Universidad de las Ciencias Informáticas

Facultad 2

Software de resolución de problemas de Programación Lineal para el apoyo a la asignatura de Investigación de Operaciones

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

Autores:

Glenda de la Caridad Arbolaez Valdés

Yunior Luis Velázquez Blanco

Tutores:

Ing. Lester González López

Ing. Victor Alejandro Roque Dominguez

La Habana, 7 junio de 2018

“Año 60 de la Revolución”

Declaración de Autoría:

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

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

Autores:

_____ _____

Glenda de la C Arbolaez Valdés Yunior Luis Velázquez Blanco

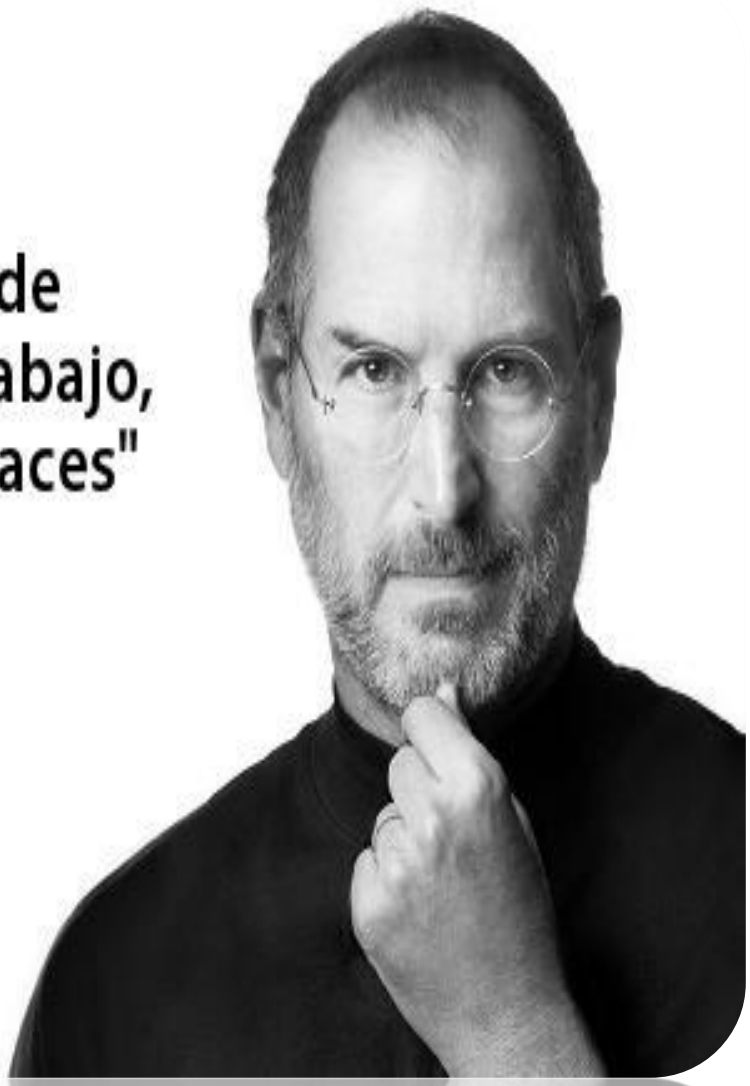
Tutores:

_____ _____

Ing. Lester González López Ing. Victor Alejandro Roque Dominguez

"La única forma de hacer un gran trabajo, es amar lo que haces"

—Steve Jobs



Agradecimientos

A mis padres María y Ramón por ser padres ejemplares, por apoyarme siempre, por alentarme a ser una profesional y por ser los mejores padres que la vida me pudo dar, este título también es suyo.

A mis hermanas Melyn, Mylem, a mi abuelo Jesús, a mi tía Lisbet y mi tío Luís.

A mi novio Yunior por estar a mi lado todos estos años y apoyarme en todo momento. Por ayudarme a salir adelante en difícil camino que es la Universidad. Te amo mucho.

A mis tutores Lester y Víctor, que este logro también es suyo.

A mis amigos Olivia, Carlos, Elena, Ian, Manuel.

A mi mejor amiga en la UCI, Jessie y su mamá Alina.

A la familia de mi novio, especialmente a mis suegros Ana María y Luís.

A mi profesora Madelín, por alentarme a ser una buena profesional. A Yidian por ser un buen amigo y apoyarme siempre.

A mis amigos, Víctor Alexis, José Ernesto, Raidel, Bienvenido, Román, Wilber, Osniel, Yuya, a mi compañero de mesa Yaicel.

A mis amigos de la FEU de la Facultad y de la Universidad, con ellos pasé muy buenos momentos y aprendí muchas cosas para la vida.

A todas mis amistades que se encuentran presentes ahora.

Les agradezco a todos.

Glenda

Agradecimientos

A mi mamá Ana, mi papá Luis por ser los mejores padres del mundo, por darme el don de conocer la vida, por guiarme por el mejor camino, por formar en mí la persona que hoy soy, por sus consejos, por su infinito amor, por ser insustituibles, por su dedicación, por confiar siempre en mí, por estar siempre a mi lado en los momentos que más los necesito, por toda la felicidad que me han dado, por el orgullo que siento de ser su hijo y porque los quiero mucho, mucho, mucho…

A mi novia Glenda por su cariño y ternura, por aconsejarme cuando más lo necesitaba, por ser una persona especial, por darme apoyo en los momentos difíciles, por ser parte de mi conciencia, por los momentos lindos que hemos tenido, por llenar de felicidad mi vida y por el infinito amor que nos une.

A mi hermano Luis Enrique por su confianza y cariño.

A mi tía Miriam, mi prima Nay y mi suegra Mary, por ser mis segundas madres aquí en La Habana.

A mi abuela Gladys y mi abuelo Braudilio, que desde el primer momento me apoyaron y brindaron siempre su cariño.

A mis tutores Lester y Víctor, que este logro también es suyo.

A mis amigos, Víctor Alexis, José Ernesto, Raidel, Bienvenido, Román, Wilber, Osniel, Yuya, Jessie.

Al profe José Nolberto por su ayuda incondicional desde el primer momento.

A todas mis amistades que se encuentran presentes ahora.

Les agradezco a todos.

Yunior Luis

Resumen

El empleo de las Tecnologías de la Información y la Comunicación en la educación ha proporcionado nuevas vías para facilitar el proceso de enseñanza aprendizaje. Dentro de las tecnologías educativas tiene gran importancia el desarrollo de software y herramientas didácticas para educandos y profesores. El modelo del profesional en la Universidad de las Ciencias Informáticas comprende la asignatura de Investigación de Operaciones, en la cual se utiliza la herramienta WinQSB para la resolución de problemas de programación lineal, la misma presenta inconvenientes que dificultan la asimilación del contenido por parte de los estudiantes, además de ser un software propietario compatible solamente con Windows Vista la primera versión y la última versión solamente para arquitecturas de 32bit de este sistema operativo. Por tales razones, el presente trabajo de diploma tiene como objetivo desarrollar un software, que sirva de apoyo al proceso de enseñanza aprendizaje en la resolución de problemas de programación lineal de la asignatura Investigación de Operaciones. Para llevarlo a cabo se utilizaron bibliotecas como JFreeChart y Apache Commons Math, además del lenguaje de programación Java. Con la realización del presente trabajo se demostró que los resultados obtenidos tras la ejecución de la solución son correctos y que puede ser utilizado por los estudiantes y profesores como software de apoyo.

Palabras clave:

Investigación de Operaciones, método simplex, Proceso de Enseñanza-Aprendizaje, programación lineal.

Summary

The use of Information and Communication Technologies in education has provided new ways to facilitate the teaching-learning process. The development of software and teaching tools for students and teachers is of great importance within educational technologies. The professional model at the University of Informatics Sciences includes the subject of Operations Research, in which the WinQSB tool is used to solve linear programming problems, it has drawbacks that make it difficult for students to assimilate content students, in addition to being a proprietary software compatible only with Windows Vista the first version and the latest version only for 32bit architectures of this operating system. For these reasons, the purpose of the present diploma work is to develop software that supports the teaching-learning process in solving linear programming problems of the Operations Research subject. To carry it out, were been used libraries such as JFreeCart and Apache Commons Math, in addition to the Java programming language. It was demostrated with the realization of the present work, that the results obtained after the execution of the solution are correct and that students and teachers can use it as support software.

Keywords:

Linear Programming, Operations Research, simplex method, Teaching-Learning.

ÍNDICE GENERAL

Introducción ..... - 1 -
Capítulo 1: Fundamentación Teórica ..... - 7 -
1.1 Proceso de Enseñanza-Aprendizaje ..... - 7 -
1.2 Investigación de Operaciones ..... - 8 -
1.2.1 Programación Lineal en la Investigación de Operaciones ..... - 8 -
1.3 Empleo de las tecnologías de la información y la comunicación en el proceso de enseñanza-aprendizaje ..... - 17 -
1.4 Análisis de soluciones similares ..... - 19 -
1.4.1 El WinQSB o QSB - Quantitative System Business ..... - 19 -
1.4.2 Solver ..... - 19 -
1.4.3 JSimplex ..... - 20 -
1.4.4 PHPSimplex ..... - 20 -
1.4.5 MATLAB ..... - 20 -
1.4.6 R ..... - 21 -
1.5 Metodologías de desarrollo de software ..... - 23 -
1.5.1 Método de Boehm y Turner ..... - 23 -
1.5.2 Programación Extrema (XP) ..... - 24 -
1.6 Tecnologías y herramientas de desarrollo ..... - 27 -
1.6.1 Lenguajes de programación empleados ..... - 27 -
1.6.2 Bibliotecas empleadas ..... - 28 -
1.6.3 Entorno de desarrollo integrado ..... - 29 -
1.7 Conclusiones parciales ..... - 30 -
Capítulo 2: Características del sistema, exploración y planificación ..... - 31 -
2.1 Descripción de la solución propuesta ..... - 31 -
2.1.1 Licencia ..... - 34 -
2.2 Funcionalidades del software ..... - 35 -
2.3 Lista de Reserva del Producto ..... - 36 -
2.4 Usuarios relacionados con el sistema ..... - 37 -
2.5 Fase de Exploración ..... - 37 -
2.5.1 Historias de usuario (HU) ..... - 37 -
2.6 Fase de Planificación ..... - 41 -
2.6.1 Estimación de esfuerzo ..... - 41 -
2.6.2 Plan de Iteraciones ..... - 42 -
2.6.3 Plan de duración de las iteraciones ..... - 43 -

2.6.4 Plan de entregas ..... - 43 -
2.7 Conclusiones parciales ..... - 43 -
Capítulo 3: Diseño, implementación y pruebas. ..... - 44 -
3.1 Tareas de la Ingeniería ..... - 44 -
3.2 Tarjetas Clase – Responsabilidad - Colaborador ..... - 46 -
3.3 Patrones de diseño ..... - 47 -
3.1.1 Patrones GRASP ..... - 47 -
3.4 Estándares de codificación ..... - 48 -
3.5 Pruebas ..... - 52 -
3.5.1 Pruebas unitarias ..... - 53 -
3.5.2 Pruebas de aceptación ..... - 54 -
3.6 Conclusiones parciales ..... - 55 -
Conclusiones generales ..... - 56 -
Recomendaciones ..... - 57 -
Referencias Bibliográficas ..... - 58 -
Anexos ..... - 62 -
Anexo 4: Preguntas y resultados de las encuestas aplicadas ..... - 62 -
Anexo 5: Comparación del método tabular del simplex ..... - 64 -

ÍNDICE DE FIGURAS

Fig. 1 Algoritmo del método simplex en su forma tabular. Fuente: (10) ..... - 12 -
Fig. 2 Estrella que representa el proyecto según la aplicación de Boehm y Turner ..... - 24 -
Fig. 3 Logo del software ..... - 34 -
Fig. 4 Identificadores ..... - 49 -
Fig. 5 Máxima longitud de líneas ..... - 49 -
Fig. 6 Expresiones y sentencias ..... - 50 -
Fig. 7 Importaciones ..... - 51 -
Fig. 8 Comentarios ..... - 51 -
Fig. 9 Variables y constantes ..... - 52 -
Fig. 10 Alineación y espacios en blanco ..... - 52 -
Fig. 11 Caso de prueba método generarDatos ..... - 53 -
Fig. 12 Pruebas de aceptación ..... - 55 -

ÍNDICE DE TABLAS

Tabla 1 Tipos de restricciones ..... - 9 -
Tabla 2 Tabla inicial del Simplex ..... - 15 -
Tabla 3 Paso antes de la primera iteración del Simplex ..... - 15 -
Tabla 4 Primera iteración del Simplex ..... - 16 -
Tabla 5 Segunda iteración del Simplex ..... - 16 -
Tabla 6 Comparación de herramientas ..... - 22 -
Tabla 7 Modelo aumentado ..... - 33 -
Tabla 8 Usuarios relacionados con el sistema ..... - 37 -
Tabla 9 Historias de Usuario (HU) ..... - 38 -
Tabla 10 HU1: Insertar modelo de programación lineal ..... - 38 -
Tabla 11 HU2: Devolver modelo aumentado, método simplex en forma tabular y SBFI. ..... - 39 -
Tabla 12 HU3: Resolver el problema paso a paso ..... - 39 -
Tabla 13 HU4: Resolver el problema de forma directa ..... - 40 -
Tabla 14 HU5: Graficar problema de 2 variables ..... - 40 -
Tabla 15 HU6: Guardar, cargar y exportar como imagen el proyecto ..... - 41 -
Tabla 16 Estimación de esfuerzo por HU ..... - 42 -
Tabla 17 Plan de duración de iteraciones ..... - 43 -
Tabla 18 Plan de entregas ..... - 43 -
Tabla 19 Tarea: Guardar proyecto en formato .ssfa ..... - 44 -
Tabla 20 Tarea: Cargar proyecto en formato .ssfa ..... - 45 -
Tabla 21 Tarea: Exportar tablas y gráficas como imagen ..... - 46 -
Tabla 22 Clase: Solución ..... - 47 -
Tabla 23 Clase: SimplexSolver ..... - 47 -

Introducción

La educación es un proceso social que surge de lo más profundo de las necesidades sociales, de su estructura y orientación, se concibe no solo como un método para mejorar la calidad de vida de las personas, sino también como la vía más idónea que permite construir las bases de la cultura de la sociedad, en respuesta a cambios educativos y sociales del país. En las condiciones actuales, el proceso de enseñanza-aprendizaje (PEA) exige una formación más independiente del docente, es de gran importancia debido a que hace del auto aprendizaje el centro del proceso de formación y demanda un elevado desarrollo de la capacidad de gestionar sus propios conocimientos a través de los materiales didácticos, entre ellos los softwares educativos.

Entre las principales herramientas de las Tecnologías de la Información y la Comunicación (TIC) se destaca el software educativo. La utilización del software educativo posibilita el cumplimiento de los objetivos que tiene la enseñanza, pues permite poner énfasis en la comprensión teórica y en el desarrollo de capacidades, habilidades y valores, a través de la resolución de problemas, facilitan nuevas formas de relación con el contenido (1).

El desarrollo de software es una de las industrias con mayor crecimiento dentro de las tecnologías de la información y la comunicación. Su evolución ha permitido que en la actualidad se puedan elegir diversas formas de desarrollar, distribuir y mantener el software satisfactoriamente. En Cuba se ha impulsado el desarrollo de la informática, y en particular el desarrollo de la industria de software, razón por la cual fue creada la Universidad de las Ciencias Informáticas (UCI) como un proyecto social de la Revolución.

La Universidad de las Ciencias Informáticas cuenta con un centro de tecnologías para la formación (FORTES) el cual tiene la misión de desarrollar tecnologías que permitan ofrecer servicios y productos para la implementación de soluciones de formación, aplicando las tecnologías de la información y la comunicación. También se encuentra el Centro de Innovación y Calidad de la Educación (CICE), que contribuye a elevar la calidad de la formación en nuestra Universidad, desarrollando y potenciando la innovación y la investigación en el campo de las Ciencias Pedagógicas y de la Educación, con el uso de los medios más modernos de la informática y las tecnologías educativas siendo esta última una línea de investigación en la cual se vinculan muchos profesores de la Universidad, y es donde se realizan numerosas investigaciones en este ámbito.

El uso de las tecnologías de la información y la comunicación ha proporcionado nuevos canales de comunicación que se deben aprovechar en el apoyo del proceso de enseñanza-aprendizaje. La educación

demanda de formas específicas de aprendizaje y de la utilización de las tecnologías de la información y la comunicación como herramientas didácticas que contribuyan a la consecución de aprendizajes significativos por parte de los estudiantes. Resulta necesario transformar la concepción del proceso de enseñanza-aprendizaje, en función de lograr un papel más activo del estudiante como centro del proceso, donde pase a formar un rol protagónico en el desarrollo de sus propios conocimientos, a través de la asimilación y manejo de las habilidades necesarias para ello.

Los programas de las disciplinas del currículo del Ingeniero en Ciencias Informáticas, incluyen objetivos y habilidades a lograr que compulsen la utilización de la computación por parte de los estudiantes. Las mismas se toman como objeto de estudio y vía para mejorar el trabajo de las organizaciones, así como elementos que lo ayudan en su trabajo como informático en particular.

En las diferentes asignaturas de la carrera se deben emplear software de aplicación específicos del perfil de graduado, ya sean paquetes profesionales o desarrollados por el claustro de profesores u otros especialistas, los mismos deben ir perfeccionando el desarrollo del plan de estudios en función de actualizarlos conforme al desarrollo de estas tecnologías. Estos softwares son usados por los estudiantes como apoyo en su labor docente e investigativa acorde a la necesidad educativa y propia de la enseñanza.

