UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"DESARROLLO DE COMPONENTES SOFTWARE COM Y SOAP, DISTRIBUIDOS Y HETEROGÉNEOS PARA ENCAPSULAR LA FUNCIONALIDAD BÁSICA Y SUS TIPOS DE DATOS DE MATLAB, Y SOPORTAR LA INTEGRACIÓN DE NUEVOS COMPONENTES COM-SOAP"

T E S I S

PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A

JOSEFINA FLORES FLORES

DIRECTORES DE TESIS

M.C. DAVID MARTÍNEZ TORRES
M.C. EMILIO GARCÍA ROSELLÓ

HUAJUAPAN DE LEÓN, OAX., DICIEMBRE DE 2006

Tesis presentada el 15 de diciembre de 2006 ante los siguientes sinodales:

M. C. Everth Haydeé Rocha Trejo.
M. C. Irma Guadalupe Solís Genesta.
M. C. Ricardo Ruiz Rodríguez.

Directores de Tesis:

M. C. David Martínez Torres.
M. C. Emilio García Roselló.

Dedicatoria

A mis padres **Toribio y Eustolia** porque ellos fueron la parte esencial de mi formación, porque gracias a su apoyo, amor y confianza pude lograr las metas que me propuse y llegar a ser lo que soy ahora, porque son un claro ejemplo de lucha y dedicación porque simplemente me siento plena y orgullosa de ser su hija…

Cada instante es más bello al pensar en ustedes, los amo con todo mi corazón.

A mis hermanos **Fernando, Ana Lilia y Azucena**, porque gracias a sus claros ejemplos de vida pude llevarme lo mejor de cada uno, me ayudaron a crecer, me empaparon de sus experiencias a lo largo de mi crecimiento, porque con sus deseos de lucha me enseñaron a buscar siempre mas allá de lo posible, porque me amaron y apoyaron en cada momento de mi vida, porque la distancia no fue impedimento para no sentirnos mas cerca que nunca, porque son mi orgullo y apoyo para seguir adelante…

Gracias por su claro ejemplo, los amo con todo mi ser.

A mis sobrinos **Axel y Roberto** porque mediante sus pequeñas sonrisas y hermosas bienvenidas me han hecho sentir la persona más especial del mundo, porque un abrazo de esas pequeñas manos te llena de una inmensa dicha y felicidad, porque la ansiedad de verlos crecer siempre fue un claro motivo para seguir adelante…

Gracias pequeños por ser parte de mi vida, los amo.

Agradezco especialmente a:

Dios, por que me ha permitido despertar cada día, porque me siento afortunada y agradecida por todo lo que me ha dado, por darme la grandiosa familia que me tocó tener, porque me ama y está conmigo en cada día de mi vida. Gracias Dios por ser parte de mi vida, por permitirme cumplir con mis objetivos y metas planteadas a lo largo de mi camino.

Mi familia, por ser la piedra angular en mi formación, porque sin ella no existiría, porque gracias a su confianza, apoyo y amor, he logrado culminar con esta etapa de mi vida, porque me ayuda a seguir adelante y comenzar el nuevo camino que está por venir. Gracias por ser mi orgullo, ejemplo a seguir y por permitirme lograr mi formación profesional y personal.

Felipe, por estar conmigo en esos difíciles momentos, por ser mi confidente y amigo, porque hemos compartido enormes tristezas y alegrías, porque eres una parte muy importante en mi vida… Gracias.

Agradecimientos

A **David**, porque mediante tu dedicación, guía y paciencia logramos cumplir con este objetivo, porque fuiste más que un profesor un gran amigo, gracias por permitirme compartir esta etapa de mi vida contigo, y por dejarme conocer a la enorme persona y excelente profesionista que eres. Muchas gracias hoy y siempre.

A **Emilio**, porque sus aportaciones para este proyecto fueron de enorme utilidad, porque su apoyo durante este desarrollo es pieza clave para haber logrado el objetivo final, porque la distancia no fue impedimento para tener una respuesta rápida hacia una duda y poder continuar. Gracias Emilio por tu enorme ayuda, paciencia y profesionalismo.

A mis sinodales **Everth, Irma y Ricardo,** por el tiempo dedicado a este trabajo y por los sabios y atinados comentarios dados con la finalidad de lograr el mejor resultado.

A la **Universidad Tecnológica de la Mixteca**, quien me hizo formarme como persona y crecer internamente, porque pude ver que el carácter, la disciplina y el trabajo son la clave fundamental para el desarrollo y crecimiento de cada individuo.

Índice

Lista de figuras

Lista de tablas

Prólogo

La tendencia actual del software educativo es claramente orientada hacia un paradigma basado en componentes reutilizables, con la finalidad de lograr un modelo de desarrollo de aplicaciones efectivo en tiempo y recursos y, al mismo tiempo, tratar de alcanzar los objetivos educativos que se reclaman a las tecnologías de la información. A pesar de esto y de que existen muchos candidatos a componentes que ofrecen una funcionalidad interesante para ser reutilizados, en muchos casos son aplicaciones propietarias de difícil integración debido a su dependencia de una interfaz de usuario y las limitaciones con las API ofrecidas [17] .

En este trabajo se abordará en específico a Matlab, uno de esos lenguajes propietarios. Aunque Matlab es un potente lenguaje que ofrece un sin fin de funciones matemáticas que pueden ser utilizadas como componentes en un amplio rango de dominios, ofrece también una limitada posibilidad de poder ser integrado con otras aplicaciones, de ahí que se propone resolver tal limitación mediante el uso de la tecnología COM-SOAP[1].

Para el desarrollo de este trabajo se utilizará el paradigma de programación orientada a componentes y en particular en la forma de trabajo distribuida de dichos componentes. De esta manera se pretende la creación de componentes software COM y SOAP, que encapsulen la funcionalidad básica y sus tipos de datos de MATLAB mediante una interfaz orientada a objetos. Con esto, los componentes funcionarán de forma distribuida y heterogénea, de tal manera que se pueda reducir la complejidad del uso de dicha funcionalidad de Matlab y utilizarse desde cualquier entorno que sea posible.

Es importante mencionar que los componentes que se van a realizar a lo largo del presente trabajo de tesis forman la base de un gran proyecto, el cual tiene como objetivo general, formar una librería de componentes de un dominio específico de software educativo, en este caso el dominio de Matlab, y poder así, ofrecer un pequeño núcleo de componentes que cubran la funcionalidad básica requerida por aplicaciones en ese dominio. Este proyecto es representado por la arquitectura de software cliente-servidor (ver figura Figura 3.1), la cual, por obvias razones, también es la base para el desarrollo de cada uno de los módulos del proyecto. Por tal motivo, para el desarrollo de este proyecto, se está trabajando a la par con otros trabajos de tesis que se centran de igual forma, en el desarrollo de componentes que formarán en su totalidad la librería de componentes (módulo de redes de neuronas, operaciones matriciales y cálculo integral/diferencial).

---

[1] COM: Component Object Model, http://www.microsoft.com/com/default.mspx
SOAP: Simple Object Access Protocol, http://www.w3.org/TR/soap12-part1/

Cabe señalar, que alguno o todos los componentes que forman parte de la arquitectura antes mencionada, pueden ser reutilizados en alguna aplicación de un mismo o de diferentes dominios, pero eso no impide que puedan trabajar de manera individual, aunque en el mejor de los casos convendría tener los componentes a desarrollar en esta tesis con alguno(s) de los otros módulos en desarrollo. Para probar la funcionalidad de dichos componentes, además de las pruebas individuales sobre los componentes, se realizará una aplicación de software educativo mediante la cual podrán ser probados de manera minuciosa para corroborar el funcionamiento de los mismos.

Actualmente la tecnología orientada a componentes dispone de gran cantidad de recursos, desde entornos de desarrollo visual, como por ejemplo Borland Delphi [URL2] o Visual Basic [URL7] , hasta arquitecturas distribuidas como DCOM [29] , JavaBeans [40] o CORBA [25] .

