Universidad de las Ciencias Informáticas

Facultad 3

Trabajo de Diploma para optar por el título de Ingeniero en ciencias informáticas

Título: Herramienta de apoyo a la asignatura Matemática IV.

Autores:

Michel Sariol Fernández.

Lián Hernández García.

Tutores:

MSC. Yuneiry Barroso Pedroso.

Lic. Sandy Díaz Ramos.

Co-tutor:

Ing. Leonardo Cabrera Nicolau.

La Habana, junio de 2015

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo de diploma y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos el presente a los ____ días del mes de _____ del año _____.

_____ _____

Michel Sariol Fernández Lián Hernández García

_____ _____

MSC. Yuneiry Barroso Pedroso Lic. Sandy Díaz Ramos

_____

Ing. Leonardo Cabrera Nicolau

DATOS DE CONTACTO

MSC. Yuneiry Barroso Pedroso: ybp@uci.cu

Se ha desempeñado como profesor de las asignaturas Matemática I, Matemática II, Probabilidades y Estadísticas, Investigación de Operaciones, Optimización Matemática III, Matemática IV, y Álgebra Lineal en la Universidad de las Ciencias Informáticas. Ha ejercido como tutor, oponente y miembro de tribunal en varias tesis de grado. Posee varias publicaciones científicas de carácter nacional e internacional.

Lic. Sandy Díaz Ramos: sdiaz@uci.cu

Licenciado en Matemática, Universidad de la Habana 2005. Profesor asistente.

Se ha desempeñado como profesor de las asignaturas Matemática II, Matemática III, Matemática IV y Álgebra Lineal en la Universidad de las Ciencias Informáticas. Ha ejercido como tutor, oponente y miembro de tribunal en varias tesis de grado. Posee una inscripción de software a título personal, más otras dos como parte del proyecto EIDMAT. Ha impartido el curso de postgrado *"Latex, tipografía para textos científicos y técnicos"* y vencido varios cursos comprendidos en el plan de la Maestría de Matemática con mención en Matemática Numérica de la Facultad de Matemática y Computación de la Universidad de la Habana. Cuenta con varias publicaciones en eventos internacionales y actualmente trabaja en un proyecto de doctorado.

Ing. Leonardo Cabrera Nicolau: nicolau@uci.cu

Ingeniero en Ciencias Informáticas, UCI 2014. Recién graduado en adiestramiento.

Se ha desempeñado como Programador en el proyecto Sistema de Informatización de los Tribunales Populares de Cuba, habiendo certificado los roles de Probador y Programador en nivel Avanzado. Cuenta con resultados relevantes en diferentes ediciones de las Olimpiadas Nacionales Universitarias de Matemática. Se desempeñó por tres cursos como Alumno Ayudante de las asignaturas Matemática III y Matemática IV en la Universidad de las Ciencias Informáticas y actualmente se desempeña como profesor de las mismas. Posee diferentes publicaciones científicas de alcance nacional e internacional.

DEDICATORIA

Michel

Agradezco a mis tutores por no permitir la mediocridad, por sabernos guiar por este camino angosto en búsqueda de la universalización del pensamiento, siempre estará su ejemplo en mí, siempre tratare de ser digno portador de su legado. Quiero agradecer también a mi compañero Lian, con quien he trabajado duro, sin quien no hubiera conseguido el mismo nivel de completitud la herramienta. Por sobre todas las cosas debo agradecer a mis padres y a mi hermanita, por quienes hago todo esperando únicamente que sientan orgullo de mí

Lián

Le agradezco a mis tutores que tanto nos apoyaron y ayudaron, a mi compañero de tesis y casi mi hermano, el miche, que ha soportado todas mis pesadeces durante los 5 años, a mi familia que durante todo el camino siempre me han apoyado, mi mamita y mis hermanos, a mis amigos que gracias a ellos he llegado a sentirme en familia, a mis profesores, en fin, a todos los que aportaron en mi formación como profesional.

RESUMEN

Entre las asignaturas que se imparten en el segundo año de la carrera Ingeniería en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas, se encuentra la Matemática IV (Matemática Numérica). En esta asignatura, debido a los avances científico-técnicos, se hace cada vez más necesario el empleo de herramientas informáticas. Las limitaciones técnico-didácticas que están implícitas al usar las herramientas existentes en la asignatura motivó el presente trabajo, que consiste en la implementación de una herramienta informática para dar solución a estas limitantes. Esta aplicación recoge la implementación de la mayoría de los métodos que se imparten en la asignatura, permitiendo su visualización desde el punto de vista numérico y gráfico, facilitando la descripción de estos y la comparación de los que resuelven un mismo tipo de problema. La solución se desarrolló para su ejecución multiplataforma y con tecnologías libres.

PALABRAS CLAVE: Asistente matemático, métodos numéricos, proceso enseñanza aprendizaje

INTRODUCCIÓN

En la Universidad de las Ciencias Informáticas (UCI) anualmente se trabaja para perfeccionar el proceso de enseñanza aprendizaje (PEA). Este perfeccionamiento está dado, entre otros motivos, por la forma en que aprenden los estudiantes, es sabido como de un curso a otro existen variaciones visibles en este sentido, e incluso en el mismo curso no todos los estudiantes aprenden de igual forma. Además, está la gran variedad de métodos y medios a disposición de los profesores.

La edición vigente de la versión digital del diccionario de la Real Academia Española (Española, 2015) define como primera acepción de la palabra *perfeccionamiento*: *"Acabar enteramente una obra, dándole el mayor grado posible de bondad o excelencia"*. Mientras, en el Plan de Estudios del Ingeniero en Ciencias Informáticas (Informáticas, 2013) para fundamentar y trazar las pautas para concebir la Disciplina de Matemática y sus asignaturas se plantea:

> *"La Matemática simbólica tradicional ha sido siempre, y lo es también hoy, un instrumento para la solución de modelos, sin embargo, pierde terreno continuamente ante el desarrollo de la computación y de los métodos numéricos. La importancia de los procedimientos de cálculo simbólico se reduce y aumenta el significado del conjunto Matemática Numérica - Computación - Inteligencia Artificial.*
>
> *Esta tendencia moderna debe reflejarse en el diseño de las asignaturas con carácter inmediato; ese reflejo se acentuará con el desarrollo computacional del país."*

Y en otro momento:

> *"Es necesario diseñar adecuadamente el modo de trabajo de la asignatura, el sistema de evaluación y el sistema de trabajo independiente, en concordancia con los fines que se han enunciado. En cualquier caso, se debe tender a:*
>
> - *Incrementar el uso de los medios de cómputo, la utilización de bibliotecas y la creación de programas sencillos."*

Además, entre los objetivos y habilidades del programa vigente de la asignatura Matemática IV se encuentran los siguientes:

" …

- *Interpretar el proceso de solución numérica iterativa como la generación de una sucesión de soluciones intermedias que converja a la solución del problema.*

- *Utilizar asistentes matemáticos y otras técnicas de computación en la solución de problemas por métodos numéricos.*

…"

Por otro lado, la Enciclopedia colaborativa *EcuRed* define el proceso de enseñanza aprendizaje como*: "una unidad dialéctica entre la instrucción y la educación; igual característica existe entre el enseñar y el aprender. Todo el proceso de enseñanza-aprendizaje tiene una estructura y un funcionamiento sistémicos, es decir, está conformado por elementos o componentes estrechamente interrelacionados"* (ecured, 2015). Uno de esos componentes son los medios, papel que juegan los asistentes matemáticos en el PEA.

Por todo lo anterior, los profesores de la asignatura Matemática IV, como sus homólogos del resto de las asignaturas, en su empeño por cumplir a cabalidad con lo estipulado en los documentos rectores, día a día tratan de perfeccionar el proceso de enseñanza aprendizaje. Entiéndase por perfeccionarlo mejorarlo, darle un mayor acabado.

No hay que perder de vista que todos estos documentos, así como los planes calendarios de las asignaturas, abundan más sobre lo que se debe impartir que en la forma de impartirlo. En este caso se limitan a dar orientaciones generales que se deben complementar con el reconocimiento por parte del profesor de las características del grupo de estudiantes. En consecuencia, el proceso antes mencionado adquiere un carácter dinámico, moldeable, que precisa del profesor para darle ese acabado, ese perfeccionamiento.

¿En qué sentido se puede lograr ese perfeccionamiento? Abordando los conceptos y métodos de la asignatura desde varios puntos de vistas, los profesores de Matemática III y IV de la Facultad 3 han experimentado como llegar a una mayor cantidad de estudiantes, haciendo

más efectivo el proceso de enseñar y aprender. Un ejemplo de esto ha sido la utilización de Matlab en el tema de series de la asignatura Matemática III. Concretamente la utilización de la herramienta Taylortool de este asistente, que permite una visualización geométrica del concepto de intervalo de convergencia. En este caso, a partir de la propia experimentación del estudiante se puede lograr una mejor comprensión del concepto, siendo capaces de reconstruirlo por si solos.

Otras herramientas que por varios años se han utilizado en la asignatura Matemática IV son: Derive, Mathematica, Maple y más recientemente Octave y Maxima. Sin embargo, el claustro de profesores de esta asignatura ha detectado varias limitaciones para su utilización en clases, entre las más importantes (Sariol Fernández, y otros, 2014):

- MatLab, Derive, Mathematica y Maple son privativas, y cómo consecuencia del bloqueo económico, comercial y financiero impuesto por el gobierno de los Estados Unidos sobre Cuba por más de medio siglo, es imposible acceder a estas licencias, siendo además muy costosas.

- El soporte para Derive está descontinuado para estaciones de trabajo.

- Octave y Maxima no presentan interfaz gráfica de usuario.

- Todos presentan una sintaxis difícil de aprender, sobre todo partiendo de que este aprendizaje debería estar dado paralelamente al transcurso de la asignatura y muchas veces de forma independiente por parte de los estudiantes, lo cual implica un escalón extra a vencer antes de estar en condiciones de dar solución a los problemas que se les planteen.

- En la mayoría de los casos, los métodos impartidos en la asignatura no están disponibles en estos asistentes, y mucho menos pueden ser manejados como se desea según el programa analítico de la asignatura.

Esta última limitación usualmente se vence a partir de implementaciones creadas por los propios profesores sobre estas herramientas. Es por ello que los profesores del colectivo de Matemática IV de la Facultad 3 se dieron a la tarea de generar una herramienta que unifique

la mayoría de los métodos numéricos que actualmente se enseñan en la asignatura, posibilitando la visualización numérica y geométrica del proceso de convergencia de los métodos y permitiendo comparar la solución de un mismo problema obtenida por métodos diferentes.

La idea inicial era utilizar las funcionalidades del asistente MatLab para la creación de interfaces visuales en la obtención de la herramienta deseada, sin embargo unido a las limitaciones antes señaladas, otras nuevas se presentan en este empeño:

- Para poder utilizar finalmente la herramienta debería estar instalado MatLab en las estaciones de trabajo, incluso estar en ejecución, lo cual implica una considerable demanda de recursos computacionales. A esto se suma las condiciones, muchas veces deficientes, de la infraestructura tecnológica de los laboratorios en los que se hará uso de la herramienta.

- Además se precisa de una herramienta ligera y portable, pues se pretende que los estudiantes puedan llevársela para utilizarla en sus computadoras personales o cualquier otra estación de trabajo fuera de los laboratorios docentes o incluso de la universidad. La instalación de MatLab además de recursos computacionales, demanda gran cantidad de memoria tanto para transportarla cómo para su instalación propiamente.

- Con la publicación cada año de una nueva versión de MatLab, en las cuales se suelen hacer cambios en la sintaxis que muchas veces implican la desaparición de funciones existentes, no es de extrañar que con la actualización de MatLab la herramienta pierda funcionalidades o incluso se vuelva inservible.

Teniendo en cuenta la situación descrita, se plantea como **problema a resolver**: las herramientas informáticas que se utilizan en la asignatura Matemática IV presentan limitaciones para el empleo de la mayoría de los métodos numéricos estudiados en la asignatura, impidiendo la visualización numérica y geométrica del proceso de convergencia, así como la comparación de las soluciones a un mismo problema obtenidas por diferentes métodos.

En correspondencia con el problema, se define como **objeto de estudio:** el proceso de desarrollo de herramientas informáticas de apoyo a la docencia. Enmarcando el **campo de acción** en: las herramientas informáticas para Matemática IV de la UCI. Defendiendo la **idea** de que: el diseño e implementación de una herramienta informática para la asignatura Matemática IV, permitirá evadir las limitaciones detectadas en la ejecución de la mayoría de los métodos numéricos estudiados en la asignatura.

Para solucionar el problema planteado, se define como **objetivo general:** desarrollar una herramienta informática que permita describir, utilizar y comparar la mayoría de los métodos numéricos que se imparten en la asignatura Matemática IV. Derivando en los siguientes objetivos específicos y tareas:

1. Formalizar el marco teórico referencial del proceso de desarrollo de herramientas informáticas para la asignatura Matemática IV.

    a. Descripción de las soluciones existentes, tendencias y tecnologías que garanticen el máximo aprovechamiento de una herramienta informática para la asignatura Matemática IV.

    b. Determinación de la metodología, herramientas y tecnologías a utilizar para el desarrollo de una herramienta informática de apoyo a la asignatura Matemática IV.

2. Obtener el modelo de diseño de la herramienta informática.

    a. Especificación y validación de los requisitos del sistema.

    b. Definición de la arquitectura del sistema.

    c. Descripción de las responsabilidades y relaciones entre clases del sistema de acuerdo a la metodología de desarrollo seleccionada.

3. Implementar la herramienta informática a partir del modelo de diseño.

    a. Formalización de una gramática libre de contexto para el reconocimiento de funciones matemáticas.

b. Implementación de las funcionalidades del sistema y las pruebas automatizadas correspondientes.

c. Descripción del uso de patrones y estándares de codificación durante la implementación de la herramienta.

4. Verificar la efectividad de la solución propuesta.

a. Ejecución de pruebas de unidad automatizadas luego de cada cambio o implementación de nuevas funcionalidades del sistema.

b. Realización de pruebas de aceptación al cliente relativas a su conformidad con la herramienta desarrollada.

Para el cumplimiento de los objetivos se consideraron los siguientes **métodos de investigación** (Hernández Sampieri, y otros, 2006)**:**

Métodos teóricos:

- **Histórico-lógico** para el análisis crítico de los trabajos anteriores que aborden el tema de la confección y uso de herramientas informáticas en apoyo al PEA en la asignatura Matemática IV.
- **Método inductivo-deductivo** para la identificación de los problemas y variantes de solución en torno al proceso de desarrollo de aplicaciones en soporte al PEA en la asignatura Matemática IV.

Métodos empíricos:

- **Entrevista** para la captura de las principales necesidades del cliente así como las deficiencias de los sistemas que se emplean como apoyo al PEA en la asignatura Matemática IV.

El presente documento se encuentra estructurado en 3 capítulos, a continuación se describe brevemente el contenido abordado en cada uno:

Capítulo 1: Fundamentación teórica

Se describen los principales conceptos asociados al dominio del problema, analizando las herramientas informáticas que se emplean en apoyo al PEA de la asignatura Matemática IV en la UCI, en cuanto a sus prestaciones y limitaciones, además se abordan las tecnologías y herramientas para el desarrollo de la solución que se propone.

Capítulo 2: Análisis y diseño

Abarca los principales elementos de la concepción del sistema propuesto, la especificación de sus funcionalidades, planificación de las iteraciones de desarrollo, arquitectura y modelación del sistema de acuerdo a la metodología elegida.

Capítulo 3: Implementación y prueba

Se describe la solución desde el punto de vista técnico, evidenciando el empleo de patrones y estándares de codificación. Además define la estrategia de pruebas para la validación de la solución propuesta según la metodología, analizando los resultados de las mismas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan algunos conceptos, que por su relación con la temática tratada es importante definir. Se describen los temas y métodos abordados en la asignatura Matemática IV y la necesidad de utilizar herramientas informáticas en apoyo a esta. Se realiza un análisis de los asistentes que hasta hoy se han utilizado en la UCI. Además se describen las herramientas, lenguajes y tecnologías de desarrollo seleccionadas para la implementación de la solución que se propone, así como los patrones y estrategias de pruebas posibles a emplear.

1.1 CONCEPTUALIZACIÓN

El contexto en el que se enmarca el presente trabajo de diploma exige que determinados conceptos sean esclarecidos con el fin de lograr el entendimiento de la solución propuesta, tal es el caso de análisis numérico, definido por Lloyd Trefethen como: *"el estudio de los algoritmos para los problemas de la matemática continua*" (Trefethen, 1992).

Por otra parte, según Manuel Álvarez Blanco, la Matemática Numérica es una rama de la Matemática en la cual el objetivo no es el estudio de un ente matemático en particular; tiene como propósito el desarrollo de métodos para la solución de los más diversos problemas matemáticos mediante una cantidad finita de operaciones numéricas. Es decir, lo que le da unidad a esta rama de la matemática no es el tipo de problema que se ha de resolver sino el método que se aplicará: operaciones numéricas en cantidad finita. El propósito aquí no es llegar a resultados exactos, ni siquiera tan exactos como sea posible. El propósito es obtener resultados tan exactos como sea necesario (Blanco, y otros, 2007).

Nótese que a pesar de ser casi idénticos los conceptos de análisis numérico y Matemática Numérica, existen sutiles diferencias entre ellos y esencialmente están dadas porque en el caso del primero hay una implicación mayor hacia el análisis en profundidad de la convergencia de los métodos, por solo citar un ejemplo.

1.2 MÉTODOS NUMÉRICOS DE LA MATEMÁTICA IV

La asignatura Matemática IV está estructurada en seis temas. Estos a su vez agrupan, respectivamente, varios métodos numéricos que tienen en común el hecho de estar concebidos para resolver un mismo problema. Por ejemplo en el Tema 2 se estudian los métodos numéricos que permiten solucionar ecuaciones no lineales. Ahí se encuentran:

1. Bisección
2. Regula Falsi
3. Iterativo general
4. Newton-Raphson
5. Secante

En la mayoría de los temas se pueden encontrar métodos que responden a distintas filosofías de cálculo. Siguiendo con el ejemplo del Tema 2 se puede ver como estos métodos se dividen en dos grupos, los de intervalo (1 y 2) y los de punto fijo (3-5), o el caso del Tema 3 en el que los métodos se dividen en directos (Método de Gauss) e iterativos (Métodos de Jacobi y Gauss-Seidel). Esta variedad, no solo de métodos sino también de tipos de métodos, permite que los estudiantes puedan realizar comparaciones y llegar a conclusiones sobre que métodos o incluso que tipos de métodos utilizar en cada caso. Para esto el estudiante debe caracterizar el problema a resolver como requisito indispensable antes de seleccionar el método a utilizar.

La mayoría de los problemas que se proponen en esta asignatura se conciben para que estén lo más cerca posible a problemas reales. Esto implica, entre otras cosas, el trabajo con grandes volúmenes de datos como un reto agregado; obligando en cada momento al estudiante a hacer un ejercicio de abstracción y concentración en el tipo de problema y método que está manejando en cada caso.

Los problemas que se pueden resolver con los métodos que se imparten en la asignatura son:

- Ecuaciones no lineales
- Sistemas de ecuaciones lineales
- Aproximar funciones
- Calcular integrales definidas
- Ecuaciones diferenciales ordinarias

Todos estos elementos, unidos a las precisiones metodológicas correspondientes, conforman la asignatura Matemática IV y a su vez ofrecen las primeras pautas para la concepción de la herramienta a implementar en este trabajo de diploma (Colectivo de autores, 2014).

1.3 ALTERNATIVAS INFORMÁTICAS DE SOPORTE AL PEA DE LA MATEMÁTICA IV, BREVE DESCRIPCIÓN TÉCNICA

El empleo de las Tecnologías de la Información y las Comunicaciones (TIC), como medio de difusión del conocimiento, constituye una alternativa que ha sido explotada en Cuba en la ejecución del PEA en todos los niveles educacionales. La inclusión en los planes de estudio de las diferentes enseñanzas, de tele-clases, aplicaciones informáticas de perfil educativo y multimedias temáticas, son evidencia de ello (de la Torre Navarro, y otros, 2012).

Un ejemplo de lo anteriormente referido es la situación descrita en la introducción, el empleo de asistentes matemáticos en la asignatura Matemática IV de la UCI, específicamente MatLab. El mismo ofrece un entorno de desarrollo integrado (IDE, por sus siglas en inglés) con un lenguaje de programación propio (lenguaje M) (Goering, 2010), disponible para las plataformas Windows, Mac OS X y Linux, en este último caso la compañía brinda un sistema emulador para su empleo. Entre sus prestaciones básicas se encuentran: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos y la creación de interfaces de usuario.

Como alternativa, Octave se considera la versión libre de MatLab (Quarteroni, 2006). Ambos permiten ejecutar órdenes en modo interactivo sobre lenguajes casi idénticos. Sin embargo, en sus colecciones de funciones no aparecen implementados la mayoría de los métodos que se imparten en la asignatura, además Octave carece de interfaz gráfica y ofrece limitadas funcionalidades para la graficación de funciones con respecto a MatLab (Octave, 2013).

Maxima es un motor de cálculo simbólico escrito en lenguaje Lisp[1], publicado bajo licencia pública general (GPL, por sus siglas en inglés) (Maxima, 2012). Cuenta con un amplio conjunto

[1] **LIS**t **P**rocessing, “proceso de listas”, lenguajes de programación de computadora de tipo multiparadigma.

de funciones para el trabajo simbólico con polinomios, matrices, funciones racionales, integración, derivación, manejo de gráficos en 2D y 3D, números de punto flotante, expansión en series de potencias y de Fourier, entre otras funcionalidades. Además tiene un depurador a nivel de código fuente. Funciona en modo consola, sin embargo existen interfaces gráficas como xMaxima y wxMaxima para facilitar su uso (Camejo Diago, y otros, 2012).

El MN2000 es una solución de desarrollo nacional que ha contribuido a la formación de varias generaciones de ingenieros en Cuba, a pesar de ser la única que recoge todos los métodos que se imparten en la asignatura de Matemática IV de la UCI, presenta limitaciones en elementos esenciales de la graficación y el trabajo con funciones y expresiones matemáticas en general, carece de facilidades para la visualización de la convergencia de los métodos estudiados y la demostración de sus hipótesis.

Otra herramienta de producción nacional es IEDMAT, desarrollado en la UCI. Este tiene como objetivo brindar una interfaz gráfica a los usuarios de Octave, lo más parecida posible a MatLab en cuanto a diseño (EIDMAT, 2012).

Otras herramientas presentan similares limitaciones y en algunos casos más, tal es el caso de Maple, Mathematica (Wolfram, 2010), Derive (Derive, 2013), Scilab (Scilab, 2012).

Aunque ninguna de las soluciones existentes se adapta a las necesidades de la UCI, todas tienen algo que aportar a la construcción de una herramienta a la medida, siendo junto a las exigencias asociadas al empleo de los métodos numéricos abordados en la asignatura, la principal fuente de requisitos de la solución que se propone (Fernández Nodarse, y otros, 2000).

1.4 ELEMENTOS A CONSIDERAR DE LAS HERRAMIENTAS ANALIZADAS

De manera general, cualquier nueva herramienta que se desarrolle debe reflejar la mayoría de los elementos positivos de herramientas precedentes concebidas para el mismo fin, en este caso los asistentes matemáticos que hoy dan soporte al PEA en la asignatura Matemática IV, es por ello que la identificación de sus bondades es fundamental.

Los autores del presente trabajo y el colectivo de la asignatura coincide en determinados elementos en este sentido, por ejemplo, el proceso de graficación debe ser dinámico y configurable, dando opciones para ampliar y disminuir el tamaño de la gráfica, resaltar ceros de las funciones representadas, graficación y diferenciación por colores de múltiples funciones en una misma gráfica, incorporando una leyenda con las funciones representadas, graficar puntos independientes y exportar gráfica como imagen.

Por otra parte estos elementos deberán estar soportados mediante un sistema que de alguna forma permita reconocer y evaluar funciones, empleando una sintaxis lo más sencilla e intuitiva posible, asociando los nombres de las funciones a la notación matemática. Esto permitirá generar un sistema de validaciones que controlen errores como la entrada de un intervalo de graficación que no pertenezca al dominio de la función o expresiones que no se correspondan con la sintaxis aceptada para el reconocimiento de funciones matemáticas.

La distribución de métodos numéricos a implementar debe hacerse de forma que no solo se separen por los tipos de problemas a los que dan respuesta sino también atendiendo a las diferentes clasificaciones que se les pueda dar, por ejemplo si son iterativos o directos. En cualquier caso se debe evitar al máximo los posibles errores, por lo que se recomienda inhabilitar opciones de las interfaces en correspondencia con las validaciones a las entradas de datos y las características de los métodos.

Otros de los elementos que sería de gran valor, desde el punto de vista didáctico, es la visualización de datos de interés para la convergencia de los métodos desde la misma introducción de las entradas, evitando pasos intermedios que dificulten la operatividad de la herramienta, tal es el caso del determinante de la matriz principal de un sistema de ecuaciones o el condicionamiento del mismo.

La generación de matrices y vectores de interés es otro elemento a tener en cuenta, no solo por la facilidad para completar datos de entrada de forma rápida y efectiva en lugar de introducirlos elemento por elemento, sino porque posibilita la ejecución de los métodos sobre datos en los que el profesor conoce previamente el comportamiento de las soluciones (convergencia rápida, lenta o falta de convergencia). Esto permitiría generar matrices especiales para la ejemplificación de diferentes elementos abordados en las clases, por ejemplo que garanticen la convergencia de determinados métodos sin cumplir con las

condiciones suficientes de convergencia de los mismos, permitiendo preparar al estudiante para enfrentarse a situaciones que incluso trascienden el marco de la asignatura.

