UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE OPTIMIZACIÓN DE RUTAS DE TRANSPORTE

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA

GUSTAVO IZCÓATL HERRERA ROMERO

DIRECTOR:
ING. ORLANDO ZALDÍVAR ZAMORATEGUI

MÉXICO, D.F. NOVIEMBRE DE 2008

Agradecimientos

A la Universidad Nacional Autónoma de México, en especial a la Facultad de Ingeniería, por la oportunidad que tuve de ser parte de ellas y por todos los conocimientos adquiridos.

A mis profesores, en especial al ingeniero Orlando Zaldívar Zamorategui, director de esta tesis, por su infinita paciencia y su invaluable apoyo y ayuda durante la realización de la misma.

Al doctor Ricardo Aceves y a mis compañeros de servicio social en el Laboratorio de Sistemas de Transportes de la División de Estudios de Posgrado de la Facultad de Ingeniería, por su amistad y por su apoyo al proporcionarme bibliografía y otra información valiosa sobre el tema que trata esta tesis, así como por sus observaciones al sistema desarrollado durante el trabajo de la misma.

A mis familiares, especialmente a mis padres, a Martha y a mi hermano, por todo su amor y su apoyo, importantes para continuar y conseguir mis logros.

A todos mis amigos y compañeros de escuela y de trabajo, por su amistad y por los momentos y enseñanzas compartidos.

Tabla de Contenido

Introducción

Hablar de la calidad de vida en las grandes ciudades alude a un marco general que incluye tanto lo relativo al empleo y rasgos socioeconómicos de la población, como a la disponibilidad y acceso al equipamiento y servicios colectivos, así como a las características del medio ambiente. En particular, el aspecto de los sistemas de transportes es de vital importancia en lo que se refiere a servicios básicos, para la reproducción de la fuerza de trabajo y para su bienestar y que en general tiene trascendencia para la vida de la urbe en su conjunto.

El transporte intraurbano de personas, se considera que representa en las grandes ciudades hasta un tercio de las necesidades y problemas que condicionan su desarrollo tanto económico como social. Y es tan íntima su relación con el desenvolvimiento de la ciudad, que una política adecuada en materia de transportes, no sólo es susceptible de dar solución a los problemas internos del correspondiente sistema, sino a muchos más con los que se relaciona.

La importancia del transporte urbano y la multiplicidad de sus efectos sobre la vida de una ciudad se pone de manifiesto si se consideran cuestiones como las siguientes. Desde la perspectiva económica, dado que el más alto porcentaje de los viajes obedece a razones de trabajo, saltan a la vista las pérdidas económicas que implica una duración excesiva del tiempo de transporte, tanto en horas-hombre perdidas y en sus efectos sobre la productividad social, como en las mermas al ingreso individual del trabajador por faltas o retrasos. Se trata de un tiempo perdido para el ocio o el descanso, que se transforma en un alargamiento de la jornada de trabajo, que a la vez no genera ningún beneficio económico, social o individual.

En términos amplios, las insuficiencias y deficiencias en los sistemas de transportes urbanos trascienden a los intereses en sí de los propios usuarios lesionando al propio sistema productivo en su conjunto. Asimismo, el transporte tiene una posición privilegiada en la vida urbana porque condiciona, de hecho, el acceso a muchos otros servicios y funciones urbanas, ya que además de los viajes originados por los negocios y el trabajo, son de gran importancia los motivados por asistencia a instituciones educativas, culturales, recreativas y de esparcimiento, así como los ocasionados por compras y necesidades domésticas, interrelaciones familiares y sociales, por ejemplo.

Por otra parte, el congestionamiento, la lentitud y continuas paradas de los vehículos automotores implican fuertes pérdidas económicas en combustible, aparte de que aumentan la contaminación atmosférica, con los consiguientes daños para la salud pública y los edificios, equipos y otros bienes materiales urbanos.

Igualmente, los problemas derivados del tráfico y del transporte generan tensiones y enfermedades físicas y psicológicas y aumentan la agresividad y el malestar social. Las malas condiciones de la vialidad y del transporte, junto con los largos desplazamientos debidos a ello y la mala relación empleo-vivienda o a la inadecuada distribución de los equipamientos y servicios públicos, producen desgastes de energía y neurosis urbana que se traducen en alteraciones del comportamiento social.

De este modo, las deficiencias en los transportes colectivos significan una prolongación de la jornada de trabajo, menos tiempo disponible para otras actividades y un intenso desgaste adicional para quienes lo utilizan para llegar a sus empleos u otros destinos.

Vialidad y transporte en el DF

La importancia de los transportes urbanos estriba básicamente en su contribución a las grandes economías de escala y especialización asociadas al crecimiento de las urbes, como un elemento indispensable para asegurar la integración y enlace de todas las actividades de la sociedad.

Desde una perspectiva amplia, los servicios de transporte proporcionan acceso tanto a los centros de negocios y comerciales y a las fuentes de trabajo, como a instituciones de sanidad y enseñanza y a otros servicios y atractivos que se ofrecen en las zonas urbanas debido a las economías de escala.

El transporte intraurbano de personas implica, tanto desde el ángulo del buen funcionamiento de la ciudad, como desde el de la calidad de vida, la posibilidad de desplazarse de unos puntos a otros de ésta con comodidad, prontitud y seguridad. Sin embargo, pese a su aparente simpleza, estas condiciones cualitativas son difíciles de lograr, ya que dependen tanto de las características del crecimiento de la demanda y la oferta de estos servicios y de la eficiencia y coordinación de las empresas que los prestan, como de las disponibilidades de vialidad e infraestructura complementaria y de su uso racional, así como de la configuración y del trazado mismo de la ciudad.

Se ha observado una tendencia a incrementar la demanda de automóviles particulares, en busca de una mayor conveniencia, privacidad, seguridad y rapidez en los traslados. Pero este aumento, así como refleja la conveniencia de utilizar automóviles particulares, también obedece en gran medida a las deficiencias del transporte colectivo.

Origen y evolución del problema

En el fondo del problema del transporte urbano del DF y de la Zona Metropolitana, subyacen como factores decisivos la irrupción del automóvil y el descuido y la falta de planeación para lograr un sistema colectivo de pasajeros suficiente y eficiente.

Los problemas de movilidad se agudizan por el intenso desplazamiento de vehículos particulares que representan el 96% del total de la circulación, pero que sólo atienden el 20% de los viajes, en tanto que en el 4% restante de los vehículos se realiza el 75% de ellos.[1]

En cuanto a la distribución horaria de los viajes de origen a destino, la mayor concentración se produce entre las 6 y 9 horas, representando una tercera parte del total del día. En este periodo se agudizan los problemas de congestionamientos viales y de saturación de los modos de transporte público. En la tarde y noche también se presentan periodos de fuerte demanda de viajes, en particular, entre las 13 y 16 horas y entre las 17 y 19 y en menor medida hasta las 21 horas, pero su concentración es menor que en las horas de máxima demanda matutinas.[2]

Respecto al transporte público, el periodo de máxima demanda se presenta de 6:30 a 9:30 horas. Respecto al transporte privado, la máxima demanda se presenta de las 7 a las 9 horas.

En lo que se refiere a los motivos de los viajes, toca el 55% a los efectuados por necesidades de trabajo o negocios, el 35% se asocia a actividades escolares y el resto corresponde a compras, diversiones y otras causas.[3]

La coincidencia de los viajes por motivos de trabajo o negocios y los escolares en el periodo matutino de máxima demanda, explica la situación conflictiva que se presenta en esas horas pico. En cambio, aunque los viajes de retorno al hogar representan casi el 50% del total, la gravedad del problema es menor debido a que se distribuyen en un periodo más largo que avanza de las 13 horas en adelante.

Entre los problemas que se presentan se encuentran las deficiencias en el mantenimiento preventivo y correctivo de las unidades, lo que provoca descomposturas frecuentes y retiros de vehículos del servicio. En cuanto a la operación, se advierten deficiencias en la elaboración de itinerarios y en su revisión periódica para mantener una adecuación permanente entre la oferta y la demanda. Asimismo se presentan fallas en el cumplimiento del tiempo establecido para el recorrido de las rutas y para el intervalo de paso entre unas y otras unidades. También hay problemas en la asignación de parque vehicular a cada ruta y en el intervalo de salida de las unidades, todo lo cual determina que algunas unidades de transporte operen con sobre cupo mientras otras son subutilizadas.

[1] Molinero, Ángel. Sánchez, Ignacio. *Transporte público: planeación, diseño, operación y administración*. Quinta del Agua Ediciones, México, 1996.
[2] Ídem.
[3] Ídem

Se considera que se cuenta solamente con el 70% de las unidades necesarias para satisfacer a la demanda actual. Aunque la capacidad óptima por vehículo varía según el tipo de unidad, entre 75 y 110 personas, en las horas de máxima demanda los autobuses transportan de 100 a 140 pasajeros y en ocasiones aún más.[4]

Automóviles particulares

Es el modo de transporte que más ha crecido en términos del número de vehículos. La relación entre 1950 y 1980 cambió de un automóvil por cada 55.5 habitantes, a un automóvil por cada 5.5 habitantes. Esto es, se incrementó diez veces la proporción. Asimismo, en ese lapso, mientras que la población creció 6 veces, los autobuses urbanos aumentaron solamente 2.8 veces, los taxis casi 20 veces y los automóviles particulares 61 veces.[5]

Los automóviles particulares constituyen la máxima expresión de derroche vial no sólo porque en promedio transportan únicamente 1.6 personas por viaje, mientras que los autobuses pueden transportar entre 50 y 80 personas y aún más, sino por los desplazamientos adicionales a sus recorridos origen-destino que realizan para encontrar estacionamiento, además de la reducción de la capacidad vial que originan durante el tiempo que están estacionados.

Cabe mencionar que cada automóvil en promedio ocupa aproximadamente 10 m^2 cuadrados si está estacionado y 45 m^2 de espacio en la calle, tomando en cuenta el espacio necesario para su circulación, mientras que una persona que lo hace en un transporte colectivo ocupa solamente 1 m^2, un trolebús o autobús grandes llevando 50 pasajeros hacen el trabajo de 29 automóviles como promedio, los que son suficientes para llenar una cuadra. Un carril de automóviles mueve un máximo de 1575 pasajeros por hora, mientras que un carril de autobuses moverá 9000 pasajeros por hora y esto aparte de que los automóviles requieren espacio para estacionarse junto a una banqueta, con la consiguiente merma de la capacidad vial, de entre 10 y 30 m^2 según la disposición de los estacionamientos. Al relacionar estas necesidades con las disponibilidades de espacio vial, se entiende el problema y sus costos económicos, sociales y ambientales.

La combinación de los factores señalados a lo largo de este capítulo determinan que en las horas pico la velocidad promedio sea de entre 5 y 20 km por hora. Estas velocidades de operación y los congestionamientos implican muchas horas/hombre perdidas diariamente, lo cual tomando en cuenta que la mayor parte de los traslados son para llegar al trabajo o los negocios, significa un alto costo económico. A este costo se suma el originado por la pérdida de energéticos por embotellamientos, ya que se estima que la marcha lenta de vehículos (15 o menos km/h) ocasiona un desperdicio de 1.2 litros de combustible por hora, lo cual también se traduce en contaminación y daños a la salud y a los bienes e instalaciones materiales.

Por otra parte, dichos vehículos tienen una importante contribución a la contaminación por ruido.

Cabe también considerar que la inversión de un tiempo excesivo en los desplazamientos generan estrés y neurosis urbana que se traducen en alteraciones emocionales y del comportamiento social que van desde actitudes de grosería y desconsideración hasta actos violentos contra los medios de transporte y las personas, accidentes, etc.

Asimismo, la prolongación de los tiempos de recorrido afectan la integración familiar, ya que reducen el tiempo de permanencia en el hogar y de convivencia con la familia y en particular, la posibilidad de compartir con ella las horas de tomar los alimentos, que es un hecho significativo.

Igualmente, cuando las fallas del transporte y del tráfico generan excesivas incomodidades y alto gasto de tiempo y energía, numerosas personas se ven obligadas a prescindir de gran parte de viajes ligados a

[4] Molinero, Ángel. Sánchez, Ignacio. *Transporte público: planeación, diseño, operación y administración*. Quinta del Agua Ediciones, México, 1996.
[5] Ídem

fines culturales, políticos, sociales y recreativos y a limitar sus desplazamientos a lo más indispensable: el trabajo, las atenciones perentorias del hogar o de la salud y la asistencia obligada a clases, por ejemplo, cuestión que se constituye en otro ángulo de la deshumanización de la ciudad.

Cabe destacar como un aspecto básico que la falta de transporte colectivo afecta particularmente a la fuerza de trabajo que es la que integra el mayor porcentaje de la demanda.

Un alto porcentaje de los usuarios del transporte de autobuses en las horas pico son trabajadores que utilizan este medio para ir y regresar de sus fuentes laborales. De manera que el empleo de entre dos y cuatro horas al día para su traslado tiene graves efectos para su productividad, ya que por un lado, llegan fatigados a sus fuentes de trabajo y por el otro, en su lento regreso ven reducido su tiempo para reponer sus energías después de la jornada laboral, en un cauce en que no sólo ven mermadas sus horas de descanso y sueño, sino que al desgaste sufrido durante el lapso de sus labores, se suma el del tiempo excesivo de viaje y las incomodidades durante el mismo y las tensiones por retrasos y esperas.[6]

Se considera que este desgaste adicional es susceptible a la larga, de reducir en una tercer parte la vida económicamente activa de quienes lo padecen. Es decir, que en lugar de que los trabajadores afectados tengan un periodo vital de trabajo de 45 años, lo tendrán únicamente de 30 en dichas condiciones. Resulta obvio el significado de este hecho no sólo para el bienestar y desarrollo personal y familiar de los que lo sufren, sino para la productividad nacional y para el desenvolvimiento futuro del país.

En suma, la dinámica y magnitud adquirida por el problema del transporte lo ha colocado en un punto en que perturba el equilibrio entre las funciones económicas, técnicas, socioculturales, recreativas y humanas de la urbe y sus habitantes. Según estimaciones oficiales, el 20% del ingreso familiar se destina al transporte y el trabajador utiliza el 30% de su tiempo en desplazarse.

La experiencia deja claro que el automóvil no constituye la solución y que el énfasis debe ponerse en el desarrollo del transporte colectivo cuyas características deben mejorarse cuantitativa y cualitativamente, tanto para cubrir las necesidades de la población mayoritaria, como para ofrecer alternativas aceptables para los estratos de ingreso medio y alto. Es decir, para desanimar el uso del automóvil particular, junto con otras medidas complementarias.

Ahora bien, como queda implícito en lo expresado hasta ahora, la transportación urbana no representa un problema, sino un complejo de problemas en interacción. Por tanto, requiere de un sistema de soluciones.

Requerimientos de un sistema de transporte

Existen tres grupos de participantes en un sistema de transporte público que se interrelacionan:

1. Usuario o consumidor del servicio

Entre sus principales requerimientos se encuentra la disponibilidad de transporte ya que el usuario requiere contar con paradas o estaciones razonablemente cercanas, un servicio regular y que lo pueda utilizar a cualquier hora del día.

También requiere un servicio puntual y confiable, que le permita abordar la unidad que lo llevará a su destino dentro de rangos aceptables de demoras. El usuario aceptará mayores demoras dependiendo de la distancia que tenga que recorrer ya que las demoras por el tránsito y las interferencias ocasionadas por otros medios de transporte son las causas de retardos que se presentan más frecuentemente.

Otro requerimiento es el tiempo de recorrido. Un tiempo de recorrido demasiado largo inhibe el uso del transporte público.

[6] Molinero, Ángel. Sánchez, Ignacio. *Transporte público: planeación, diseño, operación y administración*. Quinta del Agua Ediciones, México, 1996.

La comodidad es un requerimiento que incluye una variedad de factores cualitativos, como por ejemplo la disponibilidad de asiento, un recorrido suave, la geometría de las entradas y salidas del vehículo, el ancho de los pasillos, los niveles de ruido interior, el grado de privacidad y la apariencia interior y exterior del vehículo.

La conveniencia es un requerimiento que se refiere al sistema en general y abarca factores como cobertura del sistema de transporte, la necesidad de efectuar transbordos, la existencia de información suficiente y confiable, regularidad en el servicio, la existencia de un adecuado servicio en horas de baja demanda e instalaciones de espera correctamente diseñadas de acuerdo a las necesidades del usuario.

La seguridad en términos de la prevención de accidentes es muy importante, así como en términos de una mayor prevención de incidentes criminales.

El costo que representa el transporte para el usuario, especialmente la tarifa, es de un impacto muy importante para definir la preferencia del usuario.

2. Prestatario

Entre los requerimientos del prestatario se encuentra el logro de una adecuada cobertura de área. Esta área se define como la suma de la superficie que se encuentra a 5 ó 10 minutos de distancia, recorrida a pie, de cada estación o parada, quedando dentro del área a la que da servicio la ruta.

También estará interesado en proporcionar una frecuencia adecuada, buscando frecuencias regulares y altas que permitan atraer cualquier tipo de viaje, ya sea de trabajo, de compras, recreación o estudio.

La confiabilidad en el sistema de transporte dependerá del mantenimiento que el prestatario dé a sus unidades.

El prestatario está interesado en lograr velocidades comerciales altas en sus rutas ya que este concepto afecta el tamaño de su parque vehicular y por consiguiente sus costos laborales, energéticos y de mantenimiento, además de la atracción de pasajeros.

Otro requerimiento del prestatario es lograr el equilibrio entre la oferta y la demanda, para satisfacer las necesidades de los usuarios dentro de costos razonables.

Los costos son el factor más importante que el prestatario tomará en cuenta. Incluye tres conceptos: costo de inversión, costo de operación y los ingresos. Éstos variarán conforme a las características y condiciones locales de cada sistema, así como a lo largo del tiempo.

El prestatario tendrá como requerimiento el contar con una flexibilidad en la capacidad de pasajeros de la ruta y el tipo de vehículos con que puede operar.

La atención del prestatario a la seguridad deberá ser no sólo sobre la del usuario, sino también sobre la seguridad operacional del sistema.

La atracción de pasajeros es el requerimiento más importante del prestatario y está en función del tipo y nivel de servicio que se ofrezca, así como de la imagen del sistema. Esta imagen está compuesta por las características físicas del sistema, la simplicidad de la ruta, la confiabilidad y regularidad del servicio y los costos que representen al usuario usar el servicio.

3. Comunidad

La comunidad está interesada en que se preste un nivel y tipo de servicio adecuado, el cual permita una mayor atracción de pasajeros hacia los medios de alta capacidad. La comunidad debe reglamentar los impactos a largo plazo que fomentan el transporte tales como el desarrollo urbano, los cambios en el valor del uso del suelo y las actividades económicas, así como los aspectos relativos al medio ambiente,

el uso eficiente de la energía y el logro de una eficiencia económica en las inversiones que realice en materia de transportes.

Propuesta de contribución a la solución

Como una contribución a la solución del problema de transporte, se presenta el Sistema de optimización de rutas de transporte, como una propuesta de automatización de la captura, almacenamiento, análisis y uso de la información relacionada con una ruta de transporte, con el fin de hacer más fácil y rápida la elaboración de itinerarios para proveer un buen servicio de transporte público, satisfaciendo la demanda con el menor gasto posible de infraestructura.

Este trabajo presenta una solución al problema que significa para las empresas proveedoras de transporte público, calcular las frecuencias óptimas de paso de sus unidades, necesarias para proveer de un buen servicio de transporte público, satisfaciendo la demanda y con el menor gasto posible de infraestructura. Para esto necesitan acelerar la obtención de resultados derivados del análisis (normalmente hecho de manera manual) de los aforos o muestreos que se realizan en las rutas. Con esta solución se intenta proveer una herramienta que permita a estos distribuidores de servicios calcular de manera rápida estas frecuencias de paso y la cantidad de vehículos adecuada, con el fin de satisfacer la demanda de servicio utilizando los recursos de manera óptima.

El proyecto describe cómo con base en los aforos o muestreos hechos a lo largo de una ruta de transporte público y a los análisis hechos por el sistema aquí descrito, se pueden calcular las frecuencias de paso de las unidades de transporte, a lo largo de la ruta, en función de la demanda variante y de las características de capacidad de los vehículos que se utilizan en la ruta. Esto significa poder ofrecer el servicio deseado por la demanda de pasajeros y al mismo tiempo optimizar el uso de los vehículos de transporte de la ruta.

Este trabajo se encuentra dividido en los siguientes capítulos. Los primeros cinco se refieren a los fundamentos teóricos en los que se basó el trabajo realizado.

Capítulo 1. Ingeniería de software. Presenta los elementos que constituyen esta rama de la ingeniería, se describen los principales modelos o paradigmas de desarrollo, las cuales sirven a los ingenieros de software como posibles estrategias a seguir. Por último se presenta los elementos y métodos más conocidos para la planificación de un proyecto de desarrollo de software.

Capítulo 2. Administración de riesgos. Presenta los elementos y procesos que se toman en cuenta al analizar los riesgos que posiblemente ocurrirán durante el desarrollo de un proyecto de software y la manera de planificar y monitorear las actividades destinadas a evitar su ocurrencia o minimizar sus efectos.

Capítulo 3. Ingeniería de requerimientos. Presenta los elementos y procesos que se toman en cuenta para recopilar en un principio y dar estructura a la información necesaria para comprender las necesidades y objetivos que se intentará satisfacer mediante el desarrollo de un sistema de software.

Capítulo 4. Diseño. Presenta los elementos técnicos y métodos que se toman en cuenta para transformar las necesidades del cliente y sus objetivos a satisfacer, en un diseño de software que cumpla con estos objetivos y necesidades y sea el primer paso hacia la construcción del sistema.

Capítulo 5. Bases de datos. Se describen los elementos que forman una base de datos. Se presentan los modelos básicos existentes mediante los cuales se puede estructurar una base de datos. También se describe los conceptos correspondientes al modelo relacional.

Los capítulos 6 a 10 se refieren al trabajo de análisis de factibilidad y de riesgos, recopilación de requerimientos del sistema, su diseño, construcción y validación.

Los capítulos 11 a 13 describen un ejemplo de uso del sistema construido, las conclusiones sobre la ventaja de su construcción respecto a la situación actual del servicio de transporte, y la bibliografía consultada.

Capítulo 1
Ingeniería de software

Para definir lo que es ingeniería de software, se define primero lo que es software, su proceso y sus características más importantes.

1.1 Software

Son los programas de computadora, así como la documentación asociada a ellos y la configuración de datos necesaria para su funcionamiento correcto. Lo constituyen los programas independientes, los archivos de configuración que se usan para ejecutar esos programas, la documentación que describe la estructura del sistema y la documentación para que el usuario use el sistema.[7]

1.2 Proceso del software

Es la serie de pasos a seguir cuando se quiere construir un sistema de software, desde el punto de vista de administración de proyectos.

El software puede clasificarse de dos formas: por su tipo y por sus aplicaciones.

1.2.1 Clasificación del software por su tipo

a) Software personalizado. Son los programas requeridos por un cliente en particular y desarrollados para satisfacer las necesidades especiales de ese cliente. Las especificaciones del producto son hechas por el cliente que compra el software. Como ejemplo está el software desarrollado para el proceso de negocios de una empresa particular.
b) Software genérico. Son programas desarrollados de manera aislada y vendidos en el mercado general, a cualquier cliente que desee comprarlos. Las especificaciones del producto no son dadas por ningún cliente en particular. Como ejemplo están los procesadores de texto o los lenguajes manejadores de bases de datos.

1.2.2 Clasificación del software por sus aplicaciones

- Software de sistemas. Es un conjunto de programas que sirven a otros programas. Tienen interacción con elementos de hardware, administran y comparten los recursos de memoria y procesador. Como ejemplo está el sistema operativo de una computadora.
- Software de tiempo real. Son los programas que procesan datos de sucesos en el momento en que estos ocurren. Se componen de programas que recolectan los datos y les dan un formato apropiado, programas que dan la respuesta a estos datos y programas que muestran de alguna manera (en pantalla, impresos, etc.) estos datos. Como ejemplo está un programa que despliegue en pantalla las localizaciones geográficas de vehículos en movimiento.
- Software de administración. Son sistemas computarizados de inventarios, nóminas, etc. que permiten a sus usuarios tomar decisiones relativas a operaciones comerciales.
- Software de ingeniería y científico. Son programas que realizan cálculos numéricos y estadísticos. Se aplican en ramas de la ingeniería y ciencias como vulcanología, meteorología, etc. Algunos de estos sistemas también incluyen software de tiempo real.
- Software empotrado. Es el software que realiza funciones muy específicas, se encuentran en memorias de sólo lectura y que generalmente se encuentra en la programación de circuitos eléctricos, por ejemplo en los hornos de microondas o circuitos en sistemas de seguridad.
- Software de computadoras personales. Generalmente es software genérico, tal como los procesadores de texto, bases de datos, aplicaciones multimedia, etc.
- Software de inteligencia artificial. Son programas que utilizan algoritmos no numéricos, sino más bien booleanos (con valor falso o verdadero) y probabilísticos para resolver problemas complejos de toma de decisiones.

[7] Sommerville, Ian. *Ingeniería de software*. Addison Wesley, Sexta edición, Reino Unido, 2002.

1.3 Características de un sistema de software de buena calidad

Un sistema de software de buena calidad es aquel que cumple con los requisitos del usuario, que tiene un buen funcionamiento y comportamiento, que es fácil de utilizar por los usuarios, cuenta con la documentación adecuada para facilitar su uso, tiene una estructura definida y es fácil comprender el programa fuente. Esto último es muy importante cuando se lleva a cabo el mantenimiento del sistema.

1.4 Atributos esenciales de un buen sistema de software

- Mantenibilidad. El sistema debe poder evolucionar al cambiar los requerimientos del cliente.
- Confiabilidad. El sistema debe tener una posibilidad de fallo muy reducida y en el caso de fallar, no deben existir daños a los datos, daños físicos o económicos.
- Eficiencia. Debe haber un uso óptimo de los recursos disponibles como la memoria y los ciclos de procesamiento. Esto se refleja en menores tiempos de respuesta del sistema.
- Usabilidad. El usuario no debe tener dificultad alguna al operar el sistema. Éste debe ser accesible, con interfaces de usuario apropiadas y con documentación adecuada.

1.5 Ingeniería de software

Es el conjunto de conocimientos teóricos, métodos y herramientas que se usan tanto para las actividades técnicas de un proyecto de desarrollo de software, como para la planeación y administración de las actividades del proyecto. La ingeniería de software está formada por los siguientes elementos:

1.5.1 Proceso

Define el conjunto de pasos y el entorno de trabajo para desarrollar el sistema. Establece los pasos para la administración del proyecto, las herramientas técnicas que se utilizarán y los pasos para realizar la documentación, aseguramiento de la calidad y control de cambios en el proyecto.

1.5.2 Métodos

Son los pasos técnicos para la construcción del software. Esto incluye las tareas de análisis de requerimientos, diseño, codificación, pruebas y mantenimiento.

1.5.3 Herramientas

Son los lenguajes de programación, lenguajes para bases de datos, programas para administración y calendarización de proyectos, programas para modelado de estructuras de datos, del comportamiento del sistema y demás dispositivos que ayudan a realizar las tareas que componen las etapas del desarrollo de software.

1.6 Fases del trabajo de ingeniería de software

Las cuatro fases principales que conforman el trabajo de ingeniería de software son:

- Definición. Durante esta etapa se definen las funciones del sistema de software, la información que procesa y produce, qué restricciones de costo y rendimiento tiene, qué criterios de validación existen para verificar que el sistema definido es el deseado. Las tareas principales de esta fase son la planificación temporal del proyecto y el análisis de requisitos.
- Desarrollo. Una vez definidas las funciones, datos y demás características y elementos que se requiere que tenga el sistema, se define una arquitectura para el mismo, se diseñan las estructuras de datos, se diseñan y programan las funciones y procedimientos y se diseñan y se construyen las interfaces. Las tareas principales de esta fase son diseño y generación de código.
- Validación. El software es revisado por el equipo de desarrollo y el usuario para comprobar que hace lo que realmente se desea.
- Mantenimiento. Son los cambios y correcciones hechos al software. Los tipos de cambios durante la fase de mantenimiento son:
  a) Corrección: para corregir los defectos encontrados en el sistema por el cliente.

b) Adaptación: debido a cambios en el entorno técnico del software (sistema operativo, características de las computadoras, etc.) o en el entorno de negocios (reglas de la empresa, etc.).
c) Mejora: cuando el cliente encuentra funciones nuevas necesarias no definidas en un principio.
d) Prevención: al hacer cambios en el software, éstos pueden afectar el funcionamiento de otras partes del sistema, relacionadas con aquella en la que se hicieron los cambios. Esto significa que el software se deteriora. El mantenimiento preventivo ayuda a prevenir los errores que puede ocasionar los cambios por mantenimiento correctivo, adaptativo o de mejoras.

Otro tipo de mantenimiento que puede tomarse en cuenta es el mantenimiento al usuario, mediante ayuda por teléfono, en sitios web o atención personal.

1.7 Modelos de procesos de software

También conocidos como paradigmas de desarrollo de software. Cada modelo es una forma de desarrollo particular de un proceso de software, una organización particular de las actividades que conforman al proceso de desarrollo. El modelo o paradigma a utilizar depende principalmente del tipo de sistema a desarrollar, de la manera en que se obtienen los requerimientos del sistema, de las restricciones de tiempo y económicas impuestas por el cliente para el desarrollo del producto y entrega del producto final, entre otras.

Entre los paradigmas de desarrollo de software más conocidos se encuentran:

1.7.1 Modelos lineales

Estos modelos se caracterizan por una secuencia lineal en las actividades que componen el desarrollo del software. Debe haberse terminado la primera fase del proceso para poder hacer la segunda y así sucesivamente hasta el fin del desarrollo.

1.7.1.1 Modelo en cascada

También conocido como modelo lineal secuencial o modelo clásico. La figura 1.1 muestra el diagrama de este tipo de modelo.

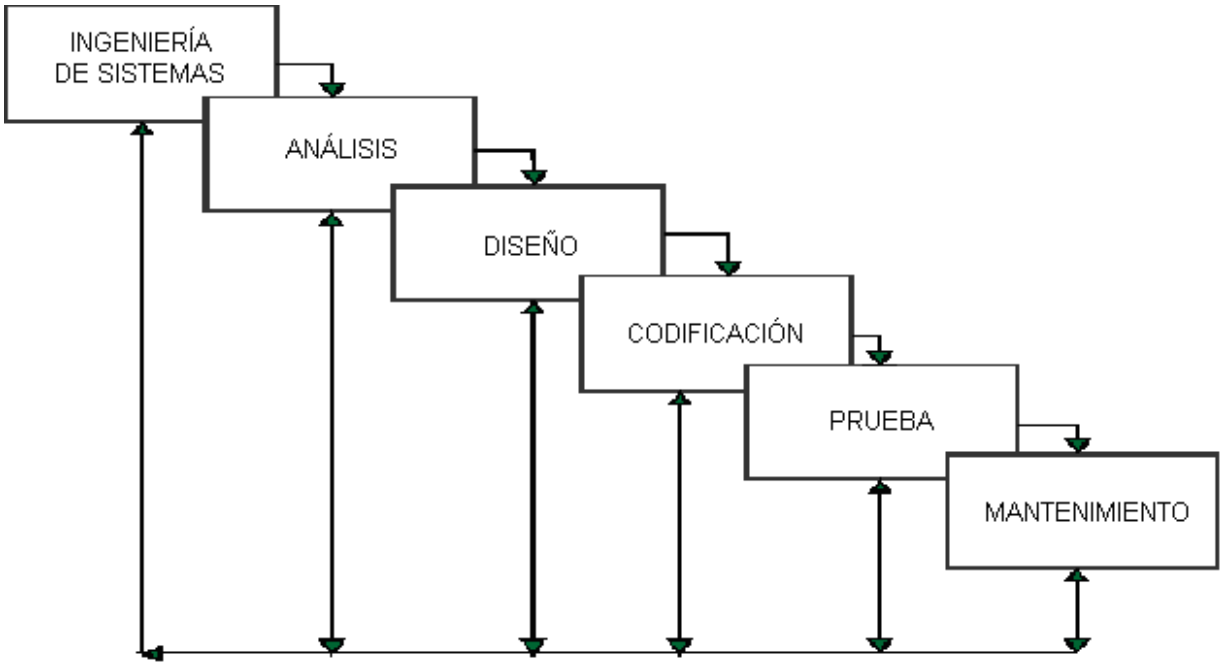


Fig. 1.1 Modelo en cascada

Como su nombre lo indica, para desarrollar un sistema de software, se sigue una secuencia de actividades, una después de otra, las cuales son:

a) Ingeniería de sistemas. En esta etapa se abarcan los requerimientos globales del sistema, además de establecer los requerimientos del mismo, aquí se realiza un análisis a grandes rasgos y se genera un diseño en el ámbito global.
b) Análisis. Se comprende el tipo de información que se manejará, el ambiente en que se usará, así como las funciones del sistema, comportamiento y rendimiento requeridos.
c) Diseño. Se traducen los requisitos en las estructuras de datos, arquitectura de software, interfaces y procedimientos necesarios para llevar a cabo el desarrollo del sistema.
d) Codificación. Se construye el programa utilizando algún lenguaje de programación o herramientas generadoras de código.
e) Pruebas. Se verifica que el sistema realice las funciones de manera correcta.
f) Mantenimiento. Posteriormente a la entrega del sistema se realizan modificaciones y adecuaciones al sistema de ser necesario.

Observaciones al modelo en cascada
Cuando se da el caso de que los requisitos del cliente son expuestos de manera clara y en su totalidad, el modelo en cascada es aplicable y el desarrollo del sistema se hace de manera relativamente sencilla y rápida. Sin embargo, en la gran mayoría de los proyectos, el cliente no expone desde el principio la totalidad de los requisitos del sistema de software, ni lo hace de manera totalmente clara. Esto hace que el modelo en cascada no sea aplicable. Otro inconveniente es que no existirá una versión del sistema, sino hasta las etapas de prueba, lo cual implica que si existe un error grave en el análisis o diseño, suponga un costo considerable tanto económico y de esfuerzo para corregirlo.

1.7.1.2 Modelo de construcción de prototipos

Este modelo es aplicable cuando el cliente no define de manera detallada los requisitos del software, sino de manera general. También es aplicable cuando el equipo de desarrollo no está seguro del buen funcionamiento de alguna función o interfaz de usuario y tiene que hacer pruebas con el cliente. La figura 1.2 muestra el diagrama del modelo.

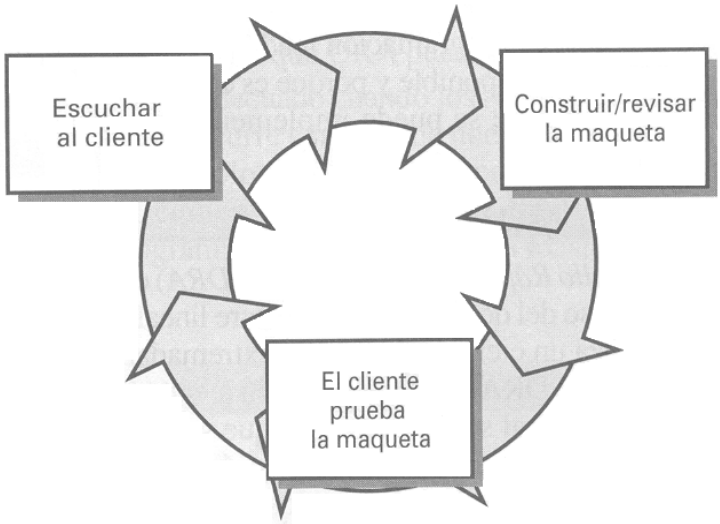


Fig. 1.2 Modelo de construcción de prototipos[8]

Al principio se recolectan los requisitos generales del cliente y con base en éstos se construye un prototipo. Este prototipo es evaluado por el cliente y se hacen mejoras y adiciones, con lo cual se

[8] Pressman, Roger. *Ingeniería del software*. McGraw-Hill, Quinta edición, España, 2002.

hacen más específicos algunos requisitos y aparecen otros nuevos, con lo cual se construye un nuevo prototipo, desechando el prototipo anterior. Esto se repite varias veces, hasta que se llega a un sistema final.

Observaciones al modelo de construcción de prototipos
De manera ideal, este modelo sirve para hacer prototipos de manera rápida, con el fin de hacer más específicos los requisitos ya existentes y obtener nuevos requisitos. Este modelo tiene el inconveniente de que en general los prototipos se hacen aprisa, para poder mostrar al cliente los prototipos en el tiempo estipulado y cuando se hacen las mejoras y correcciones, generalmente se usa el programa ya existente y sobre él se hacen los ajustes. Esto hace que la calidad del programa pueda ser mala, al no tener siempre tiempo para seguir una estructura específica, ni para documentar las decisiones y los cambios hechos.

1.7.1.3 Desarrollo rápido de aplicaciones

Este modelo es una variante del modelo lineal secuencial, con la diferencia de que durante el desarrollo del sistema se reutilizan componentes ya existentes, lo cual acorta el tiempo de desarrollo y hace más fáciles las pruebas, siendo éstas en su mayor parte pruebas de integración de los componentes ya existentes. Se requiere, al igual que en el modelo en cascada normal, la especificación clara y total de los requisitos del sistema. Comprende las siguientes fases:

a) Modelado de administración. Se define la naturaleza de la información que el sistema procesa y genera y su flujo dentro del mismo.
b) Modelado de datos. Una vez definida la información, se definen las estructuras de datos que la conforman, sus atributos y las relaciones entre ellas.
c) Modelado del proceso. Se definen las funciones para añadir, modificar, suprimir, recuperar o mover los objetos que conforman la información, dentro del sistema.
d) Generación de aplicaciones. En esta etapa no se usan lenguajes de programación de tercera generación para crear las aplicaciones, sino lenguajes que generan código automáticamente. Se utilizan componentes de programas ya existentes o se crean componentes que sean reutilizables.
e) Pruebas y entrega. Se comprueba que todos los componentes en conjunto funcionen de manera correcta.

Observaciones al modelo de desarrollo rápido de aplicaciones
Un sistema de software que vaya a ser desarrollado mediante este modelo, debe ser modularizable, es decir, debe poder ser separado en partes que realicen las funciones principales del sistema, con el fin de que la unión de los componentes ya existentes o la construcción de los componentes reutilizables sea sencilla.

1.7.2 Modelos evolutivos

Estos modelos se caracterizan por el desarrollo de versiones cada vez más completas del sistema de software. Estos modelos son iterativos y responden a los cambios en los requisitos del cliente durante el desarrollo del software.

1.7.2.1 Modelo incremental

Este modelo combina las características del modelo lineal secuencial con el modelo de construcción de prototipos. Al principio se recolectan los requisitos generales, luego se realizan todas las fases del modelo lineal secuencial, para tener un prototipo que ya realiza parte de las funciones. Este prototipo es revisado con el cliente, lo cual resulta en correcciones al prototipo y en nuevas funciones definidas. A diferencia del modelo de construcción de prototipos, este prototipo no se desecha, sino que se trabaja sobre él aplicando de nuevo las fases del modelo lineal secuencial para producir un segundo prototipo, el cual es de nuevo revisado por el cliente. Este ciclo se repite varias veces hasta llegar al producto final.

1.7.2.2 Modelo espiral

Este modelo utiliza el modelo secuencial lineal, sobre el modelo de construcción de prototipos. Se compone de las siguientes actividades:

a) Comunicación con el cliente. Durante la primera iteración del proceso, se recolectan los requisitos básicos. En las siguientes iteraciones se revisan los prototipos y se recolectan nuevos requisitos o peticiones de mejoras y correcciones.
b) Planificación. Durante esta etapa se definen los recursos de personal, tiempo y económicos necesarios para construir el prototipo que cumple con los requisitos detallados hasta el momento por el cliente.
c) Análisis de riesgos. En esta etapa se definen los posibles riesgos que conduzcan a fallas en el sistema o retrasos respecto a los tiempos planeados y se definen las soluciones a estos posibles riesgos.
d) Ingeniería. En esta etapa se diseñan y definen las estructuras de datos, interfaces y demás necesarias para la construcción del sistema.
e) Construcción y adaptación. Se realizan las tareas de construcción, prueba, instalación y generación de documentación del sistema.
f) Evaluación del cliente. El cliente revisa el sistema y se definen correcciones y mejoras para el sistema.

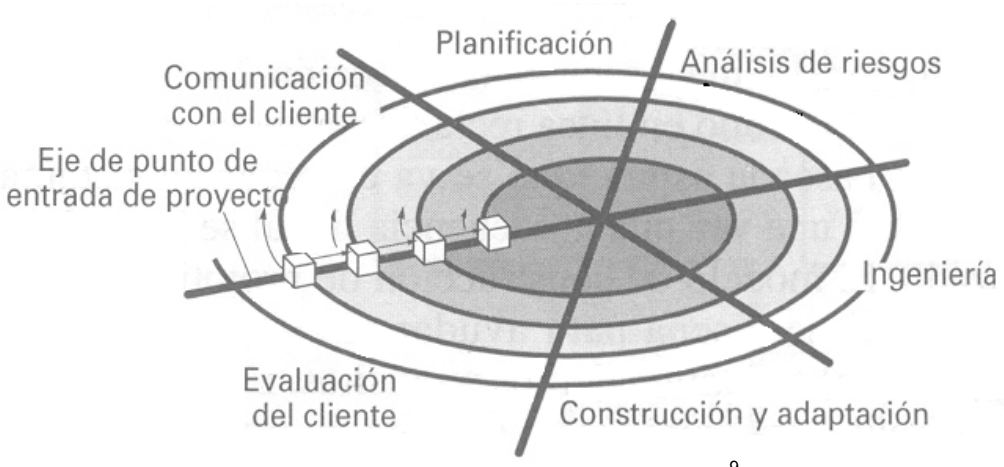


Fig. 1.3 Modelo de desarrollo en espiral[9]

La figura 1.3 muestra cómo el modelo en espiral desarrolla el sistema de manera incremental el sistema realizando en cada iteración las actividades de comunicación con el cliente, planificación, análisis de riesgos, ingeniería, construcción y adaptación y por último evaluación del cliente.

Observaciones para el modelo en espiral
Este modelo es muy recomendable dada la naturaleza evolutiva del software. Tanto durante el desarrollo de un sistema, como durante su operación, este modelo puede ser aplicado para su creación y mantenimiento, durante todo el periodo de vida útil del sistema.

1.7.2.3 Modelo espiral WINWIN

Este modelo en espiral, tiene además una etapa en la cual se realizan actividades de negociación con el cliente, con el fin de tanto el cliente obtenga el sistema que satisface sus necesidades, como el equipo desarrollador del sistema obtenga el presupuesto deseado y el tiempo necesario para realizarlo.

[9] Pressman, Roger. *Ingeniería del software*. McGraw-Hill, Quinta edición, España, 2002.

1.7.3 Otros paradigmas de desarrollo de software

1.7.3.1 Modelo de desarrollo basado en componentes

Esta metodología consiste en la reutilización de componentes ya existentes. Estos componentes o clases encapsulan en un solo componente los atributos de un objeto a utilizar, así como las funciones que se llevan a cabo con ese objeto. También puede darse el caso de crear nuevos componentes, los cuales también deben ser reutilizables. La reutilización tiene beneficios como ahorro del tiempo de prueba al estar estos componentes ya probados y ahorro de tiempo de desarrollo, al no tener que crear todo desde cero, sino usar componentes ya existentes.

1.7.3.2 Modelo de métodos formales

Este método utiliza un conjunto de especificaciones matemáticas para llevar a cabo el desarrollo de software. Se basa en la transformación matemática formal de cada especificación de los requisitos a un programa ejecutable. Se usa una notación matemática para expresar los requisitos del sistema y ésta se va detallando iterativamente a través de una serie de transformaciones matemáticas, hasta llegar al algoritmo que se programará. Este modelo funciona solamente cuando se tiene una especificación formal y completa del sistema, lo cual no sucede muy a menudo.

1.7.3.3 Técnicas de cuarta generación

Mediante estas técnicas se usan herramientas de software que permiten expresar en un alto nivel las características del sistema a desarrollar, utilizando lenguajes de cuarta generación especializados o notaciones gráficas que describan el problema. Estas herramientas generan el código de manera automática.
Estas técnicas requieren de una forma especial de expresión de los requisitos, por lo cual éstos deben obtenerse de la forma más completa y clara posible del cliente. También las estructuras de datos deben estar definidas de antemano para que el lenguaje de cuarta generación pueda generar código.

1.8 Administración de proyectos de software[10]

Es la planificación, supervisión y control de los recursos humanos, económicos y técnicos y las actividades durante el desarrollo de un sistema de software. Es necesaria para identificar el problema a resolver mediante un sistema de software de manera correcta y entregar la solución adecuada, así como para planear y calendarizar el proyecto y para supervisar que el proyecto esté siendo desarrollado de acuerdo al plan y al calendario, con los estándares adecuados, así como que satisfaga las necesidades reales del cliente.

Las cuatro partes fundamentales a administrar en un proyecto de software son:

a. Personal. Se trata de contar con personal preparado y motivado para llevar a cabo las actividades del proyecto.
b. Producto. Se trata de establecer los objetivos y el ámbito del producto, estudiar todas las soluciones alternativas, así como los riesgos técnicos y de administración.
c. Proceso. Se elige el enfoque o paradigma del software que es más recomendable para desarrollar el sistema.
d. Proyecto. Con base en la experiencia de desarrollo, los administradores y desarrolladores del proyecto deben conocer los riesgos más comunes y posibles, con el fin de ayudarse a la planificación de otros proyectos.

[10] Pressman, Roger. *Ingeniería del software*. McGraw-Hill, Quinta edición, España, 2002.

Las actividades de la administración de proyectos de software son:

1. Redacción de la propuesta. La propuesta describe los objetivos del proyecto y cómo se llevará a cabo. Incluye estimados de costo y calendarización.
2. Planeación y calendarización del proyecto. Se identifican las actividades y puntos de entrega de avances del proyecto. Se bosqueja un plan de desarrollo del proyecto.
3. Costeo del proyecto. Se estiman los recursos económicos, técnicos y humanos necesarios para desarrollar el proyecto.
4. Supervisión y revisión del proyecto. Se revisa frecuentemente el progreso del proyecto y se compara con el progreso y el costo planeados. Puede ser de manera formal, con revisiones completas del desarrollo técnico y progreso del proyecto o de manera informal, con entrevistas informales con el personal del proyecto.
5. Selección de personal. De manera ideal, se seleccionará personal con habilidades y experiencia apropiadas para el proyecto en cuestión. Otra opción es seleccionar personal con poca experiencia, pero con las habilidades necesarias, con el fin de que ganen experiencia.
6. Redacción y presentación de informes. El administrador del proyecto debe dar informes de los avances del mismo al cliente.

1.9 Planeación del proyecto

La estimación de recursos, costos y planificación temporal del proyecto no es una actividad que nos produzca conclusiones exactas. Esto se debe a la gran cantidad de variables que intervienen y a los cambios que éstas sufren durante el desarrollo del proyecto. La estimación se basa en la experiencia, en datos históricos de proyectos pasados similares y en predicciones con cierto grado de confiabilidad.

Las características del proyecto que afectan la estimación son:

- Complejidad. Esta característica es relativa, ya que depende de la experiencia que el personal del proyecto tenga con proyectos similares. La complejidad también depende del tipo de aplicación a desarrollar, la cantidad y tipo de funciones que tenga.
- Tamaño. A mayor tamaño del sistema, mayor la cantidad de componentes que la forman, mayor la cantidad de dependencias posibles entre ellos y más difícil el análisis de estas funciones.
- Estructura. Es el nivel de detalle con el que se hayan definido los requisitos por los clientes, la facilidad con que puedan subdividirse funciones para su mejor análisis.

El riesgo es el grado de incertidumbre en la estimación. Los planificadores del proyecto deben tener especificaciones del sistema lo más completas posibles, revisar las estimaciones de manera frecuente y hacer actualizaciones al plan del proyecto. También deben identificarse los riesgos que sucederán durante el desarrollo y tomar medidas para eliminarlos o atacarlos si se presentan.

La planeación del proyecto comienza con una estimación de la estructura del producto, tamaño y funciones, así como un análisis de las restricciones que lo afectan (fecha de entrega, recursos disponibles, presupuesto, etc.) y el entorno en que funcionará el sistema. Se describe el control y datos a procesar, las funciones, el rendimiento, las restricciones, las interfaces y la fiabilidad. Dado que las estimaciones de costo y planificación temporal se basan en el análisis de las funciones, éstas pueden ser detalladas o descompuestas en subfunciones.

Una vez descrito el ámbito del software, debe analizarse la viabilidad del software, saber si éste se puede construir con las características y funciones que el cliente desea, con los recursos de personal, económicos y técnicos disponibles y con las restricciones de hardware, funcionamiento y tiempo de desarrollo dadas por el cliente.

Si el desarrollo del proyecto es viable, debe estimarse los recursos requeridos para llevarlo a cabo. Estos recursos se clasifican en:

- Recursos humanos. Se definen las habilidades necesarias para desarrollar el proyecto y el número de personas necesarias con estas habilidades.

- Recursos de entorno. Es el hardware y software necesario para desarrollar el proyecto. El hardware se refiere al tipo de computadora necesaria, con la velocidad de procesador adecuada y la memoria RAM y de disco duro necesaria, así como otros recursos como tarjetas de video, sonido o dispositivos periféricos como impresora, escáner, cámara, etc. El software se refiere a los lenguajes de programación, de bases de datos, software de apoyo a la administración del proyecto o software con alguna función especial necesaria para el desarrollo del proyecto.

Por último se hace una estimación del costo del proyecto y se elabora la planificación temporal. Esto no puede llevarse a cabo de forma exacta, ya que las variables que intervienen en esta estimación (personal, restricciones técnicas, restricciones de funcionamiento, requerimientos, entorno de negocios, etc.) son muchas y varían conforme el proyecto progresa. Sin embargo es posible seguir una serie de pasos para estimar el costo y la planificación temporal del proyecto con un grado de riesgo aceptable. Las opciones son:

- Basarse en proyectos muy parecidos hechos con anterioridad. Para esto debe tenerse datos históricos de otros proyectos y éstos deben ser muy parecidos en las funciones, restricciones, presupuesto, tiempo de entrega, etc.
- Utilizar técnicas de descomposición de las actividades a realizar durante el proyecto, con esto se tiene de forma más detallada la serie de actividades a realizarse y puede definirse el costo y el tiempo de cada actividad con más exactitud.

1.9.1 Plan del proyecto

Un plan de proyecto está formado por las siguientes secciones:

1. Introducción. Describe los objetivos del proyecto y las restricciones que afectan su administración.
2. Organización del proyecto. Describe el equipo de trabajo, su organización, la gente involucrada y sus tareas en el equipo.
3. Análisis de riesgos. Describe los posibles riesgos del proyecto, la probabilidad de que ocurran y las estrategias de reducción de riesgos y combate de riesgos.
4. Requerimientos de hardware y software. Describe los elementos de hardware y software necesarios para desarrollar el proyecto.
5. División del trabajo. Describe las actividades que conforman el proyecto, los puntos en que acaban las actividades principales del desarrollo (hitos) y los productos y documentación a entregar asociados con cada actividad.
6. Programa del proyecto. Describe las actividades del proyecto, las dependencias entre actividades, el tiempo estimado para terminarlas y la asignación de personal a cada actividad.
7. Mecanismos de supervisión e informes. Describen cuándo deben producirse revisiones e informes del proyecto y cómo llevarlos a cabo.

1.9.2 Hitos y productos a entregar

Dado que el software es intangible, la única manera de conocer el avance del proyecto es mediante documentos u otros productos de las actividades, que nos informen del estado de este avance. Para esto, en la planeación del proyecto se establecen hitos (puntos finales de una actividad en el proceso del software) y en cada hito debe producirse un resultado, el cual servirá para conocer el progreso del desarrollo.

1.10 Planificación temporal del proyecto

Es una parte muy importante de la planificación del trabajo de desarrollo del proyecto de software. La planificación temporal consiste en dividir el trabajo total de desarrollo, en actividades principales, las cuales a su vez se dividen en tareas, se estima los recursos necesarios para realizar las actividades, se estima el tiempo que se requiere para llevar a cabo cada tarea y se asigna cada tarea a una persona o a un equipo de trabajo.

Durante la planificación temporal también debe calcularse los riesgos o problemas que posiblemente sucederán y tener un plan de acción en el caso de que ocurran. Y aún si ocurren problemas no previstos, debe haber revisiones periódicas y formales al calendario del proyecto, con el fin de verificar que el avance sea conforme al calendario y en el caso de no serlo, modificar las actividades y sus tiempos de manera que el proyecto total se acomode al tiempo estipulado de finalización.

1.10.1 Método de Ruta Crítica y PERT

Estos dos métodos se expresan como una red de actividades. Hay un punto de partida o inicio del proyecto, a partir del cual se puede hacer varias actividades en paralelo. Se caracteriza porque cada actividad requiere que las anteriores hayan sido terminadas para comenzar. Ambos métodos usan diagramas de red para representar gráficamente las actividades.

Método de Ruta Crítica

Se usa cuando el proyecto es similar a otros proyectos realizados anteriormente o cuando ya se conoce el tiempo necesario para desarrollar cada actividad. Consiste en acomodar las tareas en el orden en que serán hechas, considerando las dependencias entre ellas y tomando en cuenta las duraciones de cada actividad, se identifica la secuencia de tareas cuyo atraso o adelanto al ejecutarlas afecta al avance del proyecto y se llama a esta secuencia de tareas la ruta crítica del proyecto. Definir la ruta crítica y monitorearla es importante para asegurar que el avance del proyecto sucede conforme al plan.

Método PERT (Técnicas de evaluación y revisión de programa)

PERT se usa cuando el proyecto es diferente a los hechos anteriormente y no se sabe con certeza la duración de cada actividad y los tiempos se calculan de manera probabilística. Para cada actividad, se calcula el tiempo más optimista (o), el más pesimista (p) y el tiempo esperado de duración (m) y se aplica la siguiente fórmula para calcular el tiempo de duración de la actividad:

$$e = \frac{o + 4m + p}{6}$$

1.10.2 Gráfico de tiempo (gráfico de Gantt) y redes de actividades

Los gráficos se crean cuando ya se conocen las actividades a desarrollar. Estas actividades se dividen en tareas específicas, las cuales pueden estar dispuestas en una red de tareas. A partir de esto se genera un gráfico de barras con las actividades y sus divisiones en tareas, así como la posición en el tiempo y duración de cada tarea. También se genera una red de actividades, la cual nos indica además las dependencias entre las diferentes actividades que conforman el proyecto.

