Universidad de las Ciencias Informáticas

Facultad 1

Herramienta educativa para el estudio de los algoritmos de gestión de los sistemas operativos

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Eulicer Ernesto Martínez Vega

Tutores: Ing. Mónica Ma. Albo Castro

Mtr. Raudel González Echenique

La Habana, junio 2016

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del presente trabajo de diploma y se reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Eulicer Ernesto Martínez Vega

Autor

_____ _____

Ing. Mónica Ma. Albo Mtr. Raudel González

Tutora Tutor

Agradecimientos

A todos los profes del Departamento de Programación por ayudarme con esta dura tarea de hacer un documento decente y exponer en un tiempo por debajo de los 15 min. A mis tutores y amigos, Mónica y Raudel, por ayudarme, ponerme el dedo para que todo saliera bien y estar ahí para todo. A Javier por ser un digno oponente y prestarme su ayuda. A los venenosos del 124 103 y a todas aquellas personas que estuvieron ahí y posibilitaron que se pudiera desarrollar este trabajo.

A mi mamá por ser la persona más comprensiva y atenta del mundo, además de brindarme todo su apoyo moral y vale recalcar monetario Jajaja. Este trabajo va dedicado a ella.

Gracias a todos.

Resumen

En la actualidad las Tecnologías de la Información y las Comunicaciones (TIC) forman parte de la cultura tecnológica que rodea el mundo y con la que se debe convivir. A partir de la experiencia de los profesores de Sistemas Operativos se detecta que los estudiantes continúan presentando dificultad en el dominio de los temas de la asignatura. Una de las causas detectadas radica en que al momento de resolver de forma individual los ejercicios los estudiantes no tienen como comprobar si está correctamente desarrollado. Es por ello que se decide desarrollar una herramienta educativa, que permita la resolución y visualización de los ejercicios en la asignatura Sistemas Operativos. Con este fin se realizó un estudio sobre: metodologías de desarrollo de software, herramientas de modelado, lenguajes de programación, entre otros aspectos asociados a la Ingeniería de Software. Posteriormente se implementó una herramienta que permite la resolución de los ejercicios, además, simular el desempeño de los diferentes algoritmos de planificación, reemplazo de páginas y peticiones al disco. Finalmente se aplicaron varias técnicas de pruebas dirigidas a evaluar la calidad del sistema, obteniéndose resultados satisfactorios.

Palabras clave: algoritmos, dominio, herramienta educativa, sistemas operativos.

Índice

Introducción

La asignatura Sistemas Operativos (SO) es estudiada en las carreras de perfil informático, por lo tanto, la Universidad de Ciencias Informáticas (UCI) también la incluye en su currículo. La enseñanza de dicha asignatura se vuelve compleja dado que los SO son un concepto intangible, al igual que cada uno de sus componentes funcionales, que son elementos de software que solo se manejan a bajo nivel.

Precisar un concepto de SO con exactitud es complejo, parte del problema consiste en que realiza dos funciones que básicamente no están relacionadas entre sí. Se puede definir: “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”[1]. Teniendo en cuenta lo anteriormente planteado se puede definir qué SO es un software que controla y ejerce control, enfocado principalmente a administrar recursos.

La UCI se encuentra actualmente enfrascada en el proceso de perfeccionamiento de la enseñanza para la formación de profesionales en Ciencias Informáticas. 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 SO, que 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 SO.
- Establecer las diferencias entre SO 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 SO.
- Configurar SO.
- 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 SO para la programación de sistemas informáticos.
- Manejar literatura en idioma extranjero con respecto a los SO y sus técnicas de diseño. [2]

Para dar cumplimiento a todos estos objetivos, la asignatura se divide en tres temas, cada tema se refiere a uno de los componentes de un sistema operativo. En la Facultad 1 de la UCI el colectivo de profesores de la asignatura se ha dedicado en los últimos años a trabajar en la búsqueda y elaboración de medios didácticos que ayuden a la comprensión del funcionamiento del SO. Sin embargo, entre los temas que continúan presentando mayor dificultad se encuentran los algoritmos de planificación del procesador, de reemplazo de páginas y de planificación de las peticiones al disco duro. Una de las causas detectadas por el colectivo de profesores radica en que al momento de resolver de forma individual los ejercicios los estudiantes no tienen como comprobar si está correctamente desarrollado o no hasta que pueden consultar al profesor.

Una manera de poder resolver los inconvenientes expuestos es a través de los recursos educativos digitales, donde habilidades teóricas y prácticas se conjugan permitiendo un amplio desarrollo. Estos además están hechos para: informar sobre un tema, ayudar en la adquisición de un conocimiento, reforzar un aprendizaje, remediar una situación desfavorable, favorecer el desarrollo de una determinada competencia y evaluar conocimientos [3]. Otra característica es que pueden tratar las diferentes materias de formas muy diversas, entre ellas la simulación, facilitando a los estudiantes una información estructurada y posibilidades de interacción. [4]

En la actualidad el uso de entornos virtuales para el desarrollo del proceso de enseñanza y aprendizaje es creciente, a estas tendencias no ha escapado la universidad cubana, siendo destacada en el uso de las Tecnologías de la Informática y las Comunicaciones (TIC) la UCI. Actualmente la asignatura SO del perfil de profesional de la UCI no cuenta con herramientas educativas que apoyen el estudio individual de los estudiantes lo que provoca que acumulen dudas sobre los resultados de los ejercicios y tengan que esperar a consultar al profesor para aclararlas, lo cual implica una pérdida de tiempo en su autopreparación.

Bajo dichas condiciones es identificado el siguiente **problema de la investigación**: ¿Cómo contribuir al

estudio de los algoritmos que se imparten en los diferentes temas de la asignatura para facilitar a los estudiantes la comprensión de dichos algoritmos?

Esta investigación tiene como **objeto de estudio**: herramientas educativas en el proceso de enseñanza-aprendizaje. Enmarcándose en el **campo de acción:** herramientas educativas para apoyar el proceso de enseñanza-aprendizaje específicamente en la asignatura de Sistemas Operativos.

En este sentido, se presenta como **objetivo general**: desarrollar una herramienta educativa que, a través de la simulación de los algoritmos estudiados en la asignatura, muestre a los estudiantes un análisis gráfico de la resolución de ejercicios.

Para el cumplimiento exitoso del objetivo general se han definido como **objetivos específicos**:

- Sistematizar la utilización de herramientas educativas en la enseñanza de los SO en las carreras informáticas.
- Valorar las metodologías y entornos de desarrollo para la creación de una herramienta educativa que se ajuste a las necesidades identificadas.
- Desarrollar la herramienta educativa con los requerimientos definidos.
- Evaluar la calidad funcional de la herramienta desarrollada.

De este modo quedaron plasmadas las siguientes **tareas de investigación**:

- Realización de un estudio de los sistemas de auto aprendizaje, recursos didácticos y objetos de aprendizaje.
- Realización de un estudio de las metodologías de desarrollo de software, herramientas CASE[1] y lenguajes de modelado.
- Realización de la especificación de requisitos de software, diagrama de clases y artefactos de la metodología de desarrollo de software que se defina conveniente para el proceso de desarrollo de

[1] CASE por sus siglas en inglés: Computer Aided Software Engineering.

la herramienta.

- Evaluación de la calidad de los artefactos obtenidos.

Se define como **idea a defender**: el desarrollo de una herramienta educativa, que provea a los estudiantes un medio de enseñanza y auto aprendizaje donde estos puedan realizar análisis y comparaciones con los resultados de ejercicios propuestos en clases, puede mejorar la comprensión de los contenidos por parte de los estudiantes.

Para el desarrollo de la investigación se utilizaron **métodos de investigación** que son descritos a continuación:

Métodos teóricos:

- **Histórico - Lógico:** para profundizar en los antecedentes de la utilización de las Tecnologías de la Información y las Comunicaciones en los procesos de enseñanza - aprendizaje y sus tendencias actuales.
- **Analítico - Sintético**: se utiliza en la revisión bibliográfica, de trabajos de diploma concernientes a la temática, planes de estudio de carreras donde se estudie la asignatura en cuestión y las herramientas educativas que utilizan para sus actividades prácticas y de autoaprendizaje, de los libros de texto de la asignatura Sistemas Operativos que se utilizan en la Universidad**.**

Métodos empíricos:

- **Observación**: para la recopilación de información de las características y comportamiento de los estudiantes al utilizar las TIC en la enseñanza y el aprendizaje con el uso de los Objetos de Aprendizaje en la asignatura de SO.

Técnicas de recopilación de información:

- **Entrevista**: para la recopilación de información especializada o dirigida a directivos, profesores y alumnos que interactúan con las TIC en el proceso de enseñanza - aprendizaje presencial o mixto de la asignatura de SO en la Facultad 1 de la UCI.

Para presentar los resultados obtenidos durante la investigación, el trabajo consta de tres capítulos

estructurados de la siguiente forma:

Capítulo 1: **Sistemas Operativos. Herramientas educativas para su aprendizaje**.

Se describe todo lo referente al estado del arte respecto al tema a investigar, haciendo énfasis en el uso de herramientas educativas, sistemas de auto aprendizaje y objetos de aprendizaje. También se abordan sobre las metodologías, lenguajes de modelado y herramientas a utilizar.

Capítulo 2: Herramienta educativa para la simulación de algoritmos utilizados en la asignatura.

Se describe el desarrollo de la solución propuesta a partir de requisitos funcionales y no funcionales, la ingeniería de requisitos, los patrones utilizados, y una serie de diagramas diseñados para el desarrollo de las herramientas educativas.

Capítulo 3: Implementación y pruebas del sistema.

Se expone la solución que se le dará al problema, a través de un conjunto de artefactos. Se verán además las diferentes pruebas que se le aplicarán a la herramienta que se desea desarrollar y luego se procede a la validación para verificar que la aplicación desarrollada cumple con el objetivo planteado, concluyendo con el resultado de dicha validación.

Capítulo 1: Sistemas Operativos. Herramientas educativas para su aprendizaje.

En el presente capítulo se realizará un estudio de los conceptos claves de la investigación: herramientas educativas y las ventajas que estas proveen al proceso de enseñanza-aprendizaje. Se abordará sobre las metodologías, los lenguajes de programación, herramientas de modelado para el diseño y la ingeniería de requisitos.

1.1. Sistemas Operativos

Los SO permiten interactuar amenamente con los diferentes dispositivos aun cuando los conocimientos acerca de las tecnologías no sean los más altos. A raíz de lo cual se considera que su estudio posee una inmensa importancia para los profesionales de la Informática y las Comunicaciones (en lo adelante IC).

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

- **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 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 está ahí para administrar todos los componentes de un sistema complejo. En esta última la misión es asegurar un reparto ordenado y controlado de los procesadores, memorias y dispositivos de E/S entre diferentes programas que compiten por ellos.

Por su parte en el libro: “Fundamentos de Sistemas Operativos” Silverchast y colectivo de autores plantean que un sistema operativo es similar a un gobierno, no realiza ninguna función por sí mismo: simplemente proporciona un entorno en el que otros programas pueden llevar a cabo un trabajo útil. El sistema operativo proporciona los medios para hacer un uso adecuado de los recursos de hardware durante el funcionamiento del sistema informático. [5]

En el caso de William Stalin en su libro: “Sistemas Operativos” propone la idea de que un sistema operativo es el programa del sistema que controla todos los recursos del computador y ofrece el soporte básico sobre el cual pueden escribirse los programas de aplicación. [6]

También se tuvo en cuenta el concepto mencionado en la introducción, tomado del libro Sistemas Operativos Modernos del autor Tanenbaum [1]. Según las definiciones anteriores un SO es el que controla y además ejerce control, es enfocado en el sentido de que un SO como tal es un administrador de recursos. De otra forma, el SO es el encargado de brindar al usuario la manera más fácil, sencilla y amigable de operar, codificar, interpretar y emitir órdenes al procesador principal, para que éste realice las tareas necesarias para completar la orden.

Se evidencia el manejo de conceptos abstractos, ya que el SO, al contrario de lo que se piensa, no es la interfaz gráfica sino el código que permite el manejo de los recursos y de las aplicaciones o software. Por esta razón la comprensión del funcionamiento de sus componentes se ve dificultada, haciéndose necesario el uso de herramientas educativas que apoyen el proceso de enseñanza-aprendizaje.

1.2. Herramientas y/o recursos educativos

Los recursos educativos digitales son materiales compuestos por medios digitales y producidos para facilitar el aprendizaje. Se considera adecuado si ayuda a la asimilación de contenidos conceptuales, a adquirir habilidades procedimentales y mejorar la persona en actitudes o valores. Los simuladores, las aplicaciones de ejercitación y práctica, entre otras, permiten interactuar con el objeto de conocimiento para comprender procesos, desarrollar habilidades, relacionar e integrar el conocimiento. [3]

Los recursos y materiales didácticos son todo el conjunto de elementos, útiles o estrategias que el profesor utiliza, o puede utilizar, como soporte, complemento o ayuda en su tarea docente. Los recursos didácticos deberán considerarse siempre como un apoyo para el proceso educativo. [7]

Dentro de las funciones que desarrollan, se puede encontrar [7]:

- Brindar información a los estudiantes ayudando a enriquecer su conocimiento.
- Permitir a los alumnos ejercitar y desarrollar las habilidades.
- Despertar la motivación y el interés del contenido educativo, gracias a la fácil interacción que brinda.
- Los profesores pueden evaluar a los estudiantes en cada momento, manteniendo un seguimiento del comportamiento de este en todo el transcurso del contenido impartido.

