Universidad de las Ciencias Informáticas
Facultad 3

"Simulador de algoritmos de planificación de procesos para la asignatura Sistemas Operativos"

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Yamilka González Sánchez

Tutor: Ing. Yohandri Ril Gil

Cotutor: Dr.C. Pacual Verdecia Vicet

Ciudad de la Habana

Junio 2008

"El que no tenga el valor de sacrificarse, que al menos tenga el pudor de callarse ante los que se sacrifican...."

José J. Martí Pérez

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yamilka González Sánchez

_____
Firma del Autor

Yohandri Ril Gil

_____
Firma del Tutor

DATOS DE CONTACTOS

Nombre y Apellidos: Yohandri Ril Gil.
Edad: 27 años.
Ciudadanía: Cubano.
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: Ingeniero en Telecomunicaciones y Electrónica.
Categoría Docente: Profesor Asistente.
E-mail: rilltt@uci.cu

Ingeniero en Telecomunicaciones y Electrónica, graduado en el Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) en el 2004. Actualmente trabaja como Asesor Técnico docente del departamento de Sistemas Digitales y Aseguramiento básico de Programación. Posee la categoría docente de Profesor Asistente.

Nombre y Apellidos: Pascual Verdecia Vicet.
Edad: 45 años.
Ciudadanía: Cubano.
Institución: Universidad de las Ciencias Informáticas (UCI).
Títulos: Ing. Minas (1987), Ing. Civil (1994), Msc Voladura con Explosivos (1996), Dr.C (2002).
Categoría Docente: Profesor Asistente. 20 años de experiencia trabajando en la Física.
E-mail: pverdecia@uci.cu

AGRADECIMIENTOS

Son muchas las personas a las que tengo que agradecer por haberme ayudado en la realización de este trabajo, pero principalmente le doy las gracias a mis padres Daniel, Jorge y Aniuska, a mis hermanas Yaima y Zamira, a mi abuelita Deborah, a mi abuelo Luis, a mi tía Ana, a mis tíos, primos, en fin a todo el familión, por confiar a ciegas en mí, por inculcarme los hábitos de estudio, porque sin ellos yo no sería lo que soy hoy.

***A** mis amigos los que más quiero: María Antonia, Sucell y Norbeys porque siempre han estado ahí cuando los he necesitado, por el apoyo infinito que me han brindado, sin el cual todo hubiese sido más difícil, además porque me han aguantado mis malacrianzas, porque son y serán siempre mis amigos del alma, los que nunca olvidaré, Los quiero de todo corazón!!!!!!!*

***A** Pascual o mejor dicho a Bartolo por ser tan, pero tan especial para mí, por ser mi guía aquí en la universidad, por cuidarme, aconsejarme y ayudarme siempre, te quiero Bartolito como a un padre porque es lo que has sido para mí en estos años, y espero no te olvides nunca de mi porque por lo menos yo nunca me olvidare de ti….. mi viejito lindo….*

***A** Rainer por todo el amor que me ha brindado en estos últimos 3 años, por estar siempre dispuesto a ayudarme en lo más mínimo, por su preocupación, por todos los momentos tan lindos que su lado viví y que nunca olvidaré. Gracias.*

***A** mi tutor Yohandri Ril Gil, primero que nada por aceptar ser mi tutor cuando yo se lo pedí, por la gran ayuda que me ha brindado en el desarrollo de este trabajo, que no es sólo mío, es de él también, además porque más que mi tutor es mi amigo, a quien admiro, respeto y siempre le estaré agradecida…*

***A** Rene, que no puede faltar, graciasssss porque sin ti hubiese pasado muchísimo más trabajo del que pasé, por estar dispuesto a ayudarme y aclararme dudas en todo momento sin importar hora, sin importar que nos conocimos bastante tarde, por todo rene, por todo lo que has hecho por mí , mil gracias.*

***A** Yasma, Rey, Yane, Yami 2, Lidy, Samu, Juli, May, Aruca, Mili, Aliesky, Alain, Yuliet, Yonelbys, Carlos, y demás del 3112, gracias por los momentos agradables, llenos de ingenuidad, de inmadurez, e inolvidables que pasamos juntos en primer año, esos que siempre uno quiere que vuelvan a suceder, esos que siempre llevo conmigo porque en verdad fueron para mí los mejores en estos 5 años.*

***A**gradecer además a 7 personas que de una manera u otra me han ayudado sin límites, ellos son: mi primo Manolito que lo quiero mucho, mucho, Juan José, Yuniet, Noli, Albis, Albis, Leo, y César.*

***A** mis amigos de Gtmo, que aunque estemos separados eso no quiere decir que no los recuerde siempre y que no les agradezca porque la distancia no ha sido un obstáculo que entorpezca nuestra amistad, ellos son: Anita, Odalis, Alain, Claricel, Papin, Iyis, Obelkis, y Yamilka.*

***A** la Revolución y a todo aquel que ha contribuido a mi formación profesional.*

A todos, GRACIAS……

DEDICATORIA

***A** mi mamita como muestra de todo el amor que siento por ella, por hacer de mi lo que soy hoy, por la confianza que ha depositado en mí, y que nunca defraudaré. Porque es lo que más quiero en esta vida. Por su preocupación diaria, por su dedicación, por sus consejos, por los valores que ha sembrado en mi. Por tantas y tantas cosas que esta hoja y el tiempo no son suficientes para decirlas. Gracias, mil gracias y ten siempre presente que eres la razón de mí existir, eres la razón por la cual siempre me digo que tengo que esforzarme por salir adelante, para poder ayudarte, porque sin lugar a dudas mejor madre no existe, eres única y es por eso que te amo con todas las fuerzas de mi corazón……...*

***A** mi tío Jorge por su esfuerzo infinito por hacer y lograr que nuestra familia esté siempre unida, armoniosa, y llena de Felicidad, por ser como un padre para Yai y para mí, porque siempre ha estado dispuesto a ayudarnos en todo sin importar nada más que lo mucho que nosotras lo queremos a él y lo mucho que nos quiere él a nosotras….*

***A** mi papá por todo el amor que me brinda a diario, porque siempre ha estado pendiente de mi, de que todo me vaya bien, porque su apoyo ha sido mi sostén en estos 5 años, porque sus consejos no me han faltado nunca, y me han servido para enfrentar situaciones difíciles. Porque te quiero papi y espero eso nunca lo dudes….*

***A** María del Carmen, como muestra de mi agradecimiento en primer lugar por darme a la niñita que más quiero en este mundo, por cuidar de mi papá y por todo el cariño y comprensión que me brinda sin condición alguna.*

***A** mis hermanitas Yai y Zami, mis tesoros, mis niñitas, porque son las mejores hermanas del mundo, estoy orgullosa de ustedes, las amooooooooooooooooooooooooooooooooooooooooooo.*

***A** mis abuelos Deborah y Luis porque son como mis padres, tan atentos, tan preocupados, tan llenos de amor y dedicación. Mamita gracias porque siempre estas orando por mí, pidiendo a Dios que me bendiga y ayude, por el cariño y la ternura, con la que siempre me guías por el buen camino.*

***A** mi tía Ana, mi viejita linda, que tanto quiero porque sabe llegar al corazón de todos los que la conocen, porque siempre me ha ayudado a ver la parte positiva de las cosas que a diario suceden. Eres verdaderamente una persona muy especial para mí, alguien que llevo y llevaré por siempre en mi corazón.*

***A** toda mi familia quienes sin escatimar esfuerzo alguno, han sacrificado gran parte de su vida para formarme y educarme. A toda la gente que me vio crecer, a quienes la ilusión de su vida ha sido convertirme en una persona de provecho. Entre ellos: Mirtha, Juana, Daniela, Deborita, Nevis, Delvis, Kati, Dora, Dalvis, Oni, Daritza, Magalis, Marilu, mi abuela Carmela, mis tío Eglis, Luisito, Elcire, a Reyes, Yoye, Tico, a mis abuelos Nenita y Daniel, a mis primos Yarima, Lestico, Joe, Loren, Mislay, Solangel, Manolito, en fin a todo el familión que es bastante grande.*

Y por último a mis abuelitos Cecilio y Tina, que siempre llevo en mi corazón y mi pensamiento aunque no esté presentes físicamente

RESUMEN

Cualquier Sistema Operativo (SO) debe gestionar los recursos del computador de una manera eficiente. Para conseguir tal finalidad planifica el uso de los mismos antes de utilizarlos. La CPU[1] es uno de los recursos más importantes del computador, por lo tanto, su planificación es un elemento fundamental en el diseño de un SO. El proceso de planificación de la CPU se lleva a cabo siguiendo un algoritmo determinado, en concreto, el más indicado para el sistema en cuestión.

El principal objetivo de esta investigación es la creación de un simulador que represente el proceso de planificación de la CPU de acuerdo a algunos de los diferentes algoritmos de planificación existentes. Para el desarrollo del mismo se realizó un estudio de las principales metodologías, tecnologías y tendencias actuales en el desarrollo de este tipo de herramienta, a partir del cual se seleccionó la plataforma Java, con su IDE de desarrollo Netbeans, RUP como metodología de desarrollo, Visual Paradigm como herramienta CASE y UML como lenguaje de modelado.

Este simulador será utilizado con fines didácticos en la impartición de la asignatura SO correspondiente a la disciplina Sistemas Digitales de la carrera de Ingeniería en Ciencias Informáticas de la Universidad de las Ciencias Informáticas (UCI).

PALABRAS CLAVE

CPU, Planificador, Proceso, Sistema Operativo, Simulación.

[1] La unidad central de procesamiento, CPU (por sus siglas del inglés Central Processing Unit), o, simplemente, el procesador, es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de computadora.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Hace poco más de veinte años, un gran número de escritores, académicos, investigadores y otros, vienen refiriéndose a un fenómeno que ha irrumpido en nuestra sociedad con una fuerza increíble, la denominada sociedad de la información, con gran fuerza en la actualidad nombrada sociedad del conocimiento. Este fenómeno ha traído consigo una serie de transformaciones en todas las esferas de la sociedad, cambiando profundamente la forma en que hasta ese momento se pensaban y realizaban las más diversas funciones. Un papel determinante en todo este inmenso cambio, le ha correspondido al uso intensivo y extensivo de las Tecnologías de la Información y las Comunicaciones (TIC), soporte y fundamento de la sociedad del conocimiento. (1)

La informática, unida a las comunicaciones, posibilita prácticamente a todo el mundo el acceso inmediato a la información, el recurso considerado hoy más valioso. Si se quiere alcanzar un objetivo, es preciso acceder a la información pertinente para llegar a tomar las decisiones adecuadas. Puede decirse que Sociedad de la Información es, ante todo, Sociedad de Formación. Por ello hoy las TIC son consideradas esencialmente como el substrato para la formación de los individuos en esta sociedad. A su vez esta sociedad se va formando moldeada por las TIC.

En función de este enfoque, las posibilidades educativas de las TIC han de ser consideradas en dos aspectos: su conocimiento y su uso.

El primer aspecto es consecuencia directa de la cultura de la sociedad actual. No se puede entender el mundo de hoy sin un mínimo de cultura informática. Es preciso entender cómo se genera, cómo se almacena, cómo se transforma, cómo se transmite y cómo se accede a la información en sus múltiples manifestaciones (textos, imágenes, sonidos) si no se quiere estar al margen de las corrientes culturales, hay que intentar participar en la generación de esa cultura, es necesario integrarla en la Educación de los países, contemplándola en todos los niveles de Enseñanza. Es previsible que ese conocimiento se traduzca en un uso generalizado de las TIC para lograr, libre, espontánea y permanentemente, una formación a lo largo de toda la vida.

El segundo aspecto, aunque también muy estrechamente relacionado con el primero, es más técnico. Se deben usar las TIC para aprender y para enseñar. Es decir el aprendizaje de cualesquiera materias o habilidades se puede facilitar mediante las TIC. Este segundo aspecto tiene que ver muy estrechamente con la Informática Educativa.

INTRODUCCIÓN

La formación utilizando medios informáticos es una tendencia novedosa y con ostensibles resultado a nivel mundial. Esto no se refiere sólo a la formación permanente de libre iniciativa individual, sino a la enseñanza preparada con materiales informatizados para desarrollar sesiones interactivas de aprendizaje.

Hay que tener en cuenta los recursos informáticos para presentar información gráfica de todo tipo. Las computadoras producen imágenes fantásticas, estáticas y animadas. En la circunstancia apropiada "vale más una imagen que mil palabras".

Hay que buscar las oportunidades de ayuda o de mejora en la educación explorando las posibilidades educativas de las TIC sobre el terreno; es decir, en todos los entornos y circunstancias que la realidad presenta. (2)

El software educativo constituye una evidencia del impacto de la tecnología en la educación, pues es la más reciente herramienta didáctica útil para el estudiante y profesor, convirtiéndose en una alternativa válida para ofrecer al usuario un ambiente propicio para la construcción del conocimiento.

Los software educativos pueden tratar las diferentes materias (Matemática, Idiomas, Geografía, Dibujo, etc.), de formas muy diversas (a partir de cuestionarios, facilitando una información estructurada a los alumnos, mediante la simulación de fenómenos, etc.) y ofrecer un entorno de trabajo más o menos sensible a las circunstancias de los alumnos y más o menos rico en posibilidades de interacción; todos comparten las siguientes características:

- Permite la interactividad con los estudiantes, retroalimentándolos y evaluando lo aprendido.
- Facilita las representaciones animadas.
- Inciden en el desarrollo de las habilidades a través de la ejercitación.
- Permite simular procesos complejos.
- Reduce el tiempo de que se dispone para impartir gran cantidad de conocimientos facilitando un trabajo diferenciado, introduciendo al estudiante en el trabajo con los medios computarizados.
- Facilita el trabajo independiente y a la vez un tratamiento individual de las diferencias.
- Permite al usuario (estudiante) introducirse en las técnicas más avanzadas.

INTRODUCCIÓN

En Cuba, el software educativo como apoyo al proceso de enseñanza aprendizaje se ha implementado teniendo en cuenta los avances tecnológicos en este sentido, es decir, a medida que avanza la informatización de la sociedad.

Conceptualmente, la Informatización de la Sociedad se define en Cuba como el proceso de utilización ordenada y masiva de las TIC para satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad.

Cuba sostiene la idea de que a la sociedad le es necesario universalizar el conocimiento como una de las formas de alcanzar una mejor calidad de vida para todos los ciudadanos, sin distinción de edad ni condición social. La fórmula “educación para todos, durante toda la vida”, se presenta como el núcleo de un amplio movimiento educacional que abarca todo el país y a todos los ciudadanos.

El uso de las TIC como apoyo a los programas de clases en el 100% de los centros de la enseñanza primaria, secundaria, tecnológica y universitaria del país, es un ejemplo de los esfuerzos que está haciendo el estado cubano para incorporar estas tecnologías en la educación. Otro ejemplo es la creación de instituciones especializadas para la formación de profesionales soportadas en el uso de las nuevas tecnologías. Una de estas instituciones es la Universidad de las Ciencias Informáticas (UCI), la cual tiene un rol protagónico en el desarrollo de la informática y las comunicaciones para el país. En los momentos actuales se encuentra enfrascada en el proceso de perfeccionamiento de la enseñanza para la formación de profesionales. Para lograr esta meta se trabaja en la adecuación de cada una de las disciplinas del perfil de la carrera de ingeniería en ciencias informáticas de manera que la impartición de cada una de las asignaturas esté soportada en el uso de las TIC.

Dentro del plan de estudio de la universidad se incluye la asignatura Sistemas Operativos (SO), la misma forma parte de la disciplina Sistemas Digitales, se imparte en el quinto semestre de la carrera y tiene como objetivos generales los siguientes:

- Valorar las características de diferentes sistemas operativos.
- Establecer las diferencias entre sistemas operativos distribuidos y no distribuidos
- Emitir criterios acerca del sistema operativo más apropiado a ser utilizado en un ambiente de cómputo.
- Utilizar con eficiencia las facilidades brindadas por determinados sistemas operativos.
- Configurar sistemas operativos.

- Asimilar un nuevo sistema de operación a partir del estudio de sus manuales técnicos y de usuarios.
- Hacer uso de las técnicas utilizadas en la elaboración de los sistemas operativos para la programación de sistemas informáticos.
- Manejar literatura en idioma extranjero con respecto a los sistemas operativos y sus técnicas de diseño.

Para dar cumplimiento a todos estos objetivos, la asignatura se divide en temas, entre los que se encuentra “Administración de Procesos”. Uno de los contenidos que presenta este tema es “Planificación de Procesos”, realizada mediante algoritmos. Actualmente la asignatura no cuenta con una herramienta que simule el funcionamiento de dichos algoritmos de planificación, esto trae consigo un exceso del tiempo empleado en el proceso de enseñanza-aprendizaje del contenido. Todo lo cual precisa del desarrollo de una herramienta que sirva como Medio de Enseñanza basado en el uso de las TIC, que viabilice la impartición y asimilación de dicha asignatura y que contribuya al Proceso de Desarrollo de Software Educativo en sentido general.

Problema Científico de Investigación:

¿Cómo contribuir con el uso de las TIC a disminuir el tiempo empleado en el proceso de enseñanza-aprendizaje de la asignatura Sistemas Operativos?

Objeto de Estudio: Proceso de Desarrollo de Software Educativo.

Campo de Acción: Simulador de Algoritmos de Planificación de Procesos de Sistemas Operativos.

Objetivo:

Desarrollar un simulador de Algoritmos de Planificación de Procesos que contribuya a reducir el tiempo empleado en el proceso de enseñanza-aprendizaje de la asignatura Sistemas Operativos.

Hipótesis.

Si se desarrolla un simulador de Algoritmos de Planificación de Procesos para la asignatura de Sistemas Operativos se garantizará una disminución del tiempo empleado en el proceso de enseñanza-aprendizaje de la asignatura SO del 3er año de la carrera de Ingeniería en Ciencias Informáticas de la Universidad de las Ciencias Informáticas.

Tareas de la investigación:

- Realización de un estudio del Proceso de Desarrollo de Software Educativo.
- Identificación de los requerimientos que debe cumplir la herramienta de simulación.
- Diseño de la herramienta de simulación.
- Implementación de la herramienta de simulación.
- Realización de pruebas a la herramienta de simulación.

Métodos Científicos de la Investigación

Métodos Teóricos:

Analítico – sintético: con el objetivo de buscar la esencia de los fenómenos, los rasgos que lo caracterizan y los distinguen, analizar teorías y documentos, permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio.

Histórico-lógico: analiza la trayectoria completa de los fenómenos, su condicionamiento a los diferentes períodos de la historia, poniendo de manifiesto la lógica interna de su desarrollo, de su teoría y halla el conocimiento más profundo de su esencia. Este método expresa en forma teórica la esencia del objeto y explica la historia de su desarrollo.

Modelación: método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. El modelo como sustituto del objeto de investigación es semejante a él, existiendo una correspondencia objetiva entre el modelo y el objeto, siendo el investigador quien elabora dicho modelo. El modelo es el eslabón entre el sujeto y el objeto intermedio.

Métodos Empíricos:

Entrevista: como técnica fundamental para obtener información necesaria y valiosa en el desarrollo de la investigación mediante conversaciones planificadas con los clientes.

La entrevista puede ser estructurada o no estructurada.

Para la realización de esta investigación se utilizó la entrevista no estructurada que es más abierta que la estructurada, prevé el tema pero no lleva un cuestionario rígido y puede variar de una persona a otra, es más flexible. Se aplica a especialistas en el tema, es una forma de obtener criterios de expertos.

Posibles resultados:

Una herramienta de simulación de Algoritmos de Planificación de Procesos para la asignatura de Sistemas Operativos.

El documento está estructurado por: un resumen, introducción, cuatro capítulos que constituyen el cuerpo fundamental de la tesis, conclusiones generales y bibliografía. Los capítulos son:

Capítulo 1: Fundamentación Teórica. Se presenta una breve reseña de los resultados del estudio bibliográfico realizado sobre los conceptos: Software Educativo y, proceso de planificación de la CPU de acuerdo a algunos de los diferentes algoritmos de planificación existentes. Además se describen las tendencias, tecnologías y metodologías más utilizadas en la actualidad para el desarrollo de este tipo de herramienta.

Capítulo 2: Características del Sistema. En él se aborda lo referente al funcionamiento del negocio: las reglas y la descripción del mismo. En este caso se desarrolla un modelo de dominio donde se analizan cada uno de los conceptos y entidades presentes en el negocio. Además de describir la solución propuesta, se exponen elementos imprescindibles para una solución exitosa: como lo son los requisitos funcionales y no funcionales, así como la descripción de los Casos de Uso del Sistema.

Capítulo 3: Diseño del Sistema. Se describe el diseño del sistema propuesto a través de la organización de su arquitectura, y la realización de los diagramas de clases y de secuencia.

Capítulo 4: Implementación y Prueba. Comienza con el resultado del diseño, implementando el sistema en términos de componentes, así como una revisión final de las especificaciones del diseño y de la codificación mediante la realización de pruebas, garantizando la calidad del software.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

En el desarrollo del presente capítulo se realizará un análisis detallado de las principales funciones y clasificaciones de los software educativos, particular interés se tiene en los simuladores. También se analizarán bien a fondo todos los aspectos que posibilitan una mejor comprensión de cómo los sistemas operativos manejan la planificación de procesos. Además se efectuará un estudio del estado del arte de las tendencias, tecnologías y metodologías más utilizadas a nivel internacional en el desarrollo de aplicaciones de escritorio, así como las plataformas de desarrollo que las soportan.

1.1 Software Educativo.

El nuevo paradigma educativo contempla la utilización de las nuevas tecnologías. El papel del computador como medio dinámico permite hablar del proceso educativo apoyado por la herramienta computacional. Las sociedades que comprenden que la educación es el norte de su desarrollo, se orientan hacia el logro de un proceso educativo permeado por la tecnología.

En este nuevo paradigma educativo, el desarrollo de materiales computarizados (específicamente software educativos) es complejo, deben efectuarse decisiones en torno a los contenidos, a las estrategias de enseñanza de dichos contenidos y a la forma de presentación más adecuadas con el objetivo de facilitar el proceso de aprendizaje del usuario.

Utilizar la informática como apoyo a los procesos de enseñanza-aprendizaje ha sido una inquietud que durante mucho tiempo ha sido investigada y probada por muchas instituciones y docentes. Su asimilación dentro de instituciones educativas ha aumentado en los últimos años, con lo que la demanda por software educativo de calidad es cada vez mayor.

Sánchez J. (1999), en su Libro "Construyendo y Aprendiendo con el Computador", define el concepto genérico de Software Educativo como cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar. Un concepto más restringido de Software Educativo lo define como aquel material de aprendizaje especialmente diseñado para ser utilizado con una computadora en los procesos de enseñar y aprender.

Según Rodríguez Lamas (2000), es una aplicación informática, que soportada sobre una bien definida estrategia pedagógica, apoya directamente el proceso de enseñanza aprendizaje constituyendo un efectivo instrumento para el desarrollo educacional del hombre del próximo siglo.

Finalmente, los Software Educativos se pueden considerar como el conjunto de recursos informáticos diseñados con la intención de ser utilizados en el contexto del proceso de enseñanza-aprendizaje.

El uso del software por parte del docente proporciona numerosas ventajas, entre ellas:

- Enriquece el campo de la Pedagogía al incorporar la tecnología de punta que revoluciona los métodos de enseñanza - aprendizaje.
- Constituyen una nueva, atractiva, dinámica y rica fuente de conocimientos.
- Pueden adaptar el software a las características y necesidades de su grupo teniendo en cuenta el diagnóstico en el proceso de enseñanza - aprendizaje.
- Permiten elevar la calidad del proceso docente - educativo.
- Permiten controlar las tareas docentes de forma individual o colectiva.
- Muestran la interdisciplinariedad de las asignaturas.
- Marca las posibilidades para una nueva clase más desarrolladora. (34)

Los software educativos de forma general, están compuestos por elementos multimediales como son los textos, sonidos, gráficos, animaciones y videos, y poseen características que de acuerdo con su propósito se desarrollan en mayor o menor proporción. Algunas de estas características son:

- La interactividad, entendida como las acciones que el programa facilita realizar al usuario y aquellas que este realiza cuando está dentro del programa.
- La navegabilidad, es decir, la facilidad que tiene el usuario para desplazarse por las diferentes pantallas que componen el software a través de diversos caminos como botones o enlaces.
- La recursividad, entendida como las diversas posibilidades de regresar a temáticas de interés desde cualquier punto del software.
- La accesibilidad, es decir, la facilidad para entrar al programa y una vez en él a todos los módulos o partes que lo componen. (34)

Por otra parte, algunos software de tipo educativo pueden poseer uno o varios módulos que se desarrollan de acuerdo con propósitos pedagógicos. Entre estos módulos están:

- De evaluación de conocimientos o habilidades, ya sea de lo que se conoce como conducta de entrada o conocimientos previos, de tipo formativo a lo largo del desarrollo del programa o al final del mismo.

- De información, ya sea a través de bancos o de presentaciones que la contienen en grandes cantidades y a la cual se puede acceder en corto tiempo.
- Soportes simbólicos de manipulación de símbolos y lenguaje en los que el usuario por ejemplo puede consignar notas, apuntes, hacer textos, dibujos, operaciones numéricas, etc.
- De construcción, es decir, de colecciones de objetos o figuras para encajar que el usuario puede arrastrar y colocar en diversas posiciones.
- De actividades directoras, es decir un módulo que contiene la guía de cómo desenvolverse dentro del software. Ejemplo de ello son las ayudas o los mensajes que retroalimentan las acciones realizadas por los usuarios.
- De solución de problemas, en donde se presentan situaciones complejas que el usuario debe resolver con ayuda de elementos iniciales presentados en el problema.
- De juegos, en los que se pretende reforzar la presentación de una habilidad o una información obtenida a través del software. Dentro de este módulo se pueden utilizar elementos de otros módulos.

Dentro del diseño y el desarrollo de un software educativo se puede utilizar solo un módulo o varios de ellos de acuerdo con los objetivos del mismo. En la mayoría de los casos se utilizan tres o cuatro módulos de forma combinada o uno solo que contenga elementos característicos de varios de ellos. (34)

1.1.1 Funciones del Software Educativo.

El software educativo, programas educativos o programas didácticos, tienen una finalidad didáctica, deben ser interactivos, individualizan el trabajo y son fáciles de usar; además realizan diversas funciones:

INFORMATIVA: La mayoría de los programas a través de sus actividades presentan unos contenidos que proporcionan una información estructuradora de la realidad a los estudiantes.

Los programas tutoriales, los simuladores y, especialmente, las bases de datos, son los programas que realizan más marcadamente una función informativa.