Planteamiento y objetivos de la tesis

El objetivo principal de esta tesis es desarrollar componentes software heterogéneos bajo la tecnología COM-SOAP, que encapsulen la funcionalidad básica y los tipos de datos de Matlab, de tal forma que puedan ser utilizados por cualquier entorno de desarrollo que soporte esta tecnología.

Al mimo tiempo se plantean los siguientes objetivos específicos:

1. Investigar el estado del arte y de la práctica de la tecnología orientada a componentes, así como el estudio de la arquitectura de software y la importancia de la misma dentro de cualquier proyecto de desarrollo.
2. Implementar un componente COM orientado a objetos para encapsular la funcionalidad básica y sus tipos de datos de Matlab, de forma que se pueda, desde cualquier lenguaje de programación que soporte esta tecnología, manipular tales funcionalidades.
3. Desarrollar un componente SOAP para hacer que los servicios sean distribuidos y además heterogéneos. Esto una vez que se tenga desarrollado el componente COM.
4. Desarrollar una aplicación educativa para probar la utilidad de los componentes realizados, de tal manera que se vea de manera directa la utilización y funcionalidad de los mismos.

La hipótesis planteada es que, la complejidad y dificultad del desarrollo de aplicaciones de software que requieran de cálculos matemáticos puede verse favorecida si se utilizan componentes software que permitan realizar estos cálculos de forma más sencilla. Estos componentes pueden implementarse encapsulando la funcionalidad de un entorno de cálculo como Matlab.

Estructura de la tesis

Para ofrecer una visión general de lo que conlleva la realización de este tema de tesis, el trabajo se encuentra dividido en siete capítulos.

El capítulo uno proporciona una visión general de los conceptos previos manejados a lo largo del desarrollo de esta tesis, el contexto del trabajo, la importancia del mismo y las herramientas a utilizar.

En el capítulo dos se presenta un marco sistemático de las metodologías y tecnologías de la Ingeniería de Software a utilizar. Asimismo, se detalla a fondo el tipo de arquitectura, patrones de diseño y se profundiza en las tecnologías de componentes. De esta manera, se sentarán las bases para una clara comprensión de esta tesis.

En el capítulo tres se define la arquitectura, el análisis y diseño orientado a objetos para los componentes software COM y SOAP a realizar, los cuales marcarán la pauta a seguir para la implementación del proyecto.

El capítulo cuatro detalla el análisis y diseño orientado a objetos realizado para la aplicación educativa que reutilizará los componentes software a implementar en este trabajo.

El capítulo cinco contempla en su totalidad los procesos de implementación y pruebas de los componentes software desarrollados. Se muestra la forma en la que se llevó a cabo la implementación, así como las pruebas de caja negra.

En el capítulo seis, se presenta la implementación y pruebas de la aplicación educativa diseñada en el capítulo cuatro. Es aquí donde se muestran los resultados de manera visual mediante interfaces gráficas de usuario y pruebas de caja negra de la aplicación, para que el lector pueda observar detalladamente el producto final de este trabajo de tesis.

Las conclusiones de acuerdo a los resultados de la investigación realizada; el esbozo de algunas líneas de investigación futuras las cuales se orientan a la posible continuidad de este proyecto se presentan en el capítulo siete.

Por último se presentan las referencias bibliográficas utilizadas para el desarrollo de esta tesis, así como los anexos correspondientes.

1. Introducción

El continuo surgimiento de nuevas y mejores tecnologías hace que se requiera de nuevas y potentes técnicas para poder utilizar metodologías creadas anteriormente. Es por esto que, día a día, el ser humano se preocupa por desarrollar las herramientas necesarias que le permitan agilizar y mejorar el uso de las mismas. Un ejemplo claro de este suceso es lo que se conoce como Tecnología Orientada a Objetos, la cual es una potente técnica que permite la creación de software de calidad y bien organizado. La programación estructurada pasa a segundo plano a partir del nacimiento de la tecnología Orientada a Objetos, esto debido a la habilidad de crear nuevas y mejores herramientas con esta nueva tecnología y a la facilidad de implementación de software mediante su uso. La orientación a objetos influyó no sólo en los lenguajes y entornos de desarrollo, sino especialmente en las metodologías de análisis y diseño de software [12] [21] .

Es de esta manera como comienzan a aparecer nuevas tendencias que se enfocan en la reutilización, una de ellas es la programación orientada a componentes, la cual trabaja tanto en nuevos modelos de componentes como en extensiones para los ya existentes, así como en la estandarización de sus interfaces y servicios, y en la pertinaz búsqueda del cada vez más necesario mercado global de componentes software [44] .

Por esta razón, como se citó en el apartando anterior, en este trabajo de tesis se desarrollarán componentes de software que apoyen el tema de la reutilización, para un desarrollo rápido de aplicaciones de usuario final. En este caso se eligió a Matlab para encapsular su interfaz como componente software, por ser un entorno propietario poderoso en su dominio.

1.1. Conceptos previos

En este apartado se citan algunos conceptos que marcarán la pauta para la comprensión de los diferentes términos a lo largo del presente trabajo.

Se dice que un sistema es *abierto* si es concurrente, reactivo, independientemente extensible, y permite a componentes heterogéneos ingresar o abandonar el sistema de forma dinámica. Estas condiciones implican que los sistemas abiertos son inherentemente evolutivos y que la vida de sus componentes es más corta que la del propio sistema, condiciones que los sistemas cerrados no proveen y que los limita en la posibilidad de trabajar con ambientes distribuidos y heterogéneos [44] .

Se habla de la necesidad de hacer componentes de tipo abierto, por lo cual tenemos que dejar claro que la Programación Orientada a Objetos (POO) sirve como sustento del desarrollo de sistemas cerrados, por tal motivo nace lo que se conoce como *programación orientada a componentes* como una extensión natural de la orientación a objetos para los entornos abiertos [24] [41] .

Otro concepto importante a conocer es *sesión*, ya que a lo largo de este trabajo es una palabra muy común a encontrar, debido a que Matlab trabaja por sesiones. Una sesión es un espacio de tiempo ocupado por una actividad [URL3] .

Los conceptos antes mencionados aparecen muy frecuente a lo largo del documento, por lo cual fue necesario dar una idea general de los mismos.

1.2. El papel de MATLAB en el desarrollo del proyecto

Como se ha mencionado, en la realización de este trabajo se crearán componentes software que sean capaces de encapsular la funcionalidad básica y los tipos de datos de MATLAB, por tal motivo, es de suma importancia conocer la finalidad de esta tarea, y al mismo tiempo, justificar la elección del tema en curso.

Las herramientas matemáticas son fundamentales dentro de los diversos campos de las matemáticas aplicadas. Desafortunadamente, para pocos de estos campos se conoce software que permita el desarrollo de aplicaciones a la medida, ya que en su mayoría contienen ciertas reglas a seguir para solucionar estos problemas, sobre todo, una manera complicada de utilizar los mismos lenguajes. Por ello en la práctica cobra una extraordinaria importancia estudiar las diferentes herramientas matemáticas que día con día utilizamos, de tal manera que se pueda disminuir el tiempo y complejidad de ejecución de las mismas, a través del desarrollo de software a la medida que involucre el comportamiento de dichas herramientas.

Una de las herramientas informáticas más potentes para ello y más difundidas en la industria, es MATLAB, que permite ir directamente a la matemática del problema sin perder mucho tiempo en aspectos de programación [10] .

MATLAB es una herramienta de cálculo numérico orientada especialmente al ámbito de la ingeniería, que se ha convertido desde su aparición en un estándar “de facto” en esta disciplina. Además de su uso profesional, tiene una elevada presencia en el ámbito académico, colocándose como una herramienta ampliamente difundida a nivel educativo en materias y carreras relacionadas con disciplinas tecnológicas [42] . Además, MATLAB ofrece una interfaz de usuario claramente orientada al ámbito matemático-tecnológico y a un nivel de educación superior, que puede resultar excesivamente compleja o inadecuada en otros ámbitos.