Adicionalmente, se hace necesaria la visualización de elementos particulares y/o parciales de las soluciones como las matrices L, U y P, resultado de la factorización de la matriz principal de un sistema de ecuaciones lineales en el proceso de resolución según el método de Gauss, por solo citar un ejemplo.

La interpretación geométrica es una pieza clave para la compresión de los métodos numéricos, razón por la cual debe estar presente en todos los módulos que lo permitan, por ejemplo:

- El proceso de convergencia de los diferentes métodos de resolución de ecuaciones no lineales.
- El concepto de condicionamiento de sistemas de ecuaciones lineales, siempre que sus dimensiones lo permitan.
- La visualización geométrica del método de los trapecios para la resolución de integrales definidas.
- El campo de direcciones de una ecuación diferencial ordinaria.
- La ejecución de los métodos de solución de ecuaciones diferenciales ordinarias.

Los elementos anteriores son fundamentales para demostrar los conceptos de aproximación y convergencia.

Además, sería de gran utilidad la posibilidad de guardar y cargar desde fichero juegos de datos para la resolución de problemas, incluyendo la opción de especificar una descripción para los mismos. En esta descripción el profesor podría poner la orientación de una tarea o ejercicio de laboratorio (ver Anexo 1). Esta opción implica controlar que no se pueda cargar los datos de un método en otro que no le corresponda.

1.5 METODOLOGÍA, LENGUAJES Y HERRAMIENTAS DE DESARROLLO

En todos los proyectos de desarrollo de software un paso fundamental es la elección de la metodología, lenguajes y herramientas de desarrollo, a continuación se describen los principales argumentos considerados para la ejecución del presente trabajo de diploma.

1.5.1 METODOLOGÍA DE DESARROLLO

En cada proyecto de desarrollo de software la metodología constituye la columna vertebral del todo el proceso. Se define como metodología a: *"una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo*" (López Quesada, 2005).

Dadas las características del equipo de desarrollado, integrado por dos miembros, con clientes en constante comunicación con los mismos y formando parte del equipo, además de la prontitud con la que se desea desarrollar la solución, se elige una metodología con enfoque ágil, dando paso a la elección de la Programación Extrema (XP por sus siglas en inglés) por ser una de las más conocidas y ampliamente usadas de esta clasificación, además de la experiencia del equipo en el desarrollo con la misma (Corbea Rodríguez, 2007).

La Programación Extrema potencia las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen ambiente de trabajo.

La misma propone las siguientes fases (Kent Beck, 2000):

Planificación: se elabora el plan de iteraciones a ejecutar durante el desarrollo del proyecto a partir de la prioridad de los requisitos especificados en las historias de usuario y el esfuerzo necesario para su implementación.

Diseño: se define el diseño del sistema a partir de la modelación de sus clases en tarjetas de Clase Responsabilidad Colaboración (CRC), además se definen los glosarios de términos, una

correcta especificación de los nombres de métodos y clases ayudará a comprender el diseño y facilitará sus posteriores ampliaciones y la reutilización del código.

Codificación: se desarrolla la implementación de cada historia de usuario, la presencia del cliente en este momento es sumamente importante dado que ellos son los que crean las historias de usuario y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario el cliente debe especificar detalladamente su funcionamiento y estará presente cuando se realicen las pruebas que verifiquen que la implementación se corresponde con la funcionalidad especificada. La codificación debe hacerse ateniendo al uso de patrones y estándares de codificación, lo cual mantiene el código consistente y facilita su comprensión y escalabilidad.

Pruebas: uno de los pilares de la metodología XP es la ejecución constante de pruebas para comprobar el funcionamiento de los códigos implementados. Se definen dos niveles de pruebas: pruebas de unidad y pruebas de aceptación, las pruebas de unidad están enfocadas a validar la solución desde el punto de vista técnico, a evaluar el código generado, en tanto las pruebas de aceptación las realiza el cliente para verificar que las funcionalidades implementadas cumplen con los requisitos acordados. Un punto importante de las pruebas de unidad es su automatización, además se deben diseñar antes que las funcionalidades a probar, garantizando la independencia del código a evaluar.

1.5.2 LENGUAJE DE MODELADO Y HERRAMIENTA CASE

Se selecciona el Lenguaje Unificado de Modelado (UML por sus siglas en inglés) por ser el más conocido y empleado para especificar, visualizar, construir y documentar artefactos de un sistema de software. El mismo ofrece un estándar para describir el sistema, además de poseer características y propiedades que lo validan como el más aceptado (Language, 2013).

Como herramienta de soporte a la ingeniería de software (CASE por sus siglas en inglés) se selecciona Visual Paradigm que además de soportar el lenguaje UML, resulta uno de los más adecuados para ingenieros de software, analistas y arquitectos interesados en la construcción de sistemas garantizando confiabilidad y estabilidad en el desarrollo orientado a objetos.

Visual Paradigm posee un área de trabajo que brinda opciones y utilidades al usuario, su forma de organización de los diagramas es muy cómoda y organizada. Además es multiplataforma

y se integra con varios entornos de desarrollo integrado como NetBeans (Visual Paradigm, 2013), cuya elección se fundamenta en el siguiente epígrafe.

1.5.3 LENGUAJE DE PROGRAMACIÓN Y ENTORNO DE DESARROLLO INTEGRADO

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; permitiendo al programador comunicarse con los dispositivos hardware y software existentes (Definition.org, 2011).

Para el desarrollo en cuestión solo se considerarán los lenguajes de alto nivel y orientados a objetos por resultar más parecidos al lenguaje humano y manejar los conceptos de manera similar al pensamiento humano, abstrayéndose del funcionamiento de la máquina. Además ahorran tiempo al programador y le proveen mayores facilidades a la hora de programar.

Considerando la envergadura de la solución y el reducido tiempo de que se dispone para la implementación, así como el elevado nivel de familiarización del equipo de desarrollo con el lenguaje Java y la capacidad plena de este lenguaje para satisfacer las necesidades de implementación, se recurre al mismo para el desarrollo de la solución.

Debido a que el entorno de ejecución objetivo de la herramienta será los laboratorios de docencia de la UCI y la versión de la máquina virtual de java allí instalada es la 1.6 se tomará dicha versión para el desarrollo de la aplicación.

Otras características importantes para la implementación de la herramienta son la posibilidad de uso de bibliotecas de código abierto tanto de la propia arquitectura estándar de Java como otras desarrolladas por terceros, lo cual permite heredar las funcionalidades de clases existentes y personalizarlas, posteriormente empaquetar todo y distribuir el producto de manera libre. Además de ser un lenguaje multiplataforma, robusto y seguro (Oracle, 2013).

Para suplir la necesidad de un ambiente de programación para el lenguaje Java que incluya un editor de código, un compilador, un depurador y un constructor de interfaz gráfica, a lo cual usualmente se denomina entorno de desarrollo integrado (IDE por sus siglas en inglés) (Encyclopedia., 2013), se selecciona Netbeans IDE dado al alto nivel de familiarización del

equipo de desarrollo con el mismo. Según su sitio oficial **www.netbeans.org**: *"es una herramienta que permite a los programadores escribir, compilar, depurar y ejecutar programas en java. Existe además un número importante de módulos para extender sus funcionalidades, mientras el código fuente está disponible para su reutilización"* (Netbeans, 2013).

Otro aspecto importante en la elección de este IDE es su condición de producto libre y gratuito, sin restricciones de uso, además permite la integración con diferentes bibliotecas de código libres y de uso necesario para el desarrollo de la herramienta.

1.5.4 BIBLIOTECAS DE CÓDIGO DE TERCEROS

Para el desarrollo de la solución se hará uso de las siguientes bibliotecas de código de terceros, libres y especializadas en funcionalidades específicas:

- **Biblioteca jep-2.4.1**: es una biblioteca realizada sobre Java que analiza y evalúa expresiones matemáticas, permite a los usuarios introducir expresiones como texto, siendo compatible con las variables definidas por el usuario, constantes y funciones. También provee funcionalidades para la derivación de funciones (JEP).

- **Biblioteca jFreeChart 1.0.19**: es una biblioteca libre, realizada 100 % sobre Java que facilita a los desarrolladores la capacidad de mostrar gráficos de calidad profesional en sus aplicaciones (JFreeChat).

- **Marco de trabajo jUnit 4.10:** el objetivo del uso de jUnit con Netbeans IDE es escribir y ejecutar pruebas fácilmente repetibles. jUnit es una instancia de la arquitectura xUnit para marcos de trabajo encargados de realizar pruebas unitarias, fue desarrollado por el autor de la metodología XP y está orientado a la realización de pruebas de unidad en el lenguaje Java por parte del programador (Manning, 2005).

- **Jama 1.0.2: Jama**, creada por el NIST (National Institute for Standards and Technology) y MathWorks (autores de Matlab). Fue diseñada para servir como la librería estándar de tratamiento de matrices en Java (jama, 2014).

1.6 PATRONES PARA EL DESARROLLO DE SOFTWARE

Un patrón define un par problema/solución aplicable en nuevos contextos. Los patrones tienen como objetivo capturar y establecer explícitamente conocimiento abstracto de resolución de problemas, que suele ser implícito y se adquiere solo a través de la experiencia (LARMAN, 1999).

La bibliografía estudiada define los patrones de implementación como patrones de bajo nivel específicos de un lenguaje de programación. Describen cómo implementar aspectos particulares de componentes o sus relaciones, utilizando las características de un lenguaje determinado. Representan el nivel más bajo de patrones, manejan aspectos tanto de diseño como de implementación y son mayormente utilizados durante la fase de desarrollo del sistema (Pressman, 2005).

En este sentido, las “buenas prácticas de la programación” definidas por la autoproclamada “banda de los cuatro” de la ingeniería de software (GOF por sus siglas en inglés) crean un lenguaje común para comunicar ideas y experiencias acerca de problemas comunes y sus soluciones (mindview, 2012), definiendo un conjunto de patrones creacionales, estructurales y de comportamiento.

Por otro lado, existe otra tendencia de patrones de implementación muy común, incluso de forma inconciente e intuitiva por los programadores, relacionada con la asignación de responsabilidades dentro de las implementación (GRASP por sus siglas en inglés) (Patrones-GRASP, 2013).

1.7 ESTRATEGIA DE PRUEBAS

La metodología XP propone el empleo de 2 niveles de pruebas para la validación de la solución: pruebas de unidad y de aceptación, las cuales se realizan durante todo el ciclo de desarrollo de software (Kent Beck, 2000).

Las pruebas de unidad son una forma de probar el correcto funcionamiento de un módulo o una parte del sistema. Con el fin de asegurar que todos los módulos cumplan con lo requerido cada uno por separado y evitar así errores futuros en el momento de la integración de todas

sus partes. La idea es escribir casos de prueba para cada función no trivial o método en el módulo, de forma que cada caso sea independiente del resto (Manning, 2005).

Las pruebas de aceptación son pruebas definidas por el cliente para cada historia de usuario, con el objetivo de asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. En efecto marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia donde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención (Malfará, y otros, 2006).

Existen dos grupos de pruebas de aceptación definidos por la metodología XP: alfa y beta:

Las pruebas alfa las realiza el usuario en un entorno controlado por el equipo de desarrollo, mientras los desarrolladores registran los errores y problemas detectados (Pressman, 2005).

Las pruebas beta se aplican en el lugar y condiciones reales en que será ejecutado el software, en ausencia de los desarrolladores. El usuario final registra los problemas detectados (reales o imaginarios) y los informa regularmente a los desarrolladores (Pressman, 2005).

Estas pruebas deben realizarse con la mayor agilidad posible, para que los desarrolladores puedan realizar los cambios necesarios y que el tiempo de desarrollo no se vea afectado gravemente.

1.8 CONCLUSIONES PARCIALES

Con el desarrollo del presente capítulo queda constituido el marco teórico referencial de la investigación. Se analizaron las herramientas informáticas empleadas en apoyo al PEA en la asignatura Matemática IV, cada una de las cuales a pesar de presentar funcionalidades limitadas para su empleo acorde con los objetivos de la asignatura tiene algo que aportar a la confección de una nueva herramienta a la medida. Se describieron los criterios de selección de la metodología, lenguajes y herramientas de desarrollo, así como la importancia del uso de patrones y la estrategia de pruebas según la metodología.

CAPÍTULO 2: ANALISIS Y DISEÑO DE LA SOLUCIÓN

En el presente capítulo se realiza la descripción de la solución propuesta acorde con la metodología de desarrollo seleccionada. Se describe la etapa de levantamiento de requisitos, su especificación y validación, se presenta la planificación de las iteraciones de desarrollo, la arquitectura del sistema y las responsabilidades y relaciones entre clases, terminando con la validación del diseño propuesto.

2.1 LEVANTAMIENTO DE REQUITOS

La ingeniería de requisitos constituye un elemento fundamental en todo proceso de desarrollo de software que tenga como finalidad la obtención de sistemas informáticos que se ajusten a las necesidades reales de los clientes, tiene como objetivo establecer los servicios que el sistema deberá proveer y las restricciones bajo las cuales deberá ser desarrollado este (Báez, y otros, 2001).

De conjunto con el cliente, se definió como principal fuente de requisitos las exigencias presentes en los objetivos de la asignatura asociadas al empleo de los métodos numéricos, así como el estudio de las soluciones que hasta el momento se han utilizado para dar soporte al PEA de la Matemática IV en la UCI. Esto se complementó con la realización de tormentas de ideas para contribuir al carácter didáctico de la herramienta.

2.1.1 REQUISITOS FUNCIONALES

Los requerimientos o requisitos funcionales (RF) de un software son la descripción de los servicios proporcionados por el mismo y sus restricciones operativas. Reflejan las necesidades de los clientes resueltas mediante un sistema informático (Sommerville, 2005). A continuación se relaciona el listado de requisitos convenidos con el cliente:

RF1: Introducir función.

RF2: Recoger datos para resolución de métodos.

RF3: Reconocer funciones reales.

RF4: Mostrar errores léxicos en tiempo real.

RF5: Mostrar errores de sintaxis en tiempo real.

RF6: Evaluar funciones reales.

RF7: Seleccionar módulo.

RF8: Elegir método de resolución.

RF9: Ejecutar método de Bisección para aproximar una raíz de una ecuación no lineal.

RF10: Graficar proceso de convergencia del método de Bisección paso a paso.

RF11: Ejecutar método de Regula-Falsi para aproximar una raíz de una ecuación no lineal.

RF12: Graficar proceso de convergencia del método de Regula-Falsi paso a paso.

RF13: Ejecutar método iterativo general para aproximar una raíz de una ecuación no lineal.

RF14: Graficar proceso de convergencia del método Iterativo General paso a paso.

RF15: Ejecutar método de Newton-Raphson para aproximar una raíz de una ecuación no lineal.

RF16: Graficar proceso de convergencia del método de Newton-Raphson paso a paso.

RF17: Ejecutar método de Secantes para aproximar una raíz de una ecuación no lineal.

RF18: Graficar proceso de convergencia del método de Secantes paso a paso.

RF19: Mostrar el condicionamiento del sistema de ecuaciones lineales gráficamente.

RF20: Ejecutar método de Jacobi para aproximar el vector solución de un sistema de ecuaciones lineales.

RF21: Mostrar secuencia de aproximaciones del método de Jacobi.

RF22: Ejecutar método de Gauss-Seidel para aproximar el vector solución de un sistema de ecuaciones lineales.

RF23: Mostrar secuencia de aproximaciones del método de Gauss-Seidel.

RF24: Ejecutar método de Gauss con factorización LU para aproximar el vector solución de un sistema de ecuaciones lineales.

RF25: Mostrar descomposición de matriz principal del sistema en matrices L y U

RF26: Mostrar matriz de permutaciones P para la obtención de la factorización LU.

RF27: Mostrar cálculo de determinante de matriz principal de un sistema de ecuaciones lineales.

RF28: Mostrar cálculo del condicionamiento de un sistema de ecuaciones lineales.

RF29: Generar matriz principal del sistema a partir de expresiones de generación.

RF30: Generar vector de términos independientes a partir de expresiones de generación.

RF31: Ejecutar método de Lagrange para determinar un polinomio interpolador de una función a partir de un conjunto de puntos.

RF32: Graficar conjunto de polinomios básicos de Lagrange.

RF33: Graficar polinomio interpolador resultante del método de Lagrange.

RF34: Ejecutar método de Newton para determinar un polinomio interpolador de una función a partir de un conjunto de puntos.

RF35: Graficar polinomio interpolador resultante del método de Newton.

RF36: Graficar error de interpolación del método del método de Newton.

RF37: Ejecutar método de Mínimos Cuadrados para ajustar un modelo lineal a un conjunto de puntos.

RF38: Graficar modelo de ajuste obtenido por el método Mínimos Cuadrados.

RF39: Ejecutar método de Simpson para aproximar el valor de una integral definida.

RF40: Ejecutar método de Romberg para aproximar el valor de una integral definida.

RF41: Ejecutar método de Trapecios para aproximar el valor de una integral definida.

RF42: Graficar trapecios en el intervalo de integración.

RF43: Ejecutar método de Euler para aproximar la solución de una ecuación diferencial en un intervalo dado.

RF44: Ejecutar método de Runge-Kutta orden 2 para aproximar la solución de una ecuación diferencial en un intervalo dado.

RF45: Ejecutar método de Runge-Kutta orden 4 para aproximar la solución de una ecuación diferencial en un intervalo dado.

RF46: Graficar solución de un problema de Cauchy según el método de Euler en un intervalo dado.

RF47: Graficar solución de un problema de Cauchy según el método de Runge-Kutta de orden 2 en un intervalo dado.

RF48: Graficar solución de un problema de Cauchy según el método de Runge-Kutta de orden 4 en un intervalo dado.

RF49: Graficar campo de direcciones de un problema de Cauchy asociado a una ecuación diferencial.

RF50: Graficar funciones reales.

RF51: Exportar gráficos como imagen.

RF52: Configurar resolución de gráficos.

RF53: Resaltar ceros de las funciones graficadas.

RF54: Limpiar las funciones graficadas.

RF55: Guardar en ficheros ejemplos de los métodos matemáticos.

RF56: Cargar en ficheros ejemplos de los métodos matemáticos.

RF57: Mostrar ayuda integrada del sistema.

2.1.2 REQUISITOS NO FUNCIONALES

Los requisitos no funcionales, como su nombre sugiere, no se refieren directamente a las funciones específicas que proporciona el sistema, sino a sus propiedades emergentes como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa, definen restricciones del producto, como la capacidad de los dispositivos de entrada/salida y las representaciones de datos que se utilizan en las interfaces del sistema (Sommerville, 2005).

Diferentes autores definen sus propias clasificaciones y sub-clasificaciones de requisitos no funcionales, para el presente trabajo de diploma se hará uso de los criterios definidos en el estándar *IEEE-STD-830-1998: ESPECIFICACIONES DE LOS REQUISITOS DEL SOFTWARE* (IEEE, 1998), que en su sección de requisitos no funcionales establece:

- Requisitos de funciones del producto.
- Requisitos de rendimiento
- Restricciones de diseño.
- Atributos del sistema.
- Otros requisitos.

A continuación se relacionan los requisitos no funcionales (RNF) identificados con sus respectivas clasificaciones:

Restricciones de diseño:

RNF1 El sistema debe ofrecer una interfaz amigable, intuitiva y fácil de operar. Además, debe mantener una línea de diseño para alcanzar la uniformidad en la solución final.

Atributos del sistema:

RNF2 El sistema debe ser portable o de escritorio, ya que como se define en la situación problemática debe ejecutarse con total independencia de conectividad.

Otros requisitos:

RNF3 El análisis de la base tecnológica presente en los laboratorios docentes, arrojó que el nuevo sistema deberá operar con prestaciones mínimas de hardware:

1. Procesador: 1.6 GHz. o superior.
2. Memoria RAM: 512 MB o superior.
3. Disco duro: 40 GB o superior.

RNF4 Para la ejecución del sistema, con independencia del sistema operativo, constituye premisa indispensable la instalación del Entorno de ejecución de Java (JRE por sus siglas en inglés), versión 1.6 o superior.

2.1.3 ESPECIFICACIÓN Y VALIDACIÓN DE REQUISITOS

Las historias de usuario 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 funcionalidades del sistema.

El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es suficientemente

comprensible y delimitada para que los programadores puedan implementarla en unas semanas (Canos, y otros, 2011).

A continuación se presentan 3 de las historias de usuario especificadas, el resto puede ser consultada en el Anexo 2**:**

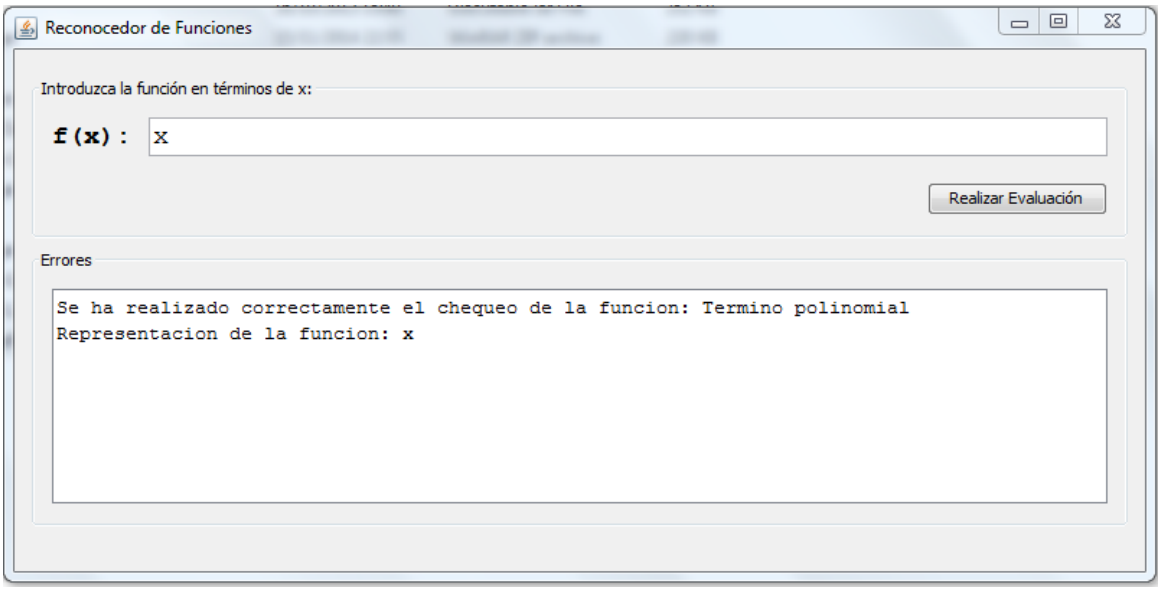
<table>
<tr><th colspan="2">Historia de usuario</th></tr>
<tr><td>Numero_3</td><td>Nombre de historia de usuario: Reconocer funciones reales.</td></tr>
<tr><td>Actor: Usuario</td><td>Iteración asignada: 1</td></tr>
<tr><td>Prioridad de negocio: Alta</td><td>Puntos estimados: 21</td></tr>
<tr><td>Nivel de complejidad: Alta</td><td>Puntos reales: 21</td></tr>
<tr><td colspan="2">Descripción: Una vez que el usuario introduce una función en el campo “función”, el sistema debe identificar los errores de sintaxis que cometidos por el usuario.</td></tr>
<tr><td colspan="2">Observaciones: El campo “función” no puede estar vacío.</td></tr>
<tr><td colspan="2">Prototipo de interfaz:
 
</td></tr>
</table>

Tabla #1 Historia de usuario: Reconocer funciones reales

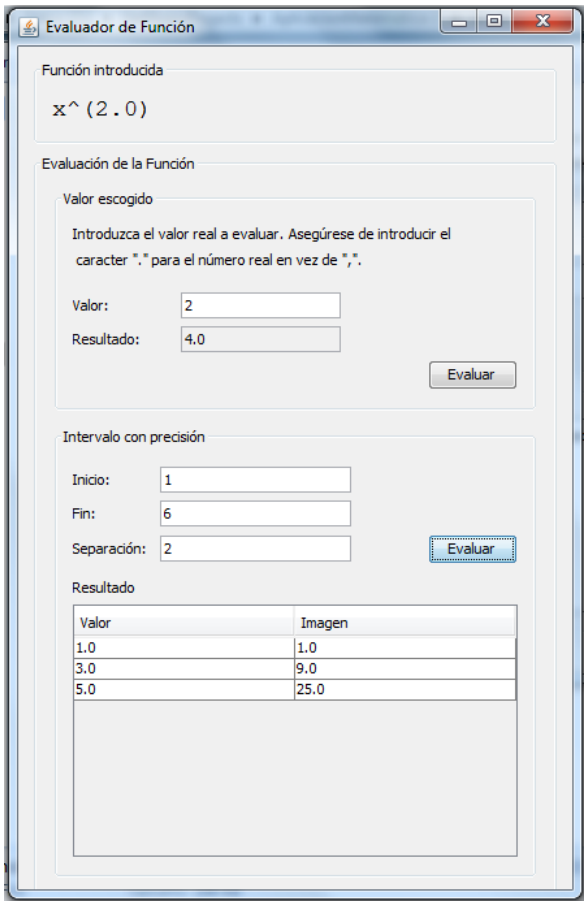
Historia de usuario	
Numero_6	**Nombre de historia de usuario:** Evaluar funciones reales.
Actor: Usuario	**Iteración asignada:** 1
Prioridad de negocio: Alta	**Puntos estimados:** 13
Nivel de complejidad: Alta	**Puntos reales: 1**3
Descripción: Una vez que el usuario introduce un valor o intervalo a evaluar acciona el botón “evaluar” y se muestran sus respectivas evaluaciones en una tabla.	
Observaciones: El campo “función” no puede estar vacío.	
Prototipo de interfaz: 	

Tabla #2 Historia de usuario: Evaluar funciones reales.