INSTRUCTIVA: Todos los programas educativos orientan y regulan el aprendizaje de los estudiantes ya que, explícita o implícitamente, promueven determinadas actuaciones de los mismos encaminadas a facilitar el logro de unos objetivos educativos específicos. Además condicionan el tipo de aprendizaje

que se realiza pues, por ejemplo, pueden disponer un tratamiento global de la información (propio de los medios audiovisuales) o a un tratamiento secuencial (propio de los textos escritos).

Si bien el ordenador actúa en general como mediador en la construcción del conocimiento y el metaconocimiento de los estudiantes, son los programas tutoriales los que realizan de manera más explícita esta función instructiva, ya que dirigen las actividades de los estudiantes en función de sus respuestas y progresos.

MOTIVADORA: Generalmente los estudiantes se sienten atraídos e interesados por todo el software educativo, ya que los programas suelen incluir elementos para captar la atención de los alumnos, mantener su interés y, cuando sea necesario, focalizarlo hacia los aspectos más importantes de las actividades.

Por lo tanto la función motivadora es una de las más características de este tipo de materiales didácticos, y resulta extremadamente útil para los profesores.

EVALUADORA: La interactividad propia de estos materiales, que les permite responder inmediatamente a las respuestas y acciones de los estudiantes, les hace especialmente adecuados para evaluar el trabajo que se va realizando con ellos. Esta evaluación puede ser de dos tipos:

- Implícita, cuando el estudiante detecta sus errores, se evalúa, a partir de las respuestas que le da el ordenador.
- Explícita, cuando el programa presenta informes valorando la actuación del alumno. Este tipo de evaluación sólo la realizan los programas que disponen de módulos específicos de evaluación.

INVESTIGADORA: Los programas no directivos, especialmente las bases de datos, simuladores y programas constructores, ofrecen a los estudiantes interesantes entornos donde investigar: buscar determinadas informaciones, cambiar los valores de las variables de un sistema, etc.

Además, tanto estos programas como los programas herramienta, pueden proporcionar a los profesores y estudiantes instrumentos de gran utilidad para el desarrollo de trabajos de investigación que se realicen básicamente al margen de los ordenadores.

EXPRESIVA: Los ordenadores son unas máquinas capaces de procesar los símbolos mediante los cuales las personas representan su conocimiento y comunicación, por lo que sus posibilidades como instrumento expresivo son muy amplias.

Desde el ámbito de la informática, el software educativo, permite la comunicación de los estudiantes con el ordenador, y con otros compañeros a través de las actividades de los programas, estos no suelen admitir la ambigüedad en sus "diálogos", de manera que los alumnos se ven obligados a cuidar más la precisión de sus mensajes.

INNOVADORA: Aunque no siempre sus planteamientos pedagógicos resulten innovadores, los programas educativos se pueden considerar materiales didácticos con esta función ya que utilizan una tecnología recientemente incorporada a los centros educativos y, en general, suelen permitir muy diversas formas de uso. Esta versatilidad abre amplias posibilidades de experimentación didáctica e innovación educativa en el aula. (3)

1.1.2 Clasificaciones de los Software Educativos.

PROGRAMAS TUTORIALES

Son programas que en mayor o menor medida dirigen, tutorizan, el trabajo de los alumnos. Pretenden que, a partir de unas informaciones y mediante la realización de ciertas actividades previstas, los estudiantes pongan en juego determinadas capacidades y aprendan o refuercen sus conocimientos y/o habilidades. Cuando se limitan a proponer ejercicios de refuerzo sin proporcionar explicaciones conceptuales previas se denominan programas tutoriales de ejercitación, como es el caso de los programas de preguntas (drill&practice, test) y de los programas de adiestramiento psicomotor, que desarrollan la coordinación neuromotriz en actividades relacionadas con el dibujo, la escritura y otras habilidades psicomotrices.

En cualquier caso, son programas basados en los planteamientos conductistas de la enseñanza que comparan las respuestas de los alumnos con los patrones que tienen como correctos, guían los aprendizajes de los estudiantes y facilitan la realización de prácticas más o menos rutinarias y su evaluación; en algunos casos una evaluación negativa genera una nueva serie de ejercicios de repaso. (3)

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

BASES DE DATOS

Proporcionan unos datos organizados, en un entorno estático, según determinados criterios, y facilitan su exploración y consulta selectiva. Se pueden emplear en múltiples actividades como por ejemplo: seleccionar datos relevantes para resolver problemas, analizar y relacionar datos, extraer conclusiones, comprobar hipótesis, etc. Las preguntas que acostumbran a realizar los alumnos son del tipo: ¿Qué características tiene este dato? ¿Qué datos hay con la característica X? ¿Qué datos hay con las características X e Y?

Las bases de datos pueden tener una estructura jerárquica (si existen unos elementos subordinantes de los que dependen otros subordinados, como los organigramas), relacional (si están organizadas mediante unas fichas o registros con una misma estructura y rango) o documental (si utiliza descriptores y su finalidad es almacenar grandes volúmenes de información documental: revistas, periódicos, etc.). (3)

CONSTRUCTORES

Son programas que tienen un entorno programable. Facilitan a los usuarios unos elementos simples con los cuales pueden construir elementos más complejos o entornos. De esta manera potencian el aprendizaje heurístico y, de acuerdo con las teorías cognitivistas, facilitan a los alumnos la construcción de sus propios aprendizajes, que surgirán a través de la reflexión que realizarán al diseñar programas y comprobar inmediatamente, cuando los ejecuten, la relevancia de sus ideas. El proceso de creación que realiza el alumno genera preguntas del tipo: ¿Qué sucede si añado o elimino el elemento X?

PROGRAMAS HERRAMIENTAS

Son programas que proporcionan un entorno instrumental con el cual se facilita la realización de ciertos trabajos generales de tratamiento de la información: escribir, organizar, calcular, dibujar, transmitir, captar datos, etc. A parte de los lenguajes de autor (que también se podrían incluir en el grupo de los programas constructores), los más utilizados son programas de uso general que provienen del mundo laboral y, por tanto, quedan fuera de la definición que se ha dado de software educativo. No obstante, se han elaborado algunas versiones de estos programas "para niños" que limitan sus posibilidades a cambio de una, no siempre clara, mayor facilidad de uso. De hecho, muchas de estas versiones resultan innecesarias, ya que el uso de estos programas cada vez resulta más

sencillo y cuando los estudiantes necesitan utilizarlos o su uso les resulta funcional aprenden a manejarlos sin dificultad. (3)

SIMULADORES

Presentan un modelo o entorno dinámico (generalmente a través de gráficos o animaciones interactivas) y facilitan su exploración y modificación a los alumnos, que pueden realizar aprendizajes inductivos o deductivos mediante la observación y la manipulación de la estructura subyacente; de esta manera pueden descubrir los elementos del modelo, sus interrelaciones, y pueden tomar decisiones y adquirir experiencia directa delante de unas situaciones que frecuentemente resultarían difícilmente accesibles a la realidad (control de una central nuclear, contracción del tiempo, pilotaje de un avión...). También se pueden considerar simulaciones ciertos videojuegos que, al margen de otras consideraciones sobre los valores que incorporan (generalmente no muy positivos) facilitan el desarrollo de los reflejos, la percepción visual y la coordinación psicomotriz en general, además de estimular la capacidad de interpretación y de reacción ante un medio concreto.

En cualquier caso, posibilitan un aprendizaje significativo por descubrimiento y la investigación de los estudiantes/experimentadores puede realizarse en tiempo real o en tiempo acelerado, según el simulador, mediante preguntas del tipo: ¿Qué pasa al modelo si modifico el valor de la variable X? ¿Y si modifico el parámetro Y? Se pueden diferenciar dos tipos de simulador:

- Modelos físico-matemáticos: Presentan de manera numérica o gráfica una realidad que tiene unas leyes representadas por un sistema de ecuaciones deterministas. Se incluyen aquí los programas-laboratorio, algunos trazadores de funciones y los programas que mediante un convertidor analógico-digital captan datos analógicos de un fenómeno externo al ordenador y presentan en pantalla un modelo del fenómeno estudiado o informaciones y gráficos que van asociados. Estos programas a veces son utilizados por profesores delante de la clase a manera de pizarra electrónica, como demostración o para ilustrar un concepto, facilitando así la transmisión de información a los alumnos, que después podrán repasar el tema interactuando con el programa.

- Entornos sociales: Presentan una realidad regida por unas leyes no del todo deterministas. Se incluyen aquí los juegos de estrategia y de aventura, que exigen una estrategia cambiante a lo largo del tiempo. (3)

1.2 Sistema Operativo (SO), Proceso y Planificación de Procesos.

1.2.1 Sistema Operativo.

Una computadora (PC) sin software es básicamente un conjunto de cables y circuitos imposibles de controlar. Con su software, la PC es un objeto útil, ya que puede realizar diversas actividades como por ejemplo: realizar búsquedas en INTERNET, almacenar, procesar y recuperar información, entre otras. El software de una PC se puede dividir en 2 grandes tipos: programas de aplicación y programas de sistema, los primeros son los encargados de realizar las tareas de los usuarios y los segundos controlan todas las operaciones de la PC. El programa de sistema más importante es el SO, el mismo se encarga de controlar todos los recursos con que cuenta la PC y establece la base sobre la que pueden escribirse los programas de aplicación.

1.2.1.1 Concepto de Sistema Operativo.

Cuando se habla de SO se piensa casi siempre en la ventanilla del logotipo de Windows, o algo relacionado con Microsoft; al menos en el medio actual. Cuando en realidad el concepto de SO, engloba muchas características que poco o nada tienen que ver con la idea original que se tiene. En los siguientes párrafos, se tratará de analizar algunas de las más populares definiciones de SO:

"Se puede definir un SO como un conjunto de programas que controlan directamente los recursos hardware o físicos de un ordenador (CPU, memoria principal y periféricos) proporcionando una máquina virtual más fácil de utilizar que el hardware subyacente". (4)

Según el libro Sistemas Operativos. Diseño e Implementación. Segunda Edición de Tanenbaum & Woodhull, no es fácil precisar con exactitud qué es un SO. Parte del problema consiste en que el SO realiza 2 funciones que básicamente no están relacionadas entre sí y, dependiendo de la persona, se podría escuchar más información acerca de una función u otra. Estas funciones son:

- *El SO como máquina extendida:*

El programa que oculta la verdad acerca del hardware y presenta al programador una vista sencilla y bonita de archivos con nombre que pueden leerse y escribirse es, por supuesto, el SO. Así como el SO aísla al programador del hardware del disco y presenta una interfaz sencilla orientada a archivos, también oculta muchos asuntos desagradables referentes a interrupciones, temporizadores, administración de memoria y otras funciones de bajo nivel. En cada caso la abstracción que el SO ofrece es más sencilla y fácil de usar que el hardware subyacente.

En esta vista, la función del SO es presentar al usuario el equivalente de una máquina extendida o máquina virtual que es más fácil de programar que el hardware subyacente.

- *El SO como administrador de recursos:*

El concepto del SO como algo cuya función primordial es ofrecer a los usuarios una Interfaz cómoda es una visión descendente. Una visión ascendente alternativa postula que el SO está ahí para administrar todos los componentes de un sistema complejo. Las computadoras modernas constan de procesadores, memoria, temporizadores, discos, ratones, interfaces con redes, impresoras láser una gran variedad de otros dispositivos. En la visión alternativa, la misión del SO es asegurar un reparto ordenado y controlado de los procesadores, memorias y dispositivos de E/S entre diferentes programas que compiten por ellos. (5)

"Un Sistema Operativo es el software encargado de ejercer el control y coordinar el uso del hardware entre diferentes programas de aplicación y los diferentes usuarios. Es un administrador de los recursos de hardware del sistema."

"Un Sistema Operativo es un conjunto de programas destinados a permitir la comunicación del usuario con un computador y gestionar sus recursos de manera eficiente. "

"Un Sistema Operativo es como una capa compleja entre el hardware y el usuario, concebible también como una máquina virtual, que facilita al usuario o al programador las herramientas e interfaces adecuadas para realizar sus tareas diversas, abstrayéndole de los complicados procesos necesarios para llevarlas a cabo."

Según las definiciones anteriores, un SO, es finalmente software. La idea original que controla, y además ejerce control, es enfocado en el sentido, de que un SO como tal, es una administrador de recursos. Esto nos amplía la idea que un SO no se limita únicamente a un computador; es decir visto desde este concepto, un SO, para fines prácticos, se encontrará en una gran diversidad de equipos electrónicos, como puede ser una cámara digital, un DVD, un impresor, un teléfono móvil, un cajero automático, un Ipod, etc., ejerciendo su función principal, de administrador de recursos, dichos equipos electrónicos, lógicamente deberán de contener al menos, un microprocesador para lograr entender órdenes pregrabadas y ejecutar acciones, con un nivel de abstracción para el usuario final; es decir, materializadas en el simple hecho de presionar un botón, o el movimiento de un dedo, sobre un dispositivo táctil. De otra forma, el SO es el encargado de brindar al usuario de manera más fácil,

sencilla y amigable la de operar, codificar, interpretar y emitir órdenes al procesador principal, para que éste realice las tareas necesarias para completar la orden.

El objetivo principal de un SO es lograr que una computadora o dispositivo electrónico se use de manera cómoda y el objetivo secundario es que el hardware del computador o dispositivo se emplee de manera eficiente. (6)

En un sistema en el que intervienen una gran cantidad de componentes (procesador, procesos, periféricos,...) debe existir algún elemento que se encargue de controlar que todo funcione correctamente. El encargado de realizar esta tarea es el SO, que toma el rol de administrador del sistema. El concepto de administrador surge porque:

- Administra la memoria: se encarga de asignar a los procesos la memoria necesaria para su ejecución.
- Administra los periféricos: posee todos los módulos necesarios para la utilización de los periféricos.
- Administra la información de archivos (File System): rutinas que permiten manipular y manejar el sistema de archivos.
- Administra las comunicaciones (Communication Manager): responsable de compartir los recursos distribuidos mediante una red de computadoras.
- Administra el procesador: consta de dos módulos: Dispatcher (decide a qué procesador asignar el proceso que tiene que ser ejecutado) y Controlador de Tráfico (se encarga de crear, modificar y actualizar el contexto asociado a un proceso).
- Administra los procesos: Surge a raíz de la aparición de la multiprogramación. (6)

1.2.2 Proceso.

Uno de los puntos fundamentales en un SO es el concepto de procesos, estos representa un programa en ejecución, y se dividen en dos grandes grupos: los procesos de sistema y los procesos de usuarios. El SO será el responsable de crear, eliminar o permitir que se ejecuten los procesos desarrollando una planificación eficiente y permitiendo una correcta sincronización.

Mientas que un programa puede ser definido como una secuencia de código ejecutable almacenada, se le va a llamar proceso, al estado en que el programa se está ejecutando. Siendo necesario para esto el uso de ciertos recursos como: tiempo de CPU, memoria, acceso a ficheros, interacción con los dispositivos de entrada/salida, entre otros. (7)

1.2.2.1 El bloque de control de proceso.

El Bloque de Control de Proceso (PCB) es la estructura de datos central y más importante de un SO. Cada PCB contiene toda la información que necesita el SO para el control de un proceso:

- Estado del proceso: Nuevo, Listo, en Ejecución, Bloqueado.
- Contador del programa: Dirección siguiente instrucción a ejecutar.
- Registros de la CPU: Contenidos al final de la última ejecución (contador de programa, puntero a pila, registros de datos, entre otros).
- Información de planificación de la CPU: prioridad, apuntadores a las colas, algoritmo usado.
- Información contable y de identificación: Número de proceso, tiempo real y de CPU utilizado.
- Información de estado Entrada/Salida (E/S): Solicitudes E/S pendientes, lista archivos abiertos, etc.

Estos bloques son leídos y/o modificados por casi todos los módulos de un SO, incluyendo aquellos que tienen que ver con la planificación, la asignación de recursos, el tratamiento de interrupciones y el análisis y supervisión del rendimiento. Puede decirse que el conjunto de los bloques de control de procesos definen el estado del SO.

El conjunto de todos los PCB's se guarda en una estructura del SO llamada tabla de procesos, que reside en memoria principal, debido a su alta frecuencia de consulta. (4)

En un sistema de multiprogramación, se requiere una gran cantidad de información de cada proceso para su administración. Sistemas distintos organizarán esta información de modo diferente. En general, se puede agrupar la información de los PCB's en tres categorías:

- Identificación del proceso.

- Información del estado del procesador.
- Información de control del proceso.

Para identificar un proceso, en casi todos los SO se le asigna un identificador numérico único (ID). Este identificador servirá para localizarlo dentro de la tabla de procesos. Además de estos, un proceso también puede tener asignado un identificador de usuario que indica a quién pertenece el proceso (UID).

Básicamente, la información de estado del procesador está formada por el contenido de los registros del procesador. Por supuesto, mientras el proceso está ejecutándose, la información está en los registros. Cuando se interrumpe el proceso, toda la información de los registros debe salvarse de forma que pueda restaurarse cuando el proceso reanude su ejecución. La naturaleza y número de registros involucrados depende del diseño del procesador. Normalmente, en el conjunto de registros se incluyen los registros visibles para el usuario, los registros de control y de estado (contador de programa y palabra de estado) y los punteros a pila.

A la tercera categoría general de información del PCB se le podría llamar información de control del proceso. Esta es la información adicional necesaria para que el SO controle y coordine los diferentes procesos activos. Como, por ejemplo, información de planificación y estado (estado del proceso, su prioridad, información de planificación, suceso), apuntadores (punteros) a estructuras de datos (los procesos que esperan en un semáforo), punteros a zonas de memoria del proceso, recursos controlados por el proceso (ficheros abiertos), entre otros.

Así pues, el PCB es la entidad que define un proceso en el SO. Dado que los PCBs necesitan ser manejados con eficiencia por el SO, muchos ordenadores tienen un registro hardware que siempre apunta hacia el PCB del proceso que se está ejecutando. A menudo existen instrucciones hardware que cargan en el PCB información sobre su entorno, y la recuperan con rapidez. (4)

1.2.2.2 Estados de un proceso.

Durante su vida, un proceso puede pasar por una serie de estados discretos, los más importantes son:

- **En ejecución:** El proceso ocupa la CPU en ese momento, es decir, se está ejecutando.
- **Listo o preparado:** El proceso dispone de todos los recursos para su ejecución, sólo le falta la CPU.

- **Bloqueado:** Al proceso le falta algún recurso para poder seguir ejecutándose, además de la CPU. Por recurso se pueden entender un dispositivo, un dato, etc. El proceso necesita que ocurra algún evento que le permita poder proseguir su ejecución.

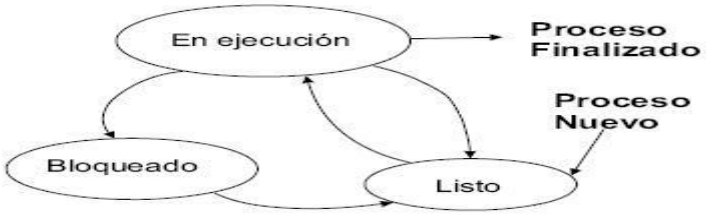


Figura 1: Estados de Proceso.

En los sistemas monoprocesadores, es decir, con una sola CPU, solamente puede haber un proceso en ejecución a la vez, pero pueden existir varios listos y varios pueden estar bloqueados. Así pues, se forman una lista de procesos listos y otra de procesos bloqueados. La lista de procesos listos se ordena por prioridad, de manera que el siguiente proceso que reciba la CPU será el primero de la lista.

La lista de procesos bloqueados normalmente no está ordenada; los procesos no se desbloquean (es decir, no pasan a ser procesos listos) en orden de prioridad, sino que lo hacen en el orden de ocurrencia de los eventos que están esperando. Hay situaciones en las cuales varios procesos pueden bloquearse esperando la ocurrencia del mismo evento; en tales casos es común asignar prioridades a los procesos que esperan. (4)

1.2.2.3 Transiciones de estado de los procesos.

A continuación se dan ejemplos de eventos que pueden provocar transiciones de estado de un proceso en este modelo de tres estados:

- **De Ejecución á Bloqueado**: al iniciar una operación de E/S.
- **De Ejecución á Listo**: por ejemplo, en un sistema de tiempo compartido, cuando el proceso que ocupa la CPU lleva demasiado tiempo ejecutándose continuamente (agota su quantum), y el SO decide que otro proceso ocupe la CPU, pasando el proceso que ocupaba la CPU a estado listo.
- **De Listo á en Ejecución**: cuando lo requiere el planificador de la CPU.

- **De Bloqueado á Listo**: se dispone del recurso por el que se había bloqueado el proceso. Por ejemplo, termina la operación de E/S.

Obsérvese que de las cuatro transiciones de estado posibles, la única iniciada por el proceso de usuario es el bloqueo, las otras tres son iniciadas por entidades externas al proceso. (4)

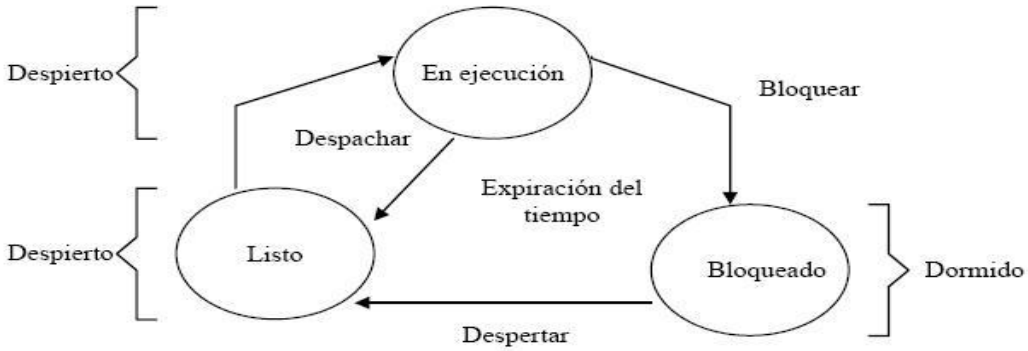


Figura 2 Transiciones de estado de los procesos.

1.2.2.4 Cambio de Proceso.

A primera vista, la función de cambio de proceso parece sencilla. En cierto momento, un proceso que se está ejecutando se interrumpe, el SO pone a otro proceso en el estado de ejecución y pasa el control a dicho proceso. Sin embargo, surgen diversas cuestiones de diseño. En primer lugar, ¿qué sucesos provocan un cambio de proceso? Otra cuestión es que se debe hacer una distinción entre cambio de contexto y cambio de proceso. Por último, ¿Qué debe hacer el SO con las diferentes estructuras de datos bajo su control para llevar a cabo un cambio de proceso?

¿Qué eventos provocan el cambio de proceso?

Un cambio de proceso puede suceder en cualquier instante en el que el SO gana el control de la CPU. En primer lugar, se van a tener en cuenta las interrupciones del sistema. Se pueden distinguir dos clases de interrupciones del sistema. La primera es originada por algún tipo de suceso que es externo e independiente del proceso que se está ejecutando, como la culminación de una E/S. La segunda tiene que ver con una condición de error o excepción generada dentro del proceso que se está ejecutando, como un intento ilegal de acceso a un fichero, una división entre cero, una instrucción máquina con código de operación no contemplado. En una interrupción ordinaria, el control se transfiere primero al gestor de interrupciones, quien lleva a cabo algunas tareas básicas y, después, se

salta a la rutina del SO que se ocupa del tipo de interrupción que se ha producido. Algunos ejemplos de estas interrupciones son:

Interrupción de reloj: Un reloj es un dispositivo que genera interrupciones periódicamente. Ante una interrupción de este tipo, un SO de tiempo compartido, entre otras cosas, determina si el proceso en ejecución ha alcanzado el máximo tiempo de ejecución que se le concedió. Si es así, el proceso pasará a estado listo, y se asignará la CPU a otro proceso.

Interrupción de E/S: El SO determina exactamente qué acción de E/S ha ocurrido. Si se trata de un evento por el que esperaban uno o más procesos, entonces el SO traslada todos los procesos bloqueados en dicho evento al estado listo, y determina (dependiendo de la política de planificación) si reanuda la ejecución del proceso interrumpido o pasa a otro de mayor prioridad.

Fallo de memoria: Un proceso hace una referencia a una dirección que no se encuentra en memoria y que debe traerse de memoria secundaria. Después de hacer la solicitud de E/S para traer esa o esas direcciones de memoria, el SO lleva a cabo un cambio de contexto para reanudar la ejecución de otro proceso; el proceso que cometió el fallo de memoria se pasa al estado bloqueado. Después de que las direcciones aludidas se carguen en memoria, dicho proceso se pondrá en estado listo.

En una interrupción del segundo tipo, el SO determina si el error es fatal. Si lo es, el proceso que se estaba ejecutando es eliminado, y se produce un cambio de proceso. Si no es fatal, la acción del SO dependerá de la naturaleza del error y del diseño del SO. Se puede hacer un cambio de proceso o, simplemente, reanudar el mismo proceso que se estaba ejecutando.

Finalmente, el SO puede activarse mediante una llamada al sistema desde el programa que se está ejecutando. Por ejemplo, está ejecutándose un proceso de usuario y se llega a una instrucción que solicita una operación de E/S, tal como abrir un fichero. Esta llamada provoca la transferencia a una rutina que forma parte del código del SO. Por lo general (aunque no siempre) el uso de una llamada al sistema hace que el proceso de usuario pase al estado bloqueado. (4)

1.2.2.5 Cambio de Contexto.

Cuando un proceso no está ejecutándose puede estar por lo menos en 2 posibles estados: esperando ser ejecutado (listo o preparado), o realizando alguna operación de E/S (bloqueado).

Los procesos ya estén listos o bloqueados se mantienen en sendas listas o estructuras llamadas cola de procesos listos y cola de procesos bloqueados respectivamente. Como se recordará cada una de las entradas en estas colas o listas es un PCB de un proceso.

En este se guarda la información que define el estado de ese proceso (registros, recursos que posee, entre otros) en un momento determinado y esta información se conoce también como información de contexto.

Así es que se produce un cambio de contexto cuando un proceso pasa a, o deja de ejecutarse. Básicamente lo que ocurre es que se salva la información de contexto del proceso que se le quita la CPU, se invoca el planificador del procesador, el cual selecciona el próximo proceso a ejecutar de la cola de listos, se carga la información de contexto del nuevo proceso y se comienza su ejecución.

Durante un cambio de contexto se producen las siguientes operaciones:

1. Cambio de modo: El procesador pasa de modo usuario a modo supervisor (posibilidad de ejecutar instrucciones privilegiadas).
2. Conservación del estado hardware: Se guarda el contador de programa, el puntero de la pila, registros, en fin se salva el estado hardware completo.
3. Conservación del estado general del proceso: Almacena los restantes campos del PCB modificados, como por ejemplo recursos adquiridos, tiempo de ejecución, ficheros abiertos o la razón de suspensión.
4. Cambio de cola. El PCB del proceso, pasa de la cola de procesos en ejecución a la cola de procesos bloqueados o suspendidos. (Puede decirse que existe una cola de procesos en ejecución, en un sistema monoprocesador solo hay un elemento en la cola).
5. Invocación del planificador del procesador: Escoge a uno de los procesos de la cola de preparados (De acuerdo a un algoritmo o política de planificación determinado).
6. Restauración: Se leen los registros hardware del PCB del próximo proceso a ejecutar y se restauran en el procesador, así como punteros que indican los recursos pedidos por el proceso como pueden ser segmentos de memoria.