El desarrollo de las tecnologías de la información y la comunicación posibilita la aplicación efectiva de la disciplina educativa. En correspondencia con ello, la disciplina proporciona los conocimientos y habilidades necesarias para la utilización, en situaciones prácticas, de paquetes de programas para computadoras que permiten elevar la efectividad del trabajo del ingeniero en ciencias informáticas y lo prepara para crear sus propios productos informáticos que le permitan contribuir a resolver problemas relacionados con la toma de decisiones.

Dentro de las disciplinas del Modelo del Profesional del plan de estudios en la Universidad de las Ciencias Informáticas está la Matemática Aplicada, donde se encuentra la asignatura de Investigación de Operaciones (IO). La asignatura tiene entre sus objetivos generales el de “formular y resolver modelos matemáticos asociados a problemas de Programación Lineal y Programación Discreta, y a problemas de Redes”, siendo la resolución de problemas de programación lineal una de las habilidades a superar por los educandos para la solución de los problemas de la asignatura y de la vida real.

El programa de formación del profesional de la Universidad de las Ciencias Informáticas se ha ido adaptando a las necesidades y exigencias de los cambios en la universidad; en consecuencia, el programa analítico de la asignatura ha sufrido cambios durante los años que lleva vigente la carrera. Para la impartición de la

misma, es necesario que los profesores se auxilien del modelo de planificación y control del proceso docente (P1) de la asignatura, programa analítico y libro de texto (Introducción a la Investigación de Operaciones, Tomos I, II y III. Lieberman. Editorial Félix Varela, La Habana, 2005), como documentos normativos y curriculares; así como también de las tecnologías de la información y la comunicación, como medios de enseñanza, donde su rol es el de facilitador en el proceso de enseñanza-aprendizaje.

El software utilizado en los laboratorios es el WinQSB, el mismo presenta limitaciones y dificulta el proceso de enseñanza-aprendizaje en la asignatura Investigación de Operaciones. Es un software propietario, que funciona exclusivamente en computadoras con sistema operativo Windows, versión de 32bit. Para trabajar con este software en los laboratorios se ha tenido que emular con el software "Wine", lo cual ha traído problemas de compatibilidad con los sistemas operativos libres o de código abierto (GNU/Linux), como interrumpir su funcionamiento en varias ocasiones inesperadamente. Desde el punto de vista metodológico, existen diferencias en el tratamiento de la teoría que se hace en el aula con la forma en que trabaja dicho software, lo cual dificulta la asimilación del contenido por parte de los estudiantes, esto se puedo constatar a partir de las encuestas aplicadas a una muestra de estudiantes donde al 56% les resulta complejo entender el contenido con las diferencias de las tablas, además de que el 76% prefieren que tenga el mismo formato la tabla del método simplex que se da en el aula con la que muestra dicho software (**Ver Anexo 4**). Se muestra una de las diferencias a partir de la tabla de salida del método simplex (**Ver Anexo 5**) donde se representa la iteración 0 en las dos formas que se ha realizado: en forma manual realizada en clases y mediante la tabla mostrada por el software WinQSB.

Tomando en referencia la situación actual surge el **problema de investigación**: ¿Cómo facilitar el proceso de enseñanza-aprendizaje del tema Programación Lineal en la asignatura Investigación de Operaciones haciendo uso de las tecnologías educativas? De lo planteado anteriormente se deriva como **objeto de estudio** el proceso de enseñanza-aprendizaje en la asignatura Investigación de Operaciones con el uso de las tecnologías en las carreras de ingeniería, determinando para la investigación como **objetivo general** desarrollar un software para la resolución de problemas de programación lineal para facilitar al proceso de enseñanza-aprendizaje en la asignatura de Investigación de Operaciones de la Universidad de las Ciencias Informáticas. Enmarcado en el **campo de acción** software educativo para la resolución de problemas de Programación Lineal en la asignatura Investigación de Operaciones de la carrera de Ingeniería en Ciencias Informáticas.

Como guía para alcanzar el objetivo propuesto se formularon las siguientes **preguntas científicas**:

1. ¿Cuáles son los principales fundamentos teóricos-metodológicos para la elaboración de un software de resolución de problemas de programación lineal para el apoyo a la asignatura Investigación de Operaciones?
2. ¿Cuál es el estado inicial de la resolución de problemas de programación lineal?
3. ¿Qué estructura tendrá el software que permitirá la resolución de problemas de programación lineal en la asignatura Investigación de Operaciones?
4. ¿Qué resultado tiene la aplicación práctica del software para la resolución de problemas de programación lineal en la asignatura Investigación de Operaciones?

Para dar respuesta a las preguntas científicas antes planteadas, fueron establecidas las siguientes **tareas de investigación**:

1. Sistematización de los principales fundamentos teóricos-metodológicos para la elaboración de un software de resolución de problemas de programación lineal en la asignatura Investigación de Operaciones.
2. Diagnóstico del estado inicial de la resolución de problemas de programación lineal en la asignatura Investigación de Operaciones.
3. Elaboración del software para la resolución de problemas de programación lineal en la asignatura Investigación de Operaciones.
4. Valoración de la aplicación práctica del software para la resolución de problemas de programación lineal en la asignatura Investigación de Operaciones.

Unidades de estudio y decisión muestral:

Para la realización de las encuestas, de una población de 87 estudiantes que en el curso 2017-2018 recibieron la asignatura IO en la Facultad 2, se selecciona de forma intencional una muestra de 30 estudiantes, que equivale al 34.48% de la población.

Para la realización de la encuesta a los profesores se seleccionaron aquellos que hayan impartido la asignatura, se toma una muestra de 4 profesores.

Métodos e instrumentos utilizados:

Métodos teóricos:

Analítico-sintético: para descomponer el problema de investigación en elementos separados y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta. Además, permitió la argumentación de la investigación y el arribo a conclusiones parciales y generales durante la misma.

Inductivo-deductivo: Se utilizó para la identificación de la problemática y de las soluciones. (Problema de investigación)

Análisis histórico-lógico: para el estudio crítico de los trabajos anteriores sobre software referentes a la resolución de problemas de programación lineal y utilizar estos como punto de referencia y comparación con los resultados alcanzados. Además de emplearse también en determinar los antecedentes y tendencias del proceso de enseñanza-aprendizaje en la Investigación de Operaciones en la UCI.

Métodos empíricos:

Análisis documental: para el estudio de documentos curriculares y normativos; como el modelo del profesional y programas analíticos de la disciplina y la asignatura.

Entrevista: se entrevistaron a profesores para la recopilación de información sobre el problema planteado y para la obtención de las funcionalidades del software, la misma será no estructurada.

Encuesta: se realizaron a profesores y estudiantes, para recoger criterios que permitan fundamentar las necesidades que se plantean en la situación problemática.

Para mejor comprensión de la investigación, se decide definir una estructura capitular que permita cierto grado de organización y facilite el estudio de la investigación. Además de contar con la introducción, posee tres capítulos.

Capítulo 1: "Fundamentación Teórica", se enuncian los referentes teóricos-metodológicos que sustentan la solución del problema, se definen conceptos fundamentales, la caracterización del proceso de enseñanza-aprendizaje y el uso de las tecnologías de la información y la comunicación en la enseñanza de la investigación de operaciones en la Universidad de las Ciencias Informáticas y un estudio de las metodologías, herramientas y tecnologías a utilizar en el proceso de implementación de la solución.

Capítulo 2: "Características del Sistema, Exploración y Planificación", en este capítulo se caracteriza el sistema, se confeccionan las Historias de Usuarios, que son de interés para la primera entrega del producto, así como una propuesta del prototipo no funcional de la aplicación. A partir de la metodología seleccionada para el desarrollo de la aplicación, también se describen las fases por la que está compuesta la misma.

Capítulo 3: "Diseño, Implementación y Prueba", en este capítulo se definen los patrones que se utilizan. Se definen las tareas de ingeniería asociadas a las diferentes Historias de Usuario. Se implementan cada una de las funcionalidades hasta lograr el producto completo y se realizan las respectivas pruebas de aceptación para garantizar su correcto funcionamiento y el cumplimiento con los requisitos definidos por el cliente.

Finalmente se exponen las conclusiones y recomendaciones, la bibliografía que sirvió de soporte y los anexos que complementan la investigación.

Capítulo 1: Fundamentación Teórica

En el presente capítulo se realiza la sistematización de los principales fundamentos teóricos-metodológicos que guían el proceso de la investigación, presentándose además conceptos importantes que se utilizan durante la investigación como son el proceso de enseñanza-aprendizaje, investigación de operaciones y programación lineal. Se hace un análisis de soluciones similares, se escoge la metodología de desarrollo de software, así como las tecnologías y herramientas que se emplean en aplicaciones de escritorio.

1.1 Proceso de Enseñanza-Aprendizaje

Como proceso de enseñanza – aprendizaje (PEA) se define "el movimiento de la actividad cognoscitiva de los alumnos bajo la dirección del maestro, hacia el dominio de los conocimientos, las habilidades, los hábitos y la formación de una concepción científica del mundo". Es el procedimiento mediante el cual se trasmiten conocimientos especiales o generales sobre una materia (2).

Este proceso ha sido históricamente caracterizado de formas diferentes, que van desde la identificación como proceso de enseñanza con un marcado énfasis en el papel central del maestro como transmisor de conocimientos, hasta las concepciones más actuales en la que se concibe el proceso de enseñanza-aprendizaje como un todo integrado en el que se pone de relieve el papel protagónico del estudiante.

El PEA es muy complejo e inciden en su desarrollo una serie de componentes que deben interrelacionarse para que los resultados sean óptimos. Los cuales son:

- Objetivos
- Contenidos
- Formas de organización
- Métodos
- Medios
- Evaluación

Los medios de enseñanza son las herramientas mediadoras del proceso de enseñanza-aprendizaje utilizadas por maestros y alumnos, que contribuyen a la participación activa, tanto individuales como colectivas, sobre el objeto de conocimiento. Los medios permiten la facilitación del proceso a través de objetos reales, sus representaciones e instrumentos, que sirven de apoyo material para la apropiación del contenido, estos complementan al método para la consecución de los objetivos y la solución de problemas de enseñanza aprendizaje. Los medios se conciben en la concepción desarrolladora en correspondencia con la diversidad de contenidos (3).

1.2 Investigación de Operaciones

A continuación, se expone la definición de Investigación de Operaciones, según los diferentes autores:

Según Taha La Investigación de Operaciones aspira a determinar el mejor curso de acción óptimo de un problema de decisión con la restricción de recursos limitados, aplicando técnicas matemáticas para representarlo por medio de un modelo y analizar problemas de decisión (6).

La Investigación de Operaciones según Witenberg es la aplicación por grupos interdisciplinarios de Método Científico a problemas relacionados con el control de las organizaciones o de sistemas en relación al hombre-máquina, con el fin de producir soluciones óptimas para dichas organizaciones (7).

La Investigación de Operaciones según Namakforoosh es la aplicación del Método Científico a los problemas de decisión en las empresas y otras organizaciones, incluyendo el gobierno y la milicia (8).

La Investigación de Operaciones según Moskowitz toma al Método Científico aplicado a la solución de problemas y la toma de decisiones de la gerencia en función a la construcción de un modelo simbólico examinando y analizando entre relaciones que lleguen a una técnica en la toma de decisiones en base a los resultados óptimos (9).

1.2.1 Programación Lineal en la Investigación de Operaciones

La Programación Lineal (PL) utiliza un modelo matemático para describir el problema. Un modelo es una representación simplificada de un sistema de la vida real, de una situación de una realidad. El adjetivo lineal significa que todas las funciones matemáticas del modelo deben ser funciones lineales. En este caso, la palabra programación no se refiere aquí a términos computacionales; en esencia es sinónimo de planeación. Por lo tanto, la Programación Lineal involucra la planeación de actividades para obtener un resultado óptimo; esto es, el resultado que mejor alcance la meta especificada entre todas las alternativas factibles[1]. Este problema consiste en elegir el nivel de ciertas actividades que compiten por recursos escasos, necesarios para realizarlas. Después, los niveles de actividad que se eligen dictan la cantidad de recursos que consumirá cada una de ellas (10).

La Programación Lineal es una de las principales ramas de la IO. En esta categoría se consideran todos aquellos modelos de optimización donde las funciones que lo componen (función objetivo y restricciones)

[1] Solución factible: es un punto donde todas las restricciones son satisfechas.

son funciones lineales en las variables de decisión. Los supuestos básicos de la Programación Lineal son: linealidad, modelos deterministas, variables reales, no negatividad.

Variables y parámetros de decisión: son las incógnitas (o decisiones) que deben determinarse resolviendo el modelo. Los parámetros son los valores conocidos que relacionan las variables de decisión con las restricciones y función objetivo.

Restricciones del problema: consiste en definir un conjunto de ecuaciones o inecuaciones que restringen los valores de las variables de decisión a aquellos considerados como factibles. Pueden ser:

Tabla 1 Tipos de restricciones

Tipo de restricción	Ecuación	Observación
Tipo 1	$\sum_{i=1}^{N} a_{i,j} * X_i = b_j$	b_j ser respetado estrictamente, donde j número de la ecuación que varía de 1 a M
Tipo 2	$\sum_{i=1}^{N} a_{i,j} * X_i \geq b_j$	b_j no debe ser superado, donde j número de la ecuación que varía de 1 a M
Tipo 3	$\sum_{i=1}^{N} a_{i,j} * X_i \leq b_j$	b_j debe ser respetado o puede ser superado, donde j número de la ecuación que varía de 1 a M

Donde:

j : número de la ecuación que varía de 1 a M. (número total de restricciones)

a : coeficiente técnico conocido.

X_i: incógnitas de 1 a N, donde i número de incógnita que varía de 1 a N.

En general no hay restricciones en cuanto a los valores de N y M, puede ser N=M; N>M; o N<M.

Sin embargo, si las restricciones de Tipo1 son N, el problema puede ser determinado, y puede no tener sentido una optimización. Los tres tipos de restricciones pueden darse simultáneamente en el mismo problema.

Función objetivo: define la medida de efectividad del sistema como una función más. La solución óptima será aquella que produzca el mejor valor de la función objetivo, sujeta a las restricciones. Es una función lineal a ser maximizada o minimizada. Puede ser:

$$Maximizar\ Z = C_1X_1 + C_2X_2 + \cdots + C_nX_n \qquad Minimizar\ Z = C_1X_1 + C_2X_2 + \cdots + C_nX_n$$

El modelo de PL es un modelo matemático que puede expresarse entonces como el problema de elegir los valores de las variables de decisión de manera que se maximice la función objetivo, sujeta a las restricciones dadas. Un modelo de este tipo, y algunas variaciones menores sobre él, tipifican los modelos analizados en Investigación de Operaciones (10).

Modelo general en su forma estándar:

$$Maximizar\ Z = C_1X_1 + C_2X_2 + \cdots + C_nX_n$$

Sujeto a:

$$a_{11}X_1 + a_{12}X_2 + \cdots + a_{1n}X_n \leq b_1$$

$$a_{21}X_1 + a_{22}X_2 + \cdots + a_{2n}X_n \leq b_2$$

$$\vdots \qquad\qquad \vdots$$

$$a_{m1}X_1 + a_{m2}X_2 + \cdots + a_{mn}X_n \leq b_m$$

$$X_1, X_2, \cdots, X_n \geq 0$$

Donde:

m: cantidad de recursos

n: cantidad de actividades

Z: valor de la medida global de efectividad

X_j: variables de decisión (o nivel de la actividad j, para j=1, 2, …, n)

C_j: coeficiente económico para actividad j

a_{ij}: cantidad del recurso i consumido por cada unidad de la actividad j

b_i: disponibilidad del recurso i

Métodos de solución:

Método gráfico: es un procedimiento de solución de problemas de PL, muy limitado en cuanto al número de variables (2 si es un gráfico 2D y 3 si es 3D) pero muy rico en materia de interpretación de resultados e incluso análisis de sensibilidad. Este consiste en representar cada una de las restricciones y encontrar en la medida de lo posible el polígono (poliedro) factible, comúnmente llamado el conjunto solución o región factible (11).

Pasos:

- ✓ Graficar las restricciones como igualdades y luego determinar el área correspondiente a la desigualdad.
- ✓ Determinar el área común a todas las restricciones.
- ✓ Evaluar la función objetivo en cada punto extremo y se selecciona la solución que proporciona el mejor valor a la misma.

Método Simplex: desarrollado por el matemático norteamericano George Dantzig en 1947, hace uso de la propiedad de que la solución óptima de un problema de Programación Lineal se encuentra en un vértice o frontera del dominio de puntos factibles, por lo cual, la búsqueda secuencial del algoritmo se basa en la evaluación progresiva de estos vértices hasta encontrar el óptimo (45). “Es un algoritmo iterativo (un procedimiento de solución sistemático que repite una serie de pasos fija, llamada iteración, hasta que se obtiene el resultado deseado)” (10) que permite ir mejorando la solución a cada paso, y concluye cuando ya no es posible que el valor de la función objetivo mejore más. Considerado un instrumento potente para la resolución de Problemas de PL, consecuencia directa del método algebraico de Gauss. La complejidad computacional práctica esperada del método simplex es del $O(m^2 n)$, aunque su complejidad computacional teórica sea exponencial (12).