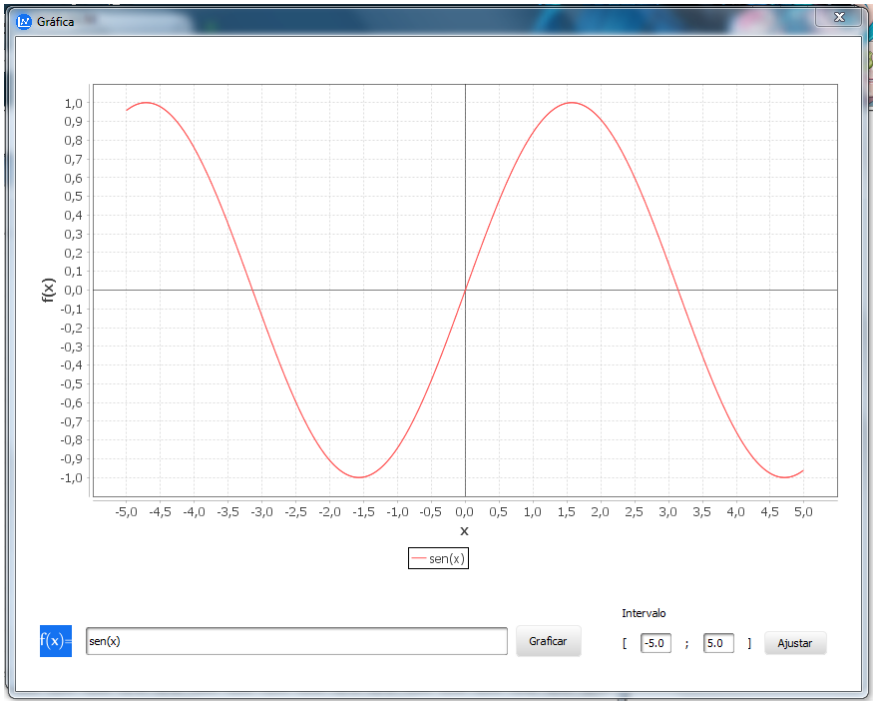
Historia de usuario	
Numero_49	**Nombre de historia de usuario:** Graficar funciones reales.
Actor: Usuario	**Iteración asignada:** 1
Prioridad de negocio: Alta	**Puntos estimados:** 13
Nivel de complejidad: Alta	**Puntos reales:** 13
Descripción: El usuario debe llenar el campo f(x) e introduce el intervalo que crea conveniente accionar el botón aceptar, en el panel principal se dibujara la función indicada.	
Observaciones: El campo "función" no puede estar vacío, el botón graficar debe estar habilitado indicando que la función es reconocida.	
Prototipo de interfaz: 	

Tabla #3 Historia de usuario: Graficar funciones reales.

Una vez especificados los requisitos, es importante proceder a su validación, verificando la correspondencia con las descripciones iniciales y revisando si responden a lo realmente deseado. En algunos casos fue necesario construir prototipos de las interfaces de usuario con

funcionalidades reducidas para que el cliente se hiciera una idea aproximada del resultado, no siendo esta técnica (Prototipado) suficiente para comprobar la ausencia de uno de los elementos más negativos: la ambigüedad. Para ello se recurrió a la métrica de especificidad, la cual define el cálculo del indicador de consistencia de interpretación de los revisores, dado por la probabilidad de elegir un requisito y que no sea ambiguo, en notación matemática (Sommerville, 2005):

$$Q1= NUI/NR$$

Donde NUI es el número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas y NR el número total de requisitos funcionales y no funcionales. Cuanto más cerca de uno esté el valor de Q1 menor será la ambigüedad de la especificación.

En este caso el equipo de revisores, compuesto por los desarrollares y clientes del producto concordaron en 55 casos, no habiendo concordancia en 2, los requisitos funcionales 25 y 52, como consecuencia de problemas teóricos en el caso del RF25 y de no quedar claro el RF52, en cuanto a si la ayuda debería referirse al contenido de la asignatura o a la interacción del usuario con la herramienta. El índice de consistencia calculado es 0.9666, una cifra aceptable según la bibliografía consultada (Sommerville, 2005).

2.2 PLANIFICACIÓN DE ITERACIONES

Una vez identificados los requisitos y descritas las historias de usuario correspondientes, el equipo de desarrollo realiza una estimación del esfuerzo que supone requerirá la implementación de cada una, mientras el cliente define la prioridad en función del valor para el negocio. Finalmente, se llega a un acuerdo sobre el orden de implementación y el contenido de las entregas, apostando por enfrentar las historias de más valor y riesgo lo antes posible (Letelier, y otros, 2003).

La técnica de Planeación Poker permitió evaluar de forma rápida y eficaz la dificultad relativa de las historias de usuario especificadas. Para la aplicación de esta técnica cada miembro del equipo de desarrollo propone un valor de estimación para cada historia de usuario, el cual se interpreta como más compleja a mayor valor propuesto. La propuesta es un valor de la sucesión de Fibonacci, como propone la técnica, si estos no coinciden se exponen los

argumentos de las estimaciones más alejadas y se repite el proceso hasta lograr un consenso (Cohn, 2006.).

El esfuerzo estimado está dado en función de cuán difícil resultaría para el equipo cumplir con la tarea, no de la estimación del tiempo de implementación de la misma.

La Tabla #4 presenta el plan de iteraciones resultado de dicha planificación.

Número de iteración	**Historia de usuario**	**Prioridad**	**Esfuerzo estimado**	
1	Introducir función.	Baja	1	48
	Recoger datos para resolución de métodos.	Baja	1	
	Reconocer funciones reales.	Alta	21	
	Mostrar errores léxicos en tiempo real.	Alta	5	
	Mostrar errores de sintaxis en tiempo real.	Alta	5	
	Evaluar funciones reales.	Alta	13	
	Seleccionar módulo.	Baja	1	
	Elegir método de resolución.	Alta	1	
2	Ejecutar método de Bisección para aproximar una raíz de una ecuación no lineal.	Media	3	40
	Graficar proceso de convergencia del método de Bisección paso a paso.	Media	5	
	Ejecutar método de Regula-Falsi para aproximar una raíz de una ecuación no lineal.	Media	3	
	Graficar proceso de convergencia del método de Regula-Falsi paso a paso.	Media	5	

	Ejecutar método iterativo general para aproximar una raíz de una ecuación no lineal.	Alta	3	
	Graficar proceso de convergencia del método Iterativo General paso a paso.	Media	5	
	Ejecutar método de Newton-Raphson para aproximar una raíz de una ecuación no lineal.	Media	3	
	Graficar proceso de convergencia del método de Newton-Raphson paso a paso.	Media	5	
	Ejecutar método de Secantes para aproximar una raíz de una ecuación no lineal.	Media	3	
	Graficar proceso de convergencia del método de Secantes paso a paso.	Media	5	
3	Mostrar el condicionamiento del sistema de ecuaciones lineales gráficamente.	Media	8	44
	Ejecutar método de Jacobi para aproximar el vector solución de un sistema de ecuaciones lineales.	Alta	5	
	Mostrar secuencia de aproximaciones del método de Jacobi.	Media	1	
	Ejecutar método de Gauss-Seidel para aproximar el vector solución de un sistema de ecuaciones lineales.	Media	5	
	Mostrar secuencia de aproximaciones del método de Gauss-Seidel.	Media	1	

	Ejecutar método de Gauss con factorización LU para aproximar el vector solución de un sistema de ecuaciones lineales.	Media	5	
	Mostrar descomposición de matriz principal del sistema en matrices L y U	Baja	3	
	Mostrar matriz de permutaciones P para la obtención de la factorización LU**.**	Alta	13	
	Mostrar cálculo de determinante de matriz principal de un sistema de ecuaciones lineales.	Media	3	
	Mostrar cálculo del condicionamiento de un sistema de ecuaciones lineales.	Media	3	
	Generar matriz principal del sistema a partir de expresiones de generación.	Media	8	
4	Generar vector de términos independientes a partir de expresiones de generación.	Media	5	52
	Ejecutar método de Lagrange para determinar un polinomio interpolador de una función a partir de un conjunto de puntos.	Media	3	
	Graficar conjunto de polinomios básicos de Lagrange.	Media	5	
	Graficar polinomio interpolador resultante del método de Lagrange.	Media	5	
	Ejecutar método de Newton para determinar un polinomio interpolador de una función a partir de un conjunto de puntos.	Media	3	
	Graficar polinomio interpolador resultante del método de Newton.	Media	5	

	Graficar error de interpolación del método del método de Newton.	Media	8	
	Ejecutar método de Mínimos Cuadrados para ajustar un modelo lineal a un conjunto de puntos.	Alta	13	
	Graficar modelo de ajuste obtenido por el método Mínimos Cuadrados.	Media	5	
5	Ejecutar método de Simpson para aproximar el valor de una integral definida.	Media	5	66
	Ejecutar método de Romberg para aproximar el valor de una integral definida.	Media	13	
	Ejecutar método de Trapecios para aproximar el valor de una integral definida.	Media	5	
	Graficar trapecios en el intervalo de integración.	Media	13	
	Ejecutar método de Euler para aproximar la solución de una ecuación diferencial en un intervalo dado.	Media	5	
	Ejecutar método de Runge-Kutta orden 2 para aproximar la solución de una ecuación diferencial en un intervalo dado.	Media	5	
	Ejecutar método de Runge-Kutta orden 4 para aproximar la solución de una ecuación diferencial en un intervalo dado.	Media	5	
	Graficar solución de un problema de Cauchy según el método de Euler en un intervalo dado.	Media	5	

	Graficar solución de un problema de Cauchy según el método de Runge-Kutta de orden 2 en un intervalo dado.	Media	5	
	Graficar solución de un problema de Cauchy según el método de Runge-Kutta de orden 4 en un intervalo dado.	Media	5	
6	Graficar campo de direcciones de un problema de Cauchy asociado a una ecuación diferencial.	Alta	5	57
	Graficar funciones reales.	Alta	13	
	Exportar gráficos como imagen.	Media	1	
	Configurar resolución de gráficos.	Media	1	
	Resaltar ceros de las funciones graficadas.	Media	13	
	Limpiar las funciones graficadas.	Media	3	
	Guardar en ficheros ejemplos de los métodos matemáticos.	Media	8	
	Cargar en ficheros ejemplos de los métodos matemáticos.	Media	8	
	Mostrar ayuda integrada del sistema.	Media	5	

Tabla #4 Plan de iteraciones.

2.3 ARQUITECTURA DE LA SOLUCiÓN

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, por sus siglas en inglés) define una arquitectura del software en el estándar STD 1471-2000 como: *"la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, así como los principios que orientan su diseño y evolución"* (IEEE, 2000).

La arquitectura es un nivel de diseño que se centra en la especificación de la estructura global del sistema (Garlan, y otros, 2004). En su forma más simple, la arquitectura es la organización de los componentes del programa (módulos), la manera en que estos componentes interactúan y la estructura de datos que utilizan (Kazman, y otros, 2003).

Para el desarrollo de la solución se hace uso de la propia arquitectura (n-capas) definida en la edición estándar del lenguaje Java (Java SE) (ver Anexo 3).

La capa de más alto nivel corresponde a la interfaz de usuario, para la cual se utiliza swing, entorno que permite incorporar en las aplicaciones elementos gráficos de una forma versátil e intuitiva de conjunto con el uso de Netbeans IDE (García de Jalón, y otros, 2000).

La siguiente capa corresponde a la lógica de negocio, abarca funcionalidades para el reconocimiento de funciones y la ejecución de los métodos numéricos estudiados en cada uno de los temas de la asignatura. Esta capa da soporte a la interfaz de usuario y se sirve de servicios provistos por diferentes bibliotecas de código de terceros, así como bibliotecas base de la edición estándar de Java, las cuales corresponden a la tercera y cuarta capas respectivamente.

En correspondencia con lo anterior, la tercera capa incluye bibliotecas para la graficación, cálculo simbólico, trabajo con matrices y pruebas unitarias, entre otros.

La capa más baja abarca el conjunto de bibliotecas base de la versión estándar de Java, las cuales se utilizan en la cada una de las capas antes mencionadas.

En la Figura #1 se muestra la disposición y los componentes utilizados en cada una de las capas antes descritas:

Herramientas de interfaz de usuario	Swing						
Lógica de negocio	Reconocedor de funciones	Graficación	Ecuaciones no lineales	Sistema de ecuaciones lineales	Aproximación de funciones	Integración numérica	Ecuaciones diferenciales ordinarias
Bibliotecas de terceros	JfreeChart 2.0.19	JCommon 1.0.23	Jep 2.4.1	Ext 1.1.1	Jama 1.0.2	jUnit 4.10	Nimrodlf 1.2
Bibliotecas base Java SE	I/O	Math	Lang	Util	Collections	Logging	Regular Expressions
	Máquina Virtual de Java						

Figura #1 Arquitectura del sistema

Dado que la mayor parte de la implementación del sistema se concentra en la capa de **lógica de negocio**, a continuación se representa su estructura atendiendo a sus componentes y las relaciones entre ellos:

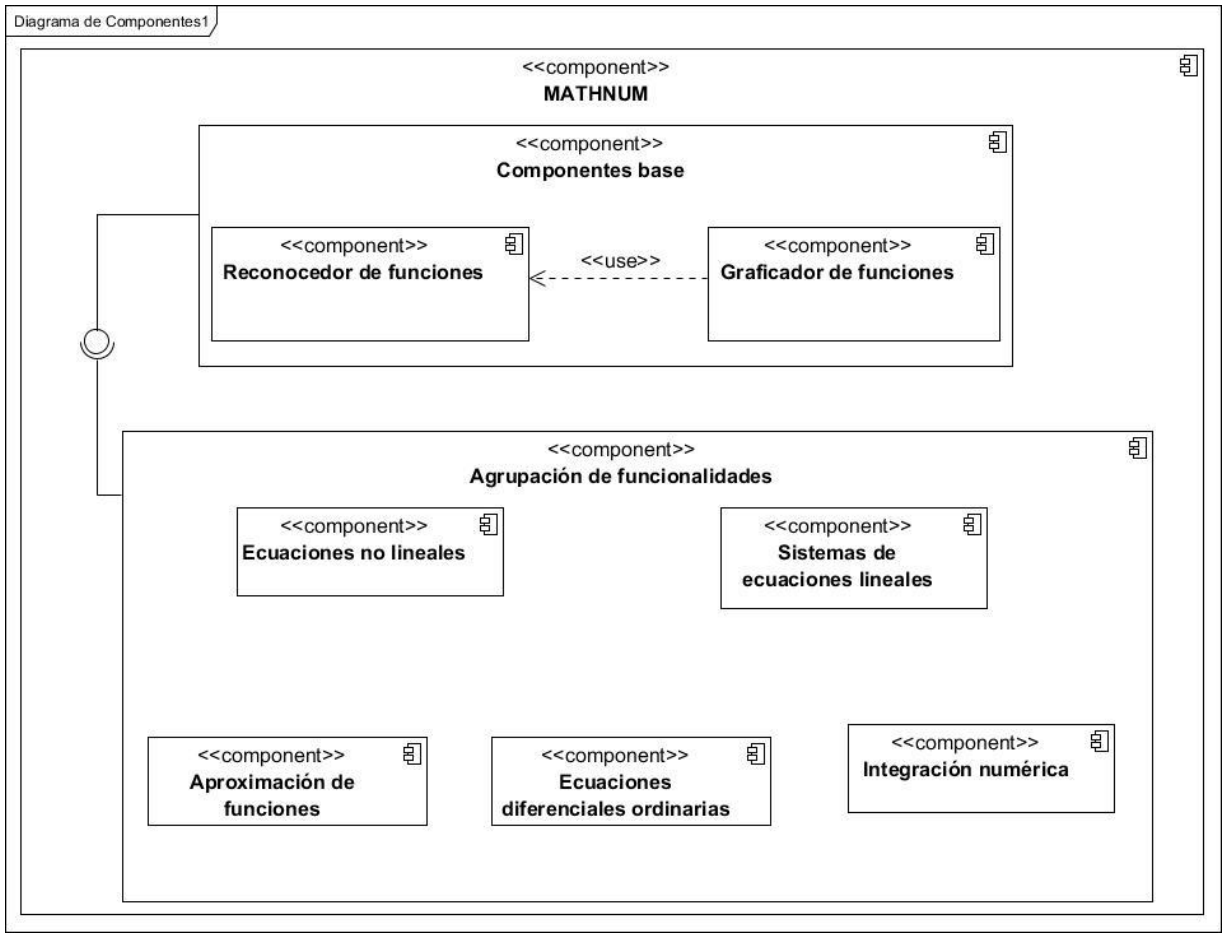


Figura #2 Diagrama de componentes.

2.4 MODELO DE DISEÑO

Uno de los principios fundamentales de la metodología XP es la simplicidad, en la etapa de diseño se propone el empleo de tarjetas CRC (Clase Responsabilidad Colaboración), en lugar de diagramas de clases para la descripción del sistema en notación UML. La característica más sobresaliente de las tarjetas CRC es su sencillez y adaptabilidad (CASAS, y otros, 2012). Dichas tarjetas representan cada una de las clases del sistema, en ellas se describen brevemente las responsabilidades de la clase y se listan las clases con las que colabora. Su utilización en el diseño potencia el uso de patrones de asignación de responsabilidades (LARMAN, 1999). A continuación se muestran 3 de las tarjetas CRC generadas, el resto pueden ser consultadas en el Anexo 4.

TARJETA CRC	
Clase	Biseccion.java
Súper Clase	MetodoUnidad1.java
Responsabilidades	**Colaboraciones**
Aproximar una raíz real de una función definida aplicando el método de Bisección.	MetodoInterfaz.java ReconocedorFuncion.jar

Tabla #5 Tarjeta CRC clase Biseccion.java

TARJETA CRC	
Clase	GaussSeidel.java
Súper Clase	MetodoUnidad2.java
Responsabilidades	**Colaboraciones**
Aproximar el vector solución de un Sistema de Ecuaciones Lineales aplicando el método de Gauss-Seidel.	MetodoInterfaz2.java ReconocedorFuncion.jar

Tabla #6 Tarjeta CRC clase GaussSeidel.java

TARJETA CRC	
Clase	Simpson.java
Súper Clase	MetodoUnidad4.java
Responsabilidades	**Colaboraciones**
Aproximar el valor de la integral de una función definida.	MetodoInterfaz3.java ReconocedorFuncion.jar

Tabla #7 Tarjeta CRC clase Simpson.java

2.5 VALIDACIÓN DEL DISEÑO

Luego de la definición de las clases y su correspondiente distribución por paquetes se procedió a la validación del diseño. En el presente caso resulta de interés medir la cohesión y el acoplamiento para garantizar un índice adecuado de reusabilidad y comprobar la correcta aplicaron de los patrones GRASP alta cohesión y bajo acoplamiento.

Para medir el grado de cohesión y acoplamiento se utilizan las métricas Tamaño Operacional de Clases (TOC) y Relaciones entre Clases (RC), analizando en cada clase indicadores como la cantidad de relaciones, índice cualitativo de responsabilidad y complejidad, entre otros de igual perfil (Góngora Rodríguez, 2011).

La herramienta desarrollada cuenta de 189 clases, pero muchas tienen un comportamiento similar pues heredan sus funcionalidades de clases abstractas que definen sus conceptos y funcionamiento de manera general, en muchos casos incluso su implementación responde a la solución de un mismo problema. De lo anterior es evidente que la validación del diseño de todas las clases según las métricas TOC y RC sería muy complejo e innecesario, pues con la selección de clases representativas de cada módulo se garantiza un resultado aproximado al que se obtendría de aplicarse la métrica a todas. El principio de selección fue elegir por cada módulo las clases con mayor importancia, complejidad y número de relaciones, resultando las siguientes:

- EvaluadorFuncion.java
- AnalizadorSintactico.java

- AnalizadorLexico.java
- GenerarMatrices.java
- CargarArchivo.java
- Graficacion.java
- FachadaBusquedaRaices.java
- FachadaSEL.java
- FachadaAproximacionDeFunciones.java
- FachadaEcuacionesDiferencialesOrdinarias.java
- FachadaIntegracionNumerica.java
- Validacion.java
- Funcion.java

Luego de analizar las clases seleccionadas es posible afirmar que el acoplamiento de los componentes y clases es bajo, los cual implica una alta cohesión, garantizando que la reutilización y mantenimiento de los mismos sea sencilla. A continuación se muestran las salidas de la métrica que indican la proporción de los indicadores acoplamiento y reutilización, las gráficas de los restantes indicadores pueden ser consultadas en el Anexo 5:

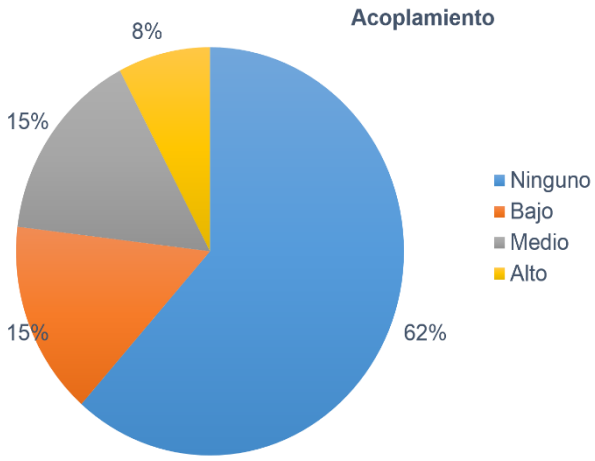


Figura #3 Resultado de métrica Acoplamiento.

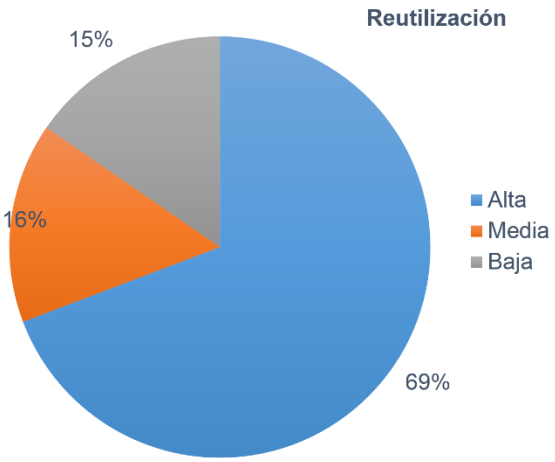


Figura #4 Resultado de métrica Reutilización.

2.6 CONCLUSIONES PARCIALES

Se realizó la especificación y validación de los requisitos funcionales y no funcionales de la aplicación. Propiciando una mejor organización y estableciendo las pautas para la implementación del sistema en las correspondientes iteraciones como parte del ciclo de desarrollo propuesto por la metodología. La arquitectura utilizada fue n-capas, definiendo cuatro capas concretamente: capa de interfaz de usuario, lógica de negocio, bibliotecas de terceros y bibliotecas base. Finalmente, la validación del diseño puso en evidencia la alta cohesión y bajo acoplamiento entre los módulos del sistema.

CAPÍTULO 3: CODIFICACIÓN Y PRUEBA

En el presente capítulo se realiza una descripción de las fases de codificación y prueba propuestas por la metodología XP. Se formaliza una gramática libre de contexto para el reconocimiento de funciones matemáticas. Además de describir el uso de patrones y estándares de codificación para garantizar la escalabilidad del código. También se presentan los resultados de la estrategia de pruebas propuesta por la metodología.

3.1 RECONOCIMIENTO DE FUNCIONES MATEMÁTICAS

Desde el mismo inicio del desarrollo del sistema se identificó la importancia de validar la entrada de una expresión matemática, reconocerla como función y evaluarla. Luego de la consulta de la bibliografía se determinó que la mejor estrategia para enfrentar este problema era la implementación de un traductor dirigido por sintaxis (TDS), un sistema que lee una entrada y genera directamente la salida a partir de la gramática que lo define (Carvajal, 2010). Para ello se hizo preciso definir una gramática (ver Anexo 6), la cual inicialmente solo contenía la precedencia de los operadores, definiéndose luego su prioridad. En este punto se eliminó la recursividad izquierda y se simplificaron los elementos comunes. Por último se verificaron los supuestos para poder afirmar que la gramática sea LL1, los cuales son (Maletti, 2008):

Dado A→ α| β

- No existe a ∈ ∑ tal que a ∈ I(α) y a ∈ I(β).
- No puede pasar que **Ø** ∈ I(α) y **Ø** ∈ I(β).
- Si β genera **Ø** entonces α no deriva en ninguna cadena que contiene **Ø.**

Donde I son los iniciales de la regla analizada y ∑ son los terminales del lenguaje definido. A continuación se demuestran las condiciones anteriormente expuestas en la siguiente regla de la gramática definida:

<TerminoTrigonometrico> -> <sen> | <cos> | <tan> | <arcsen> | <arccos> | <arctan>

<sen> -> tk_sen tk_par_abre <Funcion> tk_par_cierra

<cos> -> tk_cos tk_par_abre <Funcion> tk_par_cierra

<tan> -> tk_tan tk_par_abre <Funcion> tk_par_cierra

<arcsen> -> tk_arcsen tk_par_abre <Funcion> tk_par_cierra

<arccos> -> tk_arccos tk_par_abre <Funcion> tk_par_cierra

<arctan> -> tk_arctan tk_par_abre <Funcion> tk_par_cierra

Desde la concepción de la gramática, la misma se pensó para evitar la generación del vacío, razón por la que solo se procederá a la demostración del primer supuesto:

$$I(<sen>)\{\ tk\_sen\ \} \cap I(<cos>)\{\ tk\_\ cos\ \}, \cap I(<tan>)\{\ tk\_\ tan\ \} \cap, I(<arcsen>)\{\ tk\_arcsen\} \cap, I(<arccos>)\{\ tk\_\ arccos\ \} \cap, I(<arctan>)\{\ tk\_\ arctan\ \} = \emptyset$$

Lo anterior demuestra que los iniciales de cada una de las reglas no coinciden entre sí, lo cual determina un único camino en cada caso.