7. Cambio de modo: Vuelve a cambiar el modo de trabajo del procesador a modo usuario y se le cede el control al nuevo proceso. (7)

Un sistema multiprogramado no puede ser eficiente si no hay un eficiente cambio de contexto. Esto implica emplear el menor tiempo posible en el cambio de contexto, para entregar rápidamente el control al próximo proceso a ejecutar. Con este fin se aplican técnicas de hardware y el concepto de hilo o hebra.

1.2.3 Planificación de Procesos.

1.2.3.1 Necesidad de la planificación.

Para lograr que se mantengan el mayor tiempo posible varios procesos en ejecución, es necesaria una efectiva planificación del procesador, pues es este el que se encarga de la ejecución de las instrucciones.

Por esto: A la parte del SO que se encarga de la planificación del uso del procesador se conoce como planificador de procesos (Process Scheduler- también planificador de trabajos), y a los posibles procedimientos o formas que este emplea para llevar a cabo dicha planificación se les conoce como algoritmos de planificación. (7)

1.2.3.2 Tipos de Planificadores.

En realidad, en un sistema de operación por lo general existen varios planificadores. El nombre de estos planificadores se puede asociar de acuerdo a su frecuencia de ejecución, y se mencionan a continuación.

- Planificador a Largo Plazo (Long Term-Scheduler).

Es el que decide los procesos que se añaden al conjunto de procesos a ejecutar. Se encarga de seleccionar qué trabajos de todos los que están preparados en un dispositivo (típicamente almacenados en el disco duro, mientras no están en memoria), son admitidos en la memoria para su posterior procesamiento. Por tanto este planificador es el responsable del grado de multiprogramación (número de procesos ejecutándose en memoria en un momento determinado). Su frecuencia de ejecución es larga, generalmente de minutos, por ello su código puede ser más largo para facilitar seleccionar el proceso más conveniente a pasar a la cola de listos en memoria. En los sistemas de tiempo compartido por su propia naturaleza, no existe este planificador o si existe su acción es mínima.

- Planificador a Corto Plazo (Short Term-Scheduler).

También se conoce como planificador del CPU. Selecciona a cuál de los procesos que están cargados en memoria y listos para ejecutar (cola de listos), se le entregará el control del CPU. Este planificador se ejecuta muy frecuentemente (cada vez que un proceso solicita E/S, al terminar un proceso, al ocurrir una interrupción que puede ser de reloj o una llamada al sistema por ejemplo) cada unos pocos o decenas de milisegundos. Por tanto debe ser muy rápido, y no emplear mucho tiempo en escoger el próximo proceso a ejecutar.

- Planificador a Medio Plazo (Médium Term-Scheduler).

Su función es ejecutar el swapping o el intercambio de trabajos entre memoria y disco. La idea es que en ocasiones es conveniente eliminar procesos de la disputa activa del CPU, es decir, quitarlos de la memoria para reducir el grado de multiprogramación. Consiste en remover temporalmente de la memoria procesos que están listos para ejecutarse y trasladarlos de nuevo hacia el disco duro (o un medio de almacenamiento), para posteriormente volverlos a introducir en la memoria y darle la posibilidad que se sigan ejecutando. Esta acción se realiza si hace falta más memoria disponible para un proceso (reasignación de memoria) o para mejorar el balance en el sistema de los procesos que hacen un uso intensivo del CPU con los que hacen poco uso de CPU, logrando que la cola de listos y de bloqueados nunca se vacíen. (7)

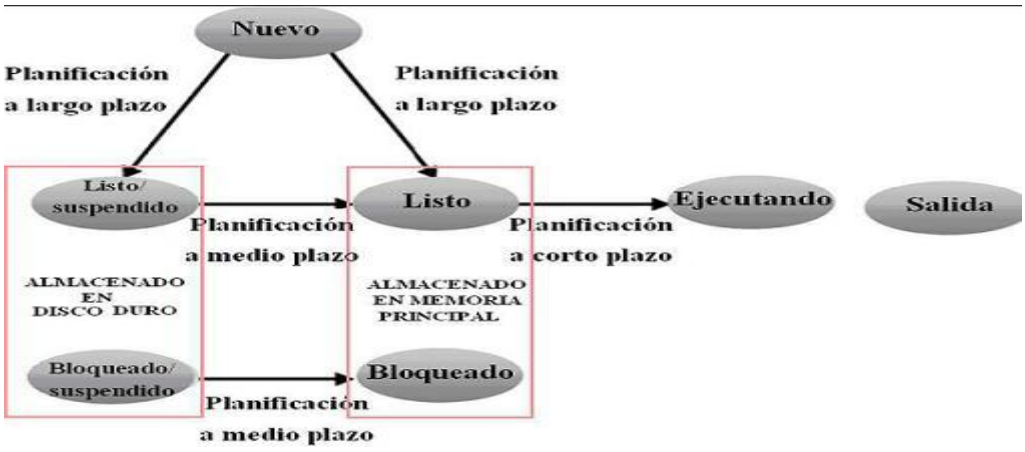


Figura 3 Acción de planificadores y transiciones de estado de los procesos.

Cada planificador representa un nivel determinado, aplicando determinado algoritmo que garantice el mayor grado de multiprogramación posible.

1.2.3.3 Criterios de planificación.

Existen numerosos algoritmos o políticas de planificación del procesador. Cada uno de ellos tiene sus propias características y pueden favorecer a un tipo de procesos respecto a otros. El privilegiar a unos siempre implica ir en detrimento de los otros. Por eso la selección de una política de planificación o algoritmos depende de las características que va a tener el sistema.

Existen varios criterios para comparar los diferentes algoritmos. Los criterios que básicamente se utilizan son:

- Eficiencia o utilización del CPU: Esto significa mantener el CPU ocupado tanto como sea posible, con un objetivo teórico del 100% (en realidad puede ir de un 40 a un 90 %).
- Productividad (Throughput): Será el número de trabajos que se completan por unidad de tiempo. La meta es maximizar este indicador y para ello se puede dar mayor prioridad a los trabajos más cortos.
- Tiempo de retorno (Turnaround): Es el tiempo que transcurre desde que un trabajo se somete al sistema (se crea) hasta que termina (finaliza); es la suma de los tiempos esperando para entrar en memoria, esperando en la cola de listo, ejecutando en el CPU y haciendo entradas/salidas. La meta es minimizar este indicador. Normalmente se usa el turnaround promedio.
- Tiempo de espera: Es la cantidad de tiempo que un trabajo gasta esperando en la cola de listos. Este es una medida más ilustrativa del efecto del algoritmo de planificación. Se indica que debe haber equidad o imparcialidad, es decir, que cada proceso debe tener su parte razonable en el uso del CPU.
- Tiempo de respuesta: En los sistemas interactivos el turnaround puede no ser un buen criterio para medir la respuesta a los usuarios. En este caso se acostumbra a medir el tiempo que media entre la solicitud de una acción y el comienzo de la respuesta (no incluye el tiempo que demora la respuesta en sí). Lógicamente, el objetivo es minimizar este indicador. (7)

En gran parte de los casos lo que se trata es de optimizar la media de todos los parámetros anteriores, en otros maximizar algunos de los criterios porque mejora la aplicación a la que va destinado el sistema.

No obstante el arte consiste en dar prioridad a aquellos más convenientes y necesarios en un medio ambiente determinado, sin provocar que los otros se deterioren en exceso.

1.2.3.4 Algoritmos de planificación de procesos.

Las técnicas o algoritmos de planificación se dividen en dos grupos:

Las técnicas con derecho de apropiación: son una fuente de condiciones de concurso o competencia que los procesos deberán tener en cuenta.

Por otro lado, las sin derecho de apropiación: pueden implicar que un proceso se adueñe del procesador por un tiempo indefinido en perjuicio de los restantes.

- **Planificación con derecho de apropiación** o expulsiva (preemptive scheduling).
- **Planificación sin derecho de apropiación** o no expulsiva (non preemptive scheduling).

El primer grupo está referido a técnicas que le quitan el CPU al proceso que se está ejecutando; es decir, se le retira el control del procesador, aún cuando todavía estaría en condiciones de continuar ejecutándose, para dársele a un nuevo proceso.

El segundo grupo tiene una base cooperativa, es decir, el proceso entrega el procesador solo cuando termina o se bloquea. (7)

Luego de estos conocimientos previos, se explicarán varios de los diferentes algoritmos de planificación, basándose en el ejemplo de la Tabla 1, que indica: el tiempo la creación de 5 procesos, el tiempo de llegada a la cola de listos de cada uno, así como el tiempo de CPU que le ocupan al procesador una vez que tienen el control de este, después de ese tiempo de servicio, finaliza el proceso.

Tabla 1 Datos de Procesos

Proceso	Instante de llegada	Tiempo de servicio
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

Algoritmo FCFS (First Come/First Served o Primero en llegar, Primero en Servirse).

Es el algoritmo de planificación más simple, se ejecutan los procesos según el orden de llegada a la cola de listos. Esta técnica es por naturaleza sin derecho de apropiación.

La instrumentación del FCFS es muy simple y solo se requiere que la cola de listos se opere realmente como una cola. Cada proceso que llega a ella se coloca (enlaza) al final y siempre se le asigna el CPU al proceso que está en la cabeza, eliminándose de la lista.

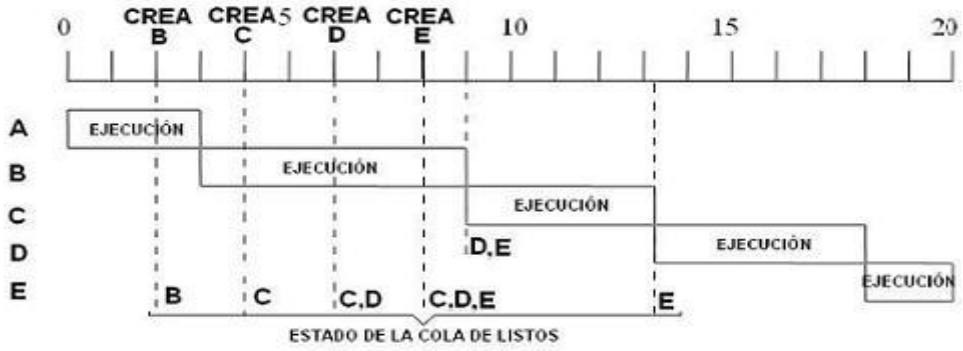


Figura 4 Algoritmo FCFS

Desventajas: Los resultados que se obtienen con este algoritmo son muy pobres debido a que no tiene en cuenta las características y el tipo de procesos. (7)

SJF (Shortest Job First- El más corto Primero).

Este algoritmo puede ser implementado con o sin derecho de apropiación aunque con el nombre SJF, nos referiremos a la variante no apropiativa.

En este algoritmo se le asocia a cada trabajo la duración de su siguiente ráfaga de CPU, de tal manera que el que tenga la próxima ráfaga más pequeña es el primero que se atiende. En caso de igualdad de utiliza el FCFS.

Ventajas: Es óptimo desde el punto de vista del tiempo de retorno promedio, la productividad y el tiempo de espera.

Desventajas:

- No resulta fácil hacer uso de esta técnica en la planificación del procesador (short term scheduling) y para utilizarla se requiere hacer continuamente un pronóstico del tiempo de CPU que requerirá cada proceso en cada intervalo de ejecución.
- Puede provocar inanición (starvation) en los trabajos largos. (7)

Figura 5 Algoritmo SJF

SRTF (Shortest Remaining Time First- Primero el de Menor Tiempo Restante).

Es la implementación apropiativa del SJF. Sus características son las mismas.

Figura 6 Algoritmo SRTF

En este caso también se le entrega el CPU por parte del planificador al proceso más corto con la única diferencia de que esto no se apropia del CPU sino que se está ejecutando hasta que haya un proceso con menor tiempo de ejecución. (7)

Round Robin.

La técnica utilizada en los sistemas de tiempo compartido es el algoritmo RR (Round Robin). En la traducción del segundo libro de Tanenbaum se le llama torneo.

En este caso, la cola de listos es tratada como una cola circular (lo que no implica que tenga que ser instrumentada en esta forma) y el planificador gira alrededor de la cola, asignando el CPU a cada proceso.

El proceso que recibe el CPU lo puede utilizar durante un quantum o ranura de tiempo (10 a 100ms), transcurrido éste lo pierde hasta que le toca de nuevo. Por supuesto, si por alguna razón se tiene que bloquear o termina, también pierde el procesador.

Es evidente que en este algoritmo existe el derecho de apropiación, pues aunque un trabajo esté en condiciones de seguir su ejecución, si se venció su quantum de tiempo se le quitará el CPU.

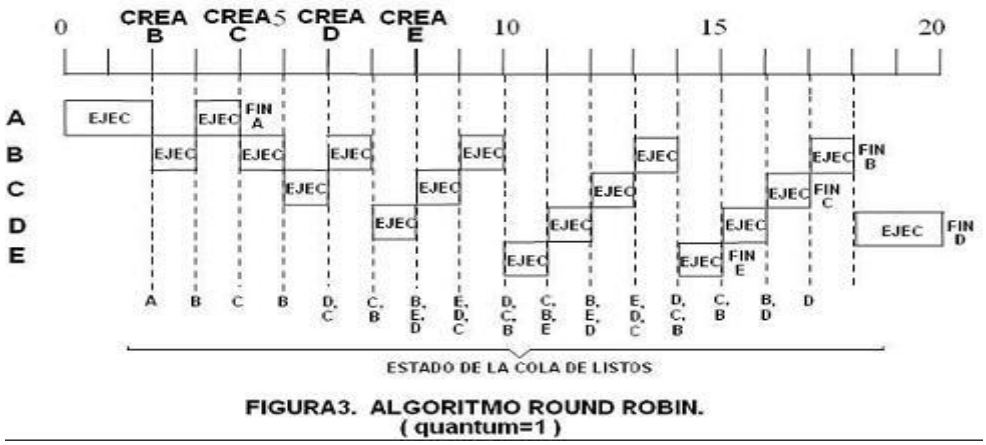


Figura 7 Algoritmo RR

Un aspecto clave en la aplicación de esta técnica consiste en la determinación de la longitud del quantum o ranura de tiempo. Si la longitud del quantum es muy pequeña, entonces existirá ineficiencia en el uso del CPU ya que una parte considerable del tiempo se gastará en el cambio de contexto.

El uso de un quantum muy grande hace que el algoritmo se convierta en un FCFS, derivando en una respuesta pobre a los procesos interactivos. La regla empírica es escoger el intervalo de forma tal que el 80% de las ráfagas de CPU sean menores que el quantum. (7)

1.3 Tendencias, tecnologías y metodologías más utilizadas a nivel internacional en el desarrollo de aplicaciones de escritorio.

1.3.1 Tecnologías más usadas en el desarrollo de aplicaciones de escritorio.

El término aplicaciones de escritorio se refiere a aplicaciones que corren en una PC local, es decir, que no son hechas principalmente para correr en un servidor. Dependiendo del tipo de programa que se desee realizar hay diferentes lenguajes en los que se puede programar: Java, C++, Smalltalk, Delphi, VisualBasic, Visual C# .NET, y otros. Es importante tener en cuenta sus características para elegir el correcto. Por ejemplo, para realizar una aplicación que no dependa de la plataforma o SO, siempre se piensa en Java primeramente, pues Java es Multiplataforma, puede correr en Unix, Windows y otros sistemas con sólo instalar la Virtual-Machine o Máquina Virtual.

Un lenguaje de programación generalmente nunca podrá sustituir a otro completamente. Sin embargo, en muchos casos es necesario combinarlos para obtener el resultado deseado. En la actualidad se

está hablando cada vez más de los llamados Metalenguajes, con los cuáles se quiere conseguir el unir varios de los lenguajes más utilizados, o sus propiedades principales, en una misma plataforma. Este es el caso de .NET, una plataforma muy usada a nivel mundial por sus múltiples ventajas. (8)

1.3.1.1 .NET.

La plataforma de desarrollo .NET es un proyecto de Microsoft, una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permite un rápido desarrollo de aplicaciones.

La base de la plataforma .NET la constituye el "framework" o marco de trabajo, éste, denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entornos de ejecución distribuido. (9)

Uno de los entornos de desarrollo integrado (IDE del inglés Integrated Development Environment) de .NET es el **Visual Studio .NET**: un conjunto completo de herramientas de desarrollo para la construcción de aplicaciones Web (ASP) y aplicaciones para escritorio (Visual Basic .NET, Visual C++ .NET, Visual C# .NET y Visual J# .NET), el cual permite compartir herramientas y facilita la creación de soluciones en varios lenguajes. Así mismo, dichos lenguajes aprovechan las funciones de .NET Framework a través de un conjunto común de clases unificadas. Las clases unificadas de .NET proporcionan un método coherente de acceso a las funcionalidades de la plataforma. (10)

MonoDevelop es un IDE libre GNOME diseñado principalmente para C#. Las principales características de MonoDevelop son: manejo de clases, ayuda incorporada, completamiento de código, soporte para proyectos, entre otras. (11) Este IDE sigue estando en desarrollo y si bien es bastante funcional para programar muchas aplicaciones serias, todavía está incompleto por lo que en ocasiones no es recomendable su uso.

C# (leído en inglés "C Sharp" y en español "C Almohadilla") es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Es un lenguaje sencillo, soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo, posibilita la seguridad en el manejo de datos, es un sistema de tipos unificados, toma las mejores características de lenguajes preexistentes como Visual Basic, C++ y los combina en uno solo. (12)

1.3.1.2 Java.

Java es toda una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa. Hoy en día ha cobrado gran fuerza en el ámbito de Internet gracias a su plataforma J2EE. Está compuesta básicamente por 2 elementos: el lenguaje Java y la máquina virtual (Java Virtual Machine).

Una de las principales características que favoreció el crecimiento y difusión del lenguaje Java es su capacidad de que el código funcione sobre cualquier plataforma de software y hardware. Esto significa que el mismo programa escrito para Linux puede ser ejecutado en Windows sin ningún problema. Además es un lenguaje de propósito general, concurrente, basado en clases y orientado a objetos, está diseñado para ser lo suficientemente simple para que los programadores puedan lograr fluidez con el lenguaje. Ofrece toda la funcionalidad de un lenguaje potente, una características importante de Java es que posee una arquitectura neutral, es decir, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará.

Más allá de la portabilidad básica por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar su desarrollo, construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos Unix, Pc o Mac. (15)

Dentro de los IDEs para el desarrollo de aplicaciones usando como lenguaje de programación Java se encuentran NetBeans y Eclipse.

- **NetBeans:** Sun MicroSystems fundó el proyecto de código abierto (en inglés, open source) NetBeans en junio 2000 y continúa siendo su patrocinador. El NetBeans IDE es un entorno de desarrollo integrado - una herramienta de desarrollo Java, escrita puramente sobre la base de la tecnología Java, de modo que puede ejecutarse en cualquier ambiente que ejecute Java, lo cual, por supuesto, es casi en todas partes. Pensado para escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender el IDE NetBeans, es un producto libre y gratuito, sin restricciones de uso. Su código fuente está disponible para su reutilización de acuerdo con la licencia CDDL (Desarrollo Común y Licencia de Distribución). (13)

- **Eclipse:** es una plataforma de desarrollo de código abierto basada en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios. En sí mismo, Eclipse es un

marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (plug-in). Hay plug-ins para el desarrollo de Java (JDT Java Development Tools) así como para el desarrollo en C/C++, COBOL, entre otros. (14)

Que una herramienta sea de código abierto, que sea popular, que existan comunidades que den soporte, y que genere todo el tiempo nuevos desarrollos e investigaciones, son pilares fundamentales a la hora de tener la tranquilidad, que cuando los problemas surjan, o cuando se tenga que innovar en algo, exista alguna comunidad a la cual consultar. Y en este punto Java presenta una mayor ventaja.

Si el desarrollo está apuntado exclusivamente para plataforma Windows, sin duda será mucho más eficiente usar .NET para aprovechar todo el potencial del entorno. Sin embargo, si se requiere un desarrollo multiplataforma, y a pesar de la existencia de proyectos como MONO que intentan migrar .NET más allá de Windows, Java lleva más de 10 años de progreso en tecnología multiplataforma, y existe todo un soporte de nivel profesional para esto.

Java tiene múltiples ventajas: es multiplataforma, robusto, de fácil uso, distribuido, posee gran cantidad de herramientas gratuitas, entre otras, que la hacen superior a .NET y a cualquier otra plataforma o lenguaje de programación, permitiéndole a los desarrolladores de software:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- Combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizados.
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier dispositivo digital.

1.3.2 Metodologías de desarrollo de software.

Dentro del desarrollo de software y a la altiva necesidad de que los proyectos lleguen al éxito y obtener un producto de gran valor y alta calidad para nuestros clientes, es conveniente el estudio de las metodologías de desarrollo, las cuales, permitirán potencializar los grupos de desarrollo, aprovechando al máximo el potencial de todos quienes lo integran.

Es por este motivo que, la selección de una metodología adecuada y robusta, permitirá la flexibilidad apropiada para la consecución de los objetivos y además ofrecerá satisfacer al cliente más allá de las necesidades definidas al inicio del proyecto.
El éxito del producto depende, en gran parte, de la metodología acogida; sea tradicional o ágil. A continuación se describen estos dos grandes enfoques:

1.3.2.1 METODOLOGÍAS TRADICIONALES

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el objetivo de conseguir un software más eficiente y predecible. Para ello, se hace un especial hincapié en la planificación total de todo el trabajo a realizar y una vez que esta todo detallado, comienza el ciclo de desarrollo del producto software. Este planteamiento está basado en el resto de disciplinas de ingeniería, a pesar de que el software no pueda considerarse como la construcción de una obra clásica de ingeniería. Con estas metodologías se lleva trabajando desde hace tiempo y no ha habido en ningún caso ninguna experiencia traumática acerca de su uso. (16)
A continuación se caracterizan las metodologías dentro de esta clasificación más usadas en la actualidad:

RUP (Proceso Unificado de Software).

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

- Inicio, El Objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración, En esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- Transición, El objetivo es llegar a obtener la liberación del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplina de Desarrollo

- Ingeniería de Negocios: Entendiendo las necesidades del negocio.
- Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.

Disciplina de Soporte

- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.
- Distribución: Hacer todo lo necesario para la salida del proyecto

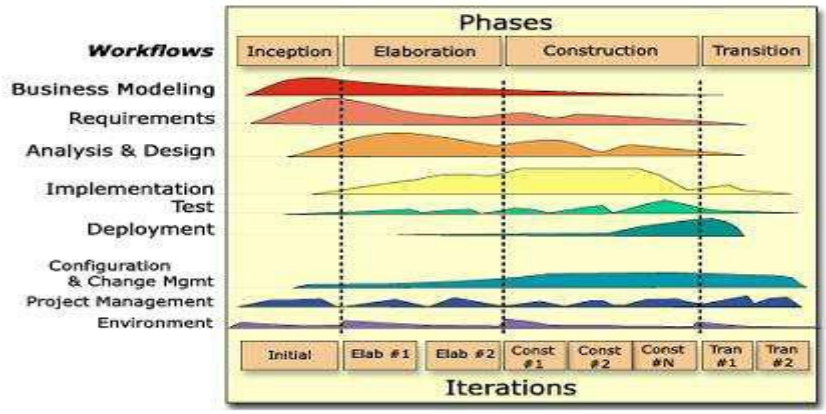


Figura 8 Fases e Iteraciones de la Metodología RUP

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

- Actividades, Son los procesos que se llegan a determinar en cada iteración.
- Trabajadores, Vienen hacer las personas o entes involucrados en cada proceso.

- Artefactos, Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (17)

Microsoft Solution Framework (MSF).

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación. (17)

1.3.2.2 METODOLOGÍAS ÁGILES.

En contraposición a estas metodologías clásicas, en los últimos años ha aparecido un nuevo grupo de metodologías, que se identifica como metodologías ágiles. Aportan como novedad, nuevos métodos de trabajo que apuestan por una cantidad apropiada de proceso. Es decir, ni se pierden en una excesiva cantidad de cuestiones burocráticas, ni defienden tampoco la falta total de procesos. Buscan el equilibrio en la relación proceso/esfuerzo. Algunos de los principales métodos ágiles son: (16)

Extreme Programming (XP).

La Programación Extrema (XP) es una de las metodologías de desarrollo de software más exitosas en la actualidad utilizadas para proyectos de corto plazo, corto equipo y cuyo plazo de entrega era ayer. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto.

Características de XP, la metodología se basa en:

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.

- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.
- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa. (17)

SCRUM.

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (35)

Crystal Methodologies.

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros). (35)

1.3.3 Herramientas CASE (Computer Aided Software Engineering)

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software.

La realización de un nuevo software requiere que las tareas sean organizadas y completadas en forma correcta y eficiente. Las Herramientas CASE fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de software. (18)
A continuacion se caracteriazan algunas de estas herramientas:

Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, importación desde Rational Rose, exportación/importación XMI, generador de informes, editor de figuras, etc. (19)

Soporta un conjunto de lenguajes, tanto en generación de código e ingeniería inversa en: Java, C++, CORBA IDL, PHP, y Python. (20)

En fin, Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.0
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad en múltiples plataformas. (21)

Rational Rose

Rational Rose es una herramienta de modelado visual para el análisis y diseño de sistemas basados en objetos. Se utiliza para modelar un sistema antes de proceder a construirlo. Cubre todo el ciclo de vida de un proyecto:

- Concepción y formalización del modelo.
- Construcción de los componentes.

- Transición a los usuarios.
- Certificación de las distintas fases. (22)

Enterprise Architect

Enterprise Architect 7.0 es una herramienta de construcción y modelado de software de alto rendimiento basado en el estándar de UML 2.1, diseñada para ayudar a construir software robusto y fácil de mantener. Brinda una solución de modelado verdaderamente ágil, fácil de usar, rápido, flexible y con una interfaz gráfica amigable. Soporta generación e ingeniería inversa de código fuente para muchos lenguajes populares, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP. Es una herramienta multi-usuario, con seguridad y administración de permisos incorporada. Soporta diferentes repositorios basados en DBMS (Sistemas Manejadores de BD), incluyendo Oracle, SQL Server, My SQL, PostgreSQL. Brinda soporte para control de versiones y posee bajos costos de licencias. (36)

1.3.4 El lenguaje Unificado de Modelado UML.

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. UML también contiene construcciones organizativas para agrupar los modelos en paquetes, lo que permite a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes, y para gestionar las versiones de las unidades del modelo, en un entorno de desarrollo complejo. Contiene construcciones para representar decisiones de implementación y para elementos de tiempo de ejecución en componentes.