Después de indagar en los conceptos propuestos por las diferentes referencias concluimos que los recursos didácticos abarcan cualquier material que sea construido con el objetivo de ayudar al profesor en su tarea de enseñar y al alumno facilitar su aprendizaje. Estos recursos han permitido mejorar la calidad del aprendizaje y es importante destacar el papel que han jugado los objetos de aprendizaje, pues han ayudado a enriquecer el proceso docente educativo. [7][8]

El uso de recursos educativos digitales y entornos virtuales para apoyar la realización del proceso de enseñanza y aprendizaje como nueva tendencia va en auge en la educación en general. Lo cual es coherente con el llamado que se realiza desde el V Congreso del Partido Comunista de Cuba cuando acerca del rol de la informática en la sociedad cubana se plantea que:

"El país debe encaminarse resueltamente a la modernización informática mediante un programa integral que involucre a las organizaciones que deben proveer los recursos materiales, financieros e intelectuales y a las entidades económicas, políticas y sociales que deben traducirlos en más y mejores productos y servicios. La industria de los servicios informáticos deberá asegurar la modernidad de su base técnica y organizativa, y la elevación constante del nivel científico-técnico de sus especialistas con vistas a garantizar esos propósitos." [9]

Ventajas de la utilización de herramientas educativas [3] [7]

- 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.
- Marca las posibilidades para una nueva clase más desarrolladora.
- Poseen la capacidad para acercar al estudiante a la comprensión de procesos, mediante las simulaciones y laboratorios virtuales que representan situaciones reales o ficticias a las que no es posible tener acceso en el mundo real cercano.
- Facilitar el autoaprendizaje al ritmo del estudiante, dándole la oportunidad de acceder desde un computador y volver sobre los materiales de lectura y ejercitación cuantas veces lo requiera.

Luego de comprobar las ventajas que ofrecen las tecnologías para la creación de herramientas y recursos educativos que apoyan los procesos de enseñanza-aprendizaje. Ahora es necesario revisar cómo se aplican estos recursos y herramientas en la enseñanza de los SO en los diferentes programas de estudio que la incluyen.

1.2.1. Uso de herramientas educativas en la asignatura SO

La enseñanza de los SO es una tarea que se torna profundamente compleja debido al alto nivel de abstracción que requiere por parte de los estudiantes para lograr un buen entendimiento acerca del funcionamiento interno de cada sistema de administración. Se consultaron en diferentes centros de altos estudios la inclusión de dicha asignatura en los perfiles de ingeniería informática y el uso de herramientas educativas como apoyo en el proceso de enseñanza aprendizaje.

La Universidad de Alicante, España, incluye dentro de su plan de estudio el uso de herramientas educativas, generalizando en prácticas con ordenadores. [10]

Por su parte la Universidad de Sevilla, España, permite optar por un grado en ingeniería informática/ingeniería en computadores donde se incluye el estudio de los SO. El plan de estudio de esta asignatura está constituido mayormente por clases magistrales, donde el profesor utiliza diapositivas, las cuales se consideran material de apoyo, nunca material de estudio. Entre las herramientas que se utilizan para el apoyo al aprendizaje solo se constata, al igual que en el centro visto anteriormente, los laboratorios o prácticas con ordenadores. [11]

En América, específicamente México, el Instituto Tecnológico Superior de Nuevo Casas Grandes; la

asignatura Sistemas Operativos, perteneciente a la carrera Ingeniería Informática, Ingeniería en Sistemas Computacionales e Ingeniería en Tecnologías de la Información y Comunicaciones, desempeña un papel fundamental en el plan de estudio de estas ingenierías, porque a través de ella el estudiante conoce en detalle los componentes, las estructuras y las funciones de un sistema operativo concreto, así como aspectos generales de la construcción de sistemas operativos. También se evidencia como actividad la práctica de laboratorios [12].

Después de haber apreciado las particularidades del uso de herramientas educativas en otros centros, se procedió a clasificar su uso en la UCI. Las entrevistas sobre el proceso de impartición de las clases de SO, realizadas a 20 estudiantes del tercer año de la Facultad 1, han arrojado resultados de comprensión muy positivos por cuanto el profesor, más que limitarse a la mera explicación, hace uso de medios didácticos basados en las TIC, que aunque no tengan toda la calidad necesaria, sí permitieron simular el funcionamiento de los diferentes algoritmos, materiales indispensables utilizados como parte del proceso de enseñanza - aprendizaje de la asignatura.

A raíz de esto, en la UCI se han desarrollado diferentes herramientas educativas para potenciar la comprensión de la asignatura, permitiendo así una mejor comprensión y dominio de los contenidos que se imparten. Entre estas herramientas se puede hacer referencia a las siguientes:

- Simulador de Algoritmos de Planificación de procesos.
- Simulador de Gestión de Dispositivos de Entrada y Salida para el Laboratorio Virtual.
- Evaluador de Memoria para el Laboratorio Virtual de Sistemas Operativos.

Simulador de Algoritmos de Planificación de procesos.

Es una aplicación multiplataforma para la representación del proceso de planificación de la CPU de acuerdo a algunas de las políticas clásicas, se trabajan con algoritmos tales como Round Robbin y se muestran, a través de gráficos y tablas los resultados obtenidos del proceso de simulación, 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.

Los resultados obtenidos por el equipo de desarrollo durante las pruebas realizadas a esta aplicación, donde se tomaron 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. [13]

Simulador de Gestión de Dispositivos de Entrada y Salida para el Laboratorio Virtual.

Es una herramienta que permite simular los diferentes algoritmos de planificación, así como otros ejercicios del tema de Gestión de Dispositivos de Entrada y Salida de la asignatura de Sistemas Operativos, donde los estudiantes pueden encontrar un entorno de apoyo en su aprendizaje del tema. Entre los algoritmos se tuvieron en cuenta FCFS ("*first come first served*"), SSF (*Short Seek-Time First*), entre otros. Algunos algoritmos tienen una gran complejidad por lo que ver de forma gráfica su funcionamiento y resultados contribuye a un alza en la comprensión del conocimiento por parte de los estudiantes durante el proceso de enseñanza-aprendizaje. Lo cual fue uno de los objetivos de citado trabajo, como parte del proceso de desarrollo de esta aplicación el equipo de desarrollo aplicó pruebas dirigidas a evaluar la calidad del sistema, obteniéndose resultados satisfactorios. [14]

Evaluador de Memoria para el Laboratorio Virtual de Sistemas Operativos.

El Evaluador de Memoria permite a los estudiantes la comprobación de los conocimientos adquiridos a través de la interacción con el Laboratorio Virtual donde podrá resolver en el evaluador ejercicios que el mismo contiene, un simulador solucionará los ejercicios, evaluando luego las respuestas devueltas por el simulador con las introducidas por el usuario, estos ejercicios a resolver solo serán algoritmos del tema de Gestión de memoria explícitamente las Técnicas de Asignación y los Algoritmos de reemplazos.[15]

A pesar de que las pruebas fueron satisfactorias, estos simuladores nunca se implantaron dentro de los materiales didácticos del programa de la asignatura de Sistemas Operativos en la UCI. Además, estas herramientas, visto desde el modo académico, están especializadas en un tema en específico de la asignatura SO y como se evidencia no existe una interrelación entre ellas, dificultando así la comprensión y aplicación de las mismas en los laboratorios de la asignatura, o el estudio individual. En la Tabla 1 se puede observar de forma concisa el alcance en materia de conocimientos que abarcan cada una de las herramientas ya desarrolladas con anterioridad en la universidad.

Después de comparar los diferentes alcances, se evidencia la necesidad de una nueva aplicación que, al ser comparada con las actuales muestre un mayor alcance en el dominio e interacción con los temas de la asignatura. Por dicho motivo, se decide desarrollar una herramienta (AlgSIM) que integre las diferentes

temáticas presentadas en la asignatura SO y permita además la simulación de los diferentes algoritmos de gestión que son estudiados en la materia.

Tabla 1: Temas abordados en las herramientas existentes.

Herramientas existentes	Contenidos abordados		
	Planificación de procesos	Algoritmos de reemplazo de páginas	Algoritmos de Planificación de peticiones al disco duro
Simulador de Gestión de Dispositivos de Entrada y Salida para el Laboratorio Virtual.			X
Evaluador de Memoria para el Laboratorio Virtual de Sistemas Operativos		X	
Simulador de Algoritmos de Planificación de procesos.	X		

1.3. *Valoración de la metodología y el entorno de desarrollo.*

Como principal objetivo de esta investigación se tiene la creación de una herramienta educativa que simule los diferentes algoritmos que se estudian en la asignatura. Como herramienta informática al fin, para su desarrollo, es necesario el uso de metodologías y tecnologías de desarrollo de software, para definir cuál utilizar se realiza un estudio de las principales metodologías y tecnologías, teniendo en cuenta las tendencias actuales al desarrollo de software libre con metodologías ágiles.

1.3.1. Metodologías de desarrollo ágiles

Se tiene como objetivo principal al evaluar una metodología el hecho de que resulta como meta principal el desarrollo óptimo de una aplicación. En este caso el equipo de desarrollo se encuentra en un entorno cambiante, se busca simplicidad en las soluciones propuestas puesto que los usuarios finales no ven soluciones parciales sino una versión final de la herramienta. En coordinación con el cliente el desarrollo de la aplicación se basa en esta idea por lo que se analizaran metodologías ágiles que son las que responden a las características que presentan el equipo de desarrollo.

Metodología Scrum:

Scrum es una metodología ágil de desarrollo de proyectos que toma su nombre y principios de los estudios realizados sobre nuevas prácticas de producción por Hirotaka Takeuchi e Ikujijo Nonaka a mediados de los 80. En Scrum se realizan entregas parciales del resultado final del proyecto, priorizadas

por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados en breve tiempo, los requisitos son cambiantes o poco definidos y la innovación, la competitividad y la productividad es fundamental. [16]

Según el propio autor, Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, además cuando es necesario identificar y solucionar ineficiencias sistemáticamente o se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.[16]

¿Cómo se desarrolla el proceso de Scrum?

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento del producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite [16].

Metodología Crystal Clear

Crystal Clear no es una metodología en sí misma sino una familia de metodologías con un "código genético" común. Las mismas presentan un enfoque ágil, con gran énfasis en la comunicación, y con cierta tolerancia que la hace ideal en los casos en que sea inaplicable la disciplina requerida. La misma se define con mucho énfasis en la comunicación, y de forma muy liviana en relación con los entregables. Crystal maneja iteraciones cortas con *feedback* frecuente por parte de los usuarios/clientes, minimizando de esta forma la necesidad de productos intermedios. Otra de las cuestiones planteadas es la necesidad de disponer de un usuario real, aunque sea a tiempo parcial para realizar validaciones sobre la interfaz de usuario y para participarse en la definición de los requerimientos funcionales y no funcionales del software. [16]

Una cuestión interesante que surge del análisis de la serie Crystal es que las personas involucradas escogen aquellos principios que les resultan efectivos y mediante la aplicación de la metodología en diversos proyectos agregan o remueven principios en base al consenso grupal del equipo de desarrollo. [16]

Prioridades establecidas por Crystal Clear

Crystal Clear establece un conjunto de prioridades y principios que sirven de guía para la toma de decisiones, como lo es [16]:

- Eficiencia en el desarrollo: para hacer que los proyectos sean económicamente rentables
- Seguridad en lo que se entrega.
- Habitabilidad: hacer que todos los miembros del equipo adopten y sigan las convenciones de trabajo establecidas por el equipo mismo.

Propiedades establecidas por Crystal Clear

Frecuencia en las entregas: entregar al usuario funcionalidad "usable" con una frecuencia de entre 2 semanas y no más de un mes [16]:

- Comunicación: Crystal Clear toma como uno de sus pilares a la comunicación. Promueve prácticas como el uso de pizarrones, pizarras y espacios destinados a que todos (miembros del equipo y visitas) puedan ver claramente el progreso del trabajo.
- Crecimiento reflexivo: es necesario que el equipo lleve a cabo reuniones periódicas de reflexión que permitan crecer y hacerse más eficientes.

XP (*Extreme Programming*)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. En vez de planificar, analizar y diseñar para el futuro distante, hacer todo esto un poco cada vez a través de todo el proceso de desarrollo. Entre sus objetivos se encuentran: [17]

- Establecer las mejores prácticas de Ingeniería de Software en el desarrollo de proyectos.

- Mejorar la productividad de los proyectos.
- Garantizar la Calidad del Software desarrollando, haciendo que este supere las expectativas del cliente.

En el contexto XP se pueden identificar ciertas regularidades como son: cliente bien definido, los requisitos pueden cambiar a través del proceso de desarrollo, grupo de desarrolladores pequeño e integrado. Otras actitudes a tener en cuenta son las características que presenta esta metodología, entre las cuales se pueden mencionar que es una metodología basada en prueba y error y que, además, está fundamentada en el concepto Valores y Prácticas [17]. El estilo XP:

- Está orientada hacia quien produce y usa el software.
- Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema.