Asimismo, Matlab es un entorno de computación y desarrollo de aplicaciones totalmente integrado y orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional [43] . En muchos casos se desearía disponer de la funcionalidad del software de forma independiente a su interfaz de usuario, lo que favorecería la reutilización de software [35] .

La capacidad de integrar este tipo de herramientas propietarias como componentes reutilizables en otras aplicaciones software posibilitaría aprovechar su amplia funcionalidad ocultando al mismo tiempo su complejidad. El no estar ligados a utilizar una interfaz de usuario predeterminada, abre además, la puerta al desarrollo de aplicaciones que integren diferentes componentes de software educativo, bajo una interfaz de usuario diseñada a la medida de necesidades y dominios concretos [31] [30] [32] .

De lo anterior, es de donde parte la idea de enfocarse en la herramienta MATLAB, de tal manera que su uso y funcionalidad se incremente y sea aprovechado por más personas, quienes en su miedo indirecto por enfrentarse a un mundo de ideas que no conciben, no son capaces de utilizarlas, y sin embargo, son necesarias para su desempeño profesional y académico.

1.3. Herramientas necesarias para el desarrollo del proyecto

En este apartado se detalla cada una de las herramientas utilizadas para el desarrollo del proyecto de tesis, esto con la intención de proporcionar un panorama claro de la funcionalidad, su contribución y beneficios.

En primer lugar, la Suite de Rational Rose, para el análisis y diseño del sistema. Esta herramienta se basa en el Lenguaje Unificado de Modelado (UML[2]).

Asimismo, la tecnología COM y SOAP, para la implementación de los respectivos componentes [URL3] [44] [URL9] [3].

Para el desarrollo de los componentes software se utilizará el lenguaje de programación visual orientado a objetos DELPHI en su versión 7.0. Delphi es un entorno muy completo, propiedad de Inprise-Borland, que se basa en el lenguaje Object Pascal. Completamente basado en los principios de Programación Orientada a Objetos (POO), con acceso a cualquier base de datos existente, con un compilador y una ejecución muy rápida [13] . Debido a que se implementará una arquitectura orientada a objetos, se ha optado por este lenguaje que tiene la capacidad de soportar dicha arquitectura y de soportar las tecnologías antes descritas, así como la facilidad para el manejo de la programación orientada a componentes, que es uno de los temas de base.

Otra herramienta indispensable para la realización de este proyecto es el conocimiento del uso de MATLAB, por la dependencia que tendrán los componentes de este entorno.

Por último el uso del Internet Information Server (IIS) para probar que realmente el componente funciona de manera distribuida.

[2] En el capítulo 2.1.3 se encuentra más información de este lenguaje
[3] Más información en el capítulo 2.5.1

2. Marco teórico

La ingeniería de software ha venido a proponer el desarrollo de software de una manera segura y confiable, dejando atrás el aventurarse al desarrollo de algo desconocido. Ahora, esta disciplina propone un análisis y diseño de los requerimientos, para detectar posibles riesgos a ocurrir antes y después del desarrollo de software. De esta manera se crea un panorama claro sobre la forma en la que debe responder dicho software y se prevén posibles fallas para lograr el éxito total a la terminación del producto, ya que hoy en día las personas e industria que requieren el uso de un software hecho a la medida, tienen todas sus esperanzas en los Ingenieros de Software, de tal manera que produzcan productos de gran calidad, que sean capaces de responder de forma estable y duradera.

A lo largo de este capítulo se mostrarán las principales prácticas de ingeniería de software como son la tecnología orientada a objetos y la tecnología orientada a componentes, pero sobre todo, la importancia que tiene esta ingeniería durante el proceso de desarrollo de software y la forma en la que se aplica dentro del desarrollo de este proyecto de tesis.

2.1. Tecnología Orientada a Objetos

La Tecnología Orientada a Objetos (TOO) ha crecido rápidamente en los últimos años, de tal manera que en estos días es una de las más importantes en el desarrollo de software. Esta técnica tiene un enorme potencial para incrementar significativamente la productividad del programador y facilitar el mantenimiento del código.

La TOO se basa en algunos modelos teóricos a partir de los cuáles se construye el software orientado a objetos. Booch94 [5] y Jacobson [20] proponen un modelo que se basa en los siguientes conceptos: abstracción, encapsulación, jerarquía, polimorfismo, modularidad, tipificación, concurrencia y persistencia. Según estos autores, si algún modelo que se dice orientado a objetos no contiene alguno de los cuatro primeros elementos entonces no es digno de llamarse orientado a objetos.

El proceso de desarrollo OO según Sommerville [38] consiste en el *análisis, diseño* y *programación orientados a objetos*. De la misma manera, según Booch [4] los beneficios que proporciona el enfoque orientado a objetos son:

- El uso del modelo OO nos ayuda a explotar el poder expresivo de todos los lenguajes de programación basados en objetos y los orientados a objetos, como Smalltalk, Object Pascal, C++, CLOS, Ada, y recientemente Java.
- El uso del modelo OO alienta el reuso no sólo del software, sino de diseños completos.
- Produce sistemas que están construidos en formas intermedias estables y por ello son más resistentes al cambio en especificaciones y tecnología.

Los beneficios ofrecidos por la tecnología OO nos ayudan a crear software bien organizado, capaz de responder a las necesidades de los usuarios. Además permite de manera eficaz la reutilización, de componentes de software, lleva a un desarrollo de software más rápido y a programas de mejor calidad [27] . La existencia de una metodología previa a la realización del software, permite conocer los requerimientos de dicho software, y con ello, tener un panorama previo de su comportamiento, tratando de evitar malos comportamientos o errores al momento de la terminación del mismo.

2.1.1. Programación orientada a objetos

Se debe distinguir que la Programación Orientada a Objetos (OOP, Object Oriented Programming) como paradigma (enfoque o manera de visualizar la realidad) y como metodología (colección de características para la ingeniería de software) no son la misma cosa. Sin embargo, la publicidad nos confunde asociando la POO más a una metodología, que al paradigma. De aquí, el interés en la POO radica más en los mecanismos que aporta para la construcción de programas que en aprovechar un esquema alterno para el modelado de procesos computacionales[19] .

Es aquí dónde el autor, Greiff, hace una notoria diferencia a través de los conceptos para cada punto de vista:

- La Programación Orientada a Objetos como paradigma, es una forma de pensar, una filosofía, de la cual surge una cultura nueva que incorpora técnicas y metodologías diferentes. Pero estas técnicas y metodologías, y la cultura misma, provienen del paradigma, no lo hacen. La POO como paradigma es una postura ontológica: el universo computacional está poblado por objetos, cada uno responsabilizándose por sí mismo, y comunicándose con los demás por medio de mensajes.
- La Programación Orientada a Objetos desde el punto de vista computacional es un método de implementación en el cuál los programas son organizados como grupos cooperativos de objetos, cada uno de los cuales representa una instancia de alguna clase, y estas clases, todas son miembros de una jerarquía de clases unidas vía relaciones de herencia.

En resumen, y tomando en cuenta ambas definiciones, se puede decir que la Programación Orientada a Objetos está basada en el mecanismo de clasificación y organización de objetos, misma que trata de solucionar problemas que los paradigmas de programación previos a este no han podido solucionar. Para resolver los problemas de la programación tradicional, el paradigma Orientado a Objetos se basa en la aplicación práctica de tres características [9] : objeto, clase y herencia.

Estos conceptos dan la pauta para poder conocer lo que el análisis y diseño orientados a objetos son en su totalidad, y al mismo tiempo, entender de una manera clara, la funcionalidad e importancia de este paradigma en el proceso de desarrollo de cualquier software.