El lenguaje UML estandariza los artefactos y la notación, pero no define un proceso oficial de desarrollo. He aquí algunas de las razones que explican esto:

1. Aumentar las probabilidades de una aceptación generalizada de la notación estándar del modelado, sin la obligación de adoptar un proceso oficial.
2. La esencia de un proceso apropiado admite mucha variación y depende de las habilidades del personal, de la razón investigación-desarrollo, de la naturaleza del problema, de las herramientas y de muchos otros factores. (23)

1.4 Estudio del estado del arte de los simuladores de planificación de procesos existentes.

La enseñanza en cualquier materia tiene como fin el aprendizaje eficaz por parte del alumnado. Hoy en día el empleo de nuevas tecnologías, como simuladores, multimedia, Internet, etc., nos permite elaborar nuevas técnicas de aprendizaje, en beneficio de una educación más extendida y de mayor calidad.

Específicamente los simuladores crean un entorno simulado, que permite al alumno, mediante la exploración y la experimentación, adquirir y reafirmar sus conocimientos. El empleo de los mismos permite la sustitución de los métodos clásicos de enseñanza, basados en la explicación con ayuda de la pizarra, por otros en los que el alumno puede tener mayor participación permitiendo el desarrollo de las habilidades cognitivas de éste. Es por tales razones que actualmente se desarrollan herramientas de simulación de apoyo a la docencia, concretamente se verán a continuación algunas herramientas de simulación de algoritmos de planificación de procesos existentes en la actualidad:

HERRAMIENTA SOFTWARE PARA LA SIMULACIÓN DE UN PLANIFICADOR DE PROCESOS: este planificador simula el proceso de planificación de la CPU para cada una de las siguientes políticas de planificación: FCFS, SPN, SRT, RR, HRRN, planificación por prioridades y MFQ. Compara los distintos algoritmos de planificación, dependiendo de los diferentes criterios de planificación seleccionados: utilización de la CPU, tiempo de espera, tiempo de retorno, tiempo de respuesta, etc. También permite modificar las características de los procesos que componen la carga del trabajo a manejar por la aplicación en la simulación. (25) A pesar de todas esas funcionalidades que tiene el simulador, así como sus ventajas este no puede ser usado en la asignatura Sistemas Operativos de la UCI pues tiene el inconveniente que no implementa los algoritmos SJF y SRTF que son básicos en la impartición del contenido de planificación de procesos de dicha asignatura.

ALGORITMO DE PLANIFICACION POR TURNO ROTATORIO O ROUND ROBIN: esta herramienta presenta una simulación creada con el objetivo de hacer entender al usuario el algoritmo de planificación de procesos por Turno Rotatorio o Round Robin. (26) Por lo que no es eficiente utilizarlo

en la asignatura Sistemas Operativos de la Universidad de las Ciencias Informáticas pues en dicha asignatura no se trata solamente este algoritmo.

PLANP (PLANificación de Procesos) es una aplicación que tiene como finalidad el servir de apoyo a las asignaturas que tengan en sus temarios aspectos relacionados con los Sistemas Operativos. Esta herramienta visual muestra la evolución de una planificación previamente configurada con determinados conceptos significativos (procesos, dispositivos, política,...). Implementa los algoritmos: FCFS (First-Come,First-Served), SRTN (Shortest Remaining Time Next), por reparto de tiempo (Round Robin), con expropiación basada en prioridades (Event Driven). A pesar de las ventajas que trae consigo presenta dificultad a la hora de entrada de datos, la interfaz es poco amigable y además no brinda las estadísticas que se necesitan, razones por la cual no es óptimo usarlo en la asignatura SO de la UCI. (37)

Conclusiones

El simulador de algoritmos de planificación de procesos que se propone en el presente trabajo tiene como objetivo mostrar el funcionamiento de los aspectos de planificación de la CPU en un sistema de multiprogramación. Para el desarrollo de esta herramienta en este capítulo se definieron los principales conceptos que ayudarán a comprender como un Sistema Operativo maneja la planificación de procesos. Además se hizo un estudio de las diferentes tecnologías y metodologías más usadas en la actualidad para el desarrollo de aplicaciones de este tipo. Este estudio posibilitó la selección de:

- La tecnología Java con su IDE de desarrollo NetBeans 6.0 para la implementación del simulador, por las ventajas que éste trae consigo, tales como: es un producto libre, gratis, sin restricciones de uso, multiplataforma, es decir, puede correr en Unix, Windows y otros SO con solo instalar la Máquina Virtual de Java. Además es robusto, posee gran cantidad de herramientas gratis, entre otras ventajas.

- La herramienta CASE Visual Paradigm porque es una herramienta multiplataforma de modelado visual UML, muy potente y fácil de utilizar. Permite una excelente interoperabilidad con IDEs de desarrollo para Java. Provee soporte para la generación de código e ingeniería inversa para múltiples lenguajes de programación.

- La metodología RUP (Rational Unified Process) por ser un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software con diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto. Sin embargo, hay tres características fundamentales que lo hacen una metodología robusta y poderosa: es dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Además provee una documentación detallada, a diferencia de las metodologías ágiles que son menos precisas en este sentido.

- El lenguaje de Modelado UML como lenguaje representativo porque es considerado hoy en día el lenguaje estándar para el análisis y diseño de sistemas computacionales. Permite modelar sistemas utilizando técnicas orientadas a objetos (OO). Permite documentar todos los artefactos de un proceso de desarrollo. Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA

En este capítulo se representa el modelo del dominio como primer paso para entender el contexto del problema planteado así como el entorno donde será utilizado el sistema. Se describen los requisitos funcionales y no funcionales, además del diagrama de casos de uso del sistema como representación de los procesos identificados.

2.1 Modelo de Dominio

Un Modelo de Dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (23)

La modelación del dominio tiene como objetivo fundamental la comprensión y descripción de las clases más importantes en el sistema.

2.1.1 Comprensión del contexto del sistema mediante el modelo de dominio.

Todos los objetos dominio o clases (para emplear una terminología más precisa) que conforman el siguiente modelo de dominio que responde al sistema propuesto, se obtuvieron a partir de las entrevistas no estructuradas realizadas a los expertos del dominio.

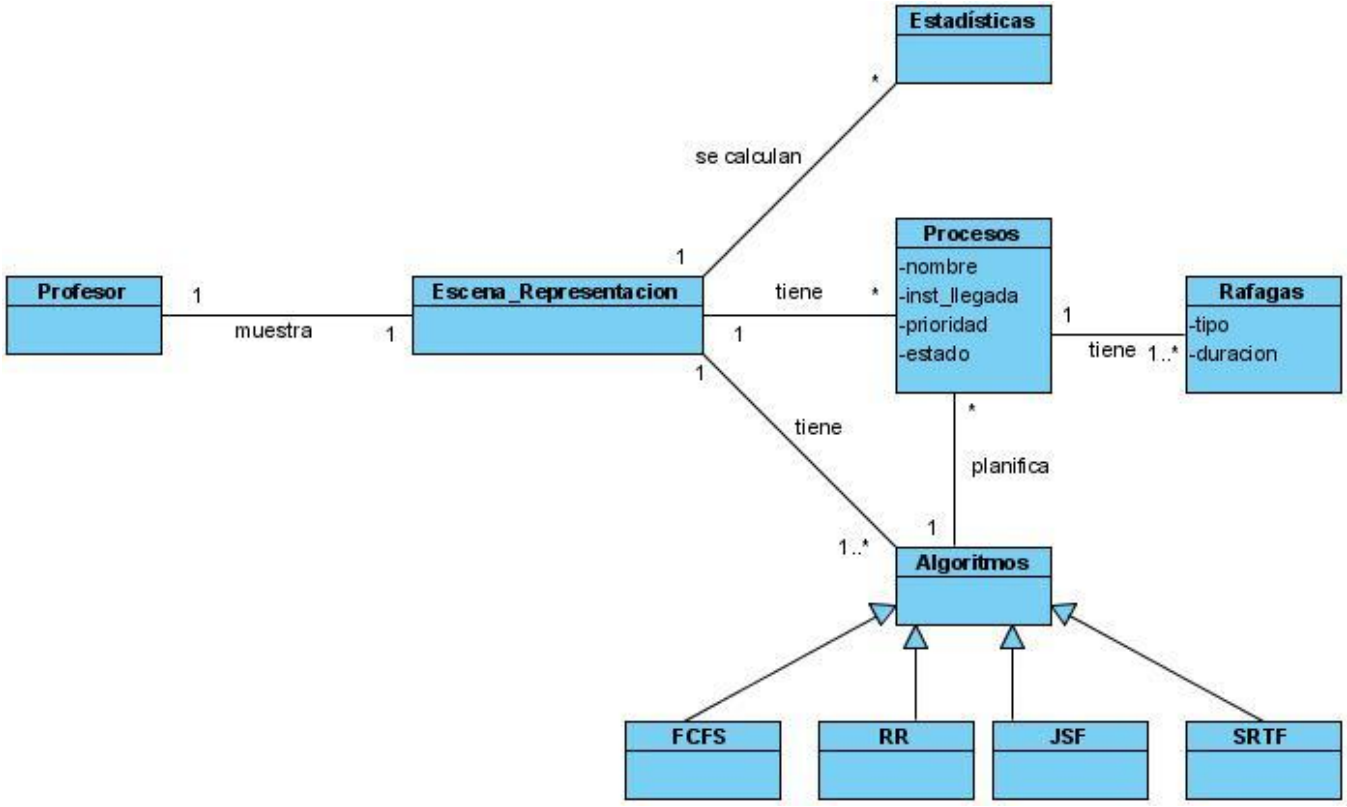


Figura 9 Modelo de Domino

Descripción textual del modelo de dominio

El profesor muestra en una escena de representación, esta puede ser la pizarra, un power point (ppt), etc., la planificación de procesos a través de un algoritmo (RR, FCFS, SJF, SRTF), y la forma en la que se calculan las estadisticas representativas de dicha planificación.

2.2 Glosario de Términos del Dominio.

Se realiza el siguiente glosario de términos para facilitar la comprensión de los conceptos presentes en el entorno.

Profesor: Es la persona que enseña determinada materia de estudio de la mejor manera posible para el alumno.

Escena_ Representación: Muestra la forma en la el profesor representa de manera visual (a través de la pizarra, de un power point (ppt), etc.) la planificación de procesos a través de un algoritmo de planificación (FCFS, RR, SJF, SRTF).

Proceso: Es un programa en ejecución.

Ráfaga: Pueden ser de 2 tipos: CPU o Entrada/Salida (E/S).

Ráfaga de CPU: representa el tiempo que un proceso va a estar haciendo uso del recurso procesador para su ejecución.

Ráfaga E/S: representa el tiempo que un proceso va a estar bloqueado, en espera de un evento como: acceso a ficheros, interacción con dispositivos de E/S, entre otros, que requiere para continuar su ejecución.

Algoritmo: Son técnicas de planificación, basadas en una política específica para asignar el procesador a un proceso y permitirle a los demás la transición de un estado a otro.

FCFS: Es el algoritmo de planificación más simple, se ejecutan los procesos según el orden de llegada a la cola de listos. Esta técnica es por naturaleza sin derecho de apropiación. La instrumentación del FCFS es muy simple y solo se requiere que la cola de listos se opere realmente como una cola. Cada proceso que llega a ella se coloca (enlaza) al final y siempre se le asigna el CPU al proceso que está en la cabeza, eliminándose de la lista.

RR: En este caso, la cola de listos es tratada como una cola circular (lo que no implica que tenga que ser instrumentada en esta forma) y el planificador gira alrededor de la cola, asignando el CPU a cada proceso. El proceso que recibe el CPU lo puede utilizar durante un quantum o ranura de tiempo (10 a 100ms), transcurrido éste lo pierde hasta que le toca de nuevo. Por supuesto, si por alguna razón se tiene que bloquear o termina, también pierde el procesador. Es evidente que en este algoritmo existe el derecho de apropiación, pues aunque un trabajo esté en condiciones de seguir su ejecución, si se venció su quantum de tiempo se le quitará el CPU.

SJF: En este algoritmo se le asocia a cada trabajo la duración de su siguiente ráfaga de CPU, de tal manera que el que tenga la próxima ráfaga más pequeña es el primero que se atiende. En caso de igualdad de utiliza el FCFS. Este algoritmo es no apropiativo.

SRTF: Es la implementación apropiativa del SJF. Sus características son las mismas.

Estadísticas: son los cálculos realizados para caracterizar el algoritmo utilizado para la planificación en función de: tiempo de retorno medio, tiempo de respuesta medio, por ciento de uso de la CPU, entre otros

2.3 Requerimientos.

La parte más difícil a la hora de construir un sistema es precisamente saber qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallados, incluyendo todas las interfaces con personas, máquinas y otros sistemas. Entonces, la tarea más importante que el Ingeniero de Software hace para el cliente es la extracción iterativa y el refinamiento de los requerimientos del producto. (32)

Los requerimientos son la base fundamental de cualquier software que se desee desarrollar, ya que determinan las capacidades y cualidades que debe cumplir el software para garantizar una buena calidad. Se pueden dividir en dos grandes grupos: los Requisitos Funcionales y los No Funcionales.

2.3.1 Requisitos Funcionales:

Los requisitos funcionales son los que responden a: ¿qué debe hacer el sistema?, y describen las capacidades o condiciones que este debe cumplir.

Los requisitos funcionales del sistema propuesto, responden a:

El sistema debe permitir:

R1. Administrar datos de procesos.

R1.1. Adicionar datos de proceso.

R1.2. Modificar datos de proceso.

R1.3. Eliminar datos de un proceso o de varios procesos.

R2. Permitir la planificación de procesos.

R2.1 Permitir la selección de un algoritmo de planificación de procesos.

R2.2 Planificar los procesos existentes a través del algoritmo seleccionado.

R3. Visualizar a través de una simulación la planificación de procesos.

R3.1 Visualizar la simulación instante a instante o de forma automática a través de un diagrama de WANTT.

R3.2 Permitir configurar la visualización de la simulación de manera lenta, normal o rápida en caso de estar simulando automáticamente.

R3.3 Permitir al usuario pausar la simulación en el momento que este desee.

R3.4 Permitir al usuario detener la simulación y volver al inicio.

R4. Emitir estadísticas de la simulación.

R4.1 Coleccionar los datos de cada proceso, en cuanto a instante de llegada y de fin, tiempo que estuvo en estado listo, tiempo que estuvo en estado bloqueado, y tiempo que estuvo en ejecución.

R4.2 Calcular y mostrar estadísticas relacionadas con: tiempo de espera medio, tiempo de retorno promedio y por ciento de uso del CPU.

R4.3 Mostrar las gráficas que representan el tiempo de espera medio y tiempo de retorno medio de cada proceso.

R5. Manipular fichero.

R5.1 Salvar para un fichero los datos de los procesos.

R5.2 Cargar datos de procesos desde un fichero.

R6. Mostrar ayuda.

R6.1 Mostrar datos del simulador.

R6.2 Mostrar breve información de los 4 algoritmos de planificación implementados.

R6.3 Mostrar datos de contenido de los módulos: datos de procesos, simulación y estadísticas.

R6.4 Mostrar datos del autor.

2.3.2 Requisitos No Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Requerimientos de apariencia o interfaz externa.

La aplicación debe ser diseñada con una interfaz amigable y sencilla, de forma tal que el usuario navegue por ella sin dificultad alguna.

Requerimientos de Portabilidad

El sistema podrá ser ejecutado sobre los Sistema Operativos: Linux y Windows, de ahí su característica de ser multiplataforma.

Requerimientos de Rendimiento

Un tiempo de respuesta rápido para evitar que los usuarios se sientan desmotivados e intenten cerrar la aplicación.

Requerimientos de Usabilidad

El sistema podrá ser usado por cualquier persona que posea conocimientos básicos de computación.

Requerimientos de diseño e Implementación.

El sistema será implementado en Java, usando el IDE NetBeans 6.0, se utilizará como herramienta de modelado Visual Paradigm, y un paradigma de Programación Orientado a Objetos.

Requerimientos de Software

Se debe disponer de la Máquina Virtual de Java versión 1.3.

Requerimientos de Hardware

Se requiere disponer de una computadora con 256 Mb de RAM y 50 Mb de capacidad del disco duro.

Requerimientos de Seguridad

El usuario tendrá control total de la aplicación, sin restricción alguna.

Requerimientos de Soporte

El sistema debe propiciar su mejoramiento.

Una vez recopilados los requisitos, se crean un conjunto de escenarios que identifican una línea de utilización para el sistema que va a ser construido. Los escenarios, llamados *casos de uso*, facilitan una descripción de cómo el sistema se usará. Para crear un caso de uso, se debe primero identificar los diferentes tipos de personas (o dispositivos) que utiliza el sistema o producto. Estos *actores* actualmente representan papeles que la gente (o dispositivos) juegan como impulsores del sistema. Definido más formalmente, un actor es algo que se comunica con el sistema o producto y que es externo al sistema en sí mismo. (33)

2.4 Actores del Sistema.

Los actores del sistema:

- No son parte de él.
- Pueden intercambiar información con él.
- Pueden ser un recipiente pasivo de información.
- Pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado.

Tabla 2 Actores del Sistema Propuesto.

Nombre del actor	Descripción
Usuario	Representa a las personas que van a interactuar con el sistema, ya sea el profesor, el estudiante o cualquier persona que requiera de su uso.

2.5 Casos de Uso definidos.

Cada forma en que los actores usan el sistema se representa con un Caso de Uso. Los Casos de Uso son “fragmentos” de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. De manera más precisa, un caso de uso especifica una secuencia de acciones que el sistema puede llevar a cabo interactuando con sus actores, incluyendo alternativas dentro de la secuencia.

Los casos de uso definidos son los siguientes:

- Gestionar proceso
- Gestionar estadísticas.
- Gestionar datos de procesos
- Simular algoritmo
- Mostrar ayuda

2.6 Modelo de Casos de Uso del Sistema.

El modelo de casos de uso es un modelo del sistema que contiene actores, casos de uso y sus relaciones. Permite que los desarrolladores de software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre las condiciones y posibilidades que debe cumplir el sistema, El modelo de casos de uso sirve como acuerdo entre clientes y desarrolladores, y proporciona la entrada fundamental para el análisis, el diseño y las pruebas. (23)

Diagrama de Casos de Uso del Sistema

Los diagramas de casos de uso se crean para visualizar las relaciones entre los actores y los casos de uso.

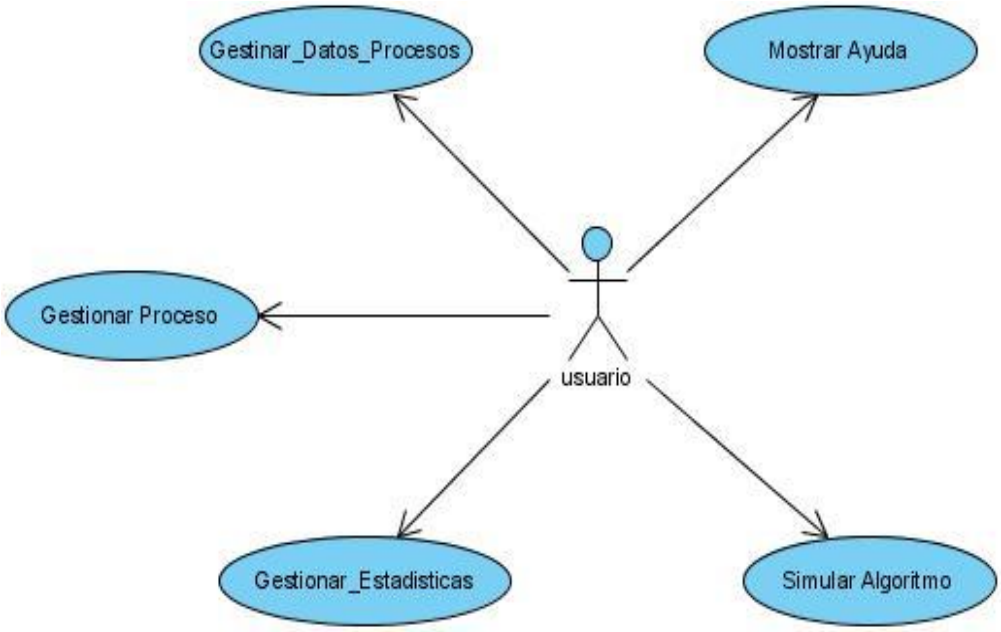


Figura 10 Diagrama de Casos de Uso del Sistema

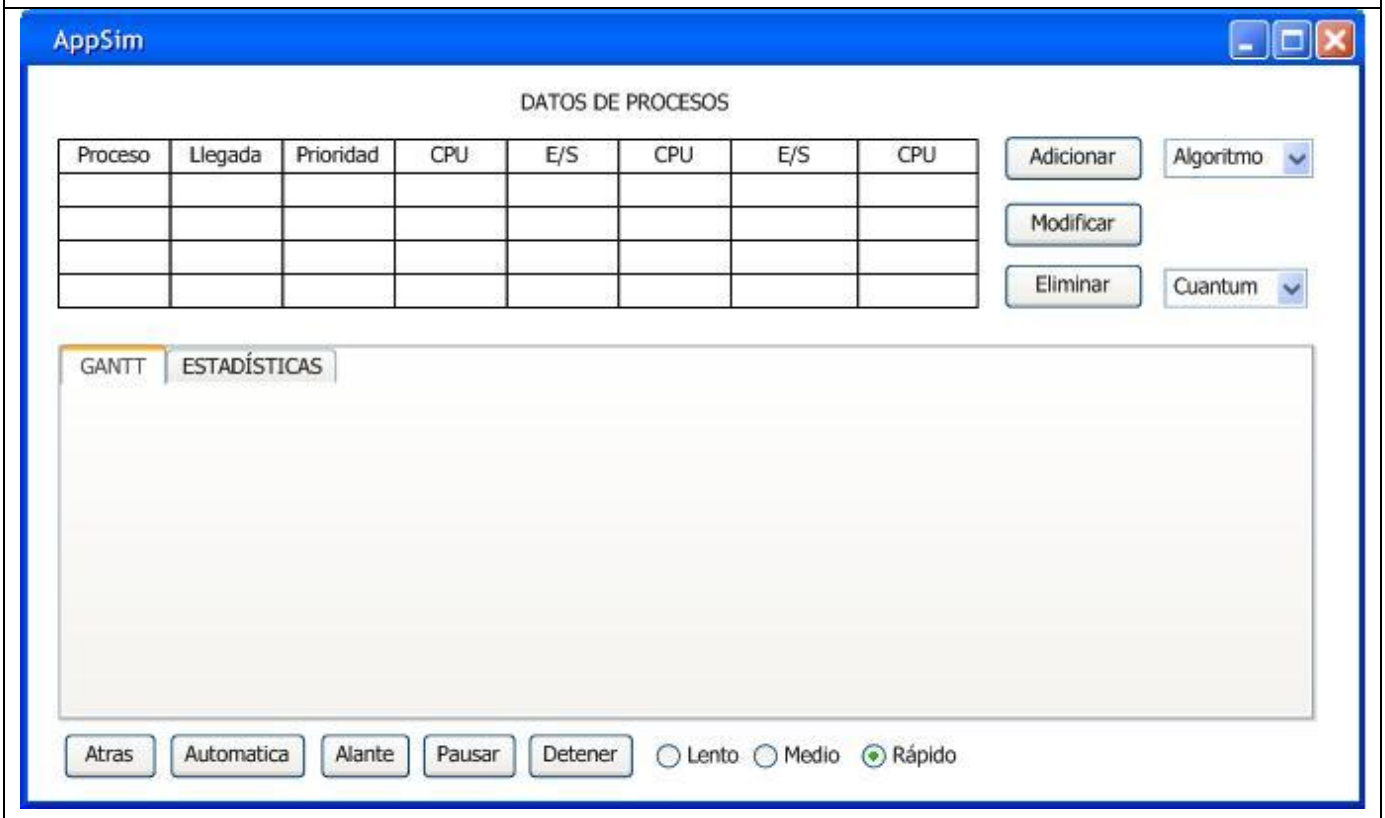
2.7 Descripción de los Casos de Uso del sistema.

Tabla 3 Descripción CU Gestionar Proceso

Caso de uso:	**Gestionar Procesos**
Actores:	Usuario
Propósito: Adicionar, modificar y eliminar procesos.	
Resumen: El caso de uso se inicia cuando el usuario accede a la interfaz del sistema y selecciona la acción que desea realizar, ya sea adicionar, modificar o eliminar procesos, y termina con la ejecución de dicha acción.	
Precondiciones:	En caso de que se desee modificar o eliminar un proceso, este

	debe encontrarse registrado en el sistema.
Referencias:	R1.1, R1.2, R1.3
Poscondiciones:	Se adiciona, modifica o eliminan procesos.