3.2 PATRONES DE IMPLEMENTACIÓN

Durante la implementación del TDS descrito anteriormente se detectó la necesidad de interpretar frases de un lenguaje, en este caso definido por las expresiones matemáticas, y garantizar el proceso de reconocimiento y evaluación del mismo, para ello se empleó el patrón **intérprete** definido en el libro “Desingn Patterns” de la autoproclamada “banda de los cuatro” (GOF) (Gamma, y otros, 1994). En el Anexo 7 se evidencia su aplicación mediante el mapa conceptual del mismo.

Una vez completado este importante componente se procedió a la implementación de los métodos numéricos correspondientes a cada tema de la asignatura, detectando que por lo general los métodos que resuelven un mismo problema tienen las mismas entradas y aunque no exactamente las mismas salidas, estas se representan por la misma estructura de datos, además comparten muchos resultados intermedios. Esta situación tiene como inconveniente la creación de varias variables con los mismos datos y la repetición innecesaria de diferentes cálculos, el manejo de la herencia y polimorfismo resultan insuficientes para cubrir esta situación, se necesita de una interfaz unificada para el conjunto de clases de cada subsistema,

identificándose el patrón **fachada** para la implementación de una interfaz de comunicación de alto nivel que hace el subsistema más fácil de utilizar (Gamma, y otros, 1994).

Una vez definido y unificado el acceso a las implementaciones de métodos de un mismo tema, ante la persistencia del problema de la creación de múltiples objetos de estas clases fachada, se debe garantizar que cada clase importante para el negocio solo tenga una instancia, proporcionando un punto de acceso global a la misma. El patrón **instancia única** (Gamma, y otros, 1994) se acopla perfectamente a la resolución de este problema, por lo que se procedió a su implementación de conjunto con el patrón fachada, garantizando la existencia de una única instancia de las clases que se emplean como punto de acceso a los métodos de un mismo tema. A continuación se muestra un ejemplo de la implementación de ambos patrones en la clase que provee el acceso a los métodos de búsqueda de raíces de ecuaciones:

```
public class FachadaBusquedaRaices {
  private FachadaBusquedaRaices fachada;
  private Biseccion biseccionMetodo;
  private Secante secanteMetodo;
  private NewtonRapson newtonRaphson;
  private RegulaFalsi regulaFalsiMetodo;

  private static FachadaBusquedaRaices() {
      FachadaBusquedaRaices final INSTANCE= new FachadaBusquedaRaices();

  }

  public static FachadaBusquedaRaices getInstance() {
      return FachadaBusquedaRaices.INSTANCE;
  }
```

Figura #5 Implementación de patrones fachada e instancia única.

3.3 ESTÁNDARES DE CODIFICACIÓN

Las buenas prácticas de programación indican el seguimiento de un conjunto de pautas desde la misma creación de los ficheros, lo cual facilita la trazabilidad y mantenibilidad del software.

Las clases serán colocadas en archivos independientes que solo contendrán el código de la misma. Se utilizará el estilo de codificación "*Upper Camel Case*", el cual establece que los

nombres iniciarán con letra mayúscula y de poseer más de una palabra, la primera letra de cada una deberá ser también mayúscula. No se permiten letras mayúsculas sucesivas a menos que se trate de siglas conocidas en el dominio del sistema (Binkley, 2009). A continuación se presenta un ejemplo en la declaración de la clase GaussSeidel.java.

```
public class GaussSeidel extends MetodoUnidad2 implements SistemaEcuacionInterfaz {

    public GaussSeidel(Object[][] matriz) throws AnalisisException {
        super(matriz);
    }
```

Figura #6 Utilización de Upper Camel Case en los nombres de las clases

Los nombres de las variables deben ser descriptivos y concisos. No se usarán grandes frases ni abreviaturas. Se utilizará el estilo de codificación *"Lower Camel Case"* (Binkley, 2009), según el cual los nombres iniciarán con letra minúscula y cada nueva palabra debe iniciar con mayúscula. A continuación se muestra un ejemplo en la implementación de la clase Raices.java:

```
public class Raices extends javax.swing.JFrame {

    private boolean funcionCorrecta;
    private boolean errorCorrecto;
    private FachadaBusquedaRaices funcion;
    private int iteracionesBiseccion;
    private int iteracionesRegulaFalsi;
    private int iteracionesIG;
    private int iteracionesNewtonR;
```

Figura #7 Utilización de Lower Camel Case en los nombres de las variables

Los nombres de las funciones deben dar una idea clara del objetivo con el que fueron concebidas. Al igual que para las variables, se utilizará el estilo *"Lower Camel Case"* (Binkley, 2009). La declaración del método generarTokens() evidencia el empleo de este estilo:

```
private void generarTokens() throws Exception {

    colaTokens = new LinkedList<Token>();
    StringBuffer entrada = new StringBuffer(cadenaFuncion);
```

Figura #8 Utilización de Lower Camel Case en los nombres de las funcionalidades

3.4 PRUEBAS DE UNIDAD

Se realizaron pruebas de unidad a las funcionalidades más significativas del sistema, con el propósito de verificar el adecuado desempeño de las mismas; para ello se utilizó el marco de trabajo jUnit en su versión 4.10, el cual viene integrado con el IDE Netbeans.

Las pruebas de unidad se implementaron antes que los métodos numéricos objetos de prueba, y se ejecutaron al terminar su codificación y luego de cada cambio realizado sobre los mismos u otras clases relacionadas. Para las pruebas se definen los casos críticos en los valores de entrada y el comportamiento esperado de los métodos analizados (Re B., 2012), sin embargo su principal fortaleza radica la automatización, y con esta la posibilidad de repetirlas cuantas veces se estime necesario con un mínimo de esfuerzo.

La siguiente imagen representa los resultados de la prueba realizada a las funcionalidades de la clase Diferenciacion.java; evidenciando su correcta ejecución en cuanto a tiempo de respuesta y validez de los resultados:

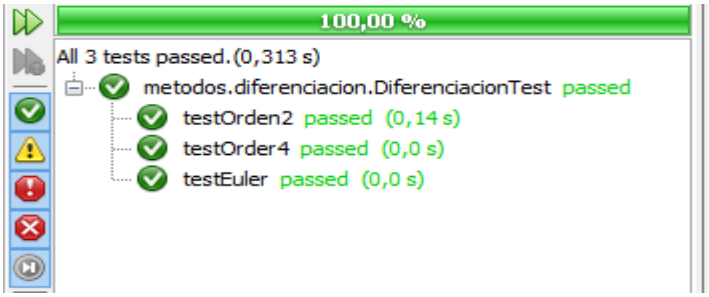


Figura #9 Prueba unitaria a los métodos de la clase Diferenciación.java.

3.5 PRUEBAS DE ACEPTACIÓN

Para la realización de las pruebas de aceptación del grupo alfa se describieron casos de pruebas para cada historia de usuario, a continuación (Tabla #8) se muestra uno de ellos, con sus respectivos juegos de datos (Tabla #9):

CASO DE PRUEBA DE ACEPTACIÓN			
Historia de Usuario: HU_31		Ejecutar método de Lagrange para determinar un polinomio interpolador de una función a partir de un conjunto de puntos	
Descripción:		Permite determinar el polinomio interpolador correspondiente a una serie de puntos entrados por el usuario mediante el método de Lagrange, visualizando su representación gráfica.	
Condiciones de ejecución:		El usuario debe entrar al menos un punto.	
Escenarios de prueba:		**Flujo del escenario:**	**Resultados esperados:**
EP:1	Determinar polinomio interpolador por el método de Lagrange con datos válidos.	Se introducen datos válidos y se acciona el botón “Resolver”.	Se muestra en el panel el polinomio interpolador generado por el método.
EP:2	Determinar polinomio interpolador por el método de Lagrange con datos incorrectos.	Se introducen datos incorrectos y se acciona el botón “Resolver”.	Se muestra un mensaje de error en una ventana emergente indicando el error: “Datos incorrectos”.
EP:3	Determinar polinomio interpolador por el método de Lagrange con datos incompletos.	Se acciona el botón “Resolver” sin haber completado los campos necesarios para la ejecución del método.	Se muestra un mensaje de error en una ventana emergente indicando el error: “Campos vacíos”.

Tabla #8 Caso de prueba de aceptación: Ejecutar método de Lagrange para determinar un polinomio interpolador.

Escenario de prueba	x	Y	Función original	Valor a interpolar
EP:1	{1,2,4}	{1,4,16}	x^2	3
EP:2	{1,e,o}	{1,t,0}	x^2	3.1
EP:3	{0, ,5}	{ ,2, }	x^2	

Tabla #9 Juego de datos de prueba: Ejecutar método de Lagrange para determinar un polinomio interpolador.

Se realizó un total de 7 iteraciones de prueba, una para cada iteración de desarrollo planificada y una última para validar la corrección de no conformidades pendientes. En **(**Figura #10) se muestra la relación entre las pruebas satisfactorias y no satisfactorias, evidenciando una evolución positiva en cuanto a la corrección de deficiencias y eliminación de errores en la aplicación. En cada iteración no solo se comprobaron los requisitos planificados, sino todos los implementados hasta el momento (Malfará, y otros, 2006), manteniéndose no conformidades en el RF3 Reconocer funciones reales, que provocaron cambios importantes hasta el final de las iteraciones.

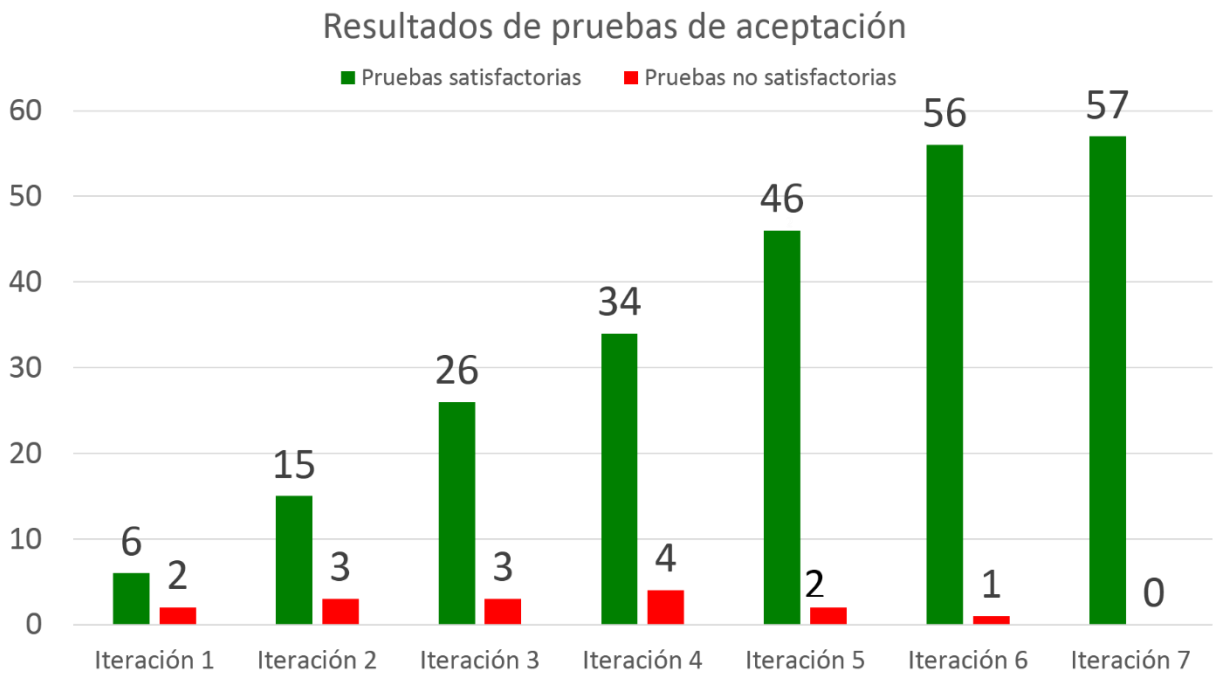


Figura #10 Resultado de las pruebas de aceptación.

Una vez concluidas las pruebas alfa y corregidas las no conformidades detectadas, se procedió a la realización de pruebas beta por parte del colectivo de profesores del departamento de Ciencias Básicas de la Facultad 3, detectándose algunos elementos a mejorar para una mayor agilidad en el uso de la aplicación, corrigiéndose de inmediato y quedando el sistema en condiciones para su explotación, lo cual consta en carta de aceptación emitida el 20 de mayo de 2015 (ver Anexo 8).

3.6 CONCLUSIONES PARCIALES

Se presentó la definición y posterior validación de la gramática para el reconocimiento de funciones demostrando que está en la forma LL (1). La identificación de patrones de implementación produjo que la solución a problemas comunes fuera de manera rápida y eficiente. Como resultado de la realización de las pruebas de unidad, de conjunto con la las pruebas de aceptación se obtuvo la validación la aplicación.

CONCLUSIONES

Luego de la consulta y estudio de la bibliografía, así como de las alternativas tomadas en la dirección de acotar el impacto del problema identificado resulta evidente la necesidad de enfrentarlo con una nueva solución.

La metodología de desarrollo elegida (XP) posibilitó la obtención del modelo de diseño de la solución propuesta.

La solución desarrollada facilita la representación numérica y geométrica de las soluciones aproximadas según los métodos estudiados en la Matemática IV en la UCI, dotándola de un potente instrumento para la comprensión de sus conceptos.

Durante el proceso de validación se ratificó la calidad y usabilidad de la solución, corroborando su aporte al PEA de la asignatura Matemática IV en la UCI.

RECOMENDACIONES

- Permitir utilizar código latex para enriquecer las descripciones de los ejemplos a guardar y así poder darle más flexibilidad al uso de la herramienta por parte de los profesores.

- En el caso del método de mínimos cuadrados permitir la evaluación y comparación entre modelos a partir del valor de la desviación cuadrática en cada caso.

- Implementar un módulo para el estudio de la aritmética de punto flotante.

REFERENCIAS

Báez G. y S. B. Brunner Metodología DoRCU para la Ingeniería de Requerimientos [Sección de libro]. - Habana : Instituto Superior Politécnico Jose Antonio Echeverría, 2001.

Binkley D. IEEE [En línea]. - IEEE, 2009. - 6 de 5 de 2015. - http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5090039&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Ficp.jsp%3Farnumber%3D5090039.

Blanco Dr.Manuel Álvarez, Guerra Alfredo y Lau Rogelio Matemática Numérica [Libro]. - La Habana : Felix Varela, 2007. - Vol. 1 y 2.

Camejo Diago Iliana, Valdés Quintana Grisel y Valdivia Linares Yailin IMPACTO DEL USO DEL ASISTENTE MATEMÁTICO WX MÁXIMA EN EL APRENDIZAJE [Publicación periódica]. - [s.l.] : UCIENCIA, 2012. - Vols. 978-959-286-019-3.

Canos J. y Letelier P. Metodologías Ágiles en el Desarrollo de Software [Sección de libro]. - Valencia : Universidad Politécnica de Valencia, 2011.

Carvajal Javier Traducción Dirigida por Sintaxis [Libro]. - Costa Rica : Universidad de Costa Rica, 2010. - Vol. 1.

CASAS S. y REINAGA H. Aspectos Tempranos: un enfoque basado en Tarjetas CRC. [Publicación periódica]. - Bogota : Sociedad Colombiana de Computación, 2012. - 1657-7663 .

Cohn Mike Agile Estimating and Planning [Libro]. - Massachusetts : Pearson Education, 2006.. - Vol. 1. - 0-13147971-5.

Colectivo de autores Programa analitico de la asignatura Matematica IV [Informe]. - Habana : [s.n.], 2014.

Corbea Rodríguez Maite La metodología XP aplicable al desarrollo del software en Cuba [Informe]. - Habana : Universidad de las Ciencias Informáticas, 2007.

de la Torre Navarro Dra. Lilia María y Domínguez Gómez Dr. José Las TIC en el proceso de enseñanza aprendizaje a través de los objetos de aprendizaje [Publicación periódica]. - La Habana : RCIM, 2012. - 1 : Vol. vol.4. - 1684-1859.

Definition.org [En línea]. - 2011. - 3 de 3 de 2015. - http://www.definicion.org/lenguaje-de-programacion.

Derive [En línea]. - 2013. - 1 de 12 de 2014. - http://www.derive.com.

ecured ecured [En línea]. - 2015. - 3 de 11 de 2014. - http://www.ecured.cu/index.php/Proceso_de_Ense%C3%B1anza_-_Aprendizaje.

EIDMAT grupo Manual de usuario [Informe]. - Habana : [s.n.], 2012.

Encyclopedia. PC Magazine [En línea]. - 2013. - 3 de 3 de 2015. - http://www.pcmag.com/encyclopedia/term/44707/ide.

Española Real Academia Real Academia Española [En línea] // Diccionario de la lengua española. - 2015. - 1 de 3 de 2014. - http://lema.rae.es/drae/?val=perfeccionamineto.

Fernández Nodarse Francisco Alberto y Lima Montenegro Sylvia Experiencias en la estructuración de clases de matemáticas empleando asistentes [Informe]. - Chile : Conferenica Iberoamericana de pedagogia, 2000.

Gamma Erich [y otros] Design Patterns [Libro]. - Sydney : Pearson Education, 1994.

García de Jalón Javier, Ignacio Rodríguez José y Mingo Iñigo Aprenda Java como si estuviera en primero [Sección de libro] // Aprenda Java como si estuviera en primero. - Madrid : Escuela Superior de Ingenieros Industriales, 2000.

Garlan David y Shaw Mary An Introduction to Software Architecture [Informe]. - Pittsburgh University : School of Computer Science Carnegie Mellon, 2004.

Goering Richard Matlab edges closer to electronic design automation world [Libro]. - California : EE Times, 2010.

Góngora Rodríguez Asnier Enrique Catálogo automatizado de métricas de calidad [Informe]. - La Habana : [s.n.], 2011.

Hernández Sampieri Roberto, Fernández-Collado Carlos y Baptista Lucio Pilar Metodología de la Investigación. [Sección de libro] // Metodología de la Investigación / aut. libro Hernández Sampieri Roberto, Fernández-Collado Carlos y Baptista Lucio Pilar. - Mexico : Mc Graw Hill, 2006. - 970-10-5753-8.

IEEE IEEE-STD-830-1998 : ESPECIFICACIONES DE LOS REQUISITOS DEL SOFWARE [Informe]. - 1998. - 610.12-1990.

IEEE SA-1471-2000 IEEE Recommended Practice for Architectural Description for Software-Intensive [En línea]. - 2000. - 2 de 2 de 2015. - http://standards.ieee.org/findstds/standard/1471-2000.html.

Informáticas Universidad de las Ciencias Plan D, Universidad de las Ciencias Informáticas [Informe]. - 2013.

jama jama [En línea]. - 2014. - 1 de 11 de 2014. - http://math.nist.gov/javanumerics/jama/.

JEP Jep Java - Math Expression Parser - Singular Systems [En línea]. - 1 de 11 de 2014. - http://www.singularsys.com/jep/.

JFreeChat Welcome To JFreeChart! [En línea]. - 1 de 11 de 2014. - http://www.jfree.org/jfreechart/.

Kazman R y Bass L Software Architecture in Practice [Libro]. - Pisttburgh : Carnegie University, 2003.

Kent Beck Addison-Wesley Extreme Programming Explained: Embrace Change [Libro]. - California : [s.n.], 2000. - 201-61641-4.

Language Unified Modeling Unified Modeling Language [En línea]. - 2013. - 3 de 3 de 2015. - http://www.uml.org.

LARMAN C. UML y Patrones. Introducción al análisis y diseño orientado a objetos. [Sección de libro]. - Mexico : Edtion ed. México: Prentice Hall, 1999. - Vols. ISBN 970-17-0261-1..

Letelier P. y M. C. Penade Metodologías Ágiles en el Desarrollo de Software: eXtreme Programming(XP) [Sección de libro]. - Valencia : Universidad Politécnica de Valencia, 2003.

López Quesada P. Fundamentos de Ingeniería del Software [En línea]. - 2005. - 3 de 2 de 2015. - http://dis.um.es/~lopezquesada/documentos/FIS_0405/Tema7.ppt.

Maletti A. A Simple Syntax-Directed Translator [Libro]. - Ucrania : University Rovira, 2008.

Malfará Dayvis [y otros] Testing in eXtreme Programming [Informe]. - [s.l.] : Gestión de Software, 2006.

Manning JUnit in Action, second edition [Libro]. - Stamford : [s.n.], 2005. - 9781935182023.

Maxima Maxima un sistema de álgebra computacional [En línea]. - 2012. - 11 de 12 de 2014. - http://maxima.sourceforge.net/es/.

mindview Thinking in patterns with java [En línea]. - 2012. - 26 de 1 de 2015. - http://www.mindview.net/Books/TIPatterns/.

Netbeans Bienvenido a Netbeans [En línea]. - 2013. - 1 de 2 de 2015. - https://netbeans.org/index_es.html.

Octave GNU GNU Octave [En línea]. - 2013. - 11 de 3 de 2015. - http://www.gnu.org/software/octave.

Oracle Java, cree el futuro [En línea]. - 2013. - 4 de 3 de 2015. - http://www.oracle.com/es/technologies/java/features/index.html.

Patrones-GRASP scribd [En línea]. - 2013. - http://es.scribd.com/doc/139017007/Patrones-GRASP#scribd.

Pressman R Ingeniería de Software. Un enfoque prático [Libro]. - 2005. - 3, ilustrada. - 8448100263, 9788448100261.

Quarteroni A Cálculo Cientifico con Octave y MatLab [Libro]. - New York : [s.n.], 2006. - 10 88-470-0503-5.

Re B. Alexander Pruebas unitarias [En línea]. - 2012. - http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

Sariol Fernández Michel, Hernández García Lian y Ruíz Zamora Efrain MATHMUN herramienta para la enseñanza de los métodos numéricos [Publicación periódica]. - [s.l.] : Conferencia científica de la Universidad de las Ciencias Informáticas, 2014. - 978-959-286-024-7.

Scilab Scilab [En línea]. - 2012. - 2 de 3 de 2015. - http://www.scilab.org.

Sommerville Ian Ingeniería del software [Libro] / ed. ilustrada. - 2005. - Vol. 7mo. - 8478290745.

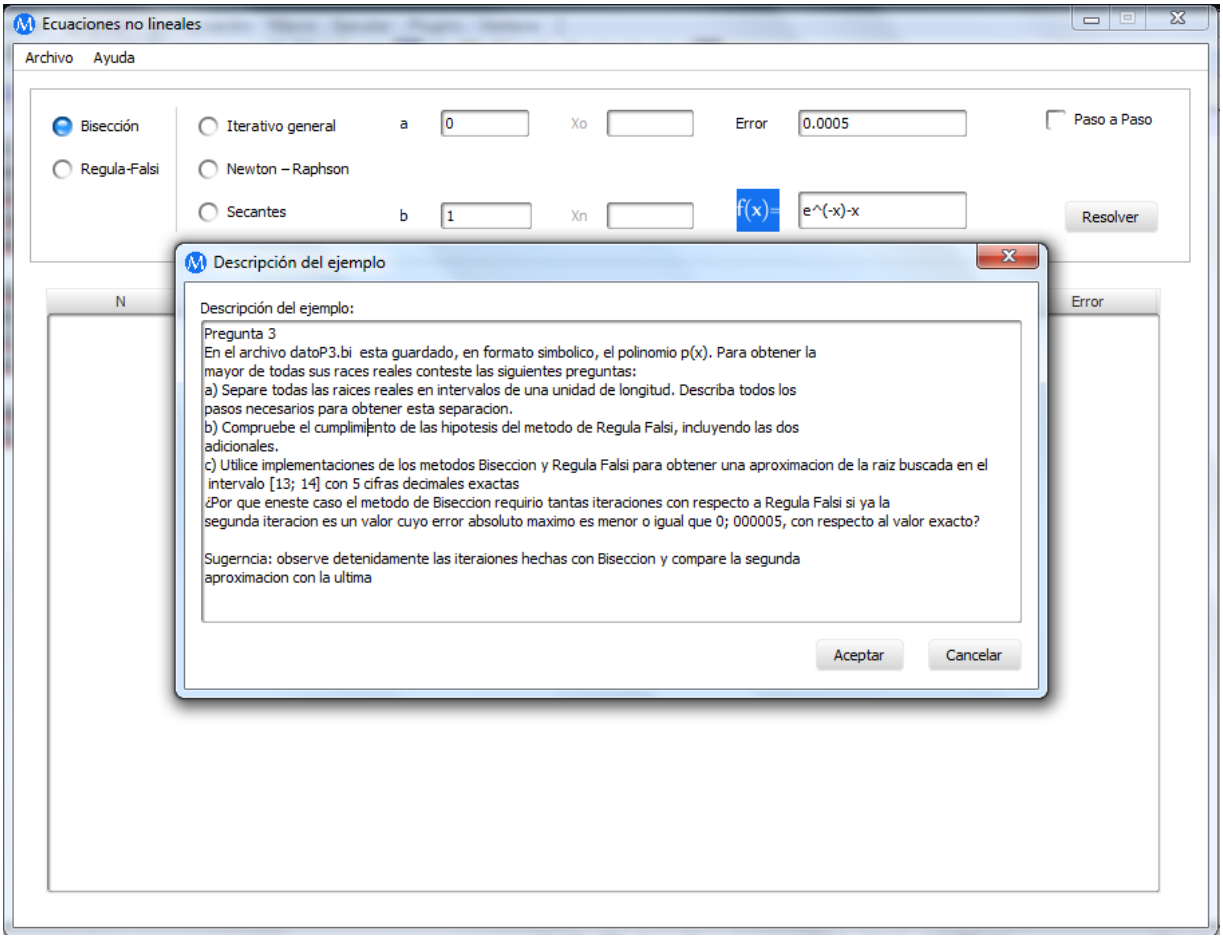
Trefethen Lloyd The definition of numerical analysis. [Informe]. - Unversidad de Cornell : Unversidad de Cornell, 1992. - 4853-7501.

