1989 se formó un comité ANSI (seguido algún tiempo después por un comité ISO) para estandarizarlo a nivel americano e internacional.

En la actualidad, el C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. El C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas de las dificultades y limitaciones del C original[25]. La evolución de C++ ha continuado con la aparición de Java, un lenguaje creado simplificando algunas cosas de C++ y añadiendo otras, que se utiliza para realizar aplicaciones en Internet.

Hay que señalar que el C++ ha influido en algunos puntos muy importantes del ANSI C, como por ejemplo en la forma de declarar las funciones, en los punteros a void, etc. En efecto, aunque el C++ es posterior al C, sus primeras versiones son anteriores al ANSI C, y algunas de las mejoras de éste fueron tomadas del C++.

El C++ es a la vez un lenguaje procedural (orientado a algoritmos) y orientado a objetos. Como lenguaje procedural se asemeja al C y es compatible con él, aunque ya se ha dicho que presenta ciertas ventajas. Como lenguaje orientado a objetos se basa en una filosofía completamente diferente, que exige del programador un completo cambio de mentalidad. Las características propias de la Programación Orientada a Objetos (Object Oriented Programming) de C++ son modificaciones mayores que sí cambian radicalmente su naturaleza anterior[25].

1.4.4 - Herramientas de desarrollo.

1.4.4.1 – Rational Rose.

La complejidad de los proyectos de software hoy en día, el constante cambio de requerimientos y la falta de una documentación durante el proceso de desarrollo provoca que los proyectos se retrasen en tiempo y se incrementen en costo. La solución a esta problemática es implantar una arquitectura de desarrollo que permita hacer seguimiento a los proyectos desde su etapa de requerimientos, hasta su implantación. Rational ofrece un Proceso Unificado (RUP) para el desarrollo de los proyectos de software, desde la etapa de Ingeniería de Requerimientos hasta la etapa de pruebas. Para cada una de estas etapas existe una herramienta que ayuda en la administración de los proyectos, Rose es la herramienta de Rational para la etapa de análisis y diseño de sistemas[26].

Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas de Rose es que utiliza la notación estándar en la arquitectura de Software (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común. Otra ventaja de Rose es que los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto. Además Rose soporta la construcción de componentes en lenguajes como C++, Visual Basic, Java, Ada, genera IDL's para aplicaciones CORBA. Por todo lo anterior Rose es la herramienta de Análisis, Diseño, Modelado y Construcción de software

Orientado a Objetos líder en el mercado y es por todo esto que fue escogida para ser utilizada en este trabajo[26].

1.4.4.2 – Borland C++ Builder 6.

C++ Builder es una aplicación Windows que proporciona un entorno de trabajo visual para construir aplicaciones Windows que integra distintos aspectos de la programación en un entorno unificado o integrado. La integración y facilidad de manejo hace que sea una herramienta indispensable para el desarrollo rápido de aplicaciones o RAD (***R**apid **A**pplication **D**evelopment*). Borland C++Builder 6 constituye la vía más rápida para desarrollar aplicaciones C++[27].

Características:

- C++Builder 6 ofrece el entorno de desarrollo visual que miles de desarrolladores C++ buscan a la hora de crear sus aplicaciones. Se pueden desarrollar rápidamente prototipos y aplicaciones completas, mediante una amplia paleta de componentes con más de 85 componentes reutilizables.

- Los conocimientos de programación quedan protegidos gracias al uso de código 100% estándar ANSI/ISO, sin extensiones propietarias que limiten la elección de futuras herramientas de desarrollo. Si se dispone de código C++ se puede importar directamente en C++Builder.

- Posibilita la gestión eficaz de proyectos mediante la herramienta Advanced Project Manager, para el control de los archivos fuentes utilizados en los proyectos, ello hace posible visualizar las aplicaciones y las distribuciones realizadas.

- Muchas de las herramientas que forman parte del entorno de desarrollo integrado han sido mejoradas para incrementar su productividad, incluyendo muchos Asistentes para la mayoría de tareas habituales.

- El compilador C++ incluido, Borland C++ Compiler 5.5, es un compilador y optimizador del código, de alto rendimiento y multihebrado, que actúa en segundo plano. Sin dejar de trabajar, sus aplicaciones se compilarán y ejecutarán más rápidamente.

1.5– Conclusiones.

Mediante todo lo analizado en el presente capítulo se concluye que:

- No existe un sistema que dé solución al algoritmo de las Margaritas.
- La programación orientada a objetos, como tendencia actual a considerar, es la de mayor aceptación por programadores y desarrolladores.
- Se selecciona la metodología RUP, como guía en el proceso de desarrollo del software propuesto, así como el uso de UML como lenguaje de modelado por ser un estándar internacional. Para la representación gráfica la herramienta Rational Rose.
- Como lenguaje de programación seleccionado está C++, además de ser considerado como el mejor lenguaje concebido de la programación orientada a objetos.
- Como herramienta de desarrollo de la aplicación se optó por Borland C++ Builder 6.

Capítulo 2: - Descripción y Construcción de la Solución Propuesta.

2.1 – Introducción.

Existen al menos dos formas fundamentales para representar el contexto de una aplicación: modelo del negocio y modelo del dominio. Se toma como punto de partida el modelo del dominio, puesto que las características de la problemática en cuestión no permiten identificar un negocio bien definido, como

resultado de este proceso se obtiene una descripción de los conceptos principales y entidades, además de los elementos necesarios para la buena comprensión del problema.

En este capítulo se recogen también, la descripción de la solución propuesta mediante algunos de los artefactos que propone la Metodología RUP. Los artefactos referidos son: el Modelo del Dominio, los Requerimientos Funcionales y No Funcionales, el Diagrama de Casos de Uso y la descripción de cada uno.

2.2 – Modelo de Dominio.

El modelo del dominio permite comprender el contexto del sistema a través de la representación de los objetos más importantes en el mismo, dichos objetos pueden representar entidades físicas (cosas que existen), así como eventos que suceden en el entorno del sistema. El mismo se describe a través de diagramas de UML (específicamente diagramas de clases). A este nivel no se representan clases del software con atributos y responsabilidades, sino clases que describen los conceptos fundamentales que se manejan entre desarrolladores, clientes y usuarios. Las tres formas típicas de aparición de las clases del dominio son[22]:

- Objetos del negocio que representan cosas manipulables en el negocio.
- Objetos del mundo real y conceptos de los que el sistema debe hacer un seguimiento.
- Sucesos que ocurrirán o han ocurrido.

A continuación se muestra la representación del modelo del dominio del sistema:

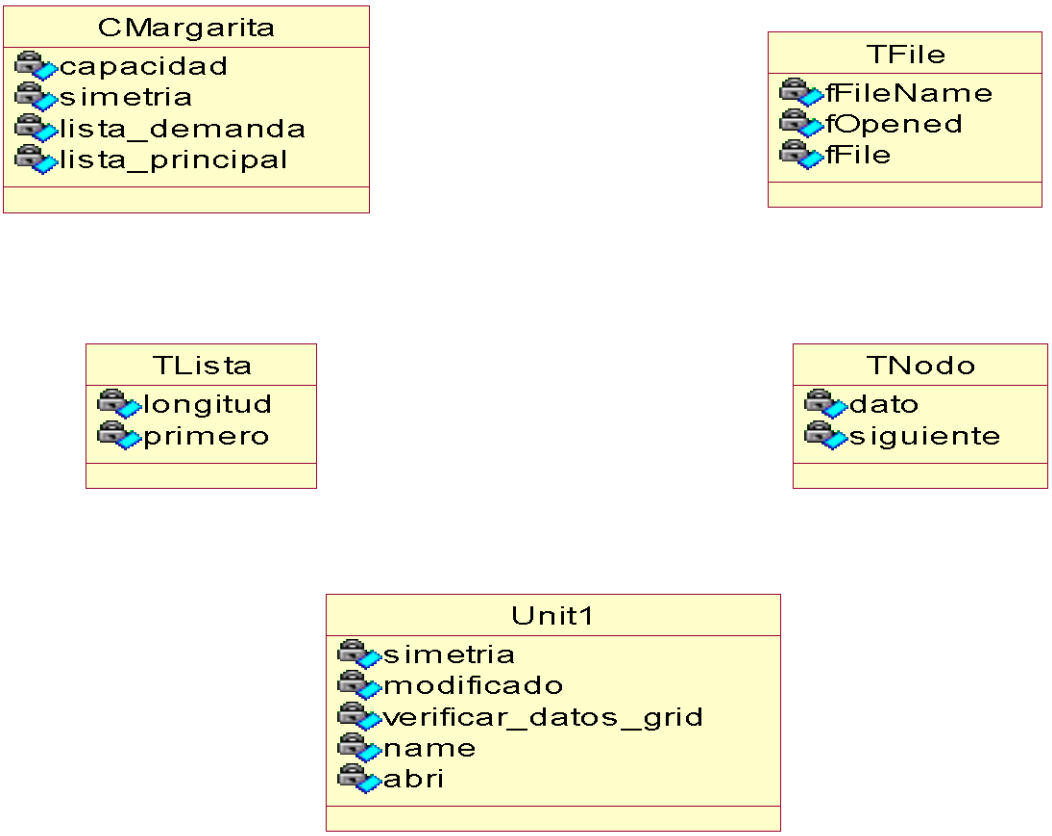


Figura 5. Modelo de Dominio.

2.3 – Requerimientos Funcionales.

Los requerimientos funcionales permiten expresar una especificación más detallada de las responsabilidades del sistema que se propone. Ellos permiten determinar, de una manera clara, lo que debe hacer el mismo[22].

Los requerimientos funcionales del software propuesto son los siguientes:

1. Crear una nueva tabla.
2. Abrir un fichero con la información de una tabla.
3. Crear un fichero con extensión propia con la información de la tabla que se elaboró.
4. Salvar una copia de resguardo de la tabla que se está elaborando en un fichero con extensión propia.
5. Insertar un vértice.
6. Modificar un vértice.
7. Eliminar un vértice.
8. Insertar distancia entre clientes.

9. Modificar distancia entre clientes.
10. Insertar demanda por cliente.
11. Modificar demanda por cliente.
12. Aplicar a una tabla el algoritmo de las Margaritas.
13. Salir de la aplicación.

2.4 – Requerimientos no Funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener, como restricciones del entorno o de implementación, rendimiento, etc.[22].

Requerimientos de apariencia o interfaz externa.

En el diseño de la interfaz de este producto informático se tomó como prioridad la idea de que el software fuese lo más entendible y sencillo de usar, siempre que no se afectase ninguna de las funcionalidades del mismo. Con este tipo de diseño se persigue el objetivo de que el producto sea aceptado y asimilado de la manera más rápida por los usuarios que interactuarán con el mismo, sin tomar en cuenta que posean o no, habilidades en el manejo de aplicaciones informáticas.

Requerimientos de Usabilidad.

Es de vital importancia tener presente que el software será de uso profesional y está encaminado a usuarios con los conocimientos mínimos necesarios en el campo de acción de los modelos de redes.

La utilización del producto no requiere de gran experiencia en el uso de las computadoras por parte de los usuarios, además se utilizó un lenguaje familiar para los mismos lo que facilita la utilización del software.

Requerimientos de Soporte.

Con el objetivo de obtener el mejoramiento del software y su avance se desarrolló analizando las ventajas para su mantenimiento. Para alcanzar lo que

con anterioridad se describe está garantizada toda la documentación técnica, así como el código fuente, el cual está escrito sobre la base de estándares adecuadamente definidos y documentados.
Otro punto que se tuvo en cuenta durante el desarrollo del software fue la posibilidad de aumentar su uso, pues mediante la arquitectura que presenta, posibilita adaptarlo a otros algoritmos que se le incluyan.

Requerimientos de Software.
El software está realizado para sistema operativo Windows, se requiere Windows 98 o superior.

Requerimientos de Hardware.
Para la explotación del sistema los requerimientos mínimos de hardware son:

- Procesador Pentium a 133MHz de velocidad.
- 64 Mb de memoria RAM.
- 100 Mb de espacio disponible en disco.

2.5 – Descripción del sistema propuesto.

2.5.1 – Concepción general del sistema

Con la presente investigación se desea obtener un producto informático que ofrezca solución a algunos de los problemas de la Teoría de Redes, además proporcionará la posibilidad de analizar tabularmente el algoritmo en cuestión con el objetivo de su mayor comprensión e interpretación de los resultados devueltos para estudiantes e investigadores.
Mediante esta aplicación se trata de eliminar algunos de los inconvenientes de las herramientas existentes, analizadas como antecedentes en el *Capítulo 1: Fundamentación Teórica*.

2.5.2 – Definición de los actores y casos de uso del sistema.

Actores del sistema.

Un actor no es más que un conjunto de roles que los usuarios de Casos de Uso desempeñan cuando interaccionan con estos Casos de Uso. Los actores representan terceros fuera del sistema que colaboran con el mismo. Una vez que hemos identificado los actores del sistema, tenemos identificado el entorno externo del sistema[22].

Nombre del Actor	**Descripción**
Usuario	Cualquier usuario que interactúe con el software, puede ser un profesor, un estudiante, investigador o cualquier otra persona que necesite conocer los resultados de cierto problema de su interés. Este usuario tendrá acceso a todos los requerimientos funcionales del sistema.