Interfaces:



Pantalla Principal

Pantalla Adicionar Proceso

Pantalla Modificar Proceso

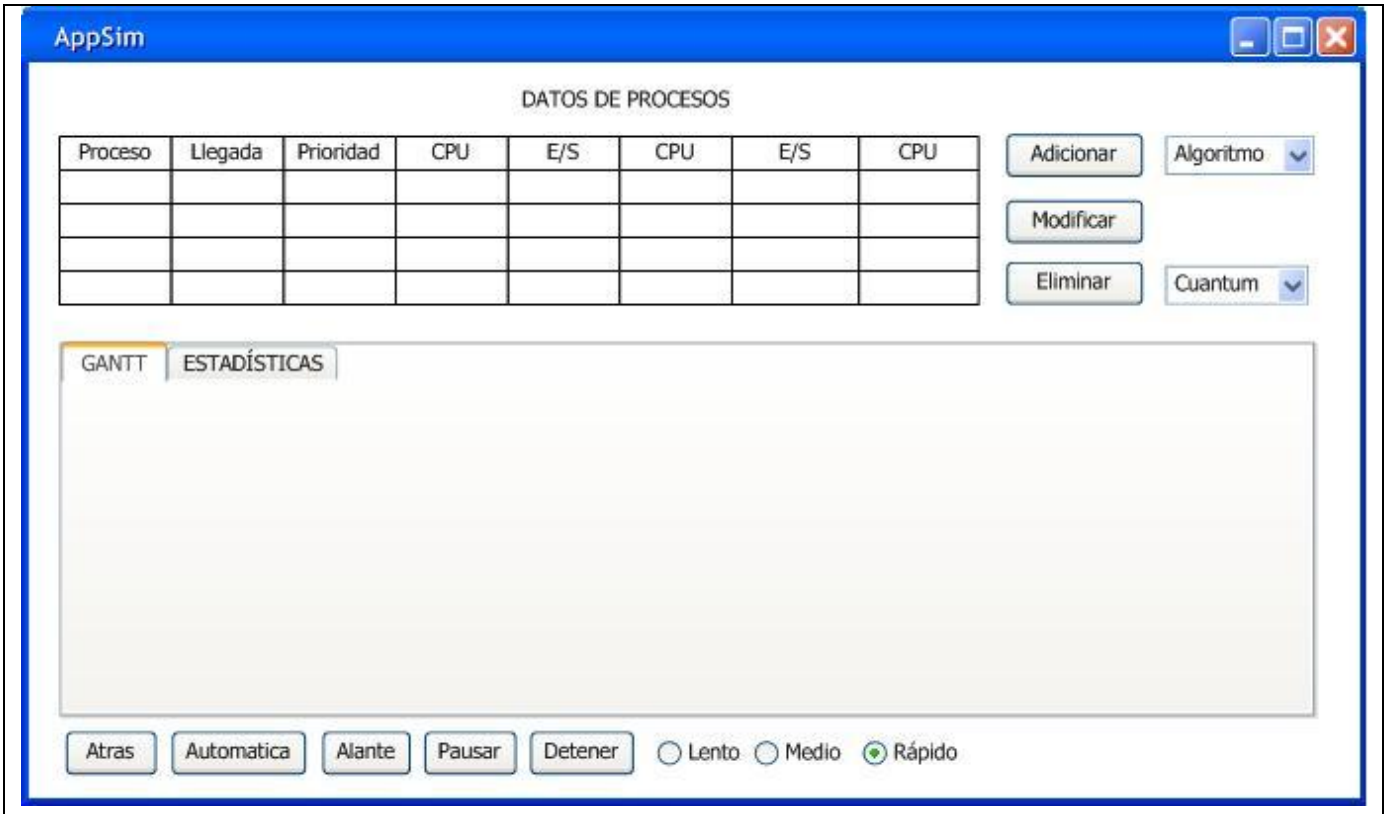
Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario accede a la interfaz del sistema.	2.El sistema muestra una serie de acciones a realizar: • Adicionar un proceso. • Eliminar procesos. • Modificar un proceso.
3. El usuario elige la acción a realizar. Si elige Adicionar un proceso, **Ir a sección "Adicionar Proceso"** Si elige Eliminar procesos**, Ir a sección "Eliminar Procesos"** Si elige Modificar un proceso, **Ir a sección "Modificar Proceso".**	
Sección Adicionar Proceso	
1. El usuario selecciona la opción de adicionar un nuevo proceso.	2. El sistema muestra el formulario Adicionar Proceso con los campos: • Nombre • Instante de llegada • Prioridad

	• Ráfagas que posee el proceso con sus respectivas duraciones.
3. Introduce los datos del proceso a adicionar.	4. El sistema verifica que las ráfagas sean mayores que cero, que la última ráfaga es de CPU, la validez de los datos introducidos, inserta el nuevo proceso en el sistema y muestra la lista de procesos actualizada en el formulario principal.
Sección Modificar Proceso	
	1. El sistema muestra todos los procesos existentes.
2. El usuario selecciona el proceso que desea modificar y elige la opción modificar.	3. El sistema muestra la interfaz de modificar proceso, con la información del proceso seleccionado.
4. El usuario realiza los cambios pertinentes.	5. El sistema chequea la validez de los datos modificados, actualiza los datos del proceso modificado y muestra la lista de procesos actualizada con la modificación en el formulario principal.
Sección Eliminar Proceso	
	1. El sistema muestra todos los procesos existentes.
2. El usuario elige el o los procesos a eliminar y elige la opción eliminar.	3. El sistema muestra un mensaje para confirmar si realmente desea eliminar el o los procesos seleccionados.
4. El usuario acepta o cancela la eliminación del o de los procesos seleccionados.	5. El sistema elimina el o los procesos seleccionados y muestra la lista de procesos actualizada.
Cursos Alternos:	
Sección: Adicionar Proceso. Si alguno o algunos de los datos entrados son incorrectos, el sistema muestra un mensaje de error y devuelve el control a la acción 3. Si la última ráfaga no es de CPU, el sistema muestra un mensaje de error y devuelve el control a la acción 3. Si el usuario intenta adicionar más de 5 ráfagas, el sistema muestra un mensaje de error informando que son como máximo 5 ráfagas.	

Sección: Modificar Proceso. Si no hay procesos en el sistema o si no hay un proceso seleccionado, y el usuario escoge la opción de modificar, el sistema muestra un mensaje de error informando que no puede realizarse dicha operación. Si existe algún problema en la información modificada, muestra un mensaje de error y devuelve el control a la acción 4. **Sección: Eliminar Proceso.** Si no hay procesos en el sistema o seleccionados, y el usuario escoge la opción de eliminar, el sistema muestra un mensaje de error informando que no es posible efectuar dicha operación. Si el usuario cancela la eliminación de los procesos seleccionados, no se eliminan.

Tabla 4 Descripción CU Simular Algoritmo

Caso de uso:	**Simular Algoritmo.**
Actores:	Usuario
Propósito: Según el algoritmo seleccionado, simular la planificación de los procesos existentes en el sistema.	
Resumen: El caso de uso se inicia cuando el usuario accede a la interfaz del sistema, selecciona el algoritmo por el cual desea simular la planificación de los procesos existentes y luego escoge cualquiera de las opciones para simular: instante a instante (pudiendo ir hacia atrás) o de forma automática, especificando la velocidad para esta última. Además se puede detener (volver al inicio) la simulación o pausarla en caso de que se haya seleccionado la opción de simular automáticamente. Este caso de uso termina con la ejecución de la acción seleccionada por el usuario.	
Precondiciones:	Que en el sistema exista al menos un proceso y se haya seleccionado un algoritmo.
Referencias:	R2.1, R2.2, R3.1, R3.2, R3.3, R3.4
Poscondiciones:	Se simula la planificación de los procesos existentes a través del algoritmo seleccionado.
Interfaces:	



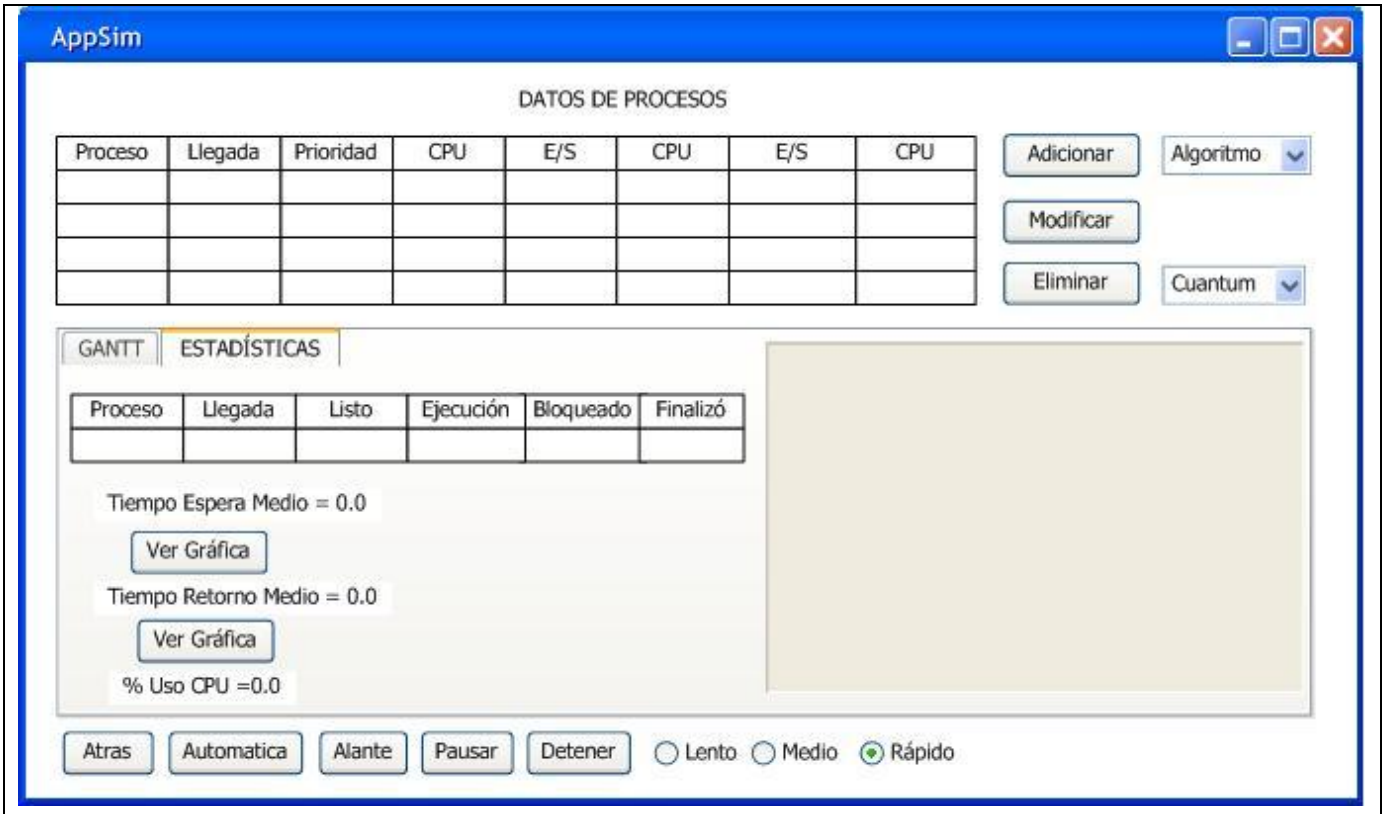
Pantalla de Simulación

Curso normal de eventos:	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona el algoritmo mediante el cual va a simular la planificación de los procesos existentes y elige la forma en la que realizará la simulación (instante a instante o de forma automática, para esta última configura la velocidad).	2. Planifica los procesos existentes según el algoritmo seleccionado y según la forma de simulación escogida, a través de un diagrama de Gantt.
Cursos Alternos	
Si no se selecciona ningún algoritmo, se muestra un mensaje de error, y se devuelve el control a la acción 1. Si no hay procesos en el sistema, se muestra un mensaje de error informando que no se puede llevar a cabo la simulación. Si escoge la opción de planificar instante a instante, en el momento que desee después de haber	

comenzado la simulación y antes de terminarse, puede ir hacia atrás instante a instante también.

Tabla 5 Descripción CU Gestionar_ Estadísticas

Caso de uso:	**Gestionar_Estadísticas**
Actores:	Usuario
Propósito: Mostrar los principales valores de rendimiento de cada proceso y del algoritmo seleccionado para la planificación.	
Resumen: El caso de uso se inicia cuando el usuario después de haber realizado una simulación selecciona la pestaña de estadística, en la cual se le muestran los valores de: tiempo en estado listo, en estado ejecución, en estado bloqueado, instante de llegada e instante de salida, de cada proceso, con los cuales se calculan y muestran los parámetros de rendimiento: tiempo de espera medio, tiempo de retorno medio y por ciento de uso de la CPU. Para los dos primeros si el usuario desea se muestran sus gráficas representativas, pero no ambas al mismo tiempo, sino una primero y la otra después en el orden que el usuario lo prefiera. Este caso de uso termina con la ejecución de la acción seleccionada por el usuario.	
Precondiciones:	Que hayan procesos en el sistema, un algoritmo seleccionado, y se halla realizado la simulación.
Referencias:	R4.1, R4.2, R4.3
Poscondiciones:	Se muestran los valores de: tiempo en listo, en ejecución, en bloqueado, instante de llegada e instante de salida, de cada proceso, y los parámetros de rendimiento: tiempo de espera medio, tiempo de retorno medio y por ciento de uso de la CPU. Además se muestran en caso que el usuario desee las gráficas representativas de los 2 primeros parámetros de rendimiento.
Interfaces:	



Pantalla de Estadísticas

Curso normal de eventos	
Acción del actor	**Respuesta del sistema**
1 .El usuario selecciona la pestaña de estadísticas después de haber realizado la simulación.	2. El sistema muestra una tabla con los valores de: tiempo en estado listo, en estado ejecución, en estado bloqueado, instante de llegada e instante de salida de cada proceso, así como el tiempo de retorno medio y de espera medio, y da la opción de ver las gráficas que representan las estadísticas: tiempo de retorno medio y de espera medio.
4. Selecciona la opción de ver la gráfica que representa el tiempo de retorno medio o de espera medio, en el orden que desee.	5. El sistema muestra la gráfica correspondiente a la opción que el usuario seleccionó.
Cursos Alternos:	
Si el usuario selecciona la pestaña de estadísticas sin haber realizado antes una simulación o sin	

haber terminado una comenzada, no se mostrarán los valores de estadísticas y si selecciona la opción de ver la gráfica de tiempo de retorno medio o tiempo de espera medio, el sistema mostrará un mensaje de error informando que no es posible realizar dicha acción.

Tabla 6 Descripción CU Gestionar_Datos_Procesos

Caso de uso:	**Gestionar_Datos_Procesos.**
Actores:	Usuario
Propósito: Salvar y cargar datos de procesos en un fichero.	
Resumen: El caso de uso se inicia cuando el usuario accede a la interfaz del sistema y selecciona en el menú del formulario principal la opción de abrir datos de procesos de un fichero existente o guardar los datos de los procesos existentes en un fichero. Y termina con la ejecución de la opción seleccionada.	
Precondiciones:	En caso de que se desee abrir datos de un fichero, este debe existir, y además debe estar en el formato que la aplicación lee los ficheros, es decir, con extensión .app. En caso de que se desee salvar datos de procesos en un fichero, en el sistema debe existir por lo menos un proceso.
Referencias:	R5.1, R5.2
Poscondiciones:	Se guardan los datos de los procesos existentes en un fichero o se abren los datos de procesos del fichero seleccionado.
Curso normal de eventos:	
Acción del actor	**Respuesta del sistema**
	1. El sistema muestra un menú con las opciones de: “Abrir” y “Guardar” datos de procesos de un fichero.
2.Si el usuario selecciona la opción de: • Abrir **“Ir a la sección cargar datos de fichero”.** • Guardar ¨**Ir a la sección salvar datos para un fichero**¨	
Sección Cargar Datos de Fichero	

1. El usuario selecciona en el menú del formulario principal la opción de “Abrir”.	2. El sistema muestra un formulario en el cual el usuario busca el fichero que desea abrir.
3. El usuario busca y selecciona el fichero a abrir.	3. El sistema abre el fichero seleccionado por el usuario y visualiza esos datos en el formulario principal.
Sección Salvar Datos para un Fichero	
1. El usuario selecciona en el menú del formulario principal la opción de “Guardar”.	2. Verifica que exista al menos un proceso en el sistema, en caso afirmativo muestra un formulario en el cual el usuario especifica el lugar donde quiere guardar el archivo y el nombre del mismo.
3. El usuario busca el directorio donde desea guardar el archivo y especifica el nombre del mismo.	4. Verifica que la ubicación seleccionada sea válida, y si lo es, guarda los datos en el fichero creado.
Cursos Alternos:	
Sección Cargar Datos de Fichero: Si el usuario selecciona un fichero sin el formato correcto, el sistema muestra un mensaje de error informando que no es posible efectuar dicha acción. **Sección Salvar Datos para un Fichero** Si la ubicación selecciona para guardar el fichero es inválida se muestra un mensaje de error informando que no es posible efectuar dicha acción. Si no existen procesos en el sistema, se muestra un mensaje de error informando que no existen datos para guardar.	

Tabla 7 Descripción CU Mostrar Ayuda

Caso de uso:	**Mostrar Ayuda**
Actores:	Usuario
Propósito: Mostrar información del simulador, en sentido general, de su contenido y de los algoritmos que implementa.	

Resumen: El caso de uso se inicia cuando el usuario accede a la interfaz del sistema, selecciona en el menú principal la opción de ayuda, y especifica el tema sobre el cual requiere obtener información: sobre AppSim, sobre Contenido, o Sobre Algoritmos. Y termina con la ejecución de la opción especificada.	
Precondiciones:	-
Referencias:	R6.1, R6.2, R6.3, R6.4
Poscondiciones:	Se muestra la información que el usuario solicitó.
Curso normal de eventos:	
Acción del actor	**Respuesta del sistema**
1. El usuario accede a la interfaz del sistema y selecciona en el menú principal la opción de Ayuda.	2.Muestra 3 opciones: • Mostrar Ayuda sobre AppSim • Mostrar Ayuda sobre contenido. • Mostrar ayuda sobre algoritmos implementados.
3. Escoge alguna de las opciones de ayuda especificadas.	4. Muestra información concerniente a la opción seleccionada por el usuario.

Conclusiones

Debido al bajo nivel de estructuración de los procesos del negocio y para lograr una mejor comprensión del mismo se definió en este capítulo el modelo de dominio, pues este ayuda a comprender de forma general los eventos que suceden en el entorno en el que trabaja el sistema. También se definieron los requisitos que debe tener el mismo, y fueron seleccionados los actores y los casos de uso que se utilizaron en la realización del diagrama de casos de uso.

En este capítulo se ganó claridad en cuanto a la concesión del sistema a construir y se sentaron las bases para las restantes fases de los procesos de diseño e implementación.

CAPÍTULO 3 DISEÑO DEL SISTEMA

Tras la descripción, en el capítulo anterior, de las funcionalidades deseadas del sistema propuesto, se hace imprescindible concretar cómo se desarrollará el mismo, para lo cual en este capítulo se define su diseño a través de la organización de su arquitectura, y la realización de los diagramas de clases y de secuencia.

3.1 Diseño del sistema propuesto.

En el diseño se modela el sistema y se encuentra su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen.

Según la Ayuda del Rational, se puede definir como propósitos del diseño:

- Transformar los requerimientos en un diseño de cómo el sistema debe ser.
- Desarrollar una robusta arquitectura del sistema.
- Adaptar el diseño para que se corresponda con el entorno de implementación, diseñando sus funcionalidades.

3.1.1. Arquitectura.

La Arquitectura de Software, según IEEE 1471-2000, es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

En el libro dedicado a la arquitectura de software, Bass y sus colegas identifican tres razones fundamentales por las cuales la arquitectura es importante:

1. Las representaciones de la arquitectura facilitan la comunicación entre todas las partes interesadas en el desarrollo de un sistema basado en computadoras.
2. Destaca decisiones tempranas de diseño, teniendo profundo impacto en todo el trabajo en la ingeniería.
3. Constituye un modelo relativamente pequeño comprensible de cómo está estructurado el sistema y como trabajan juntos sus componentes.

CAPÍTULO 3 DISEÑO DEL SISTEMA

La descripción de la arquitectura del sistema propuesto está basada en tres etapas fundamentales:

1. Elección del estilo arquitectónico.

2. Selección de los patrones de diseño.

3. Diseño de componentes.

3.1.1.1 Estilo Arquitectónico.

Un estilo arquitectónico define las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de software. En una forma más específica, un estilo determina el vocabulario de componentes y conectores que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas.

Algunos de los principales estilos arquitectónicos que se usan en la actualidad están divididos por Clases de Estilos que engloban una serie de estilos específicos:

- Estilos de Flujo de datos.
  - Tuberías y Filtros.
- Estilos centrados en datos.
  - Arquitecturas de Pizarra o repositorio.
- Estilos de Llamada y Retorno.
  - Modelo – Vista – Controlador (MVC).
  - Arquitectura en Capas.
  - Arquitectura Orientada a Objetos.
  - Arquitectura basada en Componentes.
- Estilo de Código Móvil.
  - Arquitectura de Máquinas Virtuales.

- Estilos Peer – To – Peer.
  - Arquitectura basada en Eventos.
  - Arquitecturas Orientas a Servicios (SOA).

Estos estilos son importantes porque:

- Sirven para sintetizar y tener un lenguaje que describa la estructura de las soluciones.
- Pocos estilos abstractos encapsulan una enorme variedad de configuraciones concretas.
- Definen los patrones posibles de las aplicaciones.
- Permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales.

Los estilos arquitectónicos son el nivel de abstracción mayor para estructurar un sistema, la elección del mismo está dada por el tipo de aplicación que se vaya a desarrollar.

Estilos Arquitectónicos Empleados en la Solución.

Arquitectura basada en Objetos

Los componentes de este estilo son los objetos, o más bien instancias de los tipos de dato abstractos. En la caracterización clásica de David Garlan y Mary Shaw, los objetos representan una clase de componentes que ellos llaman managers, debido a que son responsables de preservar la integridad de su propia representación.

Los componentes del estilo se basan en los principios Orientado a Objetos: encapsulamiento, herencia y polimorfismo. Son así mismo las unidades de modelado, diseño e implementación, y los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación.

Ventajas

- Refinamiento (descomposición funcional) sencillo.
- Encapsulamiento, reutilización, fácil descomposición en una colección de agentes interactivos.

Se decidió utilizar este estilo porque cumple con el paradigma orientado a objetos, donde los componentes de un sistema encapsulan los datos y funciones, que se encargan de realizar operaciones sobre ellos, además de la comunicación entre los componentes a través de mensajes.

Arquitectura en Capas.

Específicamente el sistema que se propone contendrá 2 capas: una lógica de presentación y otra lógica de negocio. Se decidió utilizar este estilo para separar el negocio de la presentación, logrando así que el desarrollo se lleve a cabo en varios niveles, y en caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Este diseño se suele usar con gran frecuencia en la actualidad pues en dicha arquitectura a cada nivel se le confía una misión simple, lo cual permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad si las necesidades aumentan).

Las principales ventajas de la arquitectura en capas son:

- Modularidad del sistema.
- Facilita la localización de errores.
- Mejora soporte del sistema

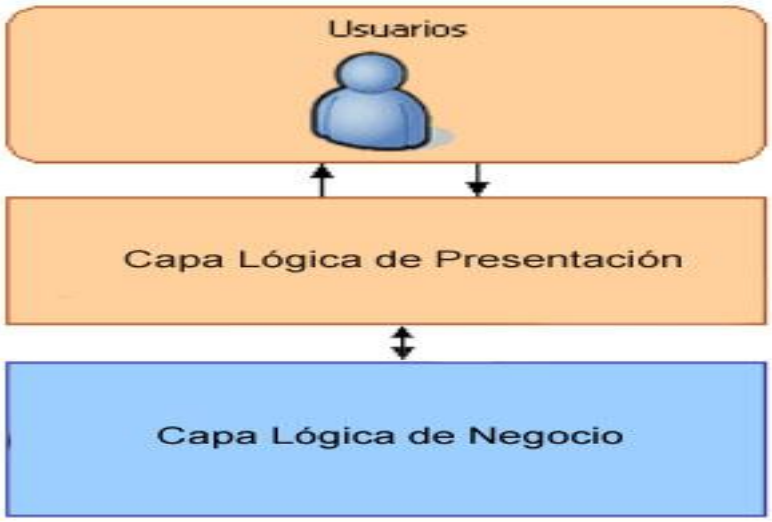


Figura 11 Estructura de la arquitectura en 2 capas

Componentes o Elementos.

Los principales componentes (elementos) que distinguen este estilo arquitectónico son las capas:

Capa Lógica de Presentación: esta capa es la responsable de todos los aspectos relacionados con la interfaz de usuario de la aplicación. Se comunica únicamente con la capa de negocio y en ella se resuelven cuestiones como:

- Navegabilidad del sistema, mapa de navegación.
- Formateo de los datos de salida: Resolución del formato más adecuado para la presentación de resultados.
- Validación de los datos de entrada, en cuanto a formatos y longitudes máximas.
- Interfaz gráfica con el usuario.

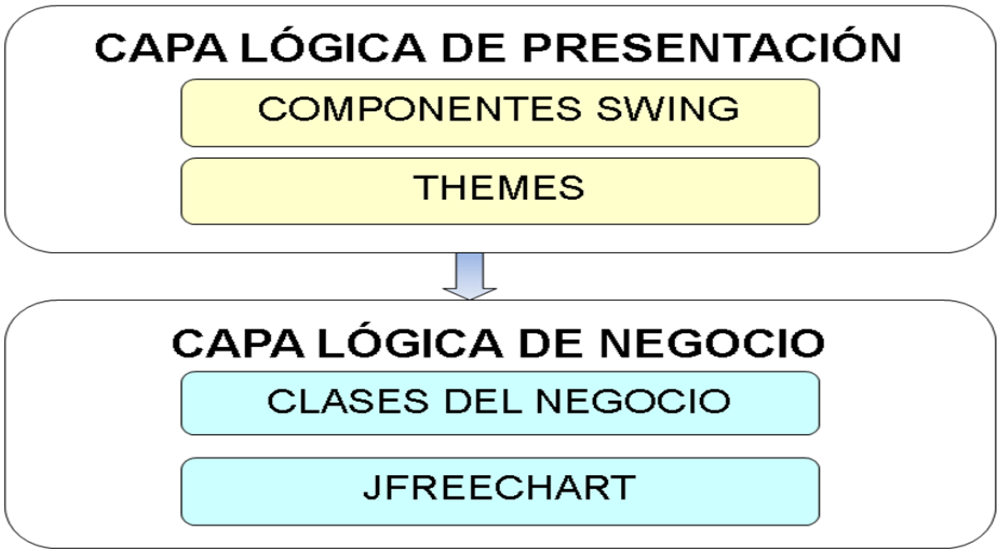
La Capa de Presentación contiene las funcionalidades necesarias para que el usuario intercambie información con la aplicación. Sus principales componentes son los desarrollados a partir del Framework Swing y un conjunto de THEMES (clases que personalizan la apariencia de cada uno de los componentes de la aplicación), que posibilitan una interfaz más amigable para la interacción con el usuario.

Lógica de Negocio: es donde se implementan todas las reglas obtenidas a partir del análisis funcional de sistema. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados a través de una clase llamada *"Fachada"*, con la cual se disminuye las dependencias entre las clases, es decir, se logra un bajo acoplamiento entre las clases de la capa lógica de presentación y lógica del negocio, de forma tal que estén lo menos ligadas posible para que en caso de que se produzca una modificación en alguna de ellas se tenga la mínima repercusión posible en el resto. Las responsabilidades que conviene abordar en esta capa son:

- Implementación de los procesos de negocio identificados en el análisis funcional del sistema.
- Control de acceso a los servicios de negocio.

- Publicación de servicios de negocio.

En ella se almacenan las clases que implementan las funcionalidades del sistema y la biblioteca de Java JFreeChart, la cual se encarga de la generación de gráficas de estadísticas.



3.1.1.2 Patrones de Diseño

Los desarrolladores con experiencia en la implementación de aplicaciones orientadas a objetos, acumulan un repertorio tanto de principios generales como de soluciones basadas en aplicar ciertos estilos que les guían en la creación de software. Estos principios y estilos, si se codifican con un formato estructurado que describa el problema y la solución, y se les da un nombre, podrían llamarse “Patrones”.

De manera más simple un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos.

Definición de un patrón de diseño

- Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.

- Un patrón de diseño identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades.

Ventajas

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifican y describen la manera de solucionar los problemas que ocurren frecuentemente en el desarrollo de software. Por tanto, están basados en la recopilación del conocimiento de los expertos, constituyen una experiencia real, probada, que funciona y nos ayuda a no cometer los mismos errores.