2.1.2. Enfoque unificado para el AOO

El **Proceso Unificado** "es un proceso de desarrollo de software configurable que se adapta a proyectos variados en tamaños y complejidad. Fue creado a base de muchos años de experiencia, en el uso de la tecnología orientada a objetos, en el desarrollo de software de misión crítica, en una variedad de industrias por la compañía Rational", donde confluyen: Grady Booch, James Rumbaugh e Ivar Jacobson [23] .

El Proceso Unificado guía a los equipos de proyectos en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto. El proceso describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una pronta guía arquitectónica, para diseñar y probar el sistema de acuerdo a los requerimientos y a la arquitectura. El proceso describe qué entregables producir, cómo desarrollarlos y también provee patrones. Asimismo, este proceso es soportado por herramientas que automatizan entre otras cosas, el modelado visual, la administración de cambios y las pruebas.

Para Booch [4] las características primordiales del proceso unificado son: iterativo e incremental, centrado en la arquitectura, guiado por casos de uso y confrontación de riesgos.

Es a partir de este proceso unificado de dónde surge el Lenguaje Unificado de Modelado (UML) con la finalidad de cumplir con cada unas de las características antes mencionadas.

2.1.3. Lenguaje Unificado de Modelado

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson (OOSE) y Jim Rumbaugh (OMT). Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que se basaran la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle e IBM, así como grupos de analistas y desarrolladores [6] [27] .

UML es un elemento importante para el desarrollo de esta tesis, mediante su uso se implementa el análisis y diseño del software. Los modelos que forman parte de esta herramienta servirán para describir gráficamente cada una de las operaciones que involucra el proyecto, por tal motivo, más adelante se comprenderá el uso de los mismos y los beneficios aportados, pero mientras tanto, veremos brevemente dos aspectos importantes para el diseño de nuestro software: la arquitectura de software y los patrones de diseño.

2.2. Arquitectura de software

Dentro del desarrollo de cualquier tipo de sistema es necesario tener un análisis y diseño bien definido, pero sobre todo una correcta arquitectura de software. El término arquitectura de software ha sido definido de diferentes maneras, pero muchos autores coinciden con la siguiente definición: es una estructura compuesta de componentes, conectores, y reglas que definen como combinarlos, caracterizando la interacción de esos componentes [18] [22] [2] [26] .

Estas definiciones nos hacen notar el conjunto de partes que conforman una arquitectura de software y la importancia de los mismos, pero, ¿Cuál es la importancia de conocer o utilizar una arquitectura de Software? Según Bass [1] existen tres razones clave por las cuales la arquitectura de software es importante:

- Las representaciones de la arquitectura de software facilitan la comunicación entre todas las partes (partícipes) interesadas en el desarrollo de un sistema basado en computadora;
- La arquitectura destaca decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería de software que sigue, y es tan importante en el éxito final del sistema como una entidad operacional;
- La arquitectura “constituye un modelo relativamente pequeño e intelectualmente comprensible de cómo está estructurado el sistema y de cómo trabajan juntos sus componentes”.

Con esto podemos decir que la arquitectura de software trata de reforzar lo que es el análisis y diseño de software, por lo cual es de vital importancia poder definirlo claramente, además, nos permitirá reducir la complejidad del proceso de desarrollo del software.

El desarrollo de esta tesis se basa en una arquitectura de capas y posteriormente en una arquitectura orientada a objetos [18] , de tal manera que mediante estas dos arquitecturas podamos realizar un análisis y un diseño bien estructurado que se descomponga en objetos, los cuales puedan comunicarse e interactuar entre sí. De igual forma, el poder determinar de que manera se lleva a cabo dicha interacción, así como el comportamiento mismo de los componentes.

Un punto curioso a notar es que a pesar de que el objetivo principal de esta tesis sea el desarrollo de componentes no podemos enfocarnos únicamente en el uso de la tecnología orientada a componentes, si no también en la orientada a objetos, esto debido a que la TOC no se ha desarrollado en su totalidad y no existe un proceso de análisis y diseño bien establecido que nos asegure una buena arquitectura de software. Es decir, la TOC utiliza el análisis y diseño orientado a objetos para el desarrollo de proyectos de software.

2.3. Patrones de diseño

Todo problema está caracterizado por ciertos patrones a seguir, mismos que corresponden a una posible solución, y que al combinarse, pueden producir cierta solución a cualquier problema establecido. En la resolución de diversos tipos de problemas de software se han encontrado dichos patrones durante el proceso de diseño, motivo por el cual, han sido llamados

patrones diseño. Estos patrones hacen que el diseño sea más flexible, elegante y extremadamente reutilizable, motivos por los cuales hacen crecer las posibilidades de encontrar una solución de manera rápida.

Para describir un patrón de diseño debe hacerse de una manera detallada y con un formato consistente. Cada patrón es dividido en secciones de acuerdo a una plantilla [16] [15] . Esta plantilla proporciona una estructura uniforme a la información, de tal manera que el patrón de diseño es sencillo de entender, comparar y usar. Cada una de estas especificaciones es importante debido a que el principal motivo de los patrones de diseño es la reutilización, por lo que su existencia no tiene sentido si es que no son aplicables en este campo.

El patrón MVC, Modelo-Vista-Controlador, fue el utilizado en esta tesis. Este patrón suele ser utilizado para el desarrollo de software con interfaz hombre-máquina y en aplicaciones basadas en ventanas (Windows, XWindows) así como en aplicaciones Web.

El MVC surge de la necesidad de separar el modelo lógico del software de la presentación, sin que existan consecuencias para el núcleo del código de la aplicación al momento del cambio. Esto se logra dividiendo el sistema en tres partes: la primera, el *Modelo*, encapsula los datos y la funcionalidad de la aplicación. La segunda, la *Vista*, despliega la información contenida en el modelo, pueden existir varias vistas. La tercera, *Controlador*, está asociado a cada vista, contiene la información y los mecanismos necesarios para asociar una acción a cada evento [44] .

De esta manera, dentro de este trabajo de tesis, las librerías forman el modelo lógico, los componentes los elementos de la vista y, los botones y eventos los controladores.

En resumen, los patrones de diseño son un conjunto de información que nos aporta ciertas características necesarias para encontrar la solución de problemas, es por esto, que en el diseño orientado a objetos son de gran utilidad, ya que mediante su uso podemos encontrar soluciones que a simple vista no son obvias, y sobre todo, el poder utilizarlos en otros análisis y diseños futuros disminuyendo tiempo y aumentando la calidad de la solución del problema.

2.4. Tecnología orientada a componentes

La tecnología de componentes es considerada actualmente como la solución más prometedora de la ingeniería de software para acortar los tiempos de desarrollo y generar aplicaciones fiables ya que se basa en componentes creados y probados con anterioridad.

El desarrollo de componentes software se centra en la Ingeniería de Software Basada en Componentes (ISBC) [11] , de tal manera que ofrece las mismas ventajas que dicha ingeniería: reutilización, optimización en tiempo y recursos, y reducción de la complejidad de desarrollo. La ISBC es un proceso que se centra en el diseño y construcción de sistemas basados en computadora que utilizan *"componentes"* de software reutilizables [11] .

Sin embargo, es aquí donde surgen un sin número de cuestionamientos, es cuando los desarrolladores comienzan a preguntarse si es posible construir sistemas complejos a partir del ensamblaje de componentes de software ya construidos y reutilizables, o si es posible que dichos sistemas puedan reducir el tiempo de construcción, y que los desarrolladores adopten esta técnica como una cultura de desarrollo de manera que los incentive a reutilizar y no crear un software desde cero. Estos son puntos importantes que, junto con muchos otros, se han planteado los investigadores y desarrolladores sobre el uso de componentes reutilizables, pero al

fin y al cabo, el beneficio en tiempo y esfuerzo que pueda traer consigo la reutilización de componentes puede ser un factor determinante en la realización de sistemas complejos.