El método en su forma tabular seguiría el siguiente algoritmo:

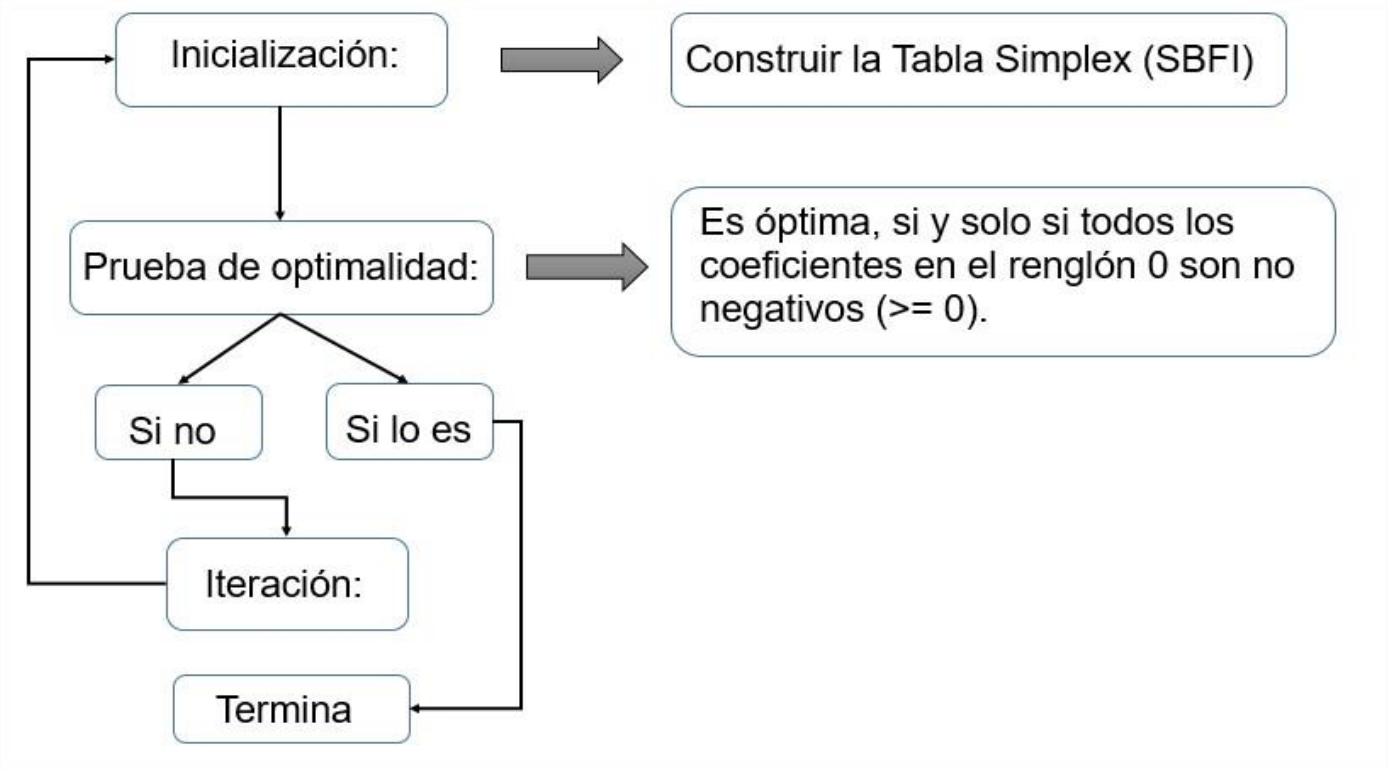


Fig. 1 Algoritmo del método simplex en su forma tabular. Fuente: (10)

Para las iteraciones se tendrá en cuenta:

- **Criterio de entrada:** aquel que más incrementa la FO. El coeficiente negativo de mayor módulo en el renglón 0.
- **Criterio de salida:** prueba de cociente mínimo.
- **Técnica del pivote:** para actualizar la tabla.

Prueba de optimalidad: la solución básica factible es óptima si y sólo si todos los coeficientes del renglón 0 son no negativos (>= 0). Si es así, el proceso se detiene; de otra manera, sigue a una iteración para obtener la siguiente solución básica factible, que incluye cambiar una variable no básica a básica y viceversa y después despejar la nueva solución (10).

Técnica del pivote: se utiliza para actualizar la tabla cuando se realiza una iteración del método simplex. El procedimiento de esta técnica se describe a continuación:

- ✓ Encontrar el elemento pivote (Intersección de la columna de la variable que entra y la fila de la variable que sale).

- ✓ Sustituir en la base la variable que sale por la que entra.
- ✓ Dividir cada elemento de la fila de la variable que sale entre el pivote.
- ✓ Determinar el valor correspondiente a las otras casillas mediante la expresión (13):

$$NV = VV - \frac{(valor\ fila) * (valor\ columna)}{Pivote}$$

Donde:

- (NV) Nuevo Valor
- (VV) Viejo Valor

Pasos para la prueba del cociente mínimo:

1. Se eligen los coeficientes estrictamente positivos de la columna pivote.
2. Se divide el elemento del lado derecho del mismo renglón entre dicho coeficiente.
3. Se identifica el renglón que tiene el menor de estos cocientes.
4. La variable básica de ese renglón es la variable básica que sale; se sustituye con la variable básica entrante en la columna de la variable básica de la siguiente tabla. (10)

Método de la M: consta de dos fases, en la primera fase todas las variables artificiales se hacen cero (debido a la penalización de M por unidad al ser mayores que cero) con el fin de obtener una solución básica factible inicial para el problema real. En la segunda fase todas las variables artificiales se mantienen en cero (por esta misma penalización), mientras que el método simplex genera una secuencia de soluciones básicas factibles que llevan a la solución óptima. (10)

A continuación, las principales características de este método:

- ✓ Plantear el problema artificial, introduciendo variables artificiales en restricciones Tipo 1 y Tipo 2, que sean las variables básicas iniciales de esas ecuaciones.
- ✓ Se plantea M como un número positivo bien grande. Se penaliza la función objetivo de la siguiente manera: restar Mx_A en caso de un modelo de maximizar y sumar Mx_A en caso de minimizar.
- ✓ Las iteraciones del Simplex obligan a las variables artificiales a hacerse cero cada una hasta que queden fuera de la solución (14).

Análisis Postóptimo: permite investigar el efecto que tendría sobre la solución óptima la existencia de otras condiciones para el problema. Se realiza después de obtener una solución óptima. Evita tener que resolver nuevamente, desde el principio, el problema planteado.

Los **precios sombra** del recurso i miden el su valor marginal, es decir, la tasa a la que Z puede cambiar para pequeñas variaciones de la cantidad que se proporciona de este recurso (b_i). En la tabla final del Simplex se identifica el precio sombra como el opuesto del coeficiente de la variable de holgura, artificial y superávit correspondiente a cada restricción (10).

El **costo reducido** mide el efecto sobre la función objetivo de un aumento unitario en el valor de cada una de las variables no básicas (46). En la tabla final de Simplex los costos reducidos son los coeficientes de las variables de decisión en la función objetivo.

Ejemplo del algoritmo del método simplex en forma tabular:

Planteamiento del problema:

$$\text{Maximizar } Z = 12x_1 + 8x_2$$

sujeta a

$$5x_1 + 2x_2 \leq 150$$
$$2x_1 + 3x_2 \leq 100$$
$$4x_1 + 2x_2 \leq 80$$

y

$$x_1 \geq 0, \quad x_2 \geq 0$$

Introducimos variables de holgura h_1, h_2, h_3, para convertir las desigualdades en igualdades:

$$\text{Maximizar } Z = 12x_1 + 8x_2$$

sujeta a

$$5x_1 + 2x_2 + h_1 = 150$$

$$2x_1 + 3x_2 + h_2 = 100$$

$$4x_1 + 2x_2 + h_3 = 80$$

Se construye la Tabla Simplex Inicial, cuya solución básica factible inicial será: $h_1 = 150, h_2 = 100, h_3 = 80, x_1 = 0, x_2 = 0$:

Tabla 2 Tabla inicial del Simplex

MAX : Z	X1	X2	H1	H2	H3	PD
Z	-12	-8	0	0	0	0
H1	5	2	1	0	0	150
H2	2	3	0	1	0	100
H3	4	2	0	0	1	80

Se realiza la prueba de optimalidad, en el cual se comprueba que $Z_i \geq 0$ para todos los i, y como existen valores negativos, esta solución no es óptima. Se construye una nueva solución básica factible. Para ello, se calcula la variable no básica que debe entrar en la base aplicando el criterio de entrada, por tanto, la variable que debe entrar a la base es x_1.

Se calcula ahora la variable básica que debe salir de la base, aplicando el criterio de salida a partir de la prueba del cociente mínimo, por tanto, la variable que debe abandonar la base es h_3.

Comenzamos a escribir la nueva tabla con la nueva base $\mathrm{h}_1, \mathrm{h}_2, x_1$. La intersección de la columna de la variable no básica que entra (columna pivote) con la fila de la variable básica que sale (fila pivote) es el elemento pivote al cual debemos darle el valor 1 al dividir la fila pivote entre el elemento pivote:

Tabla 3 Paso antes de la primera iteración del Simplex

MAX : Z	X1	X2	H1	H2	H3	PD
Z						
H1						
H2						
X1	1	0.5	0	0	0.25	20

Para actualizar los valores de la tabla se aplica la técnica del pivote:

Tabla 4 Primera iteración del Simplex

MAX : Z	X1	X2	H1	H2	H3	PD
Z	0	-2	0	0	3	240
H1	0	-0.5	1	0	-1.25	50
H2	0	2	0	1	-0.5	60
X1	1	0.5	0	0	0.25	20

Se realiza la prueba de optimalidad y se comprueba que esta solución no es óptima. Se calcula la variable no básica que debe entrar en la base, que es x_2 y se calcula ahora la variable básica que debe salir de la base, que es h_2. Construimos la nueva tabla y para actualizar los valores de la misma se aplica la técnica del pivote nuevamente:

Tabla 5 Segunda iteración del Simplex

MAX : Z	X1	X2	H1	H2	H3	PD
Z	0	0	0	1	2.5	300
H1	0	0	1	0.25	-1.375	65
X2	0	1	0	0.5	-0.25	30
X1	1	0	0	-0.25	0.375	5

Se realiza la prueba de optimalidad y se comprueba que $\mathrm{Z_i} \geq 0$, para todos los i, por lo que hemos llegado a la solución básica factible óptima, la cual es: x_1=5, $x_2 = 30, h_1 = 65, h_2 = 0, h_3 = 0$.

Tipos de soluciones: son las diferentes soluciones que puede tener un problema de programación lineal.

- ✓ Solución óptima (única o múltiple).
- ✓ Solución degenerada.
- ✓ Solución no acotada (ausencia de solución), cuando la función objetivo no tiene valores extremos, pues la región factible es no acotada.
- ✓ Solución no factible, cuando no existe región factible por falta de puntos comunes en el sistema de inecuaciones (15).

1.3 Empleo de las tecnologías de la información y la comunicación en el proceso de enseñanza-aprendizaje

Las tres grandes razones para el uso de las TIC en la educación son la innovación metodológica, la alfabetización digital y la productividad, el empleo de metodologías novedosas que faciliten la adquisición de los aprendizajes, evitando caer en la monotonía y huyendo de una concepción tradicional de la enseñanza. En el quehacer docente se debe emplear metodologías y herramientas que permitan seguir trabajando en la participación activa por parte del alumnado, que este aprenda a gestionar sus propios procesos de aprendizaje, así como diseñar materiales que faciliten la adquisición del conocimiento (16).

La docencia universitaria ha experimentado cambios radicales en cuanto a escenarios, plan de estudio, organización del PEA y medios empleados en dicho proceso (17). El empleo de las TIC en la formación de la enseñanza superior aporta múltiples ventajas en la mejora de la calidad docente, materializadas en aspectos tales como el acceso desde áreas remotas, la flexibilidad en tiempo y espacio para el desarrollo de las actividades de enseñanza-aprendizaje o la posibilidad de interactuar con la información por parte de los diferentes agentes que intervienen en dichas actividades.

La tecnología educativa es una forma sistemática de diseñar, desarrollar y evaluar el proceso total de enseñanza-aprendizaje, en términos de objetivos específicos, basada en las investigaciones sobre el mecanismo del aprendizaje y la comunicación que, aplicando una coordinación de recursos humanos, metodológicos, instrumentales y ambientales, conduzca a una educación eficaz (18).

Sobre la tecnología educativa, se hace referencia a dos enfoques: primero, la definición superficial de la tecnología educativa limitada al uso de medios; segundo, a nivel profundo mostrando a la tecnología educativa como herramienta para el diseño institucional. Ahí está la diferencia entre tecnología en la educación y tecnología de la educación. La tecnología educativa se define como el medio que establece un camino entre las ciencias educativas (psicología, pedagogía, filosofía, y otras) y sus aplicaciones para resolver problemas de aprendizaje. Es un espacio pedagógico para representar, difundir y acceder a información y conocimientos, en diferentes contextos educativos (19).

La tecnología educativa es el uso de hardware físico y teoría educativa. Abarca varios dominios, incluida la teoría del aprendizaje, la formación basada en la informática, el aprendizaje en línea y, cuando se utilizan tecnologías móviles, el m-learning (20).

A través de la Informática Educativa como disciplina que aplica las tecnologías de la información y el conocimiento, se han resuelto múltiples carencias docentes a partir de la creación de software como medios de enseñanza. Su importancia radica en que facilita a los estudiantes la generalización, sistematización e integración de los conocimientos; disminuye la tendencia al enciclopedismo al eliminar las repeticiones y detalles innecesarios, con lo cual logra establecer las esencialidades en cada disciplina; desarrolla y controla el proceso docente con un cuerpo de conocimientos ya integrados, facilita el trabajo educativo sistemático con cada alumno y permite su autoevaluación (22).

La Informática Educativa se considera emergente debido a que su misma evolución como objeto de estudio coloca a los postulados que le constituyen en una posición de búsqueda permanente de espacios de validación y debate por parte de la comunidad académica y de experiencias prácticas que realimenten sus constructos teóricos (47).

El concepto de software educativo ha sido abordado por diferentes autores, atribuyéndole disímiles definiciones a pesar de las cuales se imponen las potencialidades y su absoluto basamento en los principios de la enseñanza para su vinculación con el PEA. Bill Gates en su libro “Camino al futuro” define al software educativo como programa informático, medio de enseñanza bidireccional, interactivo, basado en una forma de presentar la información que emplea una combinación de texto, sonido, imagen, animación, video con propósitos específicos dirigidos a contribuir con el desarrollo de predeterminados aspectos del proceso docente (4). Software educativo según lo planteado por Tchounikine es un software específicamente diseñado para guiar al estudiante a desarrollar una actividad que es favorable y está orientado con los objetivos pedagógicos propuestos (5).

Un tipo de software educativo son los asistentes matemáticos, los cuales propician que el profesor se concentre en su nuevo papel de estimulador y orientador del aprendizaje. Evidentemente esto está vinculado a las teorías del aprendizaje que actualmente encontramos en el contexto de la informática educativa, dado que el proceso de aprendizaje como fenómeno subjetivo puede ser abordado desde diferentes ópticas y es altamente complejo. En esta situación el profesor se convierte en un facilitador que explora el conocimiento previo de los estudiantes y proporciona un ambiente adecuado para que el estudiante construya su propio conocimiento. El estudiante, por su parte, interactúa con el objeto de aprendizaje para lograr su objetivo (21).

1.4 Análisis de soluciones similares

A nivel internacional existen un conjunto de herramientas que permiten la resolución de problemas de programación lineal, las cuales se describen a continuación, pero antes definen los elementos que se buscan en cada una de ellas:

Elementos a considerar en las herramientas:

- Poseer interfaz para resolver los problemas.
- Que no sea necesario el acceso a internet.
- Que sea multiplataforma.
- Que grafique.
- Que no existan diferencias en la tabla simplex que se realiza en el aula con la que pueda mostrar dicho software.

1.4.1 El WinQSB o QSB - Quantitative System Business

Es un paquete de herramientas desarrolladas por el Dr. Yih-Long Chang para solucionar y automatizar problemas de carácter complejo. WinQSB es un software de ayuda a la toma de decisiones que contiene herramientas muy útiles para resolver distintos tipos de problemas en el campo de la investigación de operaciones.

WinQSB nos permite solucionar una gran cantidad de problemas administrativos, de producción, de recursos humanos, dirección de proyectos, entre otros. El software está formado por distintos módulos, uno para cada tipo de modelo o problema. En total son 19 módulos.

WinQSB utiliza los mecanismos típicos de la interface de Windows, es decir, ventanas, menús desplegables, barras de herramientas, etc. Por lo tanto, el manejo del programa es similar a cualquier otro que utilice el entorno Windows (23).

1.4.2 Solver

El estándar Microsoft Excel Solver usa una implementación básica del método simple Simplex para resolver problemas de programación lineal. Está limitado a 200 variables de decisión. El Solucionador Premium utiliza un método Simplex primordial mejorado con límites de dos lados en las variables. Maneja hasta 1,000 variables de decisión. La plataforma Premium Solver usa una versión extendida LP / cuadrática de este Simplex Solver para manejar problemas de hasta 2,000 variables de decisión. Opcionalmente, utiliza un método Dual Simplex para resolver subproblemas de programación lineal en un problema de entero mixto (24).

1.4.3 JSimplex

Es un programa para resolver problemas de programación lineal de forma online, es decir, hay que tener acceso a internet para poder utilizarlo. Resuelve el problema usando el método Simplex. Se da a escoger el objetivo, ya sea maximizar o minimizar, y dos valores de entrada, el número de variables y el número de restricciones (25).

1.4.4 PHPSimplex

Es una herramienta online para resolver problemas de programación lineal. Su uso es libre y gratuito. PHPSimplex es capaz de resolver problemas mediante el método Simplex, el método de las dos fases, y el método gráfico, y no cuenta con limitaciones en el número de variables de decisión ni en las restricciones de los problemas. No solo muestra los resultados finales, sino también las operaciones intermedias. También ofrece la solución directa para uso de profesionales. Otras de sus ventajas son que no precisa de ningún lenguaje para enunciar el problema, ofrece una interfaz sencilla, es cercano al usuario, de manejo fácil e intuitivo, no es necesario instalar algo para poder usarlo, y está disponible en varios idiomas. Posee un manual de ayuda para aprender a utilizar la herramienta (26).