Casos de Uso del Sistema.

Los actores interactúan y usan el sistema a través de Casos de Uso. Los Casos de Uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario**.** De manera más precisa, un Caso de Uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia[22].
En el presente trabajo los casos de uso del sistema quedan representados por:

1. Crear nueva tabla.
2. Abrir documento.
3. Salvar documento.
4. Gestionar vértice.
5. Gestionar distancia entre clientes.
6. Gestionar demanda por cliente.
7. Aplicar algoritmo de las Margaritas.
8. Salir de la aplicación.

2.5.3 – Diagrama de casos de uso del sistema.

Las formas en que los actores utilizan el sistema se representa como casos de uso, que no son más que un grupo de funcionalidades del sistema que

devuelven un resultado de valor al actor. O sea que es una secuencia de acciones que el sistema lleva a cabo cuando el actor interactúa con él [22].

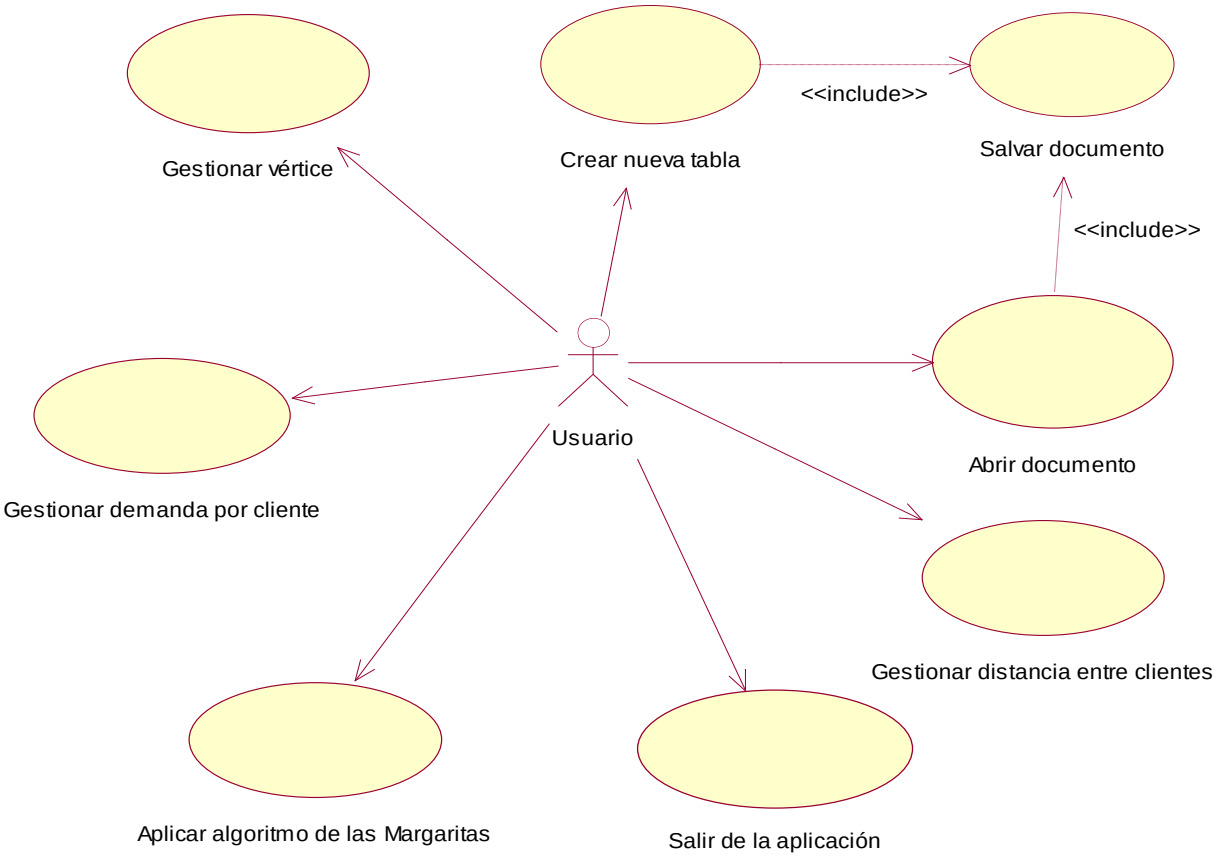


Figura 6. Modelo de Casos de Uso

La manera de ilustrar la interacción entre sistema y actores es mediante el diagrama de casos de uso del sistema. En el diagrama mencionado anteriormente se describe la forma en que cada usuario se relaciona con el software.

2.5.4 – Descripción de los Casos de Uso.

Caso de uso: Crear nueva tabla.
Actores: Usuario (inicia).
Propósito: Permitir la creación de una nueva tabla.

Resumen: El caso de uso se inicia cuando el usuario desea elaborar una nueva tabla y posee toda la información necesaria sobre la misma. El sistema verifica si hay alguna tabla creada sin guardar, dando la posibilidad al usuario de realizar el caso de uso **Salvar documento** para mantener la información o desechar la misma. Se concluye este caso de uso con la creación de una nueva tabla.
Precondiciones:
Referencias: R1 Salvar documento (Extend).
Poscondiciones: • Se creó una nueva tabla.
Requerimientos especiales: -
Prototipo: Ver Anexo 1 - Prototipo # 1

Caso de uso: Abrir documento.
Actores: Usuario (inicia).
Propósito: Cargar de un fichero la información almacenada de una tabla y representarla mediante un documento.
Resumen: El caso de uso se inicia cuando el usuario desea cargar una tabla almacenada en un fichero creado con anterioridad. El sistema verifica si hay alguna tabla creada sin guardar, dando la posibilidad al usuario de realizar el caso de uso **Salvar documento** para mantener la información o desechar la misma. Se concluye este caso de uso con la representación en un documento de la información de la tabla almacenada en el fichero.
Precondiciones: • El fichero debe ser del tipo AGP.
Referencias: R2 Salvar documento (Extend).

Poscondiciones: • Se culmina con la representación en un documento de la información almacenada de la tabla en el fichero.
Requerimientos especiales: -
Prototipo: Ver Anexo 1 - Prototipo # 2

Caso de uso: Salvar documento.
Actores: Usuario (inicia).
Propósito: Permitir el almacenamiento de la información de una tabla en un fichero con extensión propia (AGP).
Resumen: El caso de uso se inicia cuando el usuario desea salvar en un fichero con extensión propia la información de una tabla que está elaborando. El usuario entra los datos del fichero como nombre y ubicación del mismo. Se concluye este caso de uso con la salva de una tabla en un fichero con extensión propia.
Precondiciones: • Debe existir una tabla con al menos un vértice.
Referencias: R3, R4
Poscondiciones: • Se culmina con la salva de una tabla en un fichero con extensión propia (AGP).
Requerimientos especiales: -
Prototipo: Ver Anexo 1 - Prototipo # 3

Caso de uso: Gestionar vértice.
Actores: Usuario (inicia).
Propósito: Permitir gestionar un vértice.

Resumen: El caso de uso se inicia cuando el usuario desea gestionar un vértice, el sistema le brinda la posibilidad de **Insertar un vértice**, **Modificar un vértice y Eliminar un vértice** de la tabla. El usuario puede especificar el nombre con el que desea insertar un vértice, el sistema verifica que el mismo no se repita. Para modificar un vértice, se debe seleccionar la fila o columna a la que pertenece y cambiar el nombre por el que se quiera. Si se desea eliminar un vértice, se selecciona el mismo y antes de eliminar se da la posibilidad de reafirmar si se continua o se cancela la operación. Se concluye este caso de uso con la actualización de la tabla que contiene el vértice.
Precondiciones: Para Modificar un vértice y Eliminar un vértice: • Debe de existir al menos un vértice. • Debe estar seleccionado algún vértice.
Referencias: R5, R6, R7
Poscondiciones: • Se concluye con la actualización de la tabla que contiene el vértice.
Requerimientos especiales: -
Prototipo: Ver Anexo 1 - Prototipo # 4

Caso de uso: Gestionar distancia entre clientes.
Actores: Usuario (inicia).
Propósito: Permitir gestionar la distancia entre clientes.

Resumen: El caso de uso se inicia cuando el usuario desea gestionar la distancia entre clientes, el sistema le brinda la posibilidad de **Insertar distancia entre clientes** ó **Modificar distancia entre clientes** de la tabla. Para modificar la distancia entre clientes, se debe seleccionar la fila o columna a la que pertenece y cambiar la distancia por la que se quiera. Se concluye este caso de uso con la actualización de la tabla que contiene el (los) cliente (s).
Precondiciones: Para Modificar distancia entre clientes: • Debe de existir al menos un par de clientes ó vértices.
Referencias: R8, R9
Poscondiciones: • Se concluye con la actualización de la tabla que contiene los clientes.
Requerimientos especiales: -
Prototipo: Ver Anexo 1 - Prototipo # 5

Caso de uso: Gestionar demanda por cliente.
Actores: Usuario (inicia).
Propósito: Permitir gestionar la demanda por cada cliente.
Resumen: El caso de uso se inicia cuando el usuario desea gestionar la demanda de cada cliente, el sistema le brinda la posibilidad de **Insertar demanda por cliente** ó **Modificar demanda por cliente** de la tabla. Para modificar la demanda por cliente, se debe seleccionar la fila o columna a la que pertenece y cambiar la demanda por la que se quiera. Se concluye este caso de uso con la actualización de la tabla que contiene el (los) cliente (s).

Precondiciones: Para Modificar demanda por cliente: • Debe de existir al menos un cliente ó vértice.
Referencias: R10, R11
Poscondiciones: • Se concluye con la actualización de la tabla que contiene el (los) cliente (s).
Requerimientos especiales: -
Prototipo: Ver Anexo 1 - Prototipo # 6

Caso de uso: Aplicar algoritmo de las Margaritas.
Actores: Usuario (inicia).
Propósito: Permitir aplicar el algoritmo de las Margaritas a una tabla creada por el usuario.
Resumen: El caso de uso se inicia cuando el usuario desea aplicar a una tabla el algoritmo de las Margaritas; este método comienza tomando como el vértice inicial al que se encuentra en la primera posición. Este caso de uso concluye con el cierre de la ventana de las tablas resultantes del algoritmo.
Precondiciones: Para Aplicar algoritmo de las Margaritas: • Debe de existir una tabla con un centro de distribución y al menos dos vértices que contenga los datos requeridos.
Referencias: R12
Poscondiciones: • Se concluye mostrando el conjunto de rutas a seguir, en orden de prioridad.
Requerimientos especiales: -
Prototipo: Ver Anexo 1 - Prototipo # 7

Caso de uso: Salir de la aplicación.

Actores: Usuario (inicia).
Propósito: Permitir cerrar la aplicación.
Resumen: El caso de uso se inicia cuando el usuario desea cerrar la aplicación. El sistema verifica si hay algún documento creado sin guardar, dando la posibilidad al usuario de realizar el caso de uso **Salvar documento** para mantener la información o desechar la misma. Se concluye el caso de uso con el cierre de la aplicación.
Precondiciones:
Referencias: R13
Poscondiciones: • Se cierra la aplicación.
Requerimientos especiales: -
Prototipo: Ver Anexo 1 - Prototipo # 8

2.6 – Construcción del sistema propuesto.

2.6.1 – Diagrama de clases del diseño.

A continuación se muestran los diagramas de clases del diseño propuestos para la construcción del sistema.