Es por lo anterior que surge la programación orientada a componentes como medio para afrontar la implementación de componentes reutilizables en la realización de sistemas.

2.4.1. Programación orientada a componentes

La Programación Orientada a Componentes (POC) aparece como una variante natural de la programación orientada a objetos (POO) para los sistemas abiertos, debido a que ésta última presenta algunas limitaciones, tales como: la POO no permite expresar claramente la distinción entre los aspectos computacionales y meramente composicionales de la aplicación, no define una unidad concreta de composición independiente de las aplicaciones (los objetos no lo son, claramente), y define interfaces de muy bajo nivel como para que sirvan de contratos entre las distintas partes que deseen reutilizar objetos [14] .

La POC nace con el objetivo de construir un mercado global de componentes software, cuyos usuarios son los propios desarrolladores de aplicaciones que necesitan reutilizar componentes ya hechos y probados para construir sus aplicaciones de forma más rápida y robusta, en caso de no existir un componente que cubra los requerimientos del sistema, entonces sí habrá que ver la manera de cubrir tales requerimientos. Las entidades básicas de la POC son los *componentes*, es decir, cajas negras que encapsulan cierta funcionalidad y que son diseñadas para formar parte de ese mercado global de componentes, sin saber ni quién los utilizará, ni cómo, ni cuándo. Los usuarios conocen acerca de los servicios que ofrecen los componentes a través de sus interfaces y requerimientos, pero normalmente no quieren ni pueden modificar su implementación [14] . En la POC la reutilización sigue siendo la base del desarrollo y se aprovechan al máximo las ventajas de la ISBC.

Es de esta manera como se han ido creando diferentes lenguajes de programación que sean capaces de satisfacer las necesidades de la programación orientada a componentes, sin embargo, es una técnica nueva, la cual se encuentra aún en desarrollo, motivo por el cual no cuenta con técnicas propias de análisis y diseño, más no por esto no las utiliza, ya que se basa en las técnicas orientadas a objetos para el desarrollo del software.

2.4.2. CORBA

CORBA (*Common Object Request Broken Architecture*) es una norma, no un producto, que se encarga de especificar la interoperabilidad y portabilidad entre objetos en un entorno distribuido heterogéneo de manera transparente. CORBA automatiza muchos problemas en la programación distribuida como el registro de objetos, localización y activación, envío y recepción de parámetros, entre otros.

CORBA fue impulsado por la OMG (*Object Management Group*) en 1990. Todas las especificaciones elaboradas por el OMG tienen en común una infraestructura conceptual denominada OMA (*Object Management Architecture*) que define en un alto nivel, las facilidades para hacer posible la programación distribuida orientada a objetos heterogéneos [44] [37] .

2.4.3. JavaBeans

La tecnología JavaBeans es la arquitectura de componentes definida para la plataforma Java 2 (J2EE). Los componentes son denominados *JavaBeans* y se encuentran definidos como programas de Software reutilizables que uno mismo puede desarrollar y ensamblar de una manera fácil para poder crear sofisticadas aplicaciones [URL6] .

Las principales características de los *Beans* son [44] :

- La interfaz de cualquier bean está compuesta por una serie de atributos, una serie de métodos y una serie de eventos.
- Todo bean soporta inspección.
- Todo bean soporta particularización.
- Los beans soportan persistencia.

Estas son algunas de las tecnologías que resuelven problemas relacionados con los aquí tratados, es importante conocer algo de ellos para tener una idea clara sobre la manera en que podrían implementarse en otros lenguajes con otras tecnologías.

2.5. Tecnología utilizada

La selección del uso de la tecnología a utilizar en el desarrollo de este proyecto se obtiene después de un análisis minucioso de los objetivos a cubrir, de tal manera, que al querer desarrollar componentes capaces de ser utilizados por diferentes lenguajes de programación y que se encuentren estandarizados en la elaboración de los mismos, se tiene que buscar la tecnología que se adapte a estas necesidades. Por tal motivo, se encuentra que las tecnologías COM y SOAP son implementadas por un sin número de lenguajes de programación actuales, es así, como se decide sea parte de este trabajo de tesis.

2.5.1. Tecnología COM

La tecnología COM (Component Object Model) es una arquitectura e infraestructura para construir software basado en componentes, software rápido, robusto y expandible. Es una tecnología creada por la familia Microsoft-Windows para permitir la comunicación entre componentes de software. También autores conocen a COM como la especificación del modelo de Microsoft que define cómo crear dichos componentes y cómo construir aplicaciones con ellos [29] [7] .

COM es usado por desarrolladores para crear componentes de software reusables, interconectar componentes para construir aplicaciones y obtener beneficio de los servicios de Windows. La familia de tecnologías COM incluye COM+, COM Distribuido (DCOM) y controles ActiveX [URL3] .

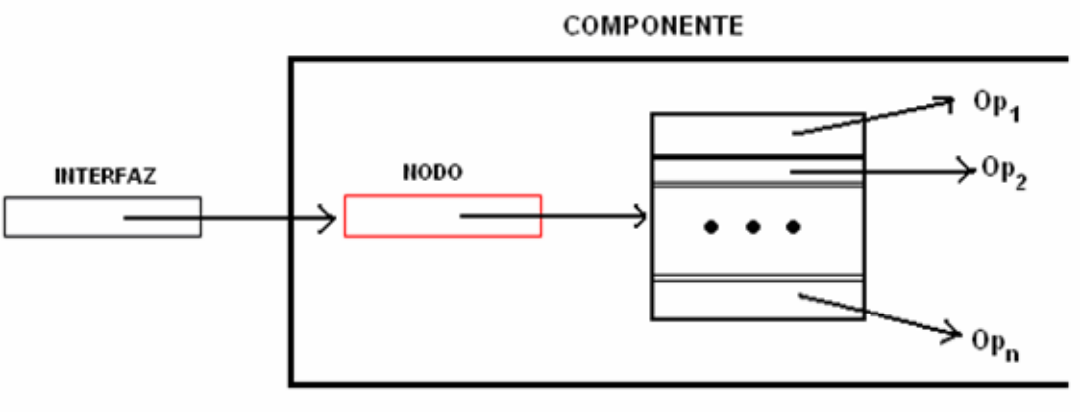


Figura 2.1. Representación binaria de una interfaz COM [4]

Frente a otras especificaciones, COM establece un estándar binario de interoperabilidad entre componentes, y por tanto, independiente de los lenguajes y plataformas en los que se diseñen e implementen. COM ni siquiera define con precisión lo que entiende por componente, sino que se apoya en el concepto de interfaz.

A nivel conceptual, una interfaz COM es un conjunto de funciones que un componente implementa, mientras que a nivel binario, una interfaz COM es un puntero a una estructura en memoria, compuesta por un puntero (denominado Nodo) a un vector de punteros a funciones (Figura 2.1 Ver [29]).

Las interfaces permiten a los componentes comunicarse de forma independiente con componentes concretos que implementen las funciones de una interfaz. De esta forma, si dos o más componentes implementan la misma interfaz, sus clientes pueden utilizar el mismo código para comunicarse con ellos. Así es como COM consigue el polimorfismo.

Con estas características, la tecnología COM hoy en día es muy utilizada. Además, está disponible en diversos lenguajes de programación (C++, VisualBasic, J++, Delphi, etc.) lo que hace posible su soporte dentro de los mismos, pero sobre todo, también es soportada por Matlab. Su enorme uso y ventajas hace que se convierta en una tecnología poderosa para la creación de componentes de software estandarizados que puedan cambiar de interfaz fácilmente sin necesidad de cambiar la funcionalidad interna de los mismos, estas son algunas de las razones por las que se decidió el uso de esta tecnología.

2.5.2. Tecnología SOAP

Esta tecnología SOAP (Simple Object Access Protocol), surgió para hacer frente al problema de crear componentes distribuidos y heterogéneos como es el caso de la tecnología COM.