La figura 1.4 es un ejemplo de red de actividades. Se observa que las actividades o tareas se dibujan como rectángulos y se indica sobre ellos el tiempo que toma realizar la tarea (horas, dìas, etc.). Cada tarea al ser terminada conduce a un punto en el proyecto (milestone) indicado mediante elipses, sobre las cuales se indica la fecha en la que se planea llegar a ese punto en el proyecto. Un milestone es el punto de partida de una o más tareas y puede ser el punto al que llegan dos o más tareas, indicando las tareas que deben ser realizadas para poder realizar la tarea siguiente (dependencias).

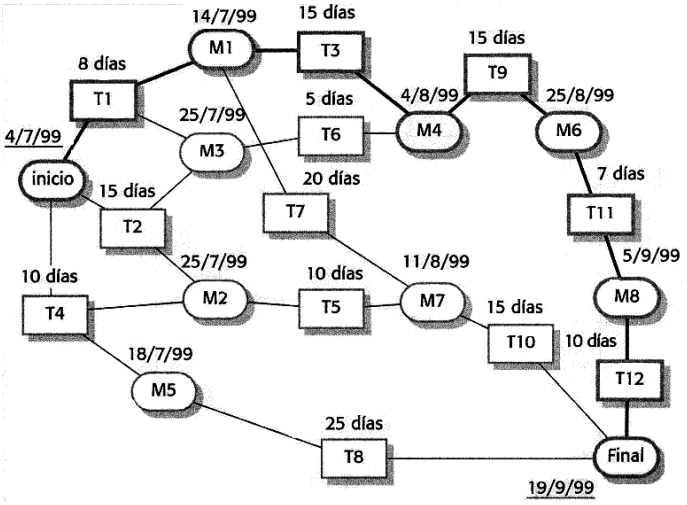


Fig. 1.4 Red de actividades[11]

La figura 1.5 muestra un ejemplo de Gráfico de Gantt. En un gráfico de este tipo se observan las actividades divididas en tareas. Se observa claramente la duración de cada tarea y también es posible observar aquellas tareas que se realizan en paralelo (al mismo tiempo) o aquellas que dependen de la terminación de otra u otras para poder comenzar.

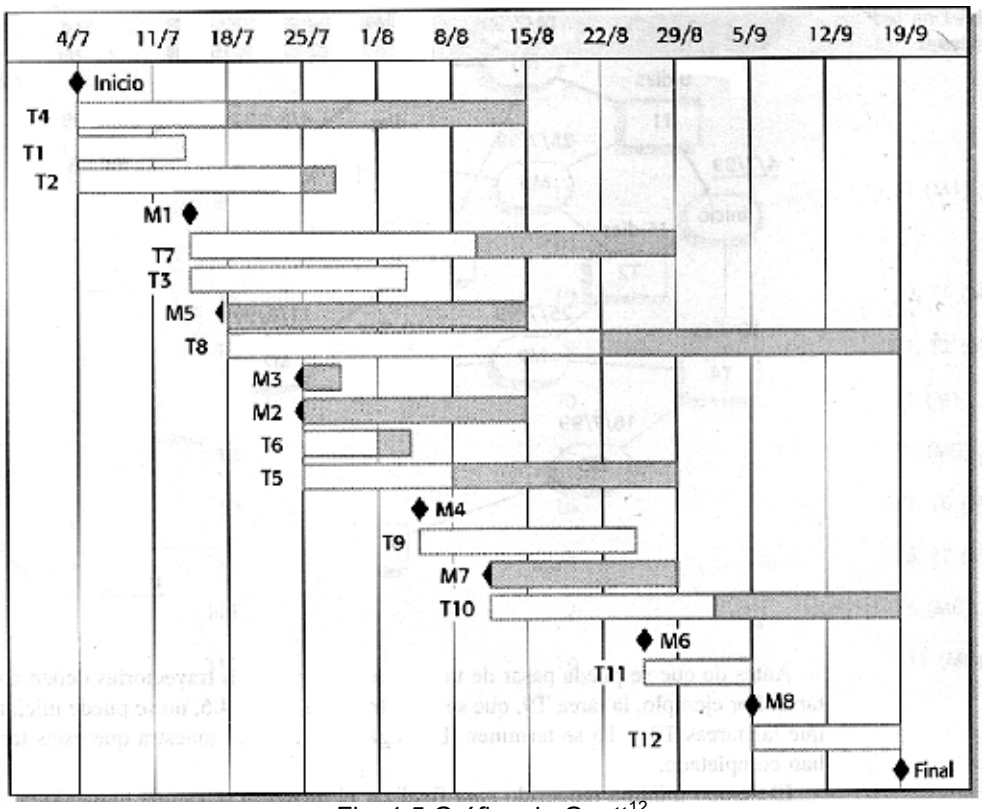


Fig. 1.5 Gráfico de Gantt[12]

[11] Sommerville, Ian. *Ingeniería de software*. Addison-Wesley, Sexta edición, Reino Unido, 2002.
[12] Ídem.

Capítulo 2
Administración de riesgos

2.1 Definición

Un riesgo es un posible problema que puede afectar la programación del proyecto o la calidad del software a desarrollar. El riesgo tiene dos características muy importantes:

Incertidumbre. El riesgo puede ocurrir con cierta probabilidad o puede no ocurrir.

Pérdida. Si el riesgo ocurre, tiene consecuencias no deseadas o pérdidas.

Las estrategias para afrontar los riesgos pueden ser reactivas o proactivas. Las estrategias reactivas son aquellas que no identifican los posibles riesgos a priori y como su nombre lo indica, se refiere a una reacción cuando el riesgo ya sucedió y se convirtió en problema, para resolverlo lo más rápido posible. Lo mejor es tener una estrategia proactiva, la cual comienza antes de las actividades técnicas y en la cual se identifican por adelantado los riesgos potenciales, se evalúa su probabilidad y su impacto y con base en la importancia de los riesgos identificados se establece un plan para evitar el riesgo o para minimizar sus consecuencias.

La administración de riesgos es una estrategia de riesgos proactiva. Es un conjunto de actividades cuyo fin es evitar los riesgos, mediante acciones que prevengan su ocurrencia o bien que minimicen sus efectos si éstos ocurren.

Existen varias categorías de riesgos. Una de ellas clasifica a los riesgos de acuerdo al aspecto que afecta su ocurrencia[13]:

- Riesgos del proyecto. Afectan a la calendarización o los recursos del proyecto.
- Riesgos del producto. Afectan a la calidad o desempeño del software.
- Riesgos del negocio. Afectan a la organización que desarrolla el software.

El tipo de riesgos que se presentan durante un proyecto de software dependen de muchos factores: el tipo de proyecto, su complejidad y tamaño, la forma en que se define los requerimientos y la posibilidad de que estos cambien con facilidad y frecuencia, la dificultad con la que se definen los recursos necesarios para desarrollar el software, el tipo de personal necesario y el tipo de personal del que se disponga, las restricciones de tiempo y dinero, etc. Las actividades de la administración de riesgos ayudan a identificar los posibles problemas y definir actividades que sirvan para evitar que esos problemas lleguen a suceder o minimizar sus efectos si suceden.

La figura 2.1 muestra las actividades que conforman la administración de riesgos, así como el orden en que estas se ejecutan:

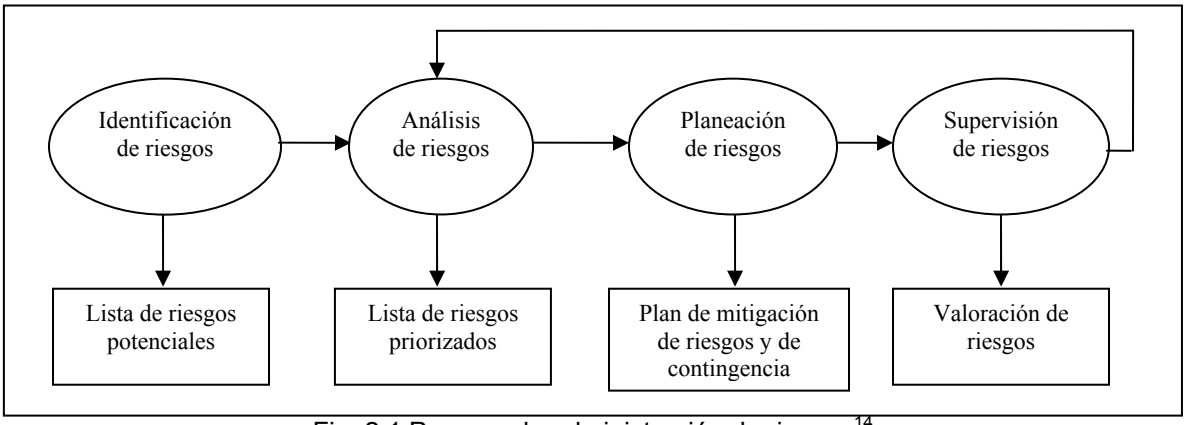


Fig. 2.1 Proceso de administración de riesgos[14]

[13] Sommerville, Ian. *Ingeniería de software*. Addison-Wesley, Sexta Edición, Reino Unido, 2002.
[14] Ídem.

2.2 Identificación de riesgos

Comprende el descubrimiento de los posibles riesgos del proyecto. Esto puede llevarse a cabo mediante una reunión del equipo de desarrollo y los clientes y mediante una lluvia de ideas o puede ser responsabilidad de los encargados de la administración del proyecto.

Como ayuda al proceso puede utilizarse una lista de tipos de riesgos, como la siguiente:

- Riesgos de tecnología. Derivados de la tecnología de software y hardware utilizada en el proyecto.

- Riesgos de personas. Relacionados con las personas involucradas en el proyecto, en cuanto a comunicación, experiencia y conocimientos necesarios, etc.

- Riesgos organizacionales. Relacionados con el entorno organizacional en que se desarrolla el proyecto o aquel en el que será utilizado el software.

- Riesgos de requerimientos. Riesgos relacionados con los posibles cambios en los requerimientos y la manera en que afectan al proyecto.

- Riesgos de estimación. Relacionados con la definición de los recursos necesarios para desarrollar el proyecto.

- Riesgos del producto. Riesgos asociados con el tamaño general, complejidad y otras características estructurales del software a desarrollar.

- Riesgos de negocio. Riesgos asociados con el posible mercado en que el software será vendido.

- Riesgos del proceso. Posibles problemas con la definición del proceso de desarrollo y con la facilidad para seguirlo por el equipo de desarrollo.

2.3 Análisis de riesgos

Una vez que se identifican los riesgos que pueden afectar al trabajo de desarrollo y que se tiene los riesgos en una lista, se procede a medir cada riesgo en cuanto a la probabilidad de que suceda y al impacto o consecuencias que tendría en el trabajo de desarrollo. Para esto se genera una tabla de riesgos de la siguiente manera:

1. Se desarrolla una escala de valoración de la probabilidad y del impacto

La escala de probabilidades puede consistir en asignar un valor de probabilidad al riesgo entre 0% y 100%. Como no es posible saber con exactitud la probabilidad de que un riesgo suceda, por lo general se utilizan rangos de valores de probabilidad.

Otra forma de asignar un valor de probabilidad es mediante letras tales como A (alta), M (media) y B (baja) y asignarles los valores 3, 2 y 1 respectivamente. Estos valores numéricos servirán para calcular la seriedad del riesgo utilizando en conjunto la probabilidad y el impacto.

Para definir una escala de impacto se utiliza un procedimiento similar. Es muy difícil asignar un costo monetario a un impacto de un riesgo, por lo cual es más fácil utilizar valores de impacto tales como A (alto), M (mediano) y B (bajo). Al igual que con las probabilidades, se asignan los números 3, 2 y 1 respectivamente a los valores de impacto.

2. Se define las consecuencias del riesgo

Para valorar las consecuencias de un riesgo, debe tomarse en cuenta su naturaleza (los problemas que ocurrirán), su alcance (la severidad del riesgo y las partes del trabajo de desarrollo que serán afectadas)

y el momento en que sucederá y por cuánto tiempo se sentirán las consecuencias. El tipo de riesgo, las partes afectadas y el momento en que se considera que sucederá el riesgo, deben ser anotadas en una columna junto a la descripción del riesgo.

3. Se asigna valores de probabilidad e impacto a los riesgos

Con base en los dos puntos anteriores, se estudia cada riesgo para asignarle una probabilidad y un valor de impacto. Estos dos valores se incluyen cada uno en una columna en la tabla de riesgos.

4. Se escoge los riesgos a supervisar

Se estudia la tabla de riesgos anterior y con base en los valores de probabilidad e impacto de cada riesgo, se eligen los riesgos más peligrosos, para elaborar un plan de mitigación y contingencia. Una forma de elegir los riesgos, es tomar todos aquellos que tengan impactos de tipo alto y todos los riesgos de impacto mediano cuya probabilidad sea mediana o alta. Otra forma de hacerlo es multiplicar los valores numéricos de probabilidad e impacto y tomar aquellos riesgos cuyos productos de ambos valores sean los más altos. Debe tenerse cuidado, ya que esta forma de escoger los riesgos puede dejar fuera de consideración los riesgos con probabilidad muy baja, pero con impacto muy alto. De suceder estos riesgos, los resultados pueden ser muy costosos.

En general, debe tomarse un número de riesgos que sea manejable. Esto se debe a que la administración de los riesgos no debe elevar los costos del proyecto hasta un nivel en el cual no convenga realizarlo.

2.4 Planeación de riesgos

En este paso del proceso, se desarrolla una estrategia, un conjunto de actividades que se realizan junto con las actividades técnicas, para tratar los riesgos escogidos, con tres objetivos:

- Evitar los riesgos. Se realizan acciones de mitigación de los riesgos, cuyo fin es evitar las condiciones que provocan que surjan los problemas. Lo ideal es evitar los riesgos.
- Supervisar los riesgos. El objetivo es monitorear los riesgos para saber cuáles riesgos se han convertido en problemas, cuáles se han podido evitar, cuáles siguen siendo posibles problemas y para identificar nuevos riesgos que hayan surgido.
- Administrar los riesgos y llevar a cabo planes de contingencia. Se planean las actividades de supervisión de riesgos, así como se planean las actividades de mitigación de los riesgos y los planes de contingencia, destinados a resolver los riesgos que se han convertido en problemas lo más pronto posible y con el menor impacto posible.

2.5 Supervisión de riesgos

Evitar el riesgo es siempre lo mejor. Para esto se lleva a cabo un plan de reducción o mitigación del riesgo, con el fin de disminuir la probabilidad de que ese riesgo surja o al menos reducir el impacto del riesgo si este sucede.

La supervisión de los riesgos se realiza a medida que avanza el proyecto. El líder de proyecto o el encargado de la supervisión de los riesgos revisa los factores que pueden indicar si los riesgos se están haciendo más o menos probables y si el impacto de los riesgos se está haciendo menor o mayor. Además revisa la efectividad real de las actividades que se definió para la reducción del riesgo.

Cuando un riesgo sucede y se ha convertido en problema, se llevan a cabo los planes de contingencia, definidos para solucionar el problema lo más pronto posible y con el menor impacto posible.

Conforme el proyecto va a avanzando y los riesgos han sido evitados, mitigados o resueltos, la administración de los riesgos se encarga de revisar los riesgos restantes, así como nuevos riesgos que hayan sido identificados, para tener actualizado el plan de riesgos.

Capítulo 3
Ingeniería de requerimientos

Definición

Son las actividades realizadas para la obtención, documentación, verificación y validación de los requerimientos del sistema. Estas actividades permiten obtener, especificar, documentar y validar con el cliente las características de las funciones del sistema, los datos que se manejan y la forma en que se manejan y los atributos de calidad deseados para el sistema.

La ingeniería de requerimientos consta de las siguientes actividades principales:[15]

- Estudio de factibilidad del sistema.
- Obtención y análisis de requerimientos.
- Especificación y documentación de requerimientos.
- Validación de requerimientos.

La figura 3.1 muestra las actividades nombradas anteriormente y los productos resultantes de estas:

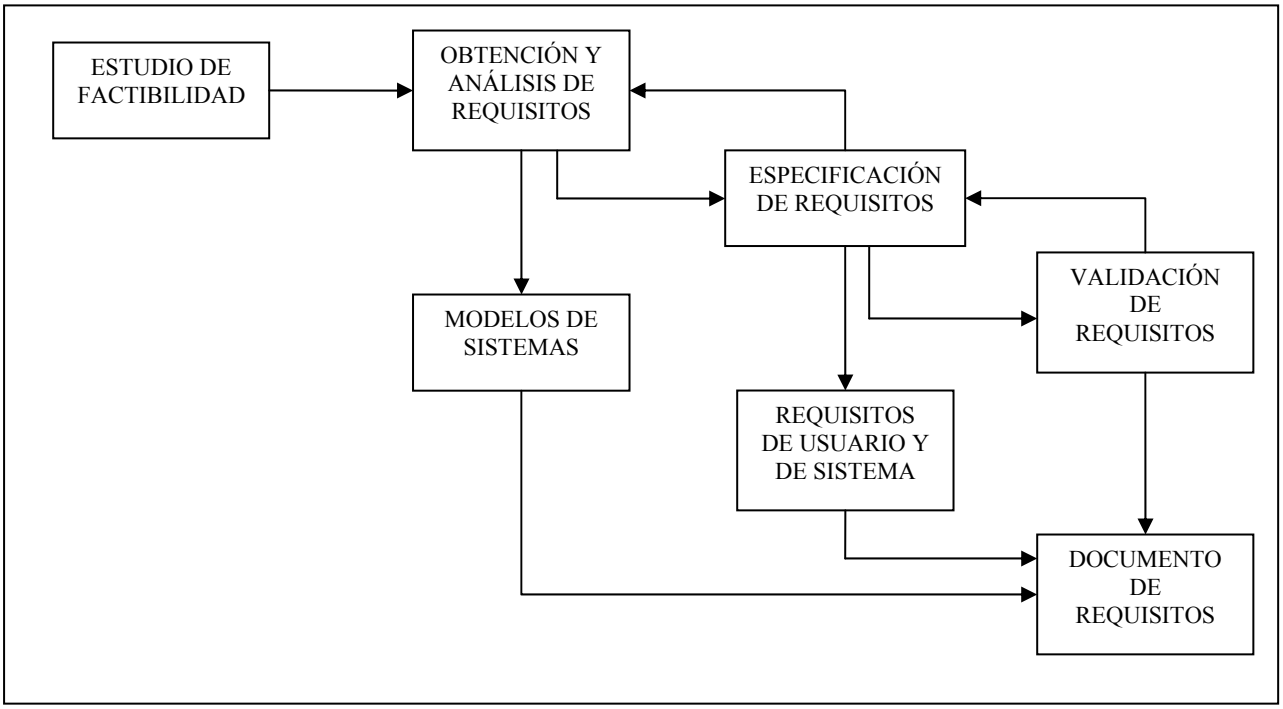


Fig. 3.1 Pasos de la ingeniería de requisitos

Estudio de factibilidad

El análisis de factibilidad determina si es posible llevar a cabo el desarrollo de un sistema que contribuya a los objetivos de negocio del cliente, con los recursos existentes de tecnología, económicas, de tiempo, de personal, etc. y con las restricciones impuestas. El desarrollo del sistema es factible si se puede idear una estrategia en la cual los requisitos establecidos para el proyecto de desarrollo se pueden satisfacer dentro de las restricciones de tecnología, económicas, de tiempo, de personal, etc.

Para realizar el análisis se toma una descripción resumida del sistema y de cómo se utilizará dentro de la organización del cliente. El estudio de factibilidad responde a varias preguntas, como las siguientes:

- ¿El sistema contribuye a los objetivos generales de la organización?
- ¿Cómo ayudará el sistema a resolver los problemas con los procesos actuales?
- ¿Se puede implementar el sistema utilizando la tecnología actual?

[15] Sommerville, Ian. *Ingeniería de software*. Addison-Wesley, Sexta Edición, Reino Unido, 2002.

- ¿Se puede implementar el sistema con las restricciones de costo y tiempo?
- ¿El sistema puede integrarse al entorno de la organización?

Primero se define la información necesaria cuya evaluación contestará a las preguntas anteriores. Las fuentes de información para responder a estas preguntas, pueden ser los gerentes de los departamentos donde el sistema se usará, los ingenieros de software relacionados con el sistema, los usuarios finales, etc. Posteriormente se analiza la información recabada y se obtiene como resultado del estudio, un documento en el que se especifica si es conveniente llevar a cabo el desarrollo del sistema tal como está definido y en caso necesario, propone cambios en el alcance, presupuesto o calendario de desarrollo. También puede proponerse cambios en los requerimientos de alto nivel del sistema o sugerir nuevos requerimientos.

El análisis de factibilidad comprende cinco aspectos a analizar:

1. Factibilidad técnica
2. Factibilidad económica
3. Factibilidad legal
4. Factibilidad operacional
5. Factibilidad de programa

Cada aspecto de los mencionados tiene un peso distinto dependiendo de las características del proyecto de desarrollo.

Factibilidad técnica

Se determina si es posible desarrollar e implementar el sistema utilizando la tecnología existente.

Factibilidad económica

Se determina si se cuenta con los fondos necesarios para desarrollar e implementar el sistema.

Factibilidad legal

Se determina si la organización que desarrolla e implementa el sistema tiene o no problemas de tipo legal al hacer este trabajo. Se verifica la legalidad del uso y las licencias del software empleado para el desarrollo del sistema, así como la legalidad en la contratación del personal, legalidad económica de la organización, etc.

Factibilidad operacional

Se determina si se cuenta con la experiencia y conocimientos necesarios en la tecnología que será necesaria para desarrollar e implementar el sistema y en el caso de que no se cuente con ellos, se determina si es posible y factible adquirirlos.

Factibilidad de programa

Se determina si es posible cumplir con la planeación temporal de las actividades del proyecto de desarrollo. Es decir, se determina si es posible cumplir con el plazo indicado para el proyecto.

Obtención y análisis de requerimientos

El personal de desarrollo del software trabaja con los clientes y usuarios finales para determinar el entorno donde se usará la aplicación, qué servicios o funciones proveerá el sistema, cuál es el desempeño requerido, las restricciones de hardware, reglas de negocio, etc. Es un proceso iterativo que comprende los siguientes pasos:

1. Comprensión del dominio o ambiente en que operará el sistema.
2. Recolección de requerimientos de los clientes y usuarios.
3. Clasificación de los requerimientos en grupos por tipo.

4. Acuerdo entre todos los implicados en el desarrollo y uso del sistema de software, acerca de la importancia de los requerimientos obtenidos.
5. Priorización. Se ordenan los requerimientos según su importancia.
6. Verificación de los requerimientos para comprobar su completitud, consistencia y que están de acuerdo con las necesidades reales de los clientes y usuarios.

La obtención y análisis de requerimientos es un proceso iterativo durante el cual se va mejorando la comprensión del entorno en que la aplicación funcionará, de los requerimientos del cliente y de las funciones que realizará el sistema.

Técnicas de obtención y análisis de requerimientos

Normalmente se utiliza una combinación de las siguientes técnicas para tener una recolección y entendimiento completos de los requerimientos.

Escenarios

Mediante este enfoque, los usuarios del sistema utilizan ejemplos de situaciones reales de interacción de los usuarios con el sistema para definir con detalle los requerimientos. De manera general, un escenario incluye:

- Estado del sistema al inicio del escenario. Se definen las condiciones en las cuales el sistema se encuentra al comenzar la interacción que define el escenario. Estas condiciones son necesarias para que sea posible ejecutar las acciones del escenario de forma correcta.
- Flujo de eventos en el escenario. Se define la serie de acciones que el usuario realiza y las reacciones del sistema a dichas acciones.
- Flujo de datos en el escenario. Se define cómo los datos son introducidos por el usuario y los utilizados por el sistema son transformados durante la secuencia de acciones del escenario.
- Posibles problemas y cómo manejarlos. Se define cómo reaccionará el sistema ante un problema que puede ser provocado por la falta o incorrección de alguna condición o dato necesario y de ser posible, cómo resolver esta situación o al menos darla a conocer al usuario.
- Otras actividades que podrían realizarse al mismo tiempo en el sistema.
- Estado del sistema al final del escenario. Se definen las condiciones en las cuales el sistema se encuentra al terminar la interacción que define el escenario.

Análisis estructurado

Se basa en la creación de modelos de los objetos de datos y control y de las funciones del sistema. Este enfoque documenta la especificación del sistema como un conjunto de modelos gráficos que describen sus funciones y comportamiento. “Estos modelos representan al sistema desde varios puntos de vista:

1. Una perspectiva del entorno o ambiente en que se usará el sistema.
2. Una perspectiva del comportamiento del sistema.
3. Una perspectiva de la arquitectura del sistema y de la estructura de los datos que este procesa.”[16]

Los diferentes tipos de modelos tienen distintos niveles de abstracción o detalle de las funciones y datos del sistema.

Los tipos de modelos son:

- Modelos que muestran cómo se procesan los datos en el sistema, en las diferentes funciones y etapas de éstas.
- Modelos que muestran las entidades participantes en el sistema, sus características y las relaciones entre las entidades.
- Modelos arquitectónicos que muestran los subsistemas que componen al sistema.
- Diagramas de estados que muestran cómo el sistema reacciona a estímulos externos o internos.

[16] Pressman, Roger. *Ingeniería del software*. McGraw-Hill, Quinta edición, España, 2002.

Modelos de contexto

Estos son modelos muy generales del sistema y sirven para definir el entorno y límites del sistema, sus partes o subsistemas principales, las bases de datos que utiliza, etc.

Modelos de comportamiento

Se usan para describir el comportamiento completo del sistema. Existen dos tipos de estos modelos, que pueden usarse juntos o separados:

Modelos de flujo de datos

Modelan el comportamiento de los datos dentro del sistema. Sirven para modelar de manera sencilla cómo los datos fluyen a través de los procesos del sistema y son transformados en cada proceso. Cada proceso será una función que realizará el sistema.

Puede partirse de un modelo de flujo de datos general de todo el sistema e ir expandiendo el modelo de flujo de datos de cada función cada vez con más detalle. La figura 3.2 muestra un ejemplo de modelo de flujo de datos general o nivel cero de un sistema. Se observa en los rectángulos las partes del sistema encargadas de enviar y recibir los datos y en las flechas se observa la información que es enviada entre las partes del sistema. El modelo de flujo de datos general o nivel cero no muestra aún cómo se generan, procesan o utilizan los datos o información, sino solamente cuál es su origen y destino.

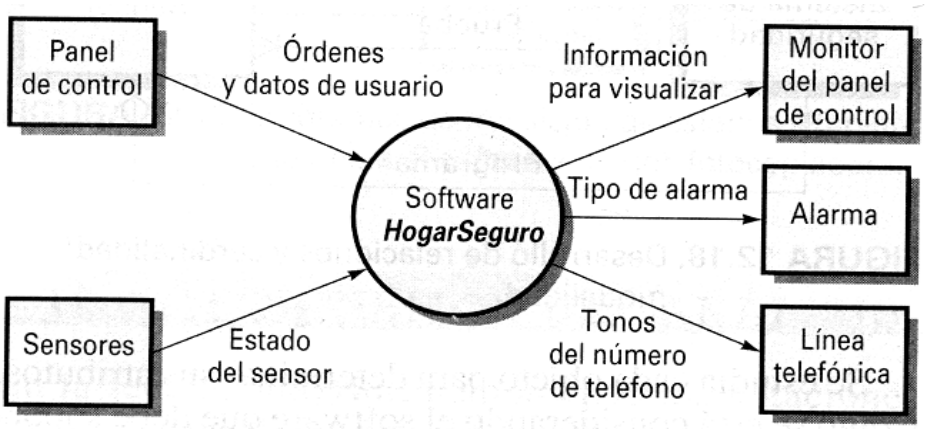


Fig. 3.2 Ejemplo de diagrama contextual o nivel cero de flujo de datos[17]

[17] Pressman, Roger. *Ingeniería del software*. McGraw-Hill, Quinta Edición, España, 2002.

La figura 3.3 muestra un diagrama de flujo de datos más detallado. Partiendo del diagrama de flujo de datos general, se detalla cada parte del sistema basándose en las funciones realizadas. En un diagrama de flujo de datos más detallado se observan nuevos elementos como las funciones (indicadas por círculos), las cuales reciben los datos provenientes de otra actividad o de una parte del sistema y como resultado de las funciones se generan otros datos. También se observan elementos tales como almacenes de datos (indicados entre dos líneas paralelas) en los cuales se consulta, almacena o actualiza la información relacionada con las actividades del sistema.

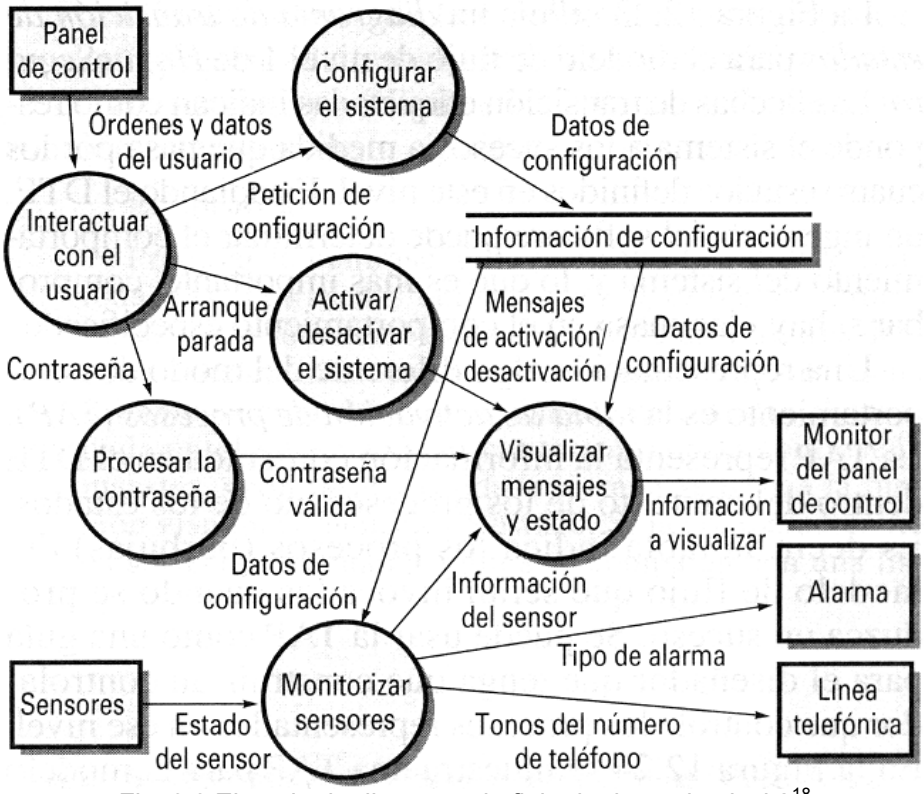


Fig. 3.3 Ejemplo de diagrama de flujo de datos de nivel 2[18]

[18] Pressman, Roger. *Ingeniería del Software*. McGraw-Hill, Quinta edición, España, 2002.

Modelos de máquinas de estado

Sirven para modelar el comportamiento del sistema como respuesta a eventos internos o externos. El sistema se encontrará en un cierto estado hasta que un evento interno o externo lo haga pasar a un estado diferente.

La figura 3.4 muestra un ejemplo de diagrama de máquina de estado. Los rectángulos muestran los estados en que puede encontrarse el sistema. Las flechas que salen de los estados indican los eventos y/o información que provocan que el sistema pase a otro estado.

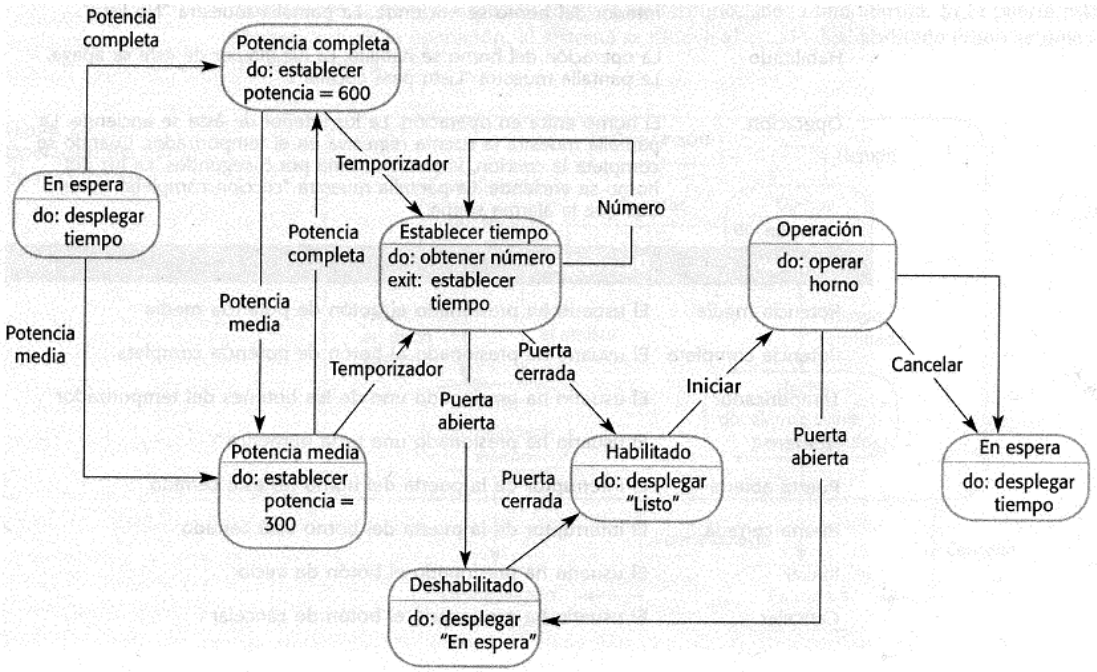


Fig. 3.4 Ejemplo de diagrama de máquinas de estado de un horno de microondas[19]

[19] Sommerville, Ian. *Ingeniería de software*. Addison-Wesley, Sexta Edición, Reino Unido, 2002.

Modelos de datos

Éstos nos definen las propiedades de los datos en sí, las entidades de datos existentes en el sistema, sus atributos y las relaciones existentes entre las entidades.

El modelo de datos más utilizado es el modelo Entidad-Relación (E/R). La figura 3.5 muestra un ejemplo de este modelo. Se observa a cada entidad de datos representada por un rectángulo, a las propiedades o atributos de las entidades representadas por elipses y a las relaciones entre las entidades representadas por rombos. También se indica en el diagrama la cardinalidad de la relación entre las entidades). Ejemplos más complejos muestran propiedades compuestas de otras propiedades o bien entidades cuya existencia depende de otras entidades.

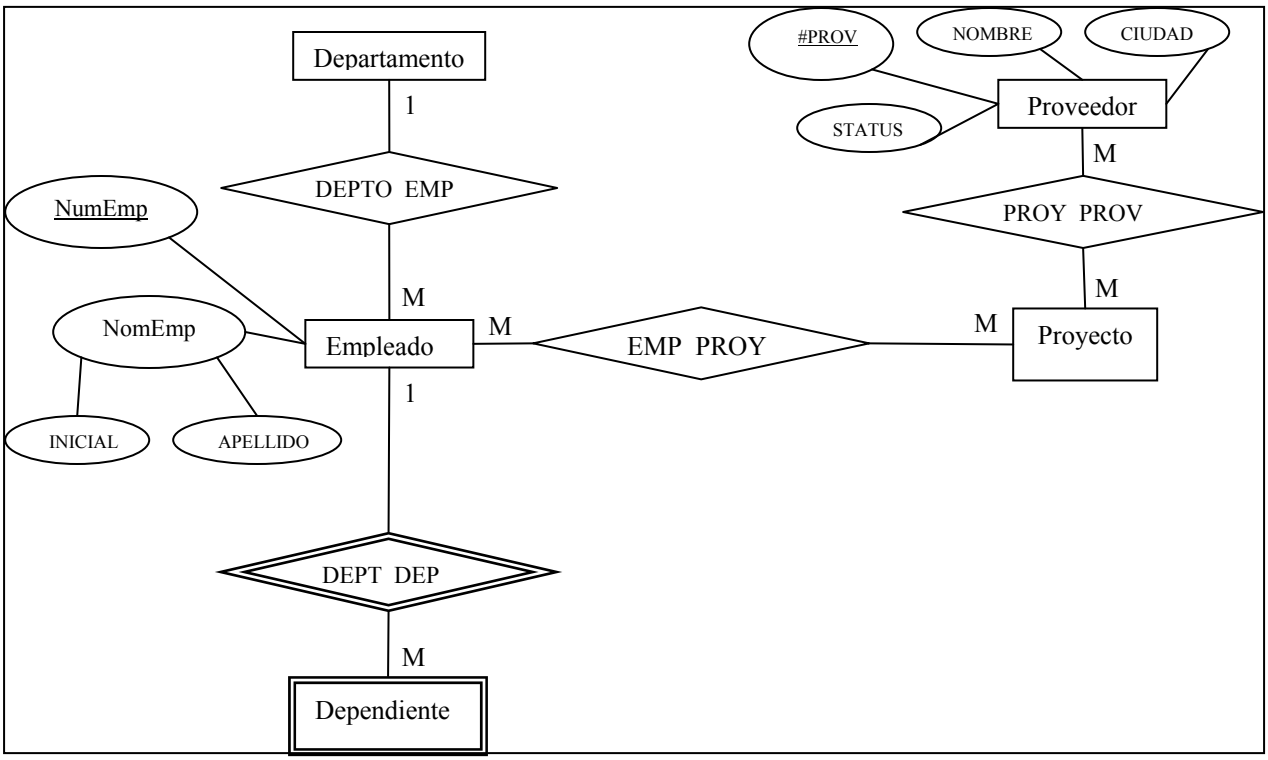


Fig. 3.5 Diagrama Entidad – Relación

El modelo E/R se complementa con un diccionario de datos, el cual es una lista alfabética en la que se detallan las descripciones de las entidades, atributos y relaciones que se encuentran en el modelo.

Construcción de prototipos

Esta técnica se usa para sistemas muy grandes y/o complejos, para los cuales es difícil o imposible tener una especificación total y detallada de los requerimientos. En este caso se recurre a la construcción de prototipos, los cuales son probados por el cliente y de esto surgen nuevos requerimientos, así como cambios en las funciones que realiza el prototipo.

Un prototipo sirve para verificar, validar o cambiar los requerimientos ya establecidos y para comprender mejor el problema que se desea solucionar y por consiguiente encontrar nuevos requerimientos.

La figura 3.6 muestra el proceso general que se sigue al desarrollar un prototipo para un sistema. Se comienza por definir los objetivos del prototipo, se diseña su funcionalidad, se construye y se evalúa para validar los requerimientos del sistema.

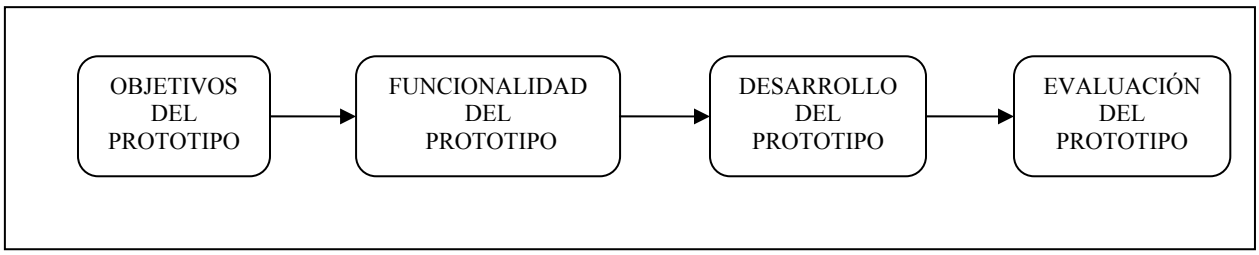


Fig. 3.6 Proceso del desarrollo de prototipos

Existen dos formas para desarrollar prototipos:

Construcción de prototipos evolutivos

Consiste en desarrollar una solución inicial, simple, que implementa los requerimientos más importantes para el usuario. Al ser evaluado por el cliente, el prototipo cambia y crece con las modificaciones y nuevos requerimientos que surgen. Estos cambios y mejoras se llevan a cabo sobre el mismo prototipo que fue mostrado al usuario. Después de varias evaluaciones y los consiguientes cambios y mejoras, se tiene una solución final definitiva.

Esta forma de construcción de prototipos tiene las siguientes características:

- El sistema puede desarrollarse rápidamente.
- Tanto desarrolladores como usuarios se involucran en la creación del sistema.
- No existe una especificación detallada del sistema y dados los cambios y mejoras frecuentes en los requerimientos, la documentación del sistema puede ser muy costosa o complicada su actualización o bien estará incompleta.
- Dado que no hay una especificación de requerimientos como tal, no es posible hacer una verificación de los mismos. Más bien se hace una validación de que el sistema realiza las funciones que el usuario necesita. Esto puede provocar problemas cuando hay usuarios con distintas ideas sobre lo que el sistema debería hacer.

Una forma de solucionar los problemas causados por el desarrollo de prototipos evolutivos es utilizar el paradigma de desarrollo incremental, ya que mediante este método se crean partes del sistema, que una vez que son aprobados ya no se cambian. Esto facilita la documentación de las partes aceptadas y éstas se van integrando a las otras, hasta tener un sistema y una documentación completos.

Construcción de prototipos desechables

En este enfoque, el prototipo construido tiene la finalidad de validar y verificar los requerimientos obtenidos, así como obtener nuevos requerimientos y en general mejorar la comprensión del problema. Una vez logrado esto, el prototipo se desecha y la parte del sistema real que abarcaba el prototipo se desarrolla.

Características de la construcción de prototipos desechables:

- Con el prototipo desechable se comprenden mejor las funciones del sistema.
- Al crear los prototipos, los desarrolladores no se preocupan por tener la mejor calidad ni por cumplir las restricciones de los requerimientos. Esto se asegurará en el sistema real.
- En los prototipos implementados, no se puede probar atributos de calidad tales como los requerimientos de confiabilidad (control y recuperación de fallos), robustez (manejo de excepciones) y seguridad (manejo de accesos y permisos), ya que éstos generalmente no tienen tanta importancia o prioridad al crear el prototipo como la funcionalidad en sí.

Validación de requerimientos

Esta etapa sirve para demostrar que los requerimientos obtenidos son los que realmente desea el cliente. Esto es muy importante ya que es mucho más rápido y menos costoso corregir la definición del sistema durante la obtención y documentación de los requerimientos, que después del trabajo de diseño e implementación, ya que estos dos últimos trabajos deberán realizarse de nuevo al cambiar un requerimiento.

Hay varios tipos de verificación de requerimientos:

- Verificación de validez. Los requerimientos obtenidos deben satisfacer las necesidades de todos los usuarios del sistema.
- Verificación de consistencia. Los requerimientos no deben contradecirse. Tampoco puede haber restricciones contradictorias o definiciones diferentes para una misma función del sistema.
- Verificación de integridad. Los requerimientos deben definir todas las funciones y restricciones del sistema.
- Verificación de realismo. Los requerimientos deben poder ser implementados con la tecnología existente y deben tomar en cuenta el presupuesto disponible y la calendarización de desarrollo existente.
- Verificabilidad. Los requerimientos deben redactarse de forma que sea fácil su verificación.

Técnicas de verificación de requerimientos.

1. Revisiones de requerimientos

Personal, tanto del equipo de desarrollo como de los clientes, revisa el documento de requerimientos. Si se hace informalmente, se tratará simplemente de una discusión de los requerimientos con todos los usuarios posibles y verificar que los requerimientos establecidos en el documento son lo que realmente desean. De manera formal, al revisar los requerimientos se verifica la consistencia de los requerimientos y su integridad como un todo. También debe comprobarse la verificabilidad de los requerimientos, su comprensión total por los usuarios, debe establecerse claramente el origen del requerimiento y debe comprobarse que los requerimientos pueden cambiarse sin impactos grandes en otros requerimientos del sistema.

2. Construcción de prototipos

Como ya se explicó antes, se crean modelos ejecutables del sistema con base en los requerimientos. Estos modelos son evaluados por los usuarios para verificar que se cumple con sus necesidades reales.

Administración de requerimientos

Los requerimientos de sistemas de software siempre están cambiando a lo largo del trabajo del proyecto. Esto se debe a la diversidad de puntos de vista de los distintos usuarios o a cambios en el entorno de negocios o técnico en el que el sistema se está desarrollando y/o se opera. La comprensión y control de los cambios en los requerimientos es muy importante.

La planeación de la administración de requerimientos consta de los siguientes pasos:

1. Identificación de requerimientos. Cada requerimiento debe poder identificarse de forma única mediante una clave o número. Esto con el fin de facilitar su referencia en otros requerimientos relacionados o en otras fases del desarrollo del sistema.
2. Administración del cambio. Se analizan y llevan a cabo los cambios en los requerimientos de forma controlada. Al ocurrir un cambio en los requerimientos se realizan los siguientes pasos:
   a) Análisis y especificación del cambio. Se analiza el problema encontrado en los requerimientos o la propuesta de cambio, para verificar su validez.
   b) Análisis y costo del cambio. Si el cambio es válido, se analiza su impacto en el costo de desarrollo del sistema, en lo que se refiere a gastos económicos, cambios en el diseño e implementación y retrasos en el calendario del proyecto. Con base en este análisis se decide si se realiza el cambio o no.
   c) Implementación del cambio. Si el cambio procede, se modifica el documento de requerimientos y si ya se ha trabajado en el diseño e implementación del sistema, se hacen los cambios necesarios en éstos, afectados por el cambio en los requerimientos.

Capítulo 4
Diseño

Introducción

Una vez que se ha hecho el análisis de requisitos y la especificación del sistema, siguen las tres actividades técnicas del proceso del software: diseño, implementación (generación de código) y pruebas.

El objetivo de la actividad de diseño es crear los modelos con base en los cuales se construirá el software. Estos modelos se basan a su vez en los modelos creados durante la etapa de análisis. El diseño sirve para convertir exactamente los requisitos del cliente en un sistema de software. También sirve para facilitar el mantenimiento del sistema.

El diseño es la descripción de la estructura del software, los datos que son parte del sistema y las interfaces entre los componentes del sistema y entre el sistema y los usuarios.

Proceso de diseño

Es un proceso iterativo mediante el cual los requisitos se transforman en modelos para construir el software. Comienza por una representación a un alto nivel de abstracción y a medida que ocurren las iteraciones, la representación va refinándose y teniendo niveles de abstracción más bajos. Es decir, en el nivel más alto de abstracción, la solución se expresa usando lenguaje del entorno del problema y en niveles inferiores se toma una orientación más procedimental. Este refinamiento ocurre para los procedimientos, para los datos y para el control.

Las actividades del proceso de diseño son:

- Se definen los subsistemas que forman el sistema y las relaciones entre ellos (diseño arquitectónico).
- Para cada subsistema se produce una especificación de sus funciones, servicios y restricciones de operación.
- Para cada subsistema se diseña y documenta su interacción con otros subsistemas y con el usuario (diseño de interfaz).
- Se diseña y especifica la estructura de datos que se usará para la implementación del sistema (diseño de datos).
- Se diseña y especifica los algoritmos que realizan las funciones del sistema.

La última etapa puede encontrarse en la etapa de implementación y no en el diseño.

Los resultados producidos por el diseño son:

- Diseño de datos. Transforma el modelo de dominio con las entidades de datos identificadas en los requerimientos, en las estructuras de datos necesarias para implementar el software. El modelo está formado por los objetos de datos, los diagramas Entidad/Relación y el diccionario de datos.
- Diseño arquitectónico. Define la relación entre los elementos estructurales del software, los patrones de diseño que se puede utilizar para lograr los requisitos definidos y las restricciones que afectan al diseño. Se basa en la especificación del sistema y en los modelos del análisis de requerimientos.
- Diseño de interfaz. Describe la manera en que se comunican las distintas partes dentro del software y la comunicación del software con los usuarios. Se basa en las características de control y de datos resultantes de la etapa de análisis.
- Diseño procedimental. Se transforma los elementos estructurales de la arquitectura del software en procedimientos. Se basa en la especificación de control, la especificación de procesos y el diagrama de transición de estados.

Características de un buen diseño.

- Debe implementar todos los requisitos explícitos del modelo de análisis y ajustarse a los implícitos del cliente.
- Debe ser comprensible para aquellos que generan código y aquellos que le dan soporte y mantenimiento al software.
- Debe proporcionar una imagen completa del comportamiento, funciones y datos del sistema, desde un punto de vista de implementación.

Modelo de diseño

El resultado del proceso de diseño es un modelo de diseño, que comprende representaciones de datos, arquitectura, interfaces y componentes. Todos los diferentes aspectos del modelo de diseño se reúnen en una especificación de diseño, la cual incluye:

- Descripción general del trabajo de diseño llevado a cabo.
- Un diseño de datos, que define las bases de datos y las estructuras internas de datos.
- Un diseño arquitectónico, que define la arquitectura del sistema y cómo se deriva del modelo de análisis.
- Un diseño de interfaces, que representa a las interfaces internas (comunicación entre componentes del sistema) y las interfaces externas que se comunican con el usuario.
- Los componentes (subrutinas, funciones y procedimientos) y se describe las funciones que realizan.
- Restricciones de diseño. Llimitaciones de hardware, aspectos importantes de las funciones, etc.

Métodos de diseño

En la mayoría de los proyectos el diseño es informal y se parte del conjunto de requerimientos para codificar inmediatamente. Conforme se codifica, el diseño se actualiza. Dado que los requerimientos cambian con mucha frecuencia y en todo el sistema, al final generalmente se tiene una documentación del diseño incompleta e incorrecta.

Diseño estructurado

Un enfoque de diseño más organizado es el diseño estructurado, el cual implica la construcción de modelos gráficos del sistema y se produce una gran cantidad de documentación. Un método estructurado incluye notaciones para representar el diseño, formatos de informes, reglas y lineamientos de diseño. Los modelos producidos mediante un enfoque estructurado son:

1. Modelo de flujo de datos. El sistema se modela con base en las transformaciones de los datos dentro del sistema.
2. Modelo Entidad-Relación. Describe las entidades más importantes, sus atributos y las relaciones entre ellas. Se usan para describir las estructuras de la base de datos.
3. Modelo estructural. Se documentan los componentes del sistema y sus interacciones.

Diseño arquitectónico

Es la parte inicial del proceso de diseño, en la cual se establece una estructura del sistema, al identificar los componentes principales y las interacciones entre ellos. Las actividades que se realizan durante el proceso de diseño arquitectónico son el diseño de datos y la descomposición del sistema en subsistemas y módulos y las relaciones entre ellos para definir un estilo arquitectónico de construcción.

Diseño de datos

Durante la etapa de análisis se obtiene un modelo de datos que representa los objetos de datos y sus relaciones desde el punto de vista del usuario. Posteriormente, durante el diseño de datos este modelo se representa utilizando diagramas Entidad-Relación para traducirlo a estructuras de datos a nivel de componentes de software y a arquitectura de bases de datos a nivel de aplicación.

Definición de un estilo arquitectónico

Los estilos arquitectónicos principales son:

Arquitecturas centradas en datos

También llamado modelo de depósito. Según este modelo, en el centro se encuentra un almacén de datos (un documento o una base de datos) al que los subsistemas y módulos de la aplicación acceden frecuentemente para añadir, borrar o modificar los datos. Sus características son:

- Es una forma eficiente de compartir grandes cantidades de datos. Un subsistema o módulo no necesita transmitir o recibir los datos de otros componentes del sistema de forma explícita.
- Los subsistemas o módulos que producen datos no necesitan saber cómo los utilizan otros subsistemas o módulos.
- Es posible hacer nuevas aplicaciones, subsistemas o módulos que tengan acceso a los datos para su modificación o uso, siempre y cuando estén acordes a la estructura de los datos.

La figura 4.1 muestra de manera general cómo están dispuestas las partes que forman un sistema con la arquitectura centrada en los datos. Se observa un repositorio central de datos y a distintos sistemas de software interactuando con los datos de este repositorio.

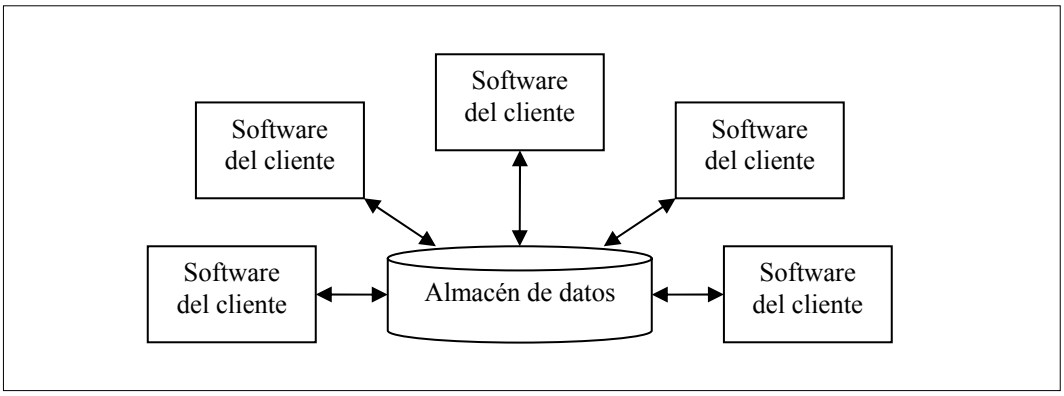


Fig. 4.1 Arquitectura centrada en los datos

Arquitecturas de flujo de datos

Se aplica cuando los datos de entrada son transformados a través de funciones en módulos dentro del sistema. Sus componentes son los filtros, que están diseñados para recibir los datos con un cierto formato y producir otros datos con un formato específico. Los filtros están conectados por tuberías, que definen el curso que siguen los datos dentro del programa. Las transformaciones de los datos se pueden llevar a cabo secuencialmente o en paralelo. La figura 4.2 muestra un ejemplo general de la disposición de los filtros y las tuberías en una arquitectura de flujo de datos.

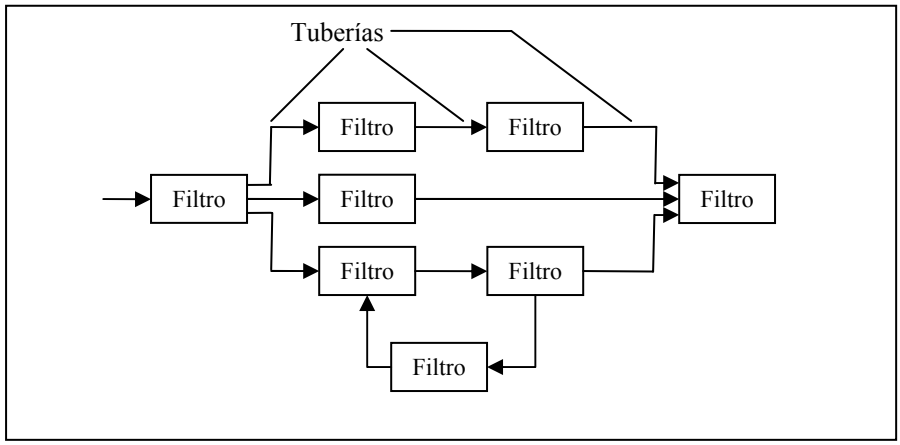


Fig. 4.2 Arquitectura de flujo de datos

Arquitectura de llamada y retorno

Existen dos subestilos dentro de esta categoría:

- Programa principal/subprograma. Se refiere a una jerarquía de control centralizado donde un programa principal llama a un número de componentes o módulos del programa, los cuales también pueden llamar a otros componentes o módulos.