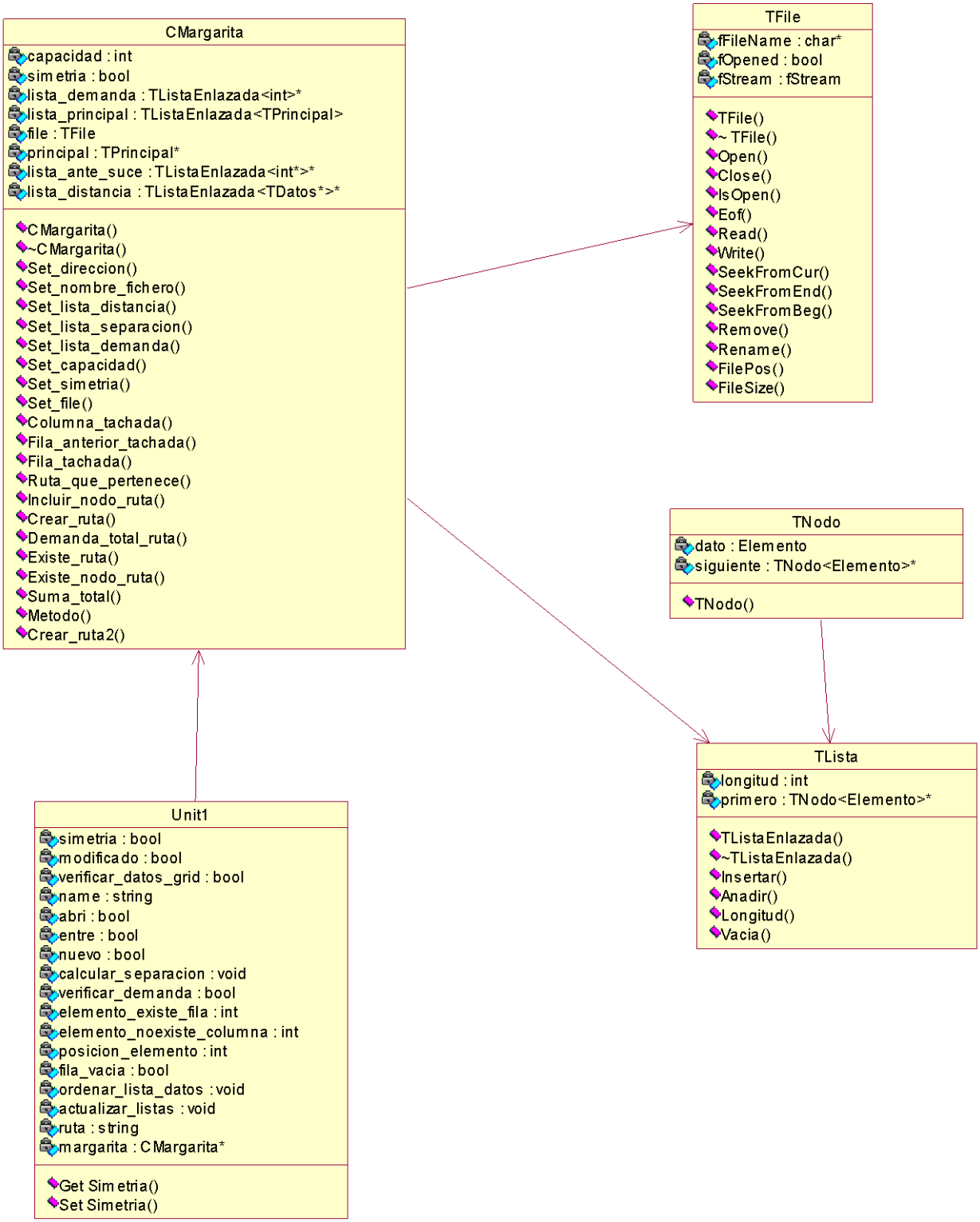


Figura 7. Diagrama de clases del diseño.

2.6.2 – Diagrama de clases persistentes.

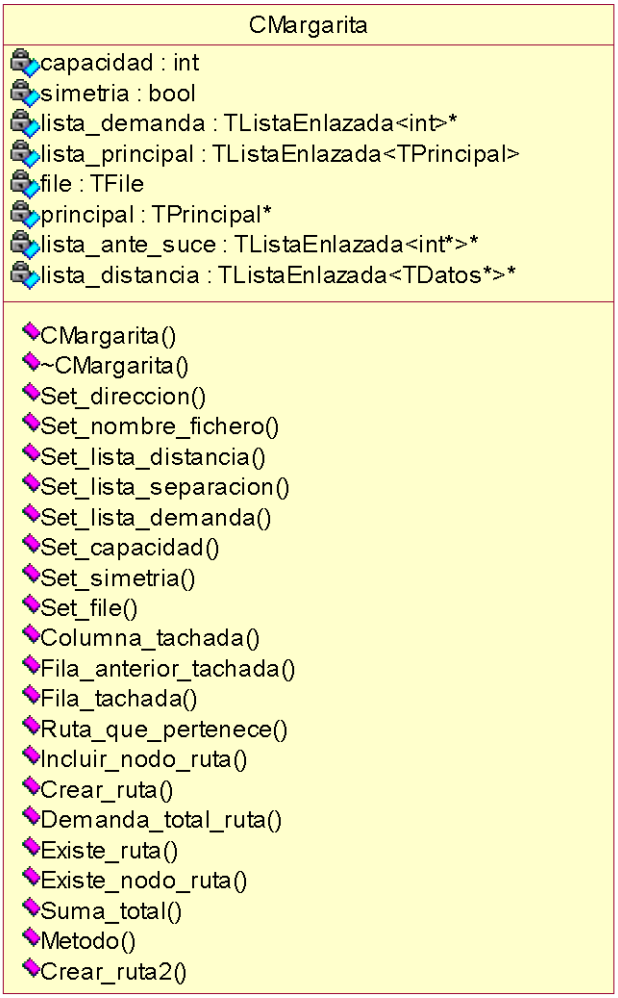


Figura 8. Diagrama de clases persistentes.

2.6.3 – Principios de diseño.

2.6.3.1 – Estándares en la interfaz del sistema.

La interfaz diseñada para el sistema está basada en el estándar de ventanas de Windows. El tipo de letra a utilizar será MS Sans Serif de estilo negrita y tamaño 8 y el diseño de la aplicación será conservador. El lenguaje de las opciones que brinda es de muy fácil comprensión para el usuario y la carga visual es adecuada. El sistema posee una barra de herramientas que permite acceso rápido a todas las opciones y en cada una de estas se han utilizado iconos para una mayor comprensión de la funcionalidad que realiza. En cuanto

a los mensajes de error e informativos que se muestran son breves, pero informando siempre en qué consiste el error.

Las figuras IX y X muestran ejemplos de lo expuesto anteriormente.

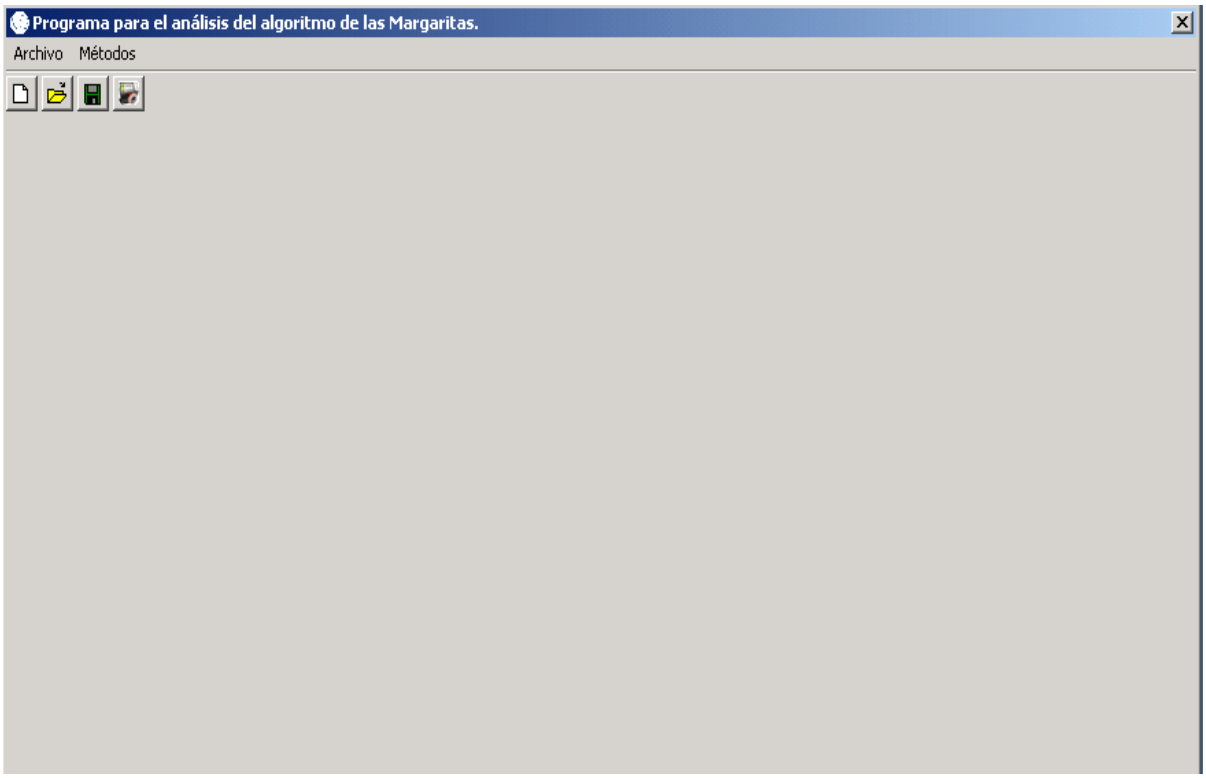
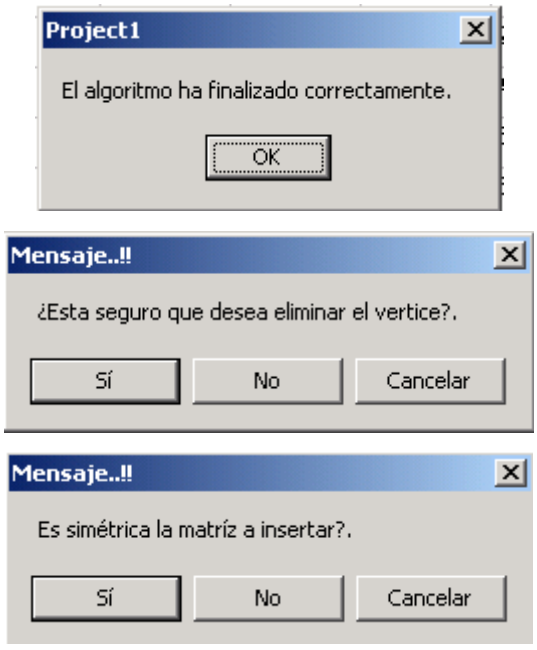
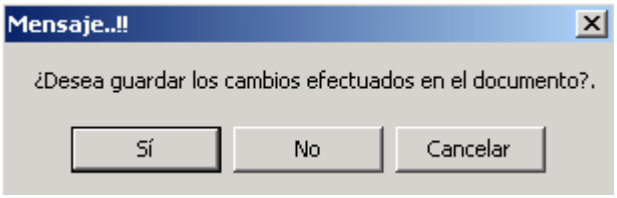


Figura 9. Vista del Diseño de la Interfaz.





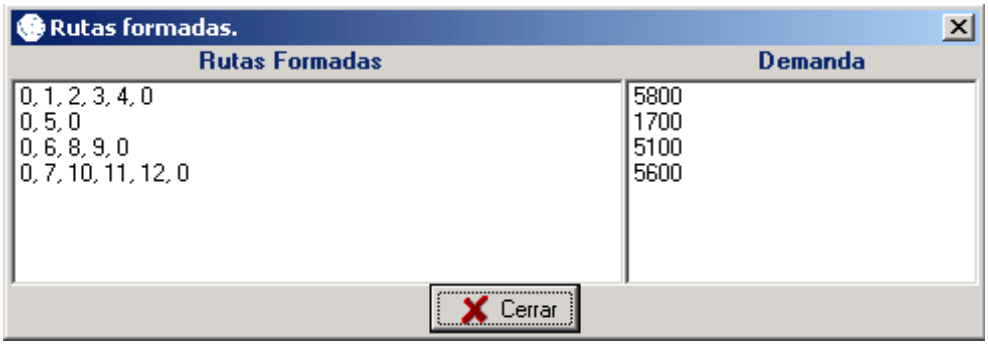


Figura 10. Tipos de mensajes del sistema.

2.6.3.2 – Concepción general de la ayuda.

El sistema proporciona una ayuda contextual que se muestra al pasar el mouse sobre cada botón de la barra de herramientas, de esta manera se le comunica al usuario cómo utilizar el sistema. Muestra de lo anteriormente mencionado lo constituye la siguiente imagen.

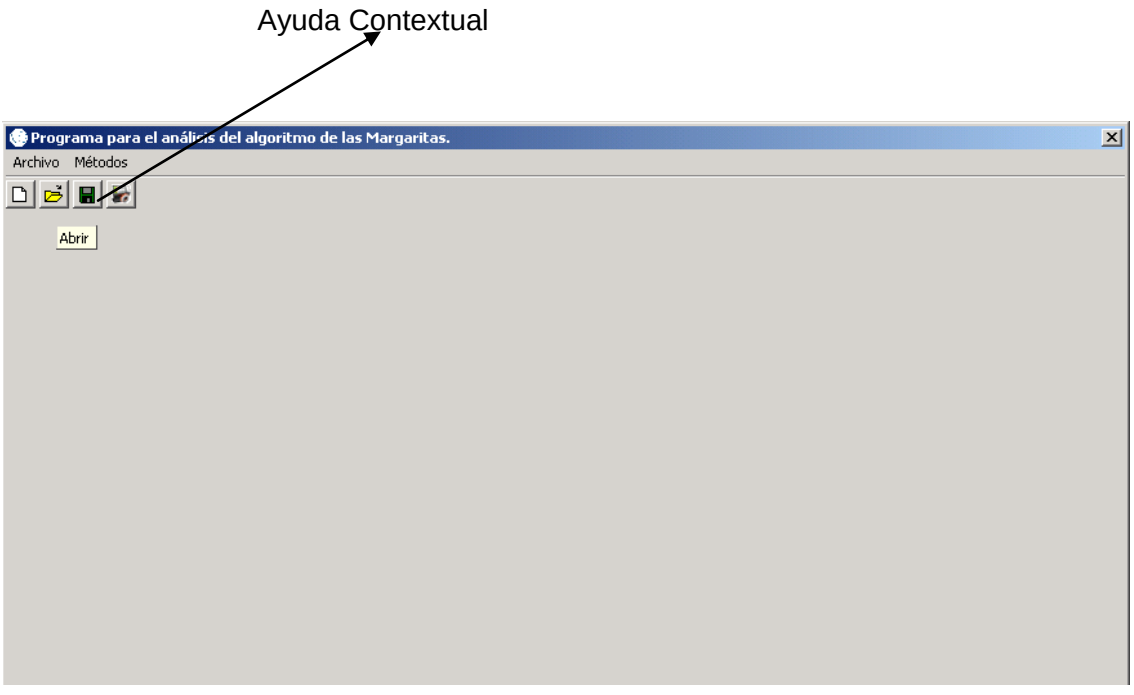


Figura 11. Ayuda del sistema.