En la implementación del simulador propuesto se emplearán los patrones de diseño: **Singleton** (Instancia única, es un patrón creacional) porque garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia; y el patrón **Strategy** (Estrategia, es un patrón de Comportamiento) pues permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución. (39)

3.1.1.3 Framework Swing

Las JFC, (Java Foundation Classes) son un conjunto de componentes y características para ayudar a construir los entornos gráficos de los programas o GUIs (Graphical User Interfaces). Incluye prácticamente todo tipo de elementos gráficos como botones, paneles, menús y ventanas, con muchas ventajas sobre el AWT.

Swing es una parte de las JFC que permite incorporar en las aplicaciones elementos gráficos de una forma mucho más versátil y con más capacidades que utilizando el AWT básico de Java.

Algunas de las características más interesantes de Swing son:

- Amplia variedad de componentes: En general las clases que comiencen por "J" son componentes que se pueden añadir a la aplicación. Por ejemplo: JButton.

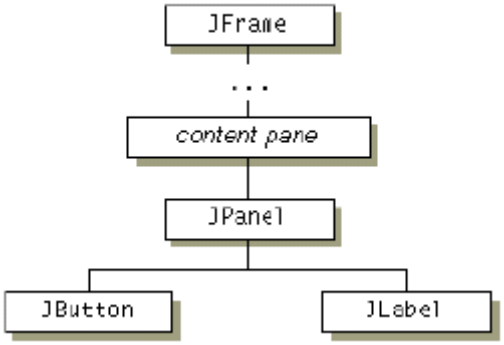


Figura 12 Ejemplo de Jerarquia de clases Swing

- Aspecto modificable (look and feel): Se puede personalizar el aspecto de las interfaces o utilizar varios aspectos que existen por defecto (Metal Max, Basic Motif, Window Win32).
- Arquitectura Modelo-Vista-Controlador: Esta arquitectura da lugar a todo un enfoque de desarrollo muy arraigado en los entornos gráficos de usuario realizados con técnicas orientadas a objetos. Cada componente tiene asociado una clase de modelo de datos y una interfaz que utiliza. Se puede crear un modelo de datos personalizado para cada componente, con sólo heredar de la clase Model.
- Gestión mejorada de la entrada del usuario: Se pueden gestionar combinaciones de teclas en un objeto KeyStroke y registrarlo como componente. El evento se activará cuando se pulse dicha combinación si está siendo utilizado el componente, la ventana en que se encuentra o algún hijo del componente.
- Objetos de acción (action objects): Estos objetos cuando están activados (enabled) controlan las acciones de varios objetos componentes de la interfaz. Son hijos de ActionListener.
- Contenedores anidados: Cualquier componente puede estar anidado en otro. Por ejemplo, un gráfico se puede anidar en una lista.
- Escritorios virtuales: Se pueden crear escritorios virtuales o "interfaz de múltiples documentos" mediante las clases JDesktopPane y JInternalFrame.

- Bordes complejos: Los componentes pueden presentar nuevos tipos de bordes. Además el usuario puede crear tipos de bordes personalizados.
- Diálogos personalizados: Se pueden crear multitud de formas de mensajes y opciones de diálogo con el usuario, mediante la clase JOptionPane.
- Clases para diálogos habituales: Se puede utilizar JFileChooser para elegir un fichero, y JColorChooser para elegir un color.
- Componentes para tablas y árboles de datos: Mediante las clases JTable y JTree.
- Potentes manipuladores de texto: Además de campos y áreas de texto, se presentan campos de sintaxis oculta JPassword, y texto con múltiples fuentes JTextPane. Además hay paquetes para utilizar ficheros en formato HTML o RTF.
- Capacidad para "deshacer": En gran variedad de situaciones se pueden deshacer las modificaciones que se realizaron.
- Soporte a la accesibilidad: Se facilita la generación de interfaces que ayuden a la accesibilidad de discapacitados, por ejemplo en Braille.

Todas las clases componentes de Swing (clases hijas de JComponent), son hijas de la clase Component de AWT. (38)

3.1.2 Diagrama de paquetes del sistema.

Los paquetes son una colección de clases, relaciones, realizaciones de casos de usos, diagramas y otros paquetes que son empleados para dividir en partes más pequeñas al modelo de diseño. Estos son utilizados para agrupar elementos del modelo de diseño que se relacionen y como herramienta organizacional.

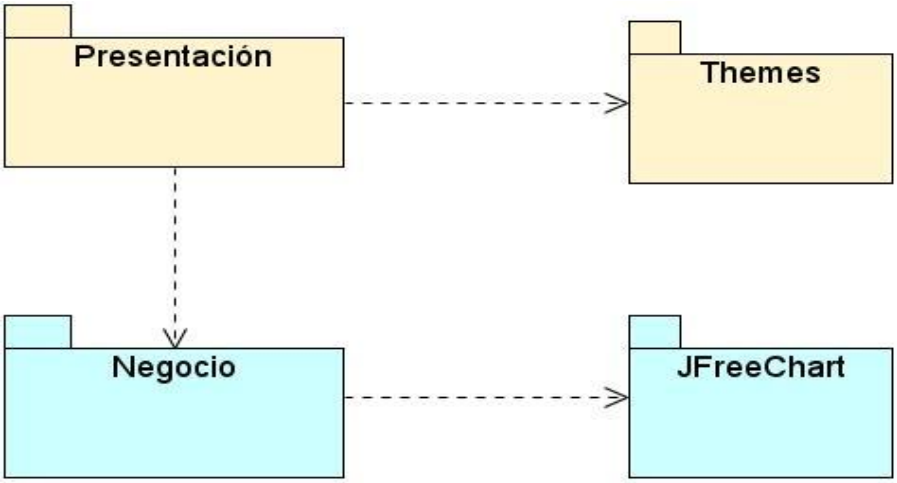


Figura 13 Diagrama de Paquetes del Diseño

- ✓ **Paquete Presentación:** contiene todas las clases y formularios pertenecientes a la capa de Presentación.
- ✓ **Paquete Themes:** brinda servicios a la capa de Presentación. Contiene todas las clases que personalizan la apariencia de los componentes de la aplicación.
- ✓ **Paquete Negocio:** contiene todas las clases que implementan las funcionalidades del sistema
- ✓ **Paquete JFreeShart:** JFreeChart es una biblioteca gratis de clase Java para la generación de gráficas de estadísticas (GNU LGPL). Esta biblioteca está contenida en el paquete que tiene su mismo nombre, que brinda servicios a la capa de Negocio para la generación de gráficas dinámicamente.

3.1.3 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además el modelo de diseño sirve de abstracción para la implementación y es, de ese modo, utilizado como una entrada fundamental de las actividades de implementación. (23)

El propósito del modelo de diseño es lograr una abstracción de la implementación del sistema. Es usado para concebir un documento del diseño del sistema de software. Es abarcador, compuesto por artefactos que engloban todas las clases del diseño, subsistemas, paquetes, colaboraciones, y las relaciones entre ellos.

3.1.3.1 Realización de caso de uso-diseño

Una realización de caso de uso-diseño es una colaboración en el modelo de diseño que describe cómo se realiza un caso de uso específico, y como se ejecuta, en términos de clases de diseño y sus objetos. Una realización de caso de uso-diseño tiene una descripción de flujo de eventos textual, diagramas de clases que muestran sus clases de diseño participantes, y diagramas de interacción que muestran las realizaciones de un flujo o escenario concreto de un caso de uso en términos de interacción entre objetos del diseño.

Diagrama de clases del diseño

En este diagrama se representan las clases participantes en las realizaciones de caso de uso, subsistemas y sus relaciones. También puede darse el caso de algunas operaciones, atributos y asociaciones sobre una clase específica que son relevantes.

Clases del diseño

Una clase del diseño es una abstracción sin costuras de una clase o construcción similar en la implementación del sistema. Esta abstracción es sin costuras en el siguiente sentido:

- El lenguaje utilizado para especificar una clase de diseño es lo mismo que el lenguaje de programación. Consecuentemente las operaciones, parámetros, atributos, tipos y demás son especificados utilizando la sintaxis del lenguaje de programación elegido.
- La visibilidad de los atributos y las operaciones de una clase de diseño se especifica con frecuencia.
- Las relaciones de aquellas clases de diseño implicadas con otras clases, a menudo tienen un significado directo cuando la clase es implementada.
- Los métodos tienen correspondencia directa con el correspondiente método en la implementación de las clases.

Diagrama de clases del diseño por casos de uso:

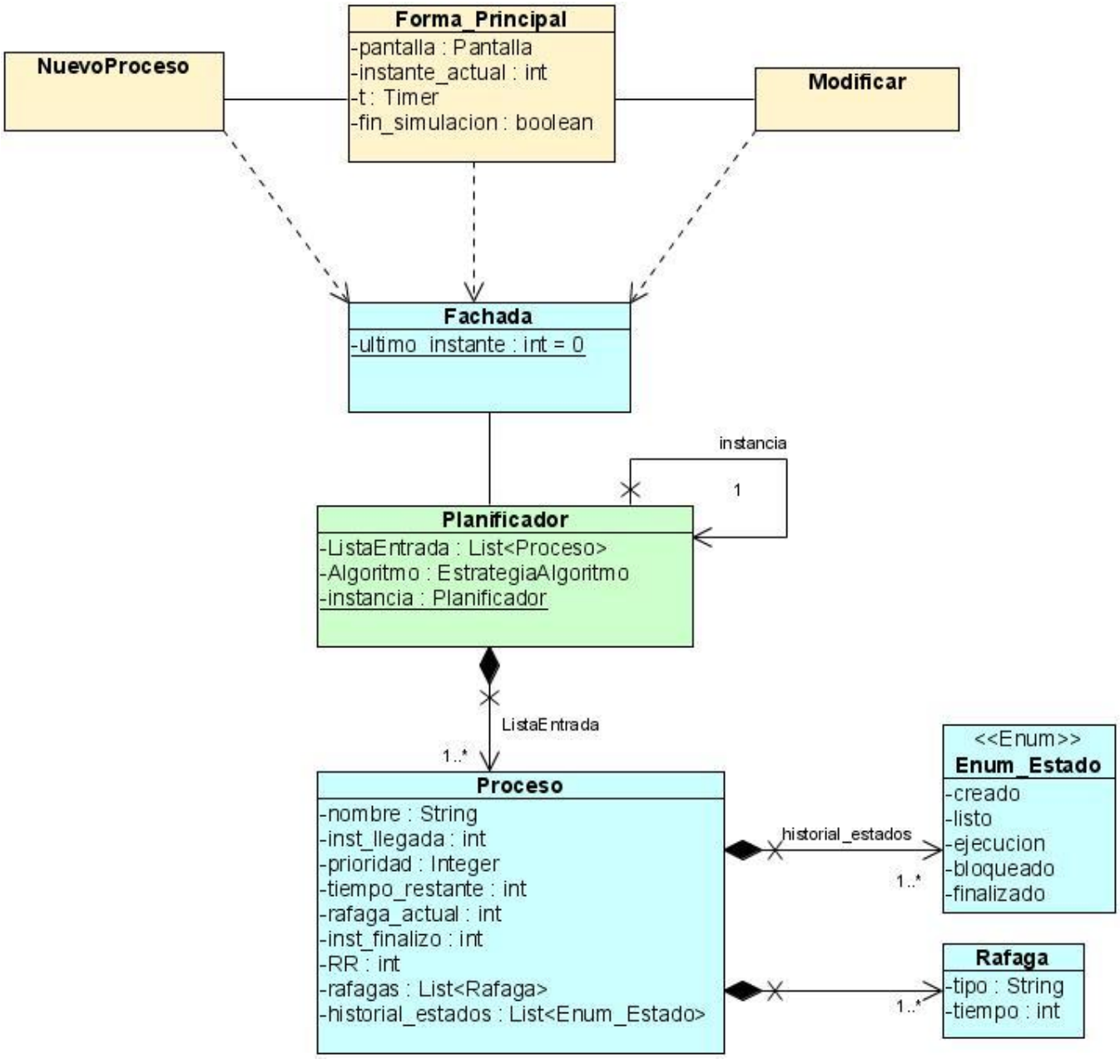


Figura 14 Diagrama de clases del diseño CU Gestionar Proceso

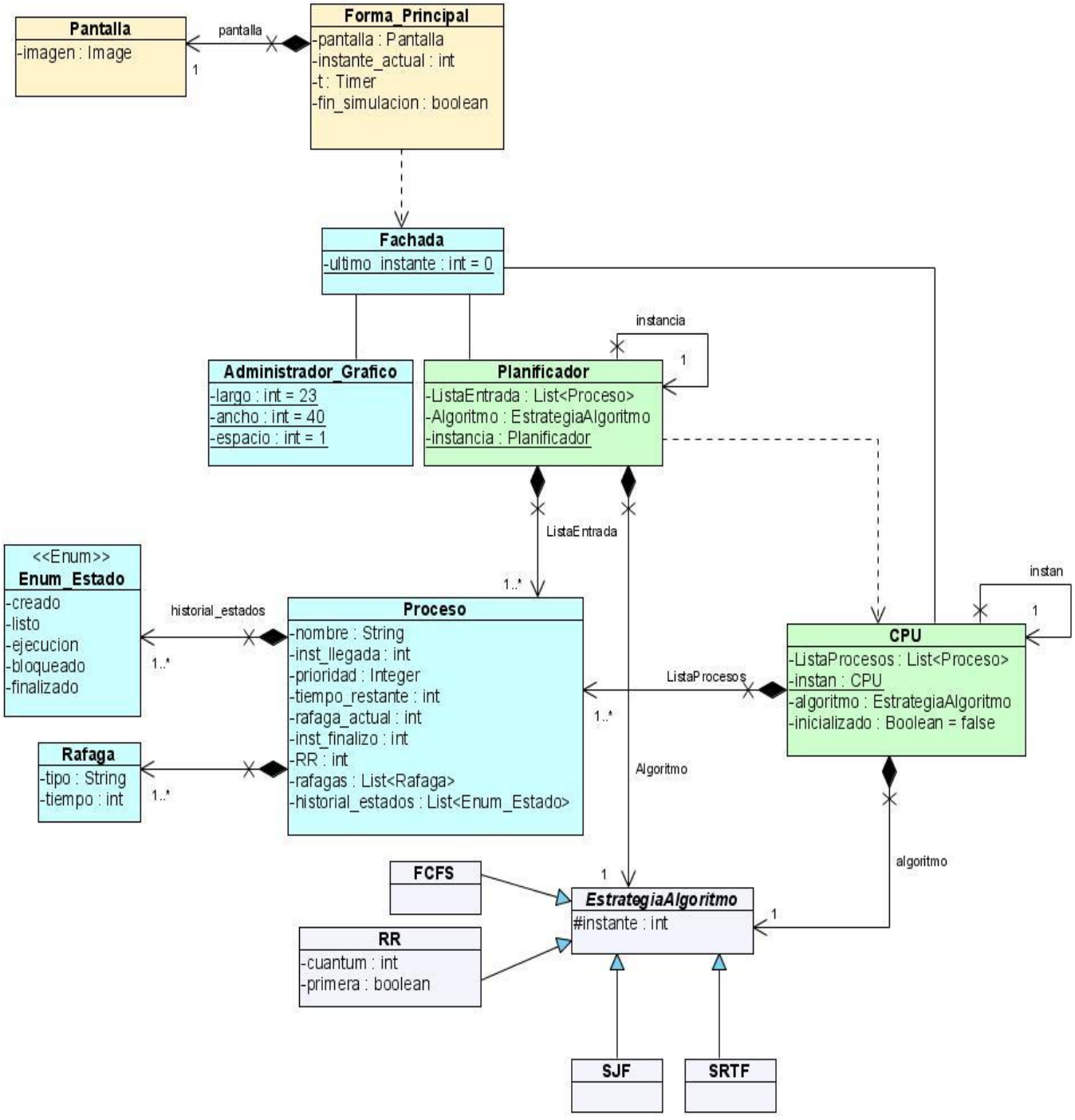


Figura 15 Diagrama de clases del diseño CU Simular Algoritmo

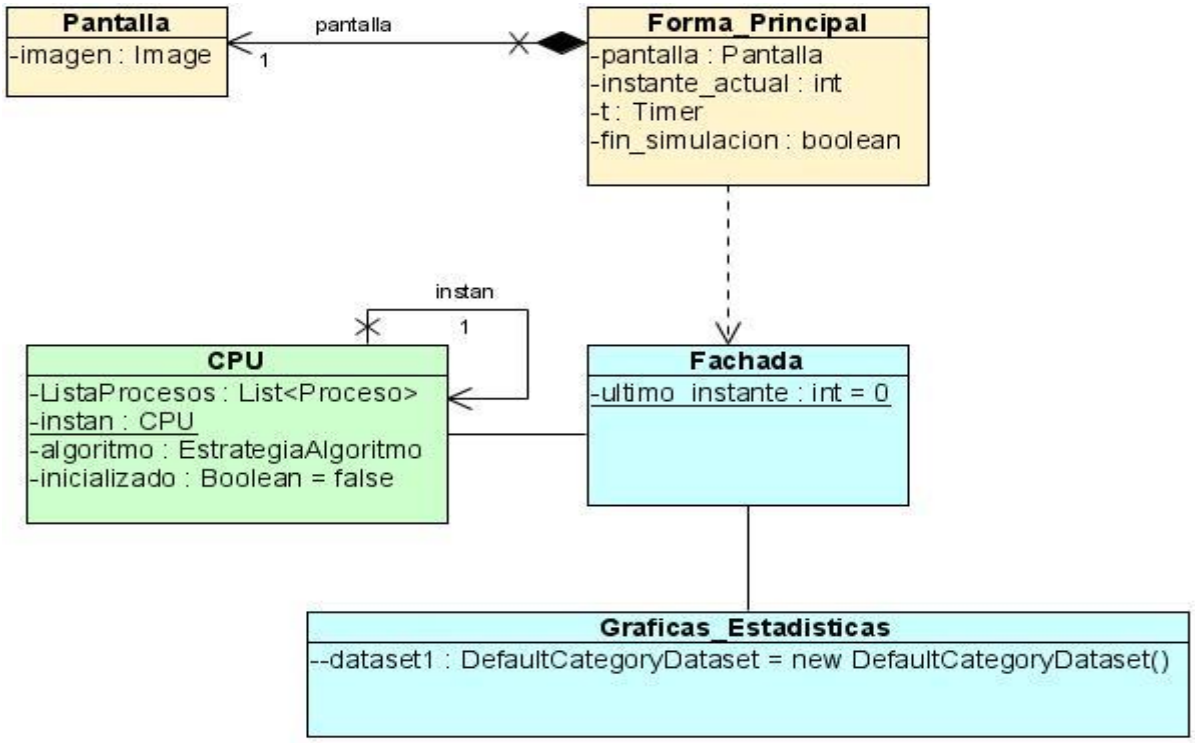


Figura 16 Diagrama de clases del diseño de CU Gestionar_Estadísticas

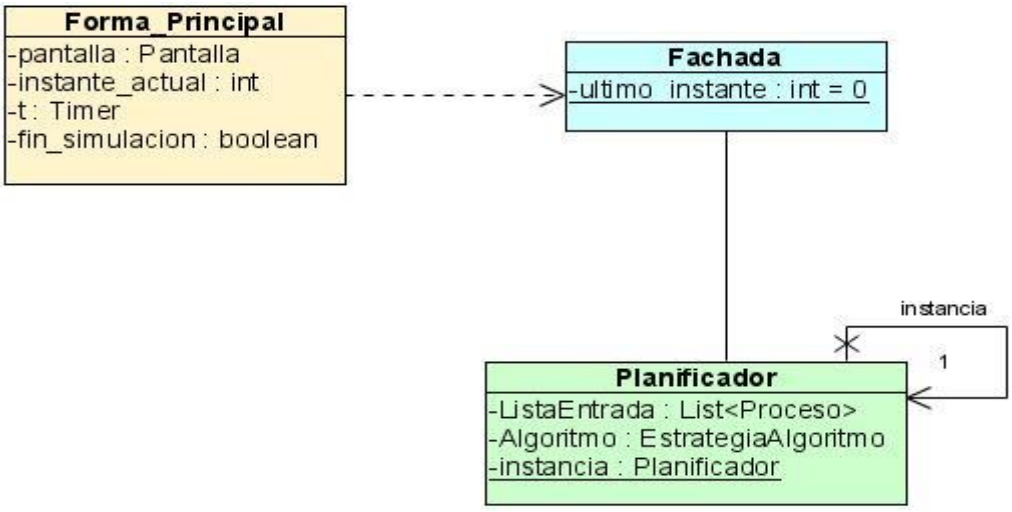


Figura 17 Diagrama de clases del diseño CU Gestionar_Datos_Procesos

Descripción de las clases del diseño más significativas.

(Ver Anexo 1)

3.1.3.2 Diagramas de Interacción

Los diagramas de secuencia y los diagramas de colaboración (ambos llamados diagramas de interacción) son dos de los cinco tipos de diagramas de UML que se utilizan para modelar los aspectos dinámicos de los sistemas. Un diagrama de interacción muestra una interacción, que consiste en un conjunto de objetos y relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Los diagramas de colaboración y secuencia están basados en la misma información, pero cada uno enfatizando un aspecto particular. Son isomorfos, es decir, se puede convertir de uno a otro sin pérdida de información. Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes; un diagrama de colaboración es un diagrama de interacción que destaca la organización estructural de los objetos que envían y reciben mensajes.

Figura 18 Diagramas de Interacción

Diagramas de Secuencia.