A pesar de las bondades que brinda Scrum, considerando además la propuesta de estructura del proceso de desarrollo de la herramienta didáctica, en la cual no se incluye entregas incrementales a corto plazo entonces desechamos la metodología Scrum como metodología de desarrollo ya que esta metodología trabaja sobre ese principio. Además, la metodología Crystal Clear promueve el crecimiento reflexivo, el cual no puede llevarse a cabo puesto que se cuenta con poco tiempo destinado a reuniones con el cliente. Por tanto, se determina utilizar XP porque se centra en el proceso de desarrollo de la aplicación, y no en otras actividades de gestión que, si tienen en cuenta las otras metodologías, y está orientada hacia quien produce y usa el software.

1.3.2. Entorno de desarrollo

En función de la portabilidad, lo cual es un requisito indispensable de la herramienta didáctica, se concluye desarrollar una aplicación de escritorio, la cual se refiere a que se ejecuta en una PC en específico. Teniendo en cuenta lo anterior, se valoraron dos lenguajes de programación: Java, Python.

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. Java es toda una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa. Es un lenguaje de propósito general,

concurrente 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 característica 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. Ofrece además una portabilidad básica por ser de arquitectura independiente e implementa otros estándares de portabilidad para facilitar su desarrollo. [18]

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables. [19]

Que una herramienta sea de código abierto, que existan comunidades que den soporte, y que genere todo el tiempo nuevos desarrollos e investigaciones, son pilares fundamentales para la resolución de problemas que surjan, o cuando se tenga que innovar en algo, tener fuentes de información abundante donde consultar. El lenguaje python presenta dificultades en el desarrollo de interfaces gráficas puesto que se trabaja mayormente con librerías agregando horas de trabajo por conceptos de investigación y entendimiento del uso de las mismas. Además, las herramientas educativas que se analizaron se desarrollaron en java, utilizar este lenguaje sería ventajoso porque permite la reutilización de código fuente, otro de los paradigmas del software libre. Por esto se decide utilizar Java como lenguaje de programación a emplear en el desarrollo de la herramienta.

Dentro de los IDE[2] 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*)

[2] IDE por sus siglas en inglés. Integrated Development Environment.

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). [20]

- 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 (plugins). Hay plugins para el desarrollo de Java (JDT Java Development Tools) así como para el desarrollo en C/C++, entre otros. [21].

Se determina utilizar NetBeans porque es muy difundido y conocido por una gran cantidad de programadores, lo cual es muy útil porque siempre se pueden consultar los sitios de las comunidades de desarrolladores. Además, hay que tomar en cuenta que en Eclipse es necesario agregar varios plugins para que funcione al cien por ciento en dependencia de lo que se necesite hacer.

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. [22]

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. 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 [22]:

Aumentar las probabilidades de una aceptación generalizada de la notación estándar del modelado, sin la obligación de adoptar un proceso oficial.

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.

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. A continuación, se caracterizan algunas de estas herramientas:

Visual Paradigm 8.0

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, entre otras. [23] 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 [23]:

- 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.

1.4. *Conclusiones parciales*

El uso de las herramientas educativas en el proceso de enseñanza-aprendizaje amplifica la capacidad de comprensión del estudiante. Dependiendo de las características para las que se haya diseñado la herramienta, este puede aclarar algunas de las dudas que le puedan surgir acerca del contenido sin necesidad de esperar al próximo encuentro con el profesor. Le permite además desarrollar habilidades y dominio sobre el contenido ya que es un medio donde puede aplicar lo aprendido en clases.

Durante el proceso de investigación se tuvieron en cuenta las herramientas que se utilizan en la impartición de la asignatura en centros tanto en el extranjero como nacionales. Las existentes se enfocan en determinadas áreas y no se han introducido dentro del plan de estudio de los Sistemas Operativos en la universidad. Es necesario la implementación de una nueva herramienta educativa que apoye el proceso de enseñanza-aprendizaje en la asignatura.

Al tener en cuenta aspectos tales como, la dificultad de la aplicación además de las características del equipo de desarrollo y el ambiente donde se implantará la herramienta a desarrollar; así como el auge de la política de la universidad de fomentar el software libre y las políticas de migración del país se concluye que: la opción óptima para desarrollar la herramienta educativa sería XP como metodología de desarrollo, Visual Paradigm 8.0 como herramienta CASE y UML como lenguaje de modelado. Además, la plataforma Java con su IDE de desarrollo NetBeans 8.0.1, con el objetivo de desarrollar una aplicación que unifique los contenidos y permita ser utilizado con fines didácticos en la impartición de la asignatura SO en el tercer año de la Facultad 1.

Capítulo 2: Herramienta educativa para la simulación de algoritmos utilizados en la asignatura.

En el presente capítulo se realiza la modelación de los procesos a simular, esto se hace a través del diagrama de flujo de procesos. También se especifican los requisitos funcionales y los no funcionales por los cuales se regirá el sistema propuesto, se identificarán mediante historias de usuario las relaciones entre los actores y los requisitos, así como la descripción de los mismos. El diseño consolidará el análisis propuesto con los diagramas de clases y la aplicación de patrones de diseño como una manera más práctica de describir ciertos aspectos; teniendo en cuenta las cualidades o propiedades que el sistema debe tener.

2.1. Descripción de la solución.

La herramienta que se desea desarrollar debe permitir a los estudiantes, a partir de los datos de ejercicios propuestos en clases: escoger el tema al que corresponde el algoritmo que se desee simular, la entrada de los datos necesarios. Debe permitir, además, simular los algoritmos estudiados en la asignatura y brindar un resultado. La posibilidad de que dicha simulación se realice de forma completa o paso a paso es otra bondad que debe poseer, mostrando las características del algoritmo y la respuesta en determinado instante de tiempo. Para poder visualizar el negocio descrito se estimó conveniente realizar un diagrama de flujo en el cual se expone de manera detallada y concisa el funcionamiento de la herramienta.

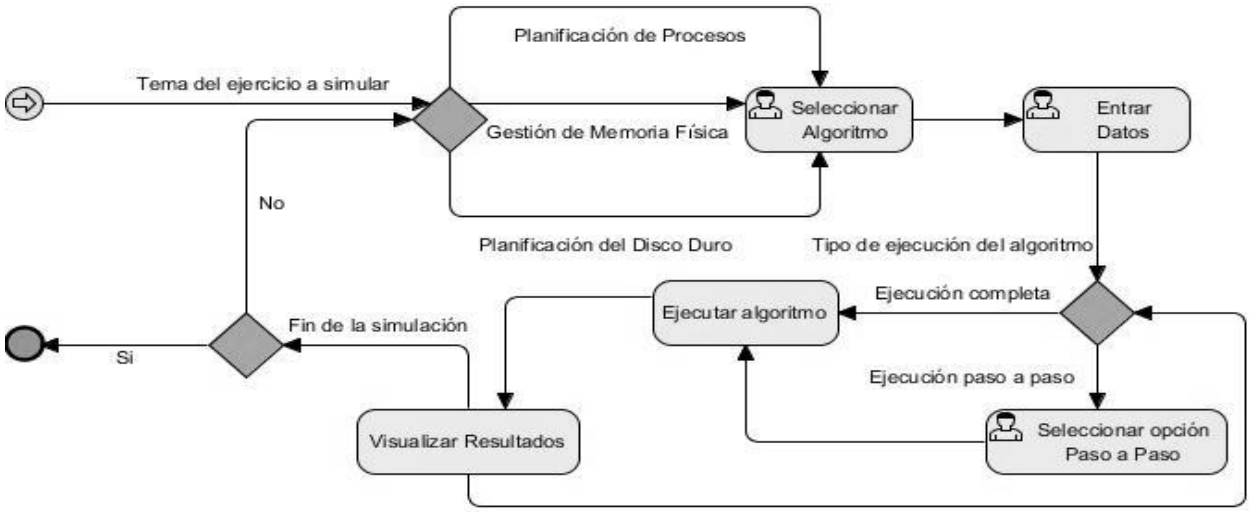

Figura 1: Diagrama de flujo de la herramienta (Elaboración propia)

2.2. Requerimientos del sistema.

Como parte del proceso de desarrollo se realizaron entrevistas con el cliente, las cuales arrojaron las siguientes especificaciones, mostrando a un nivel informal los requerimientos necesarios que debe poseer el producto:

1. Dado los datos clásicos de un ejercicio de clase práctica ejecutar los diferentes algoritmos de planificación del procesador, reemplazo de páginas, planificación de las peticiones al disco duro.
2. Visualización gráfica de la ejecución del algoritmo indicado para poder analizar y comparar con los resultados obtenidos en el desarrollo manual del ejercicio.
3. Ejecución paso a paso del algoritmo indicado para poder identificar el momento del error cometido en la resolución manual del algoritmo.
4. Visualización de las características del algoritmo indicado para refrescar el contenido teórico.

Además:

5. Debe tener una interfaz intuitiva.
6. Poder ejecutarla en cualquier tipo de computadora, sin necesidad de utilizar la red.

7. Funcione sobre cualquier plataforma de sistema operativo.
8. La visualización gráfica debe acompañarse de colores identificativos a cada estado según el algoritmo.

Se estudiaron las diferentes solicitudes, se consideró identificar los requisitos funcionales y no funcionales de la aplicación. Se evaluaron además los requisitos, los cuales se presentaron al cliente para su discusión y se aprobaron de forma satisfactoria, quedando enumerados y clasificados de esta manera:

Requisitos Funcionales:

1. **RF1** Ejecutar simulación de algoritmos de planificación de procesos seleccionado por el usuario.
2. **RF2** Ejecutar simulación de algoritmos de reemplazo de páginas seleccionado por el usuario.
3. **RF3** Ejecutar simulación de algoritmos de planificación de las peticiones al disco duro seleccionado por el usuario.
4. **RF4** Visualizar gráficamente la ejecución del algoritmo indicado por el usuario.
5. **RF5** Mostrar la ejecución paso a paso del algoritmo indicado por el usuario.

Requisitos no funcionales:

1. **RNF1** El diseño de la interfaz debe ser sencillo y fácil de usar con reconocimiento visual a través de elementos visibles que identifiquen cada una de sus acciones.
2. **RNF2** La aplicación no dependerá de un servidor para ejecutarse.
3. **RNF3** La herramienta debe ser multiplataforma, es decir, poder ejecutarse en cualquier sistema operativo.
4. **RNF4** La aplicación debe permitir la adición de más funcionalidades en el futuro (escalabilidad).

A partir de los requisitos funcionales y no funcionales aprobados en reunión con el cliente, de acuerdo con la metodología escogida, se elaboran las historias de usuario necesarias para especificar dichos requerimientos.

2.2.1. Historias de Usuario.

Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos. Son una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. En fin, son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes. [25]

A partir de los requisitos funcionales identificados se generaron tres historias de usuario, las cuales muestran una descripción de las funcionalidades de la herramienta, donde se ven involucrados los diferentes requisitos como se muestra a continuación:

Tabla 2 Historia de Usuario1

Historia de Usuario 1	
Número: RF1, RF4, RF5	**Nombre:** Ejecutar simulación del algoritmo de planificación de procesos, seleccionado por el usuario
Programador: Eulicer Ernesto Martínez Vega	**Iteración Asignada:** Primera Iteración
Prioridad: Alta	**Tiempo Estimado:** 0.13 días
Riesgo en Desarrollo: Ninguno	**Tiempo Real:** 0.25 días
Descripción: Los usuarios pueden seleccionar un algoritmo de planificación de procesos para simular su ejecución, para lo cual el sistema permite: • La entrada de los datos de los procesos activos en el sistema para ejecutar el algoritmo. • Realizar las iteraciones, ya sea ciclo completo o una sola iteración. • Mostrar en la correspondiente área de la interfaz el proceso realizado por el algoritmo que se esté simulando.	
Observaciones: 1. Si la lista de datos a procesar está vacía entonces la herramienta no permitirá la ejecución del algoritmo. 2. Si el usuario marca la casilla de ejecutar paso a paso el algoritmo seleccionado entonces la herramienta realizara una sola iteración del algoritmo. 3. Si la casilla esta desmarcada se procederá a realizar el proceso completo del algoritmo. 4. La casilla estará desmarcada (valor por defecto).	

Tabla 3 Historia de usuario 2

Historia de Usuario 2	
Número: RF2, RF4, RF5	**Nombre del requisito:** Ejecutar simulación de algoritmos de reemplazo de páginas, seleccionado por el usuario.
Programador: Eulicer Ernesto Martínez Vega	**Iteración Asignada:** Segunda Iteración
Prioridad: Alta	**Tiempo Estimado:** 0.13 días
Riesgo en Desarrollo: Ninguno	**Tiempo Real:** 0.25 días
Descripción: Los usuarios pueden seleccionar un algoritmo de reemplazo de páginas para simular su ejecución, para lo cual el sistema permite: • La entrada de los datos de las direcciones lógicas generadas por el CPU para ejecutar el algoritmo. • Realizar las iteraciones, ya sea ciclo completo o una sola iteración. • Mostrar en la correspondiente área de la interfaz el proceso realizado por el algoritmo que se esté simulando.	
Observaciones: 1. Si la lista de datos a procesar está vacía entonces la herramienta no permitirá la ejecución del algoritmo. 2. Si el usuario marca la casilla de ejecutar paso a paso el algoritmo seleccionado entonces la herramienta realizara una sola iteración del algoritmo. 3. Si la casilla esta desmarcada se procederá a realizar el proceso completo del algoritmo. 4. La casilla estará desmarcada (valor por defecto).	