2.6.3.3 – Tratamiento de excepciones.

En el diseño para la interfaz del sistema se tuvo siempre presente que toda persona en cualquier entorno en el que se desenvuelva puede cometer errores y precisamente dicho diseño está dirigido a evitarlos, siempre teniendo en cuenta la creación de interfaces amigables. Los mensajes de error que muestra el sistema se hacen visibles en un lenguaje de fácil comprensión para los usuarios.

2.6.3.4 – Estándares de codificación.

Para realizar el ejercicio de la buena programación es importante seguir un buen estilo de código. Es notablemente visible en el momento de realizar modificaciones al código, en aspectos como la comprensión y la reducción de tiempo y esfuerzo, haber escrito código que cumpla con las anteriores cuestiones.

Existen disímiles acápites que pueden hacer un código más legible; algunos de ellos son el uso de una indentación coherente, de comentarios informativos y el empleo de nombres descriptivos, entre otros. Se detallan seguidamente algunas convenciones tomadas con relación a estos acápites.

Indentación

En el caso de la indentación, se establece que todas las líneas dentro de un procedimiento o función, estarán indentadas con respecto a la instrucción que encabeza a este, lo mismo ocurre con todas las líneas que conformen el cuerpo de un ciclo estructural condicional.

Comentarios

Un buen comentario añade información al código de una manera clara y ayuda a entender el objetivo del mismo. Se tomó como regla, por tanto, comentar todos los procedimientos y funciones al principio de los mismos para explicar cómo se deben usar sin necesidad de leer el código.

2.7 – Conclusiones.

Con todo lo analizado en el presente capítulo se arribaron a las siguientes conclusiones:

Están presentes en la solución 13 requerimientos funcionales los cuales fueron agrupados en 8 casos de uso, descritos posteriormente. También se identificó el actor del sistema y la labor que realiza en el mismo. Se expusieron algunas de las especificaciones sobre principios de codificación, diseño y el tratamiento de errores, las que ayudan a una mejor implementación del producto informático.

Capítulo 3: - Estudio de factibilidad y validación del sistema.

3.1- Introducción.

En este capítulo se trata acerca de diferentes aspectos relacionados con el estudio de la factibilidad del producto. Se estiman el esfuerzo humano y el tiempo de desarrollo que se requieren para la elaboración del mismo, constituyendo estos la base para el análisis de la factibilidad, así como los costos y los beneficios tangibles e intangibles que reporta la utilización del sistema. Estas estimaciones serán realizadas a través del método Planificación basada en Casos de Uso del modelo COCOMO II.

COCOMO es una herramienta utilizada para la estimación de algunos parámetros (costes en personas, tiempo,…) en el diseño y construcción de programas y de la documentación asociada requerida para desarrollarlos, operarlos y mantenerlos, es decir, en la aplicación práctica de la Ingeniería del Software[28].

3.2 - Planificación basada en casos de uso.

Método de estimación del esfuerzo de desarrollo de un producto de software a partir de los Casos de Uso y algunos factores de complejidad técnica y ambiente que influyen en el desarrollo[28].

Pasos para la Estimación:

1. Calcular los Puntos de Casos de Uso **(PCU)**

1.1 Calcular el Factor de Peso de los Actores **(FPA)**

1.2 Calcular el Factor de Peso de los Casos de Uso **(FPCU)**

2. Calcular los Puntos de Casos de Usos Ajustados **(PCUA)**

2.1 Calcular el Factor de Complejidad Técnica **(FCT)**

2.2 Calcular el Factor de Ambiente **(FA)**

3. Calcular el Esfuerzo de desarrollo **(E)**

3.2.1 - Obtención de los Puntos de Casos de Uso sin ajustar.

La primera fase para la estimación consiste en el cálculo de los Puntos de Casos de Uso sin ajustar. Este valor, se calcula a partir de la ecuación.

PCU = FPA + FPCU

Donde:

PCU: Puntos de Casos de Uso **(UUCP)**

FPA: Factor de Peso de los Actores **(UAW)**

FPCU: Factor de Peso de los Casos de Uso **(UUCW)**

3.2.1.1 - Obtención del Factor de Peso de los Actores sin ajustar (FPA).

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Actores se establece teniendo en cuenta en primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema. Los criterios son:

Tipo	Descripción	Peso	Cant*Peso
Simple	Otro sistema mediante una interfaz de programación.	1	0 * 1
Medio	Otro sistema mediante un protocolo o una interfaz basada en texto.	2	0 * 2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	1 * 3
		Total:	3

Tabla 1 . Factor de Peso de los Actores (FPA)

3.2.1.2 - Obtención del Factor de Peso de CU (FPCU).

Este factor se calcula teniendo en cuenta la cantidad de casos de usos y su complejidad.

Complejidad: Se determina a partir de la cantidad de transacciones que se realizan.

Transacción: Es una secuencia atómica de actividades, las cuales se realizan completamente o no se realiza ninguna.

Tipo de CU	Transacciones	Peso
Simple	menos de 4	5
Medio	de 4 a 7	10
Complejo	más de 7	15

Tabla 2. Factor de Peso de CU (FPCU)

Para calcular UUCW:

No.	Caso de Uso	No. Transiciones	Tipo de CU	Peso
1	Crear nueva tabla.	2	Simple	5
2	Abrir documento.	2	Simple	5
3	Salvar documento.	2	Simple	5
4	Gestionar vértice.	5	Medio	10
5	Gestionar distancia entre clientes.	6	Medio	10
6	Gestionar demanda por cliente.	6	Medio	10
7	Aplicar algoritmo de las Margaritas.	2	Simple	5
8	Salir de la aplicación.	1	Simple	5

Tabla 3. Variables por casos de uso.

3.2.1.3 - Obtención de los Puntos de Casos de Uso sin ajustar (PCU).

Se tienen 5 casos de uso con la clasificación simple y 3 casos de uso con la clasificación media por lo que se le aplican como factor de peso 5 y 10 respectivamente.

PCU = FPA + FPCU

Luego: **PCU** = 3 + 55

PCU = 58

3.2.2 - Obtención de los Puntos de Casos de Uso ajustados.

Después de calculados los PCU (sin ajustar) estos se deben ajustar teniendo en cuenta un grupo de factores técnicos y ambientales. Este valor, se calcula a partir de la ecuación.

PCUA = PCU X FCT X FA

Donde:

PCUA: Puntos de Casos de Usos Ajustados **(UCP)**

FCT: Factor de Complejidad Técnica **(TCF)**

FA: Factor de Ambiente **(EF)**

3.2.2.1 - Obtención del Factor de Complejidad Técnica (FCT).

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores (13) que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante.

Sistema distribuido, tiempo de respuesta, usuario final, procesamiento interno, reutilización , facilidad de instalación, facilidad de uso, portabilidad , facilidad de cambio, concurrencia, objetivos especiales de seguridad, provee acceso directo a terceras partes, facilidades especiales de entrenamiento. Se estima de la forma:

$FCT = 0.6 + 0.01 \times \Sigma (Peso_i \times Valor_i)$

Factor	Descripción	Peso	Valor	$\Sigma (Peso_i * Valor_i)$
T1	Sistema distribuído	2	4	8
T2	Objetivos de performance o tiempo de respuesta	1	4	4
T3	Eficiencia del usuario final	1	5	5
T4	Procesamiento interno complejo	1	5	5
T5	El código debe ser reutilizable	1	4	4
T6	Facilidad de instalación	0.5	4	2
T7	Facilidad de uso	0.5	4	2
T8	Portabilidad	2	4	8
T9	Facilidad de cambio	1	4	4
T10	Concurrencia	1	2	2
T11	Incluye objetivos especiales de seguridad	1	0	0

T12	Provee acceso directo a terceras partes	1	3	3
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	3	3
			Total:	50

Tabla 4. Factores Técnicos

FCT = 0.6 + 0.01 * 50

FCT = 1.1

3.2.2.2 - Obtención del Factor de Ambiente (FA).

Después se multiplican por los pesos asociados a cada factor por el valor puesto y se aplica la fórmula para obtener el factor de Ambiente. Luego ya se puede proceder al cálculo. Multiplicando los desajustados por los factores de complejidad técnica y de ambiente. Se estima de forma similar al FCT:

FA = 1.4 - 0.03 x Σ ($Peso_i$ x $Valor_i$)

Factor	Descripción	Peso	Valor	Σ ($Peso_i$ * $Valor_i$)
E1	Familiaridad con el modelo de proyecto utilizado	1.5	4	6
E2	Experiencia en la aplicación	0.5	4	2
E3	Experiencia en orientación a objetos	1	5	5
E4	Capacidad del analista líder	0.5	4	2
E5	Motivación	1	5	5
E6	Estabilidad de los requerimientos	2	4	8
E7	Personal a tiempo compartido	-1	0	0
E8	Dificultad del lenguaje de programación	-1	4	-4
			Total:	24

Tabla 5. Factor Ambiente.

FA = 1.4 - 0.03 * 24

FA = 0.68

3.2.2.3 - Obtención de los Puntos de Casos de Uso ajustados (PCUA).

Se tienen 5 casos de uso con la clasificación simple y 3 casos de uso con la clasificación media por lo que se le aplican como factor de peso 5 y 10 respectivamente.

PCUA = PCU X FCT X FA

Luego: **PCUA** = 58 **X** 1.1 **X** 0.68

PCUA = 43.384

3.2.3 - Calcular el Esfuerzo de desarrollo.

Convertir los Puntos de Casos de Uso Ajustados a Esfuerzo de desarrollo.

E = PCUA x FC

Donde:

FC: Factor de Conversión

El valor de FC según Karner es de 20 H/H

Puede ser calibrado entre 15 y 30 H/H en dependencia de los FA.

Karner originalmente sugirió que cada Punto de Casos de Uso requiere 20 horas-hombre. Posteriormente, surgieron otros refinamientos que proponen mayor detalle, según el siguiente criterio:

Sumar el número de factores de ambiente que están por debajo de la media de E1 a E6 y los que están por encima de E7 y E8

- Si la suma da menor o igual a 2 se usa como factor de conversión 20 horas-hombre.
- Si la suma da 3 o 4 se usa como factor de conversión 28 horas-hombre.
- Mayor o igual a que 5 se debe ajustar pues se considera de alto riesgo con posibilidad de fracaso.

Media = 3

Total EF = Cant EF < 3 (entre E1 – E6) + Cant EF > 3 (entre E7 – E8)

Como Total EF = 0 + 1

Total EF = 1

CF = 20 horas-hombre (porque Total EF = 1)

Luego **E = PCUA x FC**

E = 43.384 * 20 horas-hombre

E = 867.68 horas-hombre

La Programación ocupa el 40 % del proyecto. Existe una aproximación estimada de la distribución del esfuerzo en función de las etapas del desarrollo de software.

Tipo de Actividad	Porciento	Esfuerzo (H/H)
Análisis	10	86.768
Diseño	20	173.536
Implementación	40	347.072
Pruebas	15	130.152
Sobrecarga (Otras actividades)	15	130.152
Total	100	867.68

Tabla 6. Distribución del esfuerzo estimado entre los flujos de trabajo de RUP

3.2.4 – Duración.

Trabajando los 26 días al mes y 9 horas al día como promedio, podemos decir que:

Duración (días) = Total de Horas/Hombre entre 9 horas al día = 867.68/9 = 96 días.

Duración (meses) = Total de días/26 días por mes = 96/26 = 3.69 ≈ 4 meses.

3.2.5 – Cálculo de costos.

Tomando como salario promedio mensual 415.00 MN

Costo = 4 * 415 = **1660 MN**

3.3 – Beneficios tangibles e intangibles.

Al desarrollar un producto informático se tiene asociado un costo y el realizarlo o no está en dependencia de los beneficios que el mismo pueda reportar. Los beneficios pueden ser económicos y de orden social. Los primeros, según al tipo que correspondan presentan diversas clasificaciones, estas son: directos, indirectos, externos (*beneficios tangibles*) y *beneficios intangibles*. Entiéndase como beneficios tangibles aquellos efectos que se alcanzan como consecuencia directa de la implantación del proyecto de manera que resulten perceptibles económicamente para la entidad generadora del mismo. Los beneficios intangibles son aquellos efectos apreciables por la comunidad como perjuicio o beneficio. Se debe destacar que los últimos mencionados tienen tanta importancia como los primeros.

Dentro de los beneficios tangibles que se desprenden del proyecto se pueden mencionar el incremento de la productividad en la actividad que se aplique ya que esta herramienta podrá ser utilizada en cualquier entidad estatal de nuestro país para beneficio de la misma y el ahorro de sus recursos.