Visual Paradigm Introducing Sleek UI [En línea]. - 2013. - 1 de 11 de 2014. - http://www.visual-paradigm.com/.

Wolfram Mathematica de Wolfram [En línea]. - 2010. - 20 de 1 de 2015. - http://www.wolfram.com/mathematica/ .

ANEXOS

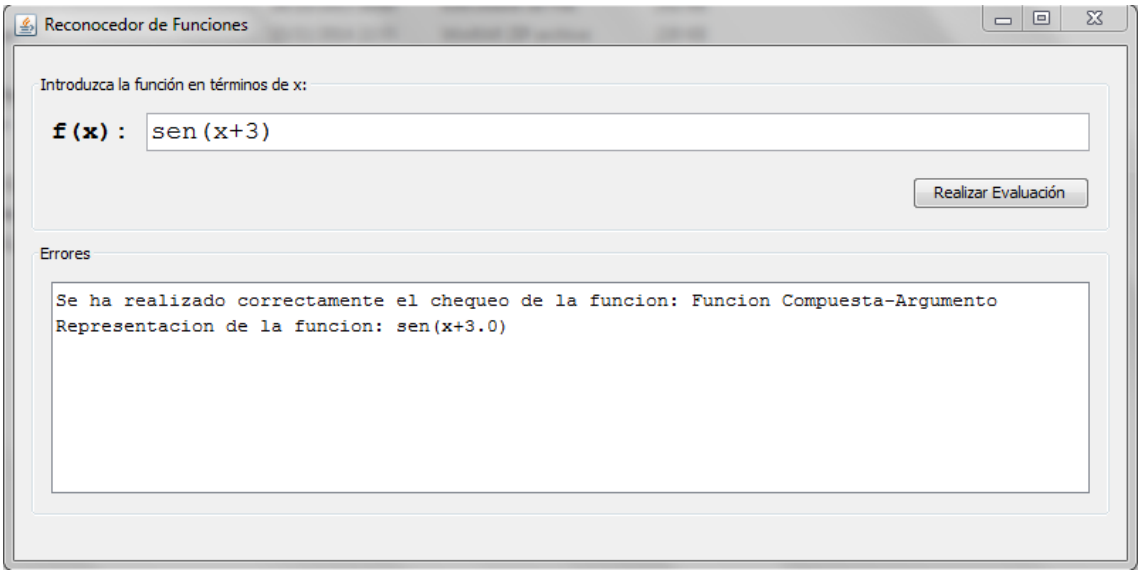
Anexo1: Descripción de ejemplo



Anexo2: Historias de usuario

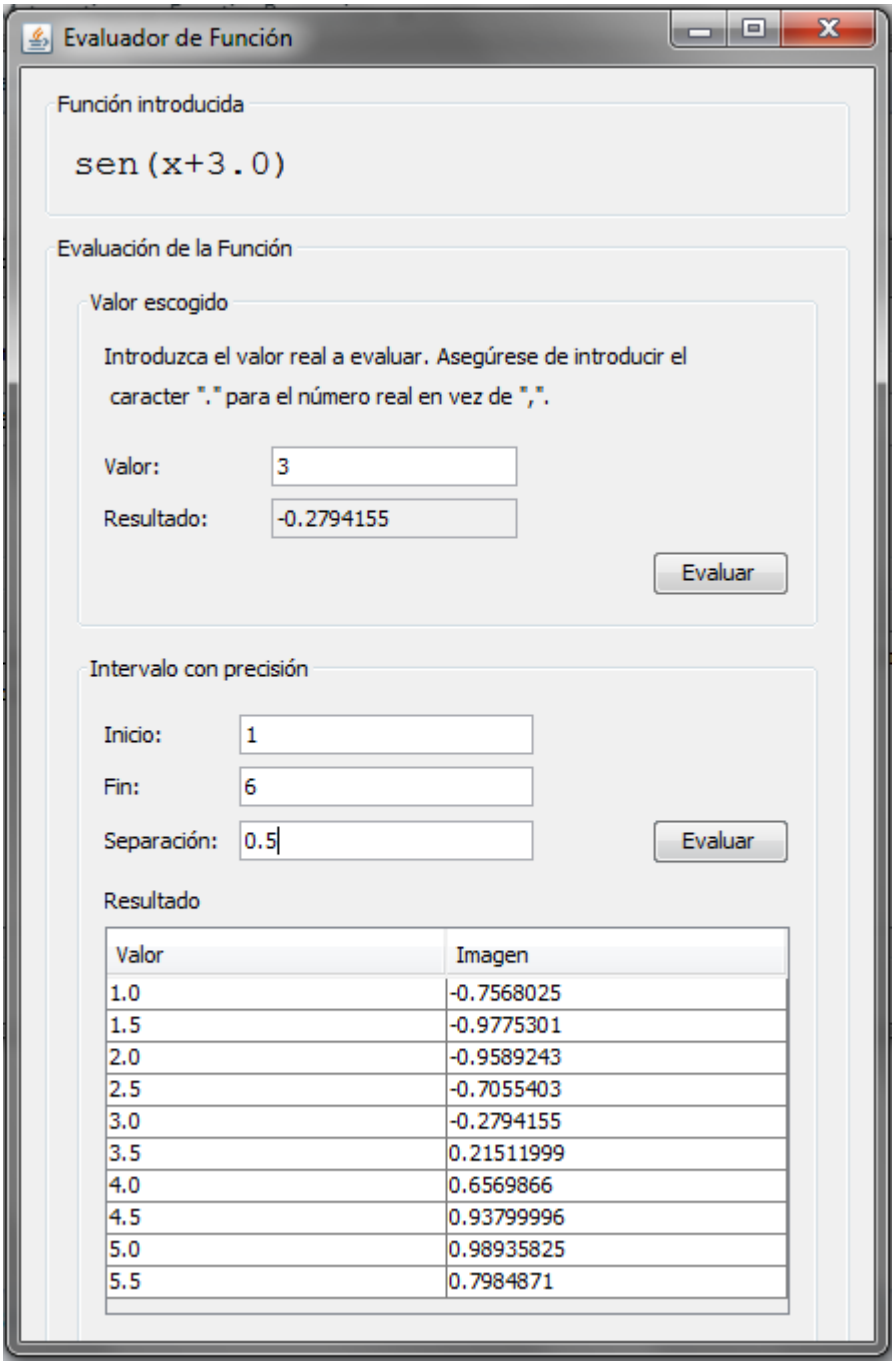
Historia de Usuario	
Numero_1	**Nombre de historia de usuario:** Reconocer funciones reales.
Actor: Usuario	**Iteración asignada:**1
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: Una vez que el usuario introduce una función en el campo “función”, el sistema debe identificar los errores de sintaxis que cometidos por el usuario.	
Observaciones: El campo “función” no puede estar vacío.	

Prototipo de interfaz:



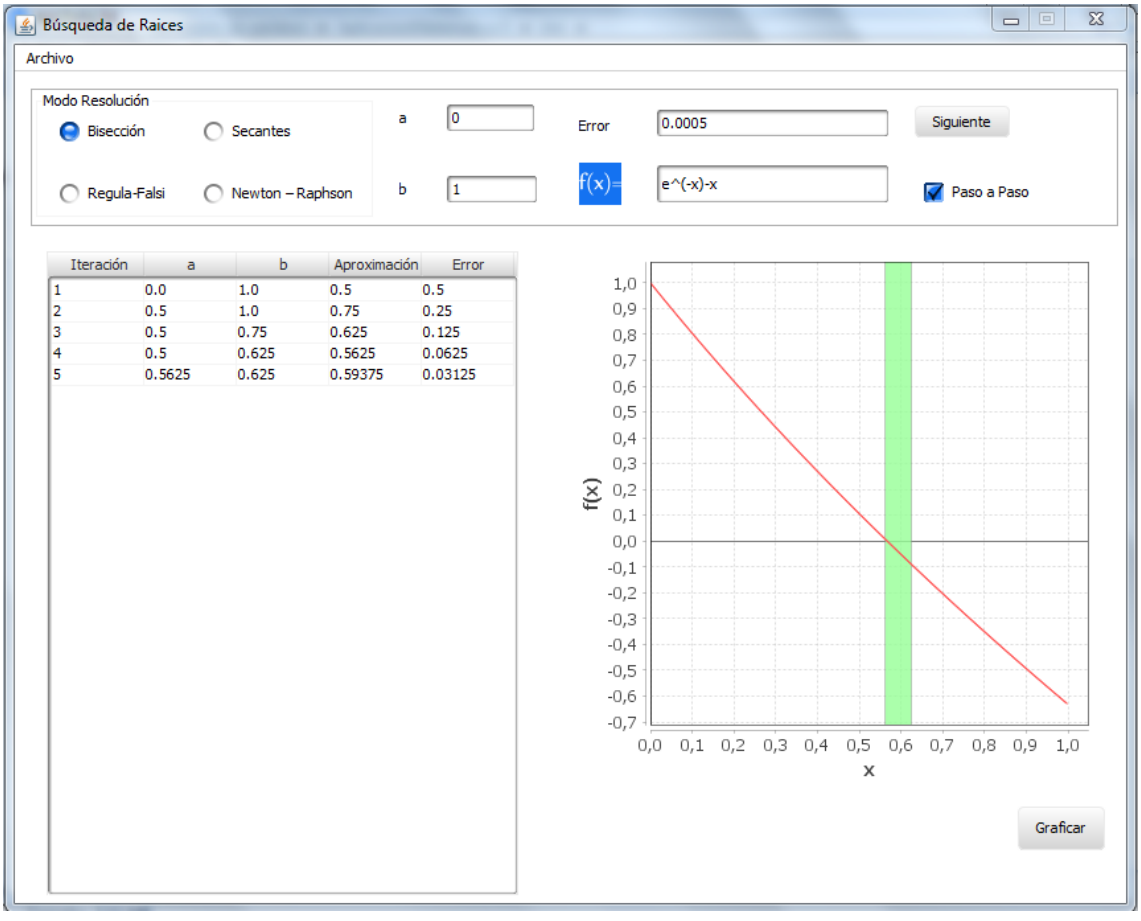
Historia de Usuario	
Numero_2	**Nombre de historia de usuario:** Evaluar funciones reales.
Actor: Usuario	**Iteración asignada:**1
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: Una vez que el usuario introduce un valor o intervalo a evaluar acciona el botón evaluar y se muestran sus respectivas evaluaciones en una tabla.	
Observaciones: El campo “función” no puede estar vacío.	

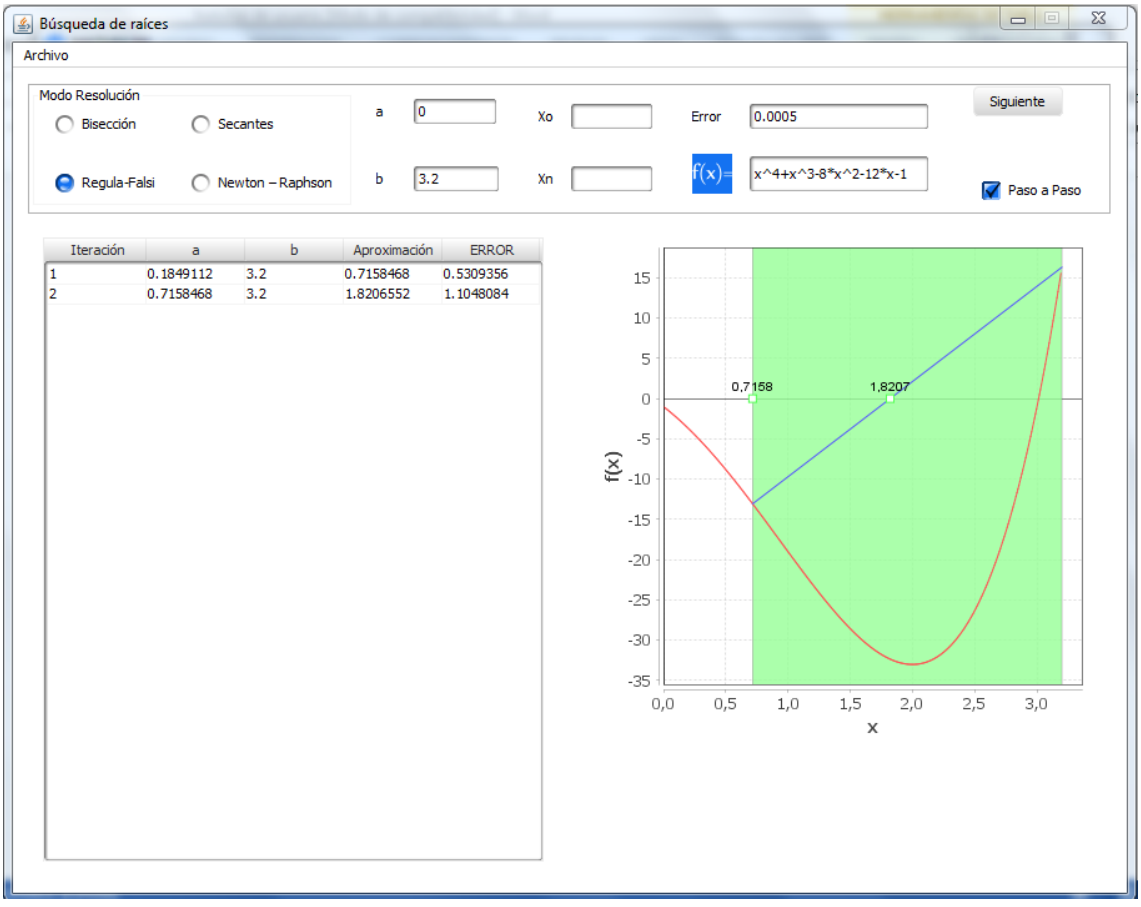
Prototipo de interfaz:



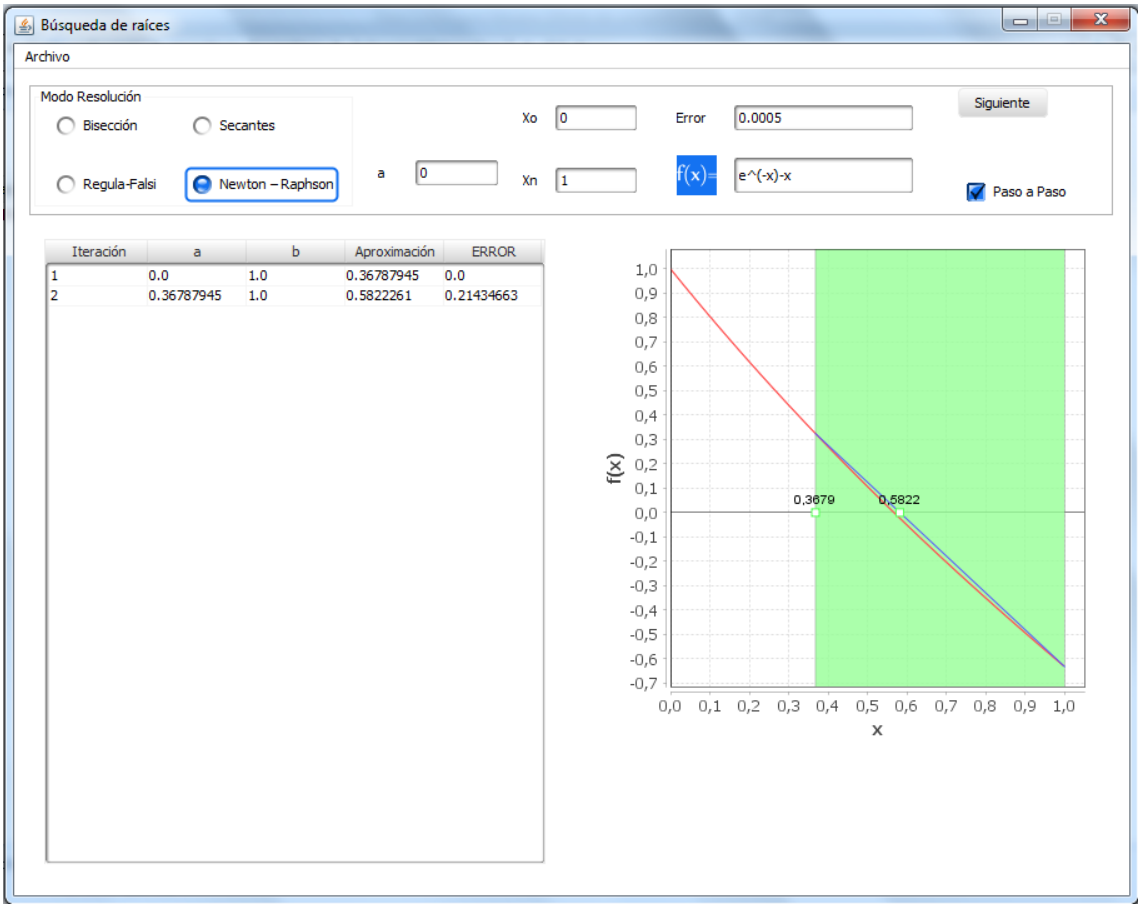
Historia de Usuario

Numero_3	**Nombre de historia de usuario:** Aplicar método de Bisección para aproximar una raíz.
Actor: Usuario	**Iteración asignada:**2
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: Una vez que el usuario selecciona el método de “Bisección” introduce los datos requeridos: intervalo, función y error, luego accionar el botón “Resolver”	
Observaciones: Los campos “función”, “a”, “b”, “error” no pueden estar vacíos.	
Prototipo de interfaz:	

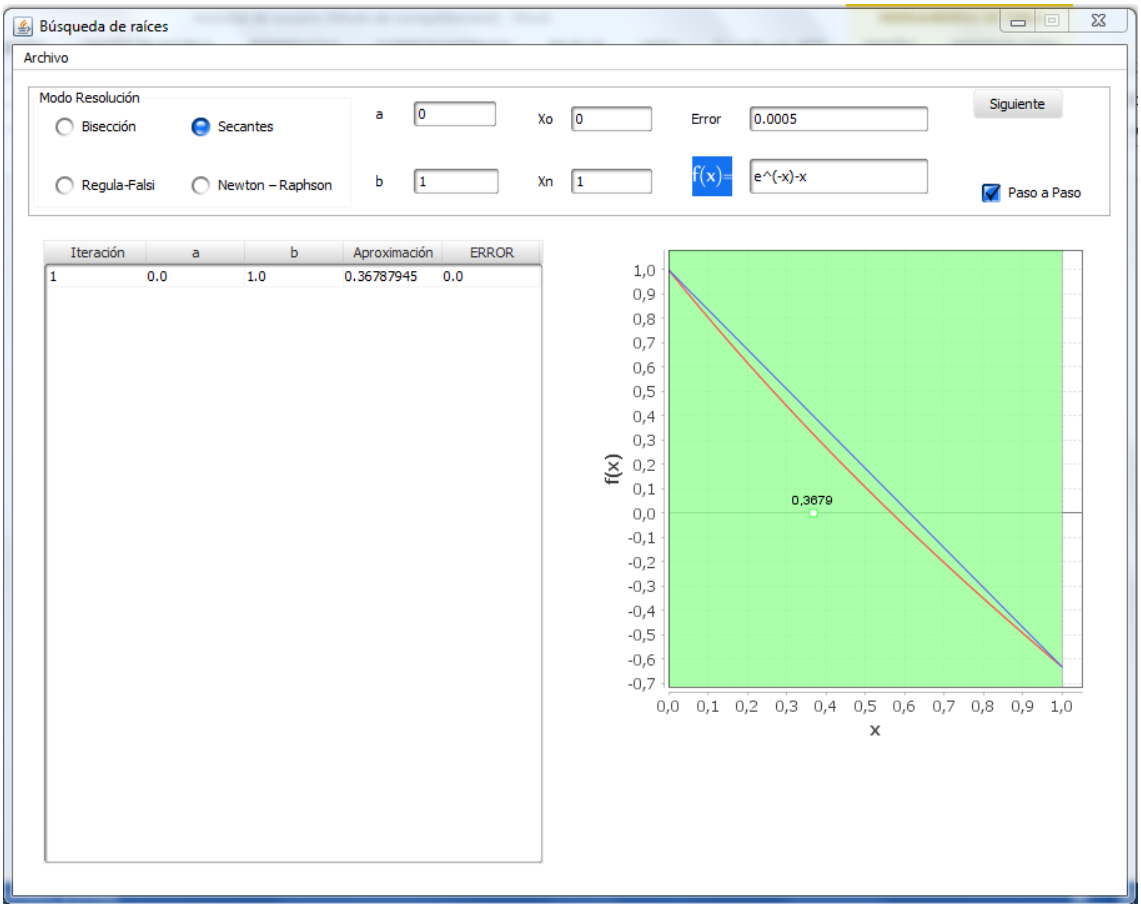


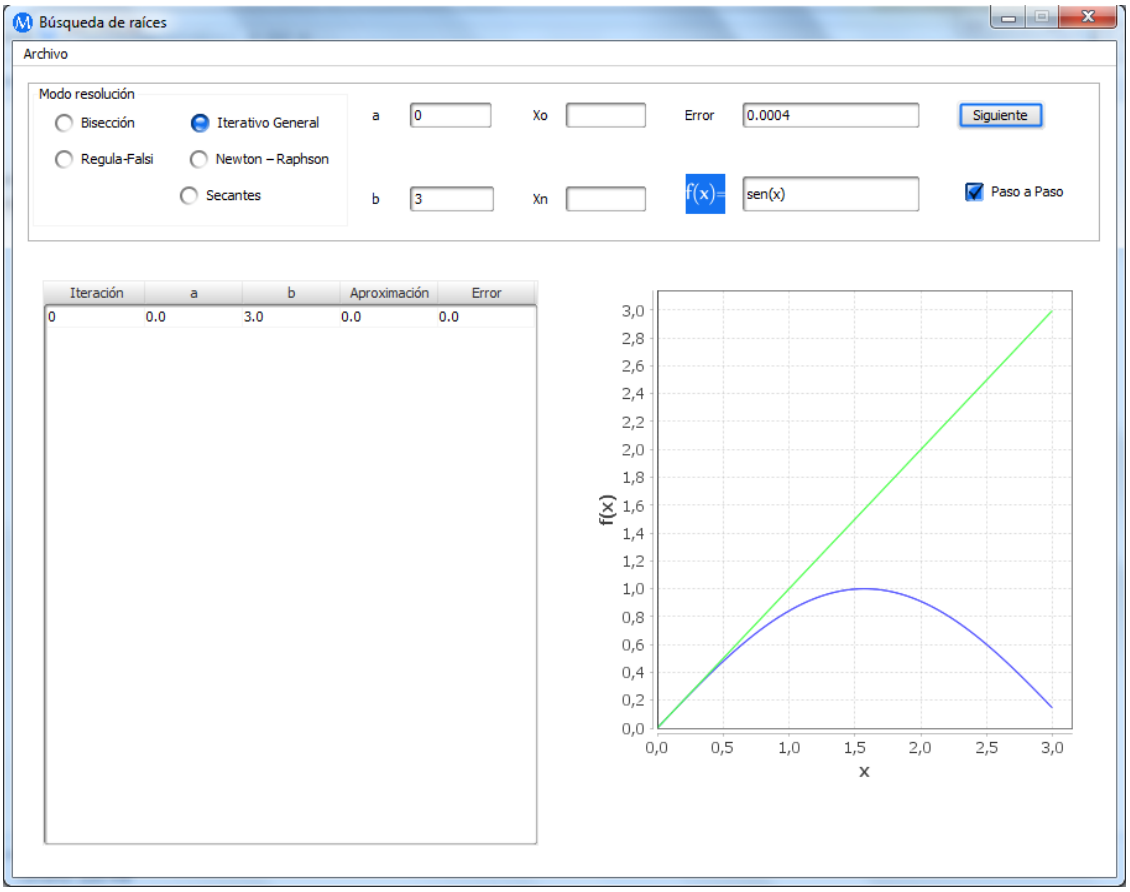
Historia de Usuario	
Numero_4	**Nombre de historia de usuario:** Aplicar método de Regula-Falsi para aproximar una raíz.
Actor: Usuario	**Iteración asignada:**2
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: Una vez que el usuario selecciona el método de “Regula-Falsi” introduce los datos requeridos: intervalo, función y error, luego accionar el botón “Resolver”	
Observaciones: Los campos “función”, “a”, “b”, “error” no pueden estar vacíos.	
Prototipo de interfaz: 	