La figura 4.3 muestra un ejemplo de sistema con arquitectura de llamada y retorno. Se observa cómo un programa principal puede llamar a otros componentes del sistema, los cuales a su vez llaman a otros. Cada componente regresa el control al componente que lo llamó, de forma que se acaba por regresar al programa principal.

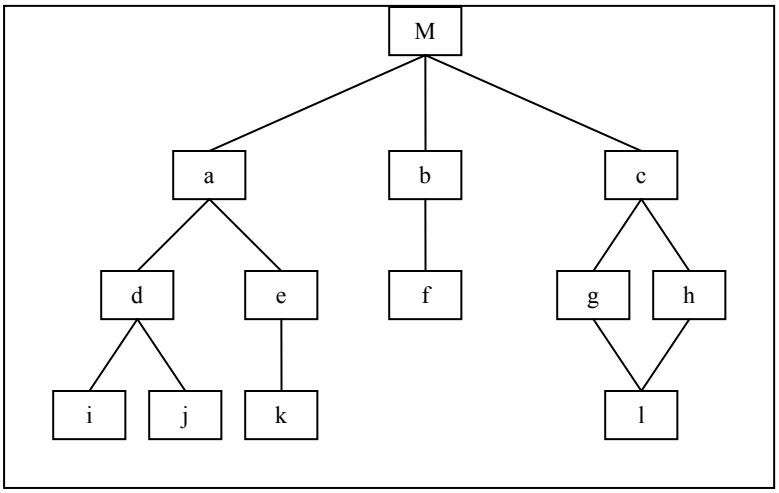


Fig. 4.3 Arquitectura de llamada y retorno

- Arquitectura de llamadas de procedimientos remotos. Los componentes y funciones están distribuidos entre varias computadoras en una red. El programa principal manda llamar un subprograma o función que se encuentra en otra computadora.

Arquitectura orientada a objetos

Los componentes de un sistema juntan bajo el nombre de un objeto los datos y las operaciones que se realizan sobre ellos. La comunicación entre componentes se realiza a través del paso de mensajes.

Arquitectura estratificada o en capas

El sistema está formado por diferentes capas, cada una de las cuales realiza operaciones en distintos niveles de abstracción. Las capas externas realizan las operaciones de interfaz de usuario, las capas intermedias realizan las funciones del sistema de software y las capas internas realizan las funciones de interfaz entre los componentes del sistema.

Arquitectura cliente-servidor

Es un modelo distribuido según el cual los datos y el procesamiento de la información se distribuyen entre varias computadoras en una red. Sus componentes principales son:

- Servidores. Ofrecen servicios de bases de datos, videos, texto, impresión, etc.
- Clientes. Contienen la aplicación de software y llaman a los servicios provistos por las máquinas servidor.
- Una red de comunicaciones entre los servidores y los clientes.

Una vez que el análisis de requisitos define las características y restricciones del sistema, se escoge el estilo o combinación de estilos arquitectónicos que se adaptan mejor a estas características y restricciones. En muchos casos es apropiado más de un estilo arquitectónico.

Después de diseñar una arquitectura estructural, la siguiente etapa del diseño arquitectónico es descomponer los subsistemas en módulos, aquí es donde se pueden aplicar los modelos de las arquitecturas para describir las funciones en los módulos. Se toman los diagramas de flujo de datos, se descomponen los módulos en funciones, subfunciones y éstos en las operaciones que las forman hasta

llegar a un nivel en el cual es posible traducir este diagrama en lenguaje de programación. El resultado es un árbol de operaciones que describe el comportamiento del sistema.

Este árbol de operaciones, una vez desarrollado y refinado, se complementa con:

- Una descripción del procesamiento de cada módulo, un texto que describe el procesamiento, las tareas y los datos de entrada y salida.
- Una descripción de las interfaces internas y externas para cada módulo, así como su interfaz con el usuario.
- Una descripción de las restricciones de diseño de los módulos, que incluyen restricciones de tipo, valores o formato de los datos, limitaciones de memoria, tiempo, etc.

Diseño a nivel de componentes

También llamado diseño procedimental, se refiere a la conversión del modelo de diseño en un software operacional. Existen dos maneras de realizar el diseño procedimental:

1. El programa se crea empleando como guía el modelo de diseño.
2. A partir del modelo de diseño se crea una representación intermedia (gráfica, tabular, basada en texto) que se puede transformar en código fuente.

Las construcciones en el diseño a nivel de componentes son de tres tipos:

- Secuenciales. Son operaciones o pasos de proceso esenciales en la especificación de un algoritmo.
- Condicionales. Proporcionan funciones para procesos seleccionados a partir de una condición lógica.
- Repetitivas. Proporcionan ciclos de repetición de operaciones.

Notaciones gráficas

Diagramas de flujo

Un diagrama de flujo es una forma muy sencilla de representar el diseño. Mediante una caja se representa una operación o paso del proceso, mediante un rombo se representa una condición lógica y las flechas indican el flujo del control. La figura 4.4 muestra los elementos más comunes de este tipo de diagramas:

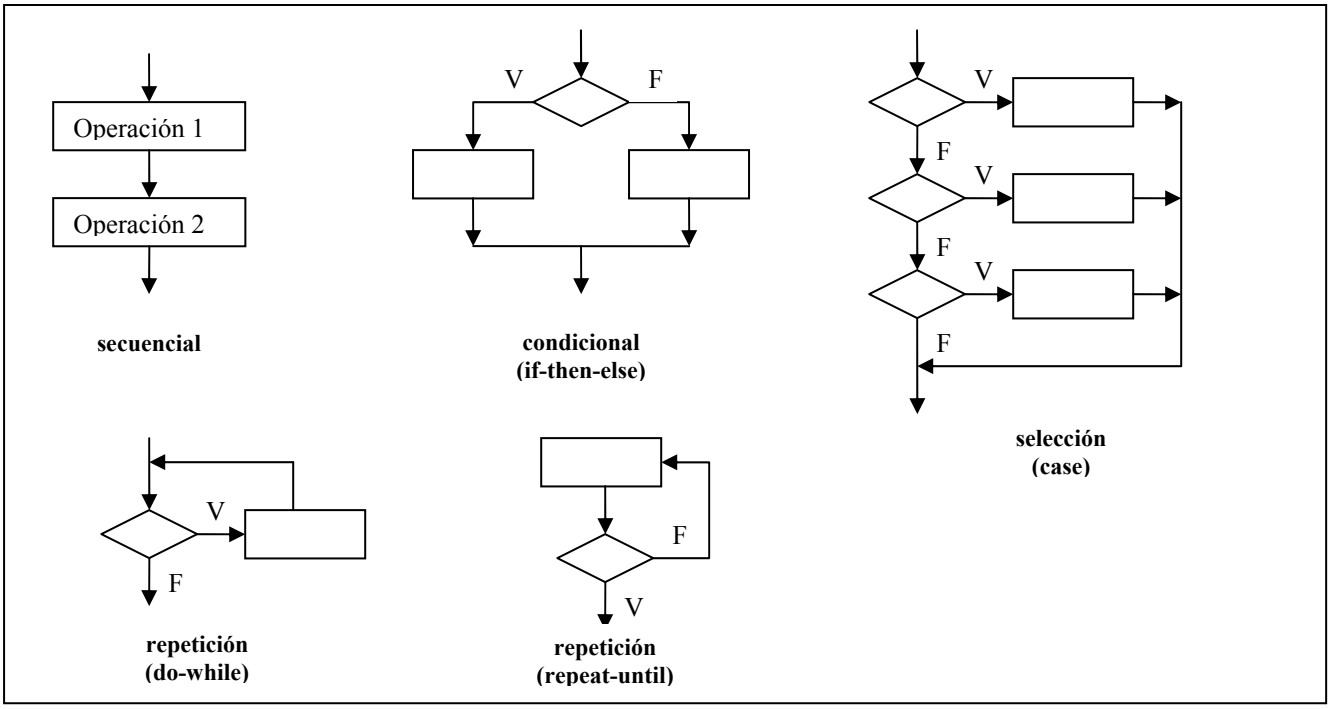


Fig. 4.4 Construcciones en diagramas de flujo

Las construcciones estructuradas pueden anidarse unas en otras.

Diagrama de cajas

Es otra manera sencilla de representar las funciones descritas en el diseño. Los elementos más comunes de este tipo de diagrama se muestran en la figura 4.5:

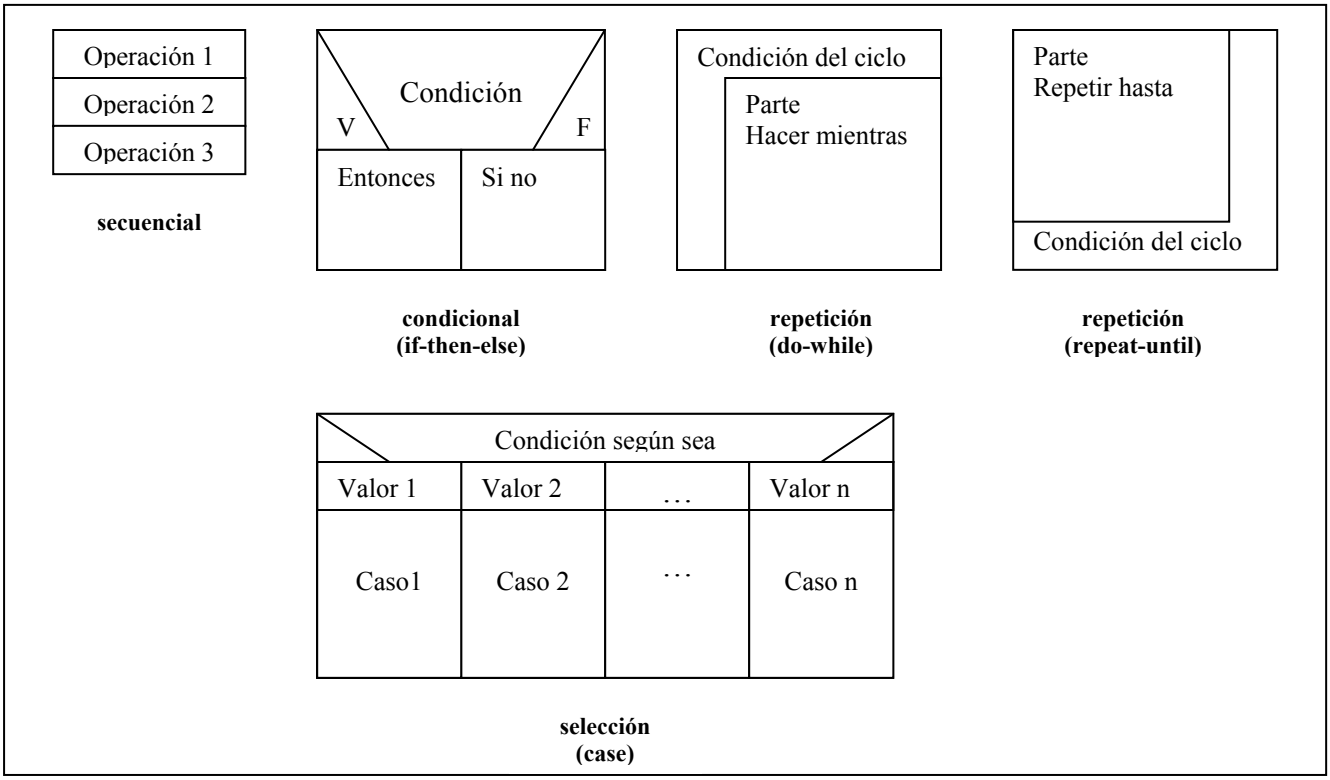


Fig. 4.5 Construcciones en diagrama de cajas

Tablas de decisión

Es una forma de notación que convierte las condiciones y las acciones en una forma tabular. Es una forma de relacionar todas las posibles condiciones con todas las posibles acciones dentro de un programa:

	Reglas				
Condiciones	1	2	3	4	5
Condición 1	V	V	F	F	F
Condición 2	V	F	V	V	F
…					
Condición n	F	V	F	V	
Acciones					
Acción 1	*				
Acción 2		*			
…					
Acción n					*

En el ejemplo anterior, la regla 1 especifica que si las condiciones 1 y 2 son verdaderas, mientras que las demás son falsas, se lleve a cabo la acción 1. De esta manera puede verse que es sencillo representar las funciones del programa mediante tablas de decisión.

Lenguaje de diseño de programas

También conocido como pseudocódigo o lenguaje estructurado. Es una combinación de un lenguaje (español, inglés, etc.) con la sintaxis de un lenguaje estructurado de programación. Utilizando la estructura del lenguaje de programación, se utiliza lenguaje natural para describir las funciones de un módulo. El siguiente es un ejemplo de un pseudocódigo para una estructura condicional if-then-else:

```
IF  condición-1 THEN
        frase-1
        IF condición-2
                frase-2
                frase-3
        ELSE
                frase-4
                frase-5
        END IF
        frase-6
ELSE
        frase-7
        IF condición-3
                frase-8
        END IF
        frase-9
END IF
```

Elección entre los distintos modos de diseño a nivel de componentes

La elección de uno de los modos de diseño a nivel de componentes descritos anteriormente depende en gran medida de las características del diseño arquitectónico. En general se elige el modelo que se acomode a la arquitectura y la refleje y que cumpla con los siguientes aspectos:

- Simplicidad. Deberá proveer una notación fácil de leer, utilizar y aprender.
- Facilidad de modificación. Deberá ser fácil de modificar en caso de cambios en el diseño procedimental.
- Facilidad de mantenimiento. Debe ser fácil, al darle mantenimiento al software, darle mantenimiento al modelo procedimental.
- Representación de datos. Deberá representar sencilla y correctamente los datos globales y locales de cada módulo y del sistema en general.
- Facilidad de codificación. La notación del modelo deberá convertirse fácilmente en código.

Diseño de la interfaz de usuario

Introducción

La interfaz de usuario es el medio de comunicación entre la máquina y el usuario. Para que un sistema tenga éxito es importante contar con un buen diseño de la interfaz de usuario. Un buen diseño de la interfaz de usuario evitará que los usuarios cometan errores, presentará la información al usuario de manera clara, permitirá realizar todas las funciones deseadas por el usuario y será fácil de comprender y usar. La interfaz de usuario es la forma en que el usuario percibe el software y por lo tanto su diseño es muy importante.

El proceso de diseño comienza con la identificación de los requisitos del usuario, las tareas que se realizan y el entorno del programa. Después se crean escenarios de uso para comprender los objetos que se manejan durante las tareas y las acciones que se llevan a cabo sobre ellos. Con base en estos objetos y acciones se crea un prototipo de interfaz de usuario, colocando los iconos, texto, títulos en las ventanas, menús, etc.

Proceso de diseño de la interfaz de usuario

El proceso comienza con la creación de modelos de funcionamiento del sistema, desde el punto de vista del usuario. Se definen las tareas que realiza el usuario y las que realiza el sistema. Por ejemplo, puede generarse escenarios de uso del sistema. Posteriormente se utilizan herramientas de diseño de interfaces para generar prototipos y para su implementación final y por último se hace una evaluación del prototipo de interfaz por el usuario. Como puede verse, el proceso es iterativo al haber cambios o nuevos requerimientos de interfaz en las evaluaciones hechas por el usuario.

"Las cuatro actividades que conforman el proceso de diseño de la interfaz de usuario son:

1. Análisis y modelado de usuarios, tareas y entornos.
2. Diseño de la interfaz.
3. Implementación de la interfaz.
4. Validación de la interfaz."[20]

Análisis

La actividad inicial de análisis consiste en definir a los usuarios qué tendrá el sistema y se hace la recolección de requisitos de los usuarios. En posteriores iteraciones del proceso de diseño, esta etapa consiste en la identificación y descripción de las tareas del usuario y del sistema. También se lleva a cabo la identificación y descripción del entorno físico (condiciones de iluminación, espacio, ruido) en que el usuario utilizará el sistema.

Para realizar un buen estudio de las tareas, primero debe entenderse las tareas que realizan los usuarios sin el sistema (manualmente) y diseñar un conjunto de tareas correspondiente mediante la interfaz de usuario.

Diseño

Durante esta etapa para cada tarea derivada del análisis, se establecen sus objetivos, la secuencia de pasos que contiene, se define los objetos y las acciones que la conforman mediante escenarios de uso y por último se lleva a cabo el diseño gráfico, colocación de los iconos, definición del texto que mostrará la pantalla, títulos para cada ventana, definición de menús y sus elementos, etc.

[20] Pressman, Roger. *Ingeniería del software*. McGraw-Hill, Quinta Edición, España, 2002.

Implementación

Una vez creado el modelo de diseño, se implementa como un prototipo que representa los escenarios de uso del sistema. Para esto hay que tomar en cuenta las diferentes formas en que el usuario puede interactuar con el sistema:

- Manipulación directa. El usuario puede manipular directamente los objetos que se presentan en la pantalla, mediante operaciones sencillas.
- Selección de menús. El usuario selecciona un comando de una lista.
- Llenado de formularios. El usuario llena los campos de un formulario. El formulario puede tener uno o más botones para realizar acciones.
- Lenguaje de comandos. Los usuarios escriben un comando para realizar una operación y los parámetros asociados a ese comando.

El modo de interacción a utilizar depende de la operación que se quiera realizar.

Otro aspecto importante al implementar la interfaz de usuario, es la manera en que se presenta la información de manera visual. La presentación de la información puede ser una representación mediante texto o bien gráficamente. Algunas consideraciones útiles para una mejor presentación de la información son:[21]

- Dependiendo de la aplicación, la información que no cambia puede presentarse en forma de texto o mediante gráficos.
- La información se representa como texto cuando se requiere información numérica precisa y cuando esta información cambia lentamente. Se utiliza una representación gráfica si los datos cambian rápidamente o si las relaciones entre los datos son muy importantes.
- La información numérica que varía dinámicamente se representa mejor de forma gráfica utilizando una representación analógica y se puede complementar con un despliegue digital preciso, si es que este valor preciso necesita conocerse.
- Los despliegues analógicos permiten representar valores relativos. Esto para que el usuario vea fácilmente la posición de un dato dado con respecto a un valor máximo, mínimo o medio.

Patrones de diseño

Un patrón de diseño de interfaz gráfica de usuario se refiere a una disposición específica de los elementos de la interfaz gráfica. Dependiendo de cómo se quiere organizar la información que se muestra en la pantalla, de cómo se quiera manipular esta información, de la manera en que se quiera navegar por la información y las operaciones disponibles y de la manera en que se desea manipular cada pantalla como un todo, los patrones más comunes existentes son:

Vista General – Vista detallada

Este patrón se utiliza cuando se quiere mostrar una gran cantidad de información tanto en una vista de estructura general de la información como en una vista detallada de una sección particular de ésta. Ambas vistas, la general y la detallada se mantienen en la misma pantalla y en ambas vistas puede uno navegar a través de la información. La figura 4.6 muestra un ejemplo de Matlab:

[21] Sommerville, Ian. *Ingeniería de software*. Addison-Wesley, Sexta Edición, Reino Unido, 2002.

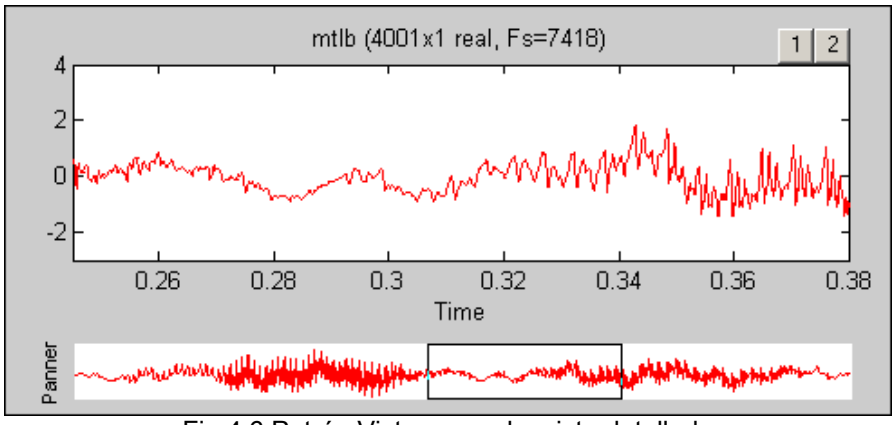


Fig 4.6 Patrón Vista general – vista detallada

Secuencia de acciones no ligadas

Este patrón se utiliza cuando se presenta una pantalla central a partir de la cual se puede navegar hacia otras pantallas en una secuencia definida de pantallas, pero no se desea ligar todas las pantallas entre sí, siendo necesario al terminar una secuencia de acciones, regresar a la pantalla principal. Un ejemplo es una página web que consiste en una página principal, la cual tiene ligas a otras páginas, pero las páginas ligadas sólo tienen una liga de regreso a la página principal, siendo necesario regresar a la página principal para tener acceso a las otras páginas ligadas.

Funcionalidades extras

Este patrón se utiliza cuando existe una gran cantidad y variedad de opciones, operaciones y funcionalidades posibles, pero no se quiere o no se necesita tener a todas ellas presentes en la pantalla, sino tener acceso a ellas con un solo clic. La figura 4.7 muestra un ejemplo de Windows Explorer:

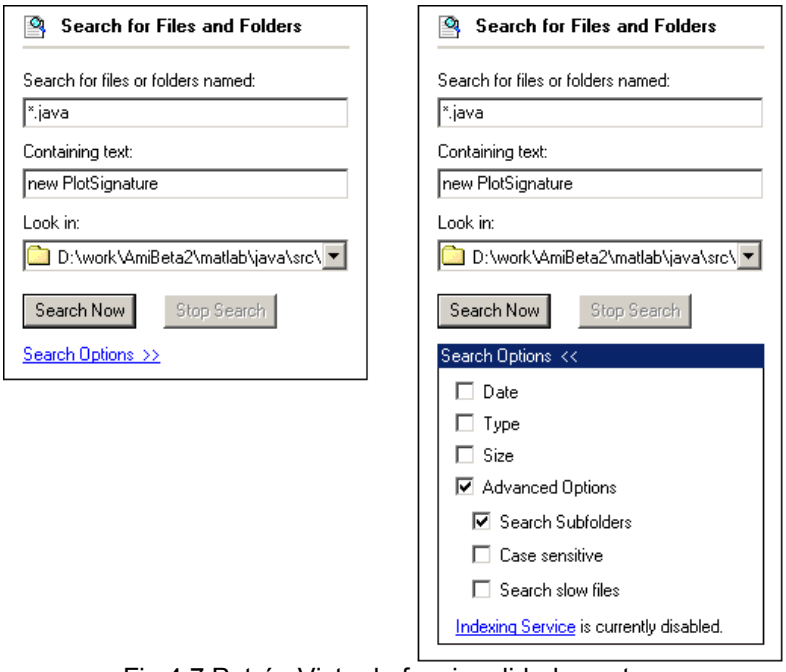


Fig 4.7 Patrón Vista de funcionalidades extras

Paso a paso (wizard)

Este patrón se utiliza cuando el usuario va a realizar una operación que es larga o complicada y que tiene varias posibles rutas de acción. Esta operación se realiza a través de pasos durante los cuales el usuario elige opciones las cuales definen la ruta de acción que la operación sigue. La figura 4.8 muestra un ejemplo de Instalación de Netscape:

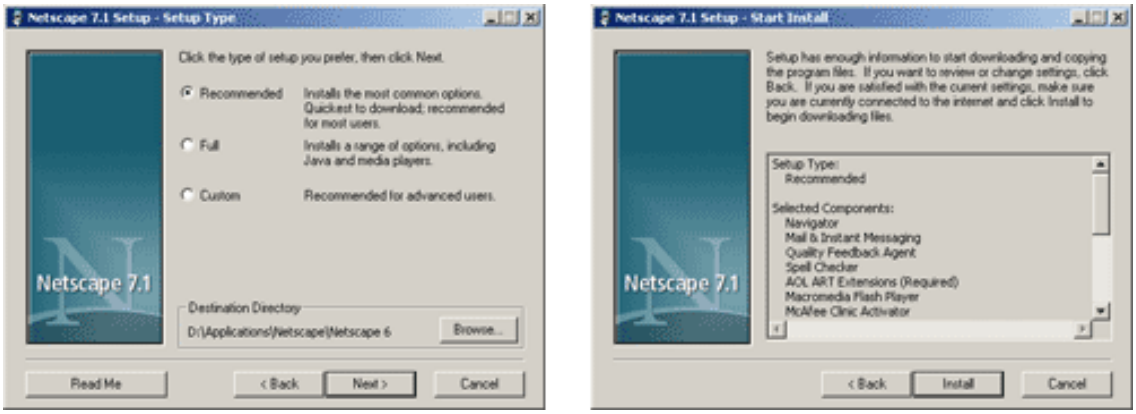


Fig 4.8 Patrón Wizard

Ligas interesantes

Este patrón de diseño se utiliza cuando se presenta información en la pantalla y además se quiere presentar al usuario información interesante relacionada pero que no es parte del tema principal de la información en la pantalla actual, lo cual se hace mediante ligas a esta información relacionada. La figura 4.9 muestra un ejemplo:

Fig 4.9 Patrón Ligas interesantes

Navegación de alto nivel

Este patrón se utiliza cuando la aplicación está dividida en partes principales de gran tamaño y se desea facilitar la navegación entre estas partes. La figura 4.10 muestra un ejemplo:

Fig 4.10 Patrón Navegación de alto nivel

División mediante códigos de color

Se utiliza este patrón cuando se quiere hacer notar los distintos tipos de información o las distintas partes de una aplicación, mediante distintos colores.

Esquema visual

Un esquema visual se refiere a una manera consistente de presentación del texto, iconos, dibujos, colores y la disposición de los elementos en todas las pantallas de una aplicación. Se utiliza un esquema para hacer más comprensible y fácil de usar una aplicación. La figura 4.11 muestra un ejemplo de www.macromedia.com:

Fig 4.11 Patrón Esquema visual

Secciones tituladas

Este patrón se utiliza cuando se tiene una gran cantidad de información u operaciones y se quiere hacer una división o agrupación en varias secciones, para hacer más fácil la comprensión al usuario. La figura 4.12 muestra un ejemplo de www.adobe.com:

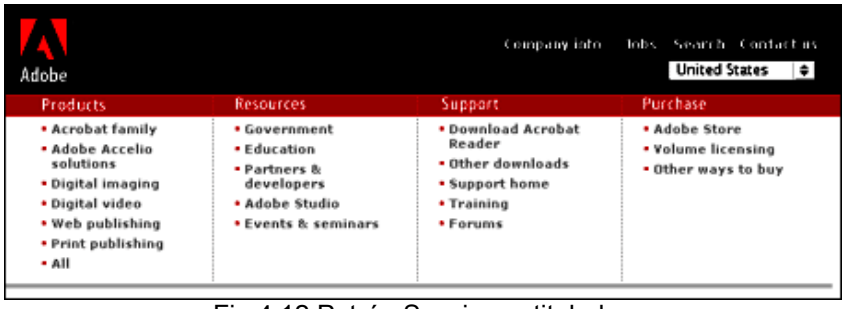


Fig 4.12 Patrón Secciones tituladas

División con tabuladores

Este patrón es otra manera de dividir la información de una pantalla o las operaciones posibles, en varias secciones, haciendo más fácil la navegación y la comprensión. La figura 4.13 muestra un ejemplo de propiedades del explorador de internet de windows:

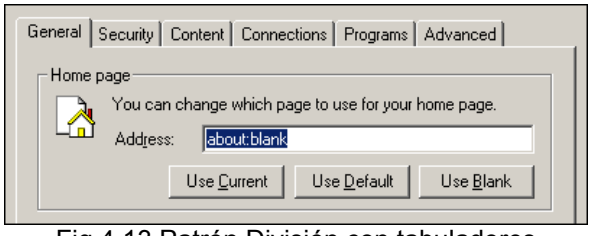


Fig 4.13 Patrón División con tabuladores

Paneles

Este patrón, al igual que el de tabuladores, permite organizar la información en claras divisiones, permitiendo mostrar solamente una parte de ella y teniendo la demás información al alcance con un solo clic. La figura 4.14 muestra un ejemplo de Dreamweaver:

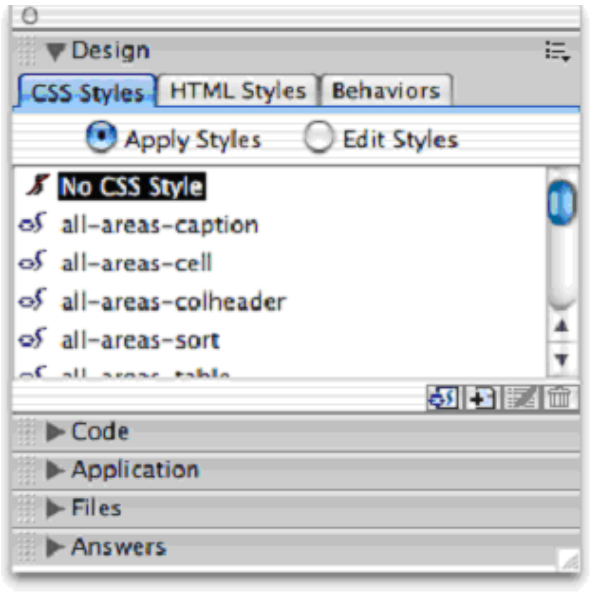


Fig 4.14 Patrón Paneles

Especificación progresiva

Este patrón permite al usuario realizar un conjunto de operaciones, paso a paso, pero sin cambiar de pantalla como en un patrón paso a paso. La figura 4.15 muestra un ejemplo de ATM:

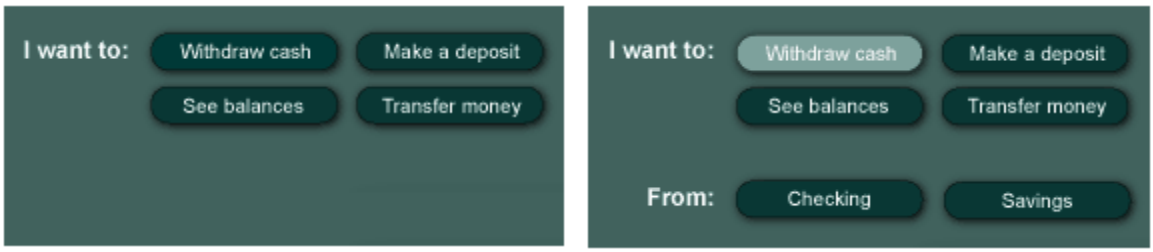


Fig 4.15 Patrón Especificación progresiva

Multiformato

Este patrón se utiliza cuando en un campo a llenar por el usuario está permitido introducir datos de distintos formatos o significados, pudiendo la aplicación distinguir entre las distintas posibles ocurrencias del dato. La figura 4.16 muestra un ejemplo de wunderground.com:

Fig 4.16 Patrón Multiformato

Espacios en blanco para llenar

Se utiliza este formato cuando se requiere que el usuario complete cierta información necesaria para realizar una operación. La manera en que el usuario introduce la información asemeja completar una oración. La figura 4.17 muestra algunos ejemplos de Amazon:

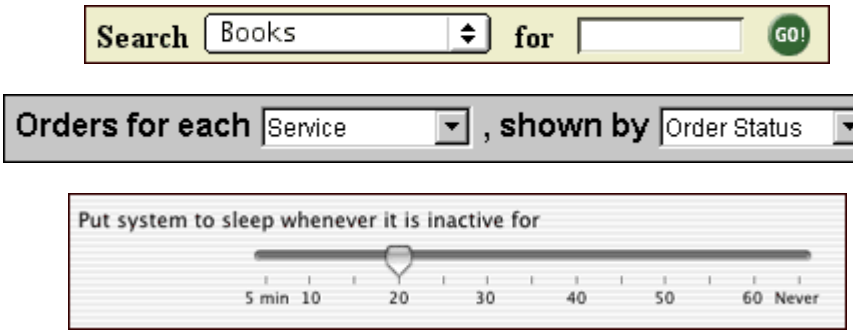


Fig 4.17 Patrón Espacios en blanco

Ejemplos de datos de entrada

Se utiliza cuando el usuario debe introducir un dato en un campo, pero el tipo de información requerida o su formato no son obvios y se muestra un ejemplo. La figura 4.18 muestra un ejemplo de MS Word:

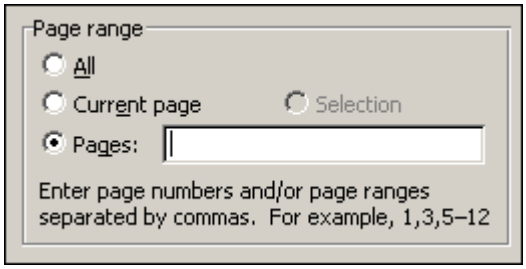


Fig 4.18 Patrón Ejemplos de datos de entrada

Menú en combo

Este patrón se utiliza cuando el usuario introduce un dato o escoge una opción que se encuentra en un menú desplegable. La figura 4.19 muestra ejemplos de MS Word y del calendario de orbitz.com:

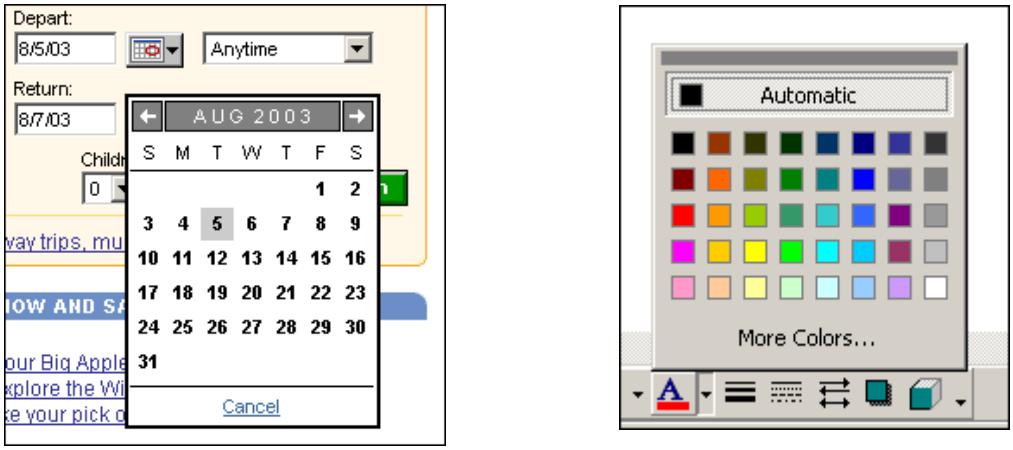


Fig 4.19 Patrón Ejemplos de datos de entrada

Opciones ilustradas

Este patrón consiste en acompañar una lista de opciones a elegir por el usuario con una imagen que ejemplifica cada opción, como una forma de prever la opción a elegir. La figura 4.20 muestra ejemplos de MS Word y de MS Excel:

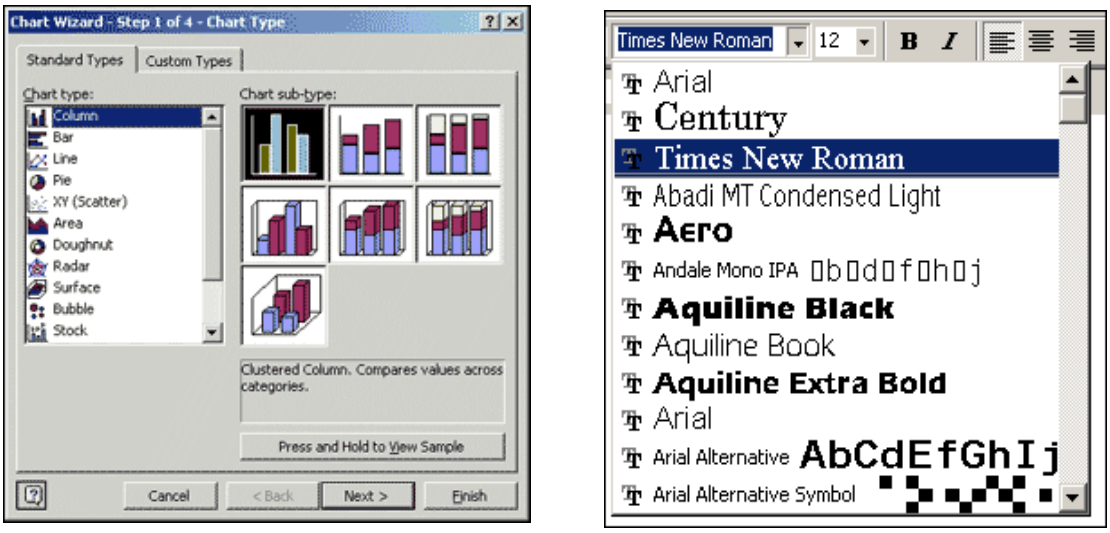


Fig 4.20 Patrón Opciones ilustradas

Tablas con datos ordenables por columna

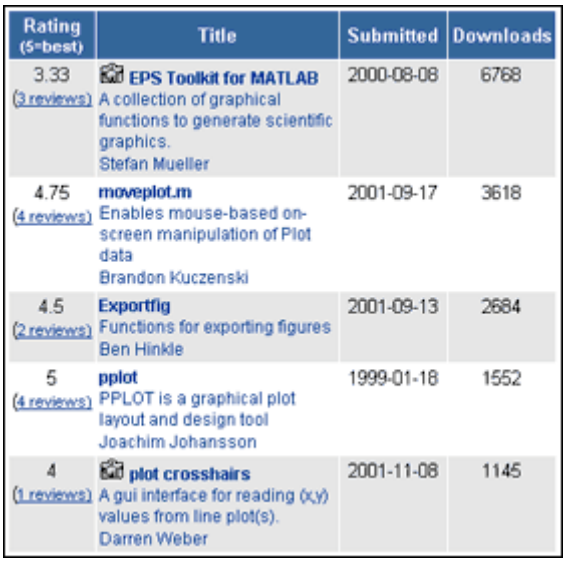
Este patrón consiste en tablas de datos, los cuales pueden ser ordenados por columna. La figura 4.21 muestra un ejemplo de explorador de Windows, en el que los datos se pueden ordenar por nombre, tamaño, tipo o fecha de modificación:

Name	Size	Type	Modified
demo		File Folder	8/12/2001 8:38 PM
doc		File Folder	8/12/2001 8:38 PM
frameworks		File Folder	8/12/2001 8:38 PM
javadoc		File Folder	8/12/2001 8:38 PM
lib		File Folder	8/12/2001 8:38 PM
index.html	1 KB	HTML Document	5/1/2001 12:03 PM
license.html	14 KB	HTML Document	5/1/2001 12:04 PM
release_notes.html	1 KB	HTML Document	5/1/2001 12:03 PM
rn_connect.html	1 KB	HTML Document	5/1/2001 12:03 PM
rn_dev.html	25 KB	HTML Document	5/2/2001 4:53 PM
rn_sync.html	1 KB	HTML Document	5/1/2001 12:03 PM

Fig 4.21 Patrón Tablas con datos ordenables por columna

Filas con colores alternados

Este patrón hace más fácil la distinción entre las líneas de una tabla de datos, aplicando colores de forma alternada a las líneas. La figura 4.22 muestra un ejemplo de mathworks.com:



Rating (5=best)	Title	Submitted	Downloads
3.33 (3 reviews)	**EPS Toolkit for MATLAB** A collection of graphical functions to generate scientific graphics. Stefan Mueller	2000-08-08	6768
4.75 (4 reviews)	**moveplot.m** Enables mouse-based on-screen manipulation of Plot data Brandon Kuczenski	2001-09-17	3618
4.5 (2 reviews)	**Exportfig** Functions for exporting figures Ben Hinkle	2001-09-13	2684
5 (4 reviews)	**pplot** PPLOT is a graphical plot layout and design tool Joachim Johansson	1999-01-18	1552
4 (1 reviews)	**plot crosshairs** A gui interface for reading (x,y) values from line plot(s). Darren Weber	2001-11-08	1145

Fig 4.22 Patrón Filas con colores alternados

Listas en cascada

Este patrón consiste en una sucesión de listas de datos relacionadas entre sí, las cuales en forma jerárquica muestran la información dependiendo del valor escogido en la lista anterior. La figura 4.23 muestra un ejemplo de Mac OS/X:

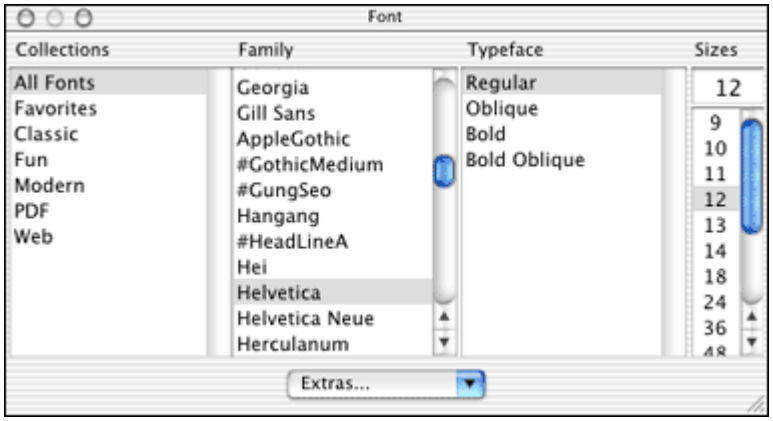


Fig 4.23 Patrón Listas en cascada

Saltar a objeto

Este patrón, utilizado generalmente en buscadores en listas o en combos, sirve para encontrar rápidamente el objeto buscado. Mediante la introducción por el teclado de uno o varios caracteres de la palabra buscada, el objeto seleccionado en la lista o combo es el que más se parece a lo introducido por el teclado. La figura 4.24 muestra un ejemplo de Mac OS/X Finder:

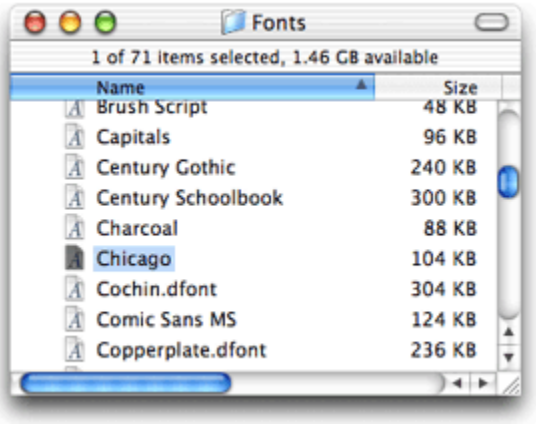


Fig 4.24 Patrón Saltar a objeto

Deshacer multinivel

Este patrón consiste en que la aplicación almacene las últimas acciones realizadas y permita deshacer una o varias de ellas en sucesión. La figura 4.25 muestra un ejemplo de Photoshop 7:

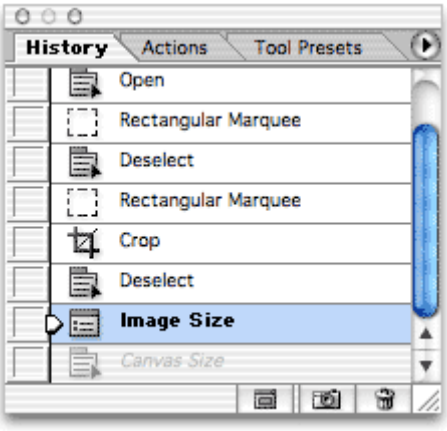


Fig 4.25 Patrón Deshacer multinivel

Menú inteligente

Este patrón consiste en que los menús contienen un filtro para que en sus opciones aparezcan solamente las acciones relacionadas con el objeto actualmente seleccionado o aquél con el que actualmente se está trabajando. La figura 4.26 muestra un ejemplo de Illustrator 10:

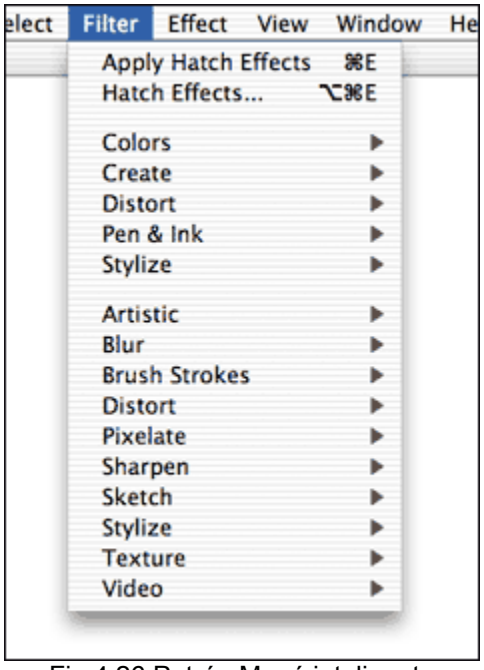


Fig 4.26 Patrón Menú inteligente

Indicadores de progreso

Este patrón se utiliza cuando la aplicación realiza una acción que necesita un periodo largo de tiempo y se desea mostrar al usuario el avance de la acción. La figura 4.27 muestra un ejemplo de explorador de internet:

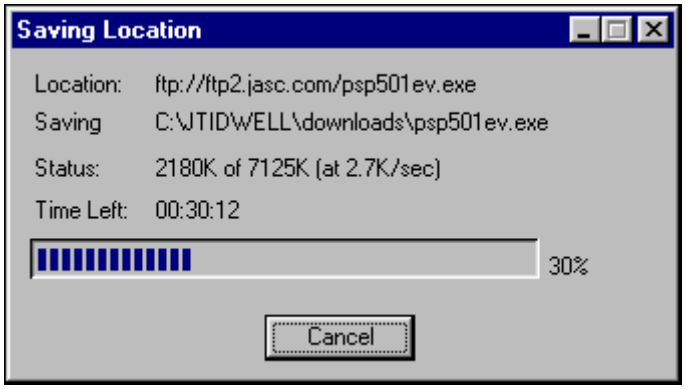


Fig 4.27 Patrón Indicadores de progreso

Historial de acciones

Este patrón se utiliza para almacenar en el sistema una serie de acciones realizadas por el usuario, de manera que éste pueda consultar las operaciones que ha hecho sin necesidad de recordarlas. La figura 4.28 muestra un ejemplo de Photoshop 6:

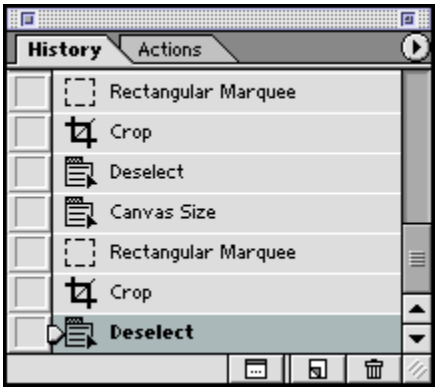


Fig 4.28 Patrón Historial de acciones

Selección inteligente

Este patrón se utiliza cuando el usuario trabaja con objetos seleccionables (texto, imágenes, etc.) y todos estos objetos se encuentran en grupos visuales coherentes, lo cual permite seleccionar todo o una parte de esos objetos. La figura 4.29 muestra un ejemplo de PDF Viewer:

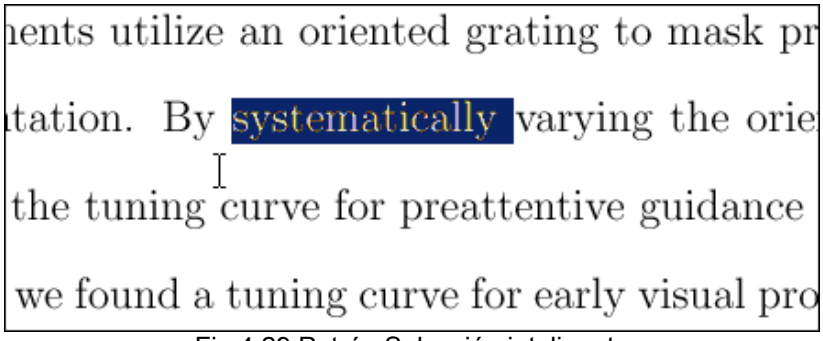


Fig 4.29 Patrón Selección inteligente

Para llevar a cabo la creación de los prototipos utilizando los patrones anteriores, existen herramientas de diseño de interfaz, que proporcionan los componentes para la creación de ventanas, menús, botones, despliegue de texto o gráficos, mensajes de error, órdenes, etc.

Validación

La evaluación que el usuario hace del prototipo de interfaz consiste en analizar la forma en que se utiliza una interfaz y verificar que cumple sus requerimientos. Los principales aspectos a tomar en cuenta para una interfaz que parezca aceptable al usuario son:

- La interfaz debe ser fácil de aprender, comprender y utilizar por el usuario.
- La interfaz debe utilizar términos familiares a los usuarios y los objetos que presenta deben estar relacionados con las actividades que el usuario realiza y con su entorno de trabajo.

- La interfaz debe presentar consistencia, esto significa que los mismos componentes dentro del sistema, como los menús, mensajes, botones, etc. deben presentar la misma apariencia (formato, color, tamaño) en todo el sistema. Esto hace que el sistema sea fácil de aprender.
- El sistema debe presentar una velocidad de respuesta aceptable.
- El sistema debe presentar tolerancia y recuperación a los errores del usuario. Para esto el sistema puede proveer medios para deshacer las acciones del usuario que provocan errores y también puede solicitar la confirmación del usuario antes de realizar una acción destructiva.

Capítulo 5
Bases de datos

Introducción

Un sistema de base de datos es un sistema computarizado para llevar registros. Una base de datos es un conjunto de archivos de datos, los cuales son creados, utilizados, modificados y administrados por computadora[22]. Las operaciones que se puede hacer sobre estos archivos son:

- Agregar nuevos archivos a la base de datos.
- Eliminar archivos de la base de datos.
- Insertar datos en los archivos.
- Consultar datos de los archivos.
- Modificar datos de los archivos.
- Eliminar datos de los archivos.

Ejemplo de un archivo:

NumeroCuenta	Nombre	ApellidoPaterno	ApellidoMaterno	FechaNacimiento	Carrera
97582344	Gustavo	Herrera	Romero	13/10/79	Computación
97324856	Francisco	Muller	Sánchez	23/06/77	Electrónica
97632514	Ricardo	Martínez	Pérez	12/07/78	Arquitectura
95123658	Eduardo	Rodríguez	Morales	30/08/80	Psicología

A los archivos computarizados se les llama tablas o relaciones, cuyos renglones o filas son los registros del archivo y las columnas son los campos de dichos registros. Los campos tienen un tipo de dato que define los valores que pueden tener dichos campos (numérico, entero, carácter, cadena, fecha, etc.).

Cada tabla tiene un conjunto de campos que identifica como único a cada registro, llamado llave primaria. En el caso de la tabla Alumno, su llave primaria es el número de cuenta.

Las operaciones sobre una base de datos se expresan en un lenguaje estándar de consultas llamado SQL (Standard Query Language). Ejemplos de instrucciones en este lenguaje y los resultados de estas instrucciones son:

a) Consulta:

```
SELECT NumeroCuenta,Nombre,ApellidoPaterno,ApellidoMaterno
  FROM Alumno
WHERE Carrera = “Computación”
```

Resultado:

NumeroCuenta	Nombre	ApellidoPaterno	ApellidoMaterno
97582344	Gustavo	Herrera	Romero

b) Inserción:

```
INSERT INTO Alumno
VALUES(“96321547”,”Rodrigo”,”Diaz”,”Ruiz”,”25/02/79”,”Computacion”)
```

c) Modificación:

```
UPDATE Alumno
      SET Carrera = “Medicina”
 WHERE NumeroCuenta = “95123658”
```

d) Eliminación:

```
DELETE Alumno
WHERE NumeroCuenta = “95123658”
```

[22] Date, C.J. *Introducción a los sistemas de bases de datos*. Prentice Hall, Séptima edición, México, 2001.

Componentes principales de un sistema de base de datos

En un sistema de base de datos los usuarios pueden consultar y modificar la información con base en peticiones formuladas al manejador de bases de datos.

Los componentes principales de un sistema de base de datos son:

Datos

Representan la información importante para los usuarios de la base de datos.

Los archivos o tablas de la base de datos tendrán la información con la cual trabajan los usuarios. La información en los archivos o tablas está relacionada entre sí.

Los datos estarán compartidos en el caso de que puedan ser consultados y/o modificados por diferentes usuarios.

Hardware

Sus componentes son:

- Componentes de almacenamiento secundario. Es donde se almacena la información de manera duradera o persistente. Se refiere principalmente a los discos magnéticos (disco duro), junto con sus dispositivos de lectura y escritura y los canales de comunicación física con estos dispositivos.

- Procesadores de instrucciones y memoria usada para carga y ejecución del software manejador del sistema de bases de datos.

Software

Es la capa de aplicaciones o programas que sirve de interfaz entre la base de datos física (donde los datos están almacenados físicamente) y los usuarios del sistema. Se le conoce como Sistema de Administración de Bases de Datos (DBMS) o servidor o administrador de bases de datos. Maneja todas las peticiones de acceso, consulta y modificación de los datos.

Usuarios

Hay tres clases de usuarios:

- Programadores de aplicaciones. Utilizando un lenguaje de programación crean aplicaciones que mediante peticiones al DBMS (generalmente en la forma de instrucciones SQL) tienen acceso y control de los datos de una base de datos.

- Usuarios finales. Interactúan con el sistema a través de las aplicaciones creadas por los programadores o mediante una aplicación o interfaz procesadora de peticiones o instrucciones en lenguaje de consulta (SQL) que esté integrada en el software del DBMS.

- DA y DBA. El administrador de datos (DA) es la persona dentro de la organización que usa el sistema de base de datos, que entiende el contenido y la importancia de los datos, así como las necesidades que serán satisfechas con su uso. Entiende los datos desde un punto de vista de administración. Es la persona que define qué datos son importantes y cómo deben ser manejados. El administrador de bases de datos (DBA) es el técnico encargado de implementar las decisiones del DA, crea la base de datos física e implementa las operaciones necesarias para cumplir con las políticas de manejo de los datos definidas por el DA.

Ventajas de usar una base de datos

Desde el punto de vista del trabajo que cuesta manejar una base de datos en comparación con tener la información en la forma tradicional sobre papel, las ventajas son:

- Compactación. Se elimina la necesidad de archivos voluminosos en papel.
- Velocidad. Las consultas y actualizaciones de datos hechas por una máquina son mucho más rápidas que las búsquedas visuales o actualizaciones manuales.
- Actualidad. Se puede disponer de información precisa y actualizada en cualquier momento.
- Menos trabajo. Además de la rapidez para operar sobre los datos, todas estas acciones repetitivas de consulta y modificación los hace la máquina en lugar de una persona.