Como beneficios intangibles se pueden citar la obtención de una novedosa herramienta para la optimización de tareas y además el ahorro de tiempo que significa la realización de este algoritmo de forma manual y la reducción de las posibilidades de errores que esto trae consigo. Otros beneficios intangibles asociados a este proyecto son mayor comodidad de los usuarios, mejor imagen de la institución y mejoramiento de las condiciones de trabajo del personal.

3.4 – Análisis de los costos y beneficios.

Sin lugar a dudas, la utilización de esta herramienta informática traería notables ventajas en cualquier sector que se desee y requiera su utilización, dada la posibilidad que brinda para la optimización de las tareas en que se aplique, lo que se traduce en un incremento productivo. Además de los beneficios tanto tangibles como intangibles, descritos anteriormente.

Un acápite muy importante para determinar la factibilidad de este producto, independientemente de los beneficios aparejados al mismo, es el costo, el cual fue estimado en 1660 MN, además supone un tiempo de desarrollo de 4 meses. Para la realización del producto no se incurrió en gastos adicionales de

equipamiento, materiales de oficina, compra de otros sistemas necesarios, ni de herramientas de desarrollo, además no hubo necesidad de contratar personal calificado que realizara el trabajo imprescindible para obtener el producto final.

Al analizar los costos se puede apreciar que los mismos son relativamente bajos, este aspecto, unido a los grandes beneficios que resultarían de la realización y posterior utilización del software propuesto, determina la factibilidad del desarrollo del producto.

3.5 – Validación del problema.

Para llevar a cabo la validación del problema se aplicó una encuesta a los estudiantes que han cursado la asignatura de Investigación de Operaciones II en la Universidad de Cienfuegos. Los resultados que arroje dicha encuesta se analizarán mediante el programa estadístico SPSS v.15.0. Se utilizará la prueba de hipótesis para la comparación de los resultados, observando así si existe acuerdo o no entre los encuestados.

La encuesta fue diseñada con el objetivo de motivar mediante preguntas sencillas, no muy extensas y legibles al personal encuestado. En el Anexo 2 se muestra la encuesta realizada.

Para que la muestra sea significativa se deben tomar como mínimo 27 encuestados por lo que para la elaboración de esta se tuvieron en cuenta 30, siendo todos estudiantes de 5to año de la carrera de Ingeniería Informática en la Universidad de Cienfuegos.

Para la continuación del análisis se realizó la prueba no paramétrica W de Kendall con el objetivo de demostrar estadísticamente que existe acuerdo entre los evaluados.

El cálculo del número de elementos de la muestra, según el Muestreo Aleatorio Simple de Proporciones, utiliza la fórmula siguiente:

$$n = (Npq) : ((N - 1)\, D + pq)$$

Donde:

p = proporción de elementos que cumplen la condición.

q = 1- p proporción de elementos que no cumplen la condición.

$D = B^2/4$ donde B = Error dado por el investigador.

N = Tamaño de la población.

Ahora se demuestra que si p = q = 0.5, entonces, se encuentra el máximo número de elementos de la muestra.

Se tomaron los siguientes datos:

p = q = 0.5

N = 48 Estudiantes (Total de la Población)

B = Error de muestreo = 0.13

Aplicando la fórmula anteriormente estudiada se calcula n = 30 estudiantes (muestra escogida, en forma aleatoria, para hacer el análisis estadístico del Producto informático).

3.5.1 – Resultados del procesamiento estadístico.

Los resultados de la encuesta se analizan mediante el programa estadístico SPSS v.15.0. Además se aplica la prueba de hipótesis para apreciar la existencia o no de acuerdo entre los encuestados (Prueba no paramétrica de Kendall).

El análisis estadístico arrojó los siguientes resultados:

	N	Minimum	Maximum	Mean	Std. Deviation
Utilidad del Producto Informático en forma general	30	1	3	1,90	,607
Utilidad del Producto Informático como apoyo a la Enseñanza de la Teoría de Redes	30	1	3	1,93	,640
Relacionado con otras Aplicaciones sobre Redes en cuanto al contenido	30	1	3	1,50	,731

Relacionado con otras Aplicaciones sobre Redes en cuanto a la presentación	30	1	3	2,23	,568
En cuanto al uso	30			1,67	,547
En qué radican las ventajas	30	1	2	1,70	,466
Valor dado a la Aplicación en escala del 1 al 5	30	3	5	4,40	,563
Valid N (listwise)	30				

Tabla 7. Estadística descriptiva de todas las variables.

Se observa en la Tabla 7 que todas las variables, excepto la última, se encuentran en el rango de 1 a 3 (Muy Bueno a Regular) con un valor medio muy cercano a 2 (Bueno) y una desviación estándar pequeña, lo que indica la aceptación de los encuestados por el software.

La última variable Valor presenta un rango entre 3 y 5 con una media muy cercana a 5 (máxima puntuación posible) y una desviación estándar muy pequeña.

N	30
Kendall's W(a)	,567
Chi-Square	102,031
df	6
Asymp. Sig.	,000

Tabla 8. Prueba de Kendall.

3.5.2 – Comprobación de la existencia de acuerdos entre los encuestados.

Prueba de Hipótesis:

H_0: Los encuestados están de acuerdo con las respuestas a las preguntas sobre el Software.

H_1: Los encuestados no están de acuerdo con las respuestas a las preguntas sobre el software.

Para decidir cuál hipótesis aceptar se debe comparar el nivel de significación con la significación asintótica del estadígrafo, si esta última es menor que la significación entonces se acepta H_0.

En la Tabla 9 podemos comprobar que se cumple la Hipótesis Alternativa H_0. De acuerdo a la prueba de concordancia W de Kendall que se realizó tomando en cuenta la hipótesis H_0 (Los encuestados están de acuerdo con las respuestas a las preguntas sobre el Software) y H_1 (Los encuestados no están de acuerdo con las respuestas a las preguntas sobre el Software) para así decidir cuál de las hipótesis aceptar, se adoptó un nivel de significación de 0.05 mayor que la significación asintótica calculada en el SPSS de 0.00, por lo que se puede concluir que la hipótesis a aceptar es H_0 y sí existe un acuerdo entre las opiniones de los encuestados.

3.6 – Conclusiones.

Posteriormente al análisis del presente capítulo se concluye que:

- Es factible realizar el sistema propuesto, teniendo en cuenta el costo y los beneficios que aportará con su implementación; resultando así un costo de 1660 MN y desarrollándose por una persona en un tiempo de 4 meses.
- En la validación del sistema la hipótesis a aceptar es H_0 y sí existe un acuerdo entre las opiniones de los encuestados luego de ser analizados los resultados en el programa estadístico SPSS.

Conclusiones.

Como resultado de las investigaciones teóricas y de la elaboración del producto informático se arribaron a las siguientes conclusiones:

- Se analizaron sistemas existentes en la actualidad para el trabajo con la teoría de redes, como son: QSB, QmWin2 y POM, determinándose que los mismos no poseen el Algoritmo de las Margaritas para el trabajo con la Teoría de Redes.

- Se logró una eficiente estructuración y concepción del sistema, gracias al empleo de los diferentes flujos de trabajos ofrecidos por la metodología RUP.

- Se analizó, diseñó e implementó una herramienta que da cumplimiento al objetivo general del presente trabajo de diploma.

- A través de un estudio de factibilidad realizado en el Capitulo 3, se concluyó que es factible la implementación del sistema propuesto.

- Se validó la aplicación, arrojando resultados satisfactorios y una buena aceptación.

Recomendaciones.

Con el objetivo de proveer al sistema desarrollado una mayor funcionalidad, de acuerdo con las necesidades reales de estudiantes y profesores que utilizan la herramienta, se recomienda:

- ✓ Realizar un nuevo análisis de necesidades a partir de la nueva propuesta, para obtener nuevos módulos para el sistema.

- ✓ Implementar más algoritmos enmarcados en la Teoría de Redes con el objetivo de fortalecerlo y alcanzar una mejora continua ampliando sus funcionalidades.

- ✓ Dotar al sistema con opciones para importar documentos en formato XML, Excel, etc.

- ✓ Implementar una solución gráfica para representar el algoritmo de las Margaritas.

- ✓ Implantar el sistema en la Universidad de Cienfuegos, como herramienta de apoyo a los estudiantes y profesores que cursen o impartan la asignatura Investigación de Operaciones II.

Referencias bibliográficas.

[1] R. R. A. G. Raúl René Alpízar Gamboa, “EUREKA. Producto Informático para la Solución a problemas de Modelos de Redes.”

[2] “apendice1.pdf.”

[3] “Teoría de redes - EcuRed.” [Online]. Available: http://www.ecured.cu/index.php/Teor%C3%ADa_de_redes. [Accessed: 09-Feb-2012].

[4] RICARDO ALBERTO HINCAPIÉ Ingeniero Electricista Estudiante Maestría Ingeniería Eléctrica ricardohincapie@utp.edu.co CARLOS ALBERTO RÍOS PORRAS Ingeniero Electricista Estudiante Maestría RAMÓN ALFONSO GALLEGO Profesor Facultad de Ingeniería Eléctrica Universidad Tecnológica de Pereira ralfonso@utp.edu.co, “TÉCNICAS HEURÍSTICAS APLICADAS AL PROBLEMA DEL VIAJERO VENDEDOR (TSP).” .

[5] “Evolución de las TIC: Oportunidades y amenazas sociales.” [Online]. Available: http://www.ibermatica.com/ibermatica/eventos/2006/mtevolucionticsoportunidadesamenazas.

[6] M. O. C. P. MsC. Osvaldo Chang Pereira, “El uso de las TIC en las Sedes Universitarias Municipales en Cuba y su influencia en el estrés de los estudiantes.” [Online]. Available: : http://www.monografias.com/trabajos77/uso-tics-sedes-universitarias-municipales-cuba/uso-tics-sedes-universitarias-municipales-cuba2.shtml.

[7] MSc. Esteban Negrín Barroso MSc. Milagro Rodríguez Andino, “Las TIC y su papel en los procesos de Innovación en el ámbito económico local.” 10-Dec-2011.

[8] “Sistema informático - EcuRed.” [Online]. Available: http://www.ecured.cu/index.php/Sistema_inform%C3%A1tico. [Accessed: 06-Mar-2012].

[9] Mateos Ginés García, *Apuntes Algoritmos y Estructura de Datos*. .

[10] “Teoría de Grafos.pdf.”.

[11] “Grafos no Dirigidos.”

[12] “Grafos Dirigidos.”

[13] “Páginas Blancas de la Teoría de Grafos.” [Online]. Available: http://www.math.gatech.edu/~sanders/graphtheory/.

[14] Laredo González , Pilar Felipe, *Introducción a la Teoría y Aplicaciones de las Redes.* .

[15] “Grafos.”

[16] Kumar Grama, Grupta Karypis, *Introduction to Parallel Computing. Design and Analysis of Algorithms*. .

[17] *QSB*. .

[18] *Qmwin2*. .

[19] *POM*. .

[20] “Programación Orientada a Objetos,” 2010. [Online]. Available: http//www.ciberaula.com.

[21] “Programación Orientada a Objetos Usando C++.”

[22] Jacobson I.; Booch G. y Rumbaugh J., *El Proceso Unificado de Desarrollo de Software*. Addison-Wesley, 2010.

[23] Jacobson Ivar, *El Lenguaje Unificado de Modelado: Manual de referencia.* Madrid: Pearson Educación S.A, 2000.

[24] Jacobson Ivar, “Applying UML in The Unified Process,” 04-Dec-2010. [Online]. Available: http://www.rational.com/uml.

[25] Stroustrup B., *The C++ Programming Language*, 3rd ed. Addison-Wesley, 1997.

[26] “Rational Rose.” .

[27] “IDE. Entorno de Desarrollo Integrado.” [Online]. Available: http://laysquad.foroactivo.com.

[28] Francisco Ruiz González, *Modelo de Estimación de Costes para proyectos de software*. Universidad de Castilla, 1999.

Bibliografía.

[1] “apendice1.pdf.”

[2] Jacobson Ivar, “Applying UML in The Unified Process,” 04-Dec-2010. [Online]. Available: http://www.rational.com/uml.

[3] Mateos Ginés García, *Apuntes Algoritmos y Estructura de Datos*. .

[4] Jacobson Ivar, *El Lenguaje Unificado de Modelado: Manual de referencia.* Madrid: Pearson Educación S.A, 2000.

[5] Jacobson I.; Booch G. y Rumbaugh J., *El Proceso Unificado de Desarrollo de Software*. Addison-Wesley, 2010.

[6] M. O. C. P. MsC. Osvaldo Chang Pereira, “El uso de las TIC en las Sedes Universitarias Municipales en Cuba y su influencia en el estrés de los estudiantes.” [Online]. Available: http://www.monografias.com/trabajos77/uso-tics-sedes-universitarias-municipales-cuba/uso-tics-sedes-universitarias-municipales-cuba2.shtml.

[7] R. R. A. G. Raúl René Alpízar Gamboa, “Eureka. Producto Informático para la Solución a problemas de Modelos de Redes.”

[8] “Evolución de las TIC: Oportunidades y amenazas sociales.” [Online]. Available: http://www.ibermatica.com/ibermatica/eventos/2006/mtevolucionticsoportunidadesamenazas.

[9] “Grafos.”

[10] “Grafos Dirigidos.”

[11] “Grafos no Dirigidos.”