Tabla 4 Historia de usuario 3

Historia de Usuario 3	
Número: RF3, RF4, RF5	**Nombre del requisito:** Ejecutar simulación de algoritmo de planificación de las peticiones al disco duro, seleccionado por el usuario.
Programador: Eulicer Ernesto Martínez Vega	**Iteración Asignada:** Tercera Iteración
Prioridad: Media	**Tiempo Estimado:** 0.13 días
Riesgo en Desarrollo: Ninguno	**Tiempo Real:** 0.25 días
Descripción: Los usuarios pueden seleccionar un algoritmo de planificación de las peticiones al disco duro para simular su ejecución, para lo cual el sistema permite: • La entrada de los datos de las peticiones de acceso al disco para ejecutar el algoritmo. • Realizar las iteraciones, ya sea ciclo completo o una sola iteración. • Mostrar en la correspondiente área de la interfaz el proceso realizado por el algoritmo que se esté simulando.	
Observaciones: 1. Si la lista de datos a procesar está vacía entonces la herramienta no permitirá la ejecución del algoritmo. 2. Si el usuario marca la casilla de ejecutar paso a paso el algoritmo seleccionado entonces la herramienta realizara una sola iteración del algoritmo. 3. Si la casilla esta desmarcada se procederá a realizar el proceso completo del algoritmo. 4. La casilla estará desmarcada (valor por defecto).	

2.3. Modelado de la aplicación.

La herramienta permitirá la simulación del funcionamiento de los algoritmos utilizados en los temas de la asignatura SO, como autor fundamental de referencia se utilizó Andrew S. Tanenbaum para el estudio de

los algoritmos a simular. Para graficar el análisis de los requerimientos y las historias de usuario descritas se utilizan los diagramas de actividades. Aunque XP como metodología no lo define como un artefacto necesario resulta de gran utilidad para el presente trabajo.

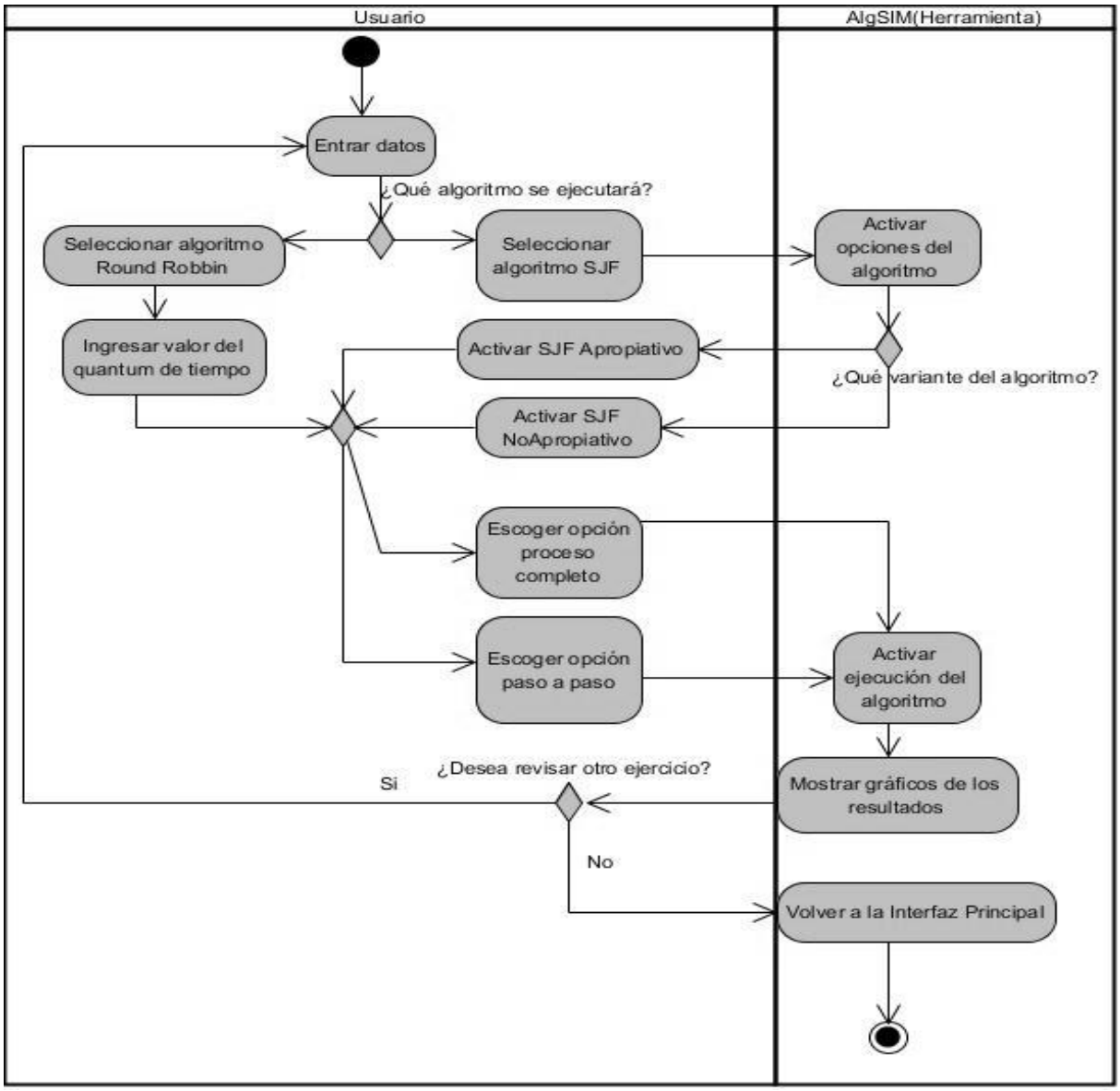

Figura 2: Diagrama de actividades Historia de Usuario 1

En el diagrama de actividades se muestran las acciones que realizan el usuario y la herramienta con el objetivo de efectuar la ejecución de un algoritmo de planificación de procesos que seleccione el usuario y

las posibles variantes del mismo, mostrando finalmente al usuario de forma gráfica el resultado para el ejercicio resuelto por dicho algoritmo.

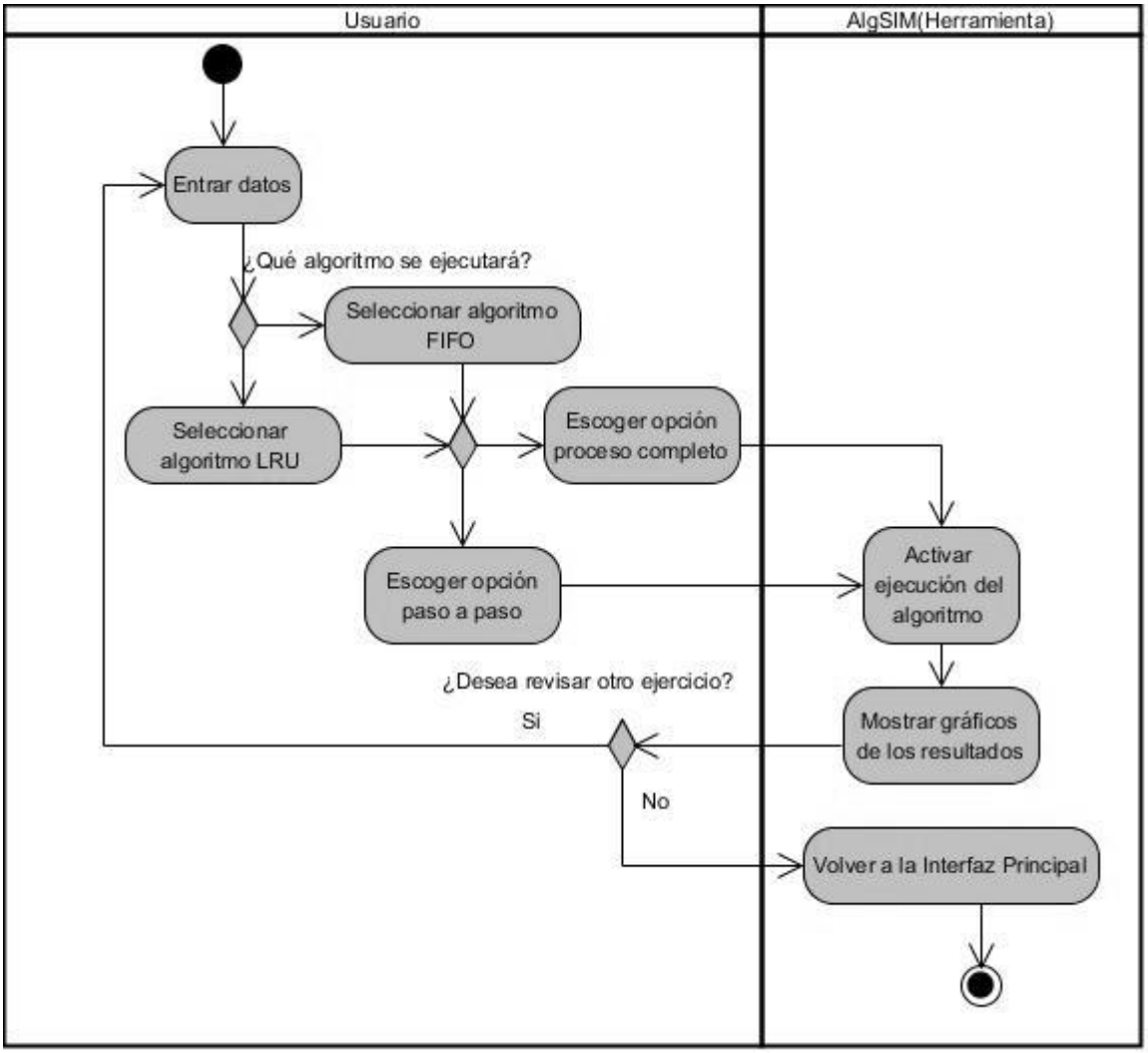

Figura 3: Diagrama de actividades Historia de Usuario 2

En el caso de los ejercicios correspondientes a los temas Planificación de peticiones al disco duro y Reemplazo de páginas, aunque los algoritmos son completamente diferentes las acciones que se realizan son relativamente iguales, por lo tanto, solo se muestra uno de los diagramas de actividades y el referente a la otra historia de usuario se muestra en los anexos (Anexo 1).

2.4. Arquitectura del Sistema.

Para definir la arquitectura es necesario determinar los elementos fundamentales del sistema, las clases, módulos u objetos principales y sus relaciones. En el ciclo de vida de la metodología XP para llevar a cabo el diseño del sistema durante la fase que lleva el mismo nombre, se define la arquitectura del sistema y una parte de la misma queda evidenciada por la metáfora. Esta se considera además una práctica que apoya el diseño de la arquitectura y representa una descripción de cómo funciona el programa [18]. Para mantener la filosofía ágil es importante el desarrollo solo de los diagramas necesarios dando seguimiento así al principio de "viajar ligero", logrando un diseño simple [26]. Para la descripción de la arquitectura de la herramienta propuesta el autor de la investigación propone describirla primeramente a través de la metáfora tal como plantea la metodología XP.

2.4.1. Metáfora

La metáfora es una historia que permite a todos en el equipo, incluido el cliente, explicar con las mismas palabras como funcionará el sistema. La metáfora del sistema propuesto sería: una herramienta que gestiona la simulación de los algoritmos estudiados en clases de sistemas operativos y además permita a los estudiantes la resolución de ejercicios clásicos propuestos por el profesor y la representación gráfica de la solución de los mismos.

Para especificar la metáfora del sistema propuesto se analizó la descripción de la solución, con el objetivo de lograr una visión concreta sobre el funcionamiento de la aplicación, donde los algoritmos que se simulan pueden verse además como un conjunto de “filtros” o “procesos”, evidencian un comportamiento similar a un flujo. Los datos ingresan en el sistema y fluyen entre los componentes (procesos del algoritmo), de uno en uno, hasta que se le asigne un destino final (salida). Teniendo esto en cuenta se considera como modelo arquitectónico el Flujo de Datos (*Dataflow*).

2.4.2. Vista arquitectónica

El tópico más urgente y exitoso en arquitectura de software en los últimos años es, sin duda, el de los patrones (*patterns*), tanto en lo que concierne a los patrones de diseño como a los de arquitectura. Inmediatamente después, en una relación a veces de complementariedad, otras de oposición, se encuentra la sistematización de los llamados estilos arquitectónicos [27].

Teniendo en cuenta que el sistema que se desea desarrollar posee un funcionamiento basado en flujo de datos, solamente se estudiará la vista arquitectónica relacionada con esta característica.

- ***Dataflow* o Flujo de datos**: se basa en un patrón tuberías y filtros. Este consta de un conjunto de componentes denominados “filtros” conectados entre sí por “tuberías” que transmiten los datos desde un componente al siguiente. Cada filtro trabaja de manera independiente de los componentes que se encuentren situados antes o después de ella. Se diseñan de tal modo que esperan que un conjunto de datos en un determinado formato. Y obtiene como resultado datos de salida en un formato específico [28].

Debido a que los estilos arquitectónicos que se utilizan en la vista arquitectónica *Dataflow* son *Pipeline and pipes* o Tuberías – filtros y *Batch sequential* o secuencial por lotes, se hace necesario destacar algunas de sus características.