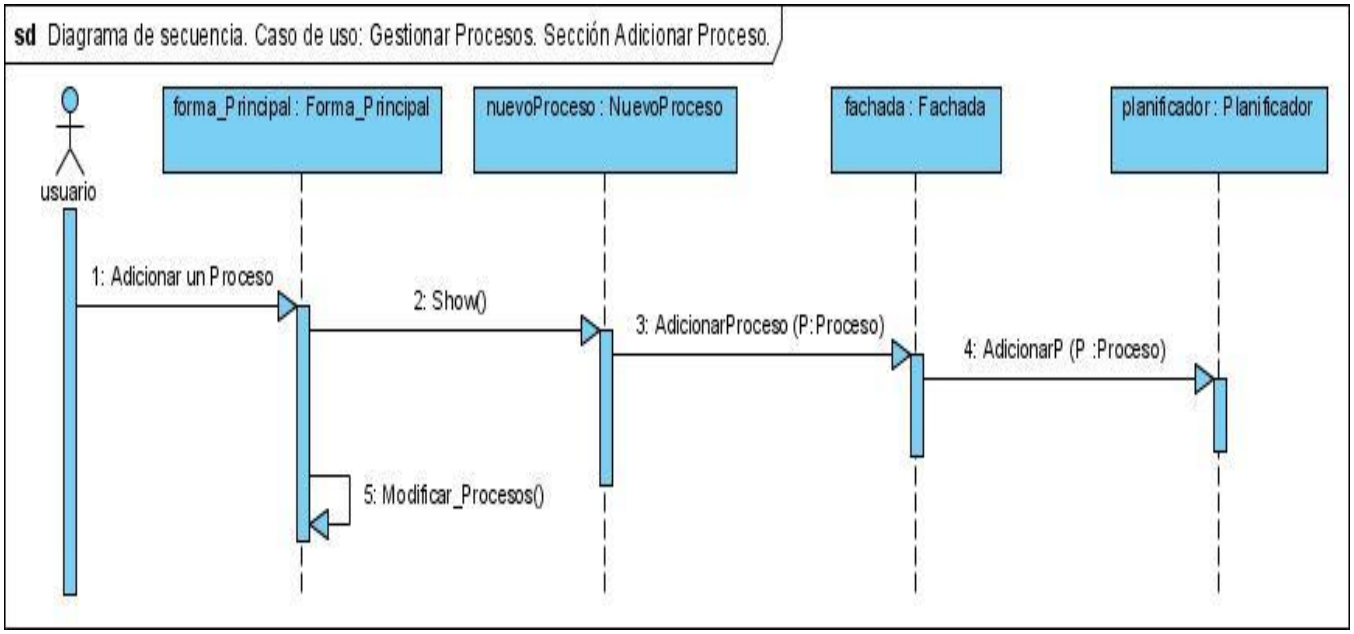


Figura 19 Diagrama de Secuencia CU Gestionar Proceso. Sección Adicionar Proceso

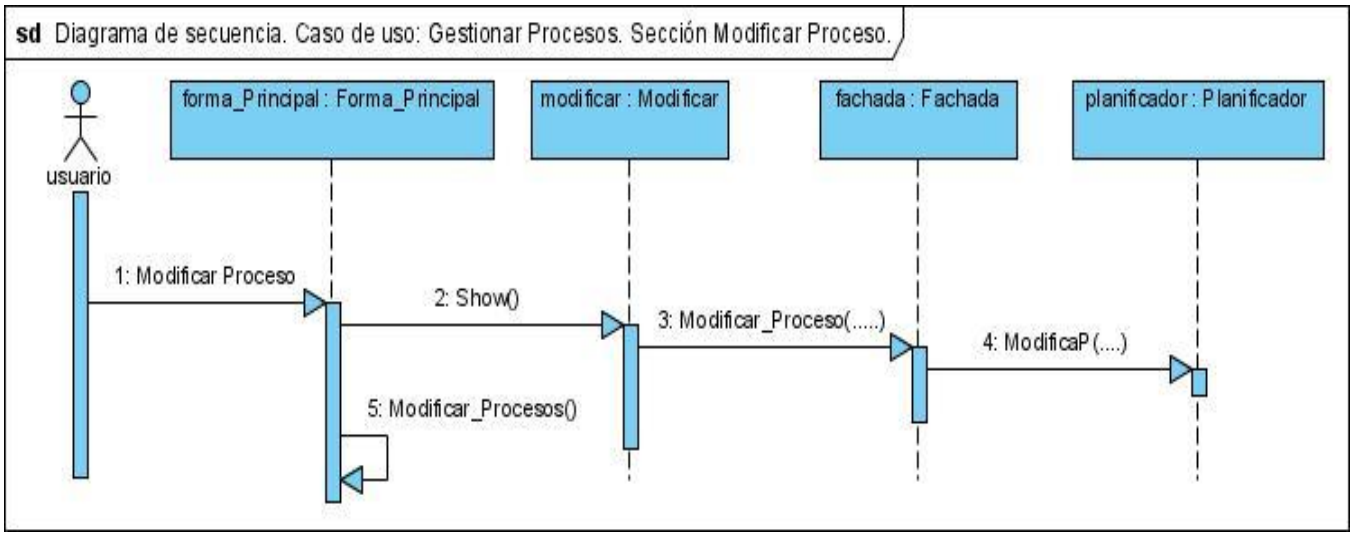


Figura 20 Diagrama de secuencia CU Gestionar Proceso. Sección Modificar Proceso

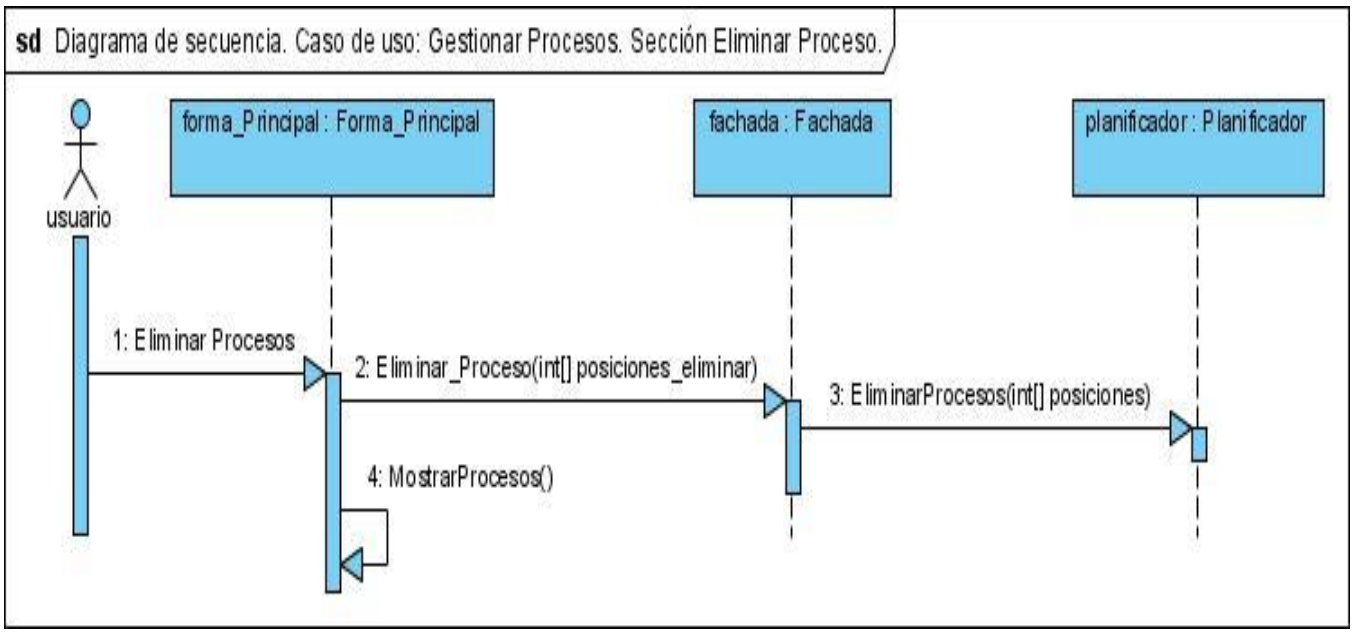


Figura 21 Diagrama de secuencia CU Gestionar Proceso. Sección Eliminar Proceso

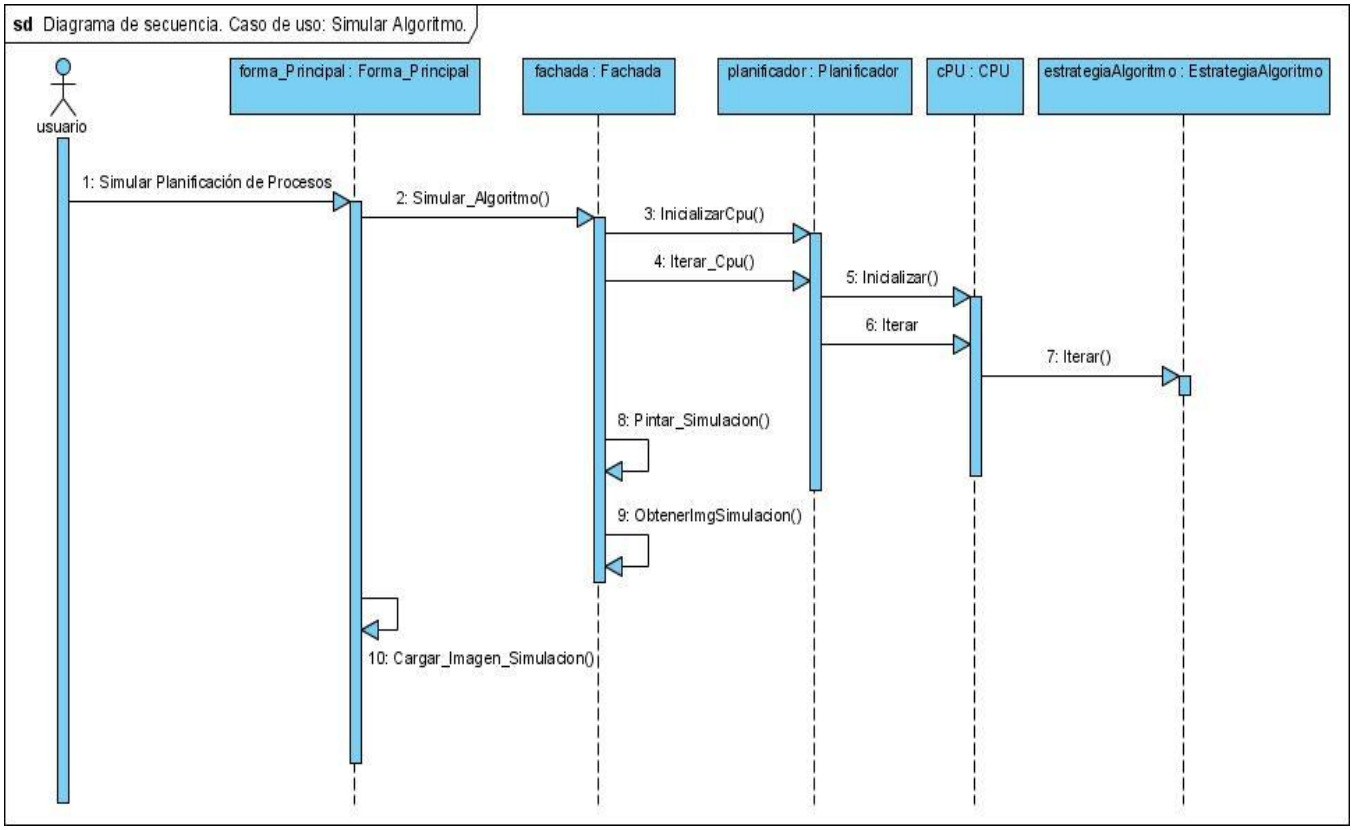


Figura 22 Diagrama de secuencia CU Simular Algoritmo

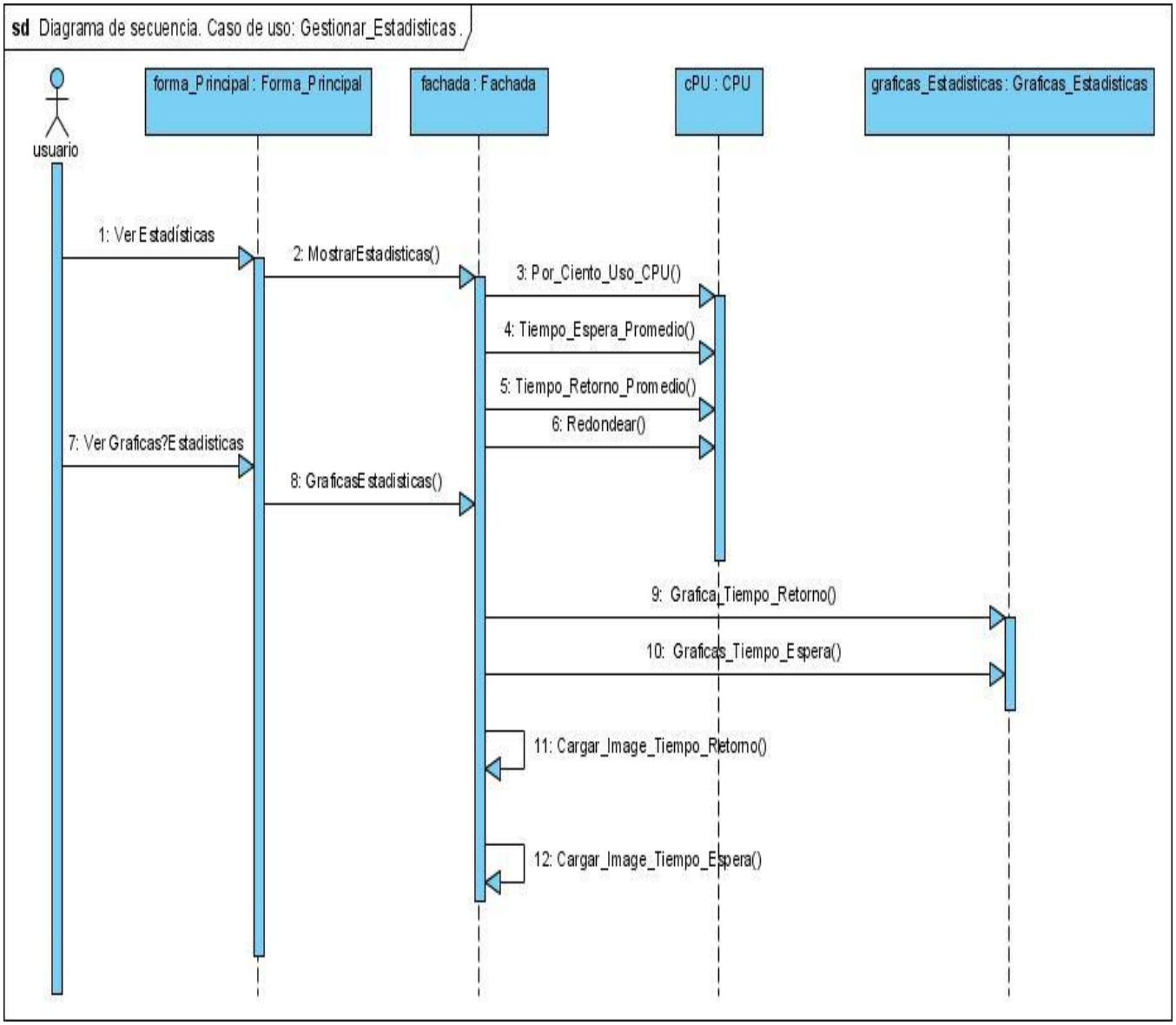


Figura 23 Diagrama de Secuencia CU Gestionar_Estadísticas

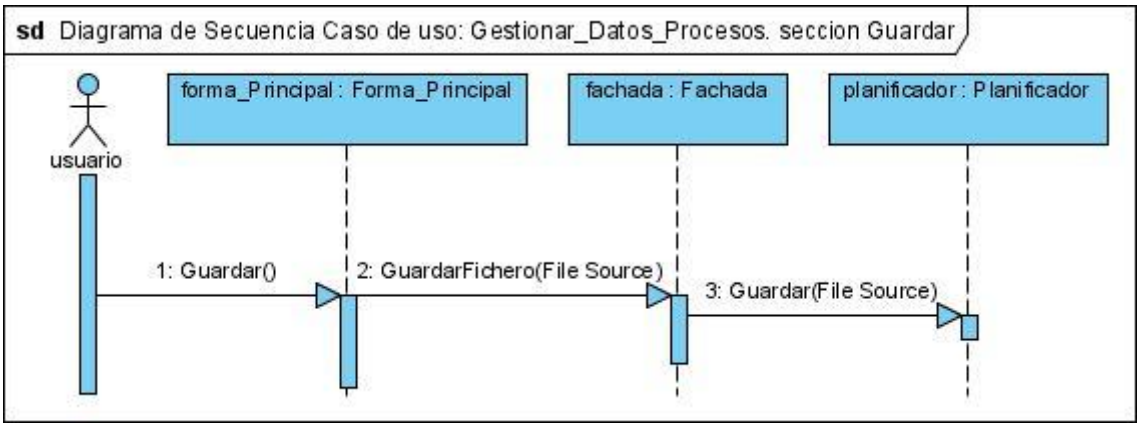


Figura 24 Diagrama de Secuencia. CU Gestionar_Datos_Proceso. Sección Salvar Datos para un fichero.

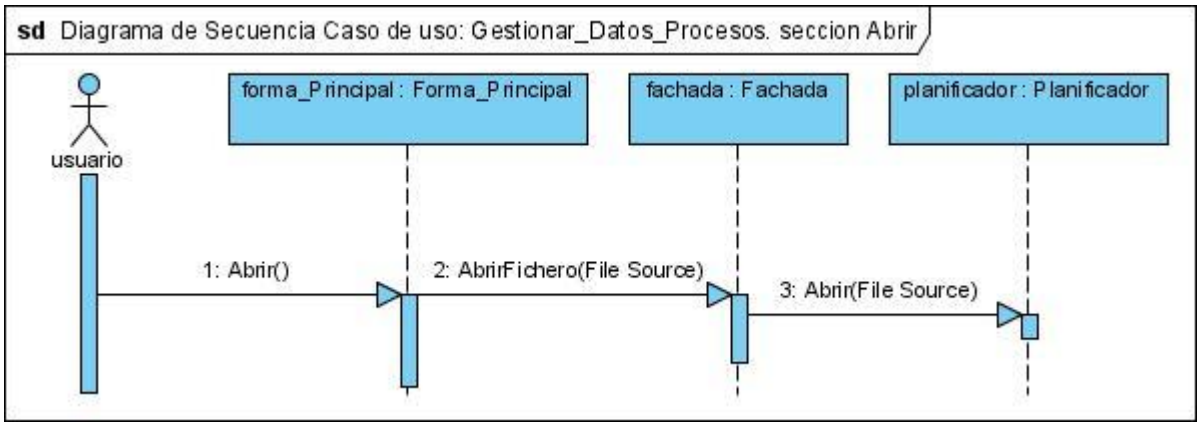


Figura 25 Diagrama de Secuencia. CU Gestionar_Datos_Proceso. Sección Cargar datos de Fichero.

3.1.4 Diagrama de Despliegue.

Los Diagramas de Despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Para aplicaciones como la que se propone, que se ejecuta en una sola máquina y todos los dispositivos con que se relaciona son los estándares (teclado, mouse), el diagrama de despliegue va a estar constituido por un nodo, por lo que no se considera relevante incluirlo en la investigación.

Conclusiones

En este capítulo, correspondiente a la etapa de diseño se definió una arquitectura flexible y consistente basada en la utilización de patrones, se desarrollaron los diagramas de clases organizados por casos de uso, el de despliegue y los de secuencia de los casos de usos más significativos.

CAPÍTULO 4 IMPLEMENTACIÓN Y PRUEBA

Este capítulo comienza con el resultado del diseño, implementando el sistema en términos de componentes. Además se hace una revisión final de las especificaciones del diseño y de la codificación mediante la realización de pruebas que garanticen la calidad del software.

4.1 Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros.

4.1.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas.

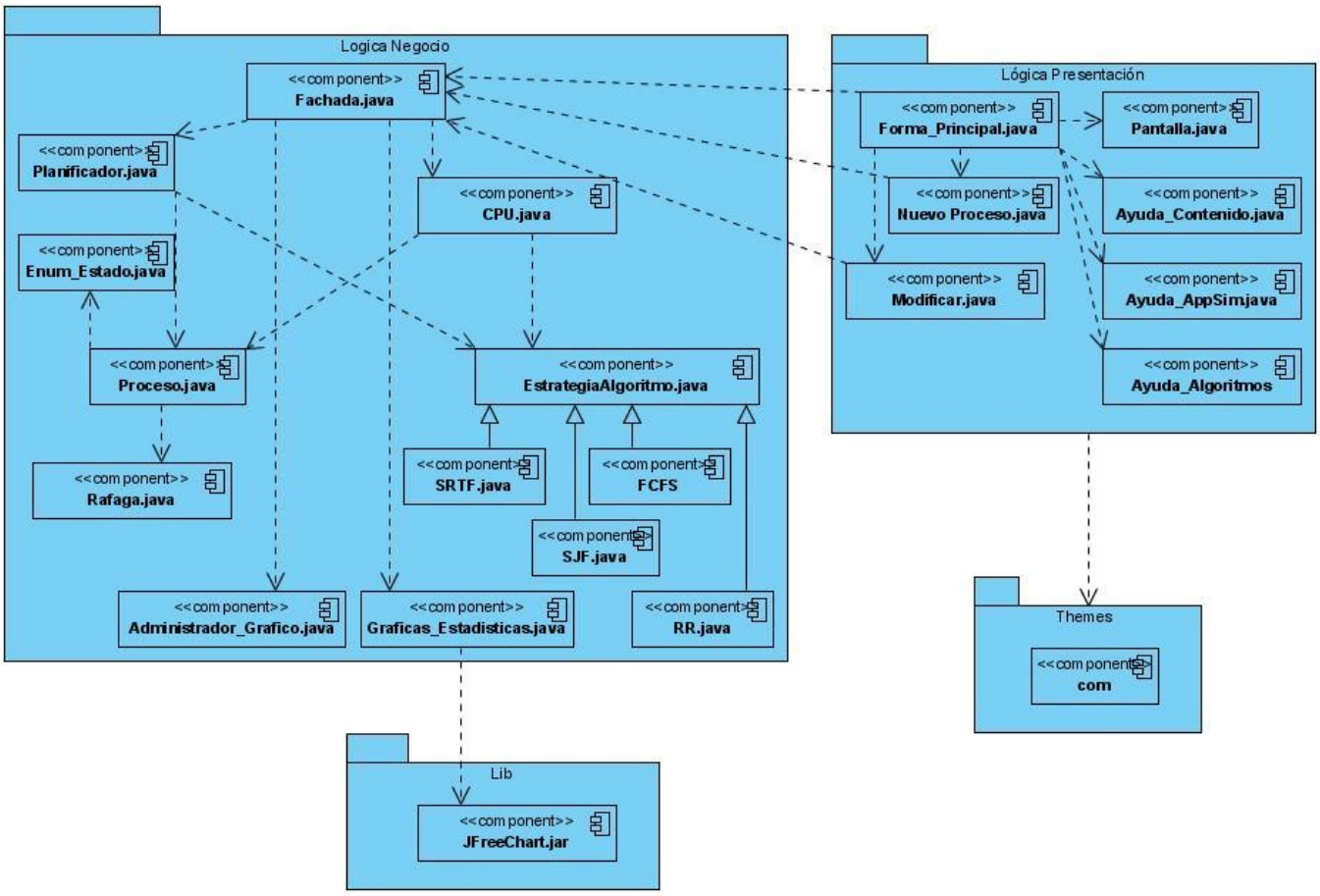


Figura 26 Diagrama de Componentes

4.2 Prueba

La IEEE [IEEE90] define las pruebas como una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente.

Cuando se habla de condiciones específicas en la definición anterior, se puede suponer la presencia de una especie de ambiente de operación de la prueba, para el cual deben existir determinados valores para las entradas y las salidas, así como también ciertas condiciones que delimitan a dicho ambiente de operación. Formalmente esto es conocido como caso de prueba.

De acuerdo a la IEEE [IEEE90] un **caso de prueba** es un conjunto de entradas, condiciones de ejecución y resultados esperados diseñados para un objetivo particular.

La prueba de software es un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones del diseño y de la codificación.

Desde hace ya algunos años, han surgido y evolucionado una variedad de métodos para realizar pruebas de software. Las alternativas más significativas en este contexto son las pruebas de caja blanca y las pruebas de caja negra; las primeras, pruebas orientadas a la estructura y las segundas al comportamiento del software. Al sistema propuesto se le realizaron pruebas de caja negra.

4.2.1 Prueba de Caja Negra

Las de pruebas de caja negra, también denominadas de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de dato externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

A diferencia de las pruebas de caja blanca, que se basan en la lógica interna del software, las pruebas de caja negra se concentran en su funcionalidad, por lo que mucho del trabajo se realiza interactuando con la interfaz del software. Los casos de prueba generados en este enfoque, se diseñan a partir de valores entrada y salida. De esta forma, se puede determinar la validez de una salida para un conjunto de entradas proporcionadas.

Pruebas de caja negra efectuadas al sistema propuesto:

Tabla 8 Casos de Prueba del CU Gestionar Proceso

Nombre del caso de uso:	Gestionar Proceso	
Nombre del caso de prueba:	Adicionar Proceso	
Entrada	**Resultados**	**Condiciones**
Se seleccionó la opción Nuevo en el menú del formulario principal.	- La aplicación mostró una interfaz para adicionar un nuevo proceso. - Se llenaron todos los campos, se presionó el botón Adicionar y se adicionó el proceso en el sistema.	- Los datos introducidos deben ser válidos. -No se pueden adicionar más de 20 procesos. - Son como máximo 5 ráfagas de CPU. - No se pueden dejar campos vacios.
Nombre del caso de prueba:	Modificar Proceso	
En la tabla que aparece en el formulario principal con todos los procesos, se seleccionó uno de ellos y se presionó el botón Modificar.	La aplicación mostró una interfaz con todos los datos del proceso seleccionado. Dando la posibilidad de modificarlos. -Se modificaron los datos, se presionó el botón Modificar y el proceso apareció actualizado en la tabla de procesos de la interfaz principal.	-Debe haber un proceso seleccionado. -Los datos que se modifiquen deben ser válidos. -No se pueden dejar campos vacios.
Nombre del caso de prueba:	Eliminar Proceso	
En la tabla que aparece en el formulario principal con todos los	- Se mostró un mensaje para confirmar la	Deben haber procesos en el sistema.

procesos, se seleccionaron todos los procesos que se querían eliminar y se presionó el botón Eliminar.	eliminación de los procesos seleccionados. - Se confirmó la acción de eliminar, y de inmediato el sistema eliminó los procesos, mostrándose la tabla del formulario principal sin esos procesos.	-Debe haber al menos un proceso seleccionado.

Tabla 9 Casos de Prueba del CU Simular Algoritmo

Nombre del caso de uso:	Simular Algoritmo	
Nombre del caso de prueba:	Simular Automáticamente.	
Entrada	**Resultados**	**Condiciones**
Se configuró la velocidad de la simulación y se presionó el botón simular en el formulario principal.	La aplicación mostró la simulación de la planificación de los procesos existentes en el sistema según el algoritmo seleccionado y de acuerdo a la velocidad especificada.	-Deben haber procesos en el sistema. -Debe haber un algoritmo seleccionado.
Nombre del caso de prueba:	Simular Instante a Instante.	
Se presionó el botón de simulación instante a instante en el formulario principal.	La aplicación mostró la simulación de la planificación de los procesos existentes en el sistema según el algoritmo seleccionado a	-Deben haber procesos en el sistema. -Debe haber un algoritmo seleccionado.

	medida que se fue presionando el botón.	

Tabla 10 Caso de Prueba del CU Gestionar_Estadísticas

Nombre del caso de uso:	Gestionar_Estadísticas	
Nombre del caso de prueba	Gestionar_Estadísticas	
Entrada	**Resultados**	**Condiciones**
Se seleccionó la pestaña de estadística después de haber terminado una simulación.	Se mostraron todos los valores estadísticos de la simulación. Incluyendo la posibilidad de ver sus respectivas gráficas.	

Tabla 11 Caso de Prueba del CU Gestionar_Datos_Procesos

Nombre del caso de uso:	Gestionar_Datos_Procesos	
Nombre del caso de prueba:	Abrir Fichero	
Entrada	**Resultados**	**Condiciones**
Se seleccionó en el menú del formulario principal la opción “Abrir”.	- Se mostró una interfaz, en la que es buscó el fichero que se quería abrir. - Se seleccionó el fichero y se presionó el botón Abrir. - Se mostraron los datos del fichero en la interfaz principal.	El fichero debe ser válido (debe tener extensión .app).

Nombre del caso de prueba:	Cargar Fichero.	
Se seleccionó en el menú del formulario principal la opción “Guardar”.	- Se mostró una interfaz, en la que se buscó el lugar donde se quería guardar los datos que se encontraban en el sistema. - Se presionó el botón guardar y se guardaron los datos en el directorio especificado satisfactoriamente.	-Deben haber procesos en el sistema.

Tabla 12 Caso de Prueba del CU Mostrar Ayuda

Nombre del caso de uso:	Mostrar_Ayuda	
Nombre del caso de prueba	Mostrar_Ayuda	
Entrada	**Resultados**	**Condiciones**
Se seleccionó en el menú del formulario principal la opción de “Ayuda”.	-Se desplegó un submenú con una serie de temas de ayuda. -Se seleccionó el tema del que se requería información. -Se mostró la información solicitada.	

Conclusiones

En este capítulo se mostró el diagrama de componentes para estructurar el modelo de implementación en términos de subsistemas de implementación, mostrando así las relaciones entre los elementos de implementación. Además se llevó a cabo la etapa de prueba al sistema para evaluar la calidad del mismo.

CONCLUSIONES GENERALES

A partir de la investigación realizada para la elaboración de esta aplicación se arriba a las siguientes conclusiones:

1. Se realizó un estudio previo que posibilitó la selección de la metodología, el lenguaje de programación y el entorno de desarrollo a utilizar en la elaboración del software.

2. Se diseñó e implementó una aplicación multiplataforma para la representación del proceso de planificación de la CPU de acuerdo a algunas de las políticas clásicas, posibilitando la reducción del tiempo empleado en la enseñanza y aprendizaje del contenido planificación de procesos en la asignatura de Sistemas Operativos de la UCI.

3. Los resultados obtenidos de las pruebas realizadas, tomando como juegos de datos ejercicios pertenecientes a las clases prácticas de la asignatura, correspondieron con los esperados, demostrando de esta manera la concordancia entre los requerimientos exigidos por el cliente y la solución automatizada.

RECOMENDACIONES

Para incrementar las prestaciones de la solución propuesta se recomienda:

- ✓ Incorporar la planificación de procesos por colas multinivel (MLQ: Multiple level queues) y la variante de planificación por prioridades.
- ✓ Integrarla a otros simuladores que respondan a los temas de la asignatura Sistemas Operativos: Planificación de Memoria y Planificación de disco duro.
- ✓ Mostrar de manera dinámica las modificaciones de las tablas de procesos.
- ✓ Incorporar la funcionalidad Imprimir.

BIBLIOGRAFÍA

1. Cano Alonso, Francisco A. www.informaticahabana.com. [Online] Diciembre 15, 2007. [Cited: marzo 2008.] www.informaticahabana.com/evento_virtual/files/EDU103.doc.

2. Lisa, García Cabrera, Martinez del Rio, Francisco and Redondo duque, M A. Sistemas OPerativos. 1999.

3. Marqués, Pere. [Online] abril 5, 2002. [Cited: 2 marzo, 2008.] http://www.lmi.ub.es/te/any96/marques_software/#capitol2.

4. Montoya, Jaime. [Online] julio 24, 2007. [Cited: febrero 8, 2008.] www.jaimemontoya.com.

5. Tanenbaum, Andrew S. and Woodhull, Albert S. Sistemas Operativos. Diseño e Implementación. [ed.] Pablo Eduardo Roig Vázquez. Segunda Edición.