1.4.5 MATLAB

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es asistente matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux.

Características principales:

- Lenguaje de alto nivel para cálculos científicos y de ingeniería.
- Entorno de escritorio mejorado para la exploración, el diseño y la solución de problemas de manera iterativa.
- Gráficos para visualizar datos y herramientas para crear diagramas personalizados.
- Aplicaciones para ajustar curvas, clasificar datos, analizar señales, ajustar sistemas de control y muchas otras tareas.
- Cuadros de herramientas complementarios para una amplia variedad de aplicaciones de ingeniería y científicas.
- Herramientas para crear aplicaciones con interfaces de usuario personalizadas.
- Interfaces para C/C++, Java®, .NET, Python, SQL, Hadoop y Microsoft® Excel®.
- Opciones de implementación sin costo en concepto de derechos de licencia para compartir programas de MATLAB con los usuarios finales (27).

1.4.6 R

R es un lenguaje y entorno para computación y gráficos estadísticos. R proporciona una amplia variedad de técnicas estadísticas (modelado lineal y no lineal, pruebas estadísticas clásicas, análisis de series de tiempo, clasificación, agrupamiento) y gráficos, y es altamente extensible. El lenguaje S suele ser el vehículo de elección para la investigación en metodología estadística, y R proporciona una ruta de código abierto para la participación en esa actividad (28).

Uno de los puntos fuertes de R es la facilidad con la que se pueden producir parcelas de calidad de publicación bien diseñadas, que incluyen símbolos matemáticos y fórmulas cuando es necesario. Se ha tenido mucho cuidado con los valores predeterminados para las opciones menores de diseño en los gráficos, pero el usuario conserva el control total. R está disponible como software libre bajo los términos de la Licencia Pública General GNU de la Free Software Foundation en forma de código fuente. Se compila y se ejecuta en una amplia variedad de plataformas UNIX y sistemas similares (incluidos FreeBSD y Linux), Windows y MacOS (28).

R, está diseñado en torno a un verdadero lenguaje informático y permite a los usuarios agregar funciones adicionales definiendo nuevas funciones. Gran parte del sistema está escrito en el dialecto R, lo que facilita a los usuarios seguir las elecciones algorítmicas realizadas. Para tareas intensivas en cómputo, el código C, C ++ y Fortran se puede vincular y ejecutar en tiempo de ejecución. Los usuarios avanzados pueden escribir código C para manipular objetos R directamente.

En la Universidad de las Ciencias Informáticas (UCI), actualmente se trabaja con la aplicación WinQSB. Sin embargo, por lo antes expuesto en la situación problemática, esta presenta algunas dificultades para que los estudiantes y profesores puedan trabajar con ella.

- WinQSB, funciona únicamente en el sistema operativo Windows con arquitectura de 32bit, por lo que para que funcione en los laboratorios de la universidad hay que emular la aplicación con el software Wine, apareciendo diferentes dificultades en su uso, como interrumpir su funcionamiento en varias ocasiones inesperadamente además de las diferencias en la tabla de salida que da el mismo con la que se da en las aulas.
- Solver, es una herramienta propietaria que funciona para Windows y GNU/Linux, y ofrece una versión de prueba de 15 días, y permite después la compra de la licencia.
- JSimplex, es una herramienta online por lo que hay que tener acceso a internet para poder utilizarlo y se estaría dependiendo de este requisito para poder utilizarla.
- PHPSimplex, es una herramienta online para resolver problemas de programación lineal por lo que hay que tener acceso a internet para poder utilizarlo.

- MatLab es una herramienta propietaria que para su uso hay que poseer licencia, además ofrece licencias gratuitas educativas cuya duración es restringida. Para su instalación y posterior uso hay que destinar una cantidad de almacenamiento de la computadora que es 1GB para instalar MatLab solamente, hasta 3 – 4 GB una instalación completa y de memoria RAM es 1GB, pero se recomienda 2GB, para la versión MatLab R2014b, lo que va aumentando si se instalan versiones más modernas de la herramienta. Además, si la herramienta informática está basada en MatLab, para su uso hay que tener instalado el Matlab o el MatLab Runtime, lo que traía consigo que fuera una versión diferente para Windows y Linux, en sus diferentes arquitecturas 32 y 64 bits. Actualmente el MatLab Runtime no está disponible en el ftp de la universidad y para sistema operativo Linux la versión de Matlab Runtime no tiene soporte para arquitecturas de 32 bits a partir de la versión MatLab R2012b. Para el sistema operativo Windows a partir de la versión R2016a no tiene soporte para arquitecturas de 32 bits.
- R es un sistema basado en comandos, por lo que el empleo de las utilidades *graphical user interface* (GUI) de estos programas puede resultar complicado para el usuario ya que debe acceder a diferentes menús antes de ejecutar un procedimiento específico. No tiene ninguna garantía legal y el usuario asume cualquier potencial problema causado por su uso (28).

A continuación, se presenta la tabla resumen con las características que fueron objeto de análisis en cada una de las herramientas.

Tabla 6 Comparación de herramientas

Características / Herramientas	Matlab	WinQSB	Solver	JSimplex	PHPSimplex	R
No existen diferencias en la tabla Simplex que se realiza en el aula con la que muestra el software	NO	NO	NO	NO	NO	NO
Graficación	SI	SI	SI	NO	SI	SI
Multiplataforma	SI	NO	SI	SI	SI	SI
No es necesario el acceso a internet	SI	SI	SI	NO	NO	SI
Posee interfaz para resolver los problemas	NO	SI	NO	SI	SI	NO

El análisis de soluciones existentes, evidenció que existen varios softwares para la solución de problemas de programación lineal que podrían apoyar de una manera u otra el proceso de enseñanza-aprendizaje en la asignatura IO. Estos sistemas no dan solución al problema expuesto para facilitar el PEA del tema Programación Lineal en la asignatura Investigación de Operaciones haciendo un uso adecuado de las

tecnologías educativas, ya que no cumplen con las características expuestas al inicio, que son las que se requieren.

1.5 Metodologías de desarrollo de software

Según Pressman, el software es un elemento clave de la evolución de sistemas y productos basados en computadoras, y una de las tecnologías más importantes en todo el mundo. La intención de la ingeniería de software es proporcionar un marco general para construir software con una calidad mucho mayor (29).

Se considera que el uso de las metodologías es una buena práctica para guiar el proceso de desarrollo de un software. El éxito del producto depende en gran parte de la metodología escogida, ya sea tradicional o ágil. Por los motivos anteriormente expuestos se decide realizar un estudio aplicando el método de Boehm y Turner para seleccionar el enfoque ágil o tradicional y posteriormente seleccionar la metodología de desarrollo adecuada.

1.5.1 Método de Boehm y Turner

El método de Boehm y Turner plantea 5 criterios fundamentales mediante los que se estará valorando el proyecto; estos son: tamaño del equipo, criticidad del producto, dinamismo de los cambios, cultura del equipo y personal con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta a la hora de seleccionar uno u otro enfoque.

Para la selección del valor que se ubicará en cada eje (uno para cada criterio) de la estrella se debe tener en cuenta el comportamiento de estos criterios en el proyecto. En lo sucesivo se describe cada uno:

Tamaño: este criterio se utiliza para representar el número de personas involucradas en el proyecto. Pueden tenerse en cuenta el nivel de complejidad que pueda presentarse en la comunicación entre los miembros del proyecto y los costos que pueden provocar cambios esperados.

Criticidad: se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto. Su evaluación puede ser cualitativa.

Dinamismo: representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto.

Personal: representa la proporción del personal con experiencia alta, media y baja. Los métodos orientados al plan no se ven afectados negativamente por este factor pues no interesa el nivel de experiencia con la que cuenten los miembros del equipo.

Cultura: las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual. Esto refleja el nivel de ceremonia necesario y aceptado: documentación, control, formalismo en las comunicaciones (30).

Después de aplicar el método Boehm y Turner, el resultado que arrojó, es que se debe utilizar una metodología ágil para el desarrollo de la aplicación, por lo que se decide seleccionar la metodología XP.

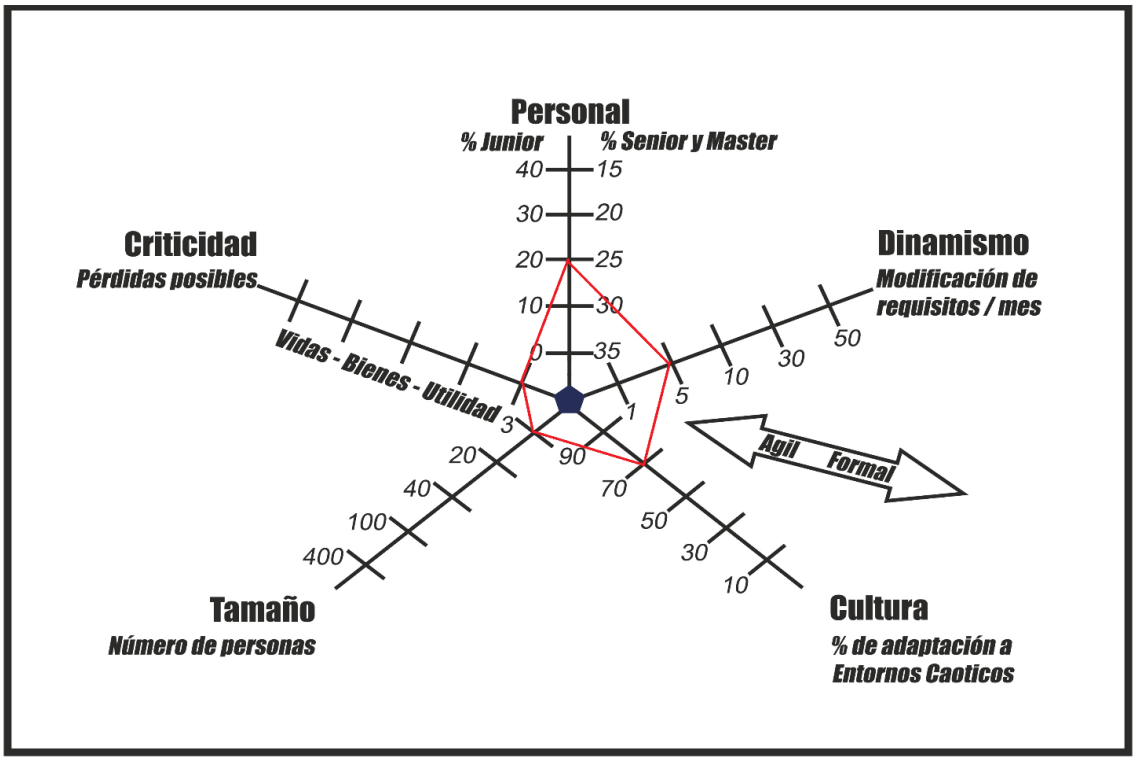


Fig. 2 Estrella que representa el proyecto según la aplicación de Boehm y Turner

1.5.2 Programación Extrema (XP)

La Programación Extrema es un proceso ágil de desarrollo de software formulada por Kent Beck, se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad (31).

XP 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 la 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. XP se define como especialmente adecuada

para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre (32).

Entre las características esenciales de XP aparecen:

Las Historias de Usuario (HU): son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las HU es muy dinámico y flexible, en cualquier momento pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (32).

Roles XP: entre los roles que propone XP se tienen los siguientes:

- Programador
- Cliente
- Encargado de pruebas (Tester)
- Encargado de seguimiento (Tracker)
- Entrenador (Coach)
- Consultor
- Gestor (Big boss)

Proceso XP: un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.
3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. El ciclo de vida ideal de XP consiste de seis fases:

1. Exploración
2. Planificación de la entrega

3. Iteraciones
4. Producción
5. Mantenimiento
6. Muerte del proyecto

Para una mejor estructuración de la investigación y un mejor entendimiento, se decidió agrupar las seis fases en cuatro, sin violar el ciclo de vida de la metodología, estas fases son:

- Fase I: Exploración
- Fase II: Planificación
- Fase III: Implementación
- Fase IV: Prueba

Prácticas XP: la aplicación disciplinada de las prácticas posibilita disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto.

- El juego de la planificación: es un espacio frecuente de comunicación entre el cliente y los programadores.
- Entregas pequeñas: la idea es producir rápidamente versiones del sistema que sean operativas.
- Diseño simple: se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
- Pruebas: la producción de código está dirigida por las pruebas unitarias.
- Refactorización: la refactorización es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios.
- Programación en parejas: toda la producción de código debe realizarse con trabajo en parejas de programadores.
- Propiedad colectiva del código: cualquier programador puede cambiar cualquier parte del código en cualquier momento.
- Integración continua: cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- 40 horas por semana: se debe trabajar un máximo de 40 horas por semana.
- Cliente in-situ: el cliente tiene que estar presente y disponible todo el tiempo para el equipo.
- Estándares de programación: XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación.

- Comentarios respecto de las prácticas: el mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras (32).

1.6 Tecnologías y herramientas de desarrollo

1.6.1 Lenguajes de programación empleados

Java 8

Para la solución propuesta se utiliza el lenguaje de programación Java. Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características muy importantes:

- Es un lenguaje que es compilado, generando ficheros de clases compilados, pero estas clases compiladas son en realidad interpretadas por la máquina virtual java, siendo la máquina virtual de java la que mantiene el control sobre las clases que se estén ejecutando.
- Es un lenguaje multiplataforma. El mismo código java que funciona en un sistema operativo funcionará en cualquier otro sistema operativo que tenga instalada la máquina virtual java.
- Es un lenguaje seguro. La máquina virtual al ejecutar el código java realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.
- Gracias al API de java podemos ampliar el lenguaje para que sea capaz de, por ejemplo, comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas, crear aplicaciones visuales al estilo Windows.

Java es el lenguaje apropiado para la implementación de la solución ya que es un lenguaje que se compila para correr en una máquina virtual (JVM) y esta máquina virtual corre en casi cualquier sistema operativo: Windows, Linux, UNIX, Solaris, hasta en dispositivos móviles. Para desarrollar en Java no es necesario comprar licencias de ningún tipo, es completamente gratuito y libre. Java restringe el uso de aspectos críticos del sistema para evitar la codificación de virus.

Existen multitud de aplicaciones gráficas de usuario basadas en Java. El entorno de ejecución Java (JRE) se ha convertido en un componente habitual en los PC de usuario de los sistemas operativos más usados en el mundo. Además, muchas aplicaciones Java lo incluyen dentro del propio paquete de la aplicación de modo que se ejecuten en cualquier computadora (33).

1.6.2 Bibliotecas empleadas

Apache Commons Math v3.6.1

Apache Commons Math es una biblioteca de componentes livianos y autónomos que consta de funciones matemáticas, estructuras que representan conceptos matemáticos (como números complejos, polinomios, vectores, etc.) y algoritmos que podemos aplicar a estas estructuras (búsqueda de raíz, optimización, ajuste de curva, cálculo de intersecciones de figuras geométricas, etc.).

Apache Commons Math es una librería *open source* a la que se puede acceder, modificar y adaptar según las necesidades del problema. En concreto, se ha hecho uso del paquete “optim.linear” implementado en la versión 3.6.1 de la librería (34).

Principios rectores:

- Los casos de uso de aplicaciones del mundo real determinan la prioridad de desarrollo.
- Este paquete enfatiza componentes pequeños y fácilmente integrados en lugar de bibliotecas grandes con dependencias y configuraciones complejas.
- Todos los algoritmos están completamente documentados y siguen las mejores prácticas generalmente aceptadas.
- En situaciones donde existen algoritmos estándar múltiples, se usa un patrón de Estrategia para admitir implementaciones múltiples.
- Dependencias limitadas Sin dependencias externas más allá de los componentes de Commons y la plataforma central de Java (al menos Java 1.3 hasta la versión 1.2 de la biblioteca, al menos Java 5 que comienza con la versión 2.0 de la biblioteca) (35).

JFreeChart v1.0.19

JFreeChart es una biblioteca de gráficos escrita en Java, la cual es gratuita y facilita a los desarrolladores la visualización de gráficos de calidad profesional en sus aplicaciones. El extenso conjunto de características de JFreeChart incluye:

- Una API consistente y bien documentada que admite una amplia gama de tipos de gráficos.
- Un diseño flexible que es fácil de extender, y se dirige tanto a aplicaciones del lado del servidor como del lado del cliente.

- Soporte para muchos tipos de salida, incluidos los componentes Swing y JavaFX, archivos de imagen (incluidos PNG y JPEG) y formatos de archivos de gráficos vectoriales (incluidos PDF, EPS y SVG).
- Presentación de datos en Servlet y Java Server Pages (JSP).
- Nos ofrece numerosas interfaces para el tratamiento de datos en base al grafico que se desea crear.
- Manejo de gráficos: funciones de zoom, impresión. Se permite la representación gráfica en los siguientes formatos: gráficos de tarta (2D y 3D), gráficos de barras, gráficos de líneas y áreas, gráficos de dispersión, gráficos de tiempos, gráficos combinados, diagramas de Gantt.
- JFreeChart es de código abierto o, más específicamente, de software libre. Se distribuye bajo los términos de la Licencia Pública General Reducida (LGPL por sus siglas en ingles) de GNU, que permite el uso en aplicaciones propietarias (36).