El protocolo SOAP según W3C en un documento de trabajo [URL9] nos dice que, es un protocolo ligero para el intercambio de información en entornos descentralizados y distribuidos, basado en XML, cuya definición establece básicamente que:

[4] Imagen Reeditada de la referencia [29]

- Todo mensaje debe ir contenido en un sobre o envoltura (*envelope*) que define un marco para describir lo que hay en un mensaje, qué estructura debe tener y cómo procesarlo;
- hay una serie de reglas de codificación para describir instancias de tipos de datos definidos por las aplicaciones, y
- hay una convención para representar llamadas a procedimientos remotos y las respuestas correspondientes.

SOAP es un protocolo elaborado para facilitar la llamada remota de funciones a través de Internet, permitiendo que dos programas se comuniquen de una manera muy similar a la invocación de páginas Web. El protocolo SOAP tiene diversas ventajas sobre otras maneras de llamar funciones de manera remota como DCOM, CORBA o directamente en TCP/IP [URL5] :

- Es sencillo de implementar, probar y usar.
- Es un estándar de la industria, creado por un consorcio del cual Microsoft forma parte, adoptado por W3C[5] y por varias otras empresas.
- Utiliza los mismos estándares de la Web para casi todo: la comunicación se hace mediante HTTP con paquetes virtualmente idénticos; los protocolos de autenticación y encriptación son los mismos; el mantenimiento de estado se hace de la misma forma; se implementa normalmente por el propio servidor Web.
- Atraviesa "firewalls" y routers, considerando que es una comunicación HTTP.
- Tanto los datos como las funciones se describen en XML, lo que permite que el protocolo no sólo sea más fácil de utilizar sino que también sea muy sólido.
- Es independiente del sistema operativo y procesador.
- Se puede utilizar tanto de forma anónima como con autenticación (nombre/clave).

Estas son algunas de las grandes ventajas que SOAP ofrece, además de que trabaja muy bien con la tecnología COM, motivo por el cual se decide implementarlo. Ahora bien, después de analizar la tecnología utilizada, se puede concluir que COM y SOAP son un estándar mediante el cual, los componentes van a poder integrarse a un sin número de lenguajes de programación, teniendo una flexibilidad considerablemente alta.

Después de poner las bases teóricas, en el siguiente capítulo, se muestra el análisis y el diseño propio de los componentes, los cuales son la piedra angular del desarrollo de los mismos.

[5] http://www.w3.org/TR/SOAP

3. Análisis y Diseño de los Componentes de Software

En todo proyecto de desarrollo de software es deseable contar como base con un concreto y funcional análisis y diseño, de aquí, que al momento de implementar de manera práctica la realización del producto se tengan los resultados esperados y arrojados por dicho análisis y diseño. Por tal motivo, en este proyecto de tesis, como en cualquier otro, se optó por realizar un análisis minucioso para la realización del producto. A continuación se muestra la manera en la que se llevaron a cabo estos puntos a lo largo del desarrollo de los componentes de software y la utilidad dentro del proyecto.

3.1. Análisis de los Componentes de software

Dentro de este apartado se presentan dos documentos para el análisis de los requerimientos: En el punto 3.1.1 el Documento de Requerimientos de Usuario (DRU) y en el punto 3.1.2 la Especificación de Requerimientos del Software (ERS), todo esto siguiendo las directrices dadas por el estándar “IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830 1998”.

3.1.1. Documento de Requerimientos del Usuario (DRU)

1. Introducción

Este documento de requerimientos tiene como finalidad presentar los resultados recavados previamente sobre los requerimientos necesarios para la construcción de los componentes de software.

2. Componentes COM-SOAP para encapsular variables y tipos de datos de Matlab.

Se desarrollarán dos componentes de software bajo las tecnologías COM y SOAP. El componente COM funcionará de manera local, y el componente SOAP será el encargado de trabajar de manera remota dentro de este proyecto. Ambos componentes tienen la finalidad de soportar y encapsular las variables y tipos de datos que usa Matlab, de tal manera que se puedan realizar operaciones similares sin usar directamente dicho lenguaje. Asimismo, servirán como base para la formación de una librería de componentes, los cuales podrán ser agregados

e implementados siempre y cuando cumplan con la tecnología de desarrollo que se utiliza para el desarrollo de los mismos.

2.1. Requerimientos

Mediante el desarrollo de estos componentes de software se pretende cubrir las necesidades de solucionar problemas que se presentan por muchas aplicaciones software al tratar de resolver problemas matemáticos que requieran tipos de datos y variables complejos o que sean de mayor dificultad para su solución. De tal manera que, mediante el uso de los componentes se facilite la solución de un gran número de estos problemas. Para lo anterior se pretende brindar un software que sea capaz de cubrir los siguientes aspectos:

1. Desarrollar dos componentes de software, local y remoto, mediante la tecnología COM-SOAP respectivamente, con la finalidad de que sean capaces de abarcar un rango mayor en cuanto a su posibilidad de uso. La tecnología a usar hace también posible el poder ser adaptados a cualquier lenguaje de programación que soporte dicha tecnología, por tal motivo, se convierten en herramientas poderosas debido a su capacidad de utilización.
2. Ambos componentes serán capaces de encapsular todos los tipos de datos y variables usados por Matlab. De tal manera que soporten operaciones complejas que requieran del uso de dichas variables y tipos de datos.
3. Los componentes serán el motor para la formación de una librería de componentes debido a que el resto de componentes (con fines comunes bajo la misma tecnología de desarrollo) tendrán que pasar por estos para acceder a Matlab.
4. Además de esto, los componentes soportarán otros tipos de datos y variables que se creen desde alguna aplicación hacia el entorno de Matlab, haciendo posible que el desarrollador interactúe de manera directa con los componentes y Matlab al momento de crear y manipular sus propias variables.
5. Los componentes soportarán todos los posibles valores que Matlab sea capaz de soportar, de tal manera que se puedan realizar un sin fin de operaciones con dichos valores y la manipulación exhaustiva de los mismos, lo que hace componentes funcionales, y sobre todo robustos.
6. Implementar el manejo de excepciones de tal manera que los componentes se comporten de manera estable, lo que los hace seguros al momento de su uso.

3.1.2. Especificación de Requerimientos de Software (ERS)

Introducción.

Durante el desarrollo de un proyecto de Software es necesario que dicho proyecto cumpla con ciertas especificaciones. Para esto, los componentes COM y SOAP deben cumplir con todas las especificaciones que en este apartado se detallan.

1. Propósito.

La ERS tiene como propósito fundamental delimitar de manera clara y precisa el dominio total y parcial de los componentes COM y SOAP a desarrollar, así como los alcances y limitaciones de los mismos. La ERS también servirá de gran ayuda tanto para el desarrollador en la realización del diseño de dichos componentes, como para los revisores de esta tesis en la comprobación de los requisitos.

2. Ámbito del Sistema.

A lo largo del documento de tesis se ha venido mencionando la finalidad de la creación de los componentes, misma que tiene como razón fundamental que al momento de crear aplicaciones que requieran el uso de operaciones complejas y de cálculos avanzados, puedan ser resueltos por los mismos lenguajes de propósito general, reutilizando componentes software, sin tener que reimplementar algoritmos o usar lenguajes matemáticos como Matlab, desarrollados exclusivamente para la resolución de éste tipo de problemas.