[12] “IDE. Entorno de Desarrollo Integrado.” [Online]. Available: http://laysquad.foroactivo.com.

[13] “Introducción a la Investigación de Operaciones.” [Online]. Available: http://antiguo.itson.mx/dii/elagarda/apagina2001/PM/uno.html. [Accessed: 09-Feb-2012].

[14] Laredo González, Pilar Felipe, *Introducción a la Teoría y Aplicaciones de las Redes.* .

[15] Kumar Grama, Grupta Karypis, *Introduction to Parallel Computing. Design and Analysis of Algorithms*. .

[16] MSc. Esteban Negrín Barroso MSc. Milagro Rodríguez Andino, “Las TIC y su papel en los procesos de Innovación en el ámbito económico local.” 10-Dec-2011.

[17] Francisco Ruiz González, *Modelo de Estimación de Costes para proyectos de software*. Universidad de Castilla, 1999.

[18] “Páginas Blancas de la Teoría de Grafos.” [Online]. Available: http://www.math.gatech.edu/~sanders/graphtheory/.

[19] *POM*. .

[20] “Programación Orientada a Objetos,” 2010. [Online]. Available: http//www.ciberaula.com.

[21] “Programación Orientada a Objetos Usando C++.”

[22] *Qmwin2*. .

[23] *QSB*. .

[24] “Rational Rose.”

[25] “Sistema informático - EcuRed.” [Online]. Available: http://www.ecured.cu/index.php/Sistema_inform%C3%A1tico. [Accessed: 06-Mar-2012].

[26] Ricardo Alberto Hincapié Ingeniero Electricista Estudiante Maestría Ingeniería Eléctrica ricardohincapie@utp.edu.co Carlos Alberto Ríos Porras Ingeniero Electricista Estudiante Maestría Ramón Alfonso Gallego Profesor Facultad de Ingeniería Eléctrica Universidad Tecnológica de Pereira ralfonso@utp.edu.co, “Técnicas Heurísticas Aplicadas Al Problema Del Viajero Vendedor (TSP).” .

[27] “Teoría de Grafos.pdf.”

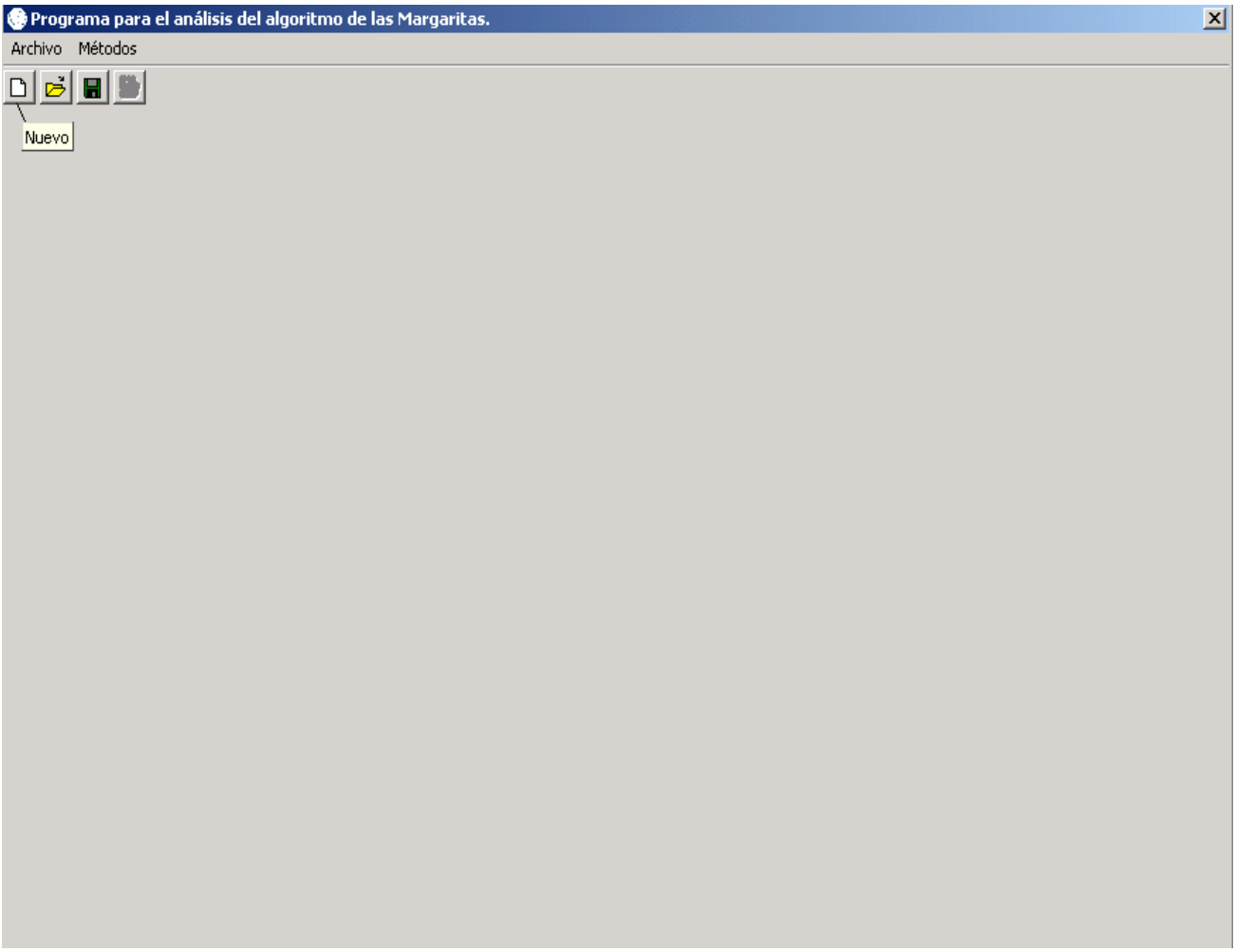
[28] “Teoría de redes - EcuRed.” [Online]. Available: http://www.ecured.cu/index.php/Teor%C3%ADa_de_redes. [Accessed: 09-Feb-2012].

[29] Stroustrup B., *The C++ Programming Language*, 3rd ed. Addison-Wesley, 1997.

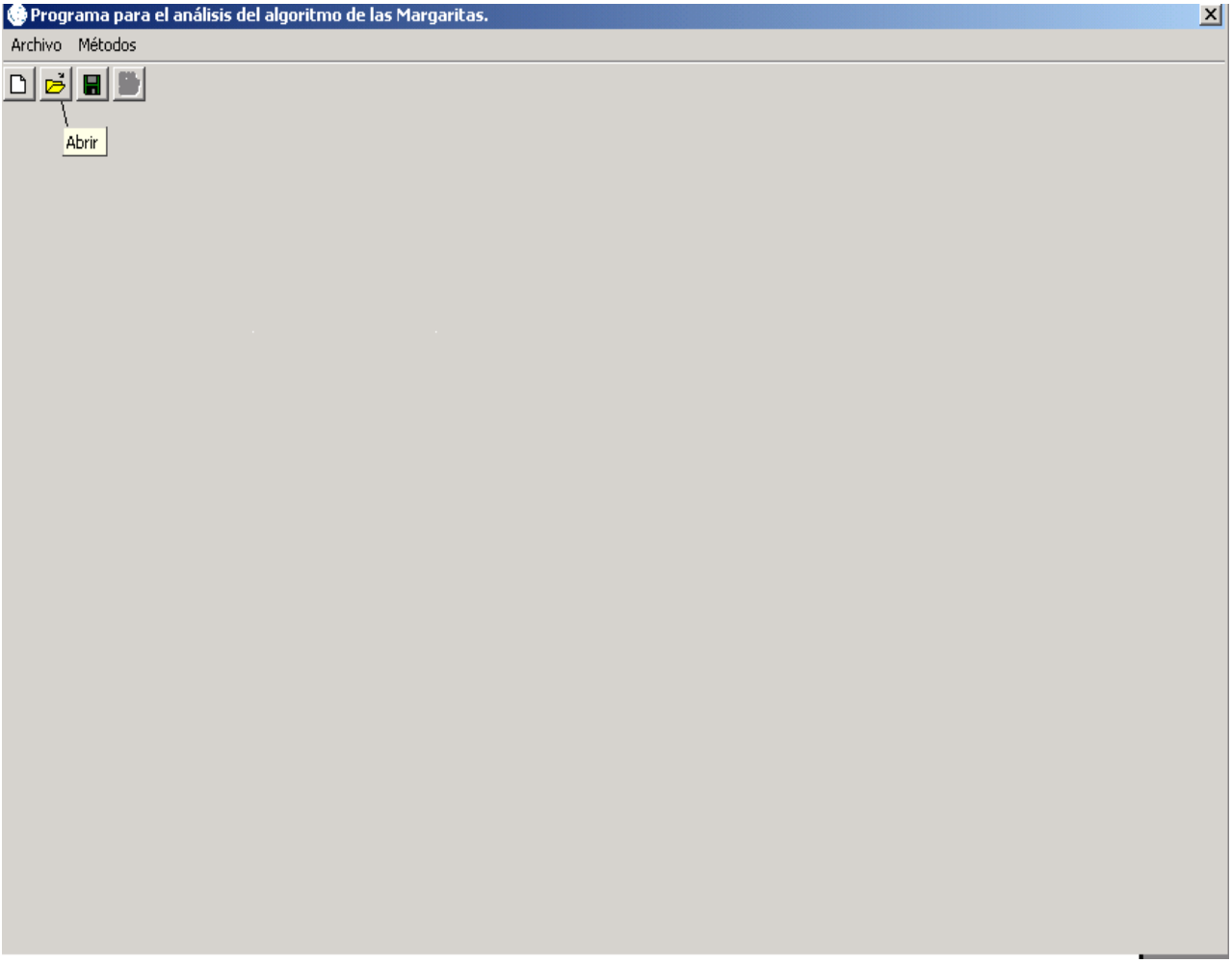
Anexos.

Anexo 1. Prototipos.

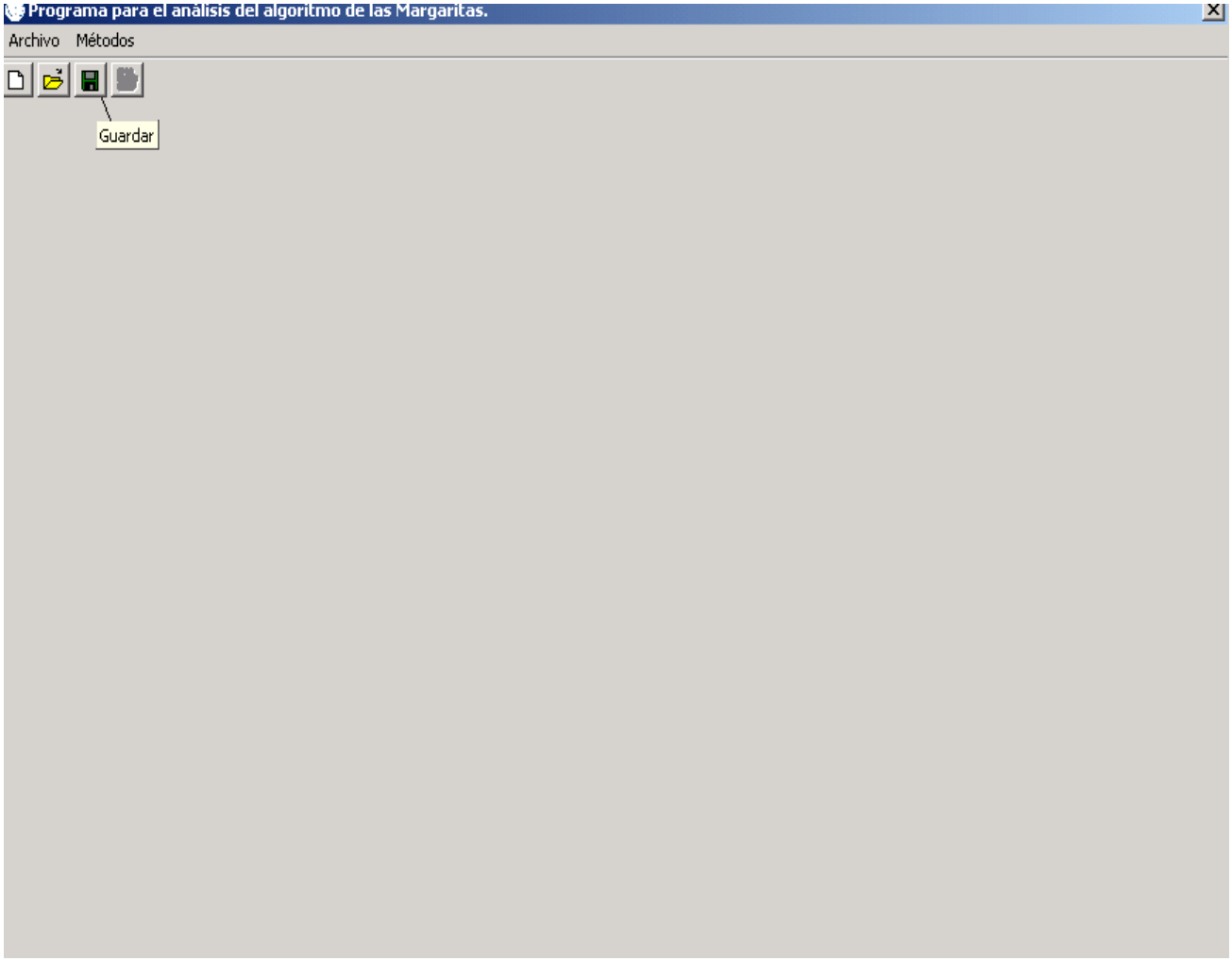
Prototipo#1 Crear nueva tabla.