Desde el punto de vista de la información, las ventajas son:

- Es posible reducir la cantidad de datos repetidos, al integrar en un solo lugar todos los archivos separados que las distintas aplicaciones que forman el sistema pueden tener por su propia cuenta.
- Lo anterior permite también reducir la inconsistencia (incongruencia en la información) que ocurre cuando la información está separada en varios lugares y una actualización que se hace en un lugar no se hace en los demás lugares donde la misma información exista. Mediante un enfoque de base de datos es más fácil y rápido actualizar toda la información involucrada en una actualización, en las distintas localidades que esta tenga en la base de datos.
- Se puede manejar transacciones. Una transacción es un conjunto de operaciones que se encuentran agrupadas bajo el nombre de una sola operación lógica. Si durante la ejecución de las operaciones de una transacción ocurre una falla en el sistema y la transacción no termina de ejecutarse, se deshacen todos los cambios causados por la transacción. Esto asegura que o bien se llevan a cabo todas las operaciones o no se lleva ninguna. Esto evita pérdida de información o datos inconsistentes al ocurrir fallas en el sistema.
- Es posible mantener la integridad. La integridad se refiere a la corrección en los valores de los datos. Se puede crear restricciones en los valores de los campos de una tabla (por ejemplo límites en cantidades, límites en la longitud de una cadena de caracteres, etc.) con el fin de asegurar que la información sea correcta y de acuerdo a las reglas específicas de la organización que usa la base de datos.
- Se puede tener un control de los usuarios que pueden tener acceso a la información y las operaciones que cada usuario puede realizar.

En el caso de sistemas multiusuario, existe la ventaja adicional de contar con una información centralizada, a la cual puede tener acceso una gran cantidad de usuarios al mismo tiempo.

Mientras más grande y más compleja sea la base de datos, más se notarán estas ventajas.

Desventajas de una base de datos

Las desventajas de una base de datos se refieren principalmente al costo que ocasiona contar con una base de datos:

- Costo del software administrador de la base de datos.
- Costo de capacitación del personal usuario del sistema.
- Costo del hardware para instalación del sistema.

Sistema de administración de bases de datos (DBMS)

Es el software que maneja todo acceso a la base de datos. Un acceso sucede de la siguiente manera:

1. Un usuario emite una petición de acceso mediante un sublenguaje de datos.
2. El DBMS recibe la petición y la analiza.
3. El DBMS revisa la estructura de los datos.
4. El DBMS ejecuta las operaciones sobre la base de datos.

Las funciones principales del DBMS son:

- Definición de datos: el DBMS toma las definiciones conceptuales de los datos hechas por el usuario y las transforma en los correspondientes objetos de datos y las relaciones entre ellos.

- Manipulación de datos: el DBMS procesa y ejecuta las peticiones de consulta, actualización o eliminación de los datos.

Un sistema de bases de datos como arquitectura cliente/servidor

Desde un punto de vista más externo, un sistema de bases de datos puede verse como un sistema cliente/servidor:

- Servidor o servicios de fondo. Es el software del DBMS. Soporta las funciones de definición y manipulación de los datos.

- Clientes, interfaces o aplicaciones de usuario. Aplicaciones desarrolladas por programadores en algún lenguaje de programación o proporcionados por el fabricante del DBMS, los cuales operan sobre el DBMS y sirven como apoyo en la creación y ejecución de aplicaciones de bases de datos.

Entre las aplicaciones o herramientas proporcionadas por el fabricante están:
- Procesadores de lenguaje de consulta.
- Generadores de informes.
- Hojas de cálculo.
- Herramientas para administración de datos.

Las aplicaciones clientes y servidores pueden ser operadas en máquinas diferentes. Esto se llama procesamiento distribuido y significa que es posible conectar varias máquinas en una red y dividir una sola tarea de procesamiento de datos entre dos o más máquinas. Las figuras 5.1 a 5.3 muestran las distintas variedades de procesamiento distribuido:

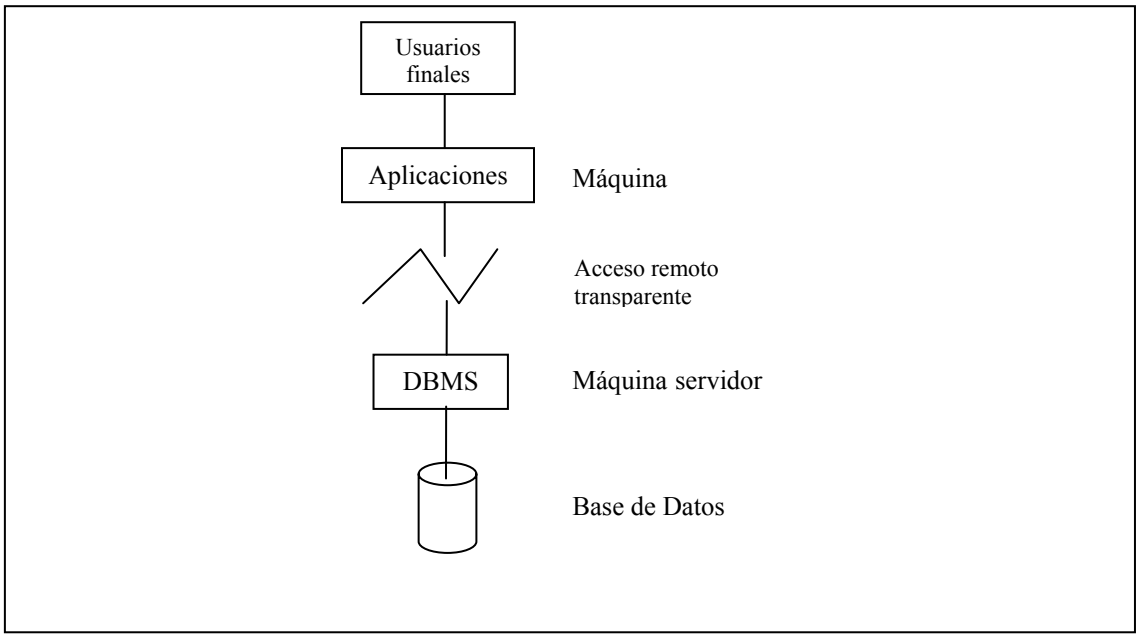


Figura 5.1 Cliente y servidor operando en máquinas diferentes

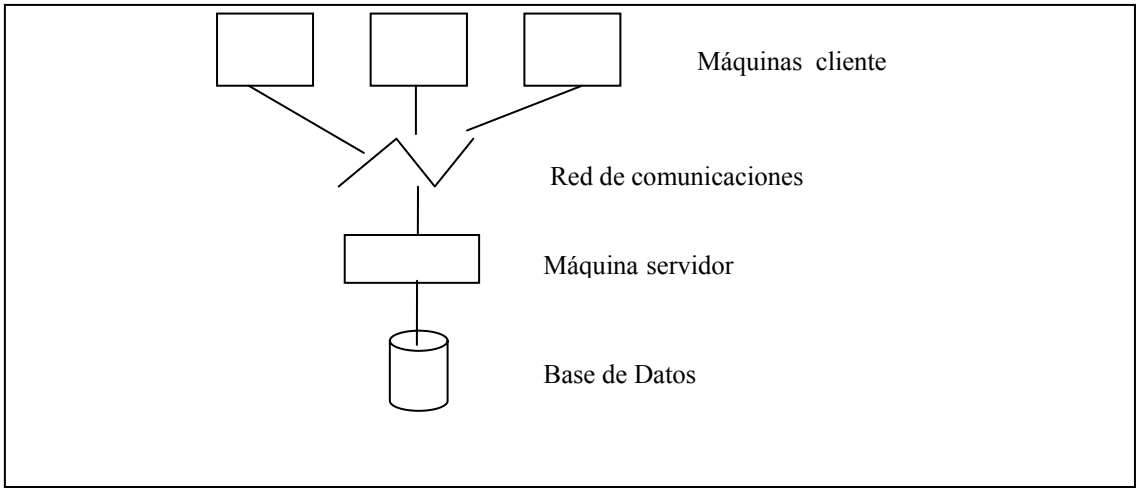


Figura 5.2 Una máquina servidor, varias máquinas cliente

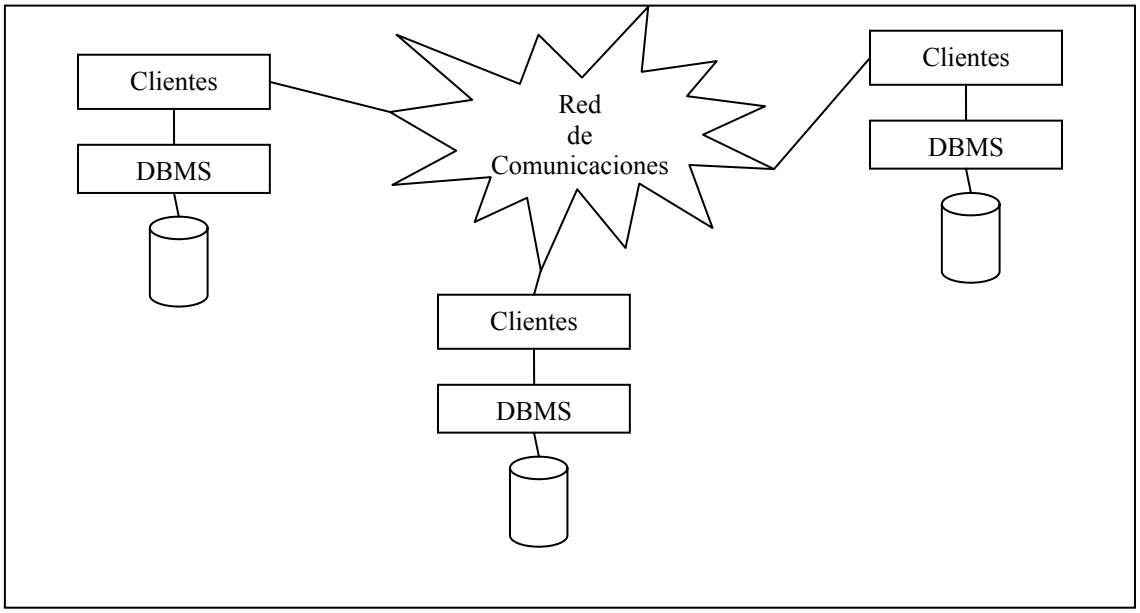


Figura 5.3 Cada máquina opera como cliente y como servidor

Ventajas del procesamiento distribuido

- Procesamiento paralelo: el proceso de la aplicación del usuario y del servidor de bases de datos se realizan en paralelo, lo cual significa un menor tiempo de respuesta de los datos y la aplicación de usuario en conjunto.
- Se puede tener máquinas diseñadas específicamente para un buen servicio de bases de datos, así como máquinas diseñadas específicamente para las aplicaciones de usuario. Esto también mejora la eficiencia.
- Varias aplicaciones de usuario pueden acceder a la misma máquina servidor. Con esto los datos se encuentran centralizados y son compartidos.
- En el caso de las distintas sucursales de un establecimiento y cada sucursal con su servidor y sus datos particulares, es posible conectar estos servidores en red y así es posible contar con toda la información de todas las sucursales.

Bases de datos relacionales

Introducción

Los tres aspectos del modelo relacional de datos son:

1. Estructural. El usuario percibe la información de la base de datos solamente como tablas.
2. Integridad. Las tablas satisfacen ciertas restricciones de integridad, las cuales definen el tipo de datos que pueden tener sus columnas y los valores válidos dentro del tipo de datos).
3. Manipulación. Los operadores disponibles para la manipulación de las tablas (consulta, actualización, eliminación, etc.) son operadores cuyo resultado son tablas.

Los tres operadores principales son:

- Restringir o seleccionar. Extrae las filas especificadas de una tabla.
- Proyectar. Extrae las columnas especificadas de una tabla.
- Juntar. Reúne dos o más tablas con base en valores comunes de sus columnas comunes.

Ejemplo:

Tabla DEPTO

DEPTO#	NOMDEPTO	PRESUPUESTO
D1	Comercialización	10000
D2	Desarrollo	12000
D3	Investigación	5000

Tabla EMP

EMP#	NOMEMP	DEPTO#	SALARIO
E1	Pedro	D1	40000
E2	Ricardo	D1	45000
E3	Arturo	D2	30000
E4	Mario	D2	35000

a) Restringir.
DEPTOs donde PRESUPUESTO > 8000
Resultado:

DEPTO#	NOMDEPTO	PRESUPUESTO
D1	Comercialización	10000
D2	Desarrollo	12000

b) Proyectar
DEPTO# y PRESUPUESTO de DEPTO

DEPTO#	PRESUPUESTO
D1	10000
D2	12000
D3	5000

c) Juntar
DEPTOs y EMPs con base en DEPTO#

DEPTO#	NOMDEPTO	PRESUPUESTO	EMP#	NOMEMP	SALARIO
D1	Comercialización	10000	E1	Pedro	40000
D1	Comercialización	10000	E2	Ricardo	45000
D2	Desarrollo	12000	E3	Arturo	30000
D2	Desarrollo	12000	E4	Mario	35000

Las bases de datos relacionales cumplen con el *principio de información*, que establece que toda la información de una base de datos relacional está expresada como valores explícitos posicionados en columnas, las cuales se encuentran dentro de filas en las tablas.

Cada tabla cuenta con un encabezado, el cual define su estructura mediante parejas nombre_columna:tipo_datos; también cuenta con un cuerpo, el cual se compone de filas que se apegan a la definición del encabezado.

Terminología estructural

La figura 5.4 muestra los términos más importantes que se refieren a una relación:

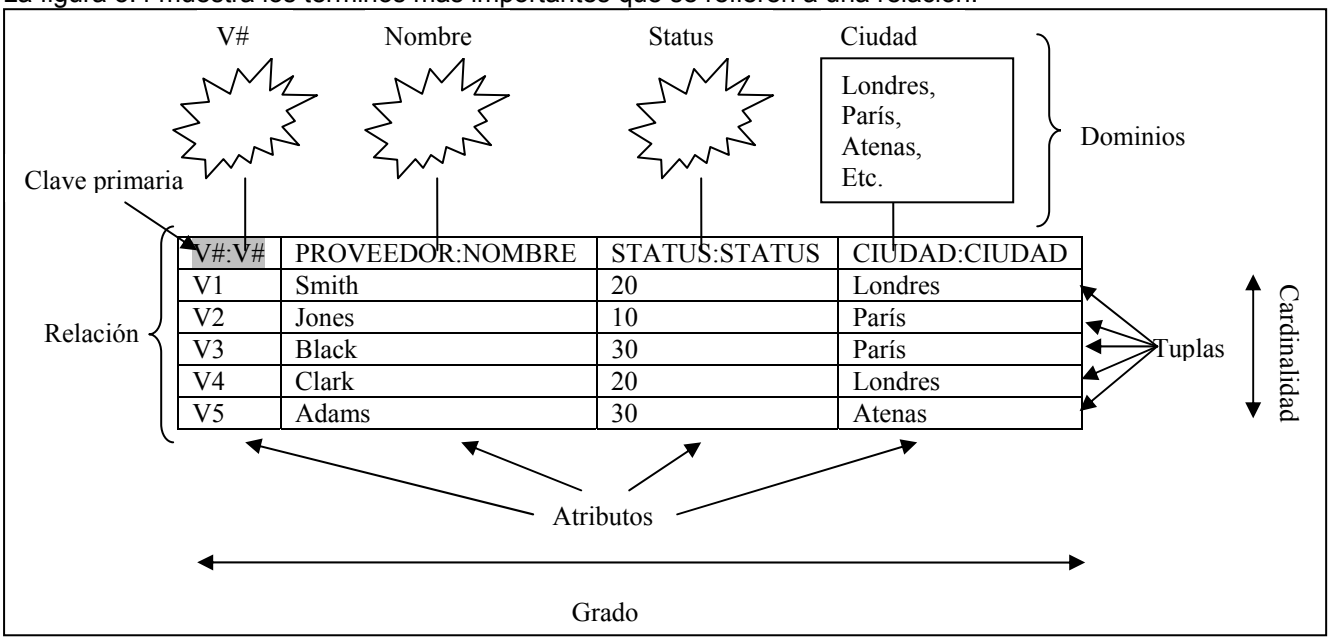


Figura 5.4 Estructura de una relación

La relación es la tabla en sí. Una tupla es una fila de la tabla. Un atributo es una columna de la tabla. La cardinalidad de la relación es el número de tuplas que contiene y el grado es el número de sus atributos o columnas. El dominio es un conjunto de valores de donde se toman los valores específicos posibles de cada atributo.

Dominios

Un dominio es un tipo de datos. Un tipo de datos es un conjunto de valores con las mismas características (por ejemplo, el conjunto de todos los enteros posibles). Cada tipo de datos tiene un conjunto de operadores válidos que se puede ejecutar sobre los valores de ese tipo de datos (en el caso de los valores enteros, se pueden ejecutar los operadores +, -, etc.).

Relaciones

Una relación consta de dos partes:

- Encabezado. Está formado por un conjunto de *n* atributos y cada atributo es de la forma *NombreAtributo*:*NombreTipo*, par que representa la definición de un atributo de la relación, donde *NombreAtributo* es el nombre que tiene el atributo y *NombreTipo* se refiere al tipo de dato del atributo. El número *n* es el grado de la relación.

- Cuerpo. Es un conjunto de *m* tuplas o filas de datos. Cada tupla tiene los valores correspondientes a los atributos en el orden especificado en el encabezado. El número *m* es la cardinalidad de la relación.

Ejemplo:
Encabezado de la tabla EMP

```
CREATE TABLE EMP
(
   EMP# :        E#,
   NOMEMP :      CHAR(20),
   DEPTO:        D#,
   SALARIO :     NUMERIC
)
```

Cuerpo de EMP

EMP#	NOMEMP	DEPTO#	SALARIO
E1	Pedro	D1	40000
E2	Ricardo	D1	45000
E3	Arturo	D2	30000
E4	Mario	D2	35000

Propiedades de las relaciones:

1. No existen filas duplicadas.
2. Las tuplas están en desorden.
3. Los atributos están en desorden.
4. Cada tupla contiene un solo valor para cada atributo. Cuando esto se cumple, se dice que la relación está normalizada o en primera forma normal.

Integridad

La integridad se refiere a la exactitud o corrección de los datos en la base de datos. Se manifiesta mediante restricciones de integridad. Hay tres niveles de restricciones de integridad:

1. Restricciones de atributo.

Es una declaración en la cual se especifica que un atributo de una relación debe ser de un tipo específico. Se utiliza cuando se crean relaciones. Ejemplo:

```
CREATE TABLE EMP
(
   EMP# :        E#,
   NOMEMP :      CHAR(20),
   DEPTO:        D#,
   SALARIO :     NUMERIC
)
```

En esta tabla, los valores de los atributos EMP#, NOMEMP, DEPTO y SALARIO están restringidos a los tipos E#, CHAR(20), D# y NUMERIC respectivamente.

2. Restricciones de tabla.

Son las que se imponen a tablas individuales. Estas restricciones se verifican cuando se intenta insertar o actualizar datos en una tabla. Si los datos insertados o actualizados no cumplen con la restricción, la operación es rechazada. Ejemplos:

```
CREATE TABLE CuentaBanco

(
   NumCuenta :    INTEGER,
   Saldo :        NUMERIC
```

```
    constraint chkSaldo
    check(Saldo >= 0)
)
```

Aquí la restricción indica que el saldo no puede ser negativo.

3. Restricciones de base de datos.
Estas restricciones son las que influyen sobre dos o más tablas relacionadas entre sí. Ejemplo:

```
CREATE TABLE CuentaBanco
(
  NumCuenta :     INTEGER,
  Saldo :         NUMERIC
)

CREATE TABLE CuentaBanco
(
  NumCtaHab :             INTEGER,
  Nombre :                CHAR(150),
  NumCuenta:              INTEGER
      References CuentaBanco(NumCuenta)
)
```

Aquí puede verse la restricción de que la columna NumCuenta en la tabla CuentaBanco hace referencia al valor NumCuenta en la tabla Cuenta. Esto quiere decir que para toda tupla existente en CuentaBanco, su valor en el atributo NumCuenta debe existir en la tabla CuentaBanco.

Integridad declarativa

Se refiere al hecho de crear restricciones de integridad en el momento en que se crean las tablas. Se basa en la observación de los atributos de las tablas que componen la base de datos y en la identificación de sus claves.

Claves

Las claves ayudan a distinguir a cada tupla de una tabla como única. Una clave es simple cuando se trata de un solo atributo o compuesta cuando el conjunto se forma con 2 o más atributos.

- Claves candidatas[23]

Una clave candidata es un conjunto K de atributos de una tabla tal que para cualquier valor válido de los datos en una tabla:
1. El valor que toma el conjunto K de atributos es único para cada tupla. Esta propiedad se llama unicidad.
2. El conjunto K es irreducible o sea que ningún subconjunto de K menor a K tendrá la propiedad de unicidad.

Ejemplo: sea la tabla ALUMNO

NumCuenta	Nombre	Domicilio
97632532	Pedro Pérez	Berlín # 15
98563214	Juan Morales (1)	Juárez # 23
97852365	Mario Domínguez	Puebla # 8
96587412	Rodrigo Díaz	Ciprés # 36
96325418	Marcos Pérez	Berlín # 15
98745316	Juan Morales (2)	Ámsterdam # 56

[23] Date, C.J. *Introducción a los sistemas de bases de datos*. Prentice Hall, Séptima edición, México, 2001.

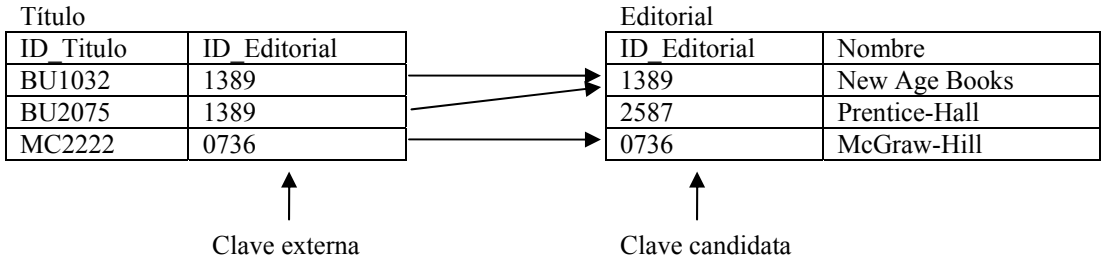
Las claves candidatas son {NumCuenta} y {Nombre, Domicilio}.

- Claves primarias y claves alternas

Una tabla puede tener varias claves candidatas. Para hacer referencia a las tuplas de una tabla, se elige a una de las claves candidatas como clave primaria y las demás son entonces claves alternas. Ejemplo: para la tabla ALUMNO, se toma {NumCuenta} como la clave primaria y {Nombre, Alumno} como clave alterna.

- Claves externas

Una clave externa es un conjunto de atributos en una tabla T1 que hacen referencia a una clave candidata de una tabla T2. Los valores de la clave externa en T1 deben coincidir con los valores de la clave candidata en T2. Cada atributo de la clave externa debe tener el mismo nombre y tipo que los atributos correspondientes de la clave candidata a la que hace referencia. Ejemplo:



Integridad referencial

Se refiere a que no puede haber un valor de clave externa en T1 sin correspondencia en la clave candidata de la tabla T2 referenciada. El valor de una clave externa debe coincidir con el valor de una clave candidata correspondiente.

Acciones referenciales

Una acción referencial es la operación o conjunto de operaciones que debe realizarse en el caso de que el valor de una clave candidata referida por una clave externa sea modificada o eliminada. Una acción referencial tiene como propósito conservar la integridad referencial.

Ejemplo: en las tablas de títulos y editoriales, si se borrara la tupla de la editorial con ID_Editorial 1389 de la tabla Editorial o si se cambiara el valor de la clave candidata, habría un problema de integridad referencial ya que los títulos en la tabla Título que hacen referencia a esa editorial han perdido la referencia. Si se modificó o eliminó la clave candidata en la tabla Editorial, igualmente debería modificarse o eliminarse la o las tuplas en Título cuya clave externa hace referencia al valor de la clave candidata que se ha eliminado o modificado. Mediante una acción referencial se le indica al DBMS que automáticamente realice las actualizaciones pertinentes cuando se borra o cambia el valor de una llave primaria referenciada:

```
CREATE TABLE Editorial
(
  ID_Editorial INTEGER,
  Nombre CHAR(100)
)

CREATE TABLE Titulo
(
  ID_Titulo INTEGER,
  ID_Editorial INTEGER references Editorial(ID_Editorial)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
```

Después de declarar la referencia, se declara la acción referencial. En este ejemplo, el CASCADE en DELETE indica que cuando se borre una editorial en la tabla Editorial, se borren los títulos en la tabla Título que hacen referencia a esa editorial. De la misma manera, el cascade en UPDATE indica que cuando se cambie el ID_Editorial de alguna editorial en la tabla Editorial, se actualice ese valor de ID_Editorial en las tuplas de la tabla Título que hagan referencia a esa editorial. Si en lugar de CASCADE se indica NO ACTION, no se realizará ningún borrado o actualización en la tabla que hace la referencia.

Triggers

Un trigger es un procedimiento programado para llevarse a cabo de manera automática cuando ocurre algún evento de modificación de datos, por ejemplo una inserción, una eliminación o una actualización.

Aunque un trigger puede ser utilizado para llevar a cabo acciones referenciales, es preferible usar la integridad declarativa lo más posible y usar los triggers para restricciones referenciales o también de integridad, complejas que no puedan cubrirse mediante la integridad declarativa.

Conclusiones sobre la integridad

La integridad de los datos es muy importante para los usuarios de la base de datos, ya que asegura la corrección en los valores de los datos, tanto en el aspecto referencial, como en el aspecto de que los datos reflejen de manera fiel la realidad.

En el caso de la integridad referencial, debe definirse la clave externa o foránea y la clave candidata a la que hace referencia y se procede a definir las acciones referenciales para asegurar la integridad referencial. Las acciones referenciales se pueden definir mediante integridad referencial declarativa (restricciones declaradas al crear las tablas) o mediante triggers (creado cuando la restricción es muy compleja para ser declarada al momento de la creación de las tablas).

Prevención y recuperación de errores en los datos

Un error en los datos sucede cuando una operación deja los datos en un estado que viola las restricciones de integridad o un error que deja los datos en un estado que no refleja la realidad.

1. Una operación simple (INSERT, DELETE, UPDATE) ha dejado los datos en un estado que viola las restricciones de integridad.

2. Se ha terminado prematuramente la ejecución de un procedimiento (conjunto de operaciones) debido a una falla en el sistema o en la aplicación de usuario.

En el caso 1, la restricción de integridad no permitirá que se realice la actualización de los datos.

En el caso 2, es probable que los datos sean erróneos.

Ejemplo. Sea una tabla con cuentas bancarias y se quiere transferir 1000 unidades monetarias de la cuenta 1 a la cuenta 2. Las operaciones a realizar son:

```
UPDATE Cuenta
      SET Saldo = Saldo – 1000
 WHERE Cuenta = Cuenta1

UPDATE Cuenta
      SET Saldo = Saldo + 1000
 WHERE Cuenta = Cuenta2
```

Si por alguna razón ocurre un error entre la primera y segunda operación y el procedimiento es abortado, la operación que sí alcanzó a ejecutarse dejó a una cuenta con la cantidad correcta, pero no se pudo actualizar la segunda cuenta o sea que hubo una pérdida grave de la información.

Una recuperación en la base de datos significa restablecerla al estado correcto anterior a la falla.

En este caso, el problema puede solucionarse mediante el uso de transacciones.

Una transacción permite realizar varias operaciones de actualización de datos bajo una misma unidad lógica. Ejemplo:

```
BEGIN TRANSACTION

UPDATE Cuenta
      SET Saldo = Saldo – 1000
 WHERE Cuenta = Cuenta1

UPDATE Cuenta
      SET Saldo = Saldo + 1000
 WHERE Cuenta = Cuenta2

IF Error = 0 THEN
   COMMIT
ELSE
   ROLLBACK
```

La operación COMMIT indica que la transacción finalizó satisfactoriamente y que las actualizaciones deben ser confirmadas y vueltas permanentes sobre la base de datos.

La operación ROLLBACK indica que hubo un error en las operaciones y que las actualizaciones deben ser deshechas.

Así, en caso de falla a la mitad de la transacción, los cambios que se llegan a hacer no se vuelven permanentes y no hay peligro de datos erróneos, aunque hay que volver a ejecutar las operaciones de la transacción.

Las transacciones conservan la consistencia de los datos. Esto quiere decir que una transacción transforma un estado correcto de la información en otro estado correcto, sin que sea necesario conservar la consistencia en todas las partes intermedias.

Diseño de bases de datos

En esta sección se trata los fundamentos teóricos necesarios y las técnicas que nos sirven para el diseño de bases de datos.

Dependencias funcionales[24]

Dada una relación R, el atributo Y en R depende funcionalmente del atributo X en R si y sólo si un solo valor de Y está asociado a cada valor de X. X y Y pueden ser compuestos (tener más de un atributo). Se representa simbólicamente como $X \rightarrow Y$.

Esta definición debe aplicarse no a los valores de una relación en un momento dado, sino a todos los valores posibles en cualquier momento. Por lo tanto, siendo más exactos y es funcionalmente dependiente de X si y solo si en todo valor válido posible de la relación R, siempre que dos tuplas coincidan en su valor X, también coincidirán en su valor Y.

[24] Date, C.J. *Introducción a los sistemas de bases de datos*. Prentice Hall, Séptima edición, México, 2001.

Ejemplo:

#S	Nombre	Zona	Ciudad
S1	Juan	20	Londres
S2	Pedro	10	París
S3	Mario	10	París
S4	Pablo	20	Londres
S5	Jorge	30	Atenas

Las dependencias funcionales son:
#S → Nombre
Nombre → #S
#S → Zona
Zona → Ciudad
Ciudad → Zona
#S → Ciudad

A la parte izquierda y derecha de una dependencia funcional se les llama determinante y dependiente, respectivamente.

Dependencias triviales y no triviales

Una dependencia es trivial si y solamente si la parte derecha o dependiente es un subconjunto de la parte izquierda o determinante[25].

Redundancia

La redundancia en la información significa que hay información repetida en la tabla.

Si X es una clave candidata de R, entonces todos los atributos de R dependen funcionalmente de X. Pero si se da el caso de que existe una dependencia funcional A → B en R y A no es una clave candidata, entonces existe redundancia en R. Ejemplo:

Sea la siguiente tabla de envíos de proveedores:

Proveedor	Ciudad	Envío	Cantidad
P1	Londres	E1	300
P1	Londres	E2	200
P1	Londres	E3	500
P1	Londres	E4	400
P2	París	E1	600
P2	París	E2	300
P3	Madrid	E1	200
P3	Madrid	E2	500
P4	Londres	E1	100

En este ejemplo hay redundancia, ya que se puede ver que para cada tupla, el proveedor P1 siempre está en Londres y el P2 siempre está en París, etc. Esto quiere decir que un proveedor dado está ubicado en una ciudad dada.

Esta redundancia conduce a varios problemas. Uno de ellos es que si un proveedor se cambia de ciudad, en la tabla es posible cambiar el valor de una ciudad en una sola tupla para el proveedor sin hacer la actualización en todas las tuplas que sean del mismo proveedor. Esto significaría una inconsistencia en la información. Como se verá más adelante, mediante el proceso de normalización se soluciona este problema.

[25] Date, C.J. *Introducción a los sistemas de bases de datos*. Prentice Hall, Séptima edición, México, 2001.

Dependencias reducibles e irreducibles

Un conjunto de dependencias funcionales de una relación es reducible si para toda dependencia funcional en ese conjunto:

1. La parte derecha (dependiente) es un conjunto de un solo atributo.
2. La parte izquierda (determinante) sea irreducible. Esto quiere decir que la dependencia deja de existir al quitar uno o más elementos del determinante.
3. No es posible eliminar una dependencia del conjunto de dependencias funcionales. Esto quiere decir que el conjunto de dependencias funcionales es irreducible.

Importancia de las dependencias funcionales

La importancia de identificar las dependencias funcionales en una relación radica en que representan restricciones de integridad referencial de los atributos de la relación y por lo tanto deben cumplirse siempre. Para asegurar esto es necesario identificar las dependencias funcionales existentes para todos los valores válidos posibles de una relación y reducir ese conjunto de dependencias a un tamaño manejable. Desde un punto de vista práctico, esto quiere decir que para un conjunto S de dependencias funcionales de una relación, hay que encontrar un conjunto T mucho menor, pero que cuyas dependencias funcionales hagan cumplir las dependencias funcionales de S. Esto significa un menor número de restricciones de integridad, pero equivalentes. Una manera inicial de reducir el conjunto de dependencias es reduciendo las dependencias triviales. A partir de ahí, hay que identificar las dependencias funcionales que provocan redundancia en la información y reducirlas mediante el proceso de normalización de las relaciones.

Normalización

La normalización es un proceso mediante el cual se analizan las relaciones para obtener sus dependencias funcionales, e identificar aquellas que provocan problemas de redundancia y/o inconsistencia al tratar de llevar a cabo una operación de inserción, eliminación o actualización de datos. Posteriormente se resuelve estos problemas reemplazando la relación analizada por dos o más relaciones con encabezados tales que en estas nuevas relaciones se han eliminado los problemas.

La normalización nos ayuda a estructurar una base de datos de modo que una actualización a una tupla individual no provoque problemas de inconsistencia de la información.

El procedimiento de normalización nos permite reemplazar una relación en un nivel dado de normalización, por un conjunto de relaciones en un nivel mayor de normalización que el nivel de la relación original. Es un proceso reversible, lo cual quiere decir que podemos tomar las relaciones resultantes del proceso y regresar a la relación original, conservando la información exactamente como estaba.

Formas normales

Primera Forma Normal (1FN).

Una relación está en 1FN si y solo si cada tupla contiene exactamente un valor para cada atributo. De acuerdo con la definición de una relación según el modelo relacional, se puede ver que todas las relaciones están siempre en 1FN.

Segunda Forma Normal (2FN).

Una relación está en 2FN si y solo si está en 1FN y todo atributo no clave es dependiente irreduciblemente de la clave primaria.

Ejemplo: sea la siguiente relación de suministros[26]:

Proveedor	Status	Ciudad	Envío	Cantidad
P1	20	Londres	E1	300
P1	20	Londres	E2	200
P1	20	Londres	E3	400
P1	20	Londres	E4	200
P1	20	Londres	E5	100
P1	20	Londres	E6	100
P2	10	París	E1	300
P2	10	París	E2	400
P3	10	París	E2	200
P4	20	Londres	E2	200
P4	20	Londres	E4	300
P4	20	Londres	E5	400

Clave primaria : {Proveedor, Envío}
Las dependencias funcionales son:

1. {Proveedor, Envío} → {Ciudad}
2. {Proveedor, Envío} → {Status}
3. {Proveedor} → {Ciudad}. Esto significa que la dependencia 1 es reducible.
4. {Ciudad} → {Status}
5. {Proveedor} → {Status}. Esto significa que la dependencia 2 es reducible.
6. {Proveedor, Envío} → {Cantidad}

Se observa que hay redundancia en la relación, ya que el proveedor determina la ciudad y la ciudad determina el status. Estas redundancias traen problemas como los siguientes:

1. No se puede insertar información de un nuevo proveedor (por ejemplo, el proveedor P5, ubicado en Madrid) hasta que haya un suministro de ese proveedor.
2. Si se eliminara el único suministro que existe del proveedor P3, se pierde también la información de ciudad y status de ese proveedor.
3. Si un proveedor se cambia de ciudad, al actualizar la relación se corre el riesgo de no actualizar todas las tuplas de ese proveedor, provocando una inconsistencia en la información.

Para conseguir la 2FN, se elimina las redundancias reducibles reemplazando esta relación por dos relaciones:

R1

Proveedor	Status	Ciudad
P1	20	Londres
P2	10	París
P3	10	París
P4	20	Londres
P5	30	Madrid

R2

Proveedor	Envío	Cantidad
P1	E1	300
P1	E2	200
P1	E3	400
P1	E4	200
P1	E5	100
P1	E6	100

[26] Date, C.J. *Introducción a los sistemas de bases de datos*. Prentice Hall, Séptima edición, México, 2001.

P2	E1	300
P2	E2	400
P3	E2	200
P4	E2	200
P4	E4	300
P4	E5	400

Con esto es posible realizar las operaciones de inserción, eliminación y actualización descritas anteriormente, sin problemas.

Tercera Forma Normal (3FN).

Una relación está en 3FN si y solo si está en 2FN y todos los atributos no clave dependen de manera no transitiva de la clave primaria.

Sea la relación R1 anterior.
Las dependencias funcionales son:

1. {Proveedor} → {Ciudad}
2. {Proveedor} → {Status}
3. {Ciudad} → {Status}

Se puede ver que la dependencia de Status sobre Proveedor, aunque es irreducible, es transitiva a través de Ciudad. Esto se puede ver como:

Si las dependencias funcionales A → B y B → C son válidas, entonces la dependencia funcional A → C también es válida[27].

Las dependencias transitivas también provocan problemas al intentar operar sobre las relaciones que las tienen:

1. No se puede insertar la información del status de una ciudad hasta que haya un proveedor en esa ciudad.
2. Si se elimina la única tupla de un proveedor, se pierde la información del status de una ciudad, además de la información del proveedor.
3. Si se quiere cambiar el status a una ciudad, se corre el riesgo de no actualizar todas las tuplas de esa ciudad, provocando información incorrecta.

Para conseguir la 3FN se elimina las dependencias transitivas, reemplazando la relación R1 por dos relaciones:

R1a

Proveedor	Ciudad
P1	Londres
P2	Paris
P3	Paris
P4	Londres
P5	Madrid

R1b

Ciudad	Status
Madrid	30
Londres	20
Paris	10
Roma	50

[27] Date, C.J. *Introducción a los sistemas de bases de datos*. Prentice Hall, Séptima edición, México, 2001.

La referencia se conserva, como se puede ver en la relación R1a, donde la columna Ciudad hace referencia a la clave primaria Ciudad en R1b. Esta restricción referencial debe crearse al crearse la tabla, mediante la integridad referencial declarativa:

```
CREATE TABLE R1b
(
    Ciudad CHAR(15) Primary Key,
    Status INTEGER
)

CREATE TABLE R1a
(
    Proveedor P# Primary Key,
    Ciudad CHAR(15) references R1b(Ciudad)
)
```

FNBC (Forma Normal Boyce-Codd)

Es un caso especial de la 3FN, que se aplica cuando se tiene que:

1. Hay varias llaves candidato.
2. Esas llaves candidato son compuestas (tienen dos o más atributos).
3. Las llaves candidato se traslapan (tienen uno o más atributos en común).

Una relación está en FNBC si y solo si los únicos determinantes son llaves candidato.

Modelado semántico

Definición general

El modelado semántico es una actividad auxiliar muy útil del diseño de bases de datos, mediante la cual se representa el significado de las entidades y sus atributos en el mundo real, más allá de solamente el tipo de dato de cada atributo. El modelado semántico también se conoce como modelado de datos, modelado de entidades o modelo entidad-relación, entre otros.

Los pasos para el modelado semántico son:

1. Una definición informal de los conceptos que definen la realidad:
   - Entidades. Una entidad es algo que puede ser identificado en forma distintiva. Las entidades pueden ser clasificadas en tipos. Un tipo de entidad tiene un conjunto específico de atributos o propiedades. Un atributo define una característica de la entidad, por ejemplo color, peso, altura, edad, densidad, etc. Toda entidad tiene una propiedad especial que tendrá un valor único para cada instancia de esa entidad y por lo tanto mediante esa propiedad especial o identidad se identificará a cada instancia de la entidad.
   - Vínculos. Una entidad puede relacionarse con otras entidades por medio de vínculos.

2. Se busca una manera de definir formalmente los conceptos anteriores. Una forma es mediante el modelo relacional, el cual proporciona relaciones para las entidades, cuyo encabezado se formaría con los atributos de esa entidad y relaciones para los vínculos, cuyo encabezado son las claves que hacen referencia a las entidades participantes en el vínculo, además de las propiedades propias del vínculo.

3. Se crea un conjunto de restricciones de integridad para cada uno de los objetos antes descritos.

Modelo Entidad/Relación (E/R)

El enfoque o modelo E/R es un método muy conocido y usado de modelado semántico. Este modelo, además de una definición de conceptos semánticos equivalentes a las entidades y los vínculos, también utiliza una técnica de elaboración de diagramas para representar gráficamente los conceptos semánticos. Estos diagramas sirven para representar la estructura lógica de una base de datos en forma de gráficos, una forma de apreciar las principales características del diseño de una base de datos. La figura 5.5 muestra un ejemplo de este modelo:

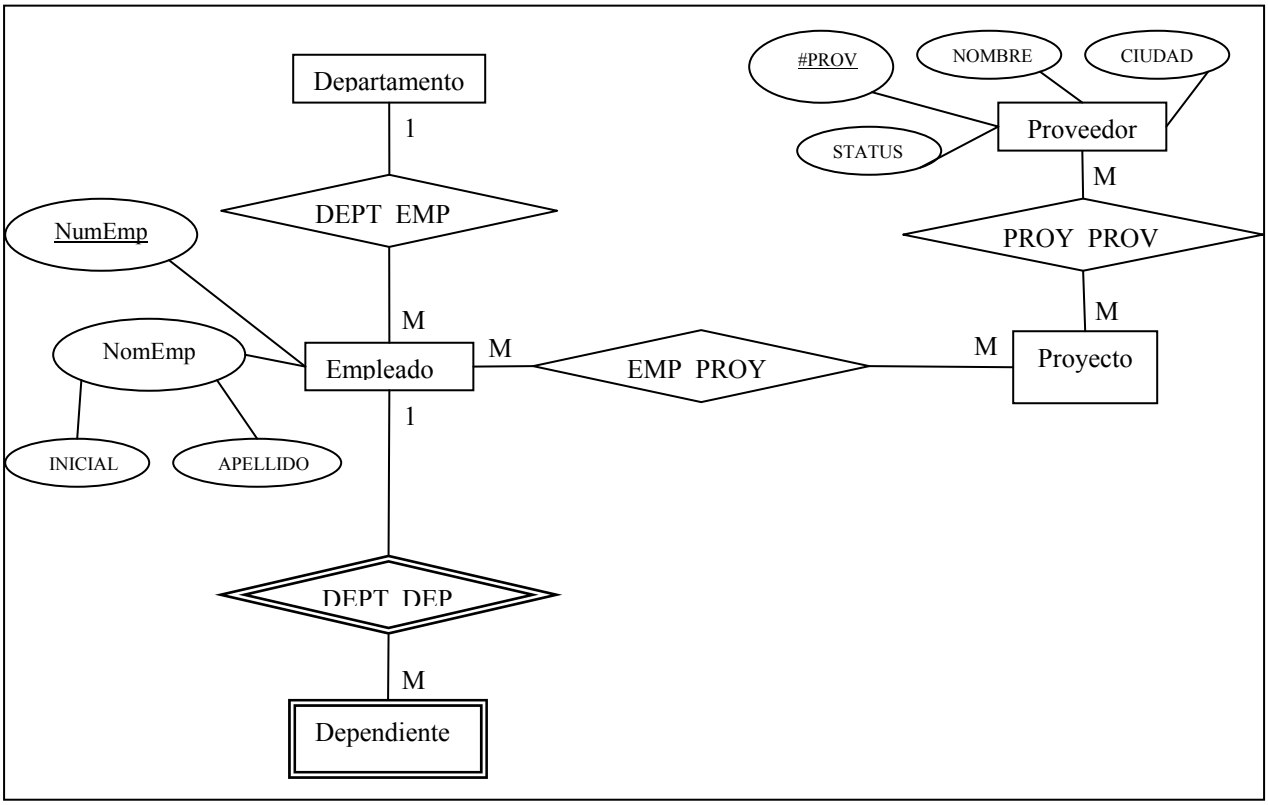


Figura 5.5 Diagrama Entidad – Relación

En la figura 5.6 se observan los siguientes elementos:

Entidades

Una entidad es algo que puede ser identificado en forma distintiva o distinguido de otras entidades a través de sus propiedades.

Hay dos tipos de entidades: una entidad débil es aquella cuya existencia depende de la existencia de otra entidad; una entidad normal o fuerte es aquella cuya existencia no depende de la existencia de otra entidad. En la figura 5.5 se observa cómo una entidad normal o fuerte se representa como un rectángulo con el nombre de la entidad dentro de él y una entidad débil se representa como un rectángulo doble.

Propiedades

Son las características o atributos de las entidades. Todas las entidades o vínculos tienen propiedades. Cada propiedad tiene un tipo de dato, lo cual quiere decir que los valores posibles de la propiedad se toman de un conjunto de valores definido, también llamado dominio en términos del modelo relacional. Una propiedad puede ser:

- Simple o compuesta. Una propiedad compuesta está formada por dos o más propiedades simples.
- Clave. Es una propiedad que toma un valor único para cada instancia de la entidad a la que corresponde.

- Base o derivada. Una propiedad derivada es aquella cuyo valor está en función del valor de otra u otras propiedades. Una propiedad que no es derivada es una propiedad base.
- Multivaluada. Una propiedad X es multivaluada cuando hay varias instancias de la entidad en las cuales sólo el valor de X cambia y las demás tienen el mismo valor.

Las propiedades se muestran como elipses que contienen el nombre de la propiedad y se conectan a la entidad o vínculo al que pertenecen mediante líneas continuas. Cuando una propiedad es derivada la elipse es punteada. Cuando la propiedad es compuesta, sus propiedades componentes se muestran como elipses con los nombres de esas propiedades y unidas mediante una línea a la propiedad compuesta. Las propiedades que son claves tienen el nombre subrayado.

Vínculos

Un vínculo es una asociación entre entidades. A las entidades involucradas en el vínculo se les llama participantes y el número de participantes indica el grado del vínculo.

Un vínculo puede ser uno a uno (1:1), uno a muchos (1:M) o muchos a muchos (M:M).

Se muestra cada vínculo como un rombo que contiene el nombre del vínculo. El rombo es doble cuando es un vínculo entre una entidad débil y aquella de la cual depende. Se conecta a cada participante mediante líneas continuas, etiquetadas con “1” o “M” para indicar que el vínculo es 1:1, 1:M o M:M.

Diseño de bases de datos con el modelo E/R

Un diagrama E/R es en cierta forma un diseño de bases de datos. Pero al transformar directamente este diseño a una construcción física, el resultado sería una base de datos sin restricciones de integridad y cuyas relaciones no estarían normalizadas, lo cual como ya vimos en la sección referente a normalización, puede traer problemas de redundancia de información e inconsistencia.

Conversión del diagrama E/R en un diseño conceptual de base de datos

Los pasos a seguir para convertir los elementos de un diagrama E/R en una definición de bases de datos desde el punto de vista del modelo relacional son:

1. Entidades normales

Cada tipo de entidad normal se transforma en una tabla. Cada tabla tendrá una clave primaria. Es necesario definir los dominios o conjuntos de valores de cada atributo.

2. Vínculos M:M

Cada uno de estos vínculos se transforma en una tabla. Cada tabla debe incluir las claves externas que hagan referencia a los participantes correspondientes. Es necesario especificar un conjunto de reglas de clave externa para cada clave externa (regla DELETE y regla UPDATE, ver sección de integridad referencial).

La clave primaria de cada tabla creada a partir de un vínculo M.M puede formarse de la combinación de las claves externas de todos los participantes, con lo cual se forma una PK compuesta. Otra opción es crear un atributo simple para este vínculo y usarlo como clave primaria.

3. Vínculos M:1

Hay dos casos:

1. Cuando se trata de un vínculo entre entidades normales. En este caso no se crea una tabla nueva para el vínculo. Basta con crear una clave externa en la tabla participante que está del lado “muchos” del vínculo. Esta clave externa debe hacer referencia a la tabla que está del lado “uno” del vínculo. Las reglas DELETE y UPDATE son en general las mismas que en los vínculos M:M

2. Cuando se trata de un vínculo donde existe al menos una entidad débil. En este caso se crea una tabla para el vínculo. Esta tabla contiene las claves externas que hacen referencia a las entidades

fuertes participantes y las claves externas que hacen referencia a las entidades débiles participantes. En este caso las reglas DELETE y UPDATE son ambas CASCADE, debido a la dependencia de la entidad débil, ya que necesitamos que cuando la variable a la que hace referencia se borre o se modifique, se haga lo mismo en la tabla con la clave externa que hace referencia.

Al igual que en el caso de los vínculos M:M, la clave primaria de esta tabla puede ser la combinación de las claves externas o se puede crear un atributo simple que tenga valores únicos.

4. Vínculos 1:1

Estos vínculos se tratan exactamente de la misma manera que los vínculos M:1.

5. Propiedades.

Cada propiedad mostrada en el diagrama E/R se transformará en un atributo de la tabla correspondiente. Sólo en el caso de que la propiedad esté multivaluada, se crearán nuevas tablas mediante los procedimientos de normalización. No hay que olvidar crear un dominio para cada atributo.

Conclusiones sobre diseño de bases de datos

Como podemos ver, las técnicas de modelado semántico, por ejemplo el modelo E/R y las técnicas de normalización son complementarias en la actividad de diseño de bases de datos. Primero se realiza la representación lógica de las entidades, propiedades y vínculos entre las entidades, que son un reflejo de la realidad de los datos que deseamos representar y posteriormente utilizamos las técnicas de normalización para crear las relaciones adecuadas, sustituyendo cuando sea necesario las relaciones "generales" del modelo E/R, por otras relaciones más simples que nos permitan cumplir con las restricciones de integridad, asegurándonos la corrección de los datos y la correcta inserción o actualización de datos.

Una aplicación importante del modelado semántico, específicamente del modelo E/R, además de ser el primer paso en el diseño de bases de datos, es la ayuda para la creación de un diccionario de datos. Un diccionario de datos es una lista donde se concentra la información que nos permite comprender la naturaleza y función de los objetos importantes de la base de datos: los tipos de objetos, sus atributos, su significado en el mundo real, las relaciones entre los objetos, las reglas de integridad, etc.

Capítulo 6
Estudio de factibilidad

Fuentes de información

La fuente principal de información para conocer la necesidad de un sistema de software y los objetivos a satisfacer fue la observación del comportamiento del servicio de transporte.

Entorno de la organización

Se trata del servicio de transporte interno gratuito de Ciudad Universitaria. Es una organización que provee el servicio de transporte público en varias rutas.

Cada ruta consiste en un número de estaciones. El tipo de ruta es cerrada, lo cual quiere decir que la estación de salida es la misma que la de llegada.

Objetivos de la organización:

- Proveer de servicio de transporte a lo largo de cada ruta que se ha definido.
- Satisfacer la demanda de transporte a lo largo de cada ruta.
- Minimizar los costos de operación de cada ruta.

La organización actualmente realiza el servicio de la siguiente manera:

- El número de unidades necesario para ofrecer el servicio, así como la frecuencia de salida de las mismas está en función principalmente de la demanda que haya en la estación de salida de la ruta (la estación en el metro Universidad).
- Por experiencia, se sabe que en ciertos periodos del día hay un considerable aumento en la demanda en algunas estaciones intermedias de la ruta. Para satisfacer esta demanda, se mandan unidades de transporte vacías directamente a esas estaciones, sin detenerse en las estaciones anteriores.

No se cuenta con análisis que muestren las variaciones en el servicio dependiendo de la demanda variable a lo largo de un día o la demanda de los distintos días de la semana o de las distintas temporadas del año. Cuando se ha hecho algún estudio o muestreo para conocer la demanda de transporte en la ruta, ésta no tiene el tamaño suficiente para ser representativo. Además, los datos se conservan en hojas de papel. Esto hace que se tenga un gran volumen de datos difíciles de manejar y procesar. Al estar los datos en papel y al hacerse el análisis de los mismos de forma manual, el análisis es muy lento y susceptible de errores.

Los resultados de los análisis distan mucho de ser óptimos, debido a la cantidad insuficiente de datos y a los errores cometidos durante el análisis.

Objetivos del sistema

Se requiere un sistema que:

- Facilite la definición de una ruta de transporte, así como su modificación.
- Facilite la división del servicio provisto por la ruta de transporte en temporadas, tipos de día y horarios diferentes por sección del día.
- Facilite la captura y almacenamiento de los datos de los muestreos que se lleven a cabo.
- Facilite y acelere el análisis de los datos de los muestreos, mediante análisis numéricos, generación de gráficas, etc. para obtener factores importantes del servicio, como demanda a lo largo de la ruta, velocidad promedio de recorrido, etc.
- Facilite y acelere la creación de horarios que de manera óptima satisfagan la demanda de transporte, en función de los factores obtenidos en el análisis definido anteriormente y en función de las características de la infraestructura de transporte. Obtener resultados como número de unidades necesarias para satisfacer la demanda, frecuencia de paso, etc.
- Sea confiable (poca probabilidad de fallos) y que asegure la integridad de los datos.

Restricciones sobre recursos para el desarrollo y operación del sistema

Se requiere que las actividades de desarrollo e implementación del sistema tomen en total un tiempo máximo de 6 meses, trabajando 6 horas efectivas diarias y de lunes a viernes cada semana.

Se dispone para el trabajo de desarrollo del sistema, de una persona y una PC con un procesador de 2.4 GHz, 256 MB de RAM y 60 GB de disco duro.

Para la operación del sistema se requiere una PC con un procesador de 1 GHz, 128 MB de RAM y al menos 10 GB de espacio en disco duro.

Análisis de factibilidad

a) Factibilidad técnica

En esta etapa del análisis se revisaron los objetivos para el sistema deseados por el cliente y se investigó la posibilidad de construir un sistema que cumpla con esos objetivos, utilizando la tecnología conocida hasta el momento por el desarrollador. Se llegó a la conclusión de que sí se puede construir el sistema deseado con la tecnología actual conocida por la persona dedicada a desarrollar e implementar el sistema. Por otro lado, se llegó a la conclusión de que los clientes cuentan con la tecnología necesaria para operar el sistema una vez que esté desarrollado. El aspecto de la factibilidad técnica tiene una importancia muy grande, ya que contar con la tecnología necesaria es esencial para lograr el sistema deseado.

b) Factibilidad económica

En esta etapa se tomaron en cuenta los recursos con que cuenta la persona dedicada al desarrollo del sistema. Se llegó a la conclusión de que sí se cuenta con la infraestructura (hardware) necesaria para desarrollar el sistema, así como los fondos suficientes para realizar el trabajo de desarrollo. También se analizó este aspecto de la factibilidad desde el punto de vista del cliente y se concluyó que éste cuenta con los recursos económicos necesarios para solventar el desarrollo del sistema y para operarlo una vez construido. El aspecto de la factibilidad económica tiene una importancia muy grande, ya que contar con los recursos necesarios es esencial para lograr el desarrollo y operación del sistema.

c) Factibilidad legal

Para desarrollar e implementar el sistema no existen problemas de tipo legal.

d) Factibilidad operacional

En esta etapa se tomaron en cuenta los conocimientos teóricos y técnicos con que cuenta la persona dedicada al desarrollo del sistema. Se llegó a la conclusión de que sí se cuenta con la experiencia y conocimientos necesarios para desarrollar el sistema. Este aspecto tiene una importancia muy grande, ya que contar con los conocimientos y experiencia necesarios es esencial para lograr el desarrollo de un sistema que cumpla con los objetivos del sistema y que funcione correctamente.

e) Factibilidad de programa

En esta etapa se tomó en cuenta la restricción de tiempo existente para el desarrollo del sistema. Después de una estimación aproximada del tiempo que será necesario para realizar las actividades de desarrollo e implementación del sistema, se llegó a la conclusión de que sí puede construirse el sistema dentro del límite de tiempo impuesto de 121.82 días hábiles. El aspecto de la factibilidad de programa tiene una importancia muy grande, ya que se desea tener el sistema a tiempo con el fin de optimizar el servicio de las rutas de transporte.