- **Tuberías y filtros**: una tubería (*pipeline*) es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (*pipes*), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. El sistema tubería-filtros se percibe como una serie de transformaciones sobre sucesivas piezas de los datos de entrada. Los datos entran al sistema y fluyen a través de los componentes [27]. Ver figura 4.
- **Secuencial por lotes:** los componentes son programas independientes; el supuesto es que cada paso se ejecuta hasta completarse antes que se inicie el paso siguiente. Varios autores sostienen que la variante por lotes es un caso degenerado del estilo, en el cual las tuberías se han vuelto residuales [27].

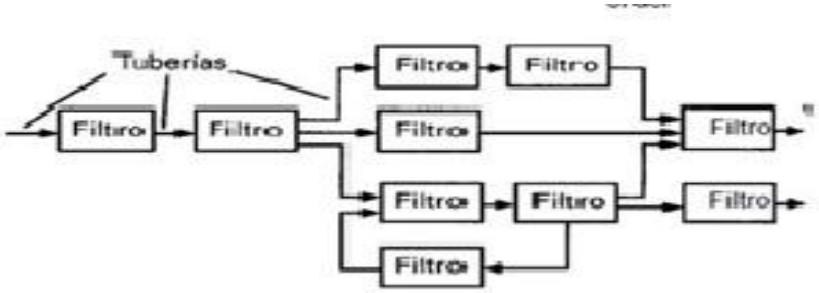

Figura 4 Arquitectura Flujo de datos [25]

En la aplicación se evidencia el uso del patrón Secuencial por lotes porque, específicamente en los módulos, los datos fluyen a través de las diferentes condiciones del algoritmo que se esté simulando, devolviendo una determinada salida: en este caso, la visualización gráfica.

2.4.3. Diseño del software.

Para realizar el diseño del software se parte del estudio de patrones de diseño, los cuales se utilizan para lograr una mejor organización, acoplamiento y funcionamiento del software, este análisis queda evidenciado en el diseño del diagrama de clases. De las familias de patrones de diseño Grasp [29] y Gof [30] se utilizarán algunas propuestas que serán explicadas a continuación.

- **Bajo acoplamiento**: debe haber pocas dependencias entre las clases y consideran las ventajas de la delegación de la herencia. En el diagrama de clases se puede observar que la relación entre las clases es la mínima posible.
- **Alta cohesión**: cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Las clases solamente se encargan de realizar las funcionalidades necesarias para desarrollar el algoritmo deseado.
- **Creador:** se asigna la responsabilidad de que una clase B cree un Objeto de la clase A solamente cuando: B contiene a A, B es una agregación (o composición) de A, B almacena a A, B tiene los datos de inicialización de A (datos que requiere su constructor) y B usa a A. Por ejemplo: tomamos como clase B a InterFProcess y la clase A Proceso. La relación se evidencia puesto que la clase B necesita crear a una instancia de la clase A.

- **Polimorfismo:** cuando identificamos variaciones en un comportamiento, asignar la clase (interfaz) al comportamiento y utilizar polimorfismo para implementar los comportamientos alternativos. En la clase MyRender (encargado del formato de las celdas de las tablas), se muestra este comportamiento puesto que las especificaciones del funcionamiento de la clase varían en dependencia del módulo donde se esté trabajando.

2.4.4. Diagrama de clases.

El diagrama de clases que se muestra a continuación representa la relación entre clases y las responsabilidades de cada una de ellas, evidenciando además los patrones aplicados.

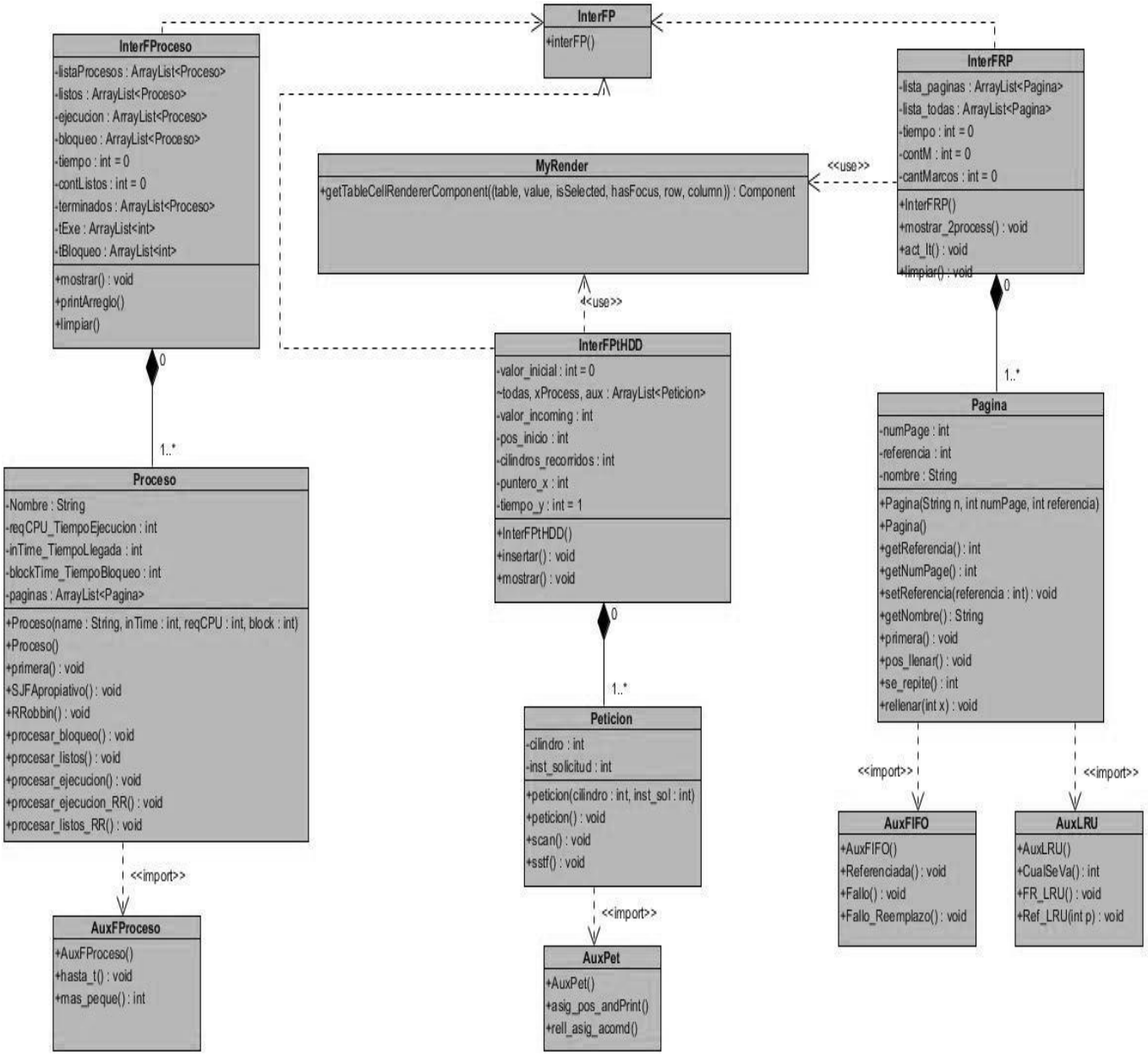

Figura 5: Diagrama de clases del diseño

2.5. Diseño de interfaz

A la hora de realizar el diseño de la aplicación uno de los requisitos es la condición de herramienta de escritorio (*desktop*). Esto se considera un inconveniente puesto que a la hora de diseñar la interfaz de usuario no se cuentan con grandes librerías como las utilizadas en el diseño web. Para resolver esta

situación se procedió a investigar sobre el tema y se consultaron algunos profesionales del diseño. Después de escuchar las diferentes opiniones se consideró tener en cuenta las siguientes sugerencias:

- **Legibilidad:** el texto que aparezca en la interfaz debería tener un alto contraste, se debe utilizar combinaciones de colores como el texto en negro sobre fondo blanco o amarillo suave. El tamaño de las fuentes tiene que ser lo suficientemente grande como para poder ser leído en monitores estándar. Es importante hacer clara la presentación visual. Para este punto se procedió a trabajar la información agrupada en bloques o paneles, distribuyéndose en forma de cascada para mejorar la usabilidad.
- Semejanza del sistema al mundo real.
- Se debe presentar una ayuda y documentación simples y fáciles de interpretar.
- Control y libertad por parte del usuario.
- Definir los formatos individuales de cada una de las interfaces para lograr uniformidad.

Teniendo en cuenta todas las sugerencias hechas se procedió a realizar diseños previos de la aplicación que se pueden ver en los anexos (Anexo 2). Los mismos se consideraron como satisfactorios, cumpliendo así las expectativas de los clientes.

2.6. Conclusiones parciales

Analizando la fase de planificación y diseño de la herramienta se puede concluir que:

- El sistema se entregará con la aprobación por el cliente sobre la calidad de los requerimientos aprobados.
- Se codificará a partir del diagrama de clases respetando las sugerencias propuestas en el diseño de interfaz y se utilizaran además patrones de diseño establecidos.

Capítulo 3: Evaluación de la herramienta educativa para la simulación de los algoritmos.

Garantizar la calidad del software se ha convertido en una tarea vital en el desarrollo de cualquier aplicación, dado por la necesidad de avalar que el producto cumpla con los requisitos especificados y que no presente errores. Por esta razón es necesario establecer pautas y buenas prácticas en el proceso de implementación y se considera necesario además desarrollar un conjunto de pruebas para determinar el correcto funcionamiento del sistema.

3.1. Implementación

En el proceso de desarrollo de sistemas informáticos, la implementación es la etapa donde efectivamente se programa el software. La implementación es la programación de un determinado algoritmo en un lenguaje específico, es convertir los procesos o algoritmos que se encuentran en pseudocódigo en forma de código de un lenguaje de programación, para realizar esto Java posee estándares de codificación. [31]

3.1.1. Estándares de codificación.

Para realizar la aplicación se tuvo en cuenta todo lo referente al establecimiento de los estándares o convenciones de programación empleados en el desarrollo de software sobre la plataforma *Java*. Este modelo de programación está basado en los estándares recomendados por *Sun Microsystems*, que han sido difundidos y aceptados ampliamente por toda la comunidad *Java*, y que han terminado por consolidarse como un modelo estándar de programación de facto.
Estas normas son muy útiles por muchas razones, entre las que destacan [31]:

- Facilitan el mantenimiento de una aplicación. Dicho mantenimiento constituye el 80% del coste del ciclo de vida de la aplicación.
- Permite que cualquier programador entienda y pueda mantener la aplicación. En muy raras ocasiones una misma aplicación es mantenida por su autor original.
- Los estándares de programación mejoran la legibilidad del código, al mismo tiempo que permiten su compresión rápida.

Entre los estándares utilizados se pueden mencionar [31]:

Organización de ficheros.

Las clases en Java se agrupan en paquetes. Estos paquetes se deben organizar de manera jerárquica. (Ver Anexo 4)

Sentencias de paquete

La primera línea no comentada de un fichero fuente debe ser la sentencia de paquete, que indica el paquete al que pertenece(n) la(s) clase(s) incluida(s) en el fichero fuente.

```
 */
package algsim.PPtHDD;
```

```
 */
package algsim.PPtHDD;
```

Sentencias de importación

Tras la declaración del paquete se incluirán las sentencias de importación de los paquetes necesarios. Esta importación de paquetes obligatorios seguirá el siguiente orden:

1. Paquetes del JDK de *Java*.
2. Paquetes de utilidades no pertenecientes al JDK de *Java*.
3. Paquetes de la aplicación.

```
package algsim.PPtHDD;

import java.awt.Color;
import java.awt.Component;
import java.awt.event.KeyEvent;
import java.util.ArrayList;
import java.util.Collections;
import javax.swing.JOptionPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableCellRenderer;
import algsim.interF;
```

Sangría

Como norma general se establecen cuatro caracteres como unidad de sangría. Los entornos de desarrollo integrado (IDE) más populares, tales como Eclipse o NetBeans, incluyen facilidades para formatear código *Java*.

Longitud de línea

La longitud de línea no debe superar los 80 caracteres por motivos de visualización e impresión. (Ver Anexo 4)

División de líneas

Cuando una expresión ocupe más de una línea, esta se podrá romper o dividir en función de los siguientes criterios:

- Tras una coma.
- Antes de un operador.
- Se recomienda las rupturas de nivel superior a las de nivel inferior.
- Alinear la nueva línea con el inicio de la expresión al mismo nivel que la línea anterior.