Los componentes pueden ser reutilizados en el desarrollo de aplicaciones tanto para el ámbito educativo de nivel medio y superior como en la investigación. En el ámbito educativo principalmente apuntaría hacia las ingenierías o licenciatura en matemáticas aplicadas, donde constantemente hacen uso de cálculos complejos, mismos que requieren de poderosos lenguajes matemáticos para su solución, que en la mayoría de las veces, no se cuenta con ellos o no se tiene un conocimiento pleno de los mismos. Es aquí en donde los componentes serían de gran ayuda, además de poder ser soportados por diversos lenguajes de programación adquiridos a lo largo del paso por la Universidad. En cuanto a la investigación se refiere, los componentes serían de gran utilidad, no sólo dentro de la Universidad si no fuera de ella más aún, ya que se encuentran abiertas líneas de investigación que requieren de su uso como base para la solución de otros problemas, ejemplo de esto es la Universidad de Vigo en España, en donde este tema es motivo de investigación a gran escala y se requiere la creación de estos componentes para su uso inmediato.

El componente SOAP juega un papel muy importante dentro y fuera de la Universidad, esto debido a que en cualquier institución que cuente con una infraestructura poderosa mediante la cual las personas que laboran en la misma puedan hacer uso del componente a través de la red. Es así, como al momento de instalar el componente SOAP y Matlab en una plataforma Windows, los usuarios podrán acceder a todos los recursos desde su lugar de ubicación de una manera rápida y sencilla.

De igual manera, un punto clave del desarrollo de estos componentes, es el poder soportar la integración de otros componentes, los cuales realicen funciones específicas y que formen una librería de componentes más poderosa para la resolución de problemas matemáticos. Como se citó en el prólogo, existen trabajos a la par en el desarrollo de otros componentes como: operaciones matriciales, redes de neuronas, cálculo integral y diferencial, ecuaciones diferenciales, entre otros, los cuales formarán una librería de componentes de este dominio.

3. Definiciones, Acrónimos y Abreviaturas

3.1. Definiciones

Tabla 3.1. Definiciones

Usuarios	Personas que tienen la necesidad de utilizar los componentes COM y/o SOAP para el desarrollo de nuevas aplicaciones software que requieran el uso de los mismos para la solución de diferentes problemas de tipo matemático.

3.2. Acrónimos

Tabla 3.2. Acrónimos

ERS	Especificación de Requerimientos de Software
DRU	Documento de Requerimientos de Usuario

3.3. Abreviaturas

Tabla 3.3. Abreviaturas

COM	*Component Object Model* modelo de Microsoft que define cómo crear dichos componentes y cómo construir aplicaciones con ellos.
SOAP	*Simple Object Access Protocol* es un protocolo que se utiliza para el intercambio de mensajes en sistemas de cómputo distribuidos.
MatLab	*Matrix Laboratory*, es una herramienta de cálculo numérico orientado al ámbito de la ingeniería e investigación. Creado por la empresa *The MathWorks* y usado en diferentes Universidades e institutos de investigación alrededor del mundo.
UTM	*Universidad Tecnológica de la Mixteca*

4. Referencias del Documento ERS

- IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE std. 830, 1998
- Sommerville, Ian. Ingeniería de software. 6a. edición. Addison Wesley

Descripción General

En este apartado se presenta una descripción de alto nivel de los componentes de software, las funciones principales a realizar, la información utilizada, así como las limitaciones y restricciones que afecten el desarrollo de los mismos.

5. Funciones de los Componentes de Software

Los componentes cumplirán con las siguientes funciones:

- Métodos para el manejo de sesiones de Matlab, así como soporte de diferentes tipos de datos, manipulación de comandos de texto, recuperación y manipulación de variables.
- Métodos para la lectura de valores de las variables.
- Métodos para la escritura de valores en variables.

Enseguida se describe de forma detallada las tres funciones.

5.1. Métodos para el manejo de sesiones, ejecución de comandos y soporte para el uso de diferentes tipos de datos.

En esta función, intervienen métodos necesarios para la comunicación con Matlab; entre ellas: apertura de sesiones y manejo de las mismas, ejecución de comandos y soporte de variables y tipos de datos de Matlab, lo cual es parte esencial del desarrollo del proyecto. Las operaciones son las siguientes:

- Borrar todas las variables creadas en el entorno de Matlab.
- Borrar alguna variable, indicando el nombre de la misma, creada dentro del entorno de Matlab.
- Cerrar sesiones de Matlab.
- Cerrar ventanas gráficas de Matlab.
- Convertir arreglos bidimensionales a multidimensionales.
- Convertir el ID de una clase al nombre de la misma.
- Convertir el nombre de una clase al ID de la misma.
- Convertir un arreglo numérico a cadena de texto.
- Convertir un arreglo bidimensional plano en un arreglo de tipo N-dimensional indicando el arreglo y las dimensiones requeridas para el mismo.
- Crear objetos de nuevos tipos para que puedan soportar más operaciones y métodos.
- Ejecutar comandos con formato de texto en el motor de Matlab.
- Ejecutar comandos con formato de texto en el motor de Matlab, devolviendo una excepción en caso de que Matlab devuelva un error.
- Copiar el contenido de una ventana gráfica de la sesión actual de Matlab al portapapeles de Windows.
- Recuperar el valor que tenga la variable temporal en la sesión de Matlab como un valor numérico doble, entero o cadena (métodos independientes para cada tipo de objeto).
- Recuperar el valor de una variable definida actualmente en la sesión de Matlab como un arreglo de caracteres, valor numérico doble, arreglo de valores numéricos dobles (parte real e imaginaria), arreglo plano bidimensional de valores dobles (parte real e imaginaria), entero, cadena. Para cada operación existen métodos independientes.
- Cargar una sesión (entorno de trabajo) en el entorno actual de Matlab.

- Transformar un valor de tipo Mat_FigureFormat, definido dentro de ésta librería COM, en una cadena de texto.
- Transformar un valor numérico real doble en una cadena de texto teniendo en cuenta el separador de decimales que usa Matlab.
- Abrir una sesión remota de tal manera que se pueda trabajar de manera distribuida mediante el uso de un equipo servidor y otros clientes.
- Abrir una sesión de Matlab la cual será encapsulada por este objeto.
- Generar excepciones cuando ocurra un error dentro del entorno de trabajo de Matlab.
- Almacenar en disco una figura indicada por el usuario.
- Guardar el espacio de trabajo o entorno de Matlab actual en un fichero.
- Ver si una sesión está actualmente activa o no.
- Hacer activa una ventana gráfica deseada en la sesión actual de Matlab.
- Crear una ventana gráfica nueva en la sesión actual de Matlab.
- Liberar el token que se encuentra siendo utilizado en la sesión actual de Matlab.
- Manipular una variable de Matlab como un tipo concreto de variable, como tipo arreglo de celdas, arreglo de caracteres, matriz de valores numéricos dobles (parte real e imaginaria), objeto, cadena de caracteres o arreglo de estructuras. Cada uno de estos tipos son soportados por métodos diferentes para poder ser manipulados.
- Cambiar el nombre de una cierta variable definida en Matlab por otro nombre más específico deseado por el usuario.
- Copiar el contenido de una variable a otra.
- Hacer una copia de una variable.
- Importar una variable que se encuentra dada de alta en una sesión de Matlab hacia otra aplicación externa.
- Remover un campo dado por el usuario, de algún registro dentro de una matriz de registros.
- Eliminar el argumento contenido en una posición dada por el usuario, dentro de una función de la cual se hizo un llamado a Matlab para su uso.
- Eliminar todos y cada uno de los argumentos contenidos en una función de la cual se hizo un llamado a Matlab para su uso.
- Ejecutar la llamada a la función de Matlab deseada.
- Cerrar la ventana de una figura abierta en la sesión actual de Matlab.
- Copiar al portapapeles de Windows una figura indicada por el usuario.
- Crear una figura dentro de una sesión de Matlab.
- Importar una figura que se encuentre en una ventana determinada dentro de una sesión de Matlab a una aplicación externa.
- Almacenar en disco una figura indicada por el usuario.

Estos son los métodos que forman la base de los componentes, de tal manera que su importancia radica en soportar cada una de las operaciones, tipos de datos y variables tal y

como lo hace Matlab, así como establecer la comunicación, manejo de sesiones y comandos con dicho lenguaje.