Resultados del análisis de factibilidad

Con base en la información anterior sobre la manera en que actualmente se realizan las actividades para proveer el servicio de transporte, se concluye que el sistema que se requiere sí ayuda a cumplir con los objetivos de la organización.

Sí se puede construir un sistema que lleve a cabo las funciones descritas anteriormente por el cliente.

El sistema sí se puede construir con la tecnología actual.

El sistema sí puede construirse dentro del límite de tiempo impuesto por el cliente.

El sistema puede integrarse fácilmente a la organización, llevando a cabo cambios en la distribución temporal del trabajo entre los operadores de las unidades de transporte.

Se puede tener acceso fácilmente a la información procesada y generada por el sistema.

El sistema no requiere de tecnología que no se haya utilizado antes en la organización o que sea difícil de conseguir o que requiera de recursos con los que la organización no cuente. La organización que requiere el sistema es capaz de comprar una PC con las características necesarias para su operación.

Capítulo 7
Análisis de riesgos del sistema

Se realiza la identificación de los riesgos que podrían poner en peligro o suspender definitivamente la realización o aceptación del sistema. Para cada riesgo se da un número único que lo identifique, una descripción, una acción de mitigación para intentar evitar el problema y una acción de contingencia para resolver el problema lo más pronto posible, si éste ocurre.

Descripción de riesgos

Número	Descripción	Mitigación	Contingencia
1	Falla o descompostura de computadora disponible. Consecuencia: pérdida de archivos del proyecto.	Hacer copia de respaldo de archivos del proyecto en otra computadora.	Continuar trabajo en otra computadora.
2	Cambios en los requerimientos existentes mayores al 20%. Consecuencia: modificaciones en requerimientos, diseño y prototipo.	Realizar análisis de requerimientos en forma efectiva desde la primera etapa. Analizar los requisitos para detectar ambigüedades o contradicciones.	Hacer cambios necesarios en los requerimientos, diseño y código.
3	Subestimación del tiempo para las actividades. Consecuencia: más tiempo necesario para las actividades, retraso en el proyecto.	Realizar planeación lo más detallada posible. Trabajar efectivamente las horas planeadas.	Replanear las actividades del proyecto. Negociar cambios en la planificación temporal.
4	Subestimación del tamaño de los componentes del sistema. Consecuencia: más tiempo necesario para construcción de los componentes, retraso en el proyecto.	Realizar descomposición del sistema lo más detallada posible para estimación. Trabajar efectivamente las horas planeadas.	Replanear las actividades del proyecto. Negociar cambios en la planificación temporal.
5	Prototipo no acorde con los requerimientos del sistema. Consecuencia: cliente insatisfecho.	Rastrear la correspondencia entre requerimientos y diseño y entre diseño y prototipo.	Realizar los cambios necesarios en el diseño y prototipo.

A continuación se asigna una probabilidad e impacto a cada riesgo, con el fin de tener una base para el monitoreo de los riesgos que representan más peligro y sus actividades de mitigación y contingencia.

Probabilidad e impacto

Número	Descripción	Probabilidad	Impacto
1	Falla o descompostura de computadora disponible.	Baja	Moderado
2	Cambios en los requerimientos existentes mayores al 20%.	Baja	Alto
3	Subestimación del tiempo para las actividades.	Moderada	Moderado
4	Subestimación del tamaño de los componentes del sistema.	Moderada	Moderado
5	Prototipo no acorde con los requerimientos del sistema.	Bajo	Alto

Los valores cualitativos posibles contemplados para la probabilidad de que cada riesgo se vuelva un problema y sus respectivos valores numéricos equivalentes son:

- Baja: entre 0% y 33%
- Moderada: entre 33% y 66%
- Alta: entre 66% y 100%

Los valores cualitativos posibles contemplados para el impacto en el caso de que cada riesgo se vuelva un problema y sus significados son:

- Bajo: no afecta de manera significativa al avance o al logro de los objetivos del proyecto
- Moderado: provoca un retraso moderado o cambios moderados en el proyecto, en la definición de los requerimientos, el diseño o el código
- Alto: provoca un retraso considerable en el avance del proyecto o cambios drásticos en los requerimientos, diseño o código

Capítulo 8
Especificación de requerimientos del sistema

8.1 Introducción

8.1.1 Propósito

Mostrar la especificación de requerimientos del sistema de optimización de rutas de transporte, además de describir a quién va dirigido, su alcance y las restricciones bajo las que debe operar el sistema. Establece las funciones que el sistema proporciona, su respuesta ante determinadas entradas de datos y el comportamiento que tendrá en situaciones particulares.

8.1.2 A quién va dirigido

La especificación de requerimientos está dirigida al personal que, directa o indirectamente, está involucrado en el desarrollo del sistema: al líder de proyecto, patrocinador, desarrollador, usuarios y verificadores del correcto funcionamiento del sistema.

Los requerimientos se encuentran organizados de la siguiente manera: primero se presentan la visión y el alcance del sistema, una perspectiva de sus características principales, las clases de usuarios involucrados en su operación, la perspectiva general del ambiente de operación del sistema, la documentación de soporte requerida, los supuestos y dependencias del sistema, los escenarios de uso del sistema y se especifican los requerimientos funcionales que conforman cada uno de dichos casos de uso. Después se presentan los requerimientos de interfaces externas con las que deberá cumplir el sistema. Posteriormente se tratan los requerimientos no funcionales o atributos de calidad. Después se presenta el modelo de datos definido para el sistema. Por último se muestra el método de cálculo de horarios óptimos que se toma como base para la administración del horario de servicio de una ruta.

8.1.3 Visión y alcance del sistema

Visión

El sistema tiene como objetivo ayudar a cualquier persona encargada de administrar una ruta de transporte, a automatizar la creación de una ruta y la división del servicio en temporadas, tipos de día y secciones de día, a agilizar y hacer más fácil el manejo de los datos que resulten de los muestreos que se hacen en una ruta para conocer la demanda existente y con base en la demanda y a las características de la ruta, calcular un horario óptimo, que satisfaga la demanda existente de transporte y optimizando los gastos de infraestructura.

Alcance

Se establece que el alcance para el sistema comprenderá los siguientes módulos funcionales:

- Creación, modificación y eliminación de una ruta.
- Creación, modificación y eliminación de un catálogo de tipos de unidades de transporte.
- Creación de temporadas de servicio.
- Creación de tipos de día de servicio dentro de las temporadas.
- Creación de horarios de servicio dentro de los tipos de día.
- Captura de los datos de los muestreos.
- Generación de gráficas de demanda para una ruta, por temporada, tipo de día o sección de un día.
- Generación de horarios óptimos.

8.2 Descripción general

8.2.1 Perspectiva del producto

Se trata de un sistema que surge de la necesidad de automatizar los procesos de captura y análisis de los datos de muestreos realizados para conocer el nivel de servicio de una ruta de transporte y para conocer la demanda de transporte existente en la misma, así como automatizar la administración de las rutas que se ha definido y generar los horarios óptimos con las frecuencias de paso que asegurarán que la demanda de transporte quede satisfecha. Los objetivos de la organización que requiere el software son proveer de servicio de transporte a lo largo de las rutas que tiene definidas, satisfaciendo la demanda de transporte existente y al mismo tiempo minimizando los costos de operación de las rutas.

Para cumplir estos objetivos, la organización realiza de manera manual y empírica la asignación de unidades de transporte a las rutas de transporte que tiene definidas. El sistema ayudará antes que nada a asegurar el rápido y buen manejo de los datos de los muestreos que se hagan en el servicio de las rutas, para conocer la demanda real existente. El sistema también ayudará a realizar el estudio de los datos provenientes de los muestreos, mediante análisis numéricos y gráficas. El sistema automatizará la creación y administración de las rutas y la creación de horarios con las frecuencias de paso de forma que se satisfaga la demanda y se minimicen los costos de operación de las rutas.

8.2.2 Características del producto

Las funcionalidades que el sistema contiene en esta versión son:

- Creación, modificación y eliminación de una ruta.
- Creación, modificación y eliminación de un catálogo de tipos de unidades de transporte.
- Creación de temporadas de servicio.
- Creación de tipos de día de servicio dentro de las temporadas.
- Creación de horarios de servicio dentro de los tipos de día.
- Captura de los datos de los muestreos.
- Generación de gráficas de demanda para una ruta, por temporada, tipo de día o sección de un día.
- Generación de horarios óptimos.

Estas características, así como los requerimientos funcionales de cada una de ellas, se describen con detalle en la definición de escenarios.

8.3 Clases de usuarios

El usuario que utilizará este sistema es el encargado de definir las rutas, las temporadas, tipos de día y secciones, así como asignar vehículos a una ruta y definir el horario de servicio de la misma. Este usuario interviene en todos los procesos que se requiere que el sistema realice.

8.4 Ambiente de operación

Con base en la información proporcionada por los interesados en el sistema, así como en la infraestructura con la que cuenta la organización, se puede concluir que el sistema operará bajo el siguiente ambiente:

Descripción	Elemento del ambiente de operación
Plataforma de hardware	PC.
Sistema operativo	*Windows 98, 2000 o XP.*
Manejador de base de datos	*SQL Server*
Lenguaje de programación	*Delphi*

8.5 Documentación de usuario

El sistema será soportado por una documentación del proceso de análisis y diseño. Esta documentación se conformará por los siguientes productos dirigidos al usuario:

Especificación de requerimientos de software. Es el documento que contiene las necesidades de los usuarios y la especificación de los requerimientos del sistema.

Documento de diseño. Proporcionará los detalles de cómo se planea implementar el sistema y el razonamiento sobre los atributos de calidad del mismo.

Prototipo. Se realizará un prototipo para que los usuarios puedan dar la retroalimentación necesaria.

8.6 Supuestos y dependencias

Para llevar a cabo el desarrollo del sistema, el encargado del análisis de requerimientos, diseño y construcción del mismo debe contar con un equipo de cómputo con las características mencionadas en el ambiente de operación.

8.7 Escenarios de uso del sistema

Un escenario es una actividad discreta que un usuario puede realizar con el sistema para obtener un resultado.

En este capítulo se especifican los escenarios de uso del sistema y para cada escenario se define: una descripción, prioridad y flujos de información y control.

8.7.1 Administración de rutas

Descripción y prioridad

El usuario crea, elimina o abre una ruta para su edición.
La prioridad de este escenario es indispensable.

Flujo de eventos

Precondiciones
El usuario ha ingresado al sistema.

Flujo básico

El sistema muestra una lista con las rutas existentes. Por default el sistema selecciona la primera ruta en la lista. El sistema muestra los siguientes datos de la ruta:

- Nombre de la ruta
- Lista de nombres de las temporadas de la ruta. Por default el sistema selecciona la primera temporada de la lista.
- Lista de nombres de los tipos de día de la temporada seleccionada. Por default el sistema selecciona el primer tipo de día.
- Lista de nombres de los horarios del tipo de día seleccionado.

Si el usuario escoge una ruta, el sistema despliega las temporadas definidas para la ruta escogida.

Si el usuario escoge una temporada de una ruta, el sistema despliega los tipos de día definidos para esa temporada.

Si el usuario escoge un tipo de día de una temporada, el sistema despliega los horarios definidos para ese tipo de día.

Si el usuario escoge la opción “Nueva ruta” se ejecuta el escenario alterno A1 Nueva ruta.

Si el usuario escoge la opción “Abrir ruta” se ejecuta el escenario alterno A2 Abrir ruta.

Si el usuario escoge la opción “Eliminar ruta” se ejecuta el escenario alterno A3 Eliminar ruta.

Flujos alternos

A1 Nueva ruta.

El usuario introduce los siguientes datos para la nueva ruta:

- Nombre de la ruta.
- Hora de inicio del servicio de transporte.
- Hora de fin del servicio de transporte.
- Define las estaciones de la ruta. (Ver escenario de uso 3.2 Administración de estaciones de una ruta)
- Selecciona de una lista de tipos de vehículos de transporte, aquél que se utilizará para el servicio en la ruta. Si el usuario desea definir un nuevo tipo de vehículo, selecciona la opción “Ver catálogo” (Ver escenario de uso 3.3 Administración de tipos de vehículos).

Si el usuario selecciona la opción Aceptar, el sistema hace la siguiente verificación de la información:

- Si hace falta el nombre de la ruta, aparece el mensaje: “Debe introducir un nombre para la ruta”.
- Si el nombre de la ruta ya existe, aparece el mensaje: “Ya existe una ruta con ese nombre”.
- Si hace falta la hora de inicio de servicio, aparece el mensaje: “La hora de inicio del servicio no tiene un valor correcto”.
- Si hace falta la hora de fin de servicio, aparece el mensaje: “La hora de fin del servicio no tiene un valor correcto”.
- Si la hora de fin de servicio es igual o menor a la hora de inicio, aparece el mensaje: “La hora de fin del servicio no puede ser igual o menor a la hora de inicio”.
- Si el usuario no definió estaciones para la ruta, aparece el mensaje: “Debe introducir las estaciones de la ruta”.
- Si el usuario no especificó un tipo de vehículo para la ruta, aparece el mensaje: “Debe especificar un tipo de vehículo”.

Si toda la información anterior es correcta, el sistema almacena los datos de la nueva ruta.

Si el usuario selecciona la opción “Cancelar”, los datos de la nueva ruta se descartan.

El escenario de uso continúa al inicio del flujo principal.

A2 Abrir ruta.

El sistema muestra los datos de la ruta seleccionada:

- Datos de la ruta:
  - Número de estaciones
  - Longitud de la ruta
  - Horario de servicio
  - Tipo de vehículo usado
- Lista de temporadas definidas para la ruta:

    - Nombre de la temporada
    - Fecha de inicio de la temporada
    - Fecha de fin de la temporada
- Lista de tipos de día para cada temporada:
    - Nombre del tipo de día
    - Descripción del tipo de día
    - Días
- Lista de secciones para cada tipo de día:
    - Nombre de la sección
    - Descripción de la sección
    - Horario de la sección
    - Frecuencia de paso para esa sección

Si el usuario escoge la opción “Estaciones”, se ejecuta el escenario de uso 3.2 Administración de estaciones de una ruta.
Si el usuario escoge la opción “Vehículo”, se ejecuta el escenario alterno A4 Cambio del tipo de vehículo de una ruta.
Si el usuario escoge la opción “Nuevo Aforo”, se ejecuta el escenario de uso 3.4 Nuevo Aforo.
Si el usuario escoge la opción “Gráfica de la ruta”, se ejecuta el escenario de uso 3.5 Gráfica de demandas de servicio de la ruta.
Si el usuario escoge la opción “Gráfica de la temporada”, se ejecuta el escenario de uso 3.6 Gráfica de demandas de servicio de la temporada.
Si el usuario escoge la opción “Gráfica del tipo de día”, se ejecuta el escenario de uso 3.7 Gráfica de demandas de servicio del tipo de día
Si el usuario escoge la opción “Gráfica de la sección”, se ejecuta el escenario de uso 3.8 Gráfica de demandas de servicio de la sección.
Si el usuario escoge la opción “Horario”, se ejecuta el escenario de uso 3.9 Generar de un tipo de día.

A3 Eliminar ruta
Aparece el mensaje: “¿Está seguro que desea eliminar la ruta [nombre de la ruta]? Esto eliminará también la demás información de la ruta (aforos, temporadas, tipos de día y secciones)”

En caso de que el usuario seleccione la opción “Sí”, el sistema elimina los datos de la ruta.
En caso de que el usuario seleccione la opción “No”, la eliminación no se realiza.
El escenario de uso continúa al inicio del flujo principal.

A4 Cambio del tipo de vehículo de una ruta.
El sistema muestra una lista con los tipos de vehículos definidos.

Si el usuario desea definir un nuevo tipo de vehículo, selecciona la opción “Ver catálogo” (Ver escenario de uso 3.3 Administración de tipos de vehículos).

El usuario escoge un tipo de vehículo de la lista.
El sistema muestra los datos de capacidad del tipo de vehículo: número de asientos y pasajeros de pie.

Si el usuario escoge la opción “Aceptar” aparece el mensaje “¿Está seguro de que desea cambiar el tipo de vehículo de la ruta?”

Si el usuario escoge la opción “Sí”, el sistema cambia el tipo de vehículo de la ruta al tipo de vehículo escogido.

Si el usuario escoge la opción “No”, el sistema no hace el cambio de tipo de vehículo.

El escenario de uso continúa al inicio del flujo principal.

Post-Condiciones

El usuario crea, modifica o elimina satisfactoriamente una ruta.

8.7.2 Administración de estaciones de una ruta

Descripción y prioridad

El usuario crea o elimina las estaciones de una ruta o modifica una estación existente.
La prioridad de este caso de uso es indispensable.

Flujo de eventos

Precondiciones

El usuario ha ingresado al sistema.

Flujo básico

El sistema muestra una lista con las estaciones de una ruta:

- Número de estación
- Nombre de la estación
- Distancia a la terminal de salida

Si el usuario escoge la opción “Agregar” se ejecuta el escenario de uso A1 Agregar estación.

Si el usuario escoge la opción “Modificar” se ejecuta el escenario de uso A2 Modificar estación.

Si el usuario escoge la opción “Eliminar” se ejecuta el escenario de uso A3 Eliminar estación.

En caso de que el usuario seleccione la opción “Aceptar”, el sistema actualiza la información de las estaciones de la ruta, con la información presente en la lista.

En caso de que el usuario seleccione la opción “Cancelar”, el escenario de uso termina.

Flujos alternos

A1 Agregar estación.

El usuario introduce los siguientes datos para la nueva estación:

- Nombre
- Distancia desde la estación de salida, en kilómetros

Si el usuario selecciona la opción “Aceptar”, el sistema hace la siguiente verificación de la información:

- Si hace falta el nombre de la estación, aparece el mensaje: “Debe introducir un nombre de la estación”.
- Si hace falta la distancia a la estación de salida, aparece el mensaje: “Debe introducir la distancia a la terminal de salida”.
- Si la distancia a la estación de salida tiene un valor no válido, aparece el mensaje: “La distancia no tiene un valor correcto”.

Si toda la información anterior es correcta, el sistema almacena los datos de la nueva estación, ordenando todas las estaciones respecto a la distancia a partir de la estación de salida.

Si el usuario selecciona la opción Cancelar, los datos de la nueva estación se descartan.

El escenario de uso continúa al inicio del flujo principal.

A2 Modificar estación
Aparece el mensaje de confirmación "¿Está seguro de que desea modificar los datos de la estación?"
Si el usuario escoge la opción "No", se cancela la modificación.
Si el usuario escoge la opción "Sí", modifica uno o varios de los siguientes atributos del tipo de vehículo:

- Nombre
- Distancia desde la estación de salida, en kilómetros

En caso de que el usuario seleccione la opción de Aceptar, el sistema hace la siguiente verificación de la información:

- Si hace falta el nombre de la estación, aparece el mensaje: "Debe introducir un nombre de la estación".
- Si hace falta la distancia a la estación de salida, aparece el mensaje: "Debe introducir la distancia a la terminal de salida".
- Si la distancia a la estación de salida tiene un valor no válido, aparece el mensaje: "La distancia no tiene un valor correcto".

Si toda la información anterior es correcta, el sistema almacena los datos de la nueva estación, ordenando todas las estaciones respecto a la distancia a partir de la estación de salida.

Si el usuario selecciona la opción "Cancelar", los datos modificados se descartan.

El caso de uso continúa al inicio del flujo principal.

A3 Eliminar estación
Aparece el mensaje "¿Está seguro de que desea eliminar la estación [número de estación]?"

Si el usuario escoge la opción "Sí", el sistema elimina los datos de la estación.
Si el usuario escoge la opción "No", no se realiza la eliminación.
El caso de uso continúa al inicio del flujo principal.

Post-Condiciones

El usuario crea, modifica o elimina satisfactoriamente la información de las estaciones de una ruta.

8.7.3 Administración de tipos de vehículos

Descripción y prioridad

El usuario crea o elimina un tipo de vehículo o modifica uno existente.
La prioridad de este caso de uso es indispensable.

Flujo de eventos

Precondiciones

El usuario ha ingresado al sistema.

Flujo básico

El sistema muestra una lista con los tipos de vehículos definidos:

- Descripción
- Número de asientos
- Número de pasajeros de pie

Si el usuario escoge la opción “Agregar” se ejecuta el escenario de uso A1 Agregar tipo de vehículo.

Si el usuario escoge la opción “Modificar” se ejecuta el escenario de uso A2 Modificar tipo de vehículo.

Si el usuario escoge la opción “Eliminar” se ejecuta el escenario de uso A3 Eliminar tipo de vehículo.

Si el usuario escoge la opción “Cerrar”, el escenario de uso termina.

Flujos alternos

A1 Agregar tipo de vehículo.

El usuario introduce los siguientes datos para el nuevo tipo de vehículo:

- Descripción
- Número de asientos
- Número de pasajeros de pie

Si el usuario selecciona la opción Aceptar, el sistema hace la siguiente verificación de la información:

- Si hace falta el nombre del tipo de vehículo, aparece el mensaje: “Debe introducir un nombre para el tipo de vehículo”.
- Si el nombre del tipo de vehículo ya existe, aparece el mensaje: “Ya existe un vehículo con esa descripción”.
- Si hace falta el número de asientos, aparece el mensaje: “El número de asientos no tiene un valor correcto”.
- Si hace falta el número de pasajeros de pie, aparece el mensaje: “El número de pasajeros de pie no tiene un valor correcto”.

Si toda la información anterior es correcta, el sistema almacena los datos de la nueva ruta.

Si el usuario selecciona la opción Cancelar, los datos de la nueva ruta se descartan.

El escenario de uso continúa al inicio del flujo principal.

A2 Modificar tipo de vehículo
El usuario modifica uno o varios de los siguientes atributos del tipo de vehículo:

- Descripción
- Número de asientos
- Número de pasajeros de pie

Si el usuario escoge la opción “Aceptar”, el sistema modifica los datos existentes por los capturados.

Si el usuario escoge la opción “Cancelar”, no se hace la modificación de los datos.

El caso de uso continúa al inicio del flujo principal.

A3 Eliminar
Aparece el mensaje “¿Está seguro de que desea eliminar el tipo de vehículo [descripción]?

Si el usuario escoge la opción “Sí”, el sistema elimina los datos del tipo de vehículo.
Si el usuario escoge la opción “No”, no se realiza la eliminación.
El caso de uso continúa al inicio del flujo principal.

Post-Condiciones

El usuario crea, modifica o elimina satisfactoriamente la información de uno o más componentes.

8.7.4 Nuevo aforo

Descripción y prioridad

El usuario introduce al sistema los datos de un aforo o muestreo hecho en una ruta.
La prioridad de este caso de uso es indispensable.

Flujo de eventos

Precondiciones

El usuario ha ingresado al sistema.

Flujo básico

El sistema muestra una tabla vacía con las siguientes columnas:

- Número de estación
- Nombre de la estación
- Ascensos
- Descensos
- Faltantes
- Hora de llegada a la estación
- Hora de salida de la estación

El usuario introduce los datos del muestreo en las columnas correspondientes, con los siguientes formatos:

- Ascensos, descensos y faltantes: números enteros.
- Hora de llegada a la estación y hora de salida de la estación: hora con formato de 00:00 a 23:59.

El usuario introduce un identificador de la unidad de transporte en la que se hizo el muestreo y la fecha en que ésta se realizó.

Si el usuario escoge la opción “Aceptar”, el sistema hace la siguiente verificación de la información:

- Si hace falta el identificador de la unidad de transporte, aparece el mensaje: “Hace falta la identificación de la unidad”.
- Si hace falta algún dato de hora de llegada, aparece el mensaje: “Hacen falta datos en las horas de llegada”.
- Si hace falta algún dato de hora de salida, aparece el mensaje: “Hacen falta datos en las horas de salida”.
- Si hace falta un número de ascensos, descensos o faltantes o tienen un valor no válido, aparece el mensaje: “Error en los datos de Ascensos, Descensos o Faltantes. Verificar que sean números enteros”.

Si toda la información anterior es correcta, el sistema almacena los datos introducidos.

Si el usuario escoge la opción “Cancelar”, el escenario de uso termina.

Flujos alternos

Ninguno.

Post-Condiciones

El usuario captura los datos de un aforo o muestreo de una ruta.

8.7.5 Gráfica de demandas de servicio de una ruta

Descripción y prioridad

El sistema muestra las gráficas de demanda de servicio de una ruta.
La prioridad de este caso de uso es indispensable.

Flujo de eventos

Precondiciones

El usuario ha ingresado al sistema.

Flujo básico

El usuario escoge una de las siguientes opciones de graficación de la demanda de servicio:

- Por día
- Por semana
- Por quincena
- Por mes

Si el usuario escoge la opción “Aceptar”, el sistema muestra una gráfica con las demandas máximas y medias, por día, semana, quincena o mes, según lo haya escogido. El sistema también muestra los datos en forma de tabla.

Si el usuario escoge la opción “Cancelar”, el escenario de uso termina.

Flujos alternos

Ninguno.

Post-Condiciones

El sistema muestra la gráfica con las demandas de servicio de una ruta.

8.7.6 Gráfica de demandas de servicio de una temporada

Descripción y prioridad

El sistema muestra las gráficas de demanda de servicio de una temporada.
La prioridad de este caso de uso es indispensable.

Flujo de eventos

Precondiciones

El usuario ha ingresado al sistema.

Flujo básico

El usuario escoge una de las siguientes opciones de graficación de la demanda de servicio:

- Por día
- Por semana
- Por quincena
- Por mes

Si el usuario escoge la opción “Aceptar”, el sistema muestra una gráfica con las demandas máxima y media, por día, semana, quincena o mes, según lo haya escogido. El sistema también muestra los datos en forma de tabla.

Si el usuario escoge la opción “Cancelar”, el escenario de uso termina.

Flujos alternos

Ninguno.

Post-Condiciones

El sistema muestra la gráfica con las demandas de servicio de una ruta.

8.7.7 Gráfica de demandas de servicio de un tipo de día

Descripción y prioridad

El sistema muestra las gráficas de demanda de servicio de un tipo de día de una temporada. La prioridad de este caso de uso es indispensable.

Flujo de eventos

Precondiciones

El usuario ha ingresado al sistema.

Flujo básico

El usuario escoge una de las siguientes opciones de graficación de la demanda de servicio:

- Cada 15 minutos
- Cada 20 minutos
- Cada 30 minutos
- Cada 60 minutos

Si el usuario escoge la opción “Aceptar”, el sistema muestra una gráfica con las demandas máxima y media, cada 15, 20, 30 ó 60 minutos, según lo haya escogido. El sistema también muestra los datos en forma de tabla.

Si el usuario escoge la opción “Cancelar”, el escenario de uso termina.

Flujos alternos

Ninguno.

Post-Condiciones

El sistema muestra la gráfica con las demandas de servicio de una ruta.

8.7.8 Gráfica de demandas de servicio de una sección

Descripción y prioridad

El sistema muestra las gráficas de demanda de servicio de un tipo de día de una sección de un tipo de día.

La prioridad de este caso de uso es indispensable.

Flujo de eventos

Precondiciones

El usuario ha ingresado al sistema.

Debe haberse generado un horario para el tipo de día al que pertenece la sección. (Ver caso de uso 3.9 Generar horario de un tipo de día)

Flujo básico

Si el usuario escoge la opción “Aceptar”, el sistema muestra una gráfica con las demandas máxima y media, para la sección. El sistema también muestra los datos en forma de tabla.

Si el usuario escoge la opción “Cancelar”, el escenario de uso termina.

Flujos alternos

Ninguno.

Post-Condiciones

El sistema muestra la gráfica con las demandas de servicio de una sección.

8.7.9 Generar horario de un tipo de día

Descripción y prioridad

En este escenario de uso el sistema utiliza los datos de demanda obtenidos de los aforos o muestreos con los que se cuenta, las características de los vehículos utilizados para dar el servicio de transporte y las características de ocupación deseadas para el vehículo, para proveer un horario para un tipo de día, el cual muestra la frecuencia de paso para cada sección. Estas frecuencias de paso aseguran la satisfacción de la demanda, minimizando el uso de las unidades lo más posible.

La prioridad de este caso de uso es indispensable.

Flujo de eventos

Precondiciones

Se ha escogido un tipo de día de una temporada de una ruta.

Flujo básico

El sistema consulta los datos de capacidad del vehículo usado en la ruta, las fechas de inicio y fin de la temporada y las secciones en que está dividido el tipo de día seleccionado.
Para cada sección, el sistema calcula el tiempo de recorrido de la ruta y la máxima demanda existente.
El sistema muestra los datos calculados y consultados en los pasos anteriores.
El usuario introduce para cada sección el factor de ocupación deseado para las unidades de transporte.
El usuario introduce para cada sección el valor de tiempo de terminal.

Si el usuario escoge la opción “Aceptar”, el sistema hace la siguiente verificación de la información:

- Si hace falta el factor de ocupación, aparece el mensaje: “Hace falta datos del factor de ocupación”.
- Si hace falta algún dato de hora de llegada, aparece el mensaje: “Hace falta datos del tiempo de terminal”.

Si toda la información anterior es válida, el sistema calcula con los datos anteriores, para cada sección, la cantidad requerida de vehículos, la frecuencia de paso, el tiempo de recorrido de toda la ruta y la velocidad comercial de las unidades de transporte.

Si el usuario escoge la opción “Guardar”, se almacenan los datos calculados para cada sección, para poder generar una gráfica de sección (Ver escenario de uso 3.8 Gráfica de demandas de servicio de una sección).

El usuario puede introducir otros valores de factor de ocupación (porcentaje de ocupación de las unidades de transporte) y de tiempo de terminal (tiempo que una unidad de transporte que ha llegado a la estación terminal tarda en comenzar una nueva corrida) para cada sección, después de lo cual el sistema volverá a verificar la validez de los datos y volverá a calcular los vehículos requeridos, la frecuencia de paso, el tiempo de recorrido y la velocidad comercial, como se explicó anteriormente. Si después de calcular de nuevo los valores el usuario escoge la opción “Guardar”, el sistema reemplazará los datos actuales por lo nuevos.

Si el usuario escoge la opción “Cerrar”, el escenario de uso termina.

Post-Condiciones

El sistema ha calculado el horario óptimo para cada sección de un tipo de día.

8.8 Interfaces

En esta sección se especifican los requerimientos de relacionados con las interfaces del sistema. De acuerdo a los requerimientos especificados y a la información proporcionada por el usuario, se identificaron las siguientes interfaces:

- Interfaces de software. El acceso a la base de datos deberá estar disponible siempre que se esté utilizando el sistema. El manejador de bases de datos deberá estar activo siempre que se esté utilizando el sistema.

8.9 Requerimientos no funcionales

Esta sección corresponde a la especificación de requerimientos de calidad o no funcionales. Este tipo de requerimientos trata con aquellas características o atributos de calidad que definen el comportamiento del sistema. Para este sistema se identificaron como requerimientos no funcionales los siguientes:

- Usabilidad. El sistema deberá ser fácil de comprender y utilizar. Para esto deberá ser claro en la manera en que se utiliza para almacenar los datos de los muestreos, crear y administrar las rutas y sus estaciones, temporadas, tipos de día y secciones, los tipos de vehículos y la creación de horarios de servicio.

- Integridad de los datos. Se refiere al aseguramiento de que la información existente en el sistema es correcta. El sistema deberá asegurar la integridad de los datos, mediante la verificación de la validez de los datos y la recuperación de errores o fallos del sistema.

- Desempeño. El sistema deberá ser rápido al ser iniciado y en la ejecución de las operaciones que conforman las funcionalidades.

8.10 Restricciones de tecnología

El sistema será implementado utilizando la siguiente tecnología:

- Manejador de Bases de Datos SQL Server.
- Lenguaje de programación Delphi (Object Pascal).

8.11 Modelo de datos

Se identificaron las entidades de datos involucradas en el sistema y las relaciones entre ellas. Esto se muestra la figura 8.1 a continuación:

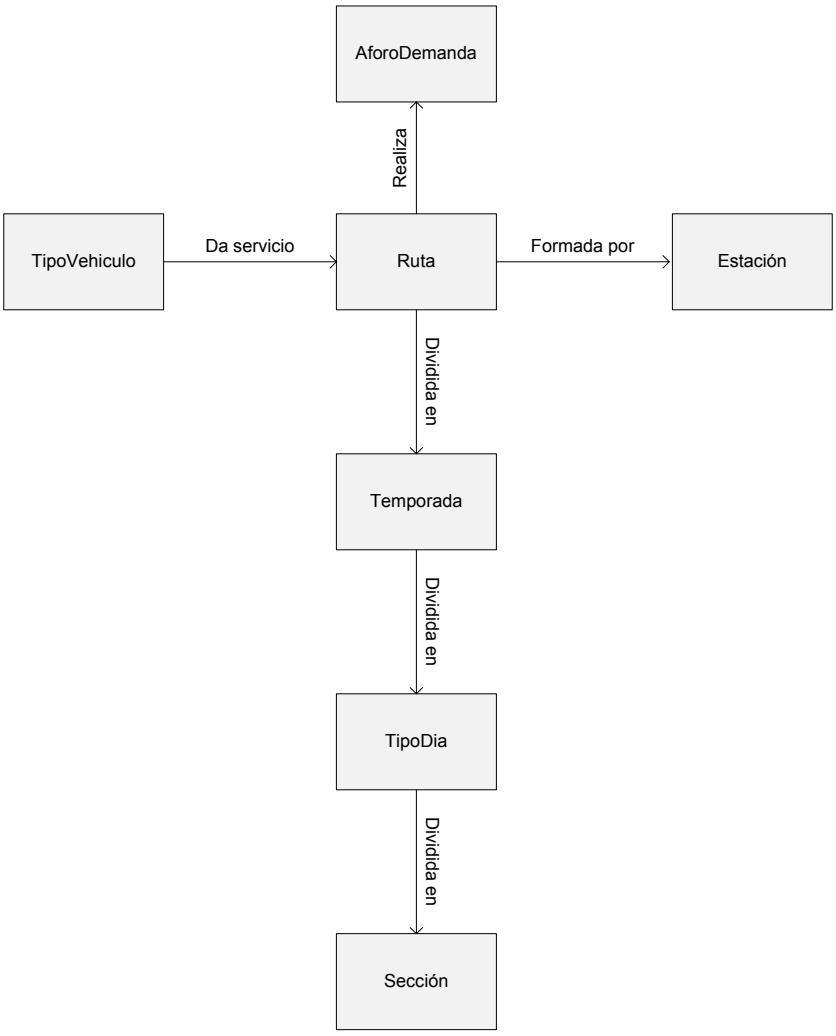


Fig 8.1 Modelo de datos

Una ruta de transporte está formada por estaciones. Un tipo de vehículo es usado en la ruta para dar servicio de transporte. En la ruta se realizan aforos de demanda para conocer la demanda de servicio existente. El servicio de transporte está dividido en temporadas, las temporadas están divididas en tipos de día (fin de semana, día hábil, etc.) y los tipos de día están divididos en secciones (mañana, tarde, etc.).

Para facilitar y mejorar la observación del comportamiento del servicio de transporte y de los cambios en las demandas de servicio, este se divide por temporadas, tipos de día y secciones.

Una temporada es un rango de tiempo definido por un día y un mes de inicio y por un día y mes de fin. Ejemplo: del 1 de enero al 1 de junio.

Un tipo de día es una clasificación de los días según su comportamiento de demanda o servicio. Ejemplo: fin de semana (sábado y domingo) o día hábil (lunes a viernes).

Una sección es una división de un día, según el comportamiento de demanda o servicio. Ejemplo: mañana (5:00 am a 12:00 pm), tarde (12:00 pm a 7:00 pm), etc.

8.12 Método para programación del servicio

La calidad de un servicio de transporte público es un concepto amplio que engloba varios aspectos, entre los que se incluyen consideraciones relativas a la comodidad y seguridad dentro de la unidad de transporte, los tiempos empleados en la realización del viaje y la conveniencia y existencia de infraestructura que apoye al servicio.

Sin embargo, al revisar el servicio que se presta en una ruta de transporte no es factible considerar un buen número de aspectos cualitativos, pero sí es factible tomar en cuenta las variables más importantes para el usuario. De esta manera, entre los principales parámetros relativos a la calidad del servicio y que se incluyen en el dimensionamiento de un servicio de transporte, se encuentran:

- capacidad del vehículo
- intervalo
- horarios de servicio
- la elaboración de itinerarios
- la determinación del tipo de vehículo

La cantidad de vehículos es el parámetro que mejor refleja el equilibrio entre la oferta y la demanda del transporte colectivo. Así, por ejemplo, en el caso de contar con más autobuses en servicio que los requeridos se produce un exceso de oferta y una ociosidad del equipo, lo cual conduce a un alto costo en la operación.

Por otra parte, si se cuenta con un parque vehicular por debajo de los requerimientos reales, se presenta una mala calidad del servicio que se traduce en molestias e inseguridad al usuario. Es por ello que tanto el área encargada de la operación dentro de la empresa como las autoridades correspondientes tendrán la difícil tarea de encontrar el balance adecuado entre la oferta y la demanda.

El conocimiento de las necesidades de la población y el uso de las técnicas de planeación del servicio y su dimensionamiento evitan el desperdicio de recursos, tanto humanos como económicos y contribuyen a lograr un ajuste racional dentro del sistema de transporte o la empresa misma.

El procedimiento general que se sigue para la programación del servicio consiste en el dimensionamiento del servicio, la preparación de itinerarios, la asignación de las jornadas de trabajo, entre otros aspectos, considerándose que el trazo de la ruta ya está definido.

8.12.1 Dimensionamiento de una ruta de transporte

El dimensionamiento de una ruta de transporte o la elaboración de su esquema de operación puede ser realizado siguiendo los procedimientos señalados a continuación.

8.12.2 Definición de elementos básicos

Intervalo

El intervalo (i) es la porción de tiempo, comúnmente expresada en minutos, entre dos salidas sucesivas de vehículos de transporte público en una ruta. El usuario está interesado en contar con un servicio con intervalos cortos para minimizar el tiempo de espera en la parada. Sin embargo, para un volumen de

pasajeros dado por hora, resulta más barato operar un número más pequeño de vehículos grandes que un número mayor de vehículos pequeños, por lo que el transportista está interesado en operar con vehículos de mayor capacidad a intervalos más grandes. Consecuentemente, los intervalos son determinados en un equilibrio entre el tiempo que espera el usuario en la parada y los costos de operación que afronta la empresa transportista.

El punto a lo largo de la ruta donde los intervalos mínimos posibles entre vehículos sucesivos son los mayores, determina el intervalo mínimo para toda la ruta. Por ello, el intervalo mínimo posible en una línea o ruta (i_{min}), se presenta en paradas con un gran número de ascensos/descensos de pasajeros, siendo el intervalo mayor de todos el crítico y por ende representa el intervalo mínimo posible en toda la ruta.

Frecuencia de servicio

La frecuencia (f) es el número de unidades que pasan en un punto dado en la ruta durante una hora (o cualquier periodo de tiempo considerado), siendo éste el inverso del intervalo. Ambos están relacionados por la expresión:

$$f = \frac{60}{i}$$

donde:

60 : factor de conversión de minutos a horas
f : frecuencia [vehículos/hora]
i : intervalo [minutos]

La frecuencia máxima de llegadas de vehículos (f_{max}) se determina por el intervalo mínimo como:

$$f_{max} = \frac{60}{i_{min}}$$

Capacidad vehicular

La capacidad vehicular (C_v) es el número total de espacios para pasajeros en el vehículo. Se calcula sumando el número de asientos más los espacios de pie. Esta definición es aceptable para el metro, autobuses urbanos y líneas de trolebuses, los cuales tienen generalmente longitudes de viaje promedio no demasiado grandes y una alta rotación de pasaje.

Volumen de pasajeros

El volumen de pasajeros (p) es el número de usuarios que pasan por un punto fijo durante una hora u otro periodo de tiempo específico. El volumen de pasajeros varía a lo largo de la ruta conforme a las variaciones de la hora del día, día de la semana y época del año.

Sección de máxima demanda (SMD)

Es la sección o punto dentro de la ruta (parada o estación) donde ocurre la máxima demanda de pasajeros a bordo de las unidades y establece el volumen de diseño de la ruta.

Volumen de diseño

El volumen de diseño (P) es el que se presenta en la sección de máxima demanda de una ruta y en consecuencia, el mayor volumen de cualquier parada o sección a lo largo de la ruta. Este volumen es el parámetro básico para determinar la capacidad de línea que debe ofrecerse.

Capacidad de línea ofrecida

La capacidad de línea (C) es el número total de espacios para pasajeros ofrecidos en un punto fijo de una ruta durante una hora. La capacidad de línea es básica para la planeación y diseño del transporte público y es resultado del producto de la frecuencia y la capacidad vehicular. Naturalmente, se debe proveer de una capacidad igual o mayor que el volumen de diseño P.

$$C = f \times C_v$$

donde:

C : Capacidad de línea [pasajeros/hora]
F : Frecuencia [vehículos/hora]
C_v : Capacidad del vehículo [pasajeros/vehículo]

Capacidad de línea máxima

La capacidad de línea máxima (C_{max}) es el número máximo de pasajeros por hora que una línea puede llevar con el intervalo mínimo posible. Este parámetro se obtiene como el producto de la frecuencia máxima y la capacidad del vehículo:

$$C_{max} = f_{max} \times C_v = \frac{60 \times C_v}{i_{min}}$$

Tiempo de recorrido

El tiempo de recorrido (t_r) es el intervalo de tiempo programado entre la salida de un vehículo de una terminal y su llegada a la terminal opuesta en la ruta o en su caso, a la misma terminal de partida. El tiempo de recorrido se expresa usualmente en minutos.

Velocidad de operación

La velocidad de operación (V_o) es la velocidad promedio de una unidad de transporte, en la cual se incluye el tiempo de parada en estaciones o paradas así como las demoras esperadas por razones de tránsito. Se calcula como la relación entre la longitud de recorrido total (L) en kilómetros y el tiempo que tarda la unidad en recorrer dicha longitud, en minutos:

$$V_o = \frac{60 \times L}{t_r}$$

donde:

V_o : Velocidad de operación [km/h]
L : Longitud de la ruta [km]
t_r : Tiempo de recorrido [min]

Tiempo de terminal

Es el tiempo adicional (t_t) que un vehículo espera en la terminal o en el cierre de circuito, requerido para el ascenso y descenso normal de pasajeros. Su propósito es contar con tiempo para dar la vuelta al vehículo en la terminal opuesta o cambio en la cabina de mando, para dar un descanso al operador o para permitir los ajustes necesarios en el horario. Este tiempo permite además de las consideraciones anteriores, mantener un intervalo uniforme y/o recuperar las demoras en las que se haya incurrido.

Por ello, el tiempo de terminal generalmente está determinado en función de los descansos de los operadores, del tiempo requerido para efectuar las actividades de chequeo por parte del despachador y de la propensión a demoras en la ruta. Normalmente, los tiempos de descanso y de recuperación de demoras están en función del tiempo que la unidad está en operación, por lo que el tiempo de terminal

para sistemas de superficie se expresa a través de un cociente γ que relaciona el tiempo terminal y el de recorrido:

$$\gamma = \frac{t_t}{t_r}$$

El rango para este coeficiente se ubica entre 0.12 y 0.18, mismo que depende de las condiciones de trabajo, del tránsito, de las variaciones en el volumen de pasajeros y otros factores locales. En ciertas líneas y durante ciertos periodos del día donde el congestionamiento es serio, el tiempo de recorrido varía considerablemente por lo que en algunos casos se permiten tiempos terminales mayores, lográndose con ello que la hora de salida del viaje de regreso pueda mantenerse y se puedan conservar los horarios aún cuando sucedan demoras moderadas.

Tiempo de ciclo o vuelta

El tiempo de ciclo (t_c) es el tiempo total de viaje redondo para una unidad de transporte, esto es, el tiempo que tarda en volver a pasar la misma unidad por un punto determinado, el cual se expresa normalmente en minutos. Este tiempo está dado, en el caso de que sus tiempos de recorrido y terminal sean iguales en cada dirección, por:

$$t_c = 2(t_r + t_t)$$

o en el caso de que se trate de una ruta donde la terminal de salida sea la misma que la terminal final de llegada, sin terminales opuestas:

$$t_c = (t_r + t_t)$$

Velocidad comercial

Es la velocidad promedio (V_c) que una unidad de transporte mantiene para dar una vuelta completa. Para rutas con dos terminales:

$$V_c = \frac{120 \times L}{t_c}$$

Para rutas de una sola terminal de salida y llegada:

$$V_c = \frac{60 \times L}{t_c}$$

donde:

V_c : Velocidad comercial [km/h]
t_c : Tiempo de ciclo [min]

La velocidad comercial determina directamente (junto con el intervalo) el tamaño requerido del parque vehicular y los costos de operación. La velocidad comercial siempre será menor que la velocidad de operación ya que la primera incluye los tiempos terminales, por lo que:

$$V_c < V_o$$

Tamaño del parque vehicular

El tamaño del parque vehicular (N_p) es el número total de unidades que operan en una ruta y la suma de éstas representa el parque total con que cuenta la empresa de transporte. El tamaño del parque vehicular

consiste del número de vehículos requeridos para el servicio durante la hora de máxima demanda en todas las rutas (N), los vehículos en reserva (N_r) y los vehículos que están en mantenimiento y reparación (N_m). Este valor se expresa por la siguiente fórmula:

$$N_p = N + N_r + N_m$$

8.12.3 Criterios para determinar los elementos básicos de dimensionamiento

Los criterios básicos están enfocados a los aspectos de intervalos, factores de ocupación, tamaño del parque vehicular y la capacidad vehicular, principalmente. A continuación se presentan los criterios más importantes a considerar:

Intervalos

Los requerimientos para determinar los intervalos son los siguientes:

- Proveer de una capacidad adecuada que permita cumplir con la demanda de usuarios.
- Ofrecer cierta frecuencia mínima con el fin de mantener un servicio.

La frecuencia que dará la capacidad necesaria para cumplir con la demanda se obtiene dividiendo la carga en la sección de máxima demanda entre el número promedio de pasajeros asignados a cada vehículo a través de la selección de un valor para el factor de ocupación (α). Esta frecuencia se expresa como:

$$f = \frac{P}{\alpha \cdot C_v}$$

o bien se expresa el intervalo como:

$$i = \frac{60 \cdot \alpha \cdot C_v}{P}$$

Para facilitar la memorización del intervalo y la elaboración de horarios, es recomendable que los intervalos mayores de 6 minutos se repitan cada hora. Por lo tanto, el intervalo debe ser divisor de 60, esto es:

i = 6, 7.5, 10, 12, 15, 20 y 30

Al utilizar intervalos mayores de 30 minutos es recomendable el manejo de valores de 40, 45 y 60 minutos por lo que el intervalo debe ser redondeado hacia abajo al valor más cercano a estos valores.

En el caso de las horas de baja demanda u horas valle, durante los fines de semana o en aquellas rutas con poca demanda, normalmente se maneja una frecuencia mínima requerida para mantener el servicio y por ello las empresas fijan un intervalo mínimo. Este intervalo es conocido como intervalo mínimo de servicio (i_s) el cual, en zonas urbanas, no debe ser mayor que una hora y es recomendable que no sea mayor a los 30 minutos.

Si se calcula el intervalo a las horas de mínima demanda en función de las cargas y factores de ocupación con la meta de alcanzar cargas en la unidad iguales a la capacidad de los asientos, se logran ahorros en cuanto a la cantidad de servicio ofrecido (vehículos-kilómetros) y operado (vehículos-horas). Sin embargo, el número de unidades se verá reducido y los intervalos de espera del usuario se verán incrementados. Es por ello que este intervalo mínimo de servicio depende de la posibilidad financiera de la empresa de prestar el servicio y del tamaño de la fuerza laboral existente durante las horas de máxima demanda y con la que se cuenta durante las horas valle.

Factor de ocupación

El factor de ocupación (α) es el cociente del número de pasajeros en un vehículo entre la capacidad del vehículo. Un valor alto de α indica que la unidad de transporte está saturada, haciendo factible que algunas unidades no cuenten con la capacidad suficiente para recoger a todos los usuarios que esperan en la parada (remanente).

El valor de este factor influye en las siguientes características de la operación del transporte público:

- El nivel de comodidad del usuario. Un valor alto de α trae como resultado un número considerable de usuarios de pie y la sobrecarga del vehículo.

- Costos de operación. El uso de un valor alto de α implica un menor número de unidades para transportar un número dado de usuarios que en el caso de utilizar un valor bajo de α. A su vez, una menor cantidad de unidades operando da en consecuencia una menor frecuencia y con ello mayores tiempos de espera al usuario. Finalmente, un valor alto de α resulta en un mayor tiempo de ascenso/descenso, con lo cual se reduce la velocidad de operación y esto afecta directamente a los costos de operación.

La selección de un valor de α debe ser realizada de tal forma que se logre un balance entre los factores antes mencionados. El operador o empresario al determinar el valor de α considera también los siguientes factores que influyen en la comodidad del usuario y en los costos de operación.

Condiciones que requieren un valor de α bajo	Condiciones que requieren un valor de α alto
Variaciones grandes en el volumen de usuarios	Volumen más o menos constante de usuarios
Se desea una relación asientos/de pie mayor (pocos pasajeros de pie)	Se desea una relación asientos/de pie menor (muchos pasajeros de pie)
Longitud promedio grande de recorrido del pasajero	Longitud promedio pequeña de recorrido del pasajero
Alto porcentaje de usuarios de la tercera edad	Alto porcentaje de niños en edad escolar

Es usual que el operador determine un valor para α para cada periodo de programación de horarios (mayor para las horas pico, menor para las horas valle) calculando primeramente el cociente del número de asientos y la capacidad total del vehículo C_s/C_v y a partir de los valores encontrados se utilizan los siguientes lineamentos:

- El valor mínimo de α debe ser un poco menor que la relación C_s/C_v. Este valor garantiza asientos a todos los usuarios excepto por algunos periodos cortos.
- El valor máximo de α recomendable es de 0.9 el cual debe ser utilizado para horas de máxima demanda en el caso de contar con una sección de máxima demanda corta y en donde volumen de pasajeros no varía significativamente de un día a otro.

Tamaño del parque vehicular y la capacidad del vehículo

Para un volumen dado de pasajeros en una línea, el servicio puede ser proporcionado por una cantidad pequeña de unidades de gran capacidad o bien, por una cantidad mayor de unidades de baja capacidad. La segunda combinación resulta en una mayor frecuencia, pero requiere una inversión y costos de operación mayores que la primera combinación.

Es saludable que una empresa de transporte realice un análisis detallado de los costos de operar determinado tipo de unidad, establezca las condiciones en que operará el equipo y evalúe la calidad del servicio que resultará del uso de cada tipo de vehículo antes de efectuar cualquier compra de unidades. Es fundamental que la empresa especifique y establezca las condiciones más importantes que deben cumplir las unidades y en función de estos resultados determine el kilometraje y vida útil esperada.

Al planear la compra de unidades nuevas, la empresa de transporte debe examinar los posibles compromisos entre un vehículo pequeño y uno grande. Si se compara las ventajas y desventajas que presentan los dos tipos de unidades, se tiene que:

- El costo de operación por vehículo-km es menor para las unidades de baja capacidad. Esto implica que por el mismo costo total de operación la empresa puede operar intervalos más cortos con unidades más pequeñas y por lo tanto atraer más pasajeros al ser sus tiempos de espera menores y contar con un servicio más frecuente.
- Los recorridos a través de zonas congestionadas son más rápidos y sencillos con unidades pequeñas.
- El costo total de adquisición y el costo de operación del parque vehicular de unidades pequeñas es mayor ya que se debe comprar y operar más de ellos para cubrir el volumen de pasajeros que se presenta a la hora de máxima demanda.

8.12.4 Ejemplo del dimensionamiento de una ruta

La naturaleza misma del procedimiento para dimensionar una ruta se facilita al mostrar los pasos que se deben seguir a partir de un ejemplo, el cual se desarrolla a continuación:

Recolección de la información requerida

Los parámetros principales que deben tenerse presente para el dimensionamiento de una ruta son los siguientes:

Longitud de la ruta en una dirección	L = 10 km
Tiempo de recorrido	t_r = 45 minutos hora de máxima demanda (HMD) t_r = 40 minutos hora valle (HV)
Volumen de diseño	P = 375 usuarios hora pico (HMD y en la SMD)
Capacidad del vehículo	C_v = 45 asientos+25 de pie = 70 espacios

Determinación de los factores operativos que inciden en la ruta.

Como primer paso se estima la velocidad a la que operarán las unidades dentro de la ruta, a partir de la siguiente ecuación:

$$V_o = \frac{60 \times L}{t_r}$$

Con lo que resultan las siguientes velocidades:

V_o = 13.3 km/h para la hora de máxima demanda (HMD)
V_o = 15.0 km/h para la hora valle (HV)

Se establece los valores de α, i_s y t_t, que se ajustarán durante el proceso de dimensionamiento de la ruta. Para nuestro ejemplo, estos valores iniciales son:

Factor de ocupación	α = 0.70
Intervalo mínimo de servicio	i_s = 15 minutos
Tiempo de terminal mínimo	t_t = 6 minutos

Determinación del intervalo

Se calcula el intervalo a partir de la siguiente ecuación:

$$i = \frac{60 \cdot \alpha \cdot Cv}{P} = \frac{60 \times 0.7 \times 70}{375} = 7.84$$

El valor del intervalo debe ser redondeado hacia abajo al valor práctico más cercano. Si el valor obtenido es mayor de seis minutos, es recomendable utilizar los siguientes valores: 7.5, 10, 12, 15, 20, 30, 40, 45 y 60. Con ello se logra que los tiempos de salida de las unidades se repitan cada hora, excepto para los intervalos de 40 y 45 minutos.

Por otra parte, el intervalo calculado debe ser comparado con el intervalo mínimo de servicio i_s para el periodo que se esté programando el servicio, seleccionándose el menor de los dos. Ya que en este caso el valor calculado del intervalo es de i = 7.84 y éste es más pequeño que is = 15 minutos, entonces el valor de 7.5 minutos es el que se considera como intervalo a la hora de máxima demanda y 15 minutos durante la hora valle. Naturalmente, el valor i_s puede estimarse mediante la fórmula en caso de conocer la carga o demanda durante la hora valle (hora de menor demanda).