1.6.3 Entorno de desarrollo integrado

Netbeans IDE 8.2

Un entorno de desarrollo integrado, llamado también (IDE por sus siglas en inglés), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

Netbeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el Netbeans IDE. Netbeans IDE es un producto libre y gratuito sin restricciones de uso.

A continuación, algunas ventajas de este IDE:

- Es un IDE multilenguaje y adaptable.
- Es software libre y gratuito.
- Intuitivo y fácil de utilizar.
- Se puede desarrollar todo tipo de aplicaciones.
- Es poderoso y extensible.
- Fácil integración con el servidor de aplicaciones “Glassfish” (37).

1.7 Conclusiones parciales

El estudio sobre el proceso de enseñanza-aprendizaje, específicamente de las tecnologías educativas y la informática educativa, permitió constatar que la educación está mediada por las TIC, además de facilitar el proceso de enseñanza-aprendizaje. La caracterización de los principales elementos teóricos de la programación lineal en la asignatura Investigación de Operaciones contribuyó a una mayor comprensión para el diseño e implementación del software. El análisis realizado a las herramientas existentes permitió constatar que los sistemas analizados no representan una vía de solución factible para el problema de investigación propuesto. Luego de haber analizado todos estos elementos, se realizó un estudio de las herramientas, tecnologías y metodologías a utilizar que permitió asumir como metodología de desarrollo XP, para la implementación se utilizó Java como lenguaje de programación apoyado en el IDE Netbeans, para la resolución del método simplex se utilizó la biblioteca Apache Commons Math y para la graficación se utilizó la biblioteca JFreeChart.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA, EXPLORACIÓN Y PLANIFICACIÓN

La metodología de desarrollo XP define un conjunto de prácticas y técnicas que le confieren al equipo de desarrollo una guía por la cual orientarse durante el tiempo de confección de la solución propuesta. El uso adecuado de estas técnicas es lo que permite lograr entre otras cosas una planificación, que, de cumplirse, tiene como resultado la entrega del producto en tiempo.

Algunos de los principios, prácticas y técnicas que sirven de guía para el equipo de desarrollo y que se describen en este capítulo son: las historias de usuario (HU), el plan de iteraciones, plan de entrega y las tarjetas Clase-Responsabilidad-Colaborador (CRC). Se describen las funcionalidades y se generarán los artefactos propuestos por la metodología seleccionada; permitiendo ganar en claridad y organización.

2.1 Descripción de la solución propuesta

Como propuesta de solución de esta investigación se plantea el desarrollo de un software multiplataforma, de código abierto y licencia libre; en la cual el usuario realice la entrada de los datos del problema de programación lineal. El software debe realizar los cálculos y mostrar al usuario la resolución del problema paso a paso o directamente además de la solución óptima, la tabla de iteración, la solución básica factible hasta el momento, mostrar variable básica saliente y variable no básica entrante, la solución gráfica del problema y permita guardar, cargar en formato Simplex Soft File Archived (.ssfa), además de exportar la solución como imagen.

Entrada de datos:

Al ejecutar el software el usuario debe tener dos formas de acceder a un nuevo problema; primero cargando un archivo de extensión (.ssfa), que es la extensión definida para los archivos del software y segundo accediendo a crear nuevo problema.

Para crear un nuevo problema el usuario debe dirigirse al menú archivo dentro de este nuevo problema, donde debe ingresar el nombre del problema, la cantidad de restricciones, cantidad de variables, además del criterio de la función objetivo (maximizar o minimizar). El nuevo problema está predefinido que se resuelva para variables continuas solamente. Después de llenar los campos se debe crear el modelo.

Luego de creado este, debe aparecer la tabla con la cantidad de restricciones y variables representadas en filas y columnas respectivamente, además de dos columnas extras que son la de desigualdad y la columna

de la parte derecha de las ecuaciones. El usuario debe llenar las casillas de la tabla con los coeficientes de la función objetivo en la primera fila y de las restricciones en las restantes.

Una vez se visualice esta interfaz se pudiera adicionar o eliminar restricción, así como modificar el criterio de la función objetivo, al dar clic derecho sobre la tabla o al dirigirse al menú de editar.

Solución paso a paso:

Esta funcionalidad consiste en resolver el método simplex paso a paso, es decir, por iteraciones, donde se comprueba la prueba de optimalidad para la solución básica factible de esa iteración, si resulta óptima termina, de lo contrario continúa hacia otra iteración, a través del criterio de entrada de una variable a la base, del criterio de salida de la variable de la base y la técnica del pivote para actualizar la tabla.

Luego de que el usuario haya entrado los datos, se debe dirigir al menú resolver y dentro de este en paso a paso. La interfaz que debe aparecer es la de la primera iteración donde se muestre la solución básica factible en esa iteración, la variable no básica que entra a la base y la variable básica que sale de la base. Además, el usuario puede avanzar o retroceder en las iteraciones, ir hacia la última iteración y guardar cada una de las mismas como imagen.

Solución directa:

Esta funcionalidad consiste en ofrecer el resultado directamente, sin ver las iteraciones del método simplex. El software lo debe mostrar de dos formas, por la tabla de iteración final del simplex, que al igual se puede guardar como imagen la misma, o mostrando la solución óptima directamente. Una vez que el usuario haya insertado el modelo, debe encontrar la solución directa en el menú resolver.

Solución por método gráfico:

Para la solución del problema a través del método gráfico el software debe utilizar la biblioteca JFreeChart. Esto solo es posible si las desigualdades entradas para cada restricción son del tipo 3, o sea ($\leq$), donde el problema está en su forma estándar.

Una vez que el usuario haya insertado el modelo, debe dirigirse al menú resolver, donde le debe aparecer una ventana con el botón graficar, y al acceder, aparece la gráfica de dicho modelo.

Método simplex en forma tabular:

En esta interfaz el software debe mostrar la tabla del Simplex en su forma tabular.

Modelo Aumentado:

Al introducir artificios matemáticos[2] en las restricciones funcionales, el modelo de programación lineal de este ejemplo se puede sustituir por el modelo equivalente llamado forma aumentada del modelo que se encuentra a la derecha:

Tabla 7 Modelo aumentado

Modelo inicial	Modelo aumentado
$\text{Maximizar} \quad Z = 3x_1 + 5x_2$	$\text{Maximizar } Z = 3x_1 + 5x_2$
sujeta a	sujeta a
$x_1 \leq 4$	$x_1 + h_1 = 4$
$2x_2 \leq 12$	$2x_2 + h_2 = 12$
$3x_1 + 2x_2 \leq 18$	$3x_1 + 2x_2 + h_3 = 18$
$x_1 \geq 0; x_2 \geq 0$	$x_j \geq 0; \quad j = 1,2$

Aun cuando ambas formas del modelo representan exactamente el mismo problema, la nueva forma es mucho más conveniente para la manipulación algebraica y la identificación de las soluciones. Se le da el nombre de Modelo Aumentado del problema, porque la forma original se aumentó con algunas variables suplementarias necesarias para aplicar el método simplex (10).

[2] Se utiliza para transformar las inecuaciones en ecuaciones.

Abrir y guardar el proyecto:

La extensión utilizada para abrir y guardar el proyecto es Simplex Soft File Archive (.ssfa). Se guardan los datos de la tabla inicial del simplex como tipo Object y el nombre del problema como tipo String. El fichero está con formato de datos serializado por lo que el usuario no puede modificar sus datos fuera del software.

Manual de ayuda:

El manual es una guía para aprender a trabajar con el software. Para que el usuario pueda acceder al mismo debe dirigirse al menú Ayuda.

Logotipo y nombre del software:

Se definió como nombre del software “Simplex Soft”, pues el mismo basa su funcionamiento en el método simplex. Se diseñó un logo que caracteriza al software, como se observa en la imagen siguiente, se puede apreciar que los cuadros que están independientes representan cada iteración de método simplex y el cuadro grande representa la solución del método.

Fig. 3 Logo del software

2.1.1 Licencia

Apache 2.0: es la licencia creada en principio por la Apache Software Foundation para publicar los paquetes del proyecto Apache, sin embargo, en la actualidad es usada para muchos otros proyectos. Se considera una licencia permisiva porque no requiere que los trabajos derivados sean publicados bajo la misma licencia y tampoco exige la liberación del código fuente. Los cambios a la versión original deben reflejarse en un archivo en el código fuente y pueden ser publicados usando cualquier otra licencia. Lo más importante de

esta licencia es que los derechos de autor deben conservarse tanto en el código fuente como en los binarios (38).

La Licencia Apache permite al usuario del software la libertad de usarlo para cualquier propósito; distribuirlo, modificarlo, y distribuir versiones modificadas de ese software. La licencia Apache es permisiva ya que no exige que las versiones modificadas del software se distribuyan usando la misma licencia, ni siquiera que se tengan que distribuir como software libre.

JFreeChart: bajo la licencia *LGPL* (GNU Lesser General Public License en español **Licencia Pública General Reducida de GNU**), es una licencia de software libre publicada por la *Free Software Foundation*. LGPL es una alternativa más permisiva que la estricta licencia copyleft GPL.
Esto supone que cualquier trabajo que use un elemento con licencia GPL **tiene la obligación** de ser publicado bajo las mismas condiciones (libre de usar, compartir, estudiar, y modificar). Por otro lado, LGPL solo requiere que los componentes derivados del elemento bajo LGPL continúen con esta licencia, LGPL es usado habitualmente para licencias de componentes compartidos como por ejemplo librerías (.dll, .so, .jar, etc.) (39).

2.2 Funcionalidades del software

El software de resolución de problemas de Programación Lineal para el apoyo a la asignatura de Investigación de Operaciones debe permitir:

1. Insertar modelo de programación lineal

    - Nombre del problema
    - Cantidad de restricciones
    - Cantidad de variables
    - Criterio de la función objetivo

2. Gestionar restricciones

    - Añadir restricción
    - Eliminar restricción

3. Devolver el modelo aumentado
4. Devolver método simplex en forma tabular
5. Devolver solución básica factible inicial

6. Resolver el problema

    - Resolución del problema paso a paso

        - ✓ Mostrar tabla de iteración
        - ✓ Mostrar tabla de iteración final
        - ✓ Mostrar solución básica factible por iteración
        - ✓ Variable básica saliente
        - ✓ Variable no básica entrante

    - Resolución directa

        - ✓ Mostrar solución óptima
    - Resolución por método gráfico

7. Guardar proyecto
8. Cargar proyecto
9. Exportar como imagen tablas y gráfica

2.3 Lista de Reserva del Producto

Para el desarrollo del software se confecciona la Lista de Reserva del Producto, la cual define las propiedades o cualidades que el producto debe tener; son las características que lo hacen atractivo, usable, rápido y confiable. Para lograr la satisfacción del cliente y una buena calidad en el sistema se listaron los siguientes requisitos no funcionales:

Usabilidad

No es necesaria una preparación previa para utilizar el sistema, pues este puede ser usado por personas con conocimientos básicos en el trabajo con problemas de programación lineal.

Disponibilidad

El sistema estará disponible siempre que el usuario tenga instalado la máquina virtual de java en su computadora personal o en los laboratorios de docencia de la facultad.

Hardware

Para la ejecución del software se debe disponer de una computadora de 256 MB de RAM o superior, además de 500 MB de disco duro o superior.

Software

En las computadoras se requiere un sistema operativo que tenga instalada el *java development kit* (jdk) de java en su versión 8 o superior.

Interfaz de usuario

La aplicación posee una interfaz sencilla, dirigida a los estudiantes que están cursando la asignatura Investigación de Operaciones, así como a los profesores que imparten la asignatura.

2.4 Usuarios relacionados con el sistema

Los usuarios relacionados con el sistema son aquellos que interactúan de una u otra forma con la aplicación para la resolución de problemas de Programación Lineal y obtienen un resultado de todos los procesos que se ejecuten en ella. La aplicación no presenta restricciones de acceso para su uso, todos los usuarios tienen los mismos privilegios sobre las funcionalidades del sistema.

Tabla 8 Usuarios relacionados con el sistema

Usuario	Justificación
Usuario anónimo	Es cualquier usuario que hace uso de la aplicación para la resolución de problemas de programación lineal.

2.5 Fase de Exploración

2.5.1 Historias de usuario (HU)

Uno de los artefactos generados por la metodología XP son las historias de usuarios (HU), utilizadas como herramientas para dar a conocer los requerimientos del sistema al equipo de desarrollo. Son pequeños textos para describir una actividad que realizara el software. Se deben programar en un tiempo de una a tres semanas. A continuación, se muestran algunos ejemplos de las HU:

Tabla 9 Historias de Usuario (HU)

Historia de Usuario
HU1: Insertar modelo de programación lineal
HU2: Devolver Modelo Aumentado, método simplex en forma tabular y solución básica factible inicial
HU3: Resolver el problema paso a paso
HU4: Resolver el problema de forma directa
HU5: Graficar problema de dos variables
HU6: Guardar, cargar el proyecto y exportar como imagen las tablas y gráfica

Tabla 10 HU1: Insertar modelo de programación lineal

Historia de usuario	
Número: 1	**Nombre Historia de Usuario:** Insertar modelo de programación lineal
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Glenda Arbolaez Valdés Yunior Luis Velázquez Blanco	**Iteración Asignada: 1**
Prioridad en Negocio: Alto	**Puntos Estimados: 1/1**
Riesgo En Desarrollo: Bajo	**Puntos Reales:**
Descripción: Se encarga de recoger la información acerca del nombre del problema, la cantidad de restricciones y de variables, el criterio de la función objetivo y el tipo de variable.	
Observaciones:	

Tabla 11 HU2: Devolver modelo aumentado, método simplex en forma tabular y SBFI[3].

Historia de usuario	
Número: 2	**Nombre Historia de Usuario:** Devolver modelo aumentado, método simplex en forma tabular y solución básica factible inicial.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Glenda Arbolaez Valdés Yunior Luis Velázquez Blanco	**Iteración Asignada: 1**
Prioridad en Negocio: Media	**Puntos Estimados: 1/1**
Riesgo En Desarrollo: Baja	**Puntos Reales:**
Descripción: Se devuelve el modelo aumentado, la método simplex en forma tabular con la solución básica factible inicial.	
Observaciones:	

Tabla 12 HU3: Resolver el problema paso a paso

Historia de usuario	
Número: 3	**Nombre Historia de Usuario:** Resolver el problema paso a paso
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Glenda Arbolaez Valdés Yunior Luis Velázquez Blanco	**Iteración Asignada: 2**
Prioridad en Negocio: Alta	**Puntos Estimados: 3/3**
Riesgo En Desarrollo: Alto	**Puntos Reales:**
Descripción: Se encarga de la resolución del problema paso a paso, además de mostrar la tabla de iteración, la solución básica factible, la prueba de optimalidad y las variables básica saliente y no básica entrante.	
Observaciones:	

[3] Solucion básica factible inicial

Tabla 13 HU4: Resolver el problema de forma directa

Historia de usuario	
Número: 4	**Nombre Historia de Usuario:** Resolver el problema de forma directa
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Glenda Arbolaez Valdés Yunior Luis Velázquez Blanco	**Iteración Asignada: 2**
Prioridad en Negocio: Alta	**Puntos Estimados: 3/3**
Riesgo En Desarrollo: Alto	**Puntos Reales:**
Descripción: Es la que se encarga de la resolución directa del problema, además de mostrar la tabla final de iteración y la solución óptima.	
Observaciones:	

Tabla 14 HU5: Graficar problema de 2 variables

Historia de usuario	
Número: 5	**Nombre Historia de Usuario:** Graficar problema de 2 variables
Modificación de Historia de Usuario Número:	
Usuario: Glenda Arbolaez Valdés Yunior Luis Velázquez Blanco	**Iteración Asignada: 3**
Prioridad en Negocio: Alta	**Puntos Estimados: 2**
Riesgo En Desarrollo: Alta	**Puntos Reales:**
Descripción: Se muestra la solución del problema de 2 variables en forma gráfica.	
Observaciones:	

Tabla 15 HU6: Guardar, cargar y exportar como imagen el proyecto

Historia de usuario	
Número: 6	**Nombre Historia de Usuario:** Guardar, cargar y exportar como imagen el proyecto
Modificación de Historia de Usuario Número:	
Usuario: Glenda Arbolaez Valdés Yunior Luis Velázquez Blanco	**Iteración Asignada: 3**
Prioridad en Negocio: Media	**Puntos Estimados: 1/1**
Riesgo En Desarrollo: Media	**Puntos Reales:**
Descripción: Se devuelven las iteraciones en forma de imagen para que el usuario las pueda exportar, además de guardar el proyecto para una próxima utilización del mismo.	
Observaciones:	

2.6 Fase de Planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que cuesta implementar cada Historia de Usuario (HU). Este se expresa utilizando el punto como medida, el cual se considera la unidad como una semana ideal de trabajo donde los miembros del equipo de desarrollo trabajan el tiempo planeado sin interrupción alguna (32).

2.6.1 Estimación de esfuerzo

El software es especialmente difícil de estimar por lo siguiente:

1. Los requerimientos son difíciles de establecer con precisión;
2. El producto es esencialmente invisible hasta que el mismo haya sido finalizado;
3. El producto es difícil de medir (por su naturaleza intangible).

Técnicas para la estimación del esfuerzo de desarrollo de software:

1. Juicio Experto: técnica ampliamente utilizada para derivar estimaciones basadas en la pericia de expertos en proyectos similares.
2. Modelos Algorítmicos: intentan representar la relación entre el esfuerzo y una o más características del proyecto.
3. Aprendizaje de Máquina: usada en años recientes como complemento o alternativa al juicio experto y modelos algorítmicos. Entre estos aparecen: redes neuronales artificiales, razonamiento basado en casos, programación genética, lógica difusa (40).