Historia de Usuario	
Numero_5	**Nombre de historia de usuario:** Aplicar método de Newton-Raphson para aproximar una raíz.
Actor: Usuario	**Iteración asignada:**2
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: Una vez que el usuario selecciona el método de “Newton-Raphson” introduce los datos requeridos: intervalo, función y error, luego accionar el botón “Resolver”	
Observaciones: Los campos “función”, “a”, “x0”, “x1” y “error” no pueden estar vacíos.	
Prototipo de interfaz:	

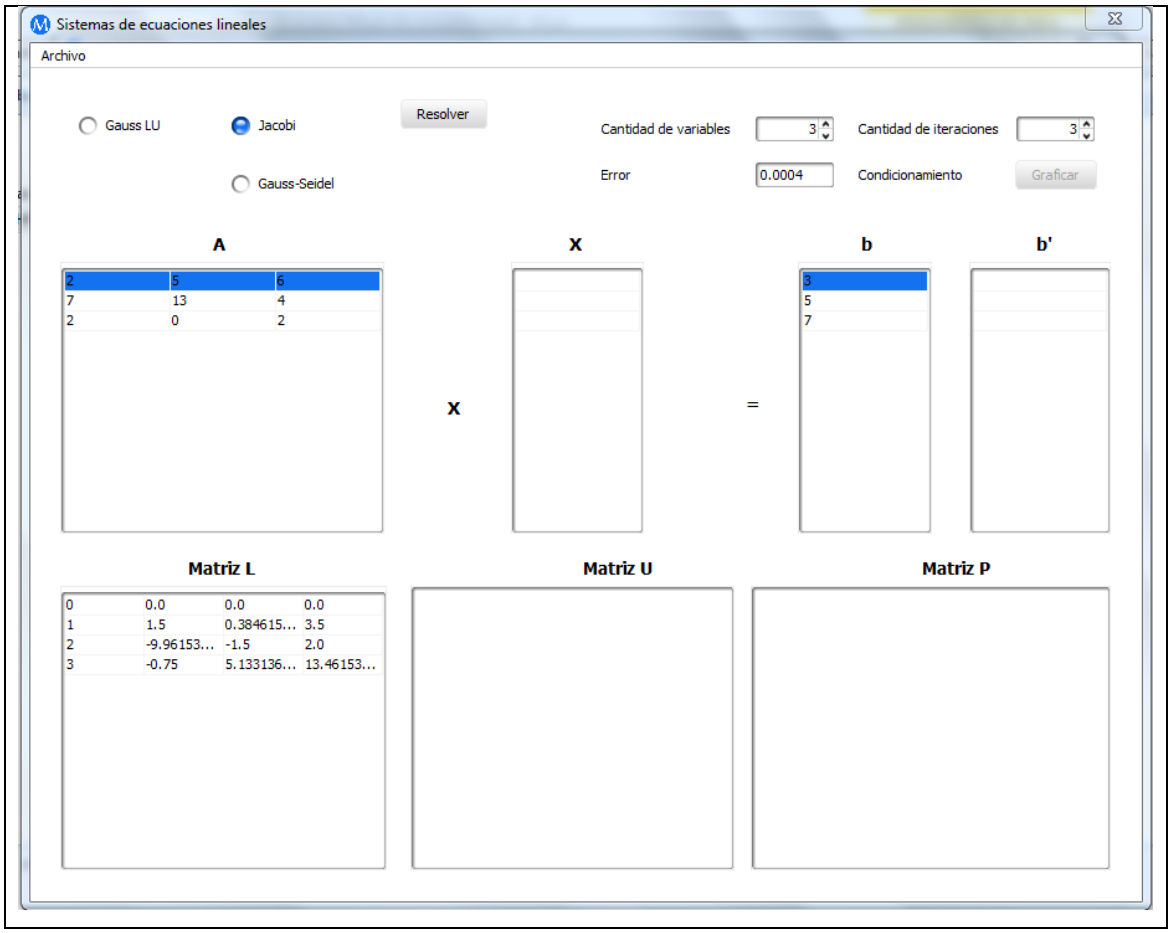


Historia de Usuario	
Numero_6	**Nombre de historia de usuario:** Aplicar método de Secantes para aproximar una raíz.
Actor: Usuario	**Iteración asignada:**2
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: Una vez que el usuario selecciona el método de “Secantes” introduce los datos requeridos: intervalo, función y error, luego accionar el botón “Resolver”	
Observaciones: Los campos “función”, “a”, “b”, “x0”, “x1” y “error” no pueden estar vacíos.	
Prototipo de interfaz:	



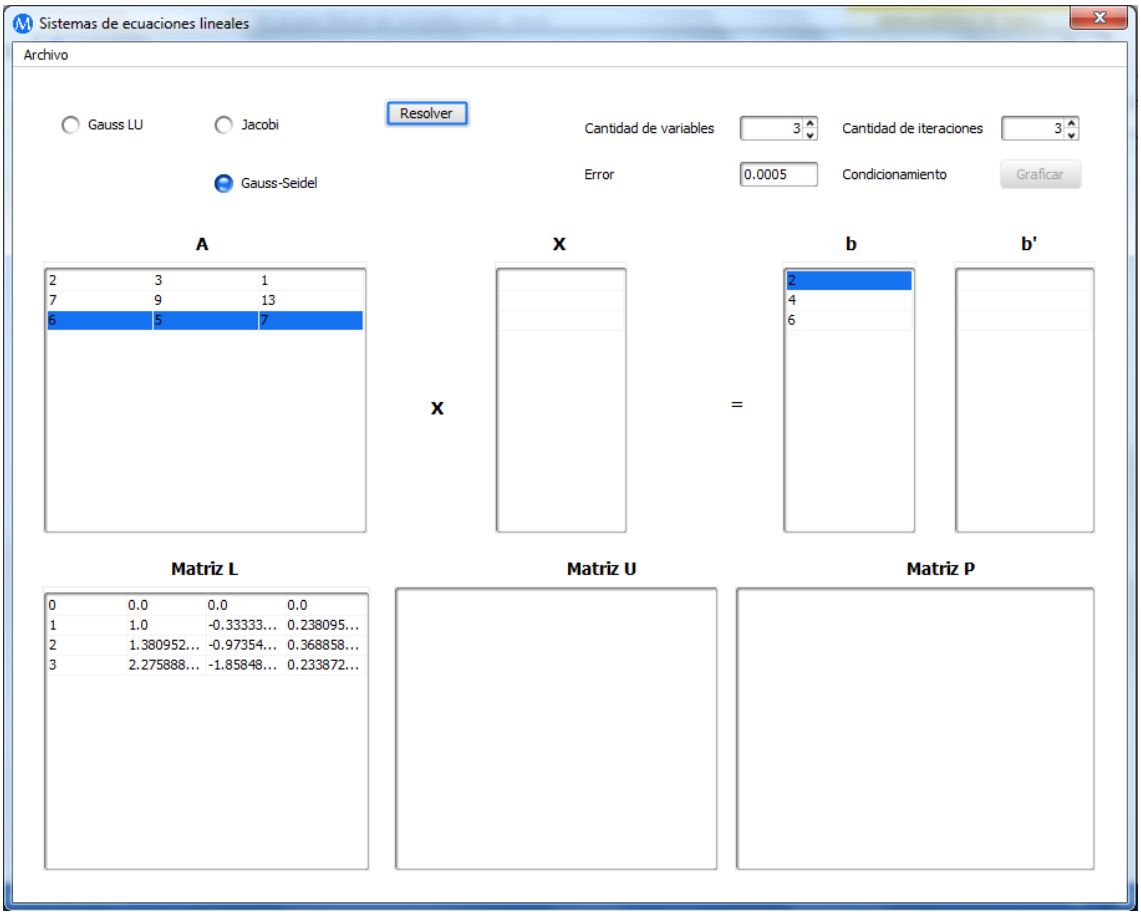
Historia de Usuario	
Numero_7	**Nombre de historia de usuario:** Aplicar método iterativo general para aproximar una raíz.
Actor: Usuario	**Iteración asignada:**2
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: Una vez que el usuario selecciona el método de "Iterativo general" introduce los datos requeridos: intervalo, función y error, luego accionar el botón "Resolver"	
Observaciones: Los campos "función", "a", "x0", "x1" y "iteraciones" no pueden estar vacíos.	
Prototipo de interfaz: 	

Historia de Usuario	
Numero_8	**Nombre de historia de usuario:** Aplicar método de Jacobi para aproximar el vector solución de un sistema de ecuaciones lineales (SEL).
Actor: Usuario	**Iteración asignada:**3
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Jacobi” introduce los datos requeridos: “iteraciones”, “matriz a”, “matriz b” y “error”, luego accionar el botón “Resolver”	
Observaciones: Los campos “iteraciones”, “matriz a”, “matriz b” y “error” no pueden estar vacíos.	
Prototipo de interfaz:	

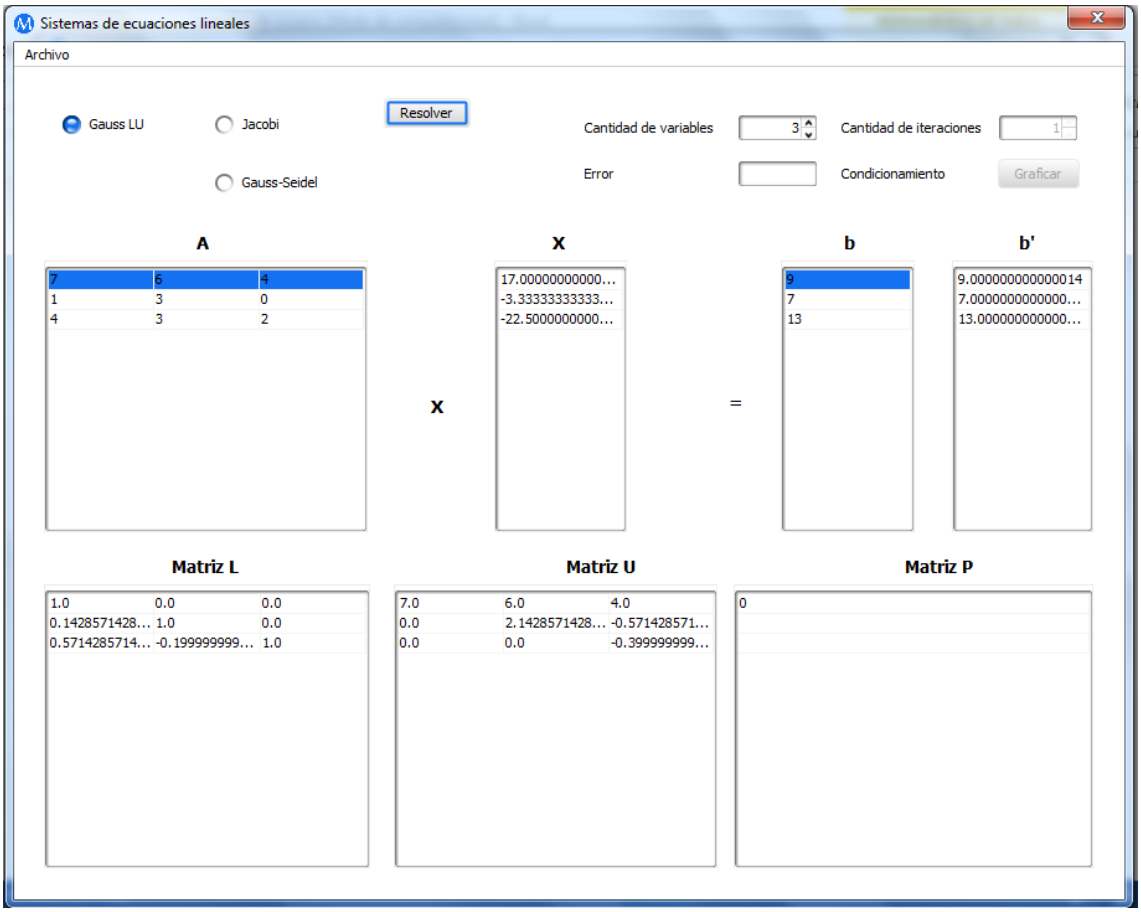


Historia de Usuario	
Numero_9	**Nombre de historia de usuario:** Aplicar método de Gauss-Seidel para aproximar el vector solución de un sistema de ecuaciones lineales (SEL).
Actor: Usuario	**Iteración asignada:**3
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de "Gauss-Seidel" introduce los datos requeridos: "matriz a" y "matriz b", luego accionar el botón "Resolver"	
Observaciones: Los campos "iteraciones", "matriz a", "matriz b" y "error" no pueden estar vacíos.	

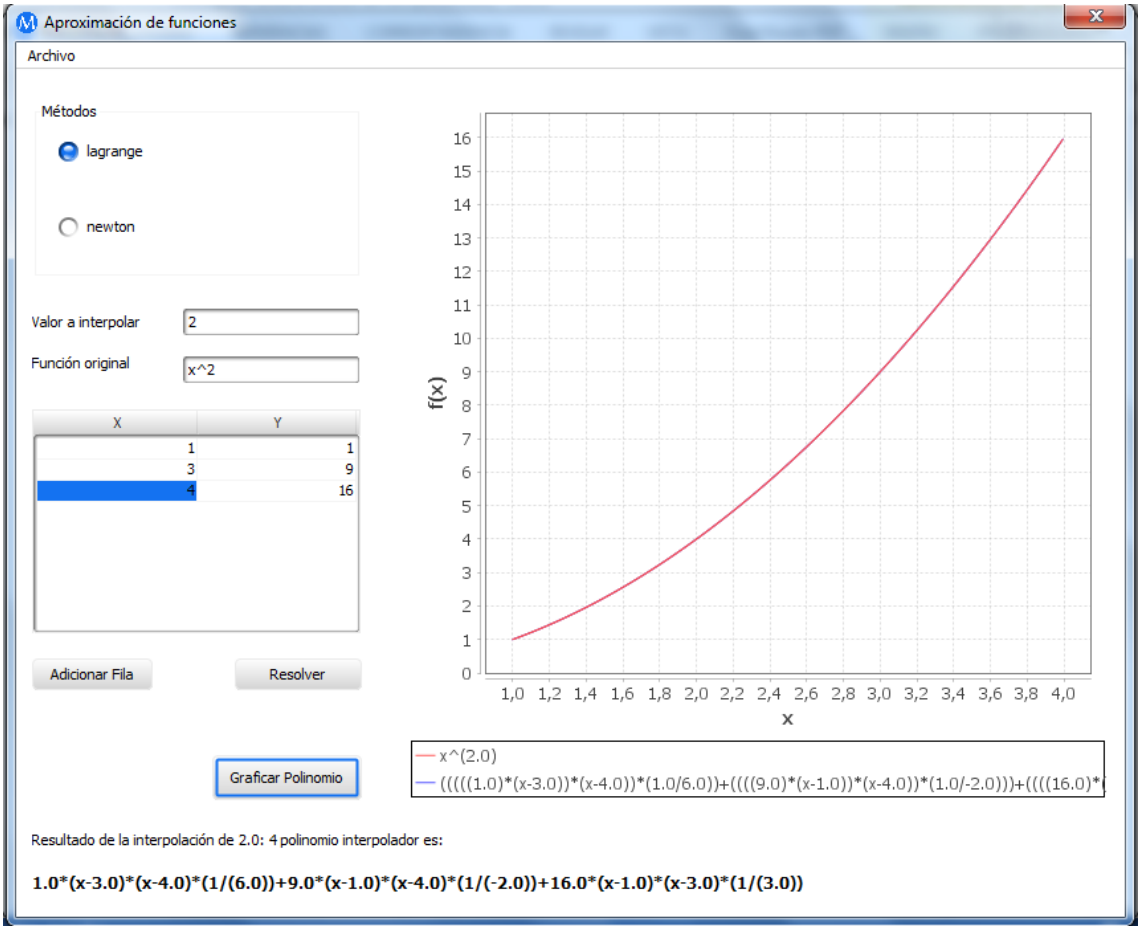
Prototipo de interfaz:



Historia de Usuario	
Numero_10	**Nombre de historia de usuario:** Aplicar método de Gauss LU para aproximar el vector solución de un sistema de ecuaciones lineales (SEL).
Actor: Usuario	**Iteración asignada:**3
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Gauss LU” introduce los datos requeridos: “matriz a” y “matriz b”, luego accionar el botón “Resolver”	

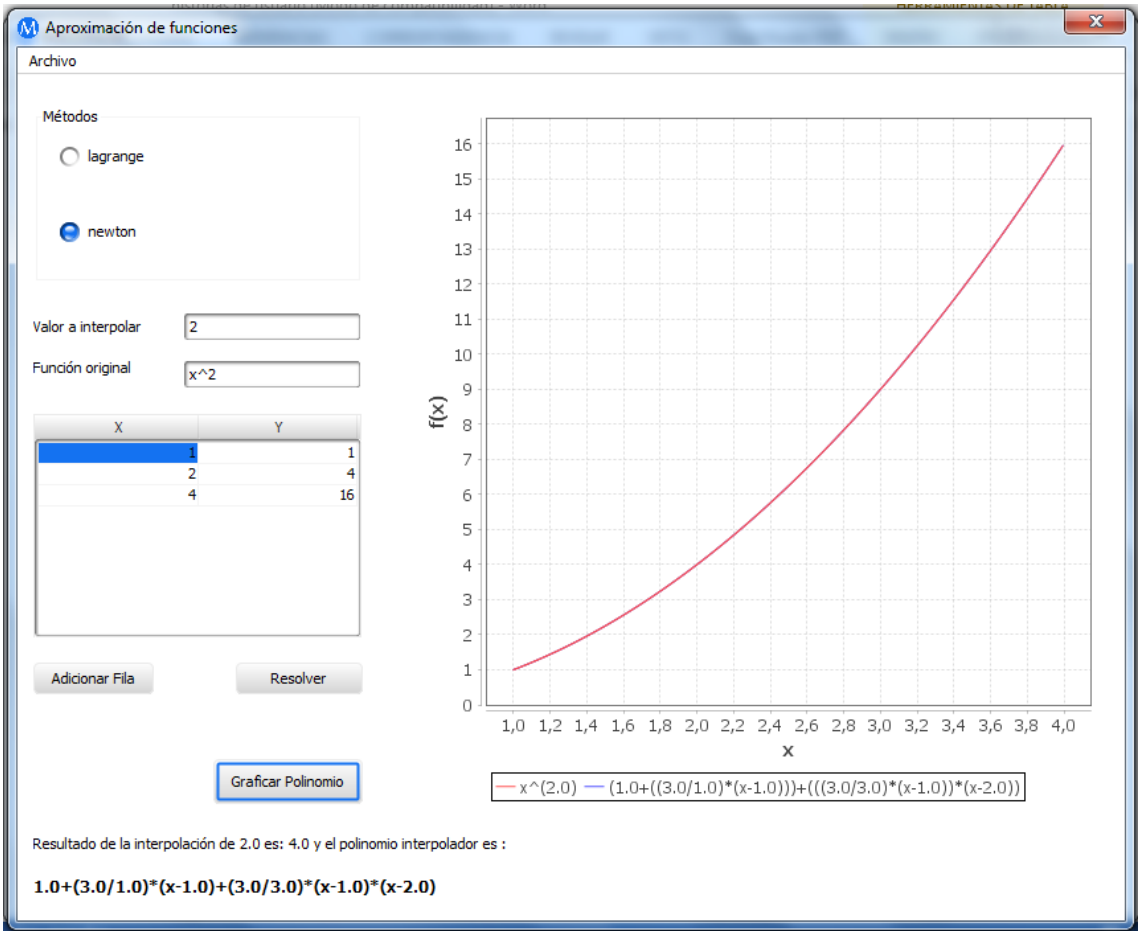
Observaciones: Los campos “iteraciones”, “matriz a”, “matriz b” y no pueden estar vacíos.
Prototipo de interfaz: 

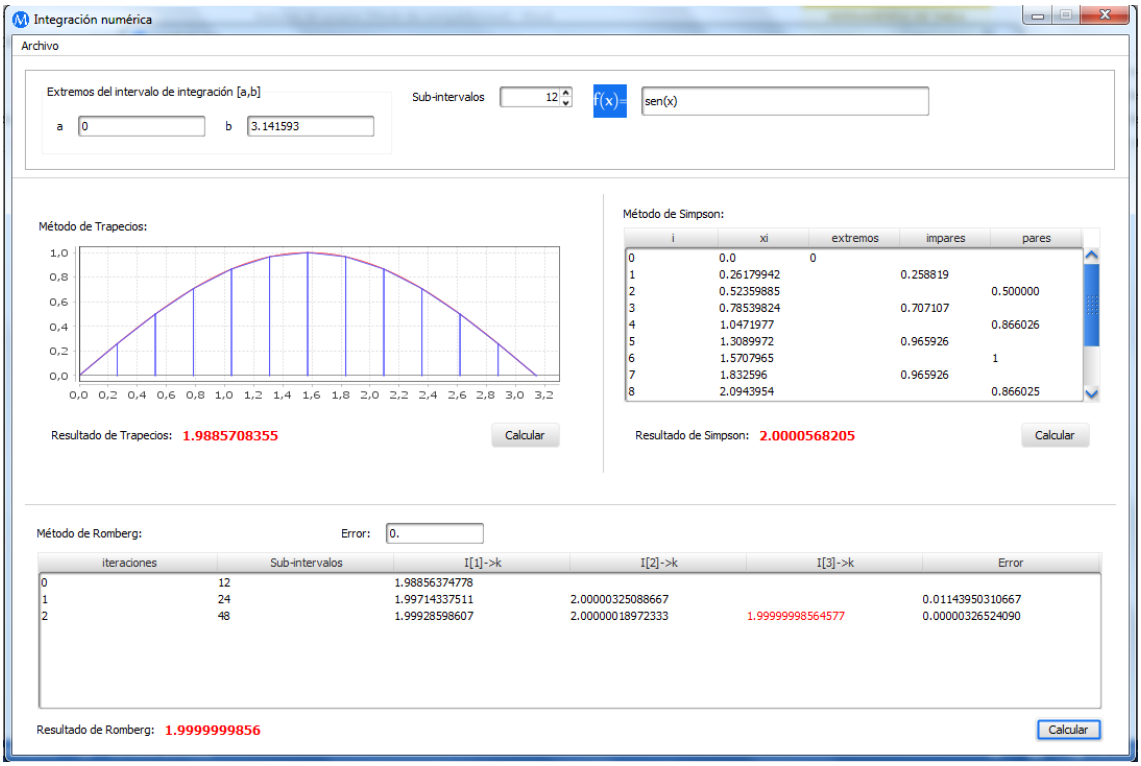
Historia de Usuario	
Numero_11	**Nombre de historia de usuario:** Aplicar método de Lagrange para determinar un polinomio interpolador.
Actor: Usuario	**Iteración asignada:**3
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5

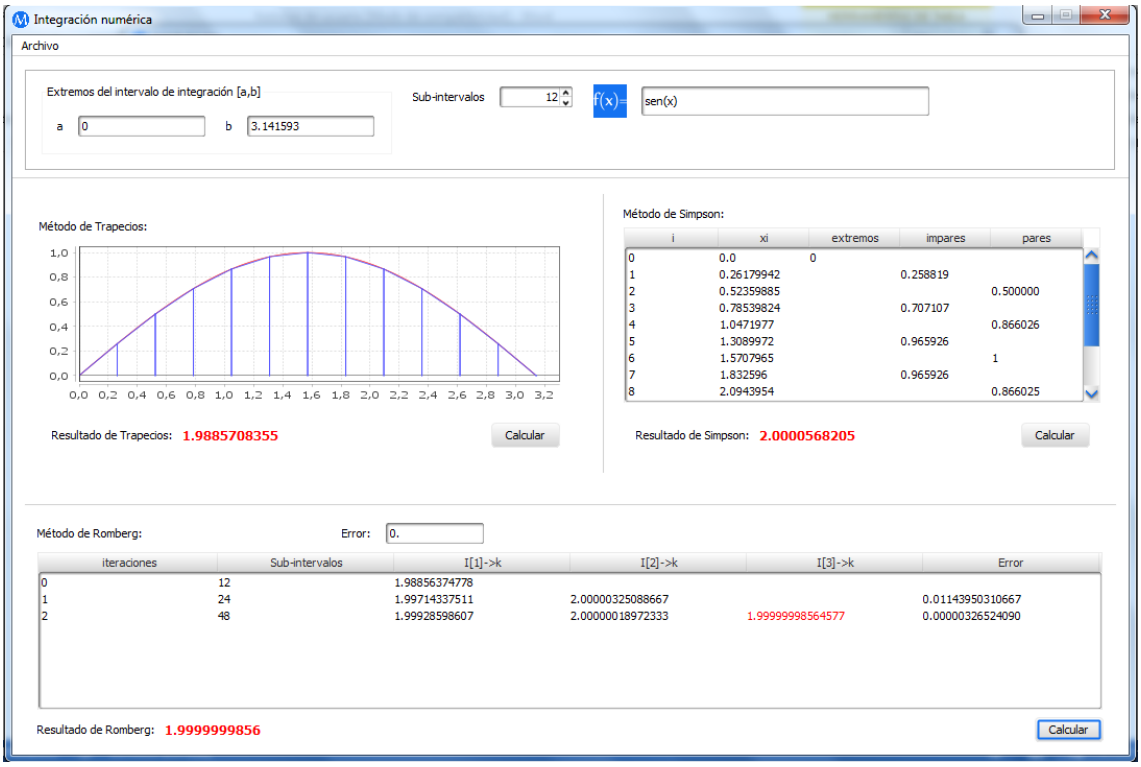
Descripción: Una vez que el usuario selecciona el método de “Lagrange” introduce los datos requeridos: “valor a interpolar”, “función original”, pares ordenados (x;y) accionar el botón “Resolver”.
Observaciones: Los campos “iteraciones”, “matriz a”, “matriz b” y “error” no pueden estar vacíos.
Prototipo de interfaz: 