6. Vaquero Sánchez, Antonio. [Online] julio 25, 2006. [Cited: marzo 25, 2008.] http://www.educa.aragob.es/cursoryc/word2/soluciones/vaquero.doc..

7. Planificación de Procesos. Cuba : UCI, 2008.

8. Al Punto. Ciencia y Tecnología. [Online] abril 5, 2007. [Cited: marzo 8, 2008.]

9. Microsoft. [Online] 2008. [Cited: febrero 2, 2008.] http://www.microsoft.com.

10. msdn. [Online] 2008. [Cited: febrero 5, 2008.] http://msdn.microsoft.com/es-es/default.aspx.

11. Mono. [Online] 2008. [Cited: febrero 5, 2008.] http://www.mono-project.com.

12. Programación en castellano. [Online] 2007. [Cited: febrero 5, 2008.] http://www.programacion.com/tutorial/csharp/3/.

13. NetBeans. [Online] 2008. [Cited: febrero 5, 2008.] http://www.netbeans.org/index_es.html.

14. [Online] 2007. [Cited: marzo 2, 2008.] http://www.uv.es/~jgutierr/MySQL_Java/TutorialEclipse.pdf.

15. [Online] 2008. [Cited: febrero 5, 2008.] http://www.java.com/es/about/.

16. Cáceres, Paloma and Marcos, Esperanza. [Online] 2006. [Cited: marzo 2, 2008.] http://www.dlsi.ua.es/webe01/articulos/s112.pdf.

17. Mendoza Sánchez, María A. informatizable. [Online] junio 7, 2004. [Cited: febrero 11, 2008.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html.

18. Informatica. Herramientas CASE. 1999.

19. Free Download Manager. [Online] marzo 5, 2007. [Cited: febrero 10, 2008.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

20. Visual Paradigm. [Online] 2008. [Cited: febrero 2, 2008.] http://www.visualparadigm.com.

21. Vizcaíno, Aurora, García, Felix Oscar and Caballero, Ismael. [Online] 2006. [Cited: marzo 1, 2008.] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/LabTr1_VP.pdf.

22. Slideshare. [Online] abril 6, 2007. [Cited: marzo 5, 2008.] http://www.slideshare.net/vivi_jocadi/rational-rose/.

23. Jacobson, Ivar, Booch, Grady and Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. Cuba : Felix Varela, 2004.

24. Larman, C. UML y Patrones. . 1999.

25. [Online] junio 5, 2006. [Cited: enero 29, 2008.] http://eupt2.unizar.es/asignaturas/itig/sistemas_operativos_1/PFC.htm.

26. Algoritmo de Planificación Por Turno Rotatorio o Round Robin. [Online] junio 2, 2005. [Cited: enero 29, 2008.] http://wwwdi.ujaen.es/~lina/TemasSO/PLANIFICACIONDEPROCESOS/PlanificadorProcesos/alg_planif_RR.html.

27. Hernández León, Rolando Alfredo and Coello González, Sayda. EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA. Ciudad de La Habana : s.n., 2002.

28. Buschmann, Frank. Pattern-Oriented SoftwareArchitecture: a system of patterns. 2004.

29. Pattern-Oriented Software Architecture. s.l. : Jhon Wiley & Sons., 1996.

30. Reynoso, Carlos Billy. Introducción a la Arquitectura de Sofware. marzo 2004.

31. Shaw, Mary y Garlan, David. An Introduction to Software Arcuitecture. 1994.

32. Bullet, S. (1987). "Essence and Accident in Software Engineering."

33. Pressman, Roger S. Ingenieria del Software. Un enfoque práctico. La Habana : Felix Varela, 2005.

34. [En línea] 20 de mayo de 2007. [Citado el: 5 de enero de 2008.] http://www.educando.edu.do

35. Canós, José H., Letelier, Patricio y Penadés, Maria Carmen. [En línea] 24 de enero de 2006. [Citado el: 25 de marzo de 2008.] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

36. sparxsystems. *sparxsystems.* [En línea] 5 de junio de 2007. [Citado el: 2 de marzo de 2008.] http://www.sparxsystems.com.ar/products/.

37. [En línea] 31 de agosto de 2007. [Citado el: 29 de enero de 2008.] http://www.lcc.uma.es/pfc/71.pdf.

38. Gómez, S. P. (2006) Swing Univ. Politécnica de Madrid

39. *Arquitectura y Patrones de Diseño.* Habana : UCI, 2008.

ANEXOS

Anexo 1: Descripción de las clases más importantes del diseño.

Tabla 13 Descripción de la clase Fachada

Nombre: Fachada	
Tipo de clase:	
Atributo	**Tipo**
ultimo_instante	int
Para cada responsabilidad:	
Nombre:	AdicionarProceso(P: Proceso)
Descripción:	Llama al método AdicionarP(P:Proceso) de la clase Planificador.
Nombre:	Modificar_Proceso(String P, int llegada, int prior, List<Rafaga> R)
Descripción:	Llama al método ModificarP(String name, int llegada, int priori, List<Rafaga> raf) de la clase Planificador().
Nombre:	Eliminar_Proceso(int[] posiciones_eliminar)
Descripción:	Llama al método EliminarProcesos(int[] posiciones_eliminar) de la clase Panificador.
Nombre:	SimularAlgoritmo()
Descripción:	Llama a los métodos InicializarCpu() e IterarCpu de la clase Planificador().
Nombre:	PintarSimulacion()
Descripción:	Crea una imagen (BufferedImage), obtiene el Graphics de esa imagen, y sobre este pinta la planificación de los procesos que existen en el sistema haciendo uso de los métodos: Pintar_ListaProceso, Pintar_Eje_horizontal, Pintar_Eje_Vertical, Pintar_Instantes, Pintar_Nombres_Procesos, Pintar_String_Vertical,

	Pintar_String_Horizontal, de la clase Administrador_Grafico, y por último almacena la imagen en la carpeta de la aplicación.
Nombre:	ObtenerImgSimulacion()
Descripción:	Devuelve la imagen donde se guardó la simulación.
Nombre:	MostrarEstadisticas()
Descripción:	Llama a los métodos que calculan las estadísticas en la clase Clase CPU.
Nombre:	GraficaEstadisticas()
Descripción:	En dependencia del tipo de grafica que se solicite llama al método que devuelve dicha grafica de la clase Gráfica_Estadística()
Nombre:	cargarImagen_Grafica()
Descripción:	Carga la imagen que pertenece a la gráfica de estadística solicitada.

Tabla 14 Descripción de la clase Planificador

Nombre: Planificador	
Tipo de clase: Controladora	
Atributo	**Tipo**
ListaEntrada	List<Proceso>
Algoritmo	EstrategiaAlgoritmo
instancia	Planificador
Para cada responsabilidad:	
Nombre:	AdicionarP(P:Proceso)
Descripción:	Adiciona el proceso especificado como parámetro en ListaEntrada.
Nombre:	ModificarP(String name, int llegada, int priori, List<Rafaga> raf)

Descripción:	Modifica el proceso especificado como parámetro en ListaEntrada.
Nombre:	EliminarProcesos(int[] posiciones_eliminar)
Descripción:	Elimina todas las posiciones especificadas como parámetro ListaEntrada.
Nombre:	InicializarCpu()
Descripción:	Llama al método inicializar de la clase CPU.
Nombre:	IterarCpu()
Descripción:	Llama al método iterar de la clase CPU.
Nombre:	OrdenarLista()
Descripción:	Ordena ListaEntrada por orden de llegada y en caso de empate decide según la prioridad.
Nombre:	Instancia()
Descripción:	Implementa el patrón Singleton.

Tabla 15 Descripción de la clase CPU

Nombre: CPU	
Tipo de clase: Controladora	
Atributo	**Tipo**
ListaProcesos	List<Proceso>
algoritmo	EstrategiaAlgoritmo
instan	CPU
iunicializado	Boolean
Para cada responsabilidad:	
Nombre:	Inicializar(List<Proceso> lista, EstrategiaAlgoritmo algoritmo)
Descripción:	Inicializa ListaProcesos y algoritmo con los datos entrados por parámetro.

Nombre:	Iterar()
Descripción:	Llama al método Iterar() de la clase EstratediaAlgoritmo.
Nombre:	Tiempo_Espera_Promedio()
Descripción:	Calcula el tiempo de espera medio de la simulación efectuada.
Nombre:	Tiempo_Retorno_Promedio()
Descripción:	Calcula el tiempo de retorno medio de la simulación efectuada.
Nombre:	Por_Ciento_Uso_CPU()
Descripción:	Calcula el por ciento de uso de la CPU de la simulación.
Nombre:	Redondear()
Descripción:	Redondea los resultados de los cálculos estadísticos efectuados.

Tabla 16 Descripción de la clase EstrategiaAlgoritmo

Nombre: EstrategiaAlgoritmo	
Tipo de clase: entidad	
Atributo	**Tipo**
instante	int
Para cada responsabilidad:	
Nombre:	Iterar()
Descripción:	Es un método abstracto, que es implementado en las clases hijas de esta clase, estas clases hijas son los diferentes algoritmos de planificación.

Tabla 17 Descripción de la clase FCFS

Nombre: FCFS	
Tipo de clase: entidad	
Para cada responsabilidad:	
Nombre:	Iterar()
Descripción:	Planifica la asignación del procesador a los procesos contenidos en ListaProcesos de la clase CPU, por orden de llegada a estado listo.

Tabla 18 Descripción de la clase SJF

Nombre: SJF	
Tipo de clase: entidad	
Para cada responsabilidad:	
Nombre:	Iterar()
Descripción:	Planifica la asignación del procesador a los procesos contenidos en ListaProcesos de la clase CPU. Se asigna el procesador al proceso que esté en estado listo, que menor tiempo restante tenga, todo el tiempo que este proceso requiera para su ejecución.

Tabla 19 Descripción de la clase SRTF

Nombre: SRTF	
Tipo de clase: entidad	
Para cada responsabilidad:	
Nombre:	Iterar()
Descripción:	Planifica la asignación del procesador a los procesos contenidos en ListaProcesos de la clase CPU. Se asigna el procesador al proceso que esté en estado listo, que menor tiempo restante tenga. En caso de que otro proceso llegue a estado listo y que tenga menor tiempo que el que se esté ejecutando, se le quita el procesador al que está haciendo uso de él y se le asigna al que llegó que posee menor tiempo.

Tabla 20 Descripción de la clase RR

Nombre: RR	
Tipo de clase: entidad	
Atributo	**Tipo**
cuantum	int
Para cada responsabilidad:	
Nombre:	Iterar()
Descripción:	Planifica la asignación del procesador a los procesos contenidos en ListaProcesos de la clase CPU. El procesador se le asigna a los procesos que estén en estado listo por un tiempo y un orden predeterminado.

Tabla 21 Descripción de la clase Graficas_Estadisticas

Nombre: Graficas_Estadisticas	
Tipo de clase: entidad	
Atributo	**Tipo**
dataset1	DefaultCategoryDataset
Para cada responsabilidad:	
Nombre:	Graficas_Tiempo_Espera()
Descripción:	Genera la gráfica que representa los valores de estadísticas tiempo de espera de cada proceso, haciendo uso de JFreeChart.
Nombre:	Graficas_Tiempo_Retorno()
Descripción:	Genera la gráfica que representa los valores de estadísticas tiempo de retorno de cada proceso, haciendo uso de JFreeChart.

Tabla 22 Descripción de la clase Administrador_Grafico

Nombre: Administrador_Grafico	
Tipo de clase: entidad	
Atributo	**Tipo**
largo	int
ancho	int
espacio	int
Para cada responsabilidad:	
Nombre:	Pintar_String_Horizontal
Descripción:	Pinta el nombre del eje horizontal ("INSTANTES").
Nombre:	Pintar_String_Vertical
Descripción:	Pinta el nombre del eje horizontal ("PROCESOS").
Nombre:	Pintar_Nombres_Procesos

Descripción:	Pinta los nombres de todos los procesos en el eje vertical.
Nombre:	Pintar_Instantes
Descripción:	Pinta todos los instantes de todo el proceso de simulación.
Nombre:	Pintar_Eje_Vertical
Descripción:	Pinta el eje vertical.
Nombre:	Pintar_Eje_horizontal
Descripción:	Pinta el eje horizontal.
Nombre:	Pintar_ListaProceso
Descripción:	Pinta la simulación de la planificación de los procesos existentes en el sistema.

Tabla 23 Descripción de la clase Proceso

Nombre: Proceso	
Tipo de clase: entidad	
Atributo	**Tipo**
nombre	String
inst_llegada;	int
prioridad;	int
rafaga_actual;	int
inst_finalizo	int
rafagas;	List<Rafaga>
historial_estados;	List<Enum_Estado>
tiempo_restante;	int
RR	int
Para cada responsabilidad:	
Nombre:	Clone()
Descripción:	Clona el proceso en cuestión y lo devuelve.

Anexo 2: Algunos Estilos de Visualización alcanzados.

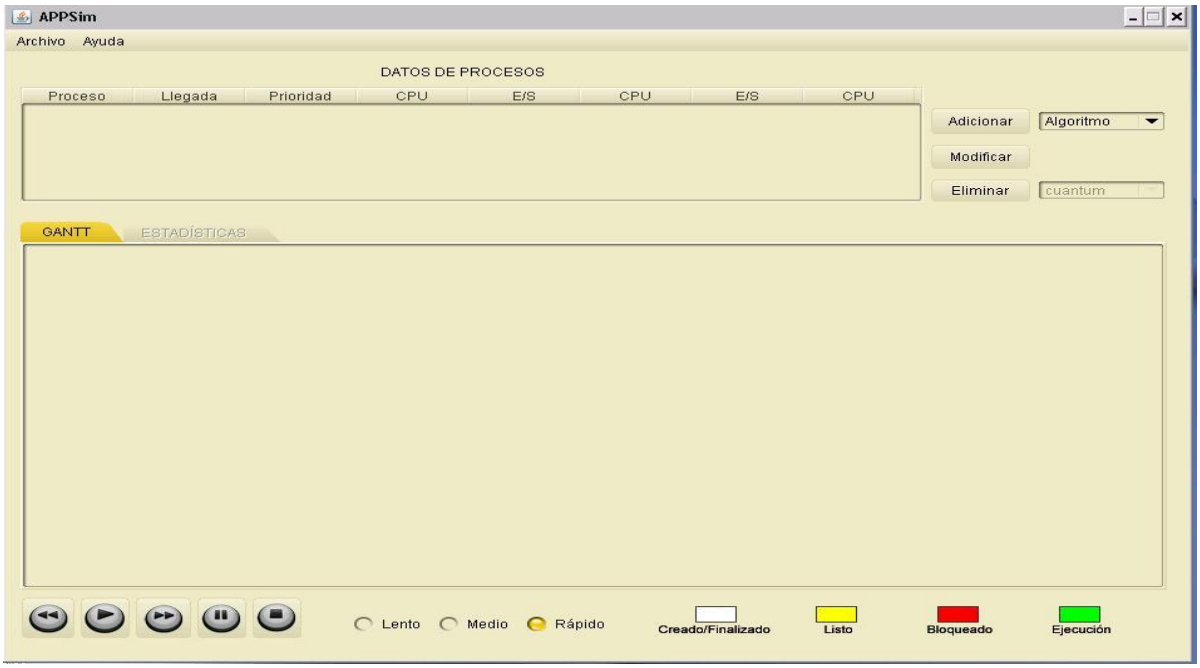


Figura 27 Interfaz Formulario Principal

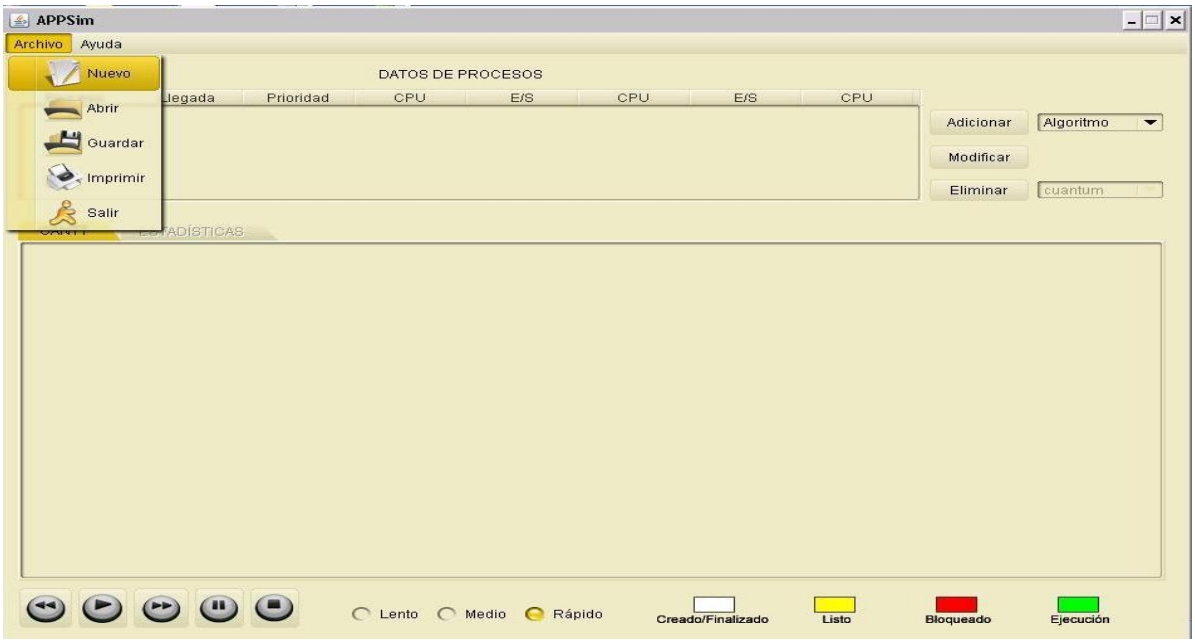


Figura 28 Interfaz Formulario Principal con Menú

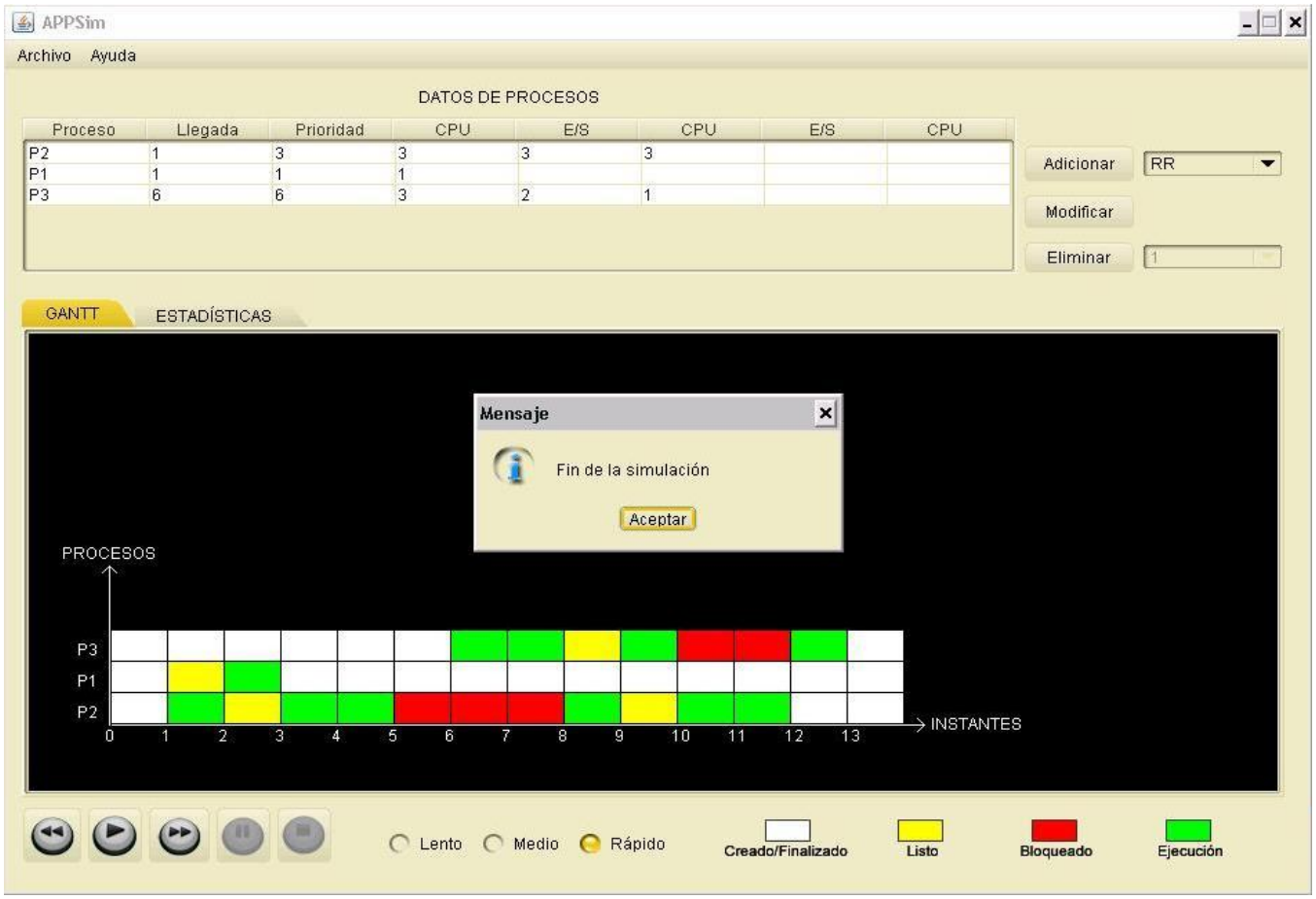


Figura 29 Interfaz de Simulación

Figura 30 Interfaz Adicionar Proceso

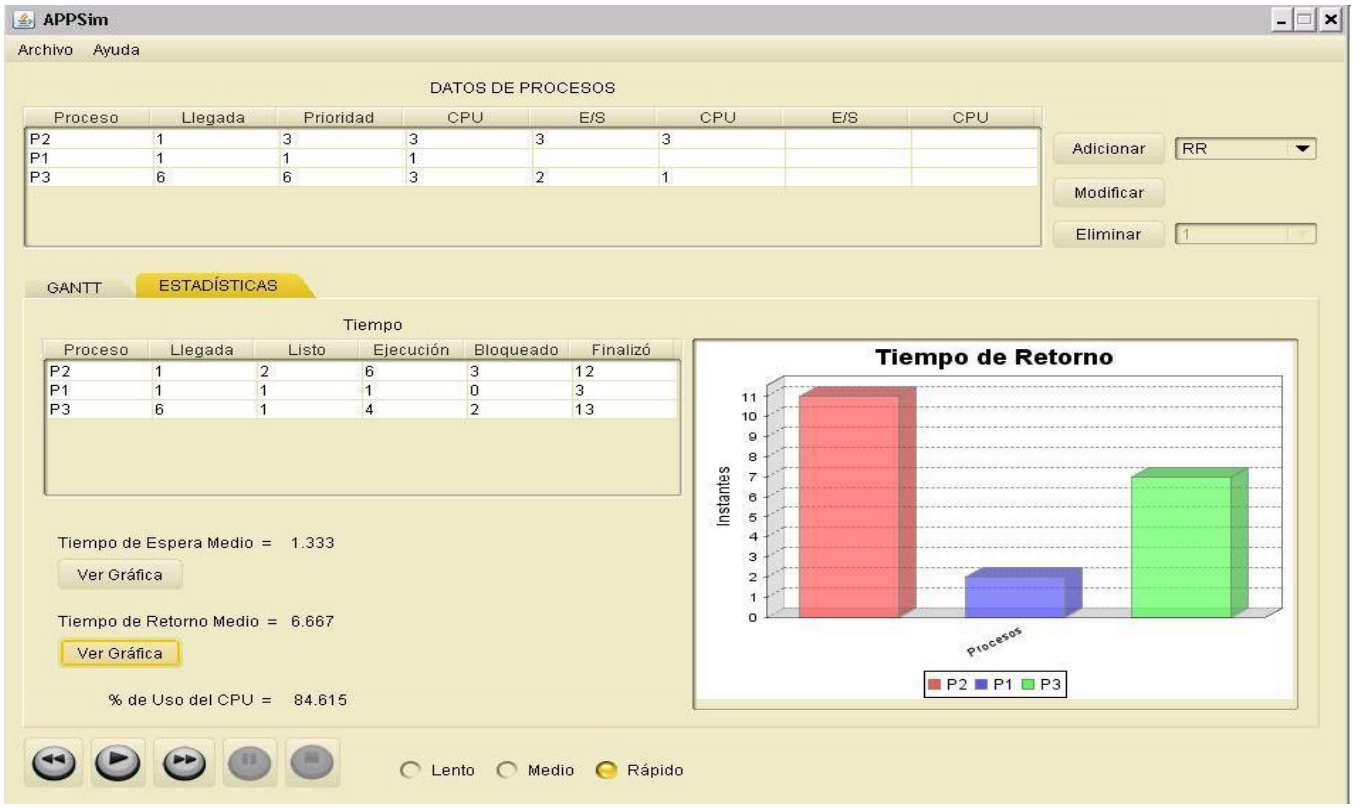


Figura 31 Interfaz Estadísticas

Figura 32 Interfaz Modificar Proceso

GLOSARIO DE TÉRMINOS

CPU: La unidad central de procesamiento, CPU (por sus siglas del inglés Central Processing Unit), o, simplemente, el procesador, es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de computadora.

GUI: Graphic User Interface o Interfaz Gráfica de Usuario.

UCI: Universidad de las Ciencias Informáticas.

Plug-ins: Aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

CASE: Computer Aided Software Engineering (Herramientas de ingeniería de software asistida por computadora).

E/S: Entrada/Salida. Dispositivos utilizados para la entrada y salida de datos

IDE: Entorno de desarrollo Integrado.

GNOME: Es un entorno de escritorio para sistemas operativos de tipo Unix bajo tecnología X Window. Forma parte oficial del proyecto GNU. Nació como una alternativa a KDE.

IEEE: corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos, una asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas.

GNU LGPL: La Licencia Pública General Reducida de GNU, o más conocida por su nombre en inglés GNU Lesser General Public License (antes GNU Library General Public License o Licencia Pública General para Bibliotecas de GNU), o simplemente por su acrónimo del inglés GNU LGPL es una licencia de software creada por la Free Software Foundation.

Framework: es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

CDDL: Common Development and Distribution License (Desarrollo Común y Licencia de Distribución).

GLOSARIO DE TÉRMINOS

Open Source: Código abierto. Es el término con el que se conoce al software distribuido y desarrollado libremente.

J2EE: Java 2 Enterprise Edition. Define un estándar para el desarrollo de aplicaciones empresariales multicapa diseñado por Sun Microsystems.