A continuación, se presenta la estimación del esfuerzo por cada HU propuesta a partir de la técnica de Juicio de Experto:

Tabla 16 Estimación de esfuerzo por HU

Historia de Usuario	Puntos de Estimación
HU1. Insertar modelo de programación lineal	1
HU2. Devolver modelo aumentado, método simplex en forma tabular y solución básica factible inicial.	1
HU3. Resolver el problema paso a paso	3
HU4. Resolver el problema de forma directa	2
HU5. Graficar problema de dos variables	3
HU6. Guardar, cargar el proyecto y exportar como imagen las tablas y gráfica	1

2.6.2 Plan de Iteraciones

El plan de iteraciones consiste en seleccionar las historias de usuarios que corresponden a cada iteración. Se establece una división de tres iteraciones.

Iteración 1

En la iteración 1 se lleva a cabo el desarrollo de las HU uno y dos, relacionadas con la inserción del modelo de programación lineal además de la visualización del método simplex en su forma tabular con su solución básica factible inicial y el modelo aumentado.

Iteración 2

En la iteración 2 se lleva a cabo el desarrollo de las HU tres y cuatro, relacionadas con la resolución del problema de forma directa y paso a paso.

Iteración 3

En la iteración 3 se lleva a cabo el desarrollo de las HU cinco y seis, se lleva a cabo la implementación de la graficación del problema de dos variables, además del proceso de guardar, cargar el proyecto y exportar las tablas y gráfica como imagen. Al terminar esta iteración se tiene la versión 1.0 del producto final.

2.6.3 Plan de duración de las iteraciones

A continuación, se presenta el plan de iteraciones, encargado de mostrar el orden en que se implementará cada HU en cada iteración, así como la duración estimada de la misma.

Tabla 17 Plan de duración de iteraciones

Iteración	Orden de las Historias de Usuarios a implementar	Duración total
1	HU1: insertar modelo de programación lineal HU2: devolver modelo aumentado, método simplex en forma tabular y solución básica factible inicial	2 semanas
2	HU3: resolución del problema paso a paso HU4: resolución del problema de forma directa	5 semanas
3	HU5: graficación de problemas de 2 variables HU6: guardar, cargar el proyecto y exportar como imagen las tablas y gráfica	4 semanas

2.6.4 Plan de entregas

El plan de entrega define la fecha de fin de cada iteración. Se traza en función de dos parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente.

Tabla 18 Plan de entregas

Sistema	Final Iteración 1	Final Iteración 2	Final Iteración 3
Versión 1.0	14 de marzo de 2018	21 de abril de 2018	21 de mayo de 2018

2.7 Conclusiones parciales

En el presente capítulo se realizó una caracterización de la propuesta del software, lo cual facilitó la comprensión del proceso para la implementación. La confección de las HU a partir de las necesidades del cliente permitió la planeación del desarrollo del software, al lograr una estimación del tiempo para ello, dejando establecido un plan de iteraciones y entregas para el producto.

Capítulo 3: Diseño, implementación y pruebas.

En este capítulo se define cada una de las tareas de la ingeniería asociadas a cada HU con el fin de que sean implementadas todas las funcionalidades, cumpliendo con los estándares de codificación y los patrones de diseño de clases definidos. También se realizan las pruebas según la metodología, para comprobar el correcto funcionamiento del software.

3.1 Tareas de la Ingeniería

Las tareas de la ingeniería son descritas por el equipo de desarrollo a partir de las HU, las cuales describen tareas asociadas a cada HU, ofreciendo de esta manera más detalles para la implementación de las mismas y estimando el tiempo más cercano a la realidad.

Estas tareas se representan mediante tablas divididas por las siguientes secciones:

- **Número de Tarea:** los números de cada tarea deben ser consecutivos.
- **Nombre de la Tarea:** nombre que identifica a la tarea.
- **Número de Historia de Usuario:** número de la historia de usuario a la que pertenece la tarea.
- **Tipo de Tarea:** las tareas pueden ser de Desarrollo, Corrección, Mejora.
- **Puntos Estimados:** tiempo estimado en días que se le asignará a cada tarea.
- **Fecha Inicio**: fecha en que comienza la tarea.
- **Fecha Fin:** fecha en que se concluye la tarea.
- **Programadores Responsables:** nombre y apellidos de los programadores que van a desarrollar la tarea.
- **Descripción:** descripción de la tarea.

A continuación, se muestran las tareas relacionadas con la HU 6 Guardar, cargar el proyecto y exportar como imagen las tablas y gráfica, el resto pueden observarse en los anexos. (**Ver Anexo 1**)

Tabla 19 Tarea: Guardar proyecto en formato .ssfa

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 6
Nombre de Tarea: Guardar proyecto en formato .ssfa .	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 2/2
Fecha de Inicio: 15/5/18	**Fecha Fin:** 17/5/18

Programador Responsable: Yunior Luis Velázquez Blanco
Descripción: Guardar el proyecto en formato .ssfa.

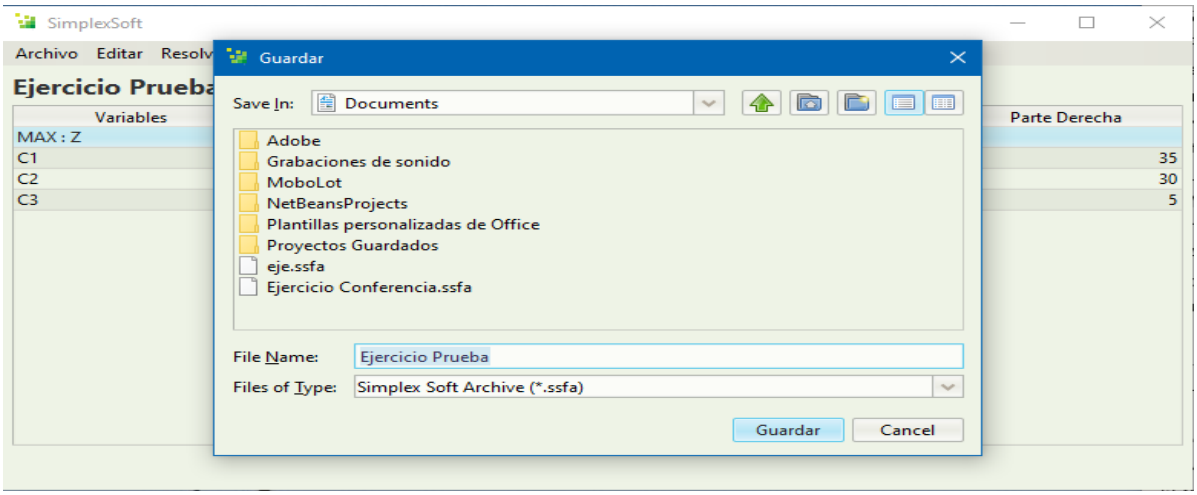


Tabla 20 Tarea: Cargar proyecto en formato .ssfa

Tarea de Ingeniería	
Número Tarea: 2	**Número Historia de Usuario:** 6
Nombre de Tarea: Cargar proyecto en formato .ssfa.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 2/2
Fecha de Inicio: 17/5/18	**Fecha Fin:** 19/5/18
Programador Responsable: Yunior Luis Velázquez Blanco	
Descripción: Cargar el proyecto en formato .ssfa	

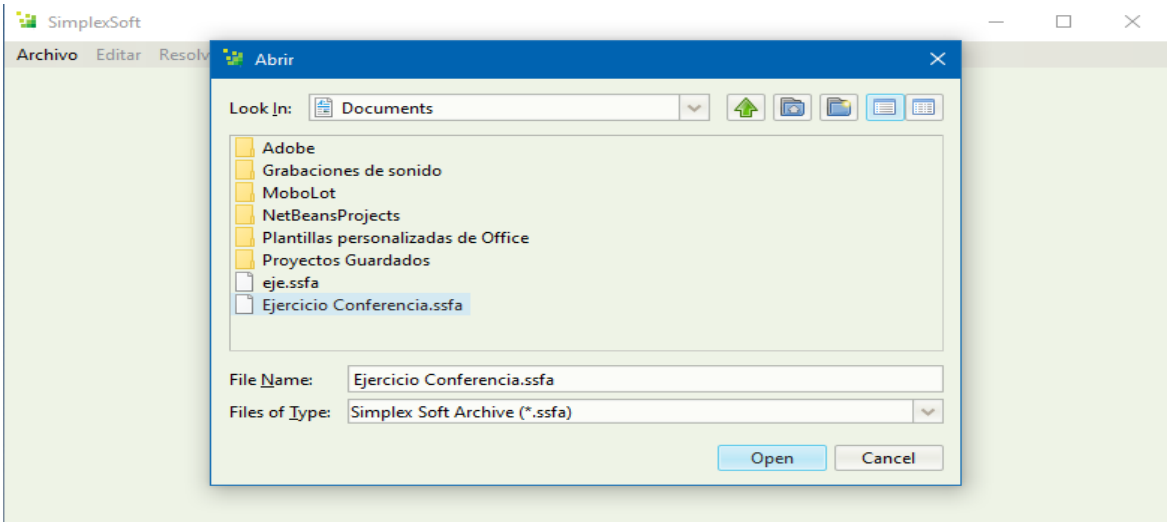
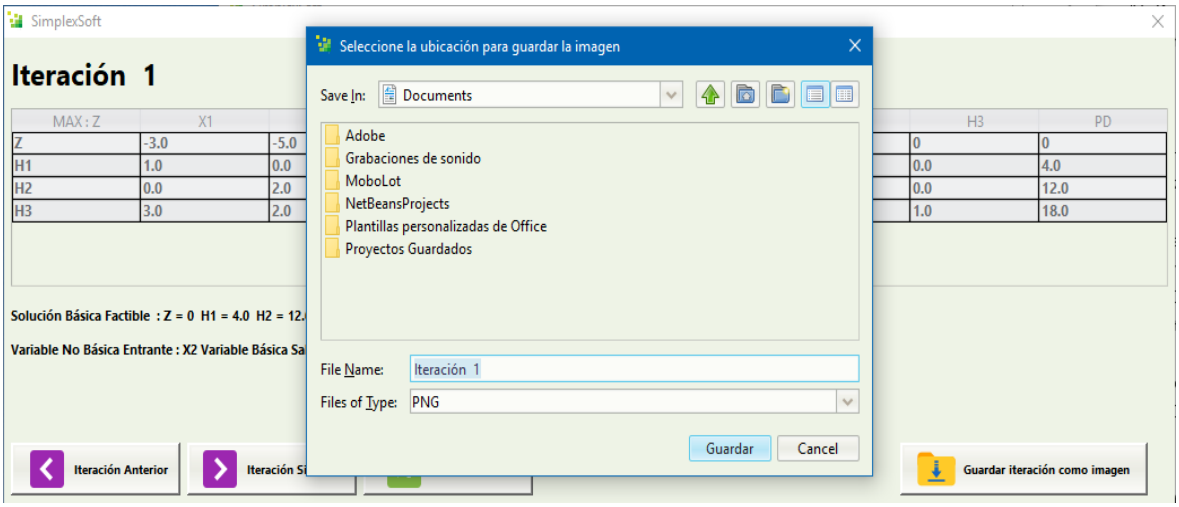


Tabla 21 Tarea: Exportar tablas y gráficas como imagen

Tarea de Ingeniería	
Número Tarea: 3	**Número Historia de Usuario:** 6
Nombre de Tarea: Exportar tablas y gráficas como imagen.	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 2/2
Fecha de Inicio: 19/5/18	**Fecha Fin:** 21/5/18
Programador Responsable: Yunior Luis Velázquez Blanco	
Descripción: Se exportan las tablas de las iteraciones como imagen, así como la gráfica.	



3.2 Tarjetas Clase – Responsabilidad - Colaborador

La metodología XP como parte de la fase de diseño propone el modelo de Clase – Responsabilidad – Colaborador (CRC), lo que permite organizar las clases más relevantes para las funcionalidades del sistema. Un modelo CRC es una colección de tarjetas índices estándar que representan clases y se analizan basándose en sus responsabilidades con respecto al sistema. Se dividen en tres secciones, a lo largo del borde superior de la tarjeta se escribe el nombre de la clase y en el cuerpo se listan las responsabilidades a la izquierda y los colaboradores a la derecha. Una responsabilidad es algo que la clase hace y los colaboradores son aquellas clases que se requieren para que una clase obtenga la información necesaria para realizar su responsabilidad. A continuación, se muestran dos tarjetas CRC, el resto de las mismas, correspondientes a las clases más relevantes de la interfaz de software, pueden observarse en los anexos. (**Ver anexo 2**)

Tabla 22 Clase: Solución

Clase: Solución	
Descripción: clase encargada de devolver la solución del problema	
Responsabilidad	**Colaborador**
Es la clase encargada de: • Generar los datos de la tabla simplex.	Pasos Pair

Tabla 23 Clase: SimplexSolver

Clase: SimplexSolver	
Descripción: clase encargada de dar solución a los problemas de programación lineal mediante el método simplex	
Responsabilidad	**Colaborador**
Es la clase encargada de: • Ejecutar el método simplex.	SolutionCallback Pasos

3.3 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos, comunicándose entre sí, adaptada para solucionar un problema en específico de diseño general. Puede identificar clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción e interfaces. También constituyen el esqueleto de las soluciones a problemas comunes en el desarrollo de software (41).

3.1.1 Patrones GRASP[4]

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos (42). Dentro de los utilizados en el desarrollo del software se encuentran los siguientes:

Patrón alta cohesión: plantea que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase. El mismo se utiliza debido a que se recomienda

[4] *General Responsibility Asignment Software Patterns* por sus siglas en inglés (Patrones generales de software para asignar responsabilidades).

tener un mayor grado de cohesión con un menor grado de acoplamiento. De esta forma se tiene menor dependencia y se especifican los propósitos de cada objeto en el sistema. A todas las clases le son asignadas responsabilidades con el objetivo de que trabajen en una misma área de aplicación y no tengan mucha complejidad (42).

Por ejemplo, la clase “Solución” es la responsable de crear una nueva instancia de la clase “Pair”.

Patrón experto: indica que la clase que cuenta con la información necesaria para cumplir la responsabilidad es la responsable de manejar la información. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantendrá encapsulada (42).

Las clases “Pasos”, “Pair” y “ModeloDatos”, poseen la información necesaria para cumplir con cada una de las responsabilidades que le corresponden.

Patrón controlador: asigna a clases específicas la responsabilidad de controlar el flujo de eventos de la aplicación. Lo anterior facilita la centralización de actividades (validaciones, seguridad, entre otros) (42).

La clase “Problema” será la responsable de atender un evento del sistema.

Patrón creador: tiene la información necesaria para realizar la creación del objeto. Es decir, que el patrón ayuda a identificar quién debe ser el responsable de la creación de nuevos objetos (42).

Por ejemplo, la clase “Solución” es la responsable de crear una nueva instancia de la clase “Pasos”.

Patrón bajo acoplamiento: indica una menor dependencia entre clases. De tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la menor repercusión posible en el resto de las clases (42).

A la clase “SimplexTableau”, le son asignadas responsabilidades de forma tal que solo se comunique con la clase “SimplexSolver”, garantizando un mínimo de dependencia, además del resto de las clases que no crean instancias de otras clases.

3.4 Estándares de codificación

Cada programador tiene su propia forma de escribir los códigos y puede ser completamente diferente a la de otros programadores, pero de la forma que se use depende de la facilidad de que otros programadores entiendan el código y se les facilite su reutilización; de ahí se desprende la importancia de los estilos de

programación; conocidos como estándares o convenciones de código los cuales definen un grupo de convenciones para escribir código fuente en ciertos lenguajes de programación. A continuación, se define la relación de estándares de codificación a utilizar en la implementación del sistema (43).

Identificadores

- Empezando con mayúscula si se trata del nombre de una clase o interfaz.
- Empezando con minúscula si es el nombre de cualquier otro identificador. De preferencia el nombre de cualquier método debe ser un verbo en infinitivo y el de todo atributo un sustantivo.

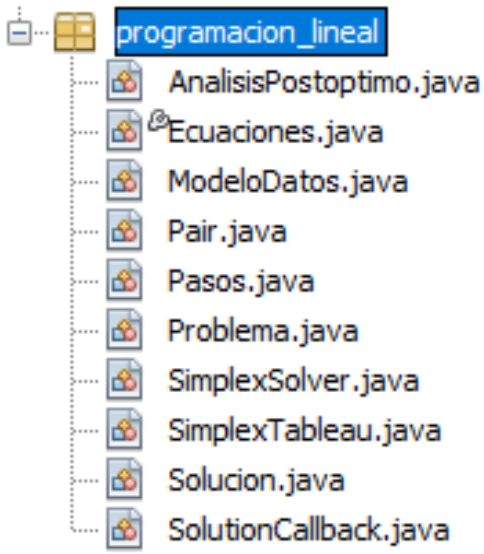


Fig. 4 Identificadores

Máxima longitud de líneas

- Todas las líneas deben estar limitadas a un máximo de cien caracteres.

```
private double getCelda(int i, int j) {
    double celda;
    if (table.getValueAt(i, j) == null) {
        celda = 0;
    } else {
        celda = Double.parseDouble(table.getValueAt(i, j).toString());
    }
    return celda;
}
```

Fig. 5 Máxima longitud de líneas

Espacios en blanco en expresiones y sentencias

- Deben rodearse con exactamente un espacio los siguientes operadores binarios:

  - ✓ Asignación (=).
  - ✓ Asignación de aumentación (+=, -=, etc.).
  - ✓ Comparación (==, <, >, >=, <=, !=, <>, in, not in, is, is not).
  - ✓ Expresiones lógicas (and, or, not).

```
public Relationship getDesigualdad(int numRest) {

    String des = table.getValueAt(numRest + 1, cant_column - 2).toString();

    if (des.equals(" >= ")) {
        return Relationship.GEQ;
    }
    if (des.equals(" <= ")) {
        return Relationship.LEQ;
    }
    return Relationship.EQ;

}
```

Fig. 6 Expresiones y sentencias

Importaciones

- Las importaciones deben estar en líneas separadas.
- Siempre deben colocarse al comienzo del archivo.
- Deben quedar agrupadas de la siguiente forma:

  - ✓ Importaciones de la librería estándar.
  - ✓ Importaciones terceras relacionadas.
  - ✓ Importaciones locales de la aplicación/librerías.