Si las reglas anteriores generan código poco comprensible, entonces se establecerán tabulaciones de 8 espacios. (Ver Anexo 4)

Comentarios de implementación

Estos comentarios se utilizan para describir el código ("el cómo"), y en ellos se incluye información relacionada con la implementación, tales como descripción de la función de variables locales, fases lógicas de ejecución de un método, captura de excepciones, entre otros. A continuación, un ejemplo de la implementación del estándar. (Para más ejemplos ver Anexo 4)

```
public void Hasta_t() {
    Collections.sort(listaProcesos);//aqui las ponemos en orden
    for (int i = 0; i < listaProcesos.size(); i++) {
        Proceso p = listaProcesos.get(i);
        if (p.getInTime()== 0) {
            listos.add(p);
            jTable1.setValueAt("P:"+p.getName(), y,0);
            String c = jTable1.getValueAt(y, 0).toString().split(":")[1];
            contListos = i;//esto funciona si las entradas estan en orden.
            y++;
        }
    }
}
```

Figura 6: Comentarios de implementación

Métodos

Los métodos deben ser verbos escritos en minúsculas. Cuando el método esté compuesto por varias palabras cada una de ellas tendrá la primera letra en mayúsculas.

```
public void Hasta_t() {
    Collections.sort(listaProcesos);//aqui las ponemos en orden
    for (int i = 0; i < listaProcesos.size(); i++) {
        Proceso p = listaProcesos.get(i);
        if (p.getInTime()== 0) {
            listos.add(p);
            jTable1.setValueAt("P:"+p.getName(), y,0);
            String c = jTable1.getValueAt(y, 0).toString().split(":")[1];
            contListos = i;//esto funciona si las entradas estan en orden.
            y++;
        }
    }
}

public String Imprimir_Arreglo(Proceso p){
String ret="[";
if(p.gettExe().isEmpty())
    return "";
ret+=String.valueOf(p.getBlockTime_TiempoBloqueo());
    for (int i = 0; i < p.gettBloqueos().size(); i++) {
     ret+="-"+String.valueOf(p.gettBloqueos().get(i));
    }
    ret+="][";
     for (int i = 0; i < p.gettExe().size(); i++) {
     ret+=String.valueOf(p.gettExe().get(i))+".";
    }
     ret+="]";
return ret;
}

public void colorear() {
```

Figura 7: Ejemplo de utilización de estándar de codificación para métodos.

Variables

Las variables se escribirán siempre en minúsculas, en el caso de variables compuestas tendrán la primera letra de cada palabra componente en mayúsculas. Las variables nunca podrán comenzar con el carácter "_" o "$". Los nombres de variables deben ser cortos y sus significados tienen que expresar con suficiente claridad la función que desempeñan en el código. Debe evitarse el uso de nombres de variables con un sólo carácter, excepto para variables temporales.

```
public void mostrar(){
    jTextAreaListos.setText("");
    jTextAreaExe.setText("");
    jTextAreaBlock.setText("");
    jTextAreaFin.setText("");
    for (Proceso listo : listos) {
      jTextAreaListos.append('\n'+"Proceso " + listo.getName() + ": "
                                              + listo.getReqCPU());
    }
    for (Proceso ejecucion1 : ejecucion) {
      jTextAreaExe.append('\n'+"Proceso " + ejecucion1.getName() + ": "
                                           + ejecucion1.getReqCPU());
    }
     for (Proceso bloqueo1 : bloqueo) {
      jTextAreaBlock.append('\n'+"Proceso " + bloqueo1.getName() + ": "
                                  + bloqueo1.getBlockTime_TiempoBloqueo());
    }
     for (Proceso terminado : terminados) {
     jTextAreaFin.append('\n'+"Proceso " + terminado.getName() + ": "
                                          + terminado.getReqCPU() + '\n');
    }

}

public int Mas_Peque(){...14 lines }
```

Figura 8: Uso del estándar de codificación para variables.

3.2. Pruebas de software.

Las pruebas de software son los procesos que permiten verificar la calidad de un producto de software. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa. Básicamente es una fase en el desarrollo de software con el objetivo de probar las aplicaciones construidas [31]. En fin, que las pruebas de software se utilizan para determinar el correcto funcionamiento del sistema, existen varias clasificaciones de pruebas que suelen ser utilizadas de acuerdo a los métodos, los tipos y los niveles.

Método de pruebas [32]

- Pruebas de caja blanca: estas pruebas comprueban los caminos lógicos del software, verificando todos los componentes internos, para determinar si el estado real coincide con el esperado.
- Pruebas de caja negra: estas pruebas se desarrollan sobre la interfaz del software comprobando todas las funcionalidades sin tener en cuenta la estructura interna del sistema.

Estrategias de pruebas [32]

- Pruebas unitarias: aseguran que un determinado módulo cumpla con un comportamiento esperado en forma aislada antes de ser integrado al sistema. Se realizan cuando: la interfaz de un método no es clara, la implementación es complicada, para testear entradas y condiciones inusuales, luego de modificar algo. Éstas deben contemplar cada módulo del sistema que pueda generar fallas. Para poder integrar el código realizado al ya existente, el mismo debe aprobar satisfactoriamente todos los casos de prueba definidos.
- Pruebas de funcionalidad: estas pruebas se realizan con el objetivo de verificar el cumplimiento de los requisitos funcionales.

En esta etapa la investigación se centró en las pruebas funcionales y del sistema con el método de caja negra. Aunque se utilizaron inspecciones al código resolviendo cualquier problema que pueda surgir durante el proceso de implementación (*refactoring*), lo cual es una de las prácticas que establece la metodología XP. Hasta cierto punto estas conceptualmente pudieran llegar a considerarse pruebas de caja blanca, aunque no se generaron diseños de las mismas.

3.3. Pruebas de aceptación.

Son realizadas por el cliente y son básicamente pruebas funcionales sobre el sistema completo o alguna parte, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcionad pactada previamente con el cliente [32].

3.3.1. Diseño de los casos de pruebas de aceptación.

En la metodología utilizada se realizan pruebas de aceptación, estas son las pruebas finales antes del despliegue del sistema. El objetivo final es garantizar que los requerimientos hayan sido cumplidos y que el sistema sea aceptable. Se pretende demostrar que:

- Las funciones del software son operativas.
- Las entradas se aceptan de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Tabla 5: Caso de prueba para la Historia de Usuario 3

Caso de Prueba		
Código Caso de Prueba: CDPHU3	**Nombre Historia de Usuario:** Ejecutar simulación del algoritmo de planificación de peticiones al disco duro, seleccionado por el usuario.	
Nombre de la persona que realiza la prueba: Mónica María Albo Castro.		
Descripción de la Prueba: Funcional.		
Condiciones de Ejecución: La entrada de los datos de las peticiones, provistos por un ejercicio clásico proporcionado por profesores de la asignatura para ejecutar el algoritmo.		
Entrada / Pasos de ejecución: El usuario ejecuta el algoritmo seleccionado. El sistema permite seleccionar un “Tipo de Ejecución” en el marco “Ejecutar” donde se pueden encontrar las casillas correspondientes a cada opción disponible.		
Escenarios:	**Resultados Esperados**	**Evaluación de la Prueba:**

EC1. Se ejecuta el algoritmo con la opción “Paso a paso” seleccionada.	La aplicación muestra en el área gráfica el resultado del algoritmo después de realizada la iteración.	Satisfactoria.
EC2. Se ejecuta el algoritmo con la opción “Ejecución Completa” seleccionada.	La aplicación muestra en el área gráfica el resultado final del algoritmo.	Satisfactoria.

Tabla 6 Caso de prueba para la Historia de Usuario 1

Caso de Prueba		
Código Caso de Prueba: CDPHU1	**Nombre Historia de Usuario:** Ejecutar simulación del algoritmo de planificación de procesos, seleccionado por el usuario.	
Nombre de la persona que realiza la prueba: Mónica María Albo Castro.		
Descripción de la Prueba: Funcional.		
Condiciones de Ejecución La entrada de los datos de los procesos, provistos por un ejercicio clásico proporcionado por profesores de la asignatura para ejecutar el algoritmo.		
Entrada / Pasos de ejecución: El usuario ejecuta el algoritmo seleccionado. El sistema permite seleccionar un “Tipo de Ejecución” en el marco “Ejecutar” donde se pueden encontrar las casillas correspondientes a cada opción disponible.		
Escenarios:	**Resultados Esperados**	**Evaluación de la Prueba:**
EC1. Se ejecuta el algoritmo con la opción “Paso a paso” seleccionada.	La aplicación muestra en el área gráfica el resultado del algoritmo después de realizada la iteración.	Satisfactoria.
EC2. Se ejecuta el algoritmo con la opción “Ejecución Completa” seleccionada.	La aplicación muestra en el área gráfica el resultado final del algoritmo.	Satisfactoria.

Tabla 7 Caso de prueba para la Historia de Usuario 2

Caso de Prueba	
Código Caso de Prueba: CDPHU2	**Nombre Historia de Usuario:** Ejecutar simulación del algoritmo de Reemplazo de Páginas, seleccionado por el usuario.
Nombre de la persona que realiza la prueba: Raudel González Echenique	
Descripción de la Prueba: Funcional.	

Condiciones de Ejecución: La entrada de los datos de las páginas provistas por un ejercicio clásico proporcionado por profesores de la asignatura para ejecutar el algoritmo.		
Entrada / Pasos de ejecución: El usuario ejecuta el algoritmo seleccionado. El sistema permite seleccionar un "Tipo de Ejecución" en el marco "Ejecutar" donde se pueden encontrar las casillas correspondientes a cada opción disponible.		
Escenarios:	**Resultados Esperados**	**Evaluación de la Prueba:**
EC1. Se ejecuta el algoritmo con la opción "Paso a paso" seleccionada.	La aplicación muestra en el área gráfica el resultado del algoritmo después de realizada la iteración.	Satisfactoria.
EC2. Se ejecuta el algoritmo con la opción "Ejecución Completa" seleccionada.	La aplicación muestra en el área gráfica el resultado final del algoritmo.	Satisfactoria.
EC3. Se ejecuta el algoritmo con la opción "Detenerse en", se ejecutará hasta llegar a la iteración seleccionada.	La aplicación muestra en el área gráfica el resultado del algoritmo hasta la iteración deseada.	Satisfactoria.

3.3.2. Resultados de la ejecución de las pruebas unitarias y las pruebas de aceptación.

A continuación, se muestran los resultados obtenidos al ejecutar los escenarios de pruebas utilizando los juegos de datos que fueron diseñados acorde a ejemplos de ejercicios clásicos de la asignatura.

Juego de datos.

Para desarrollar los juegos de datos con los que se realizarán los casos de prueba se tuvieron en cuenta las siguientes condiciones:

- Datos fuera de fronteras: se refiere a aquellos datos los cuales se encuentran por fuera de los límites establecidos por el cliente. Para este ejemplo de dato el resultado correcto sería un mensaje de error.
- Datos en frontera: se refiere a aquellos datos los cuales se encuentran en los límites establecidos por el cliente. Para este ejemplo de dato el resultado correcto sería un mensaje de error puesto que los límites no se admiten.

- Datos dentro de fronteras: se refiere a aquellos datos los cuales se encuentran en el rango permisible establecido por el cliente. Para este ejemplo de dato el resultado válido sería la ejecución correcta del algoritmo.

A continuación, propuestas de ejercicios los cuales incluyen las condiciones anteriores. Cada ejercicio se enfoca en un determinado tema de la asignatura. Incluyen además datos correspondientes a cada una de las definiciones anteriores.

Tema 1: Planificación de procesos.

A partir de la siguiente lista de procesos, los cuales poseen el formato Proceso: nombre (instante de llegada, requerimiento de CPU) [tiempos de bloqueos] [tiempos de ejecución].

A (0, 3); E (0, 348); B (1,2) [1] [1]; C (0, 4) [1,2] [1,1]; D (2, 1); F (0, 0)

1. Si se utiliza el algoritmo Round Robbin, diga:
    a) ¿Cómo quedarían las listas (listos, ejecución, bloqueo) al final de la primera iteración?
    b) Realice el proceso completo y obtenga el diagrama de Gantt.
2. Utilizando los datos anteriores, realice una iteración del algoritmo SJF.
    a) Obtenga el diagrama de Gantt correspondiente a la ejecución completa del algoritmo anterior.

Tema 2: Gestión de memoria.

A partir de la siguiente lista de peticiones de un determinado proceso, las cuales poseen el formato Petición: nombre (número de página) o nombre (dirección lógica, tamaño de página).

A (3); E (0, 348); B (2048,1024); C (0); D (1); A (3)

1. Si se utiliza el algoritmo FIFO y la cantidad de marcos disponibles es 3, diga:
    c) ¿Estados de los marcos al final de la primera iteración?
    d) Realice el proceso completo y obtenga el diagrama de Gantt.

2. Utilizando los datos anteriores, realice una iteración del algoritmo LRU.

    b) Obtenga el diagrama de Gantt correspondiente a la ejecución completa del algoritmo anterior.

Tema 3: Planificación del disco duro.

A partir de la siguiente lista de peticiones a los cilindros, las cuales entran en el instante de tiempo 0 y poseen el formato Petición: (tiempo, cilindro), se sabe además que el tiempo de desplazamiento es 2 y la suma del tiempo de latencia más transferencia es 4. La posición del puntero está en el cilindro 75 y el disco duro posee 100 cilindros

(0, 20); (0, 80); (0, 1024); (0, 99); (0, 100); (0, 30); (0, 40); (0, 50)

1. Si se utiliza el algoritmo SCAN moviéndose en sentido ascendente diga:

    e) ¿Posición del puntero, cantidad de cilindros recorridos y los tiempos correspondientes al final de la primera iteración?

    f) Realice el proceso completo y obtenga la representación gráfica del algoritmo además de los valores antes mencionados.

2. Utilizando los datos anteriores, realice una iteración del algoritmo SSTF.

    c) Obtenga el gráfico y los valores correspondientes a la ejecución completa del algoritmo anterior.