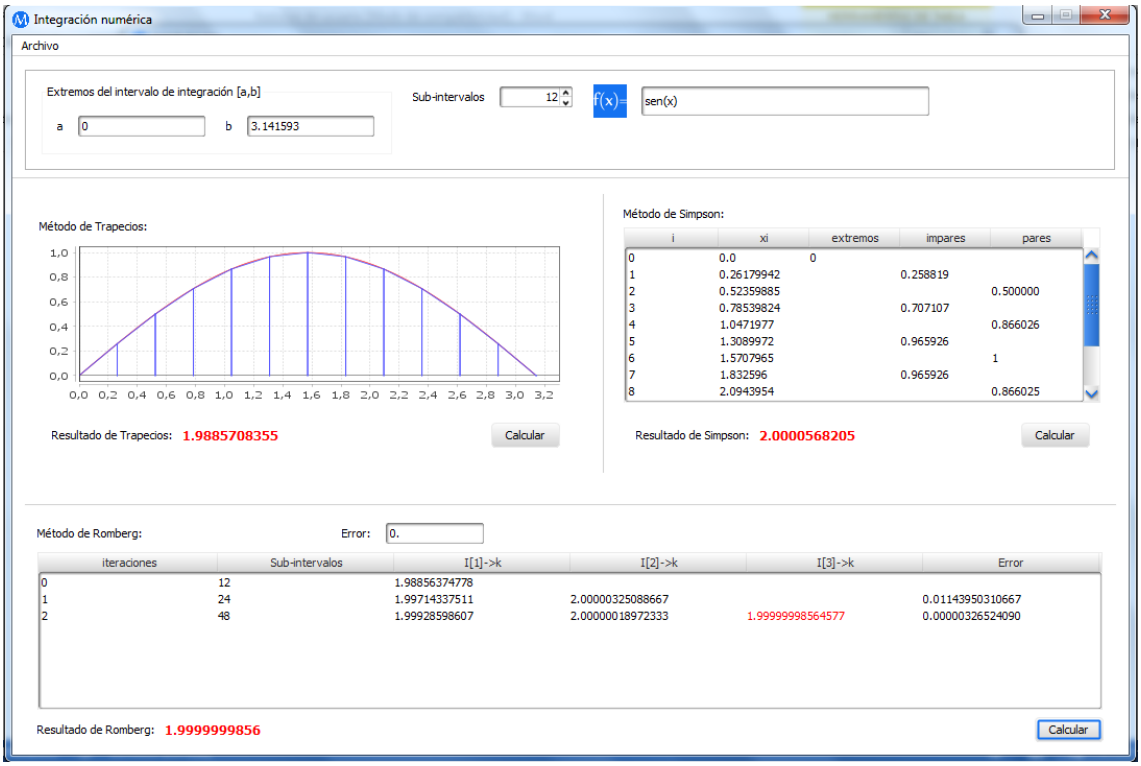
Historia de Usuario	
Numero_12	**Nombre de historia de usuario:** Aplicar método de Newton para determinar un polinomio interpolador.
Actor: Usuario	**Iteración asignada:**3

Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Lagrange” introduce los datos requeridos: “valor a interpolar”, “función original”, pares ordenados (x;y) accionar el botón “Resolver”.	
Observaciones: Los campos “iteraciones”, “matriz a”, “matriz b” y “error” no pueden estar vacíos.	
Prototipo de interfaz:	

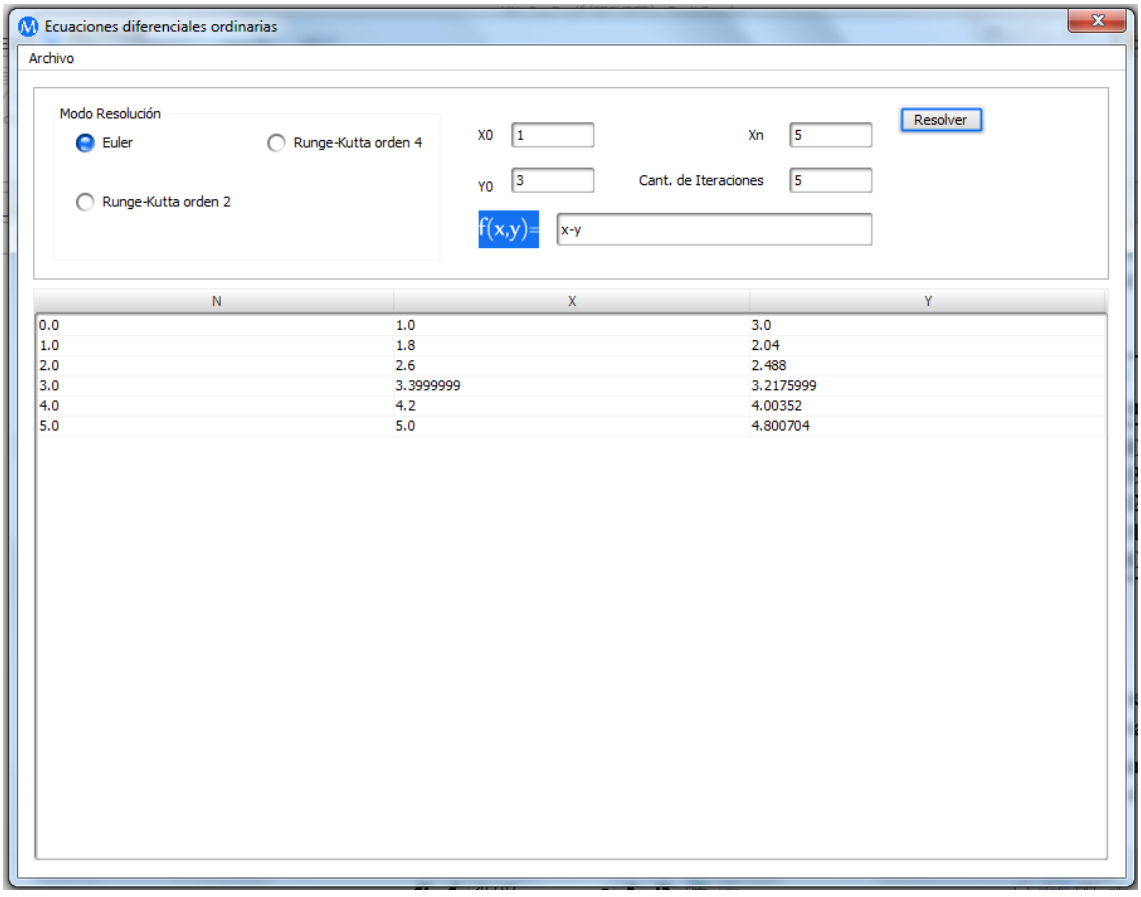


Historia de Usuario	
Numero_14	**Nombre de historia de usuario** Aplicar método de Simpson para aproximar el valor de una integral
Actor: Usuario	**Iteración asignada:**4
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Simpson” introduce los datos requeridos: intervalo, “función” y “subintervalos” accionar el botón “Resolver”.	
Observaciones: Los campos “función”, “a” y “b” no pueden estar vacíos.	
Prototipo de interfaz: 	

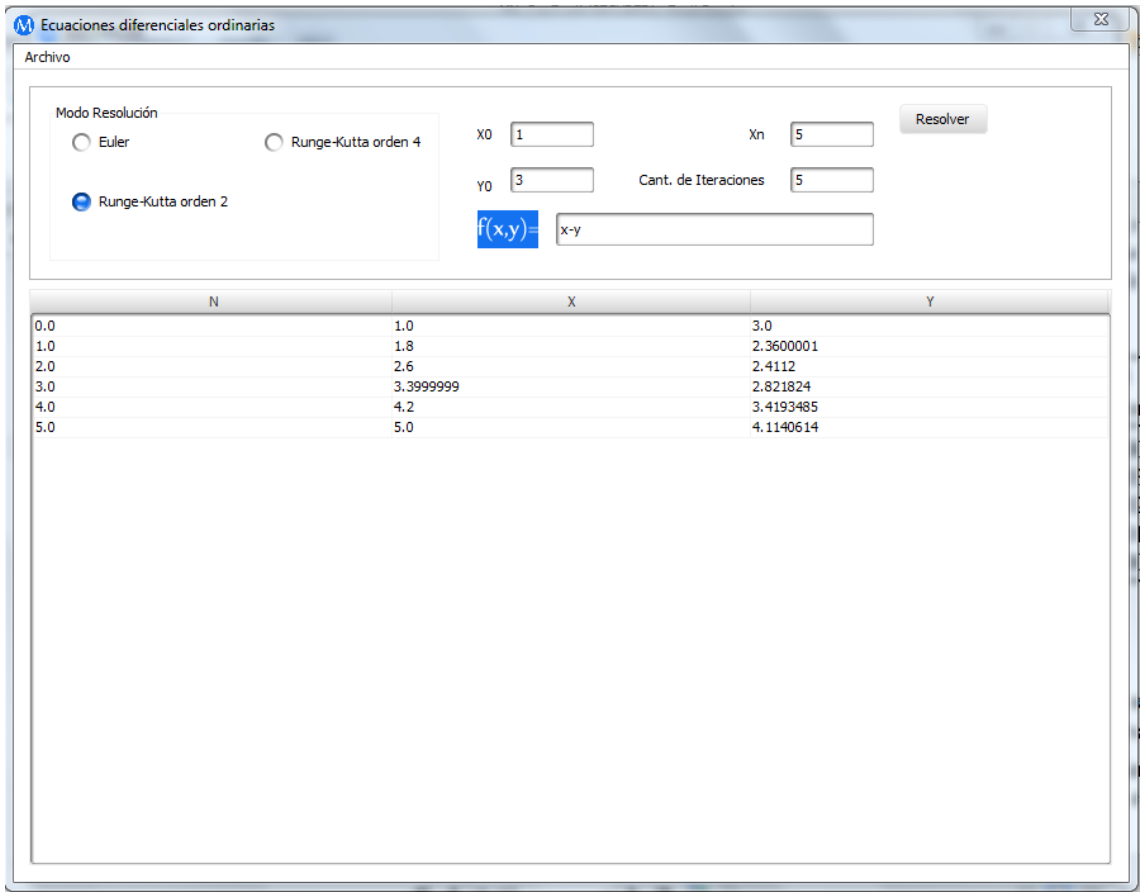
Historia de Usuario	
Numero_15	**Nombre de historia de usuario** Aplicar método de Romberg para aproximar el valor de una integral
Actor: Usuario	**Iteración asignada:**4
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Romberg” introduce los datos requeridos: intervalo, “función” y “subintervalos” accionar el botón “Resolver”.	
Observaciones: Los campos “función”, “a”, “b” y “error” no pueden estar vacíos.	
Prototipo de interfaz: 	

Historia de Usuario	
Numero_16	**Nombre de historia de usuario** Aplicar método de Trapecios para aproximar el valor de una integral
Actor: Usuario	**Iteración asignada:**4
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Trapecios” introduce los datos requeridos: intervalo, “función” y “subintervalos” accionar el botón “Resolver”.	
Observaciones: Los campos “función”, “a” y “b” no pueden estar vacíos.	
Prototipo de interfaz: 	

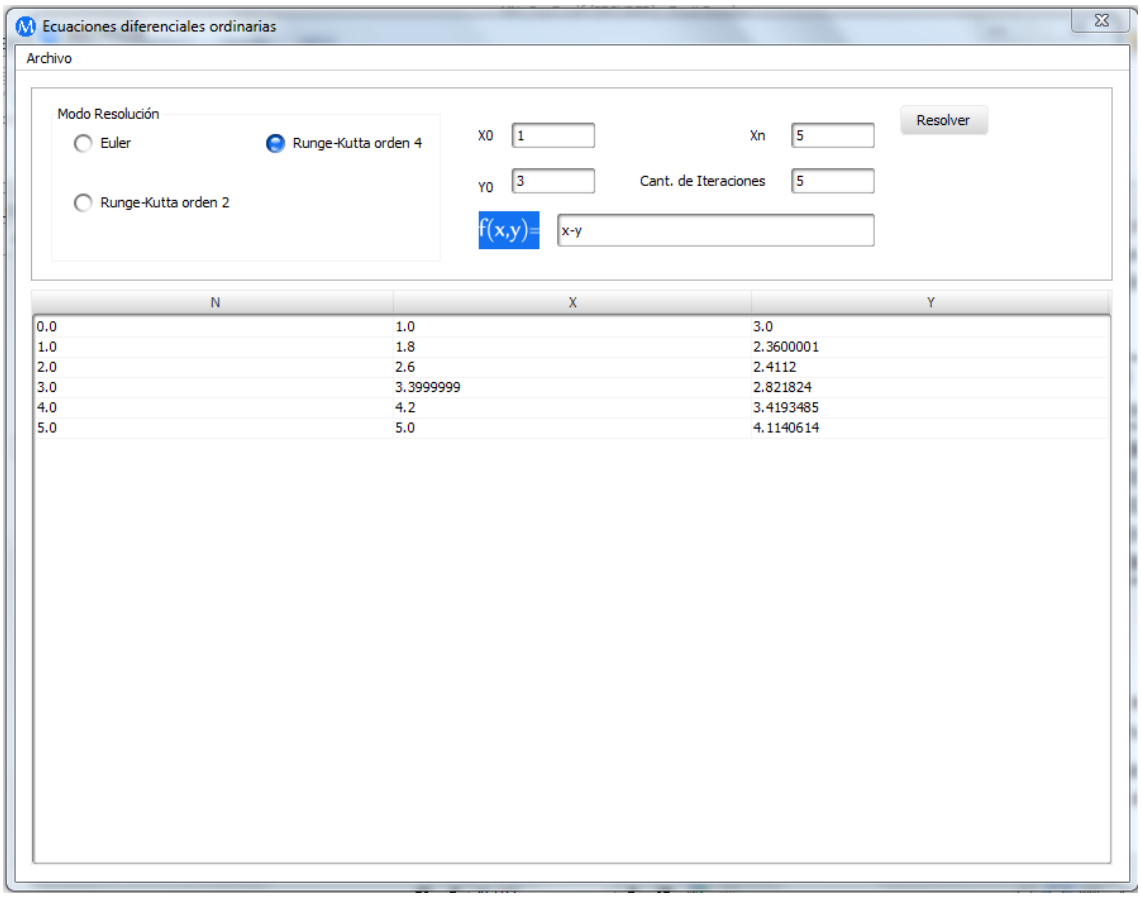
Historia de Usuario	
Numero_17	**Nombre de historia de usuario** Aplicar método de Euler para aproximar el valor de la solución de una ecuación diferencial.
Actor: Usuario	**Iteración asignada:**4
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Euler” introduce los datos requeridos: intervalo, “función” y “subintervalos” y valor inicial accionar el botón “Resolver”.	
Observaciones: Los campos “función”, “Xo” y “Xn” no pueden estar vacíos.	
Prototipo de interfaz:	

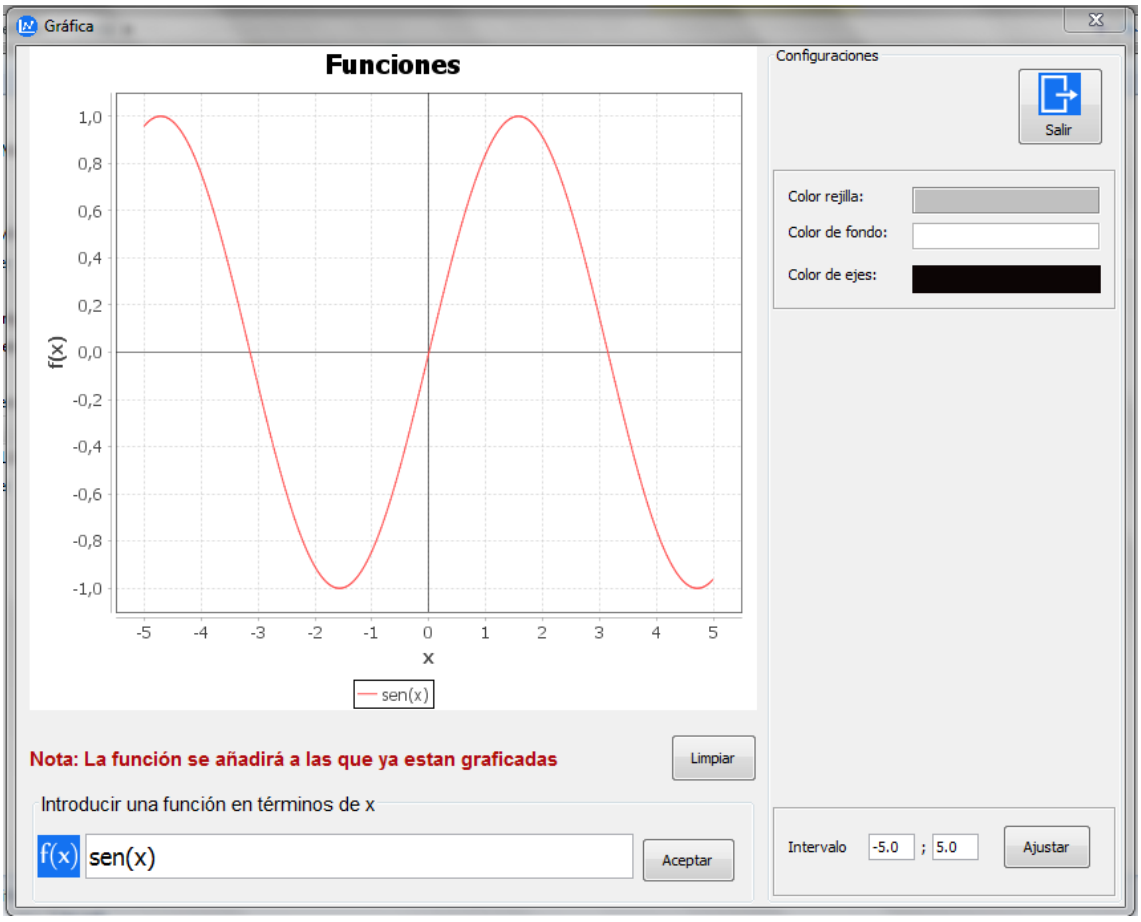


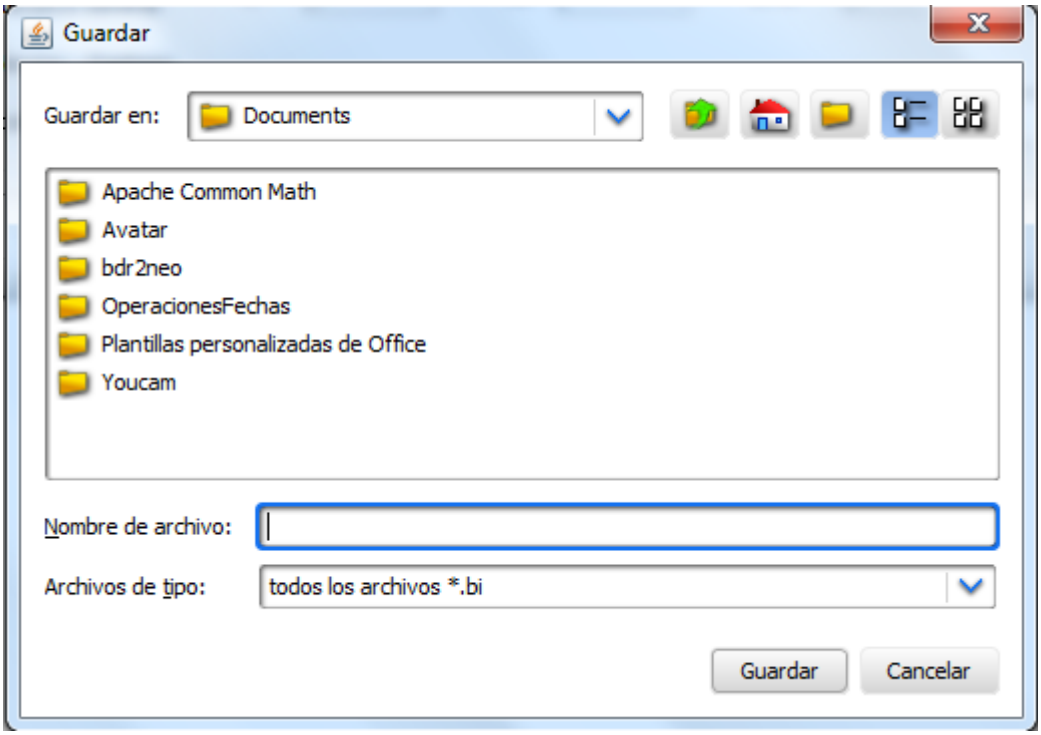
Historia de Usuario	
Numero_18	**Nombre de historia de usuario** Aplicar método de Runge Kutta orden 2 para aproximar el valor de la solución de una ecuación diferencial.
Actor: Usuario	**Iteración asignada:**4
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Runge Kutta orden 2” introduce los datos requeridos: intervalo, “función” y “subintervalos” y valor inicial accionar el botón “Resolver”.	
Observaciones: Los campos “función”, “Xo” y “Xn” no pueden estar vacíos.	
Prototipo de interfaz:	

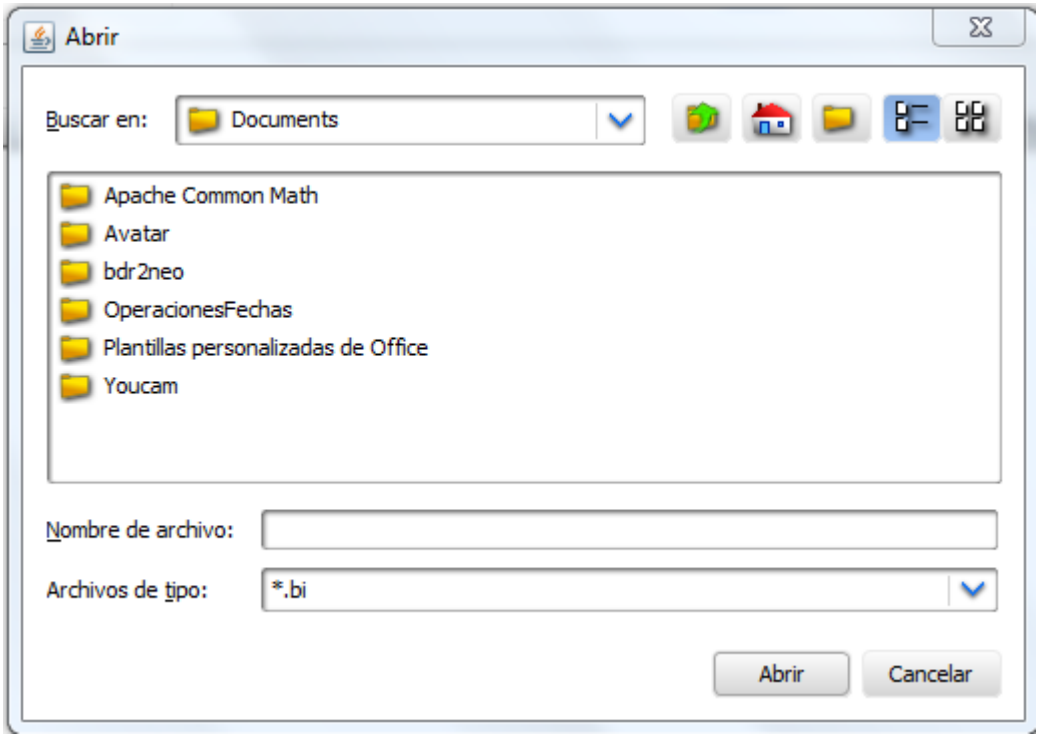


Historia de Usuario	
Numero_19	**Nombre de historia de usuario** Aplicar método de Runge Kutta orden 4 para aproximar el valor de la solución de una ecuación diferencial.
Actor: Usuario	**Iteración asignada:**4
Prioridad de negocio: Alta	**Puntos estimados:**5
Nivel de complejidad: Alta	**Puntos reales:**5
Descripción: Una vez que el usuario selecciona el método de “Runge Kutta orden 4” introduce los datos requeridos: intervalo, “función” y “subintervalos” y valor inicial accionar el botón “Resolver”.	
Observaciones: Los campos “función”, “Xo” y “Xn” no pueden estar vacíos.	
Prototipo de interfaz:	