- Cada grupo de importaciones debe estar separado por una línea en blanco.

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import javax.swing.DefaultCellEditor;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.JTable;

import org.apache.commons.math3.exception.TooManyIterationsException;
import org.apache.commons.math3.optim.PointValuePair;
import org.apache.commons.math3.optim.linear.LinearConstraint;
import org.apache.commons.math3.optim.linear.LinearConstraintSet;
import org.apache.commons.math3.optim.linear.LinearObjectiveFunction;
import org.apache.commons.math3.optim.linear.NoFeasibleSolutionException;
import org.apache.commons.math3.optim.linear.Relationship;
import org.apache.commons.math3.optim.linear.UnboundedSolutionException;
import org.apache.commons.math3.optim.nonlinear.scalar.GoalType;
```

Fig. 7 Importaciones

Comentarios

- Los comentarios deben ser oraciones completas.
- Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con mayúscula.
- Si un comentario es corto el punto final puede omitirse.

```
//Función que asignará el tipo de campo que asignaste previamente.
@Override
public Class getColumnClass(int columnIndex) {
    return types [columnIndex];
}
```

Fig. 8 Comentarios

Variables y Constantes

- No definir más de una variable por línea.

```
private int cant_row;
private int cant_column;
private double[][] restriccion;
private double[] func_obj;
private double[] parte_derecha;
```

Fig. 9 Variables y constantes

Alineación y espacios en blanco

- Separar las funciones de alto nivel y definiciones de clases con dos líneas en blanco.
- Las definiciones de métodos dentro de una clase deben separarse por una línea en blanco.
- Todos los bloques deben estar alineados de tal manera que sean fácilmente distinguibles. Dentro de un bloque todas las instrucciones van a la misma estructura.

```
@Override
public int getRowCount() {
    return dataVector.size();
}

@Override
public int getColumnCount() {
    return columnIdentifiers.size();
}

@Override
public void setColumnCount(int columnCount) {
    columnIdentifiers.setSize(columnCount);
    justifyRows(0, getRowCount());
    fireTableStructureChanged();
}
```

Fig. 10 Alineación y espacios en blanco

3.5 Pruebas

Durante esta etapa se prueban todos los componentes del producto tanto por el cliente como por el equipo de desarrollo, con el objetivo de medir la calidad del software. El proceso de pruebas está encaminado a medir el cumplimiento de las funcionalidades establecidas por el cliente, reduciendo de esta manera el número de errores no detectados. Esto es propio de la metodología XP y se efectúa en la fase de Prueba, dividiéndose en dos grupos:

- Pruebas unitarias: encargadas de verificar el código diseñado por los programadores.
- Pruebas de aceptación o pruebas funcionales: tienen como objetivo evaluar si al final de una iteración están correctas las funcionalidades requeridas por el cliente.

3.5.1 Pruebas unitarias

Estas pruebas no generan artefactos, aunque resultan importantes con el fin de disminuir las no conformidades al finalizar el software. Además, resulta necesario que durante la implementación del sistema cada desarrollador tenga que ir probando constantemente lo que va realizando para garantizar que las funcionalidades exigidas por el cliente estén bien desarrolladas.

Las pruebas unitarias son una de las piedras angulares de XP, se realizan para controlar el funcionamiento de pequeñas porciones de código. Generalmente son realizadas por el mismo programador, debido a que al conocer con mayor detalle el código, se le simplifica la tarea de elaborar conjuntos de datos de prueba para testearlo (44).

Se recomienda que estas pruebas sean automatizadas, pues esto facilita al programador identificar funciones que no ofrecen una salida acorde con la lógica que se deseaba implementar. La herramienta JUnit permite correr un conjunto de pruebas de forma automática e informa de aquellas que han fallado. A continuación, se representa una de las funcionalidades del sistema a la cual se le realiza una prueba unitaria:

```
public Object[][] generarDatos() {
    Object[][] datos = new Object[cantrest + 1][cantvar + 3];
    //poner antes de la z si se max o es min
    for (int i = 1; i <= cantrest; i++) {
        if (criterio == 0) {
            datos[0][0] = "MAX: Z";
        } else {
            datos[0][0] = "MIN: Z";
        }
        datos[i][0] = "C" + i;