En la siguiente figura (Figura 9) se muestra una captura de la herramienta donde, a partir de los datos introducidos por el usuario, se puede apreciar la ejecución del algoritmo FIFO en el modo de ejecución “Proceso completo”. La imagen presentada representa la interfaz correspondiente al tema Reemplazo de Páginas y permite la simulación de los algoritmos FIFO y LRU, pudiéndose ejecutar de manera completa o paso a paso. Permite además la ejecución hasta cierta iteración, lo cual permite a los estudiantes realizar la simulación hasta donde tengan dudas y después realizarla paso a paso. Posee además un área gráfica donde muestra la ejecución del algoritmo.

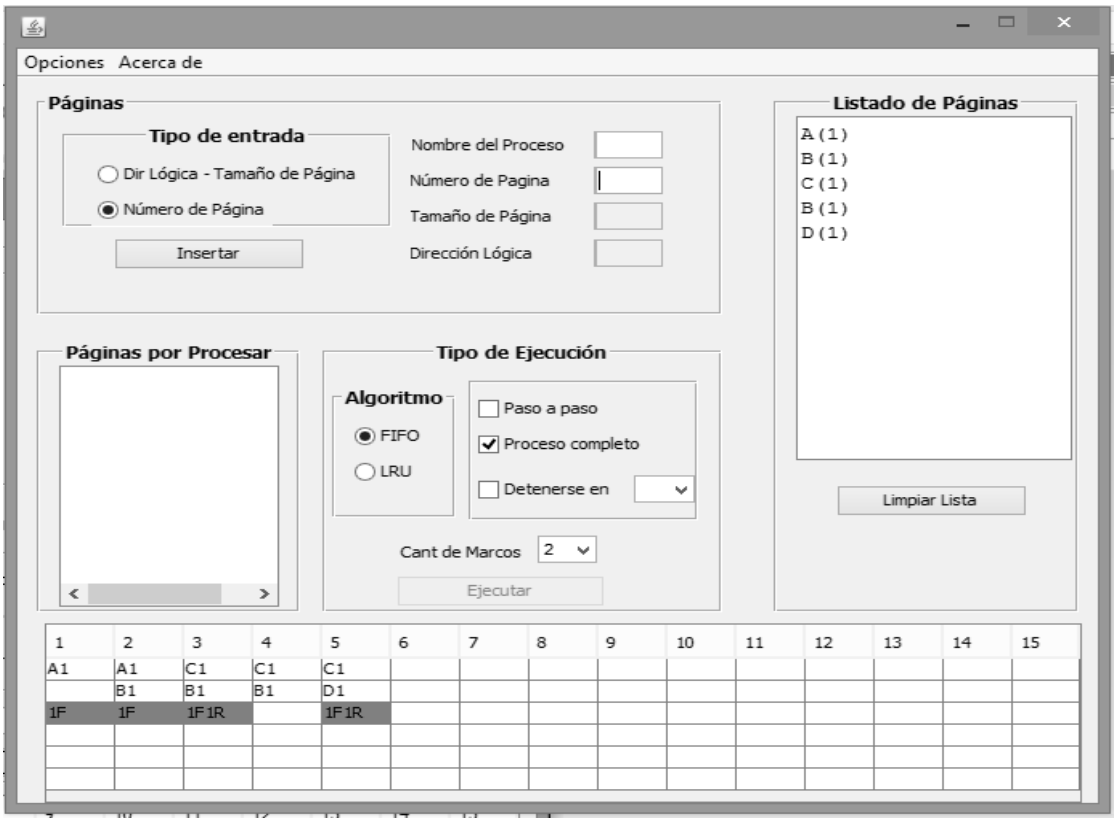

Figura 9 Algoritmo ejecutado en modo Proceso completo

En la figura 10 se muestra una captura de la herramienta, específicamente la interfaz correspondiente al tema Planificación del disco duro. A partir de los datos introducidos por el usuario, se puede apreciar la ejecución del algoritmo SSTF en el modo de ejecución “Ejecución completa”. La interfaz permite la simulación de los algoritmos SSTF y SCAN, pudiéndose ejecutar de manera completa o paso a paso. Posee además un área gráfica donde muestra la ejecución del algoritmo y los valores de las variables en cada instante de tiempo.

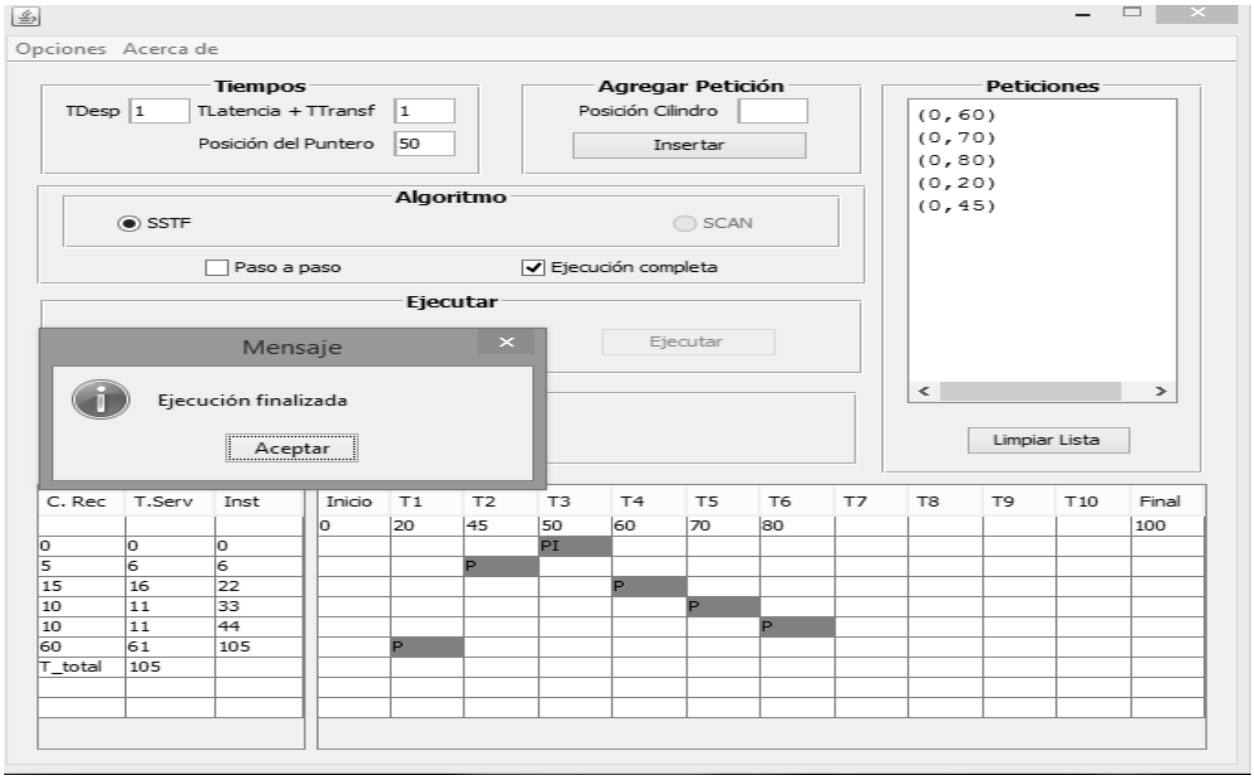

Figura 10 Ejecución del algoritmo SSTF en modo Ejecución completa

Resultados de las pruebas funcionales

Para la validación de los requisitos funcionales se realizaron 3 iteraciones. En la Figura 11 se muestran los resultados obtenidos en cada una de las iteraciones de pruebas realizadas a la herramienta AlgSIM.

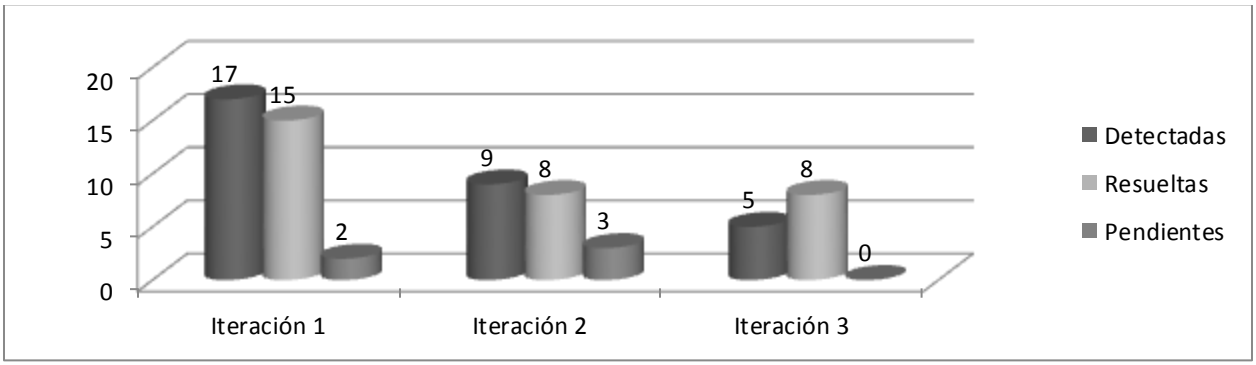

Figura 11: Comportamiento de no conformidades por iteración.

Algunas de las no conformidades encontradas durante el proceso de prueba fueron las siguientes:

- El sistema muestra mensajes de error con datos sobre las variables.
- La graficación no cumple con los estándares establecidos por el cliente.
- Existen errores de validación de datos.

3.4. Conclusiones parciales

La definición de los estándares de codificación permitió adoptar una estructura homogénea que facilita la comunicación y asegura la calidad, menos errores y fácil mantenimiento en la confección del módulo.

Las pruebas realizadas al sistema permitieron corregir errores y fallos en el funcionamiento de la aplicación, así como validar el correcto funcionamiento del sistema, lo que permite asegurar que la propuesta de solución es estable y segura.

CONCLUSIONES GENERALES

A partir del estudio de las herramientas educativas existentes en la universidad se evidenció que su uso en el proceso de enseñanza – aprendizaje es poco factible. Dichas herramientas no cumplen con las necesidades reales de los estudiantes, además de que fueron desarrolladas para un entorno en específico.

Se investigó sobre las posibles herramientas y metodologías para el desarrollo de la aplicación seleccionándose el lenguaje de desarrollo Java con el IDE de desarrollo NetBeans 8.0.1, y como metodología de desarrollo se seleccionó XP, el lenguaje de modelado seleccionado fue UML 3.0 junto con la herramienta Visual Paradigm 8 para la valorización y modificación de los diagramas propuestos por la analista.

Las tecnologías y herramientas usadas durante el desarrollo permitieron obtener una solución informática viable al problema planteado, obteniendo la documentación necesaria para garantizar el futuro mantenimiento del sistema. La herramienta se encuentra lista para la simulación de los algoritmos deseados y su correspondiente resultado gráfico, mejorando así la comprensión del contenido que se esté estudiando.

Para validar el correcto funcionamiento del código fuente se utilizó la técnica de *Refactoring* propuesta por XP. Se realizaron las pruebas funcionales o de caja negra donde se definen los casos de prueba para cada historia de usuario. Además, se realizaron pruebas de aceptación y usabilidad. Dichas pruebas arrojaron los resultados esperados, por lo que se puede concluir que los objetivos planteados se cumplieron satisfactoriamente.

Recomendaciones

Para el posterior mantenimiento y actualización de la herramienta se considera plantear las siguientes recomendaciones.

- Incluir nuevos algoritmos a los temas en próximas versiones de la aplicación.
- Posibilitar como nueva funcionalidad la exportación del área gráfica a un archivo externo (imagen).
- Incluir una funcionalidad que permita cargar los datos del ejercicio desde un fichero externo.

Referencias bibliográficas

1. TANENBAUM, Andrew S.; WOODHULL, Albert S. Sistemas Operativos. Diseño e Implementación. [ed.] Pablo Eduardo Roig Vázquez. Segunda Edición.

2. Ministerio de Educación Superior. Plan de estudio. [En línea] 2014. [Citado el: 10 de febrero de 2016.] http://intranet2.uci.cu/sites/default/files/pdf_formacion/Modelo_del_Profesional.pdf.

3. García, E Materiales Educativos Digitales. *Universia.* [En línea] 3 de 02 de 2010. [Citado el: 20 de febrero de 2016.] http://formacion.universiablogs.net/2010/02/03/materiales-educativos-digitales.

4. Marqués, Pere. El software educativo. *Universidad Autónoma de Barcelona.* [En línea] http://www.dirinfo.unsl.edu.ar/profesorado/INfyEduc/teorias/clasif_software_educativo_de_pere.pdf.

5. SILBERSCHATZ, Abraham GAGNE, et al. *Fundamentos de sistemas operativos*. McGraw-Hill, 2006.STALIN, William. Sistemas Operativos.

6. Sanchez, Isabel Blanco**.** Recursos didácticos para fortalecer la enseñanza-aprendizaje de la economía. [En línea] junio de 2012. [Citado el: 5 de marzo de 2016.]

7. García, A. & González, L. (s.f). Uso pedagógico de los Recursos Educativos. *Universidad de Salamanca* [En línea] [Citado el: 9 de marzo de 2016.] http://www.eyg-fere.com/TICC/archivos_ticc/AnayLuis.pdf

8. PCC**.** *Manifiesto del V Congreso del Partido Comunista de Cuba.* La Habana: Gaceta Oficial de Cuba.

9. CODINA, Lluís. Evaluación de recursos digitales en línea: conceptos, indicadores y métodos. *Revista española de documentación científica*, 2000, vol. 23, no 1, p. 9-44.