Cálculo del tiempo de ciclo

El tiempo de ciclo se calcula a partir de la siguiente expresión:

$t_c = t_r + t_t$ para las rutas de una sola terminal de salida y llegada
$t_c = 2(t_r + t_t)$ para las rutas con dos terminales de salida y llegada

En este caso se trata de una ruta con dos terminales, por lo tanto:

$t_c = 2(45 + 6) = 102$ minutos para la HMD
$t_c = 2(40 + 6) = 92$ minutos para la HV

Determinación del tamaño del parque vehicular

El parque vehicular se determina mediante la aplicación de la siguiente expresión:

$$N = \frac{t_c}{i}$$

Ya que el parque vehicular N debe ser un valor entero, el resultado de la expresión anterior se redondea hacia arriba al siguiente número entero. Para el ejemplo:

$N_{HMD} = 13.6 = 14$ vehículos en HMD
$N_{HV} = 6.3 = 7$ vehículos en HV

A partir de estos nuevos resultados, se requiere ajustar el nuevo tiempo de ciclo a partir de los valores estimados del parque vehicular, lo que implica:

$t_c = N \cdot i$

$t_c = 7.5$ minutos HMD
$t_c = 12.5$ minutos HV

Finalmente, con los datos anteriores se calcula la velocidad comercial V_c:

$$V_c = \frac{120 \cdot L}{t_c}$$

El factor que multiplica a la longitud L es 60 para el caso de una ruta con una sola terminal de salida y llegada.

Vc_{HMD} = 11.4 km/h para la HMD
Vc_{HV} = 14.4 km/h para la HV

Los resultados anteriores permiten dimensionar la ruta, sintetizando los parámetros de dimensionamiento de la manera siguiente:

Concepto	Hora de máxima demanda	Hora valle
Intervalo	7.5 min	15 min
Tiempo de ciclo	105 min	105 min
Tiempo de terminal	7.5 min	12.5 min
Tamaño de la flota	14 veh	7 veh
Velocidad comercial	11.4 km/h	11.4 km/h
Eficiencia itinerario	0.86	0.76
t_t/t_o (γ)	0.08	0.16

Capítulo 9
Documento de diseño

9.1 Introducción

9.1.1 Propósito

Proveer una vista global y la información referente al diseño para el sistema de optimización de rutas de transporte, de manera que sirva como medio de comunicación entre los miembros del equipo de desarrollo, como referencia para la implementación. Este documento emplea diversas vistas o modelos para ilustrar diferentes aspectos del diseño.

9.1.2 A quién va dirigido

El documento sirve como medio de comunicación entre los integrantes del equipo de desarrollo. Además, sirve para mostrar al cliente y al usuario la estructura general del sistema y el comportamiento que tendrán las partes que lo conformarán.

9.1.3 Alcance del diseño

El diseño de software expresa información acerca de cómo se relacionan los elementos del sistema unos con otros. Es una abstracción de un sistema que suprime detalles de elementos que no afectan la manera en que son utilizados, relacionados o cómo interactúan con otros elementos.

9.2 Organización del documento de arquitectura de software

El documento ha sido organizado de la siguiente manera:

Primero se proporciona información acerca del propósito, organización y alcance del documento de diseño. Posteriormente se presenta las vistas de diseño.

9.3 Definiciones de vistas de diseño

Las representación del sistema se ha dividido en vistas de diseño, dependiendo de los elementos que serán mostrados. A continuación se da una descripción de los objetivos de interés de cada una de estas vistas.

- *Diseño de datos*.

Esta vista de diseño muestra con un alto nivel de abstracción, las entidades de datos que constituyen el sistema, para facilitar la creación de la base de datos.

- *Diseño arquitectónico.*

Esta vista de diseño muestra de forma general, el estilo arquitectónico elegido para la construcción del sistema, desde el punto de vista de flujo de datos y de control.

- *Diseño de interfaz.*

Esta vista de diseño muestra las pantallas, formularios, etc. que conforman la interfaz gráfica de usuario.

- *Diseño procedimental.*

Esta vista de diseño muestra en una forma bastante cercana a la implementación, las funciones, métodos y algoritmos que llevan a cabo las funciones del sistema.

9.4 Antecedentes de la arquitectura

9.4.1 Visión general del sistema

El sistema tiene como objetivo apoyar las actividades de la organización relacionadas con el suministro de servicio de transporte público. Mediante el uso del sistema, la organización contará con un medio confiable y rápido de introducir y administrar los datos de las rutas definidas, así como de los estudios de demanda hechos sobre las rutas, importantes para la creación de horarios óptimos de servicio.

Para el diseño y construcción del sistema se consideran los siguientes módulos funcionales (ver sección de requerimientos del sistema):

- Creación, modificación y eliminación de una ruta.
- Creación, modificación y eliminación de un catálogo de tipos de unidades de transporte.
- Creación de temporadas de servicio.
- Creación de tipos de día de servicio dentro de las temporadas.
- Creación de horarios de servicio dentro de los tipos de día.
- Captura de los datos de los muestreos.
- Generación de gráficas de demanda para una ruta, por temporada, tipo de día o sección de un día.
- Generación de horarios óptimos.

De igual forma, se han definido los siguientes atributos de calidad o requerimientos no funcionales (ver sección de requerimientos del sistema):

- Integridad de los datos: el sistema debe asegurar la integridad referencial y que los datos se mantienen en un estado correcto.
- Usabilidad: el sistema debe ser fácil de usar y aprender.
- Desempeño: el sistema debe realizar sus funciones en tiempos aceptables.

9.4.2 Objetivos

Un diseño de software es la organización de un sistema representada por sus componentes, las relaciones entre dichos componentes, las relaciones con el medio ambiente y con los principios que guían su diseño y su evolución. El diseño permite cumplir con la funcionalidad así como los atributos de calidad del sistema.

9.4.3 Requerimientos significativos

Como se mencionó con anterioridad, los atributos de calidad que influyeron en el diseño de la arquitectura y que representan los requerimientos significativos, fueron la integridad de los datos, la usabilidad y el desempeño. En lo que respecta a los requerimientos funcionales del sistema, estos se encuentran especificados en la sección de requerimientos del sistema.

9.5 Panorama de la solución

9.5.1 Enfoque arquitectónico

Se utilizó una combinación de dos enfoques de diseño para el sistema de optimización de rutas de transporte. Estos dos enfoques son el modelo centrado en datos y el modelo de llamada y retorno.

Las características del modelo centrado en datos son:

- Facilidad de modificación y mejora del sistema
- Facilidad de integración de las funcionalidades entre sí o de funcionalidades nuevas
- Se facilita una forma más estructurada y más segura de almacenar y manejar los datos

Las características del modelo de llamada y retorno son:

- Es una estructura clásica donde un programa principal llama a otros componentes del programa, los cuales pueden también llamar a otros componentes. El control se pasa y se devuelve entre los componentes de forma jerárquica

Utilizar estos dos estilos ayuda a cumplir con las características del sistema, ya que el modelo centrado en los datos asegura el manejo adecuado de los datos, requerido por el usuario respecto a la corrección e integridad de los datos, así como su almacenamiento más seguro. Respecto al modelo de llamada y retorno, en su modalidad de programa principal/subprograma, éste facilita la realización de las funciones definidas en cada escenario de uso, en el orden establecido en los mismos, colaborando a cumplir con los requerimientos de usabilidad y desempeño.

9.6 Vistas de diseño

9.6.1 Diseño de datos

En esta vista de diseño, se modela el sistema como un conjunto de entidades de datos y las relaciones entre ellos.

Con base en el modelo de datos definido en el documento de requerimientos, en los atributos de las entidades definidas, las características de sus atributos y las relaciones existentes entre éstas y buscando facilitar la creación de la base de datos, el diseño de datos obtenido se muestra en la figura 9.1 a continuación:

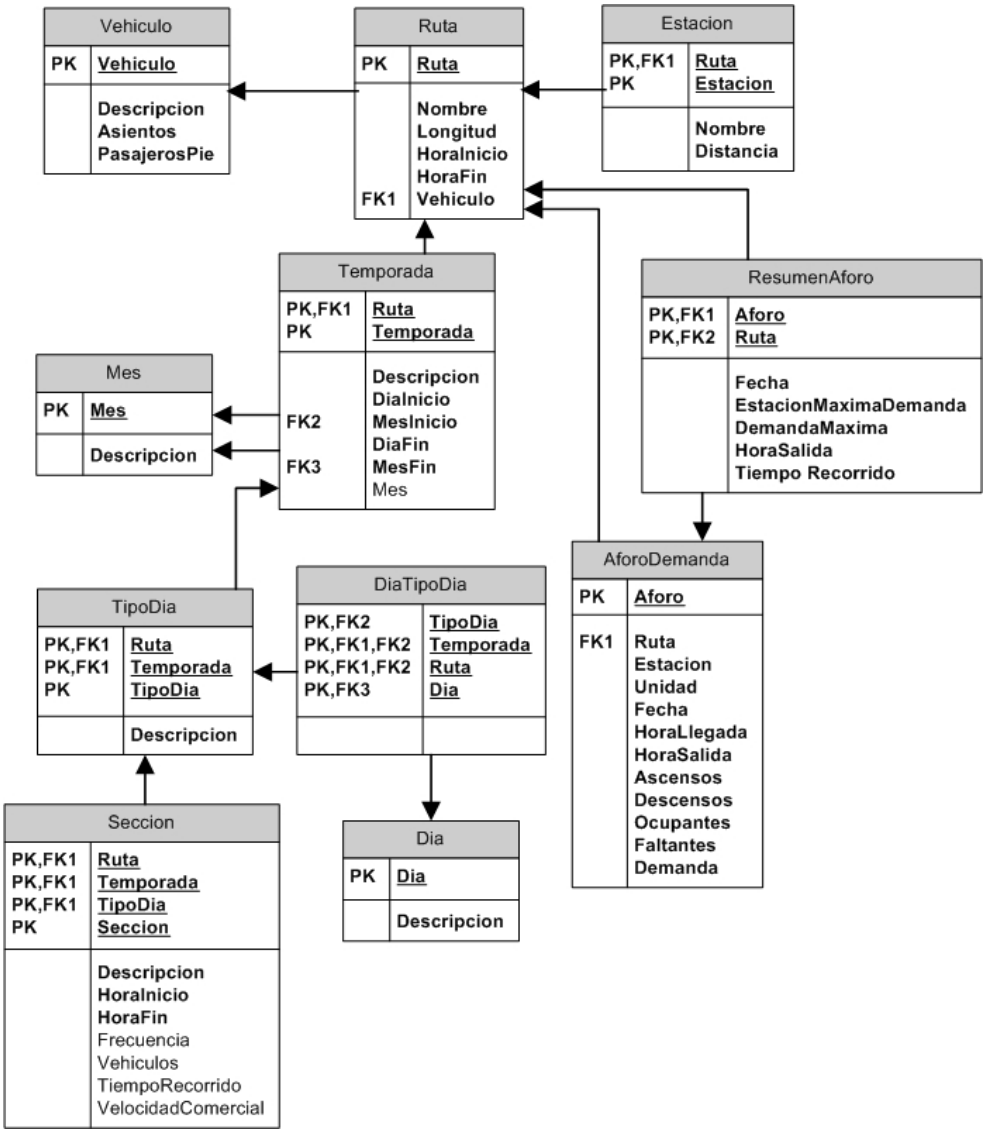


Fig. 9.1 Diseño de datos

Una ruta está formada por varias estaciones y la atiende un tipo de vehículo. El servicio de la ruta está dividido en temporadas definidas por una fecha de principio y una fecha de fin. Las temporadas están divididas en tipos de día, definidos por los días de la semana que corresponden a cada tipo y cada tipo de día está dividido en secciones del día, definidas por una hora de inicio de sección y una hora de fin de sección. Además, cada ruta tiene una cierta cantidad de aforos o muestreos de demanda.

A continuación se presenta el diccionario de datos, explicando las entidades y sus atributos.

Entidades

Nombre	Descripción
AforoDemanda	Contiene los aforos o muestreos de demanda introducidos al sistema.
Dia	Catálogo con los días de la semana.
DiaTipoDia	Entidad intermedia que relaciona las entidades Dia y TipoDia.
Estacion	Contiene las estaciones definidas para las rutas.
Mes	Catálogo con los meses del año.
ResumenAforo	Contiene los datos resumidos de los aforos, usados para la generación de horarios.
Ruta	Contiene las rutas definidas por el usuario.
Seccion	Contiene las secciones definidas por el usuario en que se ha dividido a los tipos de día.
Temporada	Contiene las temporadas definidas por el usuario en que se ha dividido el servicio de las rutas definidas.
TipoDia	Contiene los tipos de día definidos por el usuario en que se ha dividido el servicio de las temporadas.
Vehiculo	Catálogo con los tipos de vehículos definidos por el usuario.

Atributos de la entidad " AforoDemanda "

Nombre	Tipo de dato	Null/Not Null	PK	FK	Dato Obligatorio	Descripción
Aforo	smallint	NOT NULL	Sí	No	Sí	Clave única del aforo.
Ruta	smallint	NOT NULL	No	Sí	Sí	Clave de la ruta a la que pertenece el aforo. Referencia a Ruta.
Estacion	smallint	NOT NULL	No	No	Sí	Número de la estación en que se tomó el dato.
Unidad	varchar(15)	NOT NULL	No	No	Sí	Identificación de la unidad de transporte.
Fecha	smalldatetime	NOT NULL	No	No	Sí	Fecha en que se hizo el aforo.
HoraLlegada	smalldatetime	NOT NULL	No	No	Sí	Hora de llegada de la unidad de transporte a la estación.
HoraSalida	smalldatetime	NOT NULL	No	No	Sí	Hora de salida de la unidad de transporte de la estación.
Ascensos	smallint	NOT NULL	No	No	Sí	Número de ascensos a la unidad de transporte en la estación.
Descensos	smallint	NOT NULL	No	No	Sí	Número de descensos de la unidad de transporte en la estación.
Ocupantes	smallint	NOT NULL	No	No	Sí	Ocupantes de la unidad de transporte en la estación.
Faltantes	smallint	NOT NULL	No	No	Sí	Número de personas que no pudieron entrar a la unidad en la estación.
Demanda	smallint	NOT NULL	No	No	Sí	Demanda total de transporte en la estación.

Atributos de la entidad " Dia "

Nombre	Tipo de dato	Null/Not Null	PK	FK	Dato Obligatorio	Descripción
Dia	smallint	NOT NULL	Sí	No	Sí	Clave única del día.
Descripcion	varchar(10)	NOT NULL	No	No	Sí	Nombre del día.

Atributos de la entidad " DiaTipoDia "

Nombre	Tipo de dato	Null/Not Null	PK	FK	Dato Obligatorio	Descripción
TipoDia	smallint	NOT NULL	Sí	Sí	Sí	Clave del tipo de día. Referencia a TipoDia.
Temporada	smallint	NOT NULL	Sí	Sí	Sí	Clave de la temporada. Referencia a TipoDia.
Ruta	smallint	NOT NULL	Sí	Sí	Sí	Clave de la ruta. Referencia a TipoDia.
Dia	smallint	NOT NULL	Sí	Sí	Sí	Clave del día. Referencia a Dia.

Atributos de la entidad " Estacion "

Nombre	Tipo de dato	Null/Not Null	PK	FK	Dato Obligatorio	Descripción
Estacion	smallint	NOT NULL	Sí	No	Sí	Clave única de la estación.
Ruta	smallint	NOT NULL	Sí	Sí	Sí	Clave de la ruta. Referencia a Ruta.
Nombre	varchar(50)	NOT NULL	No	No	Sí	Nombre de la estación.
Distancia	real	NOT NULL	No	No	Sí	Distancia de la estación a la terminal de salida.

Atributos de la entidad " Mes "						
Nombre	**Tipo de dato**	**Null/Not Null**	**PK**	**FK**	**Dato Obligatorio**	**Descripción**
Mes	smallint	NOT NULL	Sí	No	Sí	Clave única del mes.
Descripcion	varchar(10)	NOT NULL	No	No	Sí	Nombre del mes.

Atributos de la entidad " ResumenAforo "						
Nombre	**Tipo de dato**	**Null/Not Null**	**PK**	**FK**	**Dato Obligatorio**	**Descripción**
Ruta	smallint	NOT NULL	Sí	Sí	Sí	Clave única de la ruta. Referencia a Ruta.
Aforo	smallint	NOT NULL	Sí	Sí	Sí	Clave única del aforo. Referencia a Aforo.
Fecha	smalldatetime	NOT NULL	No	No	Sí	Fecha en que se hizo el aforo.
EstacionMaximaDemanda	smallint	NOT NULL	No	No	Sí	Número de la estación con la máxima demanda en el aforo.
DemandaMaxima	smallint	NOT NULL	No	No	Sí	Demanda de transporte en la estación de máxima demanda.
HoraSalida	smalldatetime	NOT NULL	No	No	Sí	Hora de inicio del recorrido de la ruta por la unidad de transporte.
TiempoRecorrido	smallint	NOT NULL	No	No	Sí	Tiempo de recorrido de la ruta por la unidad de transporte, en minutos

Atributos de la entidad " Ruta "						
Nombre	**Tipo de dato**	**Null/Not Null**	**PK**	**FK**	**Dato Obligatorio**	**Descripción**
Ruta	smallint	NOT NULL	Sí	No	Sí	Clave única de la ruta.
Nombre	varchar(60)	NOT NULL	No	No	Sí	Nombre de la ruta.
Longitud	real	NOT NULL	No	No	Sí	Longitud de la ruta.
HoraInicio	smalldatetime	NOT NULL	No	No	Sí	Hora de inicio del servicio de transporte en la ruta.
HoraFin	smalldatetime	NOT NULL	No	No	Sí	Hora de fin del servicio de transporte en la ruta.
Vehiculo	smallint	NOT NULL	No	Sí	Sí	Tipo de vehículo usado en la ruta. Referencia a Vehículo.

Atributos de la entidad " Seccion "						
Nombre	**Tipo de dato**	**Null/Not Null**	**PK**	**FK**	**Dato Obligatorio**	**Descripción**
Ruta	smallint	NOT NULL	Sí	Sí	Sí	Clave de la ruta. Referencia a TipoDia.
Temporada	smallint	NOT NULL	Sí	Sí	Sí	Clave de la temporada. Referencia a Temporada.
TipoDia	smallint	NOT NULL	Sí	Sí	Sí	Clave del tipo de día. Referencia a TipoDia.
Seccion	smallint	NOT NULL	Sí	No	Sí	Clave única de la sección.
Descripcion	varchar(60)	NOT NULL	No	No	Sí	Nombre del tipo de Sección.
HoraInicio	smalldatetime	NOT NULL	No	No	Sí	Hora de inicio de la sección.
HoraFin	smalldatetime	NOT NULL	No	No	Sí	Hora de fin de la sección.
Frecuencia	real	NULL	No	No	No	Frecuencia de paso de las unidades de transporte durante la sección.
Vehiculos	real	NULL	No	No	No	Número de vehículos necesarios durante la sección.
TiempoRecorrido	real	NULL	No	No	No	Tiempo de recorrido de la ruta durante la sección.
VelocidadComercial	real	NULL	No	No	No	Velocidad comercial de las unidades de transporte durante la sección.

Atributos de la entidad " Temporada "						
Nombre	**Tipo de dato**	**Null/Not Null**	**PK**	**FK**	**Dato Obligatorio**	**Descripción**
Temporada	smallint	NOT NULL	Sí	No	Sí	Clave única de la temporada.
Ruta	smallint	NOT NULL	Sí	Sí	Sí	Clave de la ruta. Referencia a Ruta.
Descripcion	varchar(60)	NOT NULL	No	No	Sí	Nombre de la temporada.
DiaInicio	smallint	NOT NULL	No	No	Sí	Día de inicio de la temporada.
MesInicio	smallint	NOT NULL	No	Sí	Sí	Mes de inicio de la temporada. Referencia a Mes.
DiaFin	smallint	NOT NULL	No	No	Sí	Día de fin de la temporada.
MesFin	smallint	NOT NULL	No	Sí	Sí	Mes de fin de la temporada. Referencia a Mes.

Atributos de la entidad " TipoDia "						
Nombre	**Tipo de dato**	**Null/Not Null**	**PK**	**FK**	**Dato Obligatorio**	**Descripción**
TipoDia	smallint	NOT NULL	Sí	No	Sí	Clave única del tipo de día.
Temporada	smallint	NOT NULL	Sí	Sí	Sí	Clave de la temporada. Referencia a Temporada.
Ruta	smallint	NOT NULL	Sí	Sí	Sí	Clave de la ruta. Referencia a Ruta.
Descripcion	varchar(60)	NOT NULL	No	No	Sí	Nombre del tipo de día.

Atributos de la entidad " Vehículo "						
Nombre	**Tipo de dato**	**Null/Not Null**	**PK**	**FK**	**Dato Obligatorio**	**Descripción**
Vehiculo	smallint	NOT NULL	Sí	No	Sí	Clave única del vehículo.
Descripcion	varchar(50)	NOT NULL	No	No	Sí	Nombre del vehículo.
Asientos	smallint	NOT NULL	No	No	Sí	Número de asientos del vehículo.
PasajerosPie	smallint	NOT NULL	No	No	Sí	Número de pasajeros de pie que entran en el vehículo.

9.7 Diseño arquitectónico

En esta vista de diseño, se establece la estructura del sistema, mostrando sus componentes principales y las relaciones entre ellos.

Para definir el estilo arquitectónico del sistema, se tomaron en cuenta los siguientes aspectos:

1. La forma en que los componentes del sistema se comunican los datos necesarios para su funcionamiento, la forma en que los datos están disponibles a los componentes que los utilizan y la forma en que los datos son almacenados.

2. La forma en que los componentes del sistema se pasan el control de las acciones conforme el sistema se ejecuta.

El diseño arquitectónico definido para el sistema es el siguiente:

a) En cuanto a la forma en que se manejan los datos, el diseño consiste en una arquitectura centrada en los datos, como se muestra en la figura 9.2:

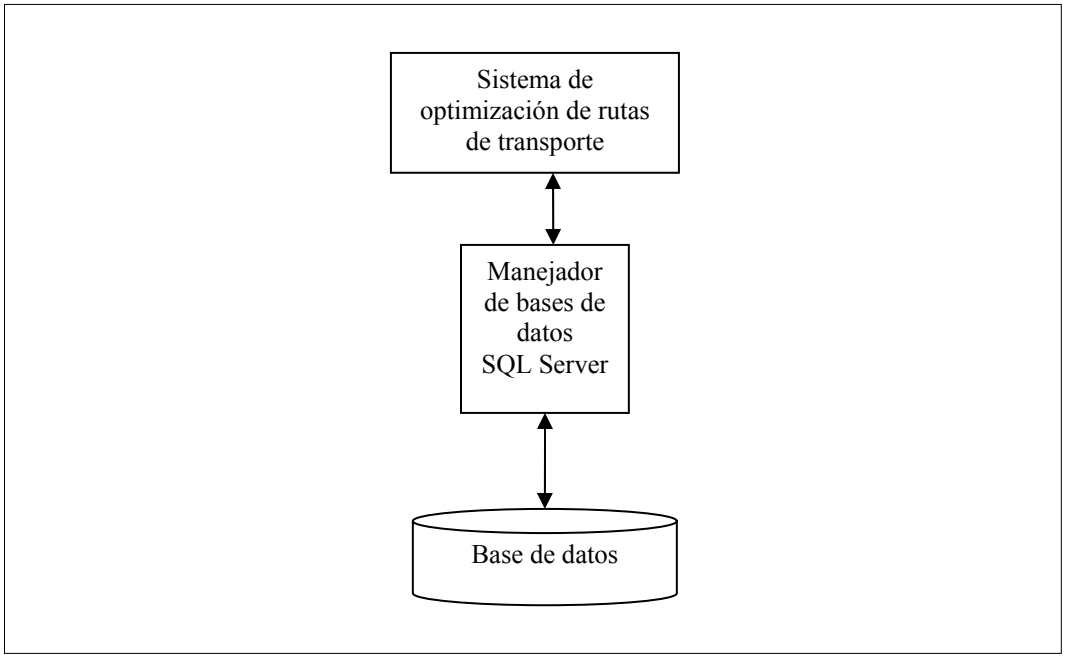


Fig. 9.2 Arquitectura centrada en los datos

Con base en este modelo de arquitectura, los datos del sistema serán almacenados en una base de datos y el sistema tendrá acceso a los datos para su consulta, modificación o eliminación a través de peticiones al manejador de la base de datos, el cual a su vez hará las operaciones necesarias sobre la base de datos.

b) En cuanto a la forma en que los componentes se pasan el control de las acciones, el diseño consiste en una arquitectura de llamada y retorno, en la modalidad de arquitectura de programa principal/subprograma, como se muestra en la figura 9.3:

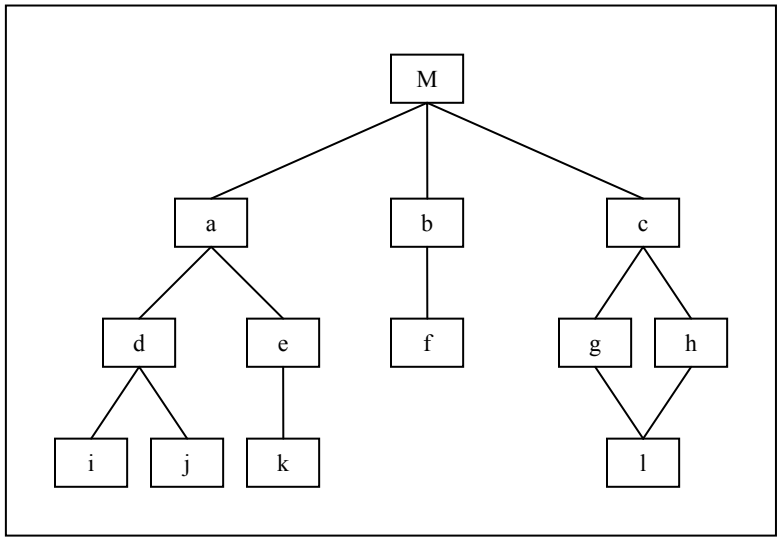


Fig. 9.3 Arquitectura de llamada y retorno

Con base en este enfoque, el sistema comenzará con una pantalla principal y dependiendo de las operaciones que el usuario realice, se irá creando las pantallas correspondientes a las operaciones y se irá pasando el control a éstas, regresando el control a la pantalla principal al final de las operaciones.

9.8 Diseño de la interfaz de usuario

Con base en la arquitectura de llamada y retorno establecida y en los escenarios de uso definidos en el capítulo relacionado con la especificación de requerimientos, se definieron las pantallas del programa haciendo uso de los siguientes patrones de diseño de interfaces gráficas de usuario (véase Marco teórico en la sección de Diseño de interfaces):

- Secuencia de acciones no ligadas
- Paso a paso (wizard)
- Listas en cascada

Las pantallas diseñadas son las siguientes:

9.8.1 Pantalla principal

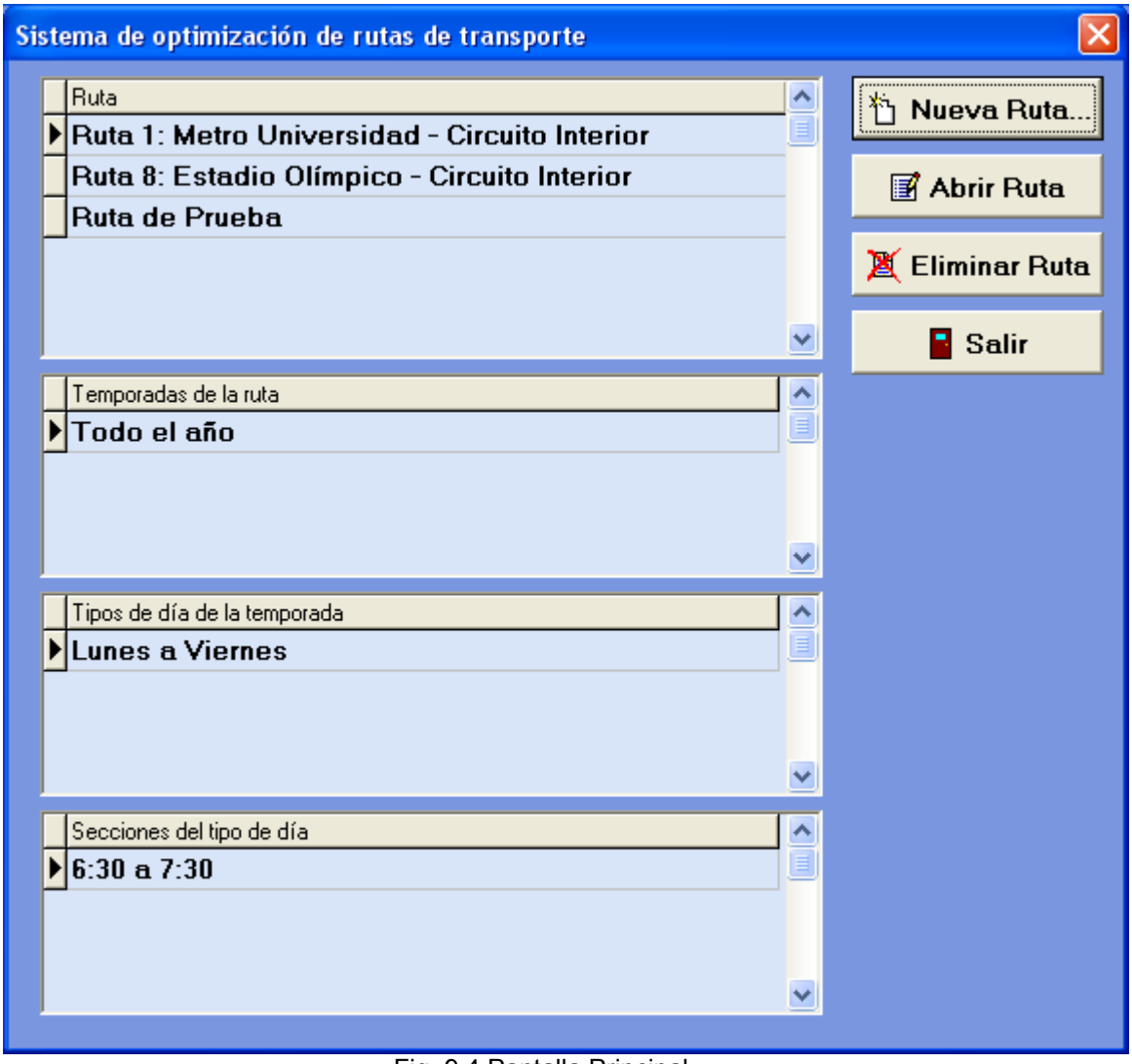


Fig. 9.4 Pantalla Principal

La figura 9.4 muestra la pantalla principal del sistema, en la que se muestran las rutas definidas con anterioridad. Para cada ruta el sistema muestra las temporadas definidas, para cada temporada el sistema muestra los tipos de días que tiene definidos y para cada tipo de día el sistema muestra las secciones definidas.

A partir de esta pantalla, el usuario puede crear una nueva ruta (ver pantalla 9.8.2 Creación de una Ruta), editar o eliminar las rutas existentes (ver pantallas 9.8.3 Edición de una ruta y 9.8.4 Eliminación de una ruta) o salir del sistema.

9.8.2 Creación de una ruta

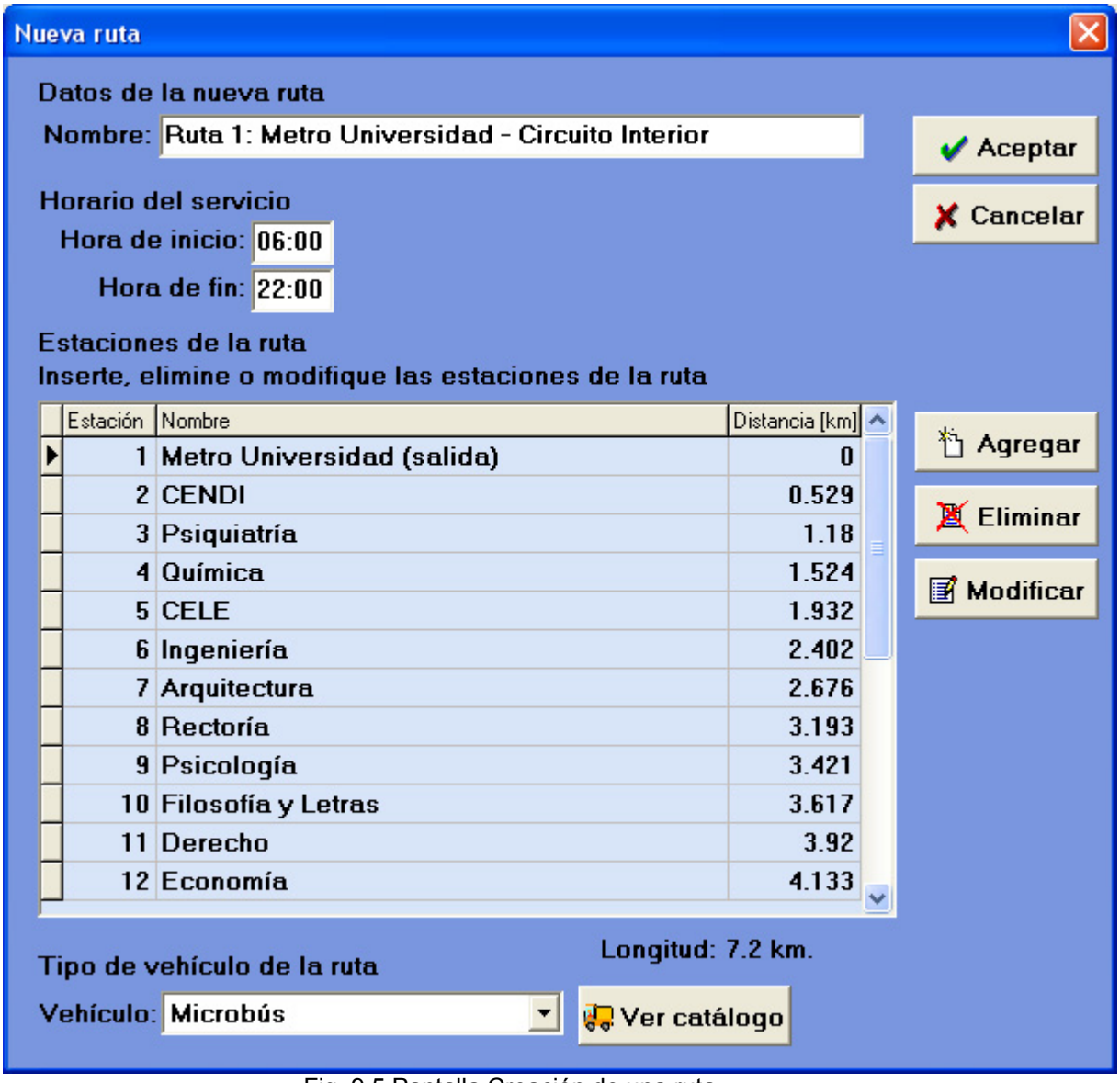


Fig. 9.5 Pantalla Creación de una ruta

La figura 9.5 muestra la pantalla de creación de una ruta, a través de cuyo uso el usuario puede crear una ruta nueva. Los datos necesarios para crear la ruta son el nombre, el horario de servicio, las estaciones y el tipo de vehículo utilizado. La definición de las estaciones consiste en agregarlas (ver pantalla 9.8.5 Agregar Estación de una Ruta) y las estaciones agregadas pueden eliminarse o se puede modificar sus datos. En cuanto a la definición del tipo de vehículo, se puede escoger un tipo de vehículo ya existente a través del combo o puede verse el catálogo de vehículos (ver pantalla 9.8.8 Catálogo de vehículos de una ruta) a través del cual se puede agregar el tipo de vehículo adecuado en caso de que éste no exista en el catálogo.

9.8.3 Edición de una ruta

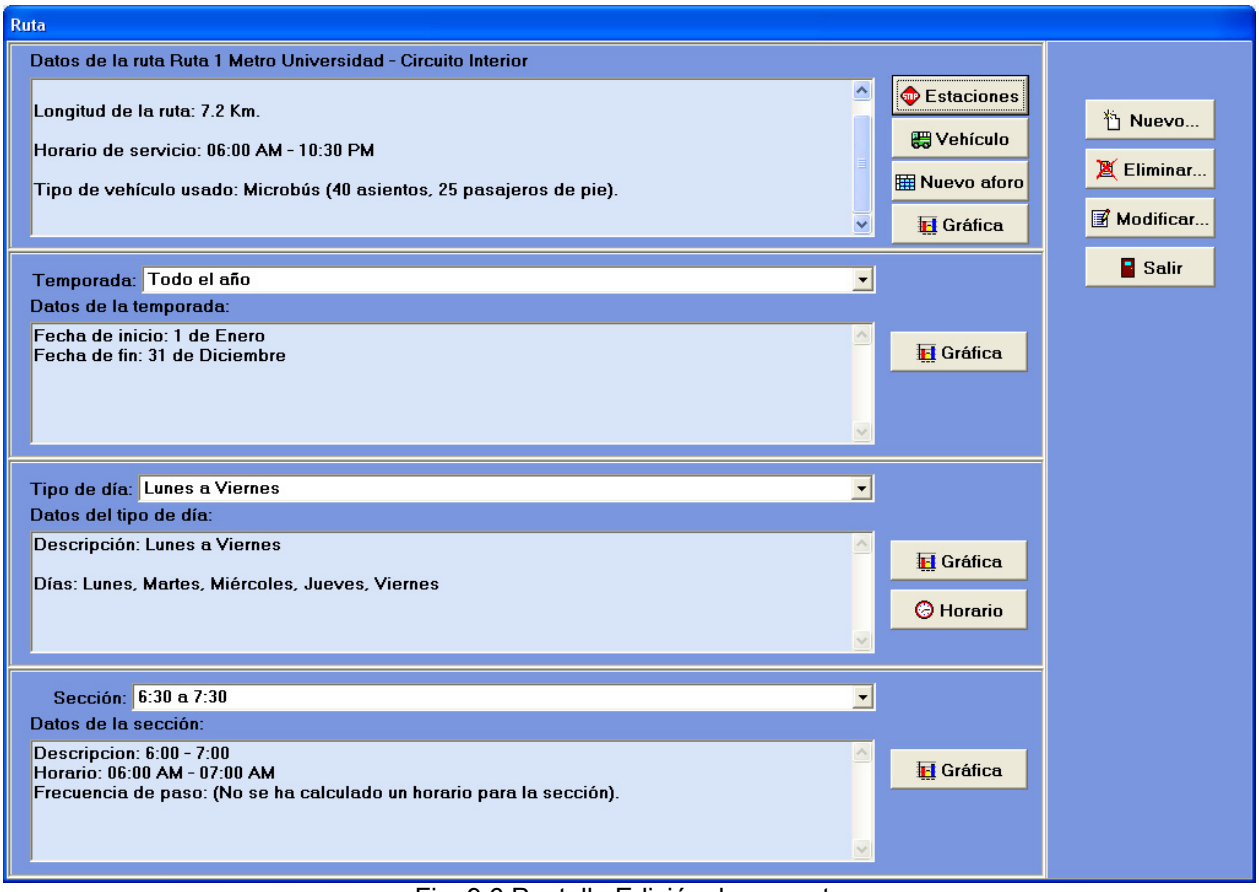


Fig. 9.6 Pantalla Edición de una ruta

La figura 9.6 muestra la pantalla de edición de una ruta, A través de esta pantalla se pueden consultar y modificar las estaciones y el tipo de vehículo de una ruta, se pueden almacenar en el sistema los datos de los nuevos aforos (ver pantalla 9.8.13 Forma para captura de un nuevo aforo), se puede crear el horario por cada tipo de día de las distintas temporadas definidas para la ruta (ver pantalla 9.8.17 Forma para crear el horario de servicio), se pueden observar las gráficas de demanda de servicio para cada temporada, tipo de día y sección de día definido para la ruta (ver pantallas 9.8.14, 9.8.15 y 9.8.16) y se pueden crear nuevas temporadas, tipos de día y secciones (ver pantallas 9.8.18 a 9.18.21).

9.8.4 Eliminación de una ruta (mensaje de confirmación)

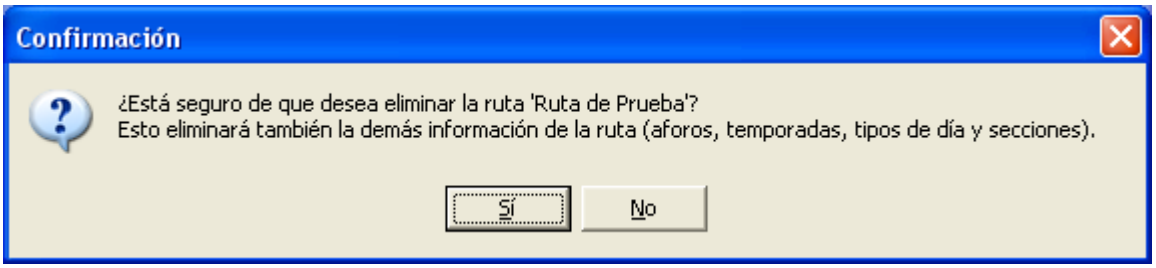


Fig. 9.7 Pantalla Eliminación de una ruta (mensaje de confirmación)

La figura 9.7 muestra el mensaje de petición de confirmación de eliminación de una ruta.

9.8.5 Agregar una estación a una ruta

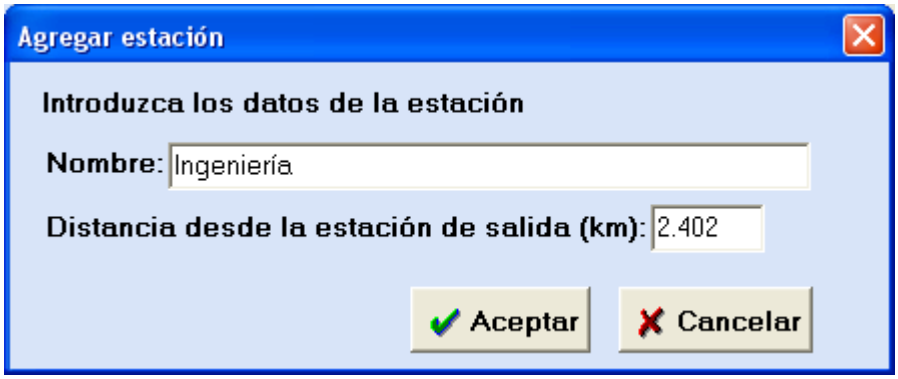


Fig. 9.8 Pantalla Agregar una estación a una ruta

La figura 9.8 muestra la pantalla para agregar estaciones a una ruta. Ésta es la pantalla que sirve para introducir una nueva estación o parada a una ruta. Los datos que debe introducirse son el nombre de la estación o parada y la distancia que existe de la estación de salida a la estación introducida, en kilómetros.

9.8.6 Modificar una estación de una ruta

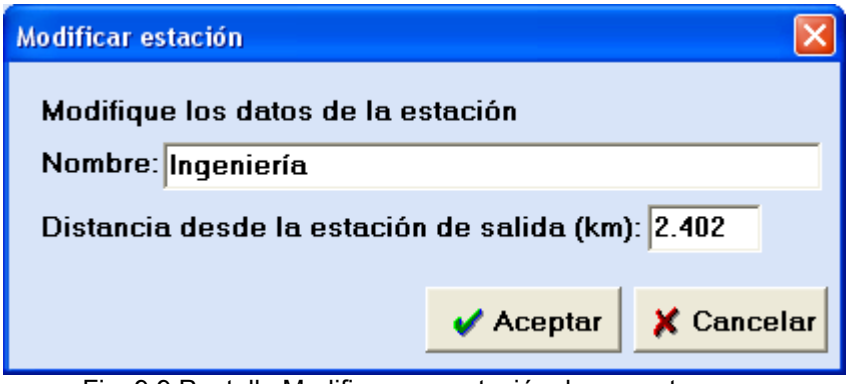


Fig. 9.9 Pantalla Modificar una estación de una ruta

La figura 9.9 muestra la pantalla para modificación de las estaciones. Ésta es la pantalla que sirve para modificar los datos de una estación o parada. Se puede modificar el nombre y la distancia que existe de la estación de salida a la estación.

9.8.7 Eliminar una estación de una ruta (mensaje de confirmación)

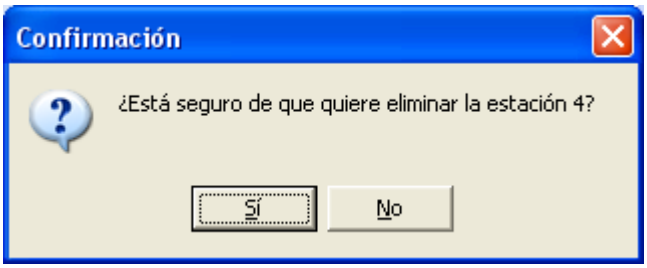


Fig. 9.10 Pantalla Eliminar una estación de una ruta (mensaje de confirmación)

La figura 9.10 muestra el mensaje de petición de confirmación para eliminar una estación de una ruta.

9.8.8 Catálogo de vehículos de una ruta

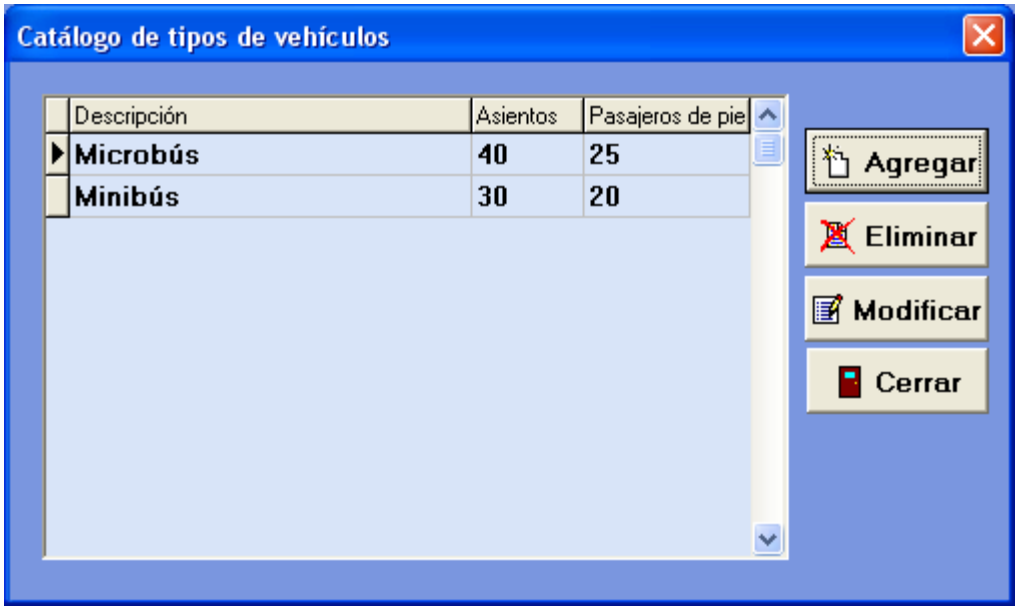


Fig. 9.11 Pantalla Catálogo de vehículos de una ruta

La figura 9.11 muestra la pantalla de catálogo de vehículos, la cual muestra el catálogo de tipos de vehículos disponibles para su uso en la creación de horarios para las rutas. Para cada vehículo se muestra su capacidad (número de asientos y capacidad de pasajeros de pie). A partir de esta pantalla se puede agregar tipos de vehículos (ver pantalla 9.8.9 Agregar un tipo de vehículo), eliminar uno o más tipos de vehículos existentes (ver pantalla 9.8.11 Eliminación de un tipo de vehículo) o modificar un tipo de vehículo ya existente (ver pantalla 9.8.10 Modificar un tipo de vehículo).

9.8.9 Agregar un tipo de vehículo

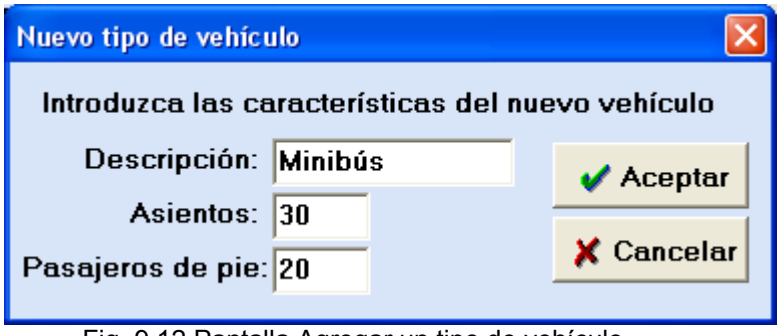


Fig. 9.12 Pantalla Agregar un tipo de vehículo

La figura 9.12 muestra la pantalla para agregar un tipo de vehículo. Esta es la pantalla que sirve para agregar un nuevo tipo de vehículo al catálogo de tipos de vehículos. Los datos que se debe introducir para cada tipo de vehículo son una descripción, el número de asientos y el número de pasajeros de pie.

9.8.10 Modificar un tipo de vehículo

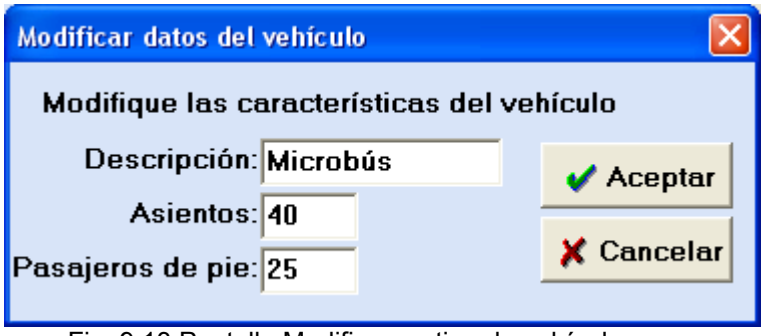


Fig. 9.13 Pantalla Modificar un tipo de vehículo

La figura 9.13 muestra la pantalla para modificar un tipo de vehículo. Ésta es la pantalla que sirve para modificar la información de un tipo de vehículo existente. Se puede modificar el nombre, el número de asientos y el número de pasajeros de pie.

9.8.11 Eliminación de un tipo de vehículo (mensaje de confirmación)

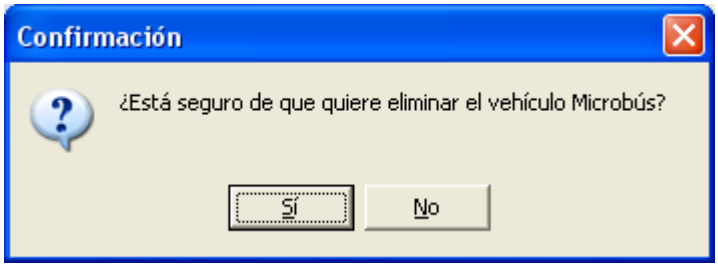


Fig. 9.14 Pantalla Eliminación de un tipo de vehículo (mensaje de confirmación)

La figura 9.14 muestra el mensaje de petición de confirmación de eliminación de un tipo de vehículo.

9.8.12 Cambio del tipo de vehículo de una ruta

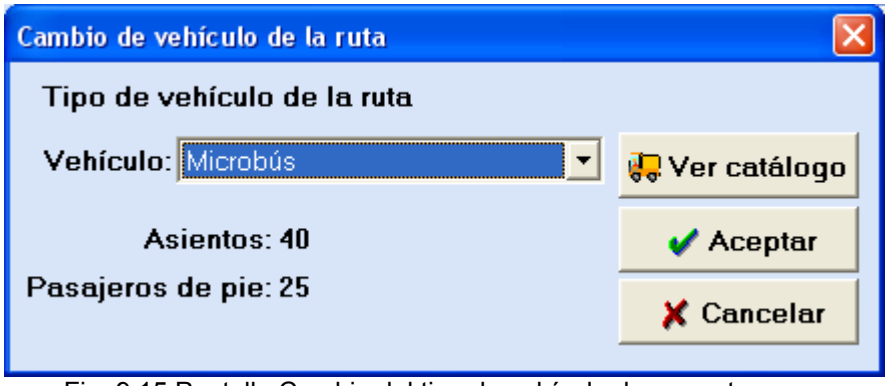
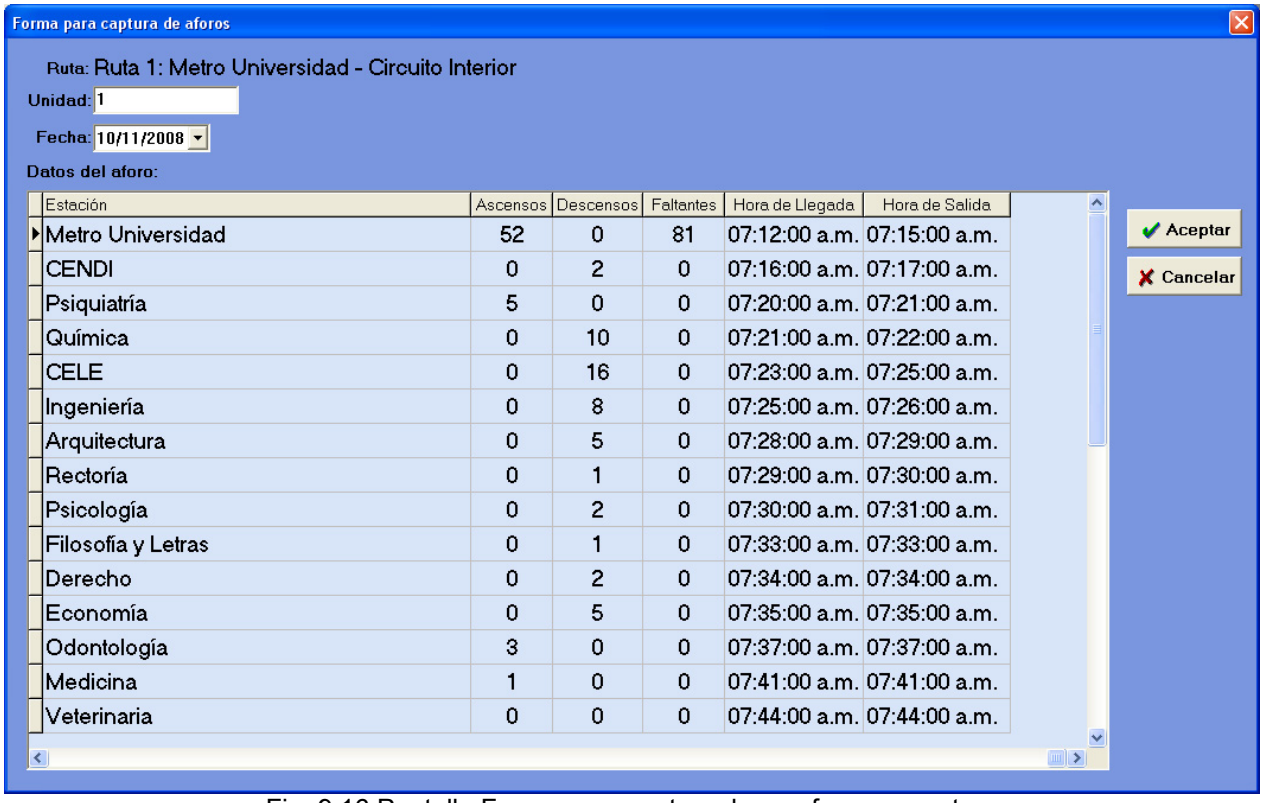


Fig. 9.15 Pantalla Cambio del tipo de vehículo de una ruta

La figura 9.15 muestra la pantalla de cambio del tipo de vehículo de una ruta. Esta pantalla sirve para modificar el tipo de vehículo asignado a una ruta. El cambio consiste simplemente en buscar en el combo el tipo de vehiculo por el que se desea cambiar el tipo de vehículo actual.