    }
    return datos;
}
```

Fig. 11 Caso de prueba método generarDatos

Una vez realizada la corrección de los errores detectados, los métodos de cada clase se ejecutaron de forma correcta, obteniendo de ellos los resultados esperados. Se tomaron precauciones para evitar errores

similares y se definieron nuevos casos de prueba para verificar el correcto desempeño de las funcionalidades.

3.5.2 Pruebas de aceptación

El principal propósito de este tipo de pruebas es verificar los requisitos del sistema, enfocándose en medir sus características generales y funcionalidades. Son creadas a partir de las HU, llegando a realizarse todas las pruebas necesarias que garanticen el correcto funcionamiento de la misma. En las pruebas se especifican, desde la perspectiva del cliente, los diferentes escenarios para que una HU haya sido implementada correctamente. Al finalizar debe garantizarse que los requerimientos han sido cumplidos y que el sistema es aceptable.

Una prueba de aceptación es como una caja negra, donde cada una de ellas representa una salida esperada del sistema y es responsabilidad del cliente verificar la corrección de las pruebas. La realización de estas y la publicación de los resultados debe ser lo más rápido posible, para que los desarrolladores puedan realizar con rapidez los cambios necesarios. Las pruebas de aceptación correspondiente a cada una de las funcionalidades de la interfaz de software se representan mediante tablas dividas en las siguientes secciones:

Clases válidas: se hace la descripción de los pasos efectuados para la realización de la prueba, se tiene en cuenta las entradas válidas que hace el usuario con el fin de obtener el resultado esperado.
Clases inválidas: se hace la descripción de los pasos efectuados para la realización de la prueba, se tiene en cuenta cada una de las entradas inválidas o erróneas que hace el usuario con el fin de ver el resultado que se obtiene y cómo reacciona el sistema, para que posteriormente pueda corregirse el problema presentado.
Resultado esperado: se hace una breve descripción del resultado que se espera para cualquier tipo de entrada.
Resultado de la prueba: se hace una breve descripción del resultado de la prueba que se obtiene.
Observaciones: algún comentario que el probador desea hacerle a la sección que se prueba.

En los anexos (**Ver Anexo 3**) se pueden observar algunas de las pruebas relacionadas con los requisitos planteados por el cliente. Las pruebas que evidenciaron no conformidades fueron comprobadas nuevamente después de haber sido resueltas por los desarrolladores, quedando el cliente satisfecho con el resultado de las mismas. También se cuenta con un acta de conformidad con el producto firmada por el cliente.

Para evaluar la solución se realizaron dos iteraciones donde se probó el software íntegramente, finalmente se realizó una tercera iteración donde se comprobó la resolución de todas las no conformidades detectadas. Encontrando en dichas pruebas en la primera iteración un total de doce no conformidades, de las cuales se resolvieron las doce; en la segunda iteración se detectaron nueve no conformidades, de las cuales las nueve fueron resueltas, en una tercera iteración se encontraron tres no conformidades, donde una de ellas no procede, resolviéndose el resto, y en la cuarta iteración no se encontraron no conformidades, quedando evidenciado el cumplimiento del objetivo general propuesto como solución a los problemas existentes. A continuación, se presentan los resultados arrojados durante las iteraciones de las pruebas aplicadas a través de un gráfico de barras:

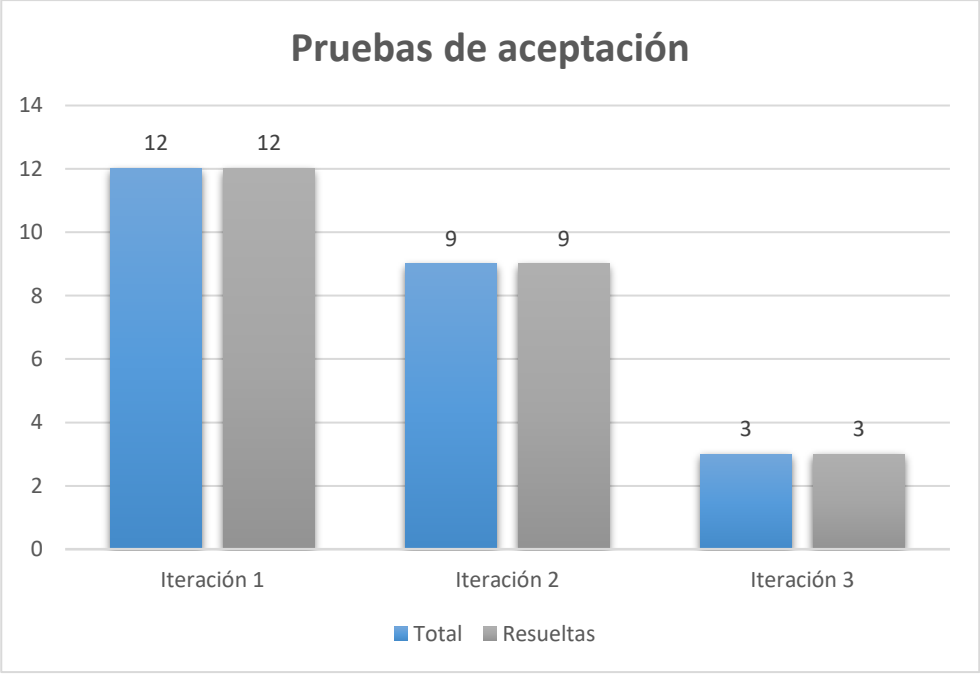


Fig. 12 Pruebas de aceptación

3.6 Conclusiones parciales

La selección de las tareas ingeniería asociadas a las HU, la confección de las tarjetas Clase-Responsabilidad-Colaborador y los patrones de diseño facilitó la construcción de la interfaz de forma ágil como plantea la metodología. También, las pruebas realizadas al sistema, demostraron la eficacia del software en cuanto a su funcionamiento y cumplimiento de los requisitos del cliente.

Conclusiones generales

Con la realización del presente trabajo de diploma se ha cumplido con el objetivo general propuesto, así como con las tareas de la investigación, arribándose a las siguientes conclusiones:

- La caracterización de los principales fundamentos teóricos-metodológicos relacionados con el objeto de estudio, contribuyeron al desarrollo del software para la resolución de problemas de programación lineal. El empleo de las herramientas, metodologías y tecnologías seleccionadas posibilitaron que la implementación del sistema se realizara exitosamente, cumpliendo de esta forma con el objetivo general.

- Con el estudio de las soluciones similares se comprobó que no cumplen con los parámetros que se consideraron a tener presente en software, para que facilite el proceso de enseñanza-aprendizaje de la asignatura Investigación de Operaciones.

- Con el diseño e implementación de las funcionalidades propuestas se obtuvo una aplicación informática que responde a las necesidades del cliente definidas durante la etapa de análisis.

- La validación del software desarrollado a partir de las pruebas unitarias y aceptación definidas por la metodología XP, demostró que los resultados obtenidos tras su ejecución son correctos.

RECOMENDACIONES

A continuación, se enumeran las siguientes recomendaciones para posibles mejoras:

- Agregar los intervalos de optimalidad y factibilidad para el análisis postóptimo.
- Implementar la resolución de los problemas de variables entera pura y binaria.

Referencias Bibliográficas

1. MADARIAGA FERNANDEZ, Carlos Jesús; ORTIZ ROMERO, Gemma Margarita; CRUZ ALVAREZ, Yamilia Bárbara y LEYVA AGUILERA, Juan José. Validación del Software Educativo Metodología de la Investigación y Estadística para su generalización en la docencia médica. *ccm* [online]. 2016, vol.20, n.2 [citado 2018-05-25], pp. 225-236. Disponible en: <http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1560-43812016000200002&lng=es&nrm=iso>. ISSN 1560-4381.
2. Colectivo de Autores; Pedagogía. Editorial Pueblo y Educación. La Habana, 2004.
3. Revista 23: Volumen 9, 2009 «La formación inicial y permanente de los docentes de la ETP». [en línea], [2009]. [Consulta: 28 mayo 2018]. Disponible en: http://revistavarela.uclv.edu.cu/index.php/numeros-de-la-revista/49-revista-23-la-formacion-inicial-y-permanente-de-los-docentes-de-la-etp.
4. GATES, Bill, et al. Camino al futuro. Santafé de Bogotá: McGraw-Hill, 1995.
5. TCHOUNIKINE, P.: Computer Science and Educational Software Design. A Resource for Multidisciplinary Work in Technology Enhanced Learning. 2011. ISBN 978-3-642-20002-1
6. TAHA, Hamdy. A. Investigación de Operaciones. Séptima Edición. México: Pearson Educación, 2004.
7. WITENBERG, Juan Prawda. Métodos y modelos de investigación de operaciones. Editorial Limusa, 1999.
8. NAMAKFOROOSH, Mohammad Naghi. Investigación de operaciones. 1985.
9. MOSKOWITZ, Herbert; WRIGHT, Gordon P.; OTALVARO, Fernando Valencia. Investigación de operaciones. Prentice Hall, 1982.
10. HILLIER, F. S. y LIEBERMAN, G. J. Introducción a la investigación de operaciones (9 ed.). Mexico: McGraw-Hill, 2010
11. SALAZAR LÓPEZ, Bryan. Método Gráfico. [2017] Disponible en: http://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigación-de-operaciones/método-gráfico/
12. DE LA FUENTE, J. L. Técnicas de cálculo para sistemas de ecuaciones, programación lineal y programación entera. Editorial Reverté, SA, Barcelona, 1998.
13. Conferencia 3: Introducción a los métodos de solución de problemas de Programación Lineal., 2017. 2017. Investigación de Operaciones. S.l.: s.n
14. TÉCNICAS DE LAS VARIABLES ARTIFICIALES. In: [online]. [Accessed 28 mayo 2018]. Available from: http://www.itlalaguna.edu.mx/academico/carreras/industrial/invoperaciones1/U2C.html.
15. Programación lineal - hiru. In: [online]. [Accessed 28 mayo 2018]. Available from: https://www.hiru.eus/es/matematicas/programacion-lineal.

16. CASTELLANOS SABARÍ, Yidian Yosbel. Sistema de tareas para la Gestión del conocimiento en el proceso de enseñanza aprendizaje de la Matemática Discreta I [en línea]. Tesis presentada en opción al título académico de Máster en Ciencias Matemáticas. Facultad de Matemática y Computación: Universidad de La Habana, 2015. Disponible en: https://repositorio_institucional.uci.cu/jspui/bitstream/ident/9039/1/Yidian%20Yosbel%20Castellanos%20Sabar%C3%AD.pdf.
17. MADARIAGA FERNÁNDEZ, Carlos & Peña, Yasnalla & Leyva Tellez, Arquimedes Rene. (2015). Evaluación de Software Educativos.
18. AREA MOREIRA, Manuel. Introducción a la Tecnología Educativa. Manual Electrónico [en línea], 2009. Disponible en: https://campusvirtual.ull.es/ocw/file.php/4/ebookte.pdf
19. CÁRDENAS, Isabel Rivero, ZERMEÑO, Marcela Georgina Gómez y TIJERINA, Raúl Fernando Abrego. Tecnologías educativas y estrategias didácticas: criterios de selección. In: *Revista educación y tecnología*. 2013, no. 3, pp. 190-206.
20. Tecnología Educativa. In: *calameo.com* [online]. [Accessed 30 mayo 2018]. Available from: https://www.calameo.com/read/0054440759ea3898302a1.
21. NODARSE, Francisco A. Fernández; MONTENEGRO, Sylvia Lima. Herramientas Computacionales en el Aprendizaje de las Matemáticas: Asistentes y Tutoriales. IE Comunicaciones: Revista Iberoamericana de Informática Educativa, 1999, no 15, p. 2.
22. CURBELO MENA, Pedro Pablo et al. Software educativo de Morfofisiología con enfoque interdisciplinario para tercer año de Licenciatura en Enfermería. Rev EDUMECENTRO [online]. 2013, vol.5, n.2 [citado 2018-05-28], pp. 172-186. Disponible en: <http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2077-28742013000200016&lng=es&nrm=iso>. ISSN 2077-2874
23. RUBIN, H. Martínez. Manual de uso del WINQSB, vol. 1. México: Instituto Tecnológico de Tepic, 2010.
24. Solver Technology - Linear Programming and Quadratic Programming. In: solver [online]. 10 enero 2011. [Accessed 29 mayo 2018]. Available from: https://www.solver.com/linear-quadratic-technology. Linear Programming and Quadratic ProgrammingFrontline Systems' optimizers solve linear programming (LP) and quadratic programming (QP) problems using these methods.
25. JSimplex Resolver problemas de Programación Lineal Online. In: [online]. [Accessed 19 October 2017]. Available from: http://ingenieria-industrial.net/software/jsimplex.
26. Modelado de problemas [Internet]. [citado 25 de abril de 2018]. Disponible en: http://www.phpsimplex.com/teoria_modelado_problemas.htm
27. Descripción del producto MATLAB - MATLAB & Simulink - MathWorks América Latina. [en línea], [sin fecha]. [Consulta: 12 febrero 2018]. Disponible en: https://la.mathworks.com/help/matlab/learn_matlab/product-description.html.

28. What is R? [en línea], [sin fecha]. [Consulta: 28 mayo 2018]. Disponible en: https://www.r-project.org/about.html.
29. PRESSMAN, R. S. Ingeniería del Software UIn enfoque práctico, Séptima edición ed. 2010.
30. BOERAS VÁZQUEZ, Mairelys, et al. 2012. Aplicando el método de Boehm y Turner. La Habana: s.n., 2012.
31. FIGUEROA, Roberth G.; SOLÍS, Camilo J.; CABRERA, Armando A. Metodologías Tradicionales vs. Metodologías Ágiles. Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación,2008.
32. LETELIER, Patricio, PENADÉS, M Carmen. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Valencia: Universidad Politécnica de Valencia. 15 Abril 2006. [Accedido 17 diciembre 2017].
33. Principales Características de Java. [en línea], [sin fecha]. [Consulta: 11 enero 2018]. Disponible en: http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm.
34. Maven Repository: org.apache.commons » commons-math3 » 3.6.1. [en línea], [sin fecha]. [Consulta: 3 abril 2018]. Disponible en: https://mvnrepository.com/artifact/org.apache.commons/commons-math3/3.6.1.
35. Math – Commons Math: The Apache Commons Mathematics Library. [en línea], [sin fecha]. [Consulta: 3 abril 2018]. Disponible en: http://commons.apache.org/proper/commons-math/.
36. Maven Repository: org.jfree » jfreechart » 1.0.19. [en línea], [sin fecha]. [Consulta: 3 abril 2018]. Disponible en: https://mvnrepository.com/artifact/org.jfree/jfreechart/1.0.19.
37. Bienvenido a NetBeans y www.netbeans.org, Portal del IDE Java de Código Abierto. [en línea], [sin fecha]. [Consulta: 11 enero 2018]. Disponible en: https://netbeans.org/index_es.html.
38. Apache License and Distribution FAQ. [en línea], [sin fecha]. [Consulta: 26 mayo 2018]. Disponible en: https://www.apache.org/foundation/license-faq.html.
39. *Documentación web de MDN* [en línea], [sin fecha]. [Consulta: 28 mayo 2018]. Disponible en: https://developer.mozilla.org/es/docs/Glossary/LGPL.
40. LÓPEZ MARTÍN, Cuauhtémoc. Estimación del esfuerzo de desarrollo de software. [2007]. Disponible en: http://www.cimat.mx/Eventos/seminariodetecnologias/EstimacionDelEsfuerzo.pdf
41. GARCÍA PEÑALVO, Francisco José, CONDE GONZÁLEZ, Miguel Ángel y BRAVO MARTÍN, Sergio. *Ingeniería de Software: diseño orientado a objetos.* Salamanca: Universidad de Salamanca, 2008.
42. LARMAN, Craig. UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2003.
43. LÓPEZ GAONA, Amparo. Guía de estilos para codificación en Java. [2014]. Disponible en: http://hp.fciencias.unam.mx/~alg/normas/estilo.html

44. MALFORÁ, Dayvis, y otros. Testing en eXtreme Programing. 2006.
45. Método Simplex - Resuelva en línea sus modelos de Programación Lineal utilizando el Método Simplex [Internet]. [citado el 9 de noviembre de 2017]. Disponible en: http://www.programacionlineal.net/simplex.html
46. VARGAS AVILÉS, J.R., *Guías Prácticas de Investigación de Operaciones (parte I)*. Universidad Nacional de Ingeniería Sede UNI-NORTE: s.n, 2008.
47. CHIAPPE, Andrés y SANCHEZ, Jorge O..Computer Education: Nature and Perspectives of an Interdiscipline. REDIE [online]. 2014, vol.16, n.2 [citado 2018-06-01], pp.135-151. Disponible en: <http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1607-40412014000200009&lng=es&nrm=iso>. ISSN 1607-4041.

Anexos

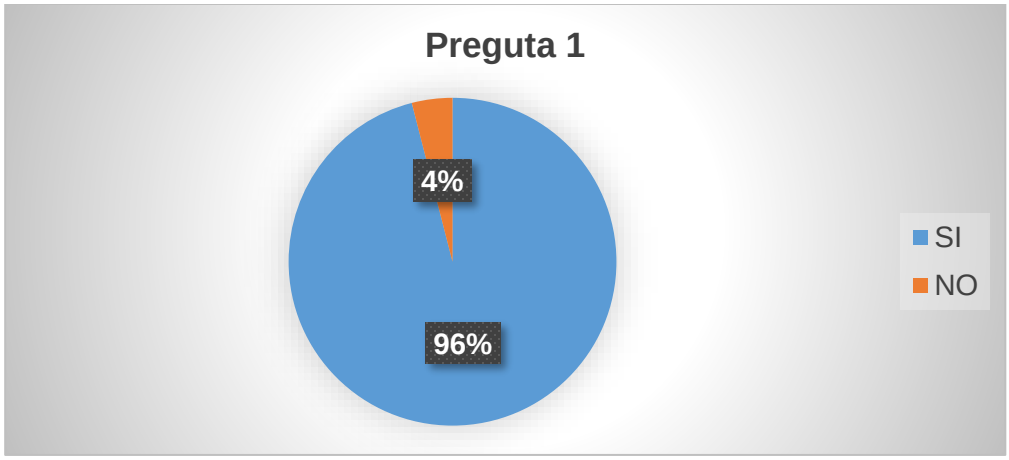
Anexo 4: Preguntas y resultados de las encuestas aplicadas

Encuesta aplicada a estudiantes de Tercer Año para comprobar la situación existente referente a la utilización del software WinQSB en la facultad. Gracias por su colaboración.

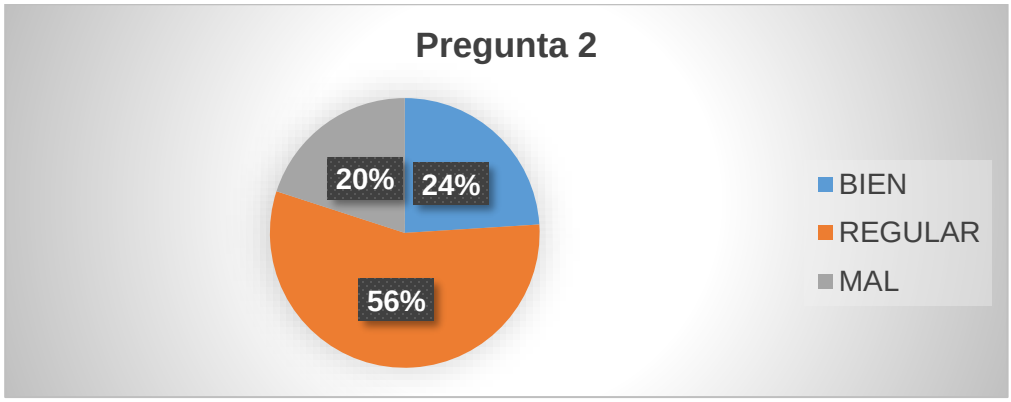
1. ¿Ha utilizado el software WinQSB? SI__ NO__
2. ¿Cómo evalúas la aplicación? Buena__ Regular__ Mala__
3. ¿Se le ha cerrado la aplicación inesperadamente? SI__ NO__
4. En el aula la prueba de optimalidad del método Simplex es contraria a la del software, pues estas tablas tienen n formato diferente. ¿Le resulta complejo entender así el contenido? SI__ NO__ ¿Prefiere que tengan el mismo formato? SI__ NO__
5. ¿Le gustaría poder disponer de otra aplicación que pueda ser ejecutada en cualquier computadora, indistintamente del sistema operativo que posea? SI__ NO__

Resultados de la encuesta aplicada a los estudiantes:

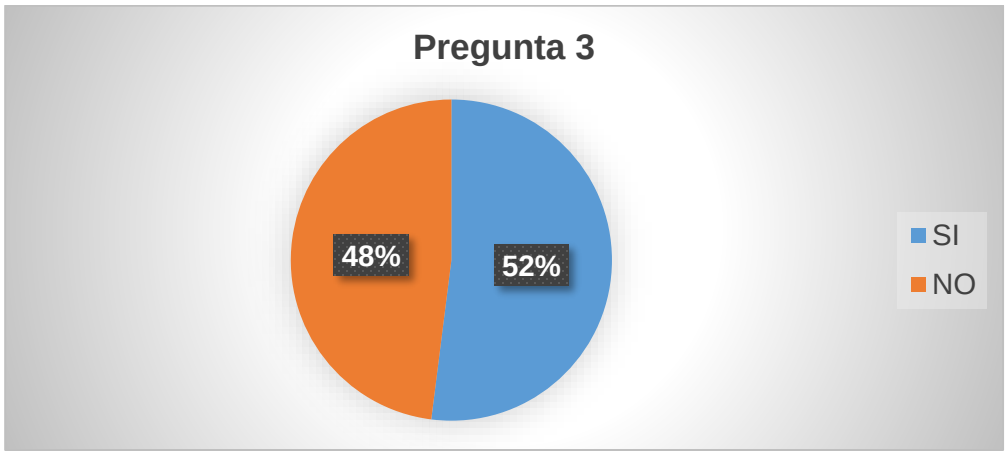
Pregunta 1:



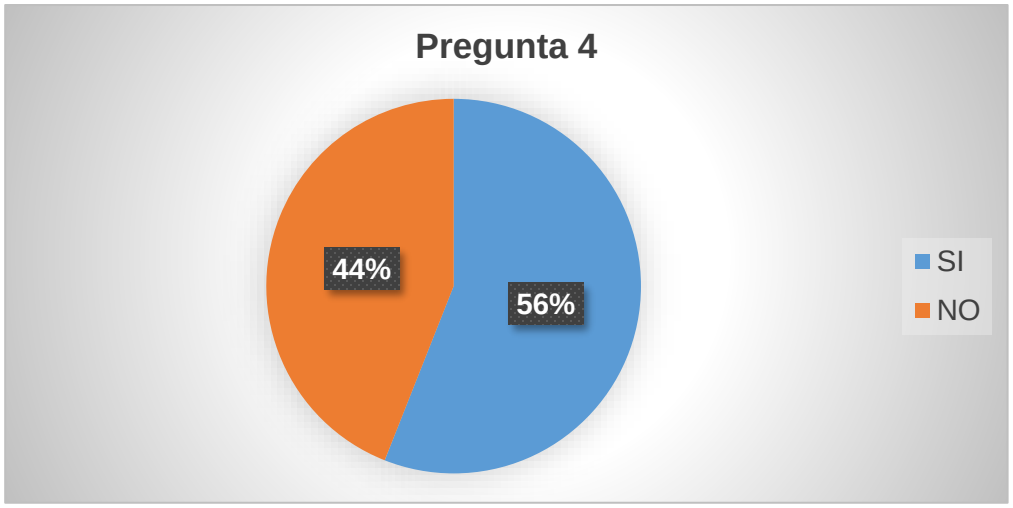
Pregunta 2:



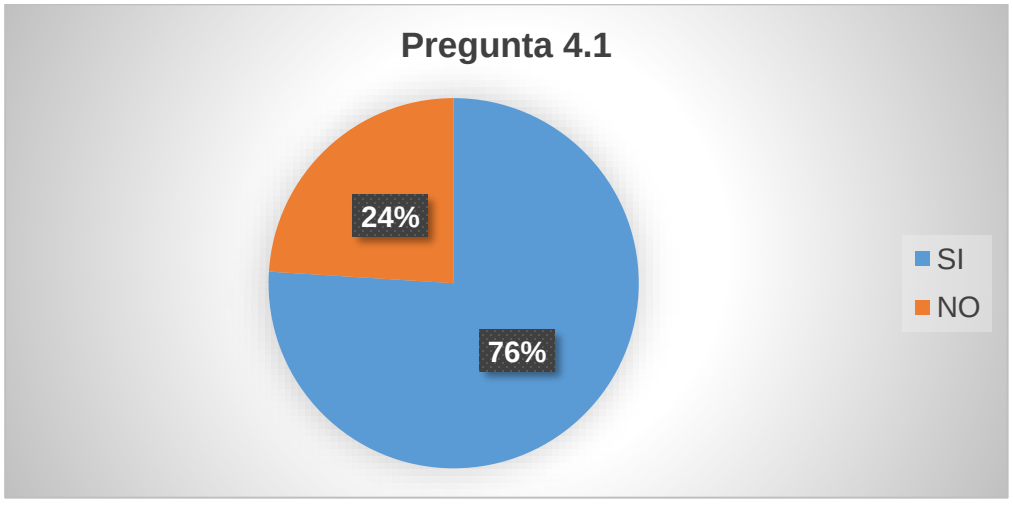
Pregunta 3:



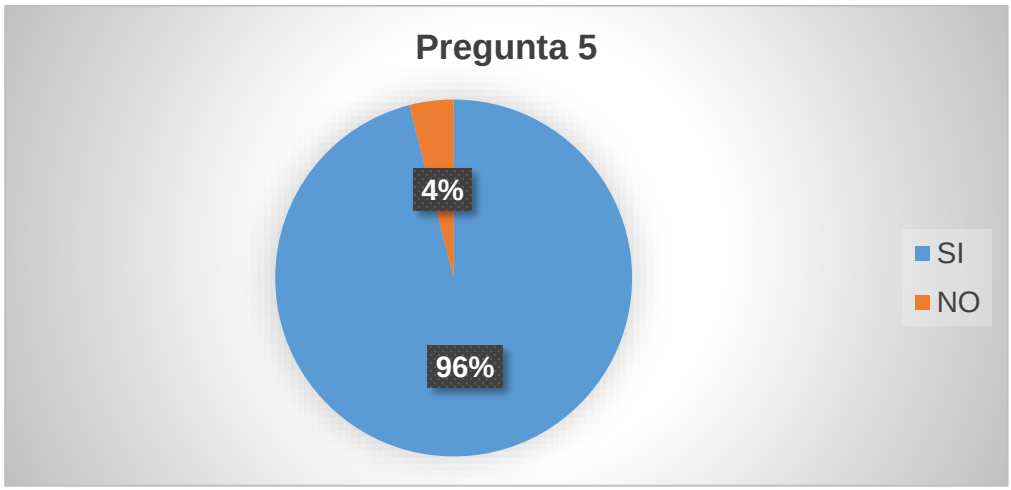
Pregunta 4:



Pregunta 4.1:



Pregunta 5:



Encuesta aplicada a profesores que han impartido la asignatura Investigación de Operaciones:

1. ¿Ha utilizado el software WinQSB? SI__ NO__
2. ¿Cómo evalúas la aplicación? Buena__ Regular__ Mala__
3. ¿Le gustaría poder disponer de otra aplicación que pueda ser ejecutada en cualquier computadora, indistintamente del sistema operativo que posea? SI__ NO__
4. ¿Considera usted que la aplicación “SimplexSoft” facilita el proceso de enseñanza-aprendizaje del tema Programación Lineal en la asignatura de IO? SI__ NO__
5. ¿Considera usted este software importante para la impartición de la asignatura en cuanto a la resolución de problemas de programación lineal? SI__ NO__
6. SimplexSoft es el primer software para sustituir en un futuro el WinQSB en la asignatura IO en la UCI. ¿Considera esto necesario e importante? SI__ NO__

Resultados de la encuesta a profesores:

1. En la primera pregunta los 4 profesores han utilizado el software WinQSB.
2. En la segunda pregunta los 4 profesores evalúan de buena a la aplicación WinQSB.
3. En la tercera pregunta los 4 profesores dijeron que si le gustaría disponer de otra aplicación.
4. En la cuarta pregunta 3 profesores dijeron que la aplicación SimplexSoft facilita el PEA y un 1 profesor que no.
5. En la quinta pregunta los 4 profesores consideran que el software SimplexSoft es importante para la impartición de la asignatura.
6. En la sexta pregunta los 4 profesores consideran a la aplicación SimplexSoft necesaria e importante.

Anexo 5: Comparación del método tabular del simplex

Método tabular del simplex realizado de forma manual en clases

VB	X1	X2	X3	H1	H2	S1	A1	PD
Z	3.33	3.66	0	0	0	0.33	M-0.33	1
H1	1.33	0.66	0	1	0	0.33	-0.33	0
H2	2	1	0	0	1	0	0	3
X3	0.66	0.33	1	0	0	-0.33	0.33	1

Método tabular presentado por el WinQSB

		X1	X2	X3	Slack_C1	Slack_C2	Surplus_C3	Artificial_C3		
Basis	C(j)	4,0000	4,0000	1,0000	0	0	0	0	R. H. S.	Ratio
Slack_C1	0	1,3333	0,6667	0	1,0000	0	0,3333	-0,3333	0	
Slack_C2	0	2,0000	1,0000	0	0	1,0000	0	0	3,0000	
X3	1,0000	0,6667	0,3333	1,0000	0	0	-0,3333	0,3333	1,0000	
	C(j)-Z(j)	3,3333	3,6667	0	0	0	0,3333	-0,3333	1,0000	
	* Big M	0	0	0	0	0	0	1,0000	0	

- Una diferencia es que la ecuación de la función objetivo Z no aparece en el renglón (0), sino que está representada en los dos últimos renglones, que aparecen con los identificadores ($C_J - Z_j y * BigM$).
- Cuando la función objetivo es de maximizar el valor de Z, se escoge como variable básica entrante, la variable que más le aporte a la función objetivo, que, en este caso, es la que tenga el mayor coeficiente positivo.
- Cuando la función objetivo es de minimizar el valor de Z, se escoge como variable básica entrante, la variable que menos le aporte a la función objetivo, que en este caso, es la que tenga el coeficiente negativo de mayor valor modular.