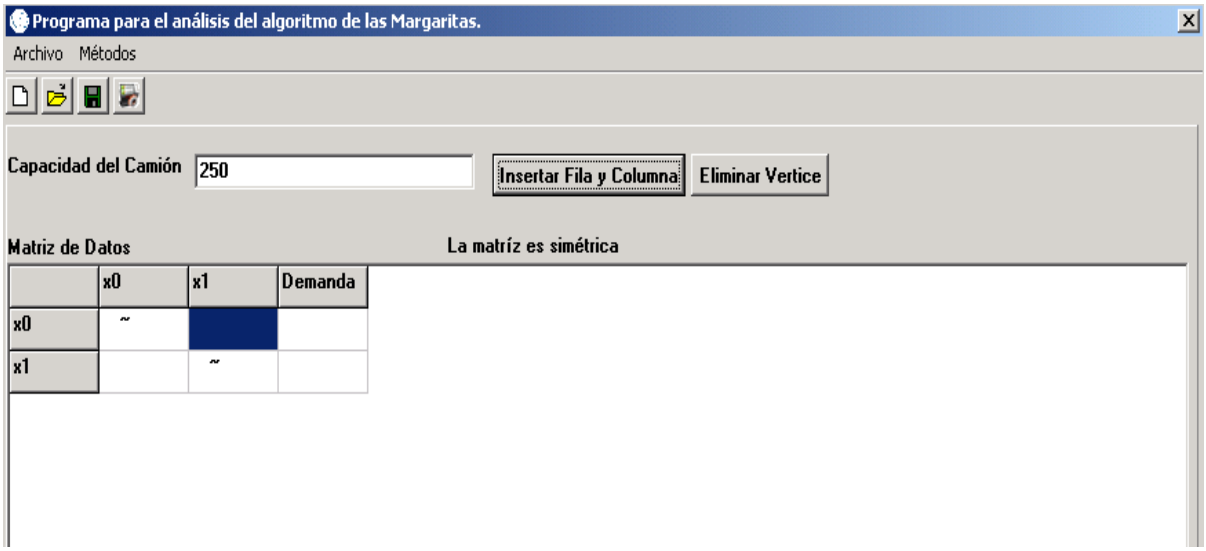
Prototipo#2 Abrir documento.

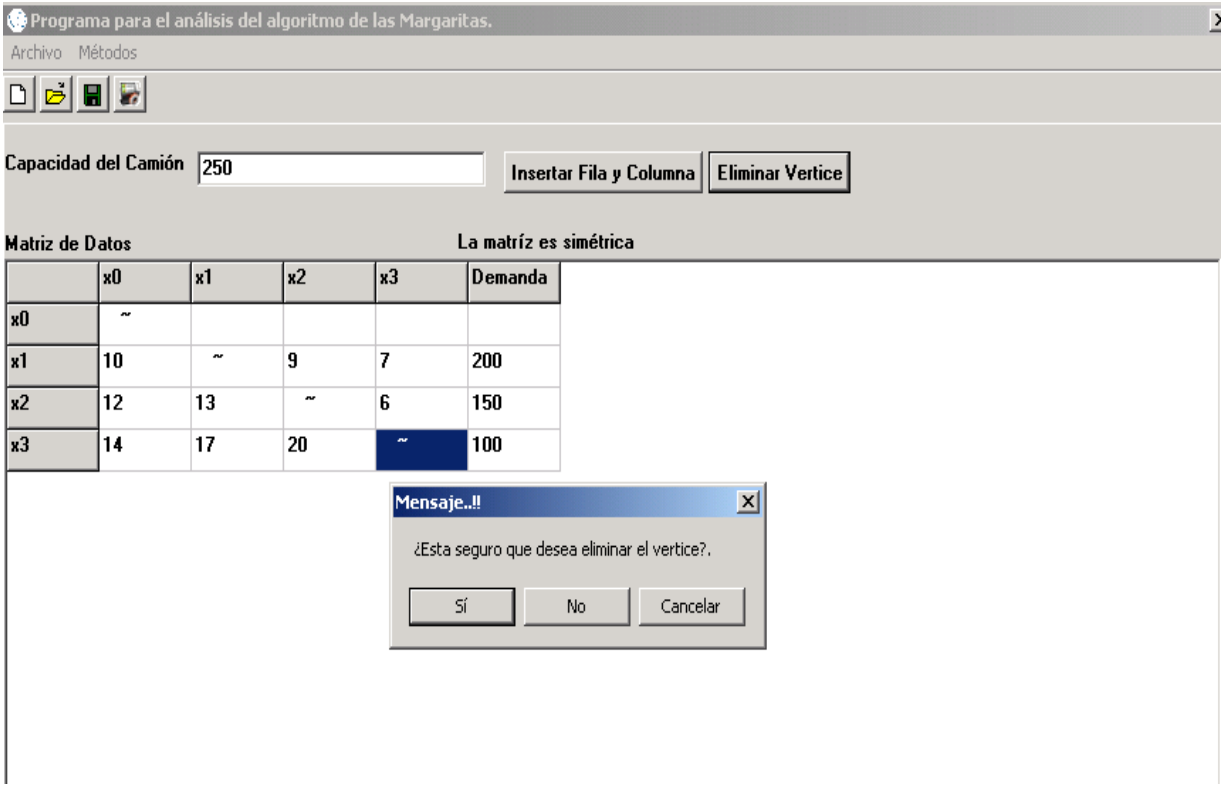


Prototipo#3 Salvar documento.

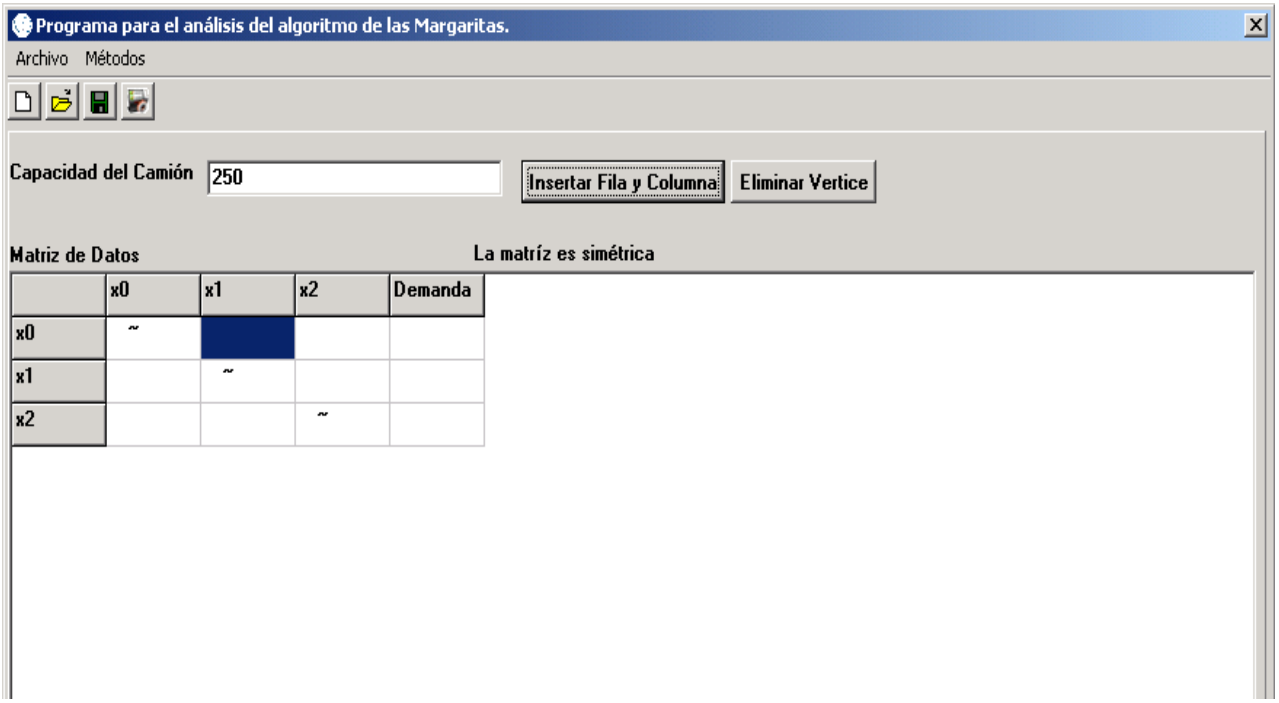


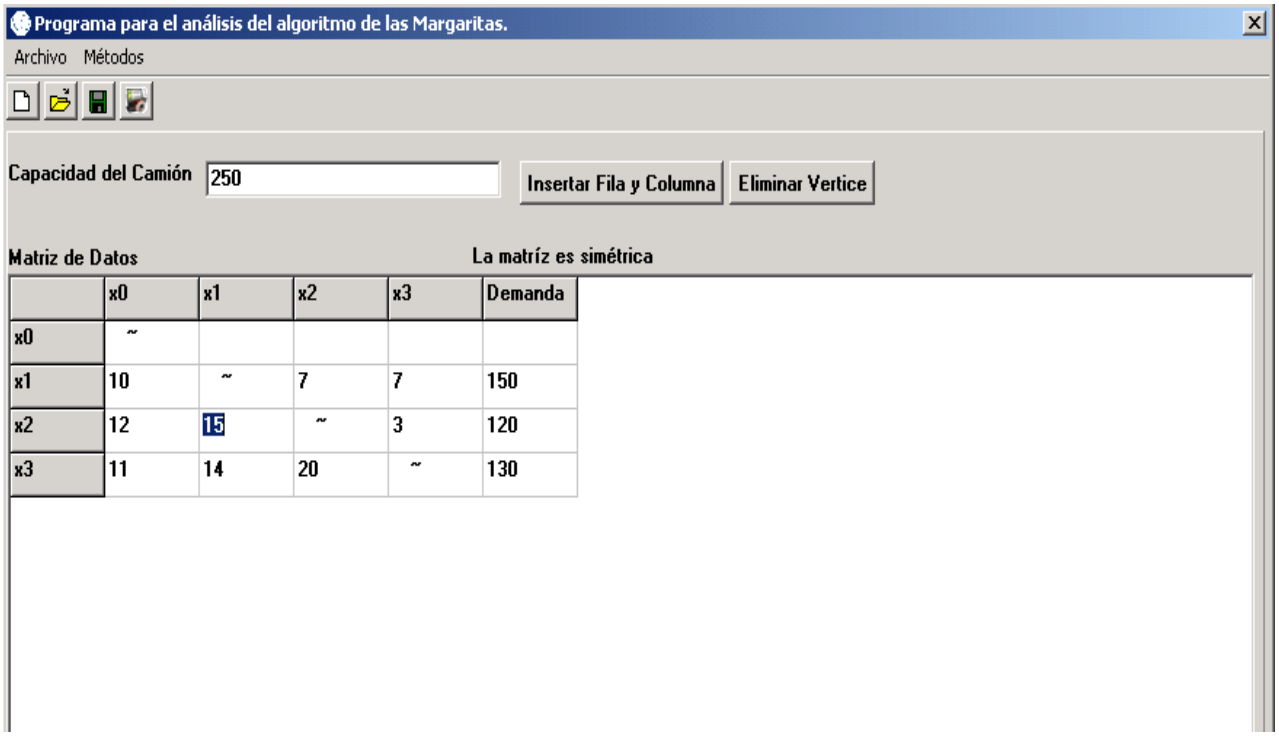
Prototipo#4 Gestionar vértice.