9.8.13 Forma para captura de un aforo o muestreo



Estación	Ascensos	Descensos	Faltantes	Hora de Llegada	Hora de Salida
Metro Universidad	52	0	81	07:12:00 a.m.	07:15:00 a.m.
CENDI	0	2	0	07:16:00 a.m.	07:17:00 a.m.
Psiquiatría	5	0	0	07:20:00 a.m.	07:21:00 a.m.
Química	0	10	0	07:21:00 a.m.	07:22:00 a.m.
CELE	0	16	0	07:23:00 a.m.	07:25:00 a.m.
Ingeniería	0	8	0	07:25:00 a.m.	07:26:00 a.m.
Arquitectura	0	5	0	07:28:00 a.m.	07:29:00 a.m.
Rectoría	0	1	0	07:29:00 a.m.	07:30:00 a.m.
Psicología	0	2	0	07:30:00 a.m.	07:31:00 a.m.
Filosofía y Letras	0	1	0	07:33:00 a.m.	07:33:00 a.m.
Derecho	0	2	0	07:34:00 a.m.	07:34:00 a.m.
Economía	0	5	0	07:35:00 a.m.	07:35:00 a.m.
Odontología	3	0	0	07:37:00 a.m.	07:37:00 a.m.
Medicina	1	0	0	07:41:00 a.m.	07:41:00 a.m.
Veterinaria	0	0	0	07:44:00 a.m.	07:44:00 a.m.

Fig. 9.16 Pantalla Forma para captura de un aforo o muestreo

La figura 9.16 muestra la pantalla que sirve para capturar un aforo o muestreo hecho a una ruta. Los datos que se introducen son la unidad de transporte que se utilizó para el muestreo, la fecha en que se realizó el muestreo y los datos recolectados en cada estación o parada (ascensos, descensos, personas que no pudieron subir a la unidad, hora de llegada a la parada y hora de partida de la parada).

9.8.14 Gráfica de demanda de una ruta

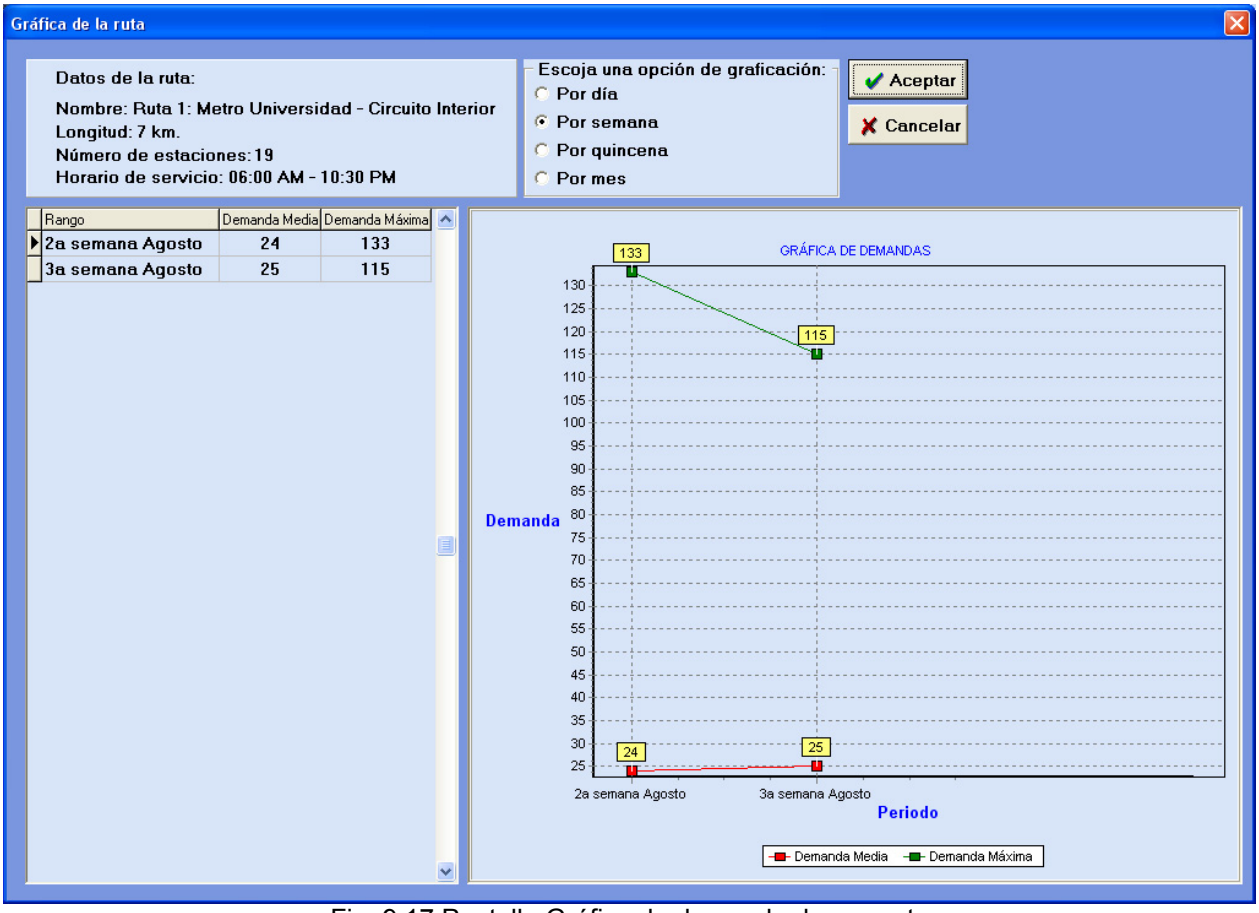


Fig. 9.17 Pantalla Gráfica de demanda de una ruta

La figura 9.17 muestra la pantalla que permite observar el comportamiento de la demanda de servicio de transporte para una ruta, a lo largo de todas las temporadas, tipos de día y secciones definidas. El usuario elige observar el comportamiento de la demanda de servicio por día, semana, quincena o mes. El sistema calcula y muestra en la gráfica la demanda media y la demanda máxima.

9.8.15 Gráfica de demanda de una temporada

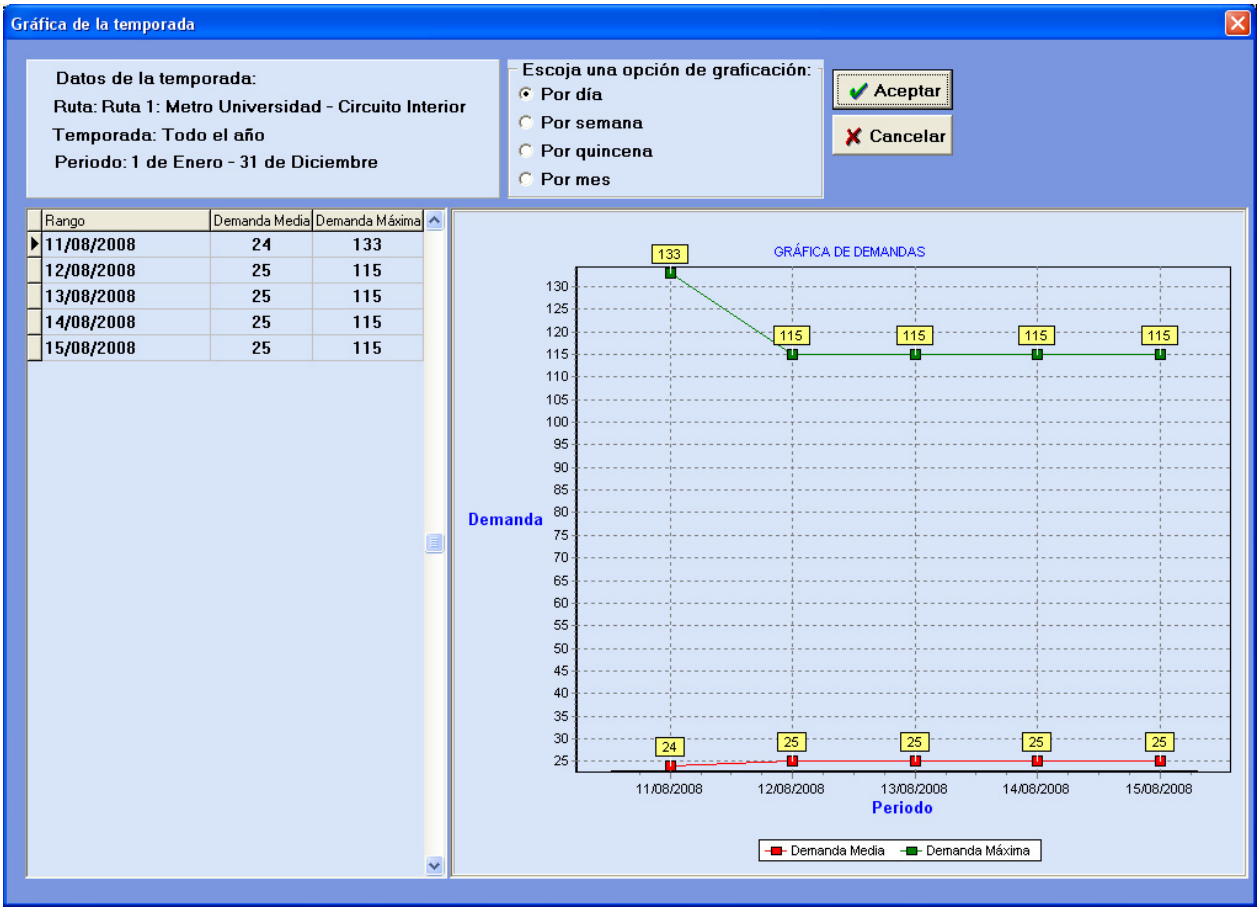


Fig. 9.18 Pantalla Gráfica de demanda de una temporada

La figura 9.18 muestra la pantalla que permite observar el comportamiento de la demanda de servicio de transporte para una temporada de una ruta, a lo largo de todos los tipos de día y secciones definidas. El usuario elige observar el comportamiento de la demanda de servicio por día, semana, quincena o mes. El sistema calcula y muestra en la gráfica la demanda media y la demanda máxima.

9.8.16 Gráfica de demanda de un tipo de día

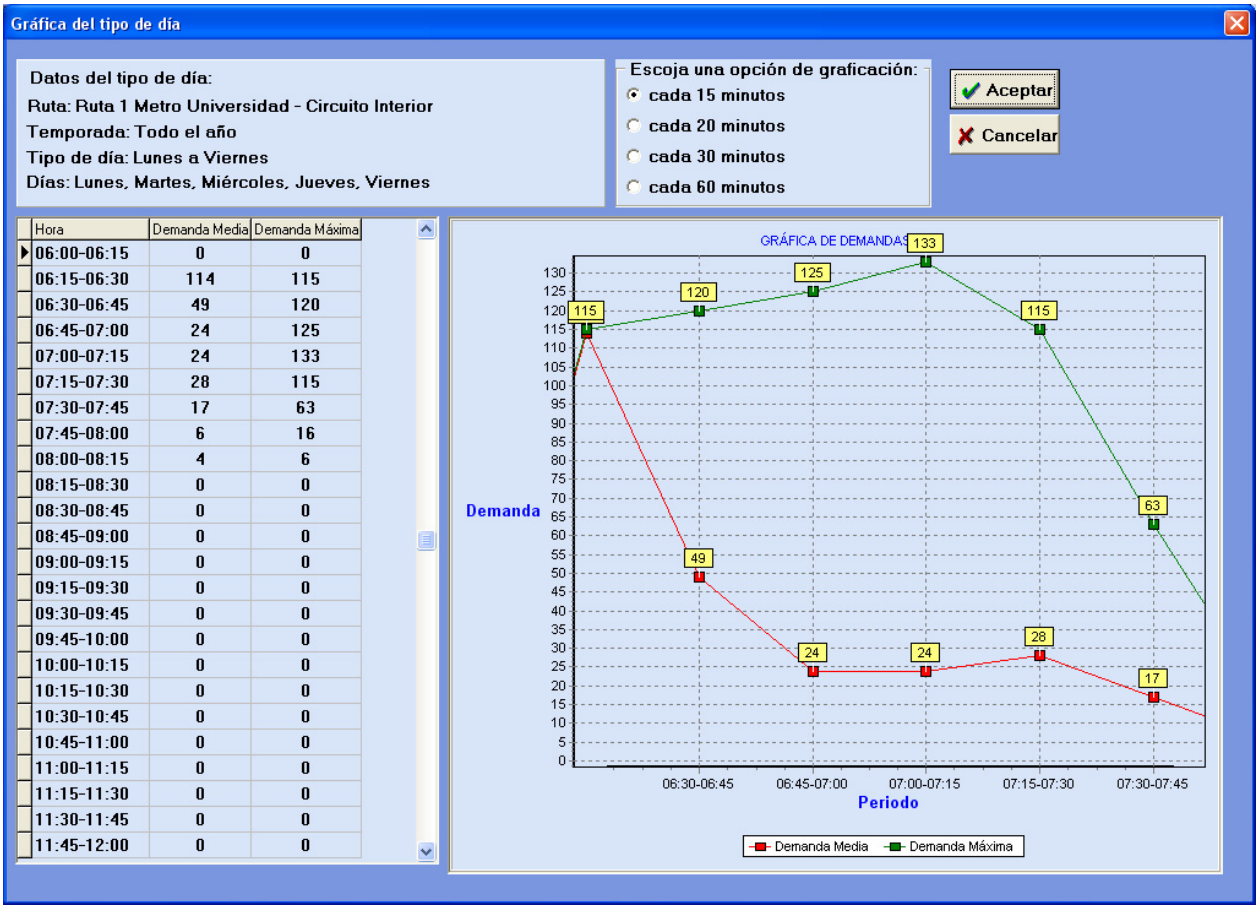


Fig. 9.19 Pantalla Gráfica de demanda de un tipo de día

La figura 9.19 muestra la pantalla que permite observar el comportamiento de la demanda de servicio de transporte para un tipo de día, a lo largo de todas las secciones definidas. El usuario elige observar el comportamiento de la demanda de servicio cada 15, 20, 30 o 60 minutos. El sistema calcula y muestra en la gráfica la demanda media y la demanda máxima.

9.8.17 Forma para creación de horarios de servicio

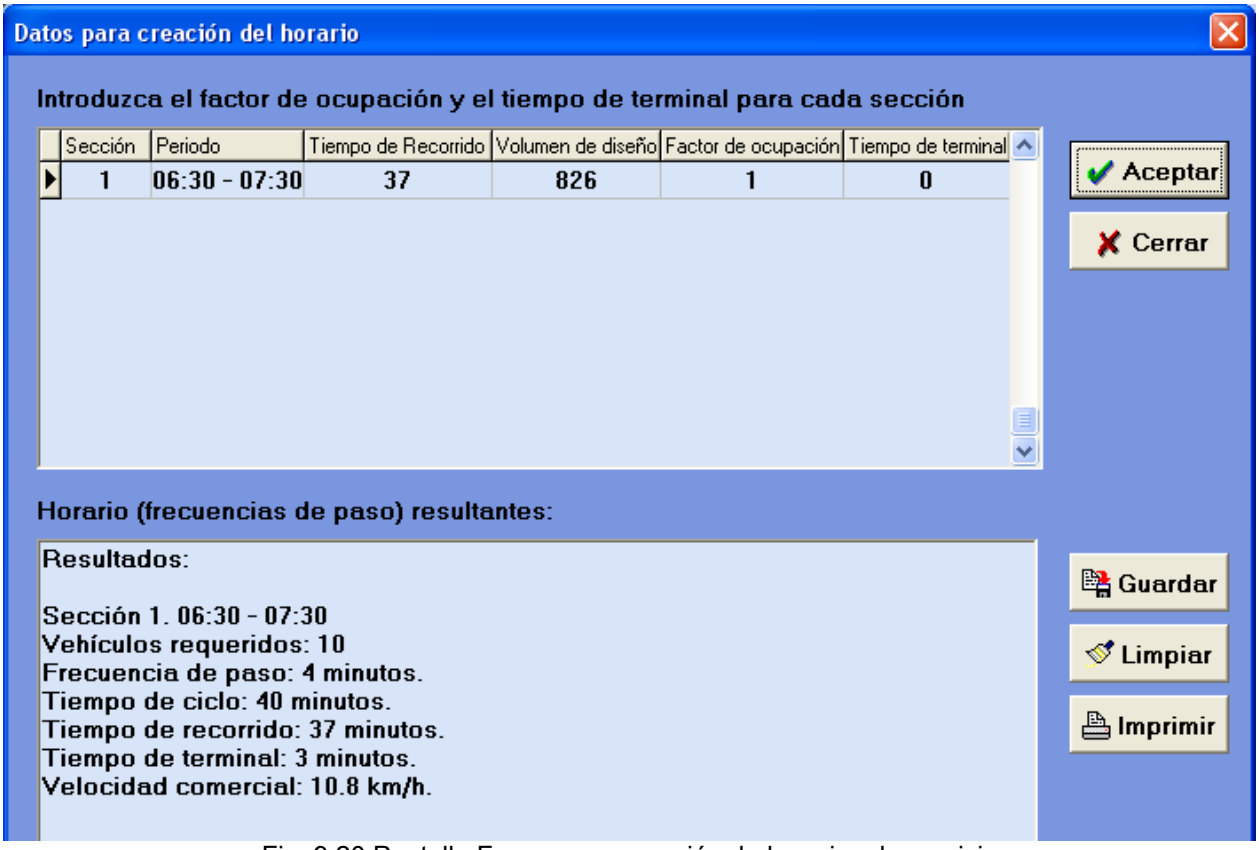


Fig. 9.20 Pantalla Forma para creación de horarios de servicio

La figura 9.20 muestra la pantalla en que se crean los horarios de servicio óptimos. El sistema muestra, con base en los datos recopilados en los aforos y almacenados, así como la configuración de la ruta, las secciones definidas, el horario de las secciones, el tiempo de recorrido de la ruta en cada sección del día y el volumen de diseño (demanda de servicio que se utiliza para saber la cantidad de vehículos necesaria para satisfacer la demanda). El usuario introduce el factor de ocupación (porcentaje que indica qué tan llenas se desean que estén las unidades de transporte durante el recorrido) y el tiempo de terminal (tiempo que las unidades tardan en llegar a la última terminal y volver a salir a dar el servicio) y con base en todos los datos anteriores, el sistema calcula el número de vehículos requeridos, la frecuencia de paso (intervalo de tiempo entre el paso de una unidad de transporte y la siguiente), el tiempo que tomará a la unidad de transporte recorrer toda la ruta, el tiempo de terminal real y la velocidad promedio a la cual la unidad de transporte recorrerá la ruta. Una vez generado el horario por el sistema, el usuario puede elegir guardarlo, borrarlo o imprimirlo.

9.8.18 Opciones de creación de temporadas, tipos de día o secciones

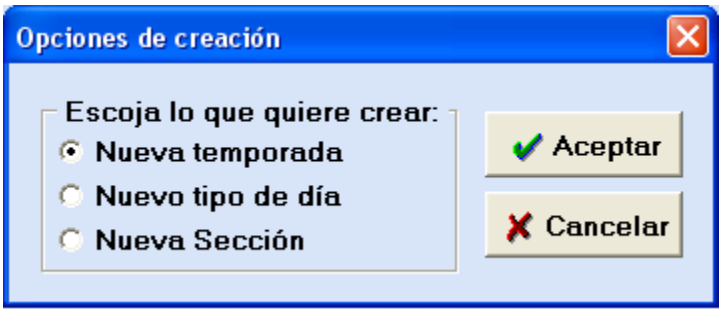


Fig. 9.21 Pantalla Opciones de creación de temporadas, tipos de día o secciones

La figura 9.21 muestra la pantalla que es usada para definir si se quiere crear una nueva temporada (ver pantalla 9.8.19 Creación de una temporada), tipo de día (ver pantalla 9.8.20 Creación de un tipo de día) o sección (ver pantalla 9.8.21 Creación de una sección de un tipo de día).

9.8.19 Creación de una temporada

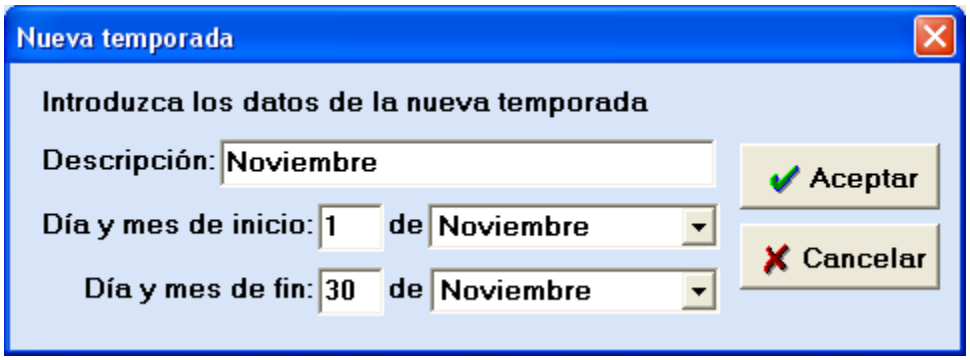


Fig. 9.22 Pantalla Creación de una temporada

La figura 9.22 muestra la pantalla que sirve para crear una nueva temporada de una ruta. Los datos a introducir son descripción de la temporada, así como la el día y mes de inicio de la temporada y el día y mes de fin de la temporada. Después de la creación de un tipo de día se regresa a la pantalla principal de la ruta (figura 9.6) mostrando la temporada creada.

9.8.20 Creación de un tipo de día

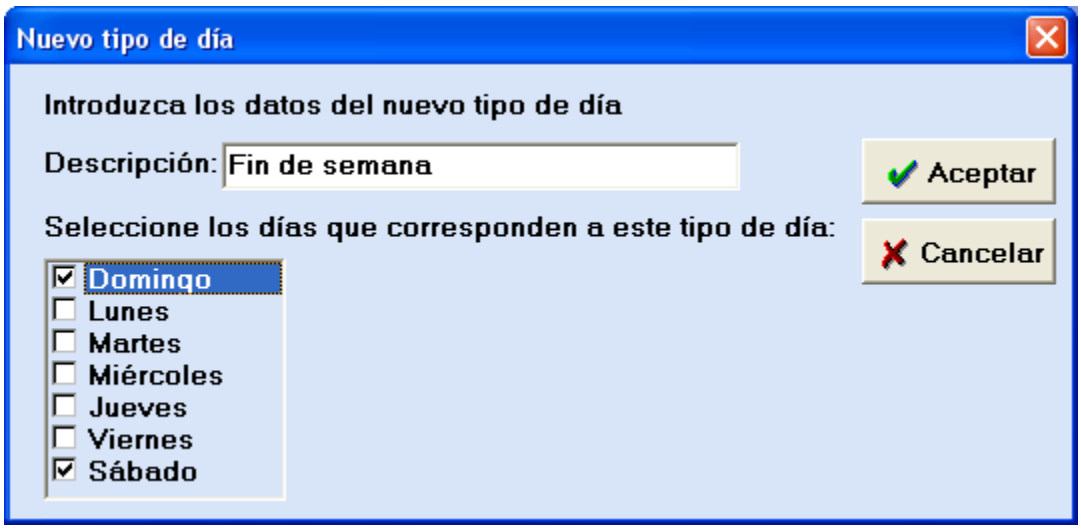


Fig. 9.23 Pantalla Creación de un tipo de día

La figura 9.23 muestra la pantalla que sirve para crear un nuevo tipo de día. Los datos a introducir son una descripción del tipo de día y los días de la semana que abarca. Después de la creación de un tipo de día se regresa a la pantalla principal de la ruta (figura 9.6) mostrando el tipo de día creado.

9.8.21 Creación de una sección de un tipo de día

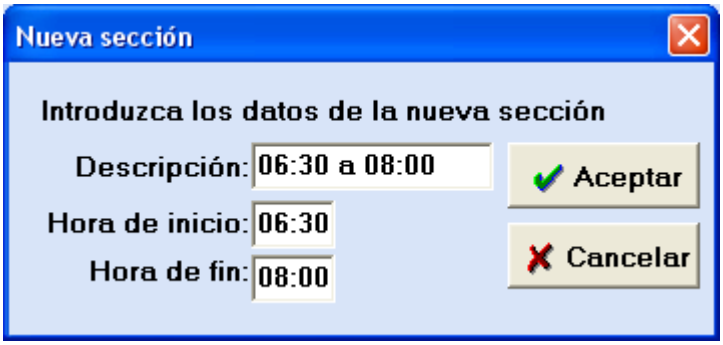


Fig. 9.24 Pantalla Creación de una sección de un tipo de día

La figura 9.23 muestra la pantalla que sirve para crear una sección dentro de un tipo de día. Los datos a introducir son una descripción de la sección y la hora de inicio y de fin de la sección. Después de la creación de un tipo de día se regresa a la pantalla principal de la ruta (figura 9.6) mostrando la sección creada.

9.9 Diseño procedimental

Una vez establecidos el diseño de datos, de arquitectura y de interfaces, tiene lugar el diseño procedimental, el cual tiene como objetivo convertir los modelos anteriormente mencionados en un sistema de software operacional. El nivel de abstracción de los elementos creados en el diseño procedimental es muy bajo, más cercano a la construcción real o codificación, a diferencia de los altos niveles de abstracción utilizados en las otras vistas de diseño.

9.9.1 Diseño de datos

A continuación se muestra los procedimientos que realizan la construcción de la base de datos y las entidades correspondientes, con base en el diseño de la base de datos:

```
/*    This SQL DDL script was generated by Microsoft Visual Studio. */

/*   Driver Used: Microsoft Visual Studio - Microsoft SQL Server Driver.          */
/*   Document: Diagrama Entidad-Relacion.vsd. */
/*    Time Created: 10 de October de 2005 12:29 a.m..                    */
/*    Operation   : From Visio Generate Wizard.                  */
/*    Connected server      : Gustavo.                    */
/*    Connected database    : Rutas.                    */


/* Creación de la base de datos Rutas */
/* Create Rutas database.                                    */
use master
go

create database "Rutas"
go

use "Rutas"
go

/* Creación de las tablas y sus claves primarias */
/* Create new table "Ruta".                                    */
create table "Ruta" (
        "Ruta" smallint not null,
        "Nombre" varchar(60) not null,
        "Longitud" real not null,
        "HoraInicio" smalldatetime not null,
        "HoraFin" smalldatetime not null,
        "Vehiculo" smallint not null)
go

alter table "Ruta"
        add constraint "Ruta_PK" primary key ("Ruta")
go
/* Create new table "Estacion".                                    */
create table "Estacion" (
        "Ruta" smallint not null,
        "Estacion" smallint not null,
        "Nombre" varchar(50) not null,
        "Distancia" real not null)
go
```

```
alter table "Estacion"
	add constraint "Estacion_PK" primary key ("Estacion", "Ruta")
go


/* Create new table "Vehiculo".                                    */
create table "Vehiculo" (
	"Vehiculo" smallint not null,
	"Descripcion" varchar(50) not null,
	"Asientos" smallint not null,
	"PasajerosPie" smallint not null)
go

alter table "Vehiculo"
	add constraint "Vehiculo_PK" primary key ("Vehiculo")
go


/* Create new table "AforoDemanda".                                */
create table "AforoDemanda" (
	"Aforo" smallint not null,
	"Ruta" smallint not null,
	"Estacion" smallint not null,
	"Unidad" varchar(15) not null,
	"Fecha" smalldatetime not null,
	"HoraLlegada" smalldatetime not null,
	"HoraSalida" smalldatetime not null,
	"Ascensos" smallint not null,
	"Descensos" smallint not null,
	"Ocupantes" smallint not null,
	"Faltantes" smallint not null,
	"Demanda" smallint not null)
go

alter table "AforoDemanda"
	add constraint "AforoDemanda_PK" primary key ("Aforo")
go


/* Create new table "Temporada".                                   */
create table "Temporada" (
	"Ruta" smallint not null,
	"Temporada" smallint not null,
	"Descripcion" varchar(60) not null,
	"DiaInicio" smallint not null,
	"MesInicio" smallint not null,
	"DiaFin" smallint not null,
	"MesFin" smallint not null,
	"Mes" smallint null)
go

alter table "Temporada"
	add constraint "Temporada_PK" primary key ("Temporada", "Ruta")
go
```

```
/* Create new table "Mes".                                                    */
create table "Mes" (
        "Mes" smallint not null,
        "Descripcion" varchar(10) not null)
go

alter table "Mes"
        add constraint "Mes_PK" primary key ("Mes")
go


/* Create new table "TipoDia".                                                */
create table "TipoDia" (
        "Ruta" smallint not null,
        "Temporada" smallint not null,
        "TipoDia" smallint not null,
        "Descripcion" varchar(60) not null)
go

alter table "TipoDia"
        add constraint "TipoDia_PK" primary key ("TipoDia", "Temporada", "Ruta")
go


/* Create new table "DiaTipoDia".                                             */
create table "DiaTipoDia" (
        "TipoDia" smallint not null,
        "Temporada" smallint not null,
        "Ruta" smallint not null,
        "Dia" smallint not null)
go

alter table "DiaTipoDia"
        add constraint "DiaTipoDia_PK" primary key ("TipoDia", "Temporada", "Ruta", "Dia")
go


/* Create new table "Dia".                                                    */
create table "Dia" (
        "Dia" smallint not null,
        "Descripcion" varchar(10) not null)
go

alter table "Dia"
        add constraint "Dia_PK" primary key ("Dia")
go


/* Create new table "Seccion".                                                */
create table "Seccion" (
        "Ruta" smallint not null,
        "Temporada" smallint not null,
        "TipoDia" smallint not null,
        "Seccion" smallint not null,
        "Descripcion" varchar(60) not null,
        "HoraInicio" smalldatetime not null,
```

```
        "HoraFin" smalldatetime not null,
        "Frecuencia" real null,
        "Vehiculos" real null,
        "TiempoRecorrido" real null,
        "VelocidadComercial" real null)
go

alter table "Seccion"
        add constraint "Seccion_PK" primary key ("Ruta", "Temporada", "TipoDia", "Seccion")
go


/* Creación de las llaves foráneas */

/* Add foreign key constraints to table "Ruta".                    */
alter table "Ruta"
        add constraint "Vehiculo_Ruta_FK1" foreign key ("Vehiculo")
         references "Vehiculo" ("Vehiculo")
go

/* Add foreign key constraints to table "Estacion".                    */
alter table "Estacion"
        add constraint "Ruta_Estacion_FK1" foreign key ("Ruta")
         references "Ruta" ("Ruta")
go


/* Add foreign key constraints to table "AforoDemanda".                    */
alter table "AforoDemanda"
        add constraint "Ruta_AforoDemanda_FK1" foreign key ("Ruta")
         references "Ruta" ("Ruta")
go
/* Add foreign key constraints to table "Temporada".                    */
alter table "Temporada"
        add constraint "Ruta_Temporada_FK1" foreign key ("Ruta")
         references "Ruta" ("Ruta")
go


alter table "Temporada"
        add constraint "Mes_Temporada_FK1" foreign key ("MesInicio")
         references "Mes" ("Mes")
go

alter table "Temporada"
        add constraint "Mes_Temporada_FK2" foreign key ("MesFin")
         references "Mes" ("Mes")
go


/* Add foreign key constraints to table "TipoDia".                    */
alter table "TipoDia"
        add constraint "Temporada_TipoDia_FK1" foreign key ("Temporada", "Ruta")
         references "Temporada" ("Temporada",          "Ruta")
go
```

```
/* Add foreign key constraints to table "DiaTipoDia".                    */
alter table "DiaTipoDia"
        add constraint "Temporada_DiaTipoDia_FK1" foreign key ("Temporada",        "Ruta")
         references "Temporada" ("Temporada",        "Ruta")
go


alter table "DiaTipoDia"
        add constraint "TipoDia_DiaTipoDia_FK1" foreign key ("TipoDia", "Temporada", "Ruta")
         references "TipoDia" ("TipoDia", "Temporada", "Ruta")
go

alter table "DiaTipoDia"
        add constraint "Dia_DiaTipoDia_FK1" foreign key ("Dia")
         references "Dia" ("Dia")
go


/* Add foreign key constraints to table "Seccion".                       */
alter table "Seccion"
        add constraint "TipoDia_Seccion_FK1" foreign key ("TipoDia", "Temporada", "Ruta")
         references "TipoDia" ("TipoDia", "Temporada", "Ruta")
go

/* This is the end of the Microsoft Visual Studio generated SQL DDL script.   */
```

Los procedimientos anteriores constituyen el script que se utiliza para construir la base de datos del sistema de optimización de rutas de transporte, en el sistema manejador de base de datos. Esta construcción incluye la creación de la base de datos, las tablas con sus campos y los tipos de datos de los campos, así como las restricciones de llaves primarias y foráneas de las tablas.

A continuación se mostrará la forma en que se realizó el diseño procedimental de la aplicación que usa la información contenida en la base de datos.

9.9.2 Diseño de la funcionalidad del sistema

Para el diseño procedimental de las funcionalidades del sistema, se hará a través de diagramas de flujo de datos, como se muestra a continuación:

1. Diagrama de flujo de datos de nivel cero del sistema. La figura 9.24 muestra a nivel general la pantalla principal con la que comienza el sistema y se muestran las tres pantallas a las que puede accederse y la instrucción que lleva a cada una de ellas. Este diagrama muestra a un muy alto nivel la forma en que inicia el sistema, y las tres funcionalidades principales que es posible hacer a partir de la pantalla de inicio.

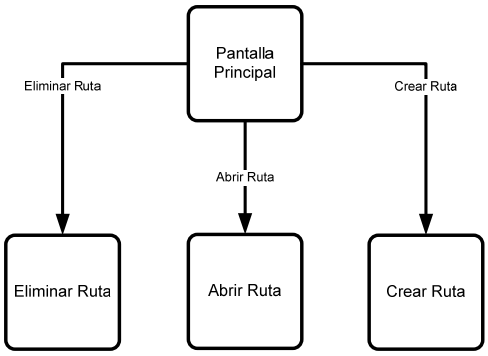


Figura 9.24 Diagrama de flujo de datos de nivel cero

2. Diagrama de flujo de datos de nivel uno de la funcionalidad de Abrir Ruta. Al abrir una ruta, el sistema consulta sus datos y los despliegan en la pantalla. A partir de ahí, pueden hacerse varias consultas o ediciones a las propiedades de la ruta: las estaciones que la forman, el tipo de vehículo usado, las temporadas, tipos de día o secciones de día. También puede consultarse el servicio actual que ofrece la ruta mediante las gráficas de la ruta, temporada, tipo de día o sección. Asimismo puede agregarse muestreos hechos a la ruta o definir un nuevo horario de servicio. Esto se muestra en la figura 9.25:

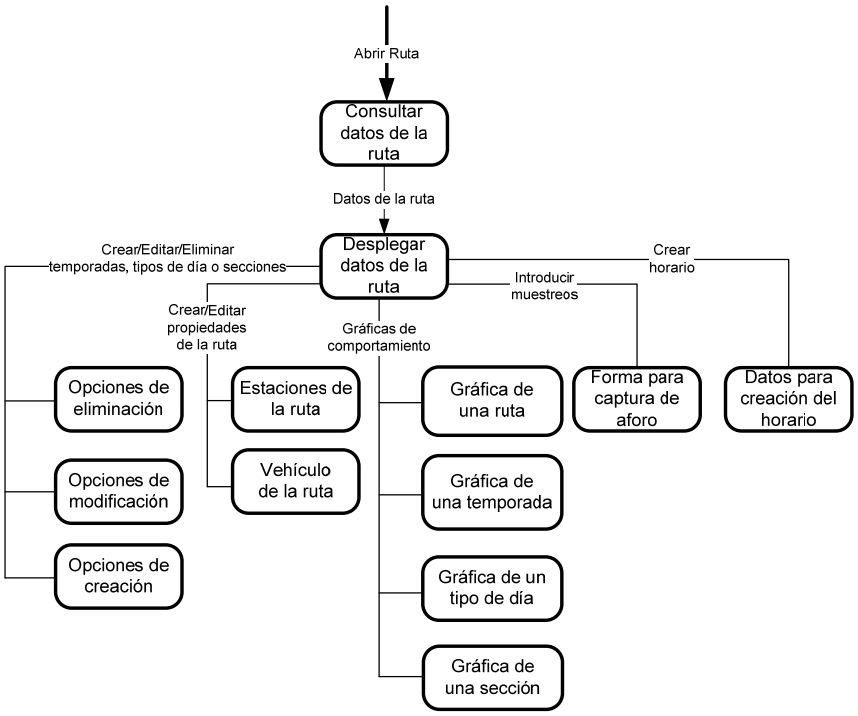


Figura 9.25 Diagrama de flujo de datos de nivel uno

3. Diagrama de flujo de datos de nivel dos para creación del horario. El sistema consulta las secciones del tipo de día para el cual se quiere generar un horario y los datos de cada sección. El usuario introduce el factor de ocupación que desea tengan las unidades de transporte. El sistema genera con estos datos, el horario con las frecuencias de paso para cada sección. Esto se muestra en la figura 9.26:

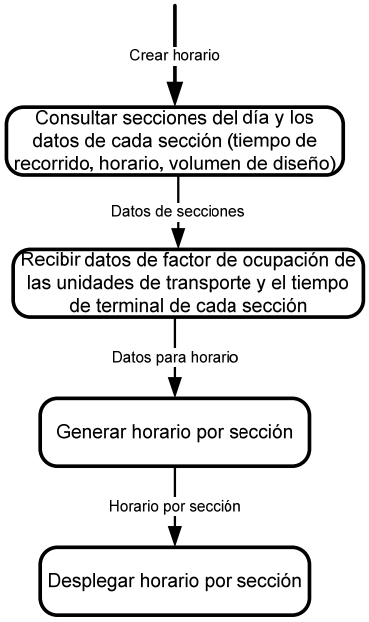


Figura 9.26 Diagrama de flujo de datos de nivel dos

4. Diagrama de flujo de datos de nivel tres para Generar horario por sección. El sistema, utilizando los datos introducidos para definir el horario, calcula el tiempo de recorrido de la ruta, la velocidad promedio de las unidades de transporte y los intervalos o frecuencias de paso de las unidades de transporte. Con estos datos se calcula el tiempo de ciclo de la sección, el número de vehículos necesarios para cubrir la ruta con esa frecuencia de paso y la velocidad promedio final de las unidades de transporte. Estos resultados son desplegados en pantalla. Esto se muestra en la figura 9.27:

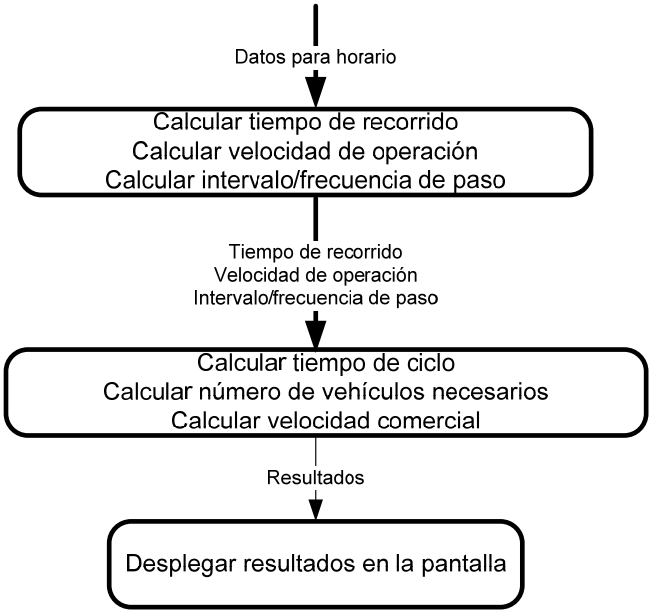


Figura 9.27 Diagrama de flujo de datos de nivel tres

Con base en el detalle del diseño procedimental mostrado en las figuras 9.24 a 9.27, a continuación se tiene el código en lenguaje de programación Delphi para la funcionalidad “Generar horario por sección”.

```
procedure TfrmHorario.btnAceptarClick(Sender: TObject);
var
  hora : String;
  i : Integer;
  hora2 : Real;
  entero : Integer;
  decimal : Real;
  TiempoRecorridoSeccion : Integer;
  VelocidadOperacion : Real;
  Intervalo : Real;
  TiempoCiclo : Real;
  NumVehiculos : Real;
  TiempoTerminal : Real;
  VelComercial : Real;
  qryAux : TQuery;
begin
  // Revisar que no haga falta datos.
  with mryDatosSecciones do
  begin
    First;
    while not EOF do
    begin
      if Trim(FieldByName('fldFactorOcupacion').AsString) = '' then
      begin
        MessageDlg('Hace falta datos del factor de ocupación.', mtWarning, [mbOk], 0);
        Exit;
      end;

      if Trim(FieldByName('fldTiempoTerminal').AsString) = '' then
      begin
        MessageDlg('Hace falta datos del tiempo de terminal.', mtWarning, [mbOk], 0);
        Exit;
      end;
      Next;
    end;
  end;

  // Cálculos para horario óptimo y presentación de resultados.
  edResultados.Lines.Add('Resultados:');
  edResultados.Lines.Add('');

  qryAux := TQuery.Create(Self);
  qryAux.DatabaseName := frmPrincipal.dbRutas.DatabaseName;

  with mryDatosSecciones do
  begin
    First;
    while (FieldByName('fldMaximaDemanda').AsFloat > 0) and (not EOF) do
    begin
      hora := FieldByName('fldTiempoRecorrido').AsString;

      i:=0;
      while i<5 do
```

```
      begin
        if hora[i]=':' then
          hora[i]:='.';
        i:=i+1;
      end;
      hora2:=StrToFloat(hora);

      entero:=Trunc(hora2);
      decimal:=hora2-entero;

      TiempoRecorridoSeccion := (entero * 60) + Trunc(100 * decimal);
      VelocidadOperacion := 60 * (LongitudRuta / TiempoRecorridoSeccion);

      Intervalo := (60 * FieldByName('fldFactorOcupacion').AsFloat * (Asientos + PasajerosPie)) /
FieldByName('fldMaximaDemanda').AsFloat;
      Intervalo := RedondeaIntervalo(Intervalo);

      TiempoCiclo := TiempoRecorridoSeccion + FieldByName('fldTiempoTerminal').AsFloat;

      NumVehiculos := TiempoCiclo / Intervalo;
      NumVehiculos := RedondeaVehiculos(NumVehiculos);

      TiempoCiclo := NumVehiculos * Intervalo;

      TiempoTerminal := TiempoCiclo - TiempoRecorridoSeccion;

      VelComercial := (60 * LongitudRuta) / TiempoCiclo;

      mryDatosSecciones.Edit;
      mryDatosSecciones.FieldByName('fldFrecuencia').AsFloat := Intervalo;
      mryDatosSecciones.FieldByName('fldVehiculos').AsFloat := NumVehiculos;
      mryDatosSecciones.FieldByName('fldVelComercial').AsFloat := VelComercial;

      edResultados.Lines.Add('Sección ' + FieldByName('fldSeccion').AsString + '. ' +
FieldByName('fldPeriodo').AsString);
      edResultados.Lines.Add('Vehículos requeridos: ' + FloatToStr(NumVehiculos));
      edResultados.Lines.Add('Frecuencia de paso: ' + FloatToStr(Intervalo) + ' minutos.');
      edResultados.Lines.Add('Tiempo de Ciclo: ' + FloatToStr(TiempoCiclo) + ' minutos.');
      edResultados.Lines.Add('Tiempo de recorrido: ' + FloatToStr(TiempoRecorridoSeccion) + ' minutos.');
      edResultados.Lines.Add('Tiempo de terminal: ' + FloatToStr(TiempoTerminal) + ' minutos.');
      edResultados.Lines.Add('Velocidad comercial: ' + FloatToStr(VelComercial) + ' km/h.');
      edResultados.Lines.Add('');

      Next;
    end;
  end;
end;
```

/*Función que redondea el intervalo*/

```
function TfrmHorario.RedondeaIntervalo(Intervalo : Real): Real;
var
  Valores: array[0..14] of Real;
  i : Integer;
  j : Real;
begin

  Valores[0] := 1;
  Valores[1] := 2;
  Valores[2] := 3;
  Valores[3] := 4;
  Valores[4] := 5;
  Valores[5] := 6;
  Valores[6] := 7.5;
  Valores[7] := 10;
  Valores[8] := 12;
  Valores[9] := 15;
  Valores[10] := 20;
  Valores[11] := 30;
  Valores[12] := 40;
  Valores[13] := 45;
  Valores[14] := 60;

  i := 0;
  j := 0;
  while i<15 do
  begin
    if Intervalo>Valores[i] then
      j := Valores[i];
    i := i+1;
  end;

  result := j;
end;
```

/*Función que redondea el número de vehículos calculado*/

```
function TfrmHorario.RedondeaVehiculos(Vehiculos : Real): Real;
var
  Entero : Integer;
  Fraccion : Real;
begin
  Entero := Trunc(Vehiculos);
  Fraccion := Vehiculos - Entero;

  if Fraccion > 0 then
    Entero := Entero + 1;

  result := Entero
end;
```

Capítulo 10
Pruebas del sistema

Las pruebas realizadas al sistema fueron:

1. Pruebas de caja negra. Pruebas para verificar que las funcionalidades del sistema se realizan correctamente, verificando los resultados esperados con los resultados obtenidos. Se realizaron partiendo de pruebas a cada función del código, verificando que el resultado fuera el esperado, posteriormente se hicieron pruebas de funcionalidad a conjuntos de funcionalidades, por ejemplo las funcionalidades de cada pantalla del sistema y se siguió subiendo de nivel hasta llegar a una prueba de caja negra de cada funcionalidad del sistema, representada por cada escenario de uso definida en la sección de requerimientos.

2. Pruebas de caja blanca. Pruebas para verificar que se realizan los caminos principales de la lógica del programa.

Estos dos tipos de pruebas se realizaron para cada unidad del sistema y para la integración de las unidades.

3. También se realizó la verificación de la interfaz gráfica de usuario.

10.1 Pruebas de caja negra y caja blanca

Se realizaron las pruebas de funcionalidad y de seguimiento de los caminos principales de la lógica del programa, de unidad y de integración, a las siguientes funcionalidades:

- Creación, modificación y eliminación de una ruta
- Creación, modificación y eliminación de estaciones en una ruta
- Creación, modificación y eliminación de un tipo de unidad de transporte
- Asignación o cambio de un tipo de unidad de transporte a una ruta
- Creación de temporadas de servicio
- Creación de tipos de día de servicio dentro de las temporadas
- Creación de horarios de servicio dentro de los tipos de día
- Captura de los datos de los muestreos
- Generación de gráficas de demanda para una ruta, por temporada, tipo de día o sección de un día

10.2 Verificación de la interfaz gráfica de usuario

Se realizó la verificación de la interfaz de usuario para las siguientes pantallas:
- Pantalla principal
- Creación de una ruta
- Edición de una ruta
- Eliminación de una ruta (mensaje de confirmación)
- Agregar una estación a una ruta
- Modificar una estación de una ruta
- Eliminar una estación de una ruta (mensaje de confirmación)
- Catálogo de vehículos de una ruta
- Agregar un tipo de vehículo
- Modificar un tipo de vehículo
- Eliminación de un tipo de vehículo (mensaje de confirmación)
- Cambio del tipo de vehículo de una ruta
- Forma para captura de un aforo o muestreo
- Gráfica de demanda de una ruta
- Gráfica de demanda de una temporada
- Gráfica de demanda de un tipo de día

- Forma para creación de horarios de servicio
- Opciones de creación de temporadas, tipos de día o secciones
- Creación de una temporada
- Creación de un tipo de día
- Creación de una sección de un tipo de día

10.3 Experimento simulado

Por último se realizó la validación del método para calcular el horario óptimo a través de la toma de datos de la ruta. El experimento consistió en realizar muestreos a una ruta de transporte público, la introducción en el sistema de los datos recolectados en los muestreos, la creación de la ruta en el sistema, la definición de sus estaciones, tipo de vehículo, temporadas, tipos de día y secciones y la creación del horario óptimo con base en los datos introducidos. Este experimento se explica en la sección 11 Experimento simulado, que trata de un ejemplo de utilización del sistema con datos reales de una ruta y los muestreos realizados en la misma para alimentar al sistema y generar un horario.

Capítulo 11
Experimento simulado

11.1. Introducción

El método para calcular el horario óptimo en una ruta de transporte, descrito en este trabajo debe ser validado de forma experimental, es decir, se tiene que aplicar en una ruta de transporte real, a lo largo de todo el periodo de tiempo que dure el servicio y a lo largo de todo el horario de trabajo. Sin embargo, tomar los suficientes muestreos a lo largo de todo el periodo de tiempo y a lo largo de todo un día de trabajo de una ruta de transporte con el fin de poder identificar la división del comportamiento de la ruta en temporadas, tipos de día y secciones de los tipos de día requiere de recursos de tiempo y personas con los que actualmente no se cuenta y tardaría mucho tiempo. Por estas circunstancias, la validación experimental del método a través del estudio del comportamiento completo de una ruta está fuera del alcance de esta tesis. A pesar de esta situación, se ha propuesto una simulación que prueba el comportamiento del método, tomando como base los datos de una sección de tiempo del servicio con un alta demanda fácilmente identificable, en una de las rutas del transporte interno gratuito de la UNAM.

El punto más importante del método que tiene que ser validado es la frecuencia de paso óptima que implique la mínima cantidad de unidades de transporte necesaria para satisfacer la demanda existente.

A continuación se muestra un ejemplo de uso del sistema, a partir de los datos de un muestreo realizado sobre la ruta 1 del sistema de transporte interno gratuito de la UNAM.

11.2. Datos de la ruta utilizada para la validación del sistema

La ruta tomada como base para la validación parcial del sistema es la ruta 1 del transporte interno gratuito de la UNAM. Tiene las siguientes características:

Terminal	Distancia desde la terminal de salida [km.]
Metro Universidad (salida)	0
CENDI	0.529
Psiquiatría	1.180
Química	1.524
CELE	1.932
Ingeniería	2.402
Arquitectura	2.676
Rectoría	3.193
Psicología	3.421
Filosofía y Letras	3.617
Derecho	3.920
Economía	4.133
Odontología	4.400
Medicina	4.800
Veterinaria	5.633
Instituto de Geofísica	6.038
Química (Conjunto D y E)	6.492
Tienda UNAM	6.907
Metro Universidad (llegada)	7.200

Longitud de la ruta: 7.2 km.

Tipo de unidad de transporte: microbús. Capacidad: 40 asientos, 20 personas de pie.

11.3. Temporada, tipo de día y sección utilizadas para la toma de datos de muestra

Se escogió la temporada, tipo de día y sección del día con base en la disponibilidad de tiempo para realizar los muestreos y con el objetivo de tomar datos en un horario del servicio en que se identificara una hora pico importante, es decir, un horario con alta demanda de servicio. También se buscó tomar datos en un horario con un servicio que se notara que no fuera óptimo. La temporada escogida fue durante el mes de agosto, el tipo de día fue identificado como “de lunes a viernes” y la sección del día se identificó como “de 6:30 a.m. a 8:00 a.m.”

11.4. Características del servicio ofrecido actualmente por la ruta

El servicio que actualmente ofrece la ruta en la temporada, tipo de día y sección del día escogidos tiene las siguientes características:

- La terminal de salida de la ruta 1 es la terminal de salida y de llegada de 10 rutas incluyendo la ruta 1.
- La frecuencia de salida y de paso de las unidades de transporte a lo largo de cada ruta y en nuestro caso particular de la ruta 1, es irregular. Se observaron intervalos de tiempo entre dos salidas sucesivas de entre 1 minuto y 15 minutos, siendo las más comunes entre los 10 y 15 minutos y las menos comunes entre los 1 y 10 minutos.
- Durante el periodo de tiempo escogido en la terminal de salida, después de la salida de una unidad de transporte, la cantidad de personas que aún había en la cola y tuvieron que esperar a la siguiente unidad era suficiente para ocupar entre 1 y 2 unidades de transporte más y la espera tomaba un tiempo considerable. Tomando en cuenta esto y las frecuencias de servicio anotadas en el punto anterior, se observa una demanda del servicio muy alta y una frecuencia de salida de las unidades que no es suficiente para cumplir con la demanda.
- Se observa que las unidades de transporte van llenas al máximo, e incluso van con sobrecarga. Esto ocasiona una mala calidad del servicio al ser incómodo e incluso peligroso para las personas ocupar una unidad de transporte que está más llena de lo que debería y debido a las dificultades al subir y bajar para las personas tanto dentro de las unidades de transporte como las que esperan en las paradas, por el sobrecupo que tiene la unidad de transporte.
- También se observó, sobre todo en la terminal de salida, que muchas personas al ver la gran cantidad de personas esperando por el servicio de transporte, prefieren caminar hacia los lugares a los que se dirigen o bien utilizar un servicio de taxis que se encuentra enfrente de las salidas de las rutas del transporte gratuito.

En conclusión, además de no satisfacer la demanda de servicio existente debido a las frecuencias irregulares y en general insuficientes, el servicio se da en condiciones de mala calidad y peligro debido a la poca comodidad y seguridad de ir en una unidad con sobrecupo. Además, a lo largo de la ruta, puede suceder que la gente que está esperando en las paradas, además de tener que esperar un rango de tiempo que puede ir de 1 minuto a 20 minutos, no pueda subir a la unidad de transporte por estar llena.

11.5. Uso del sistema para obtener el servicio óptimo

Se comenzó por realizar muestreos consistentes en subirse a una unidad de transporte en la ruta 1, en la terminal de salida y en cada parada tomar los siguientes datos:

- Hora de llegada a la parada
- Número de ascensos
- Número de descensos
- Número de personas que no pudieron subir y se quedaron esperando a la siguiente unidad
- Hora de partida de la parada

A continuación se muestra cómo se usó el sistema para introducir los datos recolectados en los muestreos y obtener los dos datos que se necesita saber para dar un servicio óptimo de transporte en la ruta 1. Estos datos son el número óptimo de unidades de transporte necesarias para satisfacer la demanda y las frecuencias de paso necesarias en esa sección del día para dar un servicio que satisfaga la demanda sin que esta se sobre-acumule en las paradas.