10. Sistemas Operativos 2015 -2016. UNIVERSIDAD DE ALICANTE. *Universidad de Alicante [: Universitat d'Alacant*. Universidad de Alicante, 2000.

11. Ingeniería Informática. Sistemas Operativos. (Grado en Ingeniería Informática-Ingeniería de Computadores). *Universidad de Sevilla* [En línea] [Citado el: 12 de marzo de 2016.] http://www.us.es/estudios/grados/plan_204/asignatura_2040013.

12. Instituto Tecnológico Superior nuevo Casas Grandes. ingeniería-en-sistemas-computacionales. Tecnológico Nacional de México. [En línea] [Citado el: 14 de marzo de 2016.] http://www.itsncg.edu.mx/carreras/ingenieria-en-sistemas-computacionales.

13. GONZÁLEZ SÁNCHEZ, Yamilka. Simulador de algoritmos de planificación de procesos para la asignatura Sistemas Operativos, junio 2008. [En línea]. http://catalogoenlinea.uci.cu/cgi-bin/koha/opac-detail.pl?biblionumber=6452.

14. VIDAL FERNÁNDEZ, Santiago. Implementación del Simulador del Tema de Gestión de Dispositivos de Entrada y Salida para el Laboratorio Virtual de la asignatura Sistemas Operativos, 2011. [En línea]. http://repositorioinstitucional.uci.cu/jspui/bitstream/ident/TD_04263_11/1/TD_04263_11.pdf.

15. ESTRADA GRANDALES, Yunet. Desarrollo del evaluador de memoria para el laboratorio virtual de Sistemas Operativos [En línea]. http://repositorioinstitucional.uci.cu/

16. HEREDIA RUIZ, JAVIER; et al. Comparación y tendencias entre metodologías ágiles y formales. Metodología utilizada en el Centro de Informatización para la Gestión de Entidades, 2011. [En línea]. http://publicaciones.uci.cu/index.php/SC/article/view/484/469.

17. LINDSTROM, Lowell; JEFFRIES, Ron. Extreme programming and agile software development methodologies. 2004, vol. 21, no 3, p. 41-52. [En línea]. [Consultado el 25 de marzo de 2016]. Disponible en: http://www.oracle.com.

18. Python software foundation, septiembre 2009. [En línea]. Disponible en: http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf.

19. NetBeans, 2015. [En línea]. Disponible en: http://www.netbeans.org/index_es.html.

20. [En línea] 2007. Disponible en: http://www.uv.es/~jgutierr/MySQL_Java/TutorialEclipse.pdf.

21. Free Download Manager, marzo 2007. [En línea]. [Consultado el 10 de febrero del 2008]. Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

22. Visual Paradigm, 2008. [En línea]. [Consultado el 27 de marzo del 2016]. Disponible en:

http://www.visualparadigm.com.

23. COHN, Mike. User Stories Applied, 2004. Addison Wesley, ISBN 0-321-20568-5.

24. MALBO CASTRO, Mónica. Sistema para automatizar la generación de paquetes binarios de la distribución Nova. Tesis de pregrado. Universidad de las Ciencias Informáticas. La Habana, 2008.

25. REYNOSO, Carlos; KICILLOF, Nicolás. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea]. [Consultado el 10 de abril del 2016]. Disponible en: http://carlorreynoso.com.ar.

26. Diseño de software. [En línea]. Arquitectura de diseño, 2015. [Consultado el 10 de abril del 2016]. Disponible en: https://pnfioem1.wordpress.com/category/unidad-ii/.

27. CANALES MORA, Roberto. Patrones GRASP. [En línea] Adictos al trabajo, 2007. [Consultado el 11 de abril del 2016]. Disponible en: https://www.adictosaltrabajo.com/tutoriales/grasp/.

28. Mundo informático. [En línea] Archivo de la categoría: Patrones de diseño (GoF). [Consultado el 11 de abril del 2016]. Disponible en: https://infow.wordpress.com/category/patrones-de-disenogof/.

29. JavaFoundations. [En línea] 2 de 7 de 2010. [Citado el: 12 de abril de 2016.]http://javafoundations.blogspot.com/2010/07/java-estandares-de-programacion.html.

30. CHAVIANO JIMÉNEZ, Danis Carlos. Herramienta informática para el monitoreo de errores de las aplicaciones web en el centro Fortes. Tesis de grado. La Habana: Universidad de las Ciencias Informáticas, 2014.

31. FERNÁNDEZ PONS, Yanet. [En línea]. Pruebas de aceptación del cliente. [Consultado el 8 de mayo del 2016].

Anexos

Anexo 1: Diagrama de actividades Historia de Usuario 3

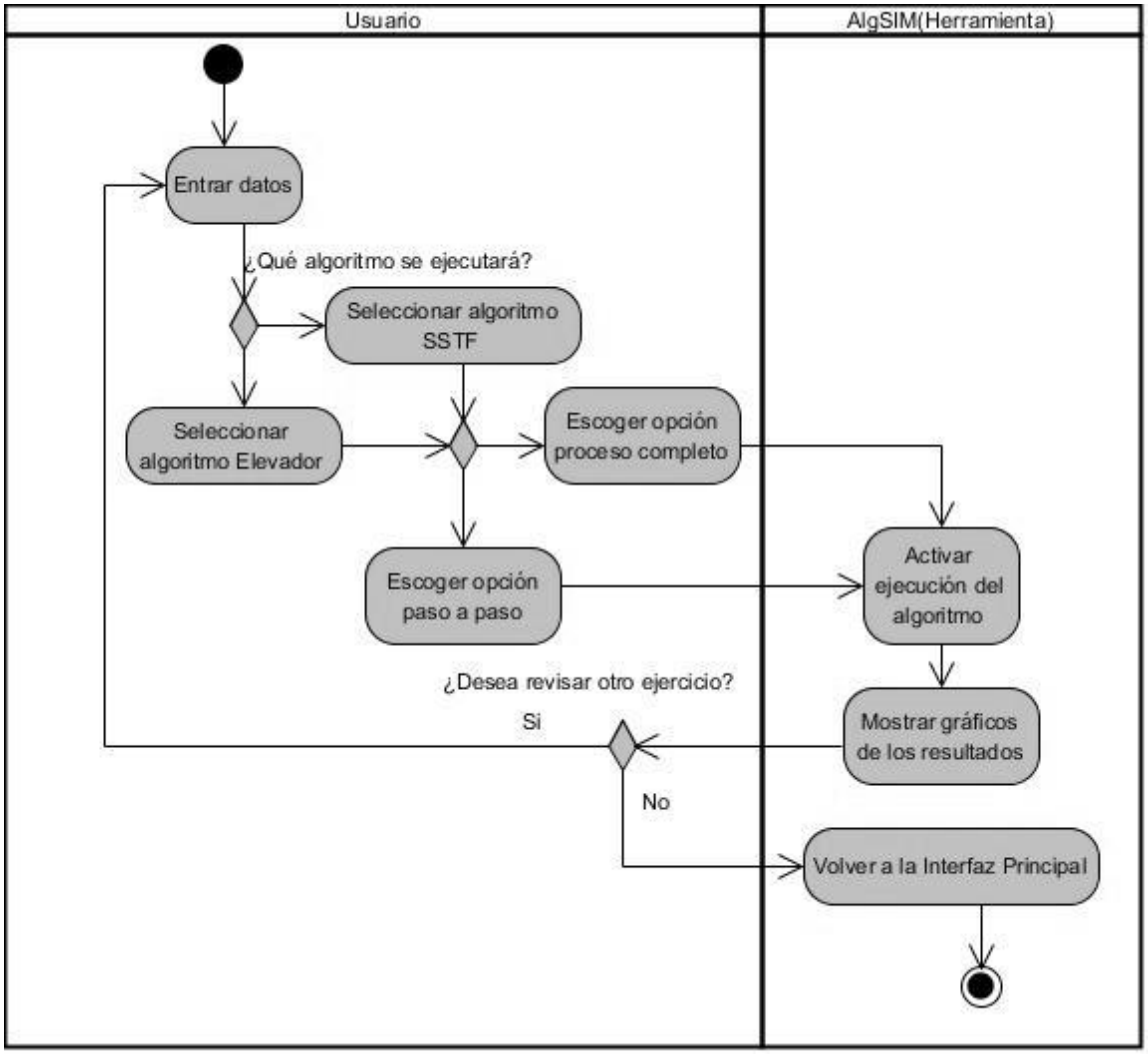

Figura 11: Diagrama de actividades Historia de usuario 3

Anexo 2: Propuestas de interfaces

Figura 12: Propuesta de interfaz Gestión de memoria

Anexo 3: Entrevista

- **Profesores (2 encuestados):**

  - ¿Cuáles son las principales dificultades que han presentado en las clases, a la hora de mostrar a los estudiantes un determinado algoritmo?:

    1. Presentarles el funcionamiento del algoritmo en clase mediante presentaciones de *Power Point* ocasiona que a veces los estudiantes no comprenden la idea que se les quiere hacer llegar. (100%)

    2. Preparar una diapositiva animada que muestre el funcionamiento del algoritmo en ocasiones resulta engorroso y no queda con la calidad deseada. (50%)

  - ¿Qué métodos emplean para mostrar a los estudiantes, de forma gráfica, el funcionamiento de un algoritmo que se esté impartiendo en clases?

    1. Presentaciones de *Power Point.* (100%)

  - ¿Cuáles serían los principales inconvenientes que muestran los estudiantes en el proceso de enseñanza – aprendizaje, específicamente en la parte donde se estudian los diferentes algoritmos de gestión?

    1. No aclaran todas las dudas con respecto a los ejercicios dejados de estudio independiente. (50%)

    2. En ocasiones la explicación no es asimilada por el estudiante y se debe volver a explicarle varias veces. (100%)

- **Estudiantes: (20 encuestados de una matrícula de 67)**

  - ¿Cuáles son las principales dificultades que han tenido en las conferencias, cada vez que se les imparte un nuevo algoritmo de gestión?

    1. A veces no se entiende lo que el profesor explica en la diapositiva. (80%)

2. Cuando se le pide al profesor que vuelva a explicar se pierde mucho tiempo en reiniciar la diapositiva que contiene la representación del algoritmo. (65%)
3. Cuando se explica en la pizarra a veces no queda muy clara la idea. (30%)

- ¿Qué métodos emplean los profesores para, de forma gráfica, representarles el funcionamiento de un algoritmo?
    1. Presentaciones de *Power Point*. (100%)
    2. Dibujos en la pizarra. (40%)
- ¿Cuáles serían los principales inconvenientes que muestran a la hora de resolver los trabajos independientes relacionados con un algoritmo?
    1. No se hace la tarea. (35%)
    2. Si existe alguna duda con la respuesta solo se puede comprobar con el profesor al otro día o con un estudiante aventajado, lo que ocasiona una pérdida de tiempo. (65%)
- ¿Qué métodos o herramientas poseen para confirmar los resultados obtenidos de los ejercicios independientes?
    1. No conozco ninguna. (100%)
    2. Consultar con compañeros o el profesor. (90%)
    3. Volver a revisar la resolución del ejercicio. (25%)
- ¿Qué recomendaciones considerarían útiles con respecto al proceso de enseñanza - aprendizaje?
    1. Hacer más dinámicas las presentaciones del funcionamiento del algoritmo. (25%)
    2. Crear algún medio para comprobar las respuestas de los ejercicios. (65%)
    3. Ninguna recomendación. (35%)

Anexo 4: Estándares de codificación

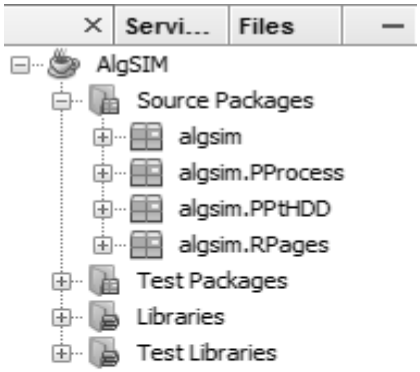

Figura 13: Organización de ficheros

```
if (daux < dxp) {
    puntero_x = aux.get(0).getPosicion();
    valor_incoming = aux.get(0).getCilindro();
    time_Serv = (valor_inicial - valor_incoming) * time_desplz
            + time_lat_transf;
    cilindros_recorridos = valor_inicial - valor_incoming +
        hasta100;
    valor_inicial = valor_incoming;
    jTable2.setValueAt("P", tiempo_y, puntero_x);
    aux.remove(0);
} else {
```

Figura 14: Longitud de líneas y División de líneas.

```
public int Mas_Peque(){ // este metodo devuelve el mas pequeño de la lista
if(listos.isEmpty())   // o retorna -1 si la lista esta vacia
    return -1;
int aux = listos.get(0).getReqCPU();
int pos = 0;
    for (int i = 0; i < listos.size(); i++) {
        listos.get(i);
        if(aux> listos.get(i).getReqCPU()){
            pos = i; // se guarda la posición del mas pequeño
            aux =  listos.get(i).getReqCPU();
        }
    }
return pos;
}
```

Figura 15: Comentarios de implementación