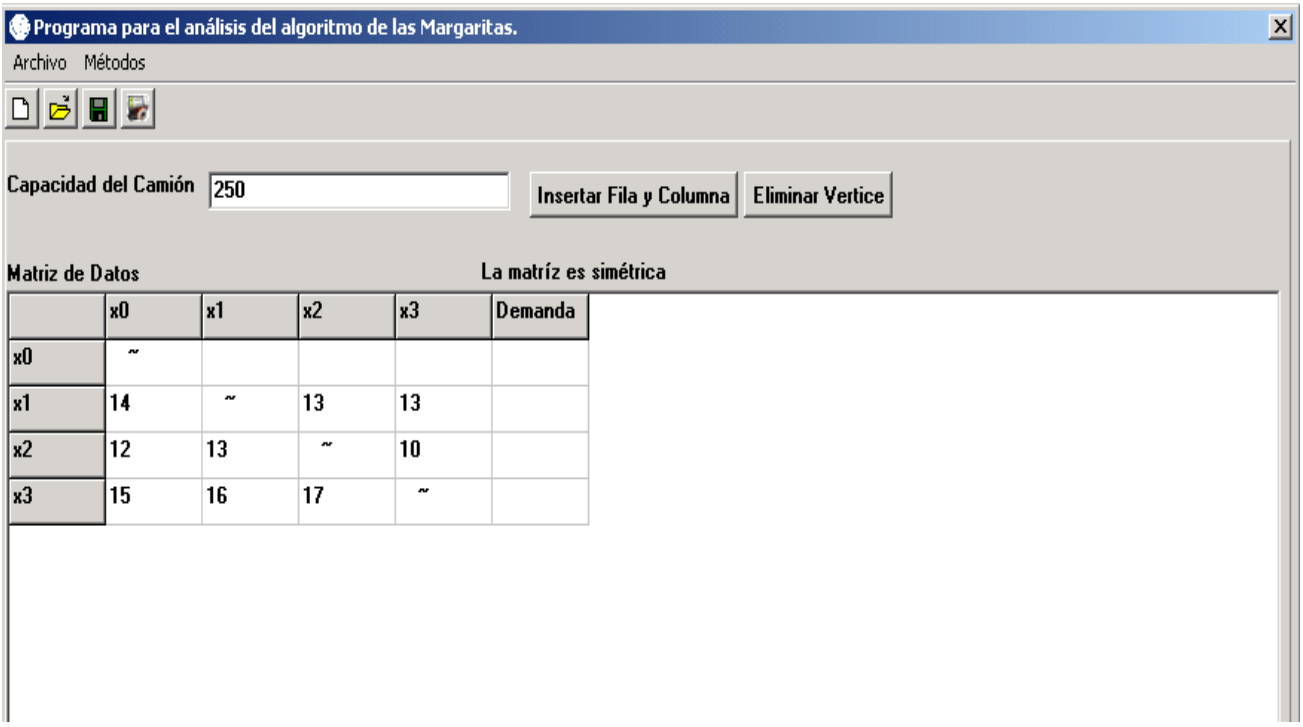


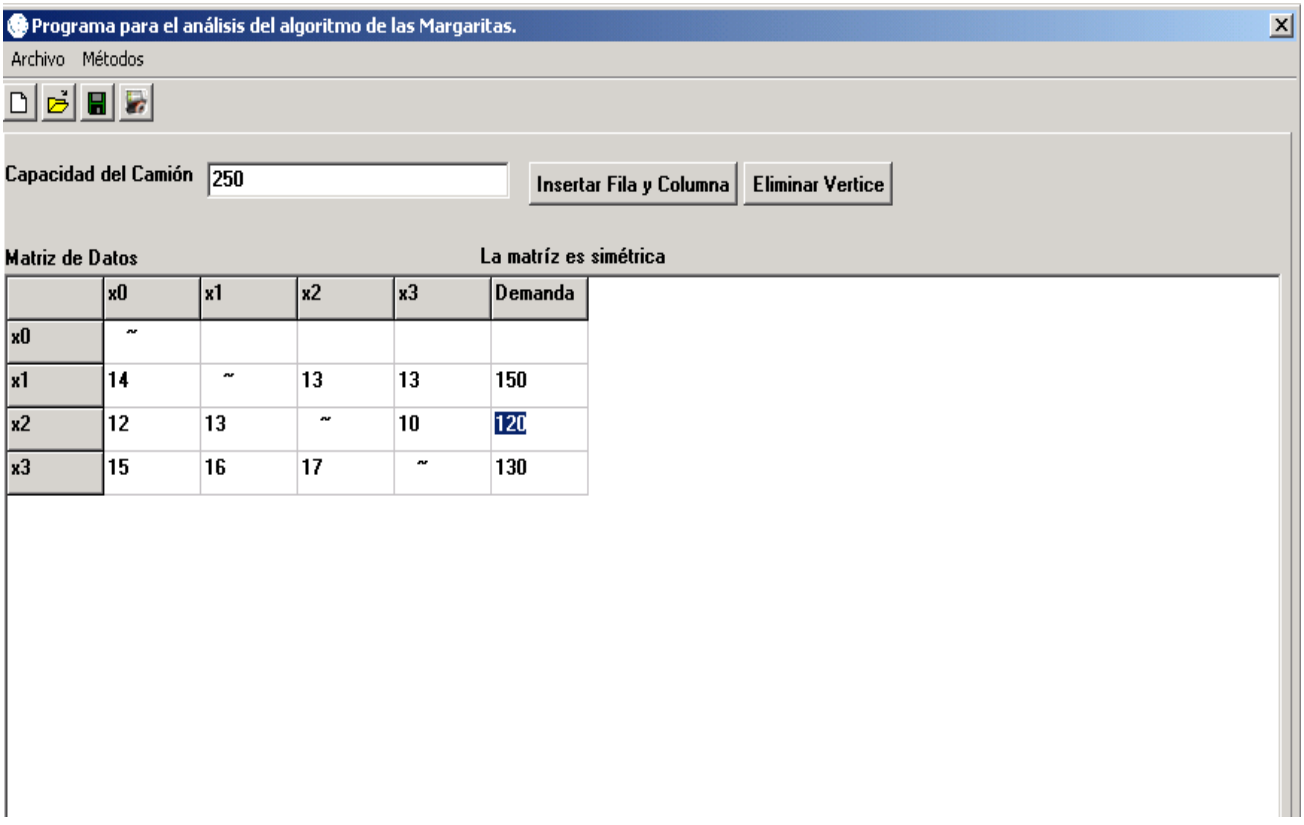
Prototipo#5 Gestionar distancia entre clientes.



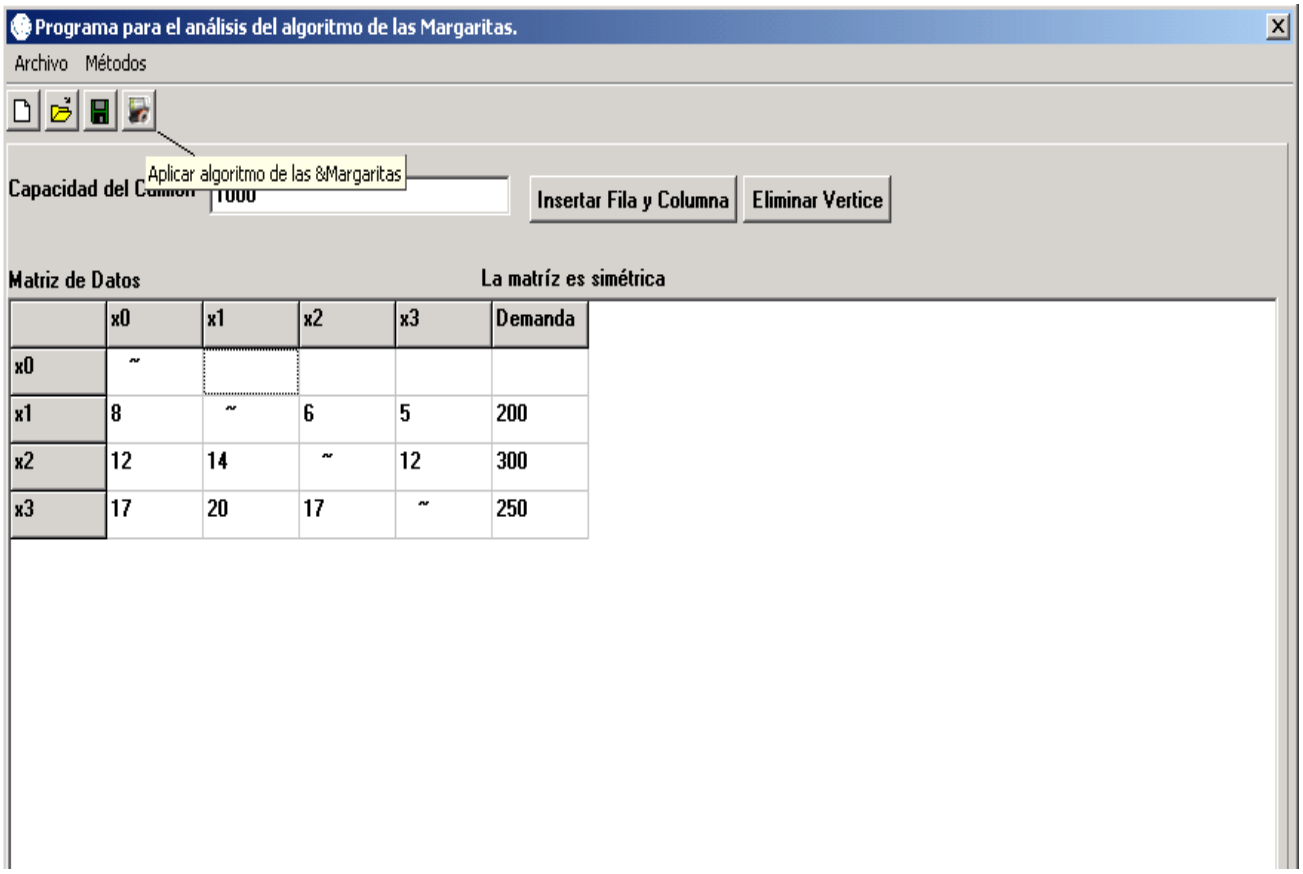


Prototipo#6 Gestionar demanda por cliente.

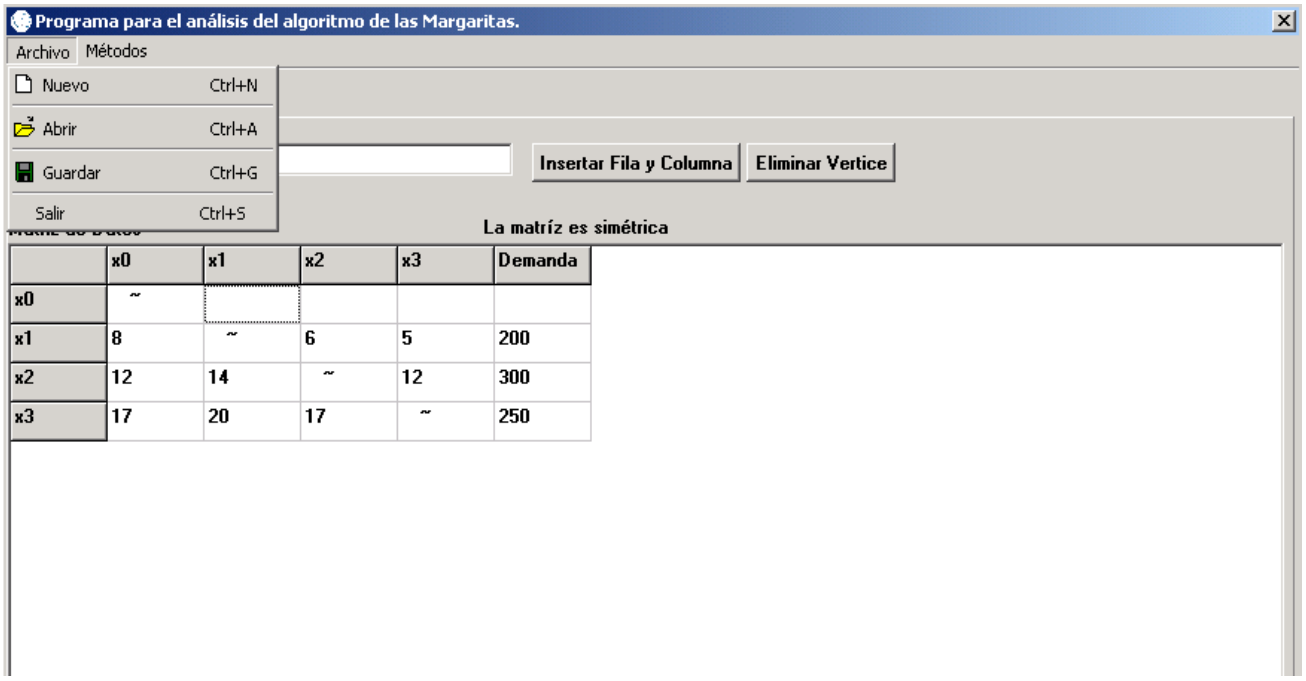




Prototipo#7 Aplicar algoritmo de las Margaritas.



Prototipo#8 Salir de la aplicación.



Anexo 2. Encuesta.

Encuesta sobre Producto Informático de Teoría de Redes.

Estimado usuario la presente encuesta forma parte de la Validación de un Producto Informático para un Trabajo de Diploma en la carrera de Ingeniería Informática.

Muchas gracias por su participación.

1- Utilidad del Producto Informático:

a) Como Software de Teoría de Redes:

Muy Buena_____ Buena_____ Regular_____ Mala_____

b) Como apoyo a la Enseñanza de la Teoría de Redes:

Muy Buena_____ Buena_____ Regular_____ Mala_____

2- Relacionado con otras Aplicaciones Informáticas sobre Redes:

a) En cuanto a los contenidos:

Es novedoso_____ Tiene Mejoras_____ Es Igual_____ Es más malo_____

b) En cuanto a la presentación:

Muy Buena____ Buena_____ Regular_____ Mala_____

3- En cuanto al uso:

a) Es más fácil de usar que otros____

b) Es igual a los otros____

c) Es más difícil____

4- En qué radican las ventajas:

a) En la entrada de datos _____

b) En la calidad de la visualización tabular _____

c) No tiene ventajas_____

5- Si usted fuera a valorar el Producto Informático en una escala de 5. ¿Cuántos puntos le daría?_____

6- Algún comentario al respecto.