Se comenzó por definir las características de la ruta dentro del sistema:

11.5.1. Pantalla de inicio del sistema

Al entrar al sistema aparece la pantalla que se muestra en la figura 11.1:

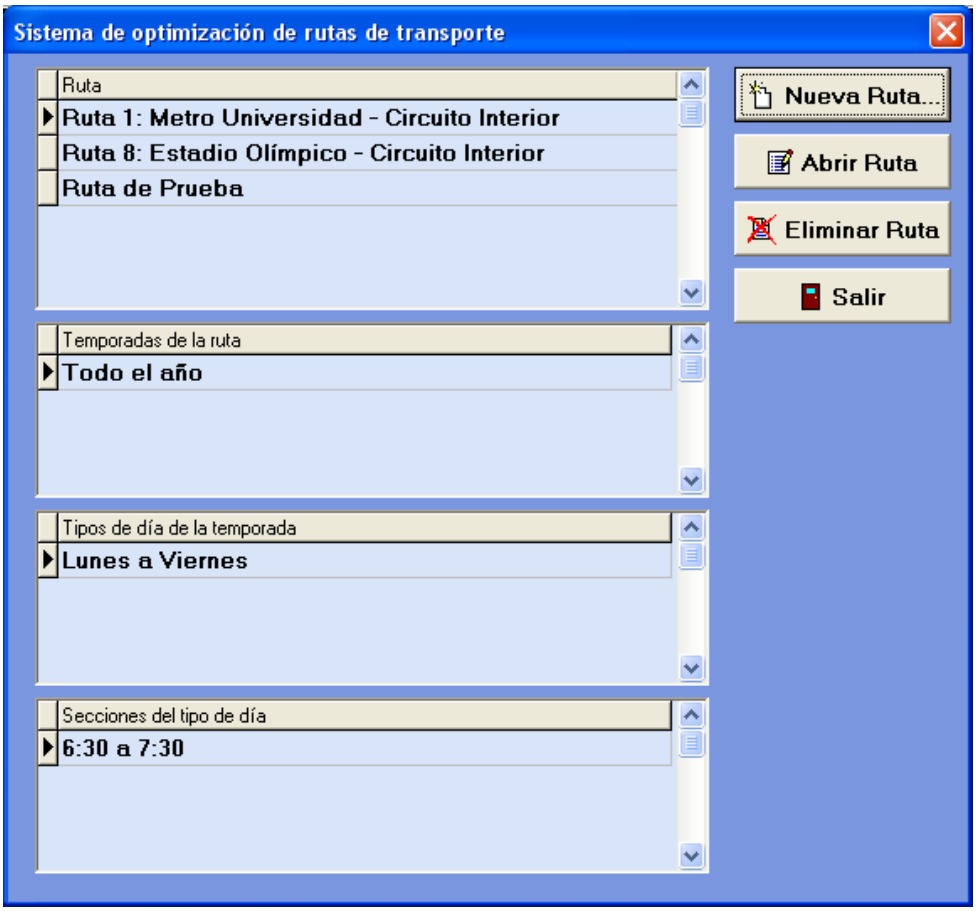


Fig. 11.1 Pantalla de inicio del sistema

11.5.2. Creación de la ruta nueva e introducción de los datos de horario, estaciones y tipo de unidad de transporte de la ruta

Haciendo clic en “Nueva Ruta…” e introduciendo los datos de la ruta aparece la pantalla que se muestra en la figura 11.2:

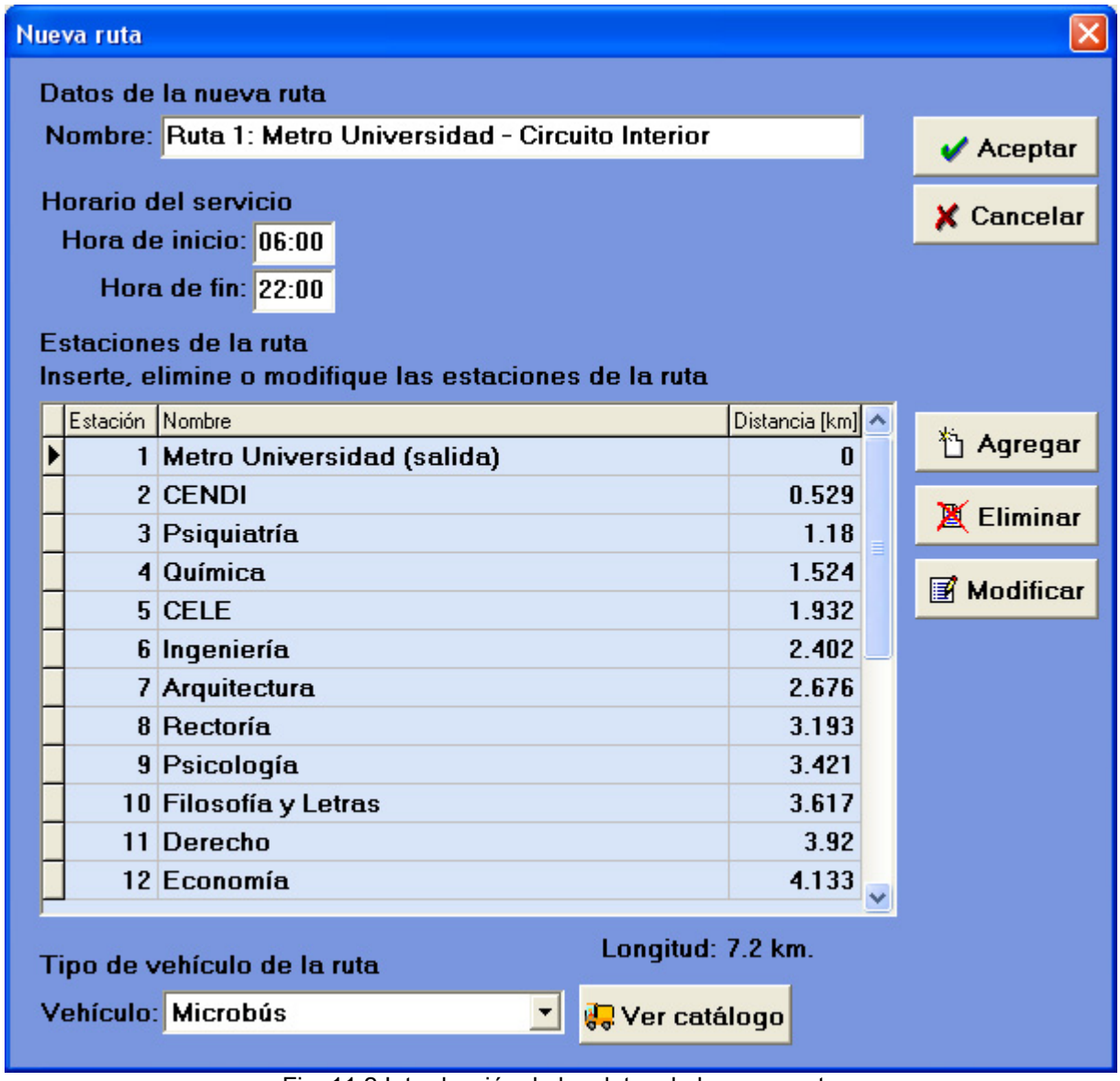


Fig. 11.2 Introducción de los datos de la nueva ruta

11.5.3. Introducción de las estaciones y de la unidad de transporte

Las estaciones se introdujeron haciendo clic en “Agregar” en la sección de estaciones de la ruta y utilizando la interfaz gráfica que se muestra en la figura 11.3 a continuación:

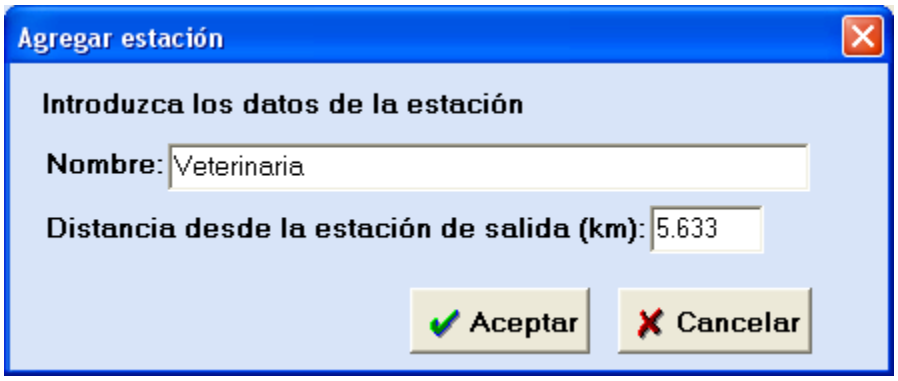


Fig. 11.3 Pantalla para introducción de las estaciones

El sistema toma cada estación introducida y ordena el total de estaciones respecto a la distancia desde la terminal de salida.

Para introducir el tipo de unidad de transporte si éste no existe se hace clic en “Ver catálogo” en la sección de tipo de transporte y aparece la pantalla que se muestra en la figura 11.4:

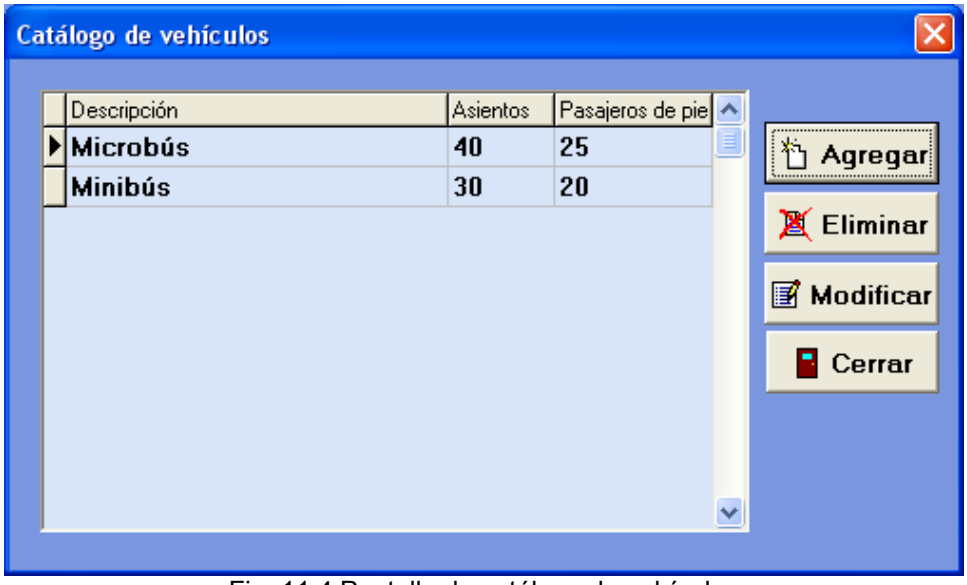


Fig. 11.4 Pantalla de catálogo de vehículos

Y haciendo clic en “Agregar” se introduce el tipo de unidad de transporte en la pantalla que aparece en la figura 11.5:

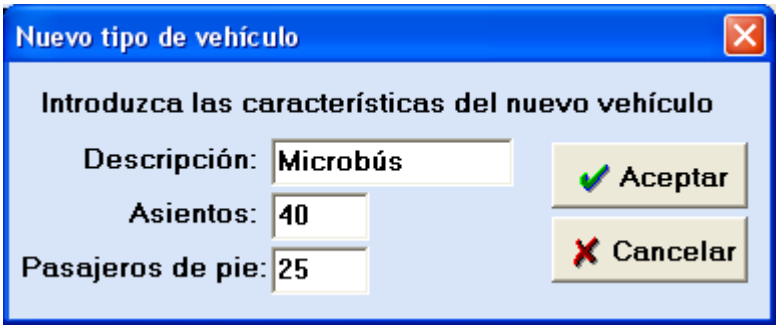


Fig. 11.5 Introducción de los datos del nuevo tipo de vehículo

Al dar de alta el tipo de unidad y cerrar la pantalla con el catálogo de vehículos, regresamos a la pantalla de Nueva Ruta (figura 11.2) en la que originalmente nos encontrábamos. En esa pantalla hacemos clic en “Aceptar” y con esto queda guardada en el sistema la definición de las características generales de la nueva ruta.

La pantalla de la ruta queda como se ve en la figura 11.6:

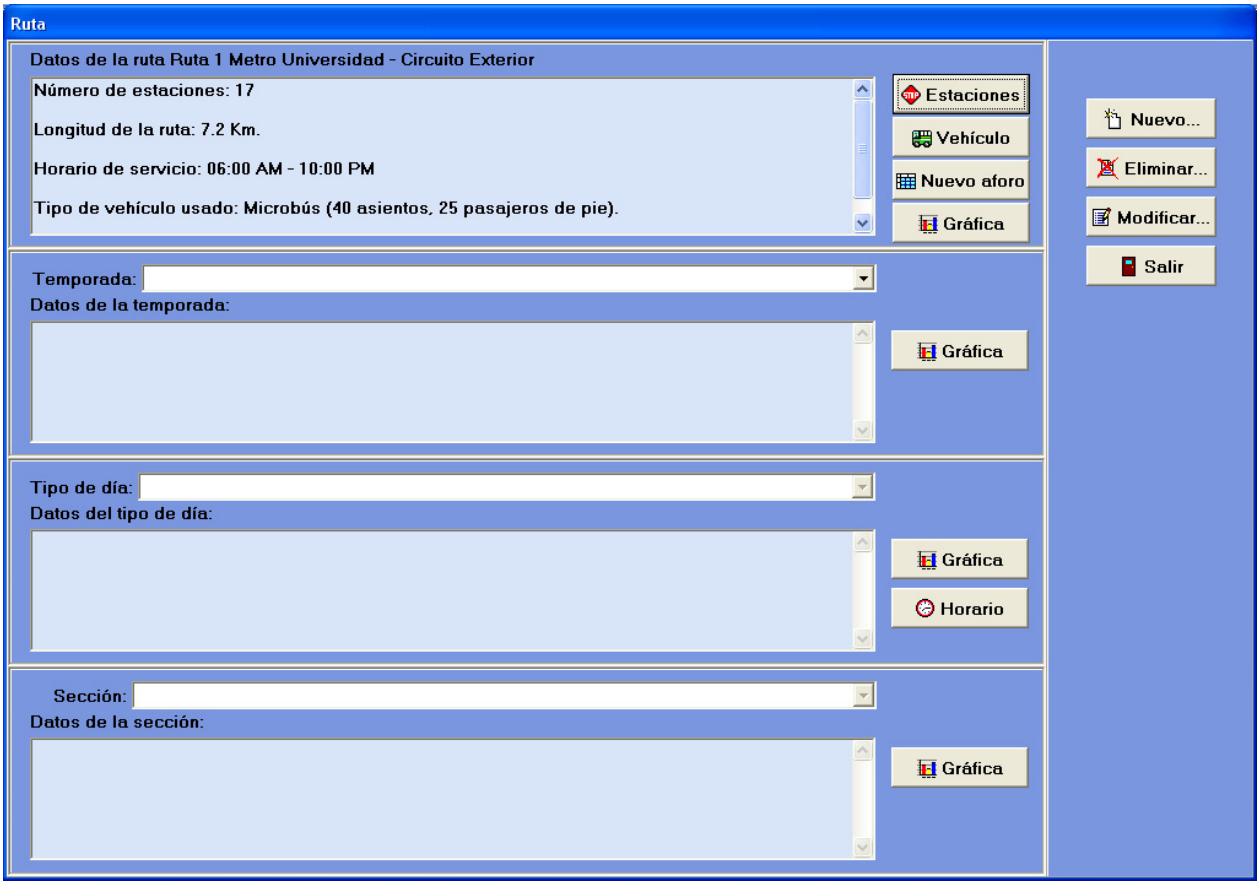


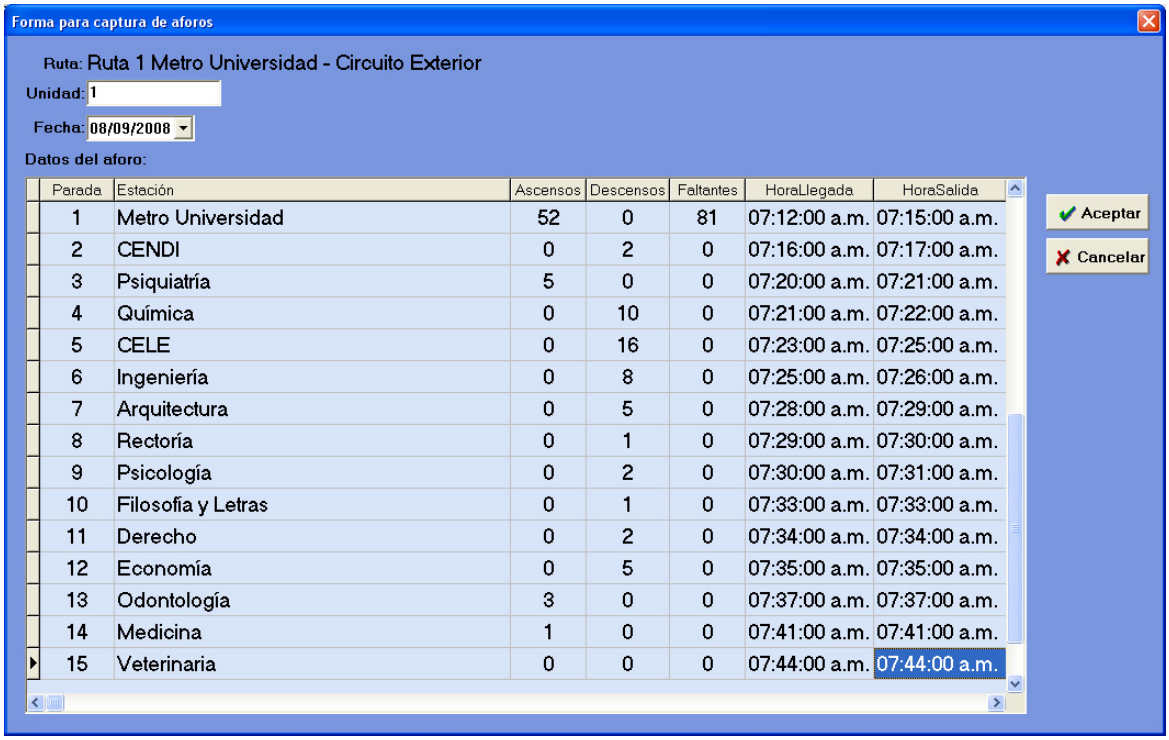
Fig. 11.6 Pantalla de la ruta

En cualquier momento se puede modificar las estaciones y vehículo de la ruta haciendo clic en el botón correspondiente.

La ruta está lista ahora para introducir los muestreos y separar el horario de servicio en temporadas, tipos de día y secciones. A continuación realizaremos dichas operaciones.

11.5.4. Introducción de los muestreos hechos a la ruta

Haciendo clic en “Nuevo Aforo” el sistema nos muestra una pantalla como la que se ve en la figura 11.7, donde podemos introducir los muestreos hechos a la ruta:



Parada	Estación	Ascensos	Descensos	Faltantes	HoraLlegada	HoraSalida
1	Metro Universidad	52	0	81	07:12:00 a.m.	07:15:00 a.m.
2	CENDI	0	2	0	07:16:00 a.m.	07:17:00 a.m.
3	Psiquiatría	5	0	0	07:20:00 a.m.	07:21:00 a.m.
4	Química	0	10	0	07:21:00 a.m.	07:22:00 a.m.
5	CELE	0	16	0	07:23:00 a.m.	07:25:00 a.m.
6	Ingeniería	0	8	0	07:25:00 a.m.	07:26:00 a.m.
7	Arquitectura	0	5	0	07:28:00 a.m.	07:29:00 a.m.
8	Rectoría	0	1	0	07:29:00 a.m.	07:30:00 a.m.
9	Psicología	0	2	0	07:30:00 a.m.	07:31:00 a.m.
10	Filosofía y Letras	0	1	0	07:33:00 a.m.	07:33:00 a.m.
11	Derecho	0	2	0	07:34:00 a.m.	07:34:00 a.m.
12	Economía	0	5	0	07:35:00 a.m.	07:35:00 a.m.
13	Odontología	3	0	0	07:37:00 a.m.	07:37:00 a.m.
14	Medicina	1	0	0	07:41:00 a.m.	07:41:00 a.m.
15	Veterinaria	0	0	0	07:44:00 a.m.	07:44:00 a.m.

Fig. 11.7 Introducción de un aforo o muestreo de la ruta

Al hacer clic en “Aceptar” el sistema guarda los datos del aforo y regresa a la pantalla de la ruta. Se hace lo mismo para todos los muestreos realizados.

A partir de ahora es necesario dividir el servicio en temporadas, tipos de día y secciones del día. Para esto uno puede basarse en el comportamiento del servicio observado en los muestreos de demanda que se hagan o como en este caso particular, ya se tiene una división observada en la experiencia de la ruta, esta división se define en el sistema y luego se introducen los muestreos hechos en la sección definida.

11.5.5. Definición de la temporada de servicio

Haciendo clic en “Nuevo…” en la pantalla de la ruta (figura 11.2) y escogiendo la opción “Nueva temporada” aparece una pantalla como la que se muestra en la figura 11.8:

Fig. 11.8 Creación de una nueva temporada

Al hacer clic en “Aceptar” el sistema regresará a la pantalla de la ruta y mostrará la nueva temporada creada.

11.5.6. Definición del tipo de día

Haciendo clic en “Nuevo…” en la pantalla de la ruta y escogiendo la opción “Nuevo tipo de día” aparece la pantalla mostrada en la figura 11.9:

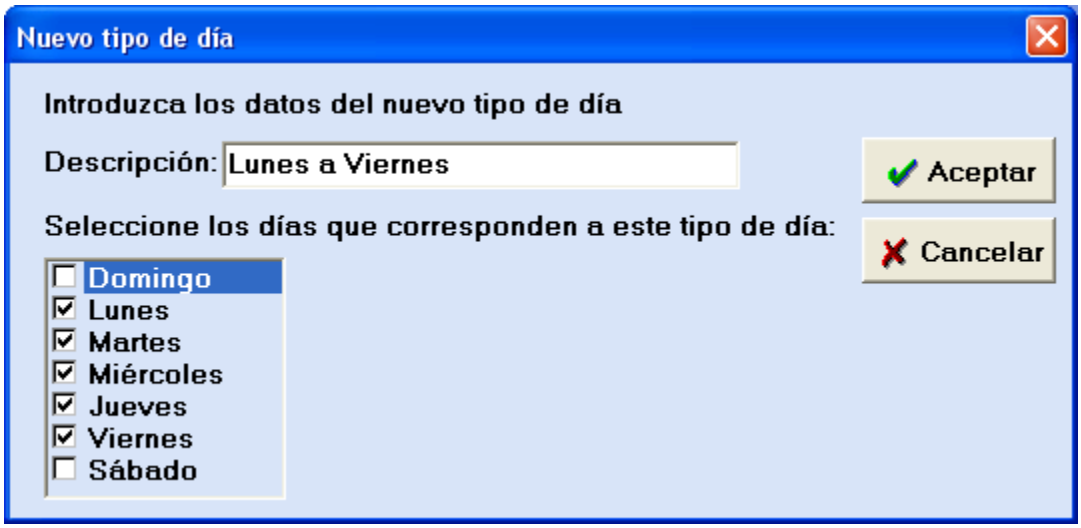


Fig. 11.9 Creación de un nuevo tipo de día

Al hacer clic en “Aceptar” el sistema regresará a la pantalla de la ruta y mostrará el nuevo tipo de día creado.

11.5.7. Definición de la sección del tipo de día

Haciendo clic en “Nuevo…” en la pantalla de la ruta y escogiendo la opción “Nueva sección” aparece la pantalla mostrada en la figura 11.10:

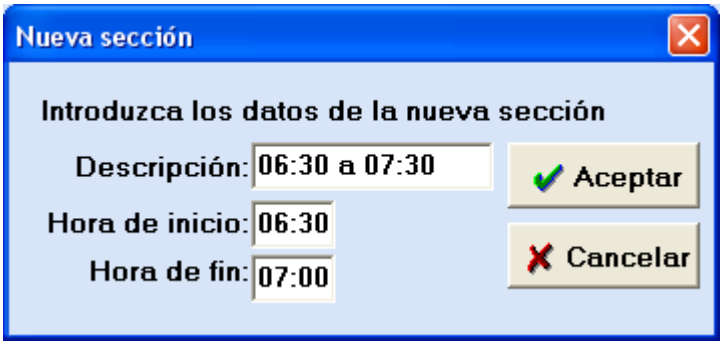


Fig. 11.10 Creación de una nueva sección de un tipo de día

Al hacer clic en “Aceptar” el sistema regresará a la pantalla de la ruta y mostrará la nueva sección creada.

Los datos de la ruta después de haber creado la temporada, tipo de día y sección se muestran en la pantalla de la ruta como se muestra en la figura 11.11:

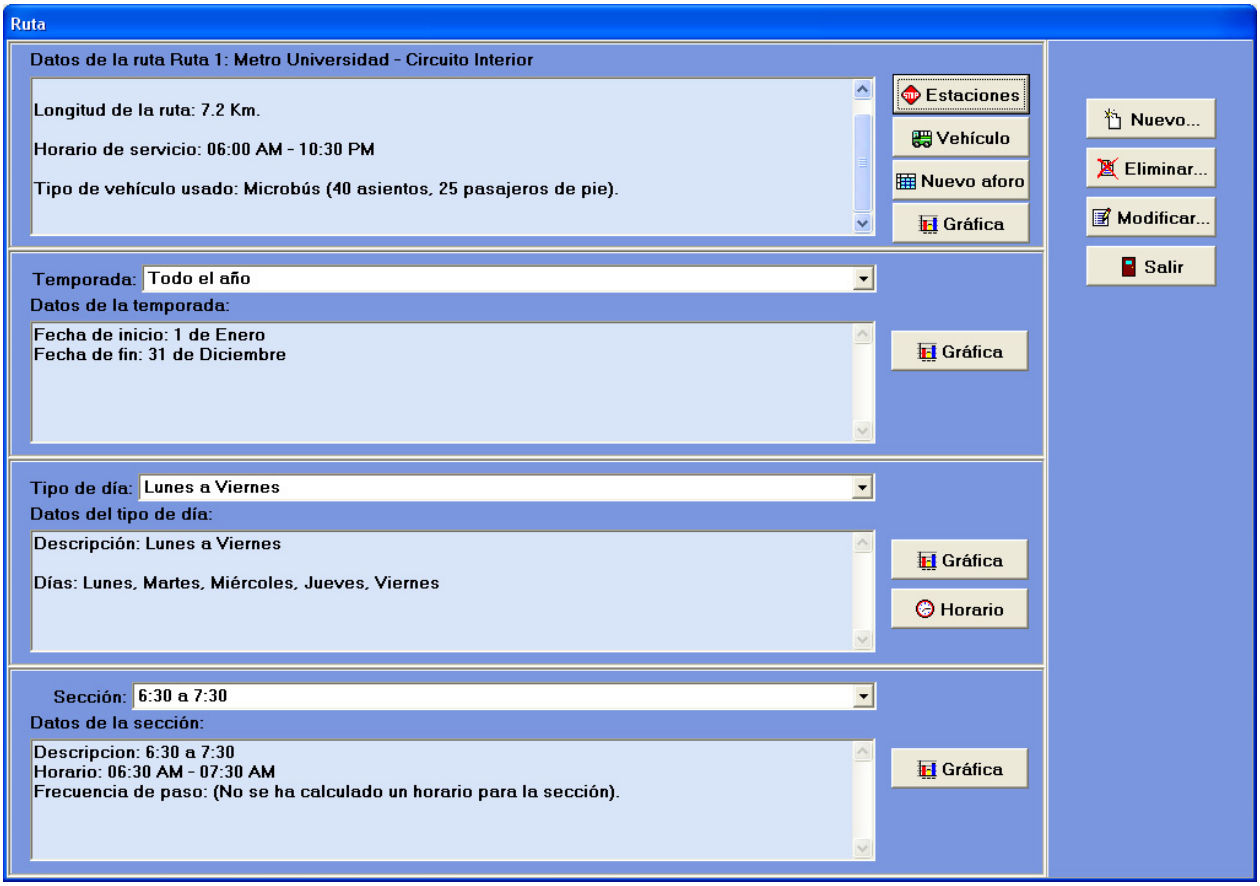


Fig. 11.11 Pantalla de la ruta con la temporada, tipo de día y sección creados

Habiendo definido los datos de la ruta dentro del sistema, así como la división del servicio en temporadas, tipos de día y secciones del día y habiendo introducido los muestreos hechos, se puede definir un horario de servicio óptimo para la sección del tipo de día creada.

11.5.8. Definición del horario (frecuencia de paso) para la sección

En la siguiente pantalla aparecen los datos de la sección para la cual se quiere definir el horario óptimo. El sistema, con base en los muestreos introducidos, calcula el tiempo que toma recorrer toda la ruta durante esa sección y el volumen de diseño, el cual representa la máxima demanda existente en esa temporada, tipo de día y sección. Ésta es la demanda para la cual se busca el número óptimo de unidades de servicio y la frecuencia de paso óptima que la satisfaga.

Introduciremos el factor de ocupación, el cual significa el porcentaje de cupo que deseamos que tengan las unidades de transporte, es decir, qué tan llenas deseamos que se encuentren. A mayor porcentaje de cupo, más llenas irán las unidades y se necesitará un menor número de ellas, pero el servicio será menos cómodo y con menor posibilidad para encontrar un asiento disponible. En este caso introducimos un 1 o sea 100% de cupo de las unidades. También introducimos el tiempo de terminal, el cual es el tiempo mínimo que se desea que la unidad de transporte permanezca en la terminal antes de ser puesto de nuevo en circulación en la ruta. En nuestro caso el valor de ese parámetro es 0 minutos.

Al hacer clic en “Aceptar” el sistema calculará el número de vehículos requeridos y la frecuencia de paso que satisface la demanda calculada anteriormente por el sistema y tomando en cuenta el factor de ocupación y tiempo de terminal deseados. Este resultado se muestra en la figura 11.12, en el cuadro de texto de resultados:

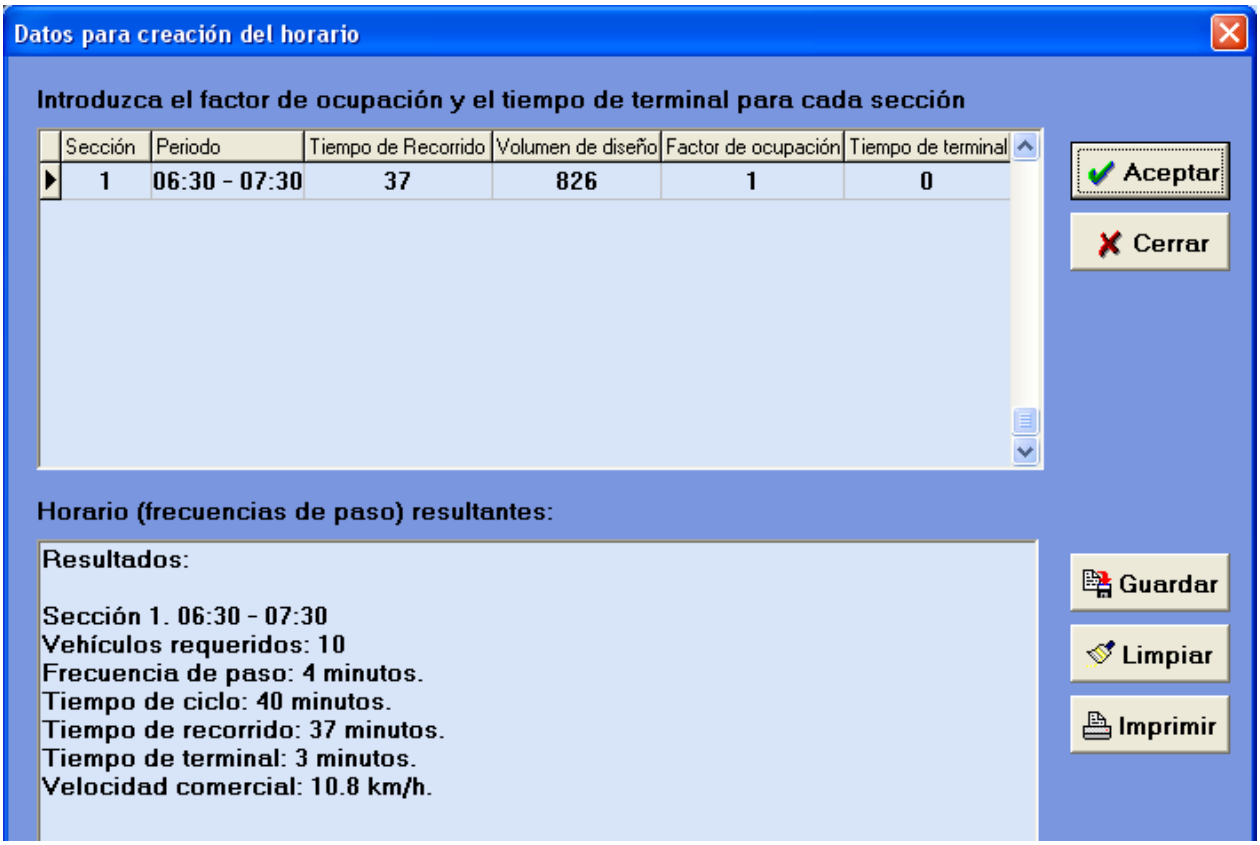


Fig. 11.12 Pantalla de creación de un horario

Podemos modificar el factor de ocupación y el tiempo de terminal y dar clic de nuevo en “Aceptar” para observar cómo cambia el número de vehículos requeridos y la frecuencia de paso en la sección, en función de los nuevos valores introducidos.

Al hacer clic en “Guardar” el sistema guardará el último valor calculado para el número de vehículos requeridos y la frecuencia de paso. Al hacer clic en “Cerrar” el sistema regresará a la pantalla de la ruta.

Después de lo anterior la pantalla de la ruta queda como en la figura 11.13:

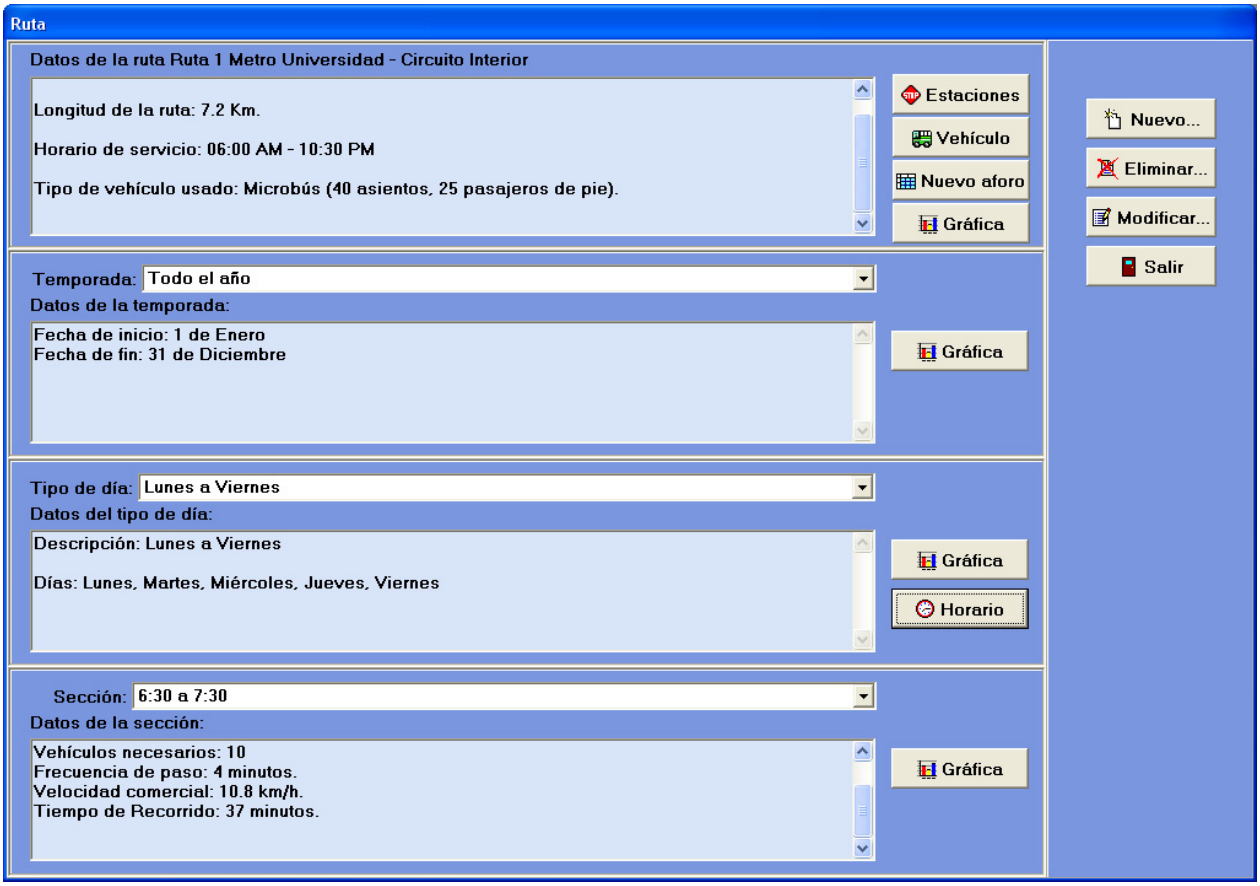


Fig. 11.13 Pantalla de la ruta con datos del horario creado

De existir más temporadas, tipos de día y secciones, estas pueden ser escogidas a través de los combos y de esa forma se puede calcular el horario óptimo para cada sección, para de esta manera saber cuándo cambiar la frecuencia de salida de las unidades de transporte.

En el caso de nuestro ejemplo, podemos notar la diferencia entre el servicio real actualmente provisto por la ruta de transporte público, con su alta demanda sobre todo en los horarios pico, la frecuencia de paso irregular de las unidades de transporte y con la gente que se queda esperando en las paradas sin saber cuánto tiene que esperar (de 1 a 20 minutos aproximadamente) para tomar una unidad de transporte, en comparación con el servicio óptimo propuesto por el sistema, que permite saber que la frecuencia de paso es regular y aceptable (cada 4 minutos). También se puede notar la diferencia entre administrar la frecuencia de paso de las unidades de transporte “a ojo” o con base solamente en la demanda de servicio existente en la estación de salida, sin tomar en cuenta la demanda que pudiera existir en las demás estaciones, a hacerlo apoyándose en el sistema para saber qué frecuencia de paso es adecuada para satisfacer la demanda de servicio a lo largo de toda la ruta.

Desde el punto de vista de manejo de la información se puede notar una mejora ya que es posible almacenar en el sistema rápida y fácilmente los datos recolectados en los aforos. Asimismo, no será necesario hacer un análisis “manual” de los datos recolectados para conocer el comportamiento de la demanda de servicio en las temporadas, tipos de día y secciones, ni será necesario calcular de manera manual el horario de servicio adecuado (frecuencia óptima de paso de las unidades).

De implementarse un sistema como el mostrado en este trabajo en una ruta de transporte público para dar el servicio óptimo, sería necesario monitorear el comportamiento de la ruta de transporte posterior a la

implementación, ya que si la gente observa que el servicio de transporte mejora, podría aumentar la demanda al sumarse las personas que actualmente prefieren irse caminando o las que usan el servicio de taxis que se encuentra frente a la terminal de salida. Esto haría necesario hacer muestreos del nuevo comportamiento de la demanda y el servicio y calcular un nuevo servicio óptimo.

Capítulo 12
Resultados y conclusiones

Resultados

Con la construcción del sistema de optimización de rutas de transporte se obtuvieron los siguientes resultados, que colaboran a satisfacer las necesidades y requerimientos de negocio de una organización dedicada a proveer el servicio de transporte:

El sistema provee las siguientes funcionalidades:

- Creación, modificación y eliminación de una ruta.
    - Definición, modificación y eliminación de las estaciones.
    - Definición del horario de servicio.
    - Definición del tipo de unidad de transporte.
- Creación, modificación y eliminación de tipos de unidades de transporte.
- Captura y almacenamiento de los datos de los muestreos.
- División del comportamiento del servicio en temporadas, tipos de día y secciones de día.
- Análisis del servicio mediante gráficas de demanda para cada temporada, tipo de día y sección.
- Generación de horarios óptimos para cada sección de día.

A través de estas funcionalidades el sistema permite resolver los siguientes problemas que caracterizan a una mala administración de una ruta de transporte:

- Incapacidad para satisfacer adecuadamente la demanda de transporte existente debido a que:
    - No se conoce el comportamiento de la demanda (temporadas, días y horas de alta y baja demanda en un día).
    - No se conoce la cantidad de unidades de transporte necesarias para satisfacer la demanda.
    - No se conoce la frecuencia de paso óptima para satisfacer la demanda.
- Frecuencias de paso irregulares, poco confiables.
- Tiempos de espera demasiado largos en las paradas.
- Unidades de transporte demasiado llenas o casi vacías, producto de las frecuencias de paso irregulares.
- Incomodidad e inseguridad derivadas del uso de las unidades de transporte demasiado llenas.
- Falta de un método formal para el análisis de muestreos y cálculo de las frecuencias de paso óptimas.
- Falta de una manera rápida y confiable de capturar, almacenar y analizar los datos de los muestreos.
- Falta de una manera rápida y confiable de dividir el comportamiento de la demanda en temporadas, tipos de día y secciones de día.
- Servicio de transporte de mala calidad, provocando malestar y preferencia por el uso de otros medios, entre los que se encuentran los vehículos privados, lo cual a su vez provoca un aumento en el tránsito y congestiones viales.

El sistema ayuda a satisfacer las siguientes necesidades y objetivos del servicio de transporte:

- Contar con una manera rápida de capturar, almacenar y analizar los datos de los muestreos.
- Contar con una manera rápida y confiable de conocer las variaciones en la demanda de transporte.
- Satisfacer la demanda de transporte a lo largo de la ruta y en sus distintas variaciones.
    - Contar con frecuencias de paso aceptables y regulares.
    - Conocer la cantidad necesaria de unidades de transporte que satisfarán la demanda.
- Optimizar los costos de operación de la ruta al conocer la cantidad óptima necesaria de unidades de transporte.

En el desarrollo del sistema se utilizaron los siguientes elementos de administración de proyectos e ingeniería de software:

- Se definió el ciclo de vida espiral con base en las características de los requerimientos funcionales y de calidad y las restricciones del proyecto.
- Se realizó la estimación y planeación de las actividades a realizarse durante el proyecto.
- Se realizó el análisis de factibilidad y de riesgos para saber si sería posible realizar el proyecto, y para mitigar cualquier problema que pudiera surgir durante el mismo.
- Para la especificación de los requerimientos se utilizaron escenarios de uso, con el fin de capturar, comprender mejor y reflejar el comportamiento del sistema desde el punto de vista del usuario, la interacción de éste con el sistema, y como base para la definición de pruebas de aceptación.
- Se consultó bibliografía sobre administración de rutas de transporte y se definió una manera para analizar los datos de los muestreos y un método para generar las frecuencias de paso óptimas de las unidades de transporte, con base en las prácticas recomendadas en la literatura y adaptadas para las necesidades específicas del sistema.
- Para el análisis de los requerimientos se utilizaron diagramas de flujo de datos, una manera estructurada de análisis que resulta en una buena base para el diseño y construcción del sistema. También del análisis surgió un modelo de datos que captura las entidades que intervienen en el funcionamiento del sistema.
- La arquitectura del sistema fue establecida con base en las necesidades y el comportamiento requeridos. Se utilizó un modelo centrado en los datos para hacer más fácil y seguro el manejo de la información de las rutas, así como un modelo de llamada/retorno para facilitar el diseño y la implementación de los escenarios de uso definidos.
- Se dividió el sistema en módulos y funcionalidades y se analizaron las relaciones entre todas estas partes para asegurar un buen manejo y una buena comprensión de la complejidad del sistema y para administrar adecuadamente su implementación, pruebas e integración.
- La interfaz gráfica de usuario se diseñó con base en el comportamiento que se requirió que tuviera el sistema, reflejado en los escenarios de uso, buscando cumplir con la usabilidad requerida y con base en patrones de diseño de interfaces de usuario recomendados en la literatura.
- Se diseñó una base de datos para asegurar un almacenamiento sencillo y confiable de la información del sistema, así como para asegurar la integridad de los datos al ser manejados. El diseño de datos viene acompañado de un diccionario de datos con la descripción de cada entidad.
- Se realizó el diseño detallado a través de pseudocódigo y diagramas de flujo, para tener un diseño y una comprensión aún más completos de las funcionalidades del sistema y facilitar con esto la implementación de las mismas en código.
- La implementación se llevó a cabo utilizando Delphi 5 como lenguaje de programación, lo cual facilitó realizar la implementación de los escenarios de uso, cumplir con la usabilidad requerida y con la arquitectura establecida de llamada/retorno a través de formularios.
- La implementación de la base de datos se llevó a cabo utilizando SQL Server 2000, lo cual permitió cumplir con la arquitectura centrada en los datos establecida para el sistema y el aseguramiento de la integridad requerida para los datos, a través de la integridad referencial que es posible implementar mediante SQL Server. También ayudó a cumplir con el requerimiento de facilidad de almacenamiento y uso de los datos y una interfaz adecuada con Delphi 5 a través del uso de procedimientos almacenados.
- Las pruebas realizadas al sistema consistieron en pruebas a la funcionalidad en sus distintas fases: unitarias para funcionalidades individuales; de integración para ir uniendo las distintas funcionalidades, de sistema para probar módulos completos y el sistema como un todo y de aceptación para probar los escenarios de uso establecidos. Las técnicas utilizadas para las pruebas fueron de caja blanca para probar el seguimiento adecuado de los flujos de las funcionalidades, y de caja negra para probar conjuntos de funcionalidades, módulos y escenarios de uso.

- Todo el desarrollo realizado durante el proyecto fue documentado y se mantuvo esta documentación actualizada para reflejar el último estado del proyecto y del sistema.

En resumen, en este trabajo se mostraron los siguientes elementos teóricos:

- Administración de proyectos.
  - Definición del ciclo de vida.
  - Estimación, planeación y seguimiento de las actividades del proyecto.
  - Análisis de factibilidad y de riesgos.

- Ingeniería de software.
  - Levantamiento y análisis de requerimientos.
  - Arquitectura y diseño de alto nivel.
  - Diseño detallado y codificación.
  - Pruebas.
  - Diseño e implementación de bases de datos.

- Administración de rutas de transporte .
  - Análisis de los muestreos.
  - Análisis de variaciones en la demanda y división en horas de alta y baja demanda.
  - Análisis y división del comportamiento de la demanda en temporadas, tipos de día y secciones de día.
  - Generación de horarios óptimos para satisfacer la demanda existente.

los cuales en conjunto sirvieron para analizar la problemática, las necesidades y los objetivos involucrados en la construcción de un sistema de optimización de rutas de transporte que sirve para mejorar, facilitar y hacer más rápida la administración del servicio de una ruta de transporte optimizando el uso de las unidades de transporte al establecer frecuencias de servicio dependientes de la demanda existente y logrando con esto brindar un servicio adecuado.

Posteriormente se mostró el uso de estos elementos teóricos, aplicados a la solución del problema. Se utilizaron las técnicas descritas de levantamiento y análisis de requerimientos basados en la problemática existente, establecimiento de una arquitectura del sistema y los modelos de diseño, diseño detallado de la funcionalidad requerida, diseño e implementación de bases de datos y por último la construcción del sistema y la realización de las pruebas. Esto permitió desarrollar el sistema asegurando su calidad y buen funcionamiento, cumpliendo con las necesidades y objetivos establecidos de definir y administrar una ruta, facilitar el almacenamiento y análisis de la información de los muestreos, la creación de temporadas, tipos de día y secciones de día y la creación de horarios para las rutas, con el fin de satisfacer el comportamiento cambiante de la demanda de transporte. Por último, al utilizar técnicas adecuadas de planificación y monitoreo de proyectos, así como de administración de riesgos, fue posible estimar los recursos técnicos, humanos y de tiempos necesarios para el análisis, desarrollo e implementación del sistema, así como también fue posible monitorear, prevenir y resolver los posibles problemas que pudieran surgir a lo largo del proyecto.

El producto de todo lo anterior fue un sistema desarrollado y documentado adecuadamente, que mejora y facilita la administración de una ruta de transporte y permite resolver los problemas relacionados con la manera actual en que esto se hace, entre los cuales se encuentra el problema de definir horarios de servicio que satisfagan adecuadamente la demanda existente. Además el sistema resultante es sencillo de mantener y mejorar con funcionalidades adicionales.

Conclusiones

Después de haber realizado las actividades descritas en este trabajo de tesis, fue posible construir un sistema adecuado que ayudara de manera significativa a solucionar los problemas descritos relacionados con una mala administración de una ruta de transporte, gracias a que se contó con los elementos teóricos

y técnicos adecuados de administración de proyectos, ingeniería de software y administración de rutas de transporte:

- Al haber realizado en el orden adecuado las actividades de administración de proyectos y de ingeniería de software y al tener la documentación relacionada actualizada (escenarios de uso, arquitectura y diseño, código, etc.), se logra un sistema de software fácil de mantener y modificar.
- El sistema fue creado de tal forma que la manipulación de la información y la generación de horarios para las rutas se realiza de manera mucho más rápida y sin los errores cometidos cuando se hace de forma manual.
- El haber utilizado escenarios de uso permitió describir al sistema requerido desde el punto de vista del usuario, lo cual permitió asegurar su aceptación más rápida por el mismo.
- La arquitectura y el diseño estructurado de la aplicación permitió hacer una transformación adecuada de los requerimientos en modelos más comprensibles desde el punto de vista de la implementación y por lo tanto más fáciles de codificar, así como más fáciles de probar e integrar.
- La arquitectura centrada en los datos permitió cumplir con el requisito de tener un almacenamiento y un manejo más fácil, rápido y seguro de los datos. La arquitectura de llamada/retorno permitió cumplir con los requisitos de comportamiento del sistema, desde el punto de vista de la interacción del usuario con el mismo, tal como se indicó en los escenarios de uso. Además de la interacción con el usuario, la arquitectura permitió comprender las interacciones entre las distintas partes del sistema, lo cual facilitó la definición y ejecución de las pruebas de integración.
- Utilizar una base de datos para el sistema hizo que el almacenamiento de la información fuera más sencillo y menos voluminoso que si fuera en papel, así como su análisis más fácil y rápido y menos propenso a errores que si fuera de manera manual. También aseguró un mejor manejo e integridad de los datos.
- El sistema construido permite la rápida captura de la información de los muestreos hechos.
- La interfaz de usuario desarrollada es fácil de aprender y utilizar, ya que se basó en los escenarios de uso, y el sistema se comporta de manera muy similar al método manual de administración de rutas, ya que utiliza los mismos elementos visibles. La interfaz de usuario desarrollada de esta manera también permitió facilitar las pruebas en sus distintos niveles: unitarias, de integración, de sistema y de aceptación.
- Con base en el uso del sistema es posible dividir el comportamiento de una ruta a lo largo del tiempo en temporadas, tipos de día y secciones.
- Con base en el uso del sistema se realizan consultas de la información existente y se pueden generar gráficas de demanda de transporte para cada ruta, a lo largo de las temporadas, tipos de día y secciones definidas.
- Con base en el uso del sistema se crean horarios que satisfacen la demanda existente de transporte, optimizando el uso de las unidades de transporte.
- La utilización de métodos adecuados para la planificación y seguimiento de proyectos, así como para la identificación y administración de riesgos, hizo posible el monitoreo del avance del desarrollo del sistema, e hizo más fácil la resolución de los problemas ocurridos durante el desarrollo.
- El haber contado con un método formal para el análisis de los datos de los muestreos hechos en una ruta de transporte y para el establecimiento de horarios óptimos a partir de ellos, basado en la literatura relacionada con los problemas de transporte existentes en nuestro país, hizo posible definir una manera adecuada, efectiva y (a través del sistema) rápida de solucionar los problemas específicos de nuestros servicios de transporte, relacionados con una mala administración de una ruta de transporte.
- Gracias a la manera en que fue desarrollado, utilizando las técnicas de ingeniería de software adecuadas y descritas anteriormente, al sistema puede hacérsele varias mejoras y agregársele funcionalidades adicionales como las siguientes:
    - Administración de redes de rutas, considerando más de una ruta.
    - Administración de usuarios (permisos y seguridad).
    - Cálculo de costos de administración de una ruta.

- Generación de reportes.
- Simulaciones con animaciones.
- Importación y exportación de datos a Word, Excel, etc.
- Mejoras a la interfaz gráfica de usuario.
- Funcionalidades con cartografía y dispositivos GPS.

Este trabajo puede ayudar a las organizaciones dedicadas a proveer servicios de transporte intraurbano a solucionar el problema de la mala administración del transporte público, para proveer un servicio de transporte adecuado, con el tipo de unidades de transporte adecuado, en la cantidad adecuada y óptima, con frecuencias de horario adecuadas, satisfaciendo la demanda y optimizando los costos de operación, y ayudando a la gente a decidir el uso del transporte público sobre los vehículos privados, con el fin de reducir los problemas de transporte existentes en nuestras ciudades, relacionados con las congestiones viales, los tiempos exagerados invertidos en transporte, y las consecuencias económicas, de salud y demás que estos problemas tienen.

Bibliografía

- Bass, L. Clements, P. Kazman, R. *Software Architecture in Practice.* Addison Wesley, 2ª Edición, Estados Unidos, 2003.

- Date, C. J. *Introducción a los Sistemas de Bases de Datos.* Prentice Hall, 7ª Edición, México, 2001.

- DeMarco, Tom. Lister, Timothy. *Waltzing with Bears: Managing Risk on Software Projects.* Dorset House Publishing, Estados Unidos, 2003.

- Hall, Elaine M. *Managing Risk: Methods for Software Systems Development.* Addison Wesley, SEI Series in Software Engineering, Estados Unidos,1998.

- Molinero, Ángel. Sánchez, Ignacio. *Transporte Público: Planeación, Diseño, Operación y Administración.* Quinta del Agua Ediciones, México, 1996.

- Pressman, Roger S. *Ingeniería del Software. Un enfoque práctico.* McGraw Hill, 5ª Edición, España, 2002.

- Sommerville, Ian. *Ingeniería de Software.* Addison Wesley, 6ª Edición, Reino Unido, 2002.

- Zaldivar Zamorategui, Orlando. *Apuntes de Ingeniería de Programación.* UNAM, Facultad de Ingeniería, México, 2001.