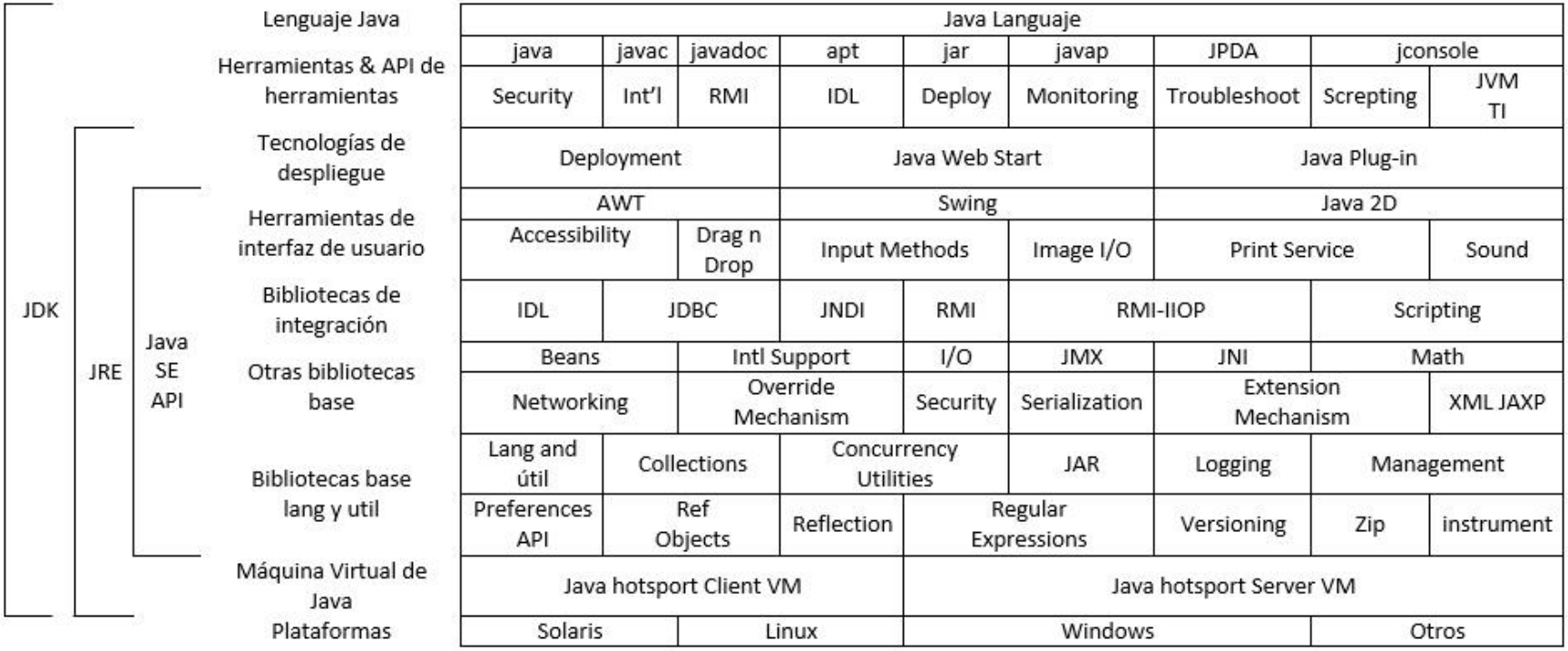


Historia de Usuario	
Numero_20	**Nombre de historia de usuario:** Graficar funciones reales.
Actor: Usuario	**Iteración asignada:**1
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: El usuario debe llenar el f(x) campo y accionar el botón aceptar, en el panel principal se dibujara la función indicada.	
Observaciones: El campo "f(x)" no puede estar vacío.	
Prototipo de interfaz: 	

Historia de Usuario	
Numero_21	**Nombre de historia de usuario:** Guardar en fichero ejemplos de los métodos matemáticos.
Actor: Usuario	**Iteración asignada:**4
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: El usuario selecciona la opción guardar fichero en el menú	
Observaciones: La interfaz no debe contener campos vacíos	
Prototipo de interfaz: 	

Historia de Usuario	
Numero_22	**Nombre de historia de usuario:** Cargar en fichero ejemplos de los métodos matemáticos.
Actor: Usuario	**Iteración asignada:**4
Prioridad de negocio: Alta	**Puntos estimados:**3
Nivel de complejidad: Alta	**Puntos reales:**3
Descripción: El usuario selecciona la opción cargar fichero en el menú	
Observaciones: Se cargan los archivos con la extensión correspondiente a la interfaz en la que se está.	
Prototipo de interfaz: 	

Anexo3: Arquitectura de java



Anexo4: Tarjetas de CRC

TARJETA CRC	
Clase	Biseccion.java
Súper Clase	MetodoUnidad1.java
Responsabilidades	**Colaboraciones**
Calcular las aproximaciones a una raíz real de una función definida aplicando el método de Bisección.	MetodoInterfaz.java ReconocedorFuncion.jar

TARJETA CRC	
Clase	GaussSeidel.java
Súper Clase	MetodoUnidad2.java
Responsabilidades	**Colaboraciones**
Calcular las aproximaciones del vector solución de un Sistema de Soluciones Lineales aplicando el método de Gauss-Seidel.	MetodoInterfaz2.java ReconocedorFuncion.jar

TARJETA CRC	
Clase	Lagrange.java
Súper Clase	MetodoUnidad3.java
Responsabilidades	**Colaboraciones**
Determina el polinomio interpolador dado un las aproximaciones a una raíz real de una función definida.	MetodoInterfaz3.java ReconocedorFuncion.jar

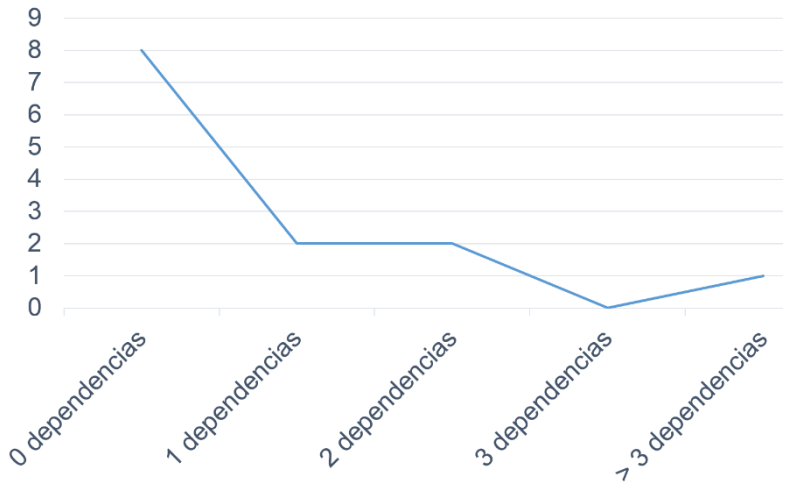
TARJETA CRC	
Clase	Simpson.java
Súper Clase	MetodoUnidad4.java
Responsabilidades	**Colaboraciones**
Calcular las aproximaciones de la integral de una función definida.	MetodoInterfaz3.java ReconocedorFuncion.jar

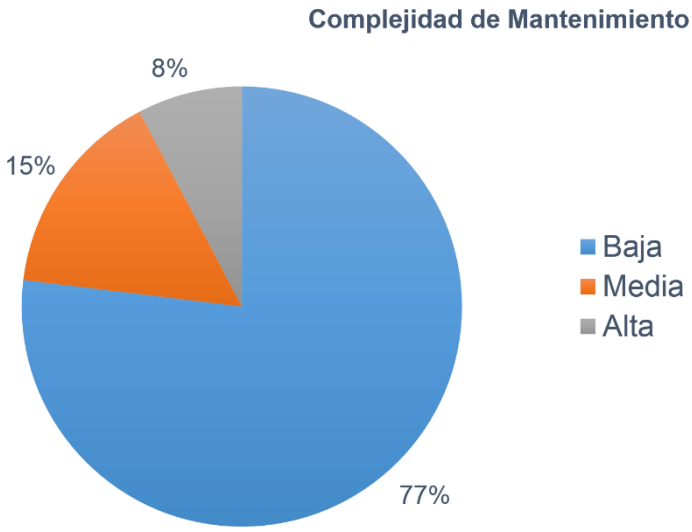
TARJETA CRC	
Clase	Euler.java
Súper Clase	MetodoUnidad4.java
Responsabilidades	**Colaboraciones**
Calcular las aproximaciones del conjunto solución de una ecuación diferencial ordinaria	ReconocedorFuncion.jar

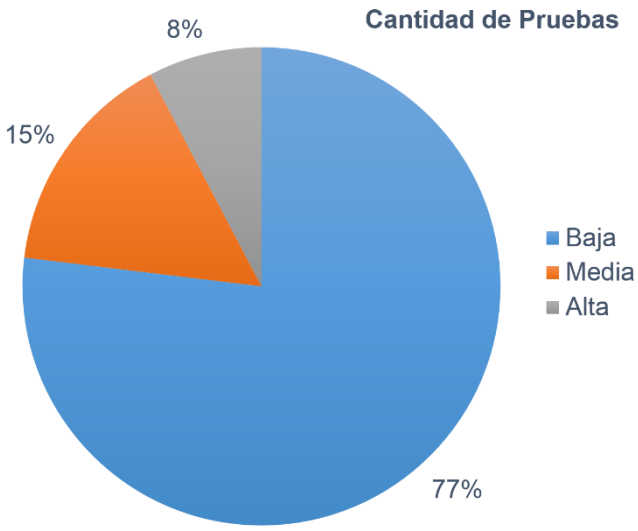
TARJETA CRC	
Clase	graficacion.java
Súper Clase	
Responsabilidades	**Colaboraciones**
Calcular las aproximaciones del conjunto solución de una ecuación diferencial ordinaria	jfreechart.jar ReconocedorFuncion.jar

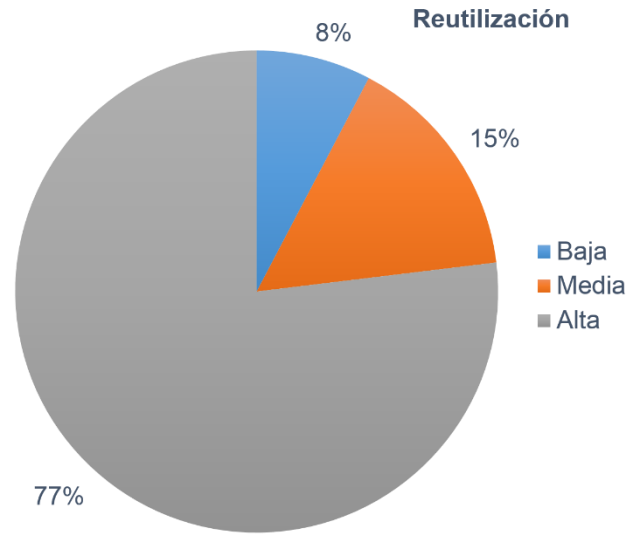
TARJETA CRC	
Clase	CargarArchivo.java
Súper Clase	
Responsabilidades	**Colaboraciones**
Guardar en un fichero de texto plano los datos necesarios para el cálculo o resolución del método indicado.	

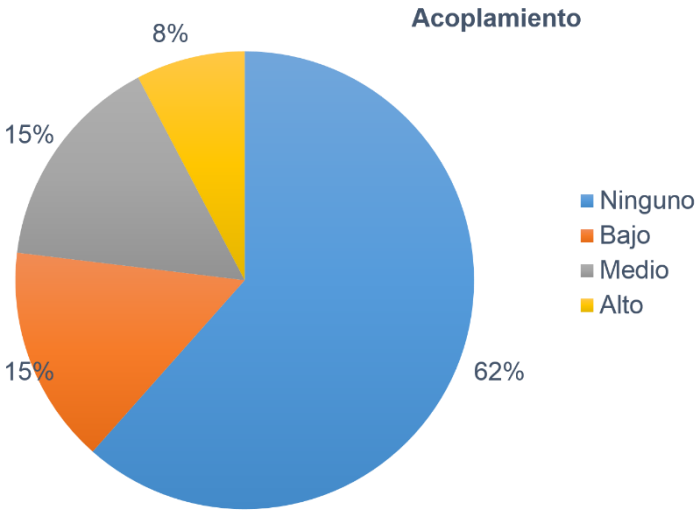
Anexo5: Resultados para la validación del diseño.




Complejidad de Mantenimiento
8%
15%
77%
Baja
Media
Alta


Cantidad de Pruebas
8%
15%
77%
Baja
Media
Alta





Anexo6: Gramática en LL1

N: {<Termino>; <MasTermino>; <Factor>; <MasFactor>; <ExpFuncion>; <MasExponenciacion>; <TerminoTrigonometrico>; <TerminoRadical>; <TerminoExponencial>; <FuncionInterna>; <Constante>; <Signo>; <TerminoPolinomial>; <sen>; <cos>; <tan>; <arcsen>; <arccos>; <arctan>; <Signo>}

∑{tk_suma, tk_resta, tk_multiplicacion, tk_division, tk_exp, tk_par_abre, tk_par_cierra, tk_entero, tk_flotante, tk_sen, tk_cos, tk_tan, tk_arcsen, tk_arccos, tk_arctan, tk_e, tk_ln, tk_log, tk_raiz, tk_coma, tk_sgn}

P {<Funcion>}

S{tk_suma, tk_resta, tk_multiplicacion, tk_division, tk_exp, tk_par_abre, tk_par_cierra, tk_entero, tk_flotante, tk_sen, tk_cos, tk_tan, tk_arcsen, tk_arccos, tk_arctan, tk_e, tk_ln, tk_log, tk_raiz, tk_coma, tk_sgn, <Funcion>; <Termino>; <MasTermino>; <Factor>; <MasFactor>; <ExpFuncion>; <MasExponenciacion>; <TerminoTrigonometrico>; <TerminoRadical>; <TerminoExponencial>; <FuncionInterna>; <Constante>; <Signo>; <TerminoPolinomial>; <sen>; <cos>; <tan>; <arcsen>; <arccos>; <arctan>; <Signo>}

G= (N, ∑, P, S):

<Funcion> -> <Termino><MasTermino>

<Termino> -> <Factor> <MasFactor>

<MasTermino> -> tk_suma <Termino><MasTermino> | tk_resta <Termino> <MasTermino>

<Factor> -> <ExpFuncion><MasExponenciacion>

<MasFactor> -> tk_multiplicacion <Factor> <MasFactor> | tk_division <Factor> <MasFactor>

<ExpFuncion> -> <TerminoTrigonometrico> | <TerminoRadical> | <TerminoExponencial> | <FuncionInterna> | <Constante> | <Signo> | <TerminoPolinomial>

<MasExponenciacion> -> tk_exp <ExpFuncion> <MasExponenciacion>

<FuncionInterna> -> tk_par_abre <Funcion> tk_par_cierra

<Constante> -> tk_entero | tk_flotante

<TerminoTrigonometrico> -> <sen> | <cos> | <tan> | <arcsen> | <arccos> | <arctan>

<sen> -> tk_sen tk_par_abre <Funcion> tk_par_cierra

<cos> -> tk_cos tk_par_abre <Funcion> tk_par_cierra

<tan> -> tk_tan tk_par_abre <Funcion> tk_par_cierra

<arcsen> -> tk_arcsen tk_par_abre <Funcion> tk_par_cierra

<arccos> -> tk_arccos tk_par_abre <Funcion> tk_par_cierra

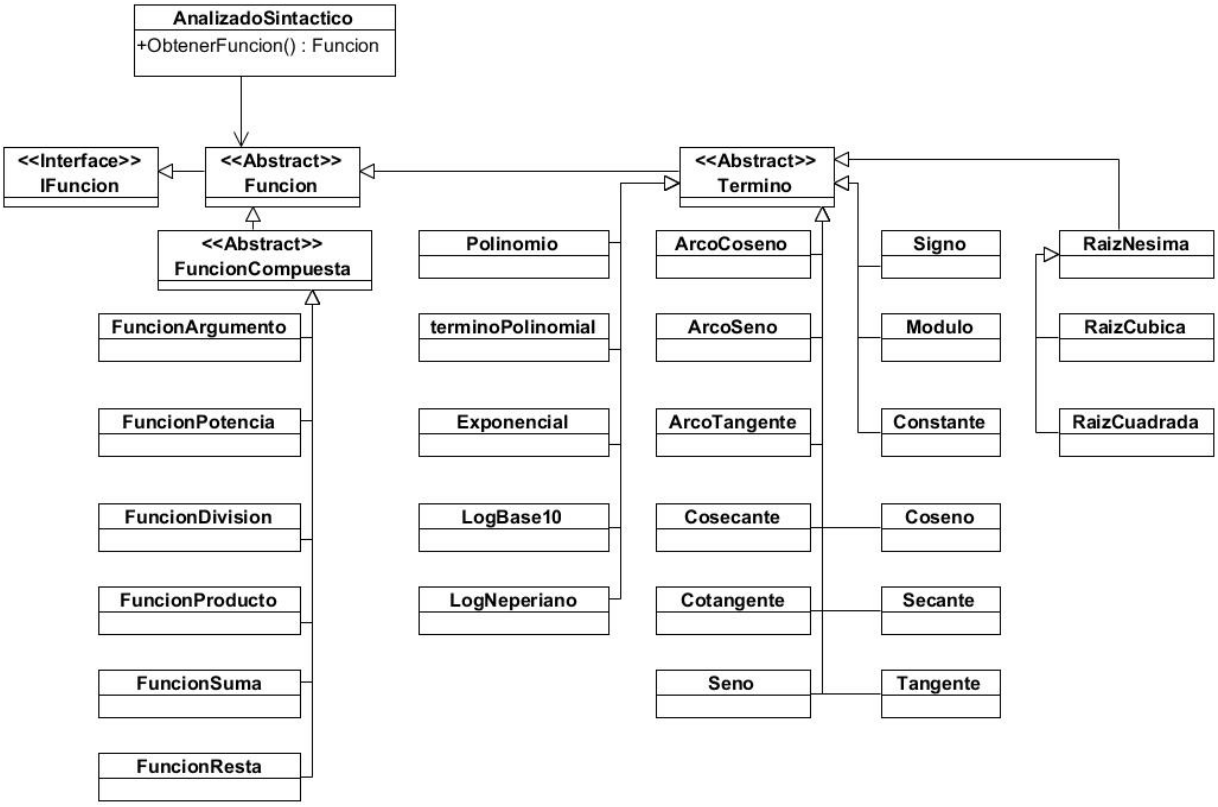
<arctan> -> tk_arctan tk_par_abre <Funcion> tk_par_cierra

<TerminoExponencial> -> tk_e | tk_ln | tk_log

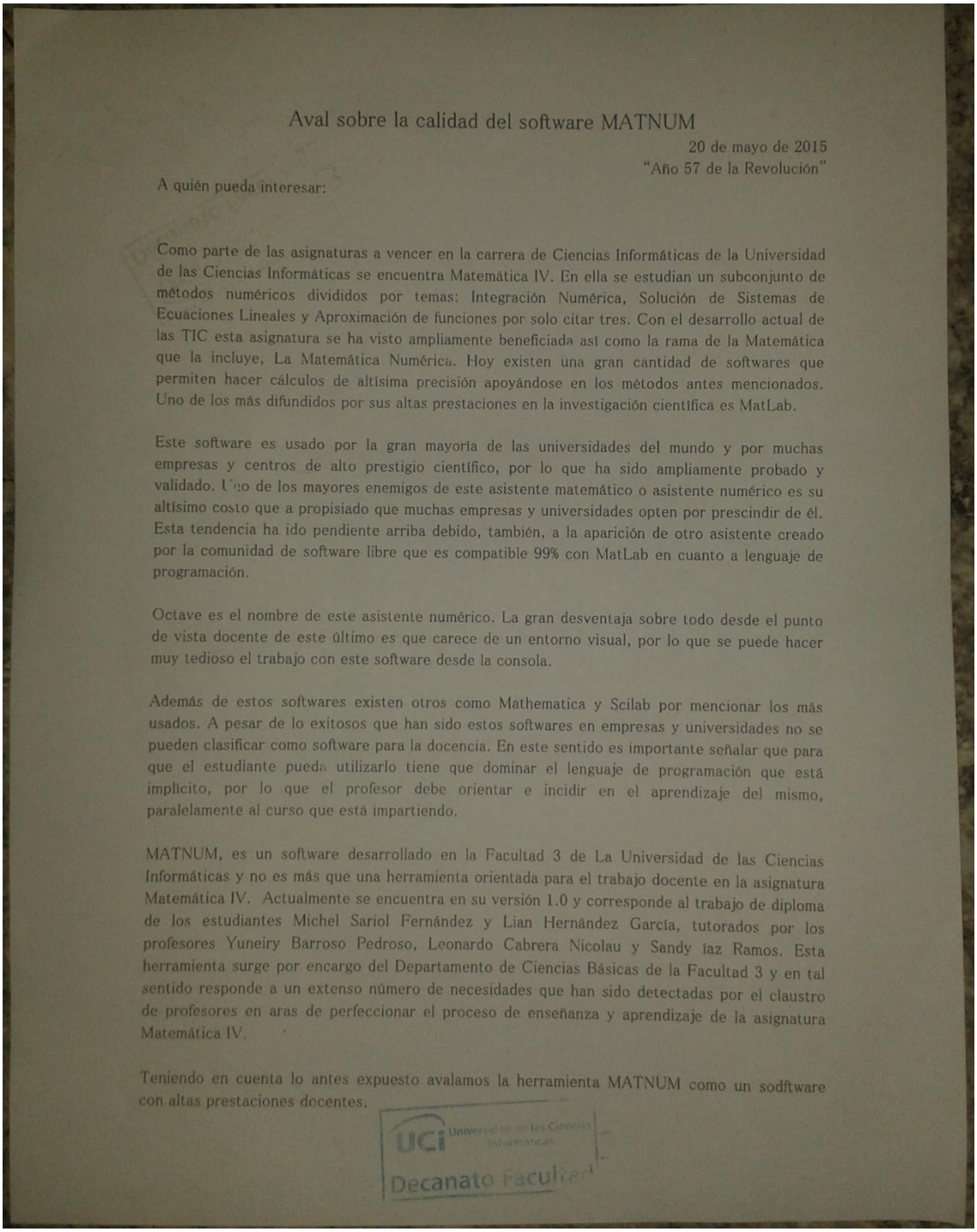
<TerminoRadical> -> tk_raiz tk_par_abre <Constante> tk_coma <Funcion> tk_par_cierra

<Signo> -> tk_sgn tk_par_abre <Funcion> tk_par_cierra

Anexo7: Mapa conceptual del patrón intérprete



Anexo8: Carta de aceptación



Aval sobre la calidad del software MATNUM

20 de mayo de 2015
"Año 57 de la Revolución"

A quién pueda interesar:

Como parte de las asignaturas a vencer en la carrera de Ciencias Informáticas de la Universidad de las Ciencias Informáticas se encuentra Matemática IV. En ella se estudian un subconjunto de métodos numéricos divididos por temas: Integración Numérica, Solución de Sistemas de Ecuaciones Lineales y Aproximación de funciones por solo citar tres. Con el desarrollo actual de las TIC esta asignatura se ha visto ampliamente beneficiada así como la rama de la Matemática que la incluye, La Matemática Numérica. Hoy existen una gran cantidad de softwares que permiten hacer cálculos de altísima precisión apoyándose en los métodos antes mencionados. Uno de los más difundidos por sus altas prestaciones en la investigación científica es MatLab.

Este software es usado por la gran mayoría de las universidades del mundo y por muchas empresas y centros de alto prestigio científico, por lo que ha sido ampliamente probado y validado. Uno de los mayores enemigos de este asistente matemático o asistente numérico es su altísimo costo que a propisiado que muchas empresas y universidades opten por prescindir de él. Esta tendencia ha ido pendiente arriba debido, también, a la aparición de otro asistente creado por la comunidad de software libre que es compatible 99% con MatLab en cuanto a lenguaje de programación.

Octave es el nombre de este asistente numérico. La gran desventaja sobre todo desde el punto de vista docente de este último es que carece de un entorno visual, por lo que se puede hacer muy tedioso el trabajo con este software desde la consola.

Además de estos softwares existen otros como Mathematica y Scilab por mencionar los más usados. A pesar de lo exitosos que han sido estos softwares en empresas y universidades no se pueden clasificar como software para la docencia. En este sentido es importante señalar que para que el estudiante pueda utilizarlo tiene que dominar el lenguaje de programación que está implícito, por lo que el profesor debe orientar e incidir en el aprendizaje del mismo, paralelamente al curso que está impartiendo.

MATNUM, es un software desarrollado en la Facultad 3 de La Universidad de las Ciencias Informáticas y no es más que una herramienta orientada para el trabajo docente en la asignatura Matemática IV. Actualmente se encuentra en su versión 1.0 y corresponde al trabajo de diploma de los estudiantes Michel Sariol Fernández y Lian Hernández García, tutorados por los profesores Yuneiry Barroso Pedroso, Leonardo Cabrera Nicolau y Sandy Iaz Ramos. Esta herramienta surge por encargo del Departamento de Ciencias Básicas de la Facultad 3 y en tal sentido responde a un extenso número de necesidades que han sido detectadas por el claustro de profesores en aras de perfeccionar el proceso de enseñanza y aprendizaje de la asignatura Matemática IV.

Teniendo en cuenta lo antes expuesto avalamos la herramienta MATNUM como un sodftware con altas prestaciones docentes.

UCI Universidad de las Ciencias Informáticas
Decanato Facultad