5.2. Métodos para la lectura de valores de las variables.

En este apartado se presentan los métodos que hacen posible la lectura de una variable. Mediante ellos es posible verificar el contenido de las variables que se han creado, así como comprobar si los datos son correctos.

- Verificar la existencia de figuras o ventanas gráficas en la sesión actual de Matlab.
- Ver las figuras o ventanas gráficas que se encuentran activas en la sesión actual de Matlab.
- Obtener el ID de clase de una variable.
- Obtener la clase de una variable indicada por el usuario.
- Verificar si al momento de cerrar la sesión de Matlab se limpia o no el entorno de trabajo antes de realizar la operación.
- Ver cuál es la cadena que se está utilizando como separador de decimales dentro de Matlab.
- Obtener de manera detallada las características, de cada una de las variables que se encuentran en el entorno actual de Matlab.
- Obtener la cadena que devuelve Matlab al momento de ocurrir un error tras la ejecución de un comando.
- Obtener una lista de las figuras creadas dentro del entorno de Matlab en forma de arreglo, el cual contiene el identificador de las figuras.
- Obtener el valor que Matlab considera como infinito.
- Obtener de manera detallada las características de todas y cada una de las ToolBoxes que se encuentran instaladas en Matlab.
- Poder ver si el último comando ejecutado ha ocasionado un error o no.
- Obtener el último mensaje de respuesta generado por Matlab tras la ejecución de un comando.
- Obtener el texto completo de la respuesta dada por Matlab en la última ejecución de un comando.
- Obtener el último mensaje de texto devuelto por Matlab al momento de la ejecución de un comando antes de una excepción.
- Obtener la cadena de texto que contiene el mensaje de advertencia devuelto por Matlab tras la ejecución de un comando.
- Ver la referencia que se encarga de establecer comunicación entre la interfaz COM y el entorno Matlab.
- Obtener la versión, el número de licencia, y otra información que hace referencia al entorno de Matlab con el cual se está trabajando actualmente.
- Obtener un valor el cual es considerado como no numérico por el entorno de Matlab.
- Ver el nombre del servicio que se está comunicando con el servicio SOAP remoto que permite manejar los diferentes tipos de objetos, cabe mencionar que existe un método para verificar este punto para cada tipo de objeto creado.

- Ver la URL raíz en la cual se encuentran situados los servicios SOAP a los que se tendrá acceso.
- Ver la cadena usada como delimitador de cadenas dentro del entorno de Matlab.
- Obtener el texto del último mensaje de error que haya sido creado por Matlab tras la ejecución de un comando.
- Obtener el nombre de la variable temporal creada en el entorno de Matlab.
- Obtener en forma de vector las dimensiones de una variable indicada.
- Obtener la lista de variables que se encuentran actualmente definidas dentro del entorno de Matlab en forma delimitada y entrecomillada.
- Ver si la ventana de la sesión de Matlab es o no visible en el entorno de trabajo de Windows.
- Obtener la cadena de texto, la cual se usa para distinguir de modo interno si Matlab ha devuelto un mensaje de Advertencia.
- Ver si un valor de entrada dado es finito.
- Ver si un valor de entrada dado es infinito.
- Ver si la sesión de entorno de trabajo actual de Matlab es local o no.
- Ver si un valor de entrada es de tipo no numérico.
- Ver si el último comando que se ha ejecutado en el entorno actual de Matlab ha devuelto un mensaje de advertencia.
- Obtener una cadena con un valor generado de manera aleatoria, dicha cadena es utilizada como el nombre de la variable generada en Matlab.
- Ver si al momento de eliminar un objeto se habrá de eliminar también el elemento de Matlab que se encuentra encapsulado por dicho objeto.
- Ver el identificador del objeto SOAP mediante el cual se podrá establecer una comunicación con un servicio SOAP remoto establecido.
- Ver el nombre del servicio SOAP que se utiliza para comunicarse con el SOAP remoto.
- Ver si una variable de entrada dada existe o no en el entorno de trabajo actual de Matlab.
- Ver el ID de clase que identifica a una variable.
- Ver el nombre de la clase que está siendo utilizado por la variable actual en la sesión actual de Matlab.
- Ver el tamaño del vector de una dimensión deseada en una variable.
- Devolver las dimensiones de una variable de Matlab.
- Devolver el contador de dimensiones de una variable de Matlab como un número entero.
- Obtener el índice lineal correspondiente a la posición de una matriz expresada por n coordenadas.
- Obtener la longitud (o número de elementos) de una variable de Matlab.
- Obtener la sesión de Matlab asignada a la variable sobre la que se está haciendo la manipulación de las diferentes variables utilizadas.

- Obtener el código de tipo que identifica a una variable.
- Obtener el nombre de alguna variable.
- Obtener el valor que se encuentra en una cierta posición dentro de un plano bidimensional X, Y de una variable.
- Obtener el valor que se encuentra en una cierta posición dentro de un plano tridimensional X, Y, Z de una variable.
- Verificar si una variable encapsulada dentro de un objeto, es de una cierto tipo de clase, misma que es indicada por el programador.
- Ver si alguna variable creada dentro del entorno de Matlab es de tipo arreglo de cerdas, celda de una matriz de celdas, matriz de caracteres, matriz de registros, arreglo numérico, objeto, valor escalar, cadena de caracteres o arreglo de estructuras. Para cada tipo de objeto existe un método específico para realizar la función indicada.
- Verificar si una variable de Matlab es una propiedad de un objeto en Matlab.
- Ver la existencia de un arreglo de valores numéricos dobles del tipo del objeto que encapsula este tipo de dato.
- Obtener un valor de tipo complejo (parte real e imaginaria) perteneciente a un arreglo de valores dobles de una posición deseada por el usuario.
- Obtener un valor de tipo real perteneciente a un arreglo de valores dobles de una posición deseada por el usuario.
- Ver la existencia de un arreglo serializado de valores numéricos del tipo del objeto que encapsula este tipo de dato.
- Ver si alguna variable del tipo del objeto contiene o no algún valor en su parte imaginaria.
- Ver/Leer los valores que tiene asignados cierta variable de tipo cadena de caracteres.
- Obtener el contenido de un arreglo de celdas que contengan sólo arreglos numéricos simples como un vector de arreglos.
- Obtener el contenido de un arreglo de celdas que contengan sólo valores numéricos simples como un único vector.
- Obtener un vector plano que contiene información sobre las dimensiones de cada arreglo contenido en cada celda, además de los valores que contenga cada arreglo.
- Obtener un arreglo de celdas que contengan sólo cadenas (string).
- Obtener una variable que encapsula a la celda con la posición indicada por el índice, es decir, crea un apuntador al contenido de la celda encapsulada en un objeto.
- Ver/leer un valor perteneciente a una matriz de celdas (cellarray) de valores de cualquier tipo, de una posición deseada por el usuario.
- Obtener cada una de las características correspondientes a una celda de un arreglo de celdas.
- Ver el contenido de alguna celda, indicada por su posición dentro del arreglo de celdas, como un valor de tipo numérico.
- Obtener el contenido de una celda indicada por el usuario como una cadena de caracteres.

- Obtener un arreglo de matrices numéricas el cual contiene todos los valores contenidos en el campo fieldName de tipo matrices numéricas.
- Obtener el contenido del arreglo fieldName recuperado como un vector plano que contiene información sobre las dimensiones de cada arreglo contenido en cada campo, además de los valores que contenga cada arreglo.
- Recuperar el contenido de una matriz de registros que contenga sólo valores numéricos simples como un único vector.
- Obtener todos los valores contenidos en el campo fieldName de todos los registros del arreglo, que se supone son de tipo cadena, y los devuelve como un único cadena, con los valores de cada campo delimitados.
- Obtener cada una de las características correspondientes a un campo de una matriz de registros.
- Obtener una variable que encapsula a un campo de un arreglo de registros con la posición indicada por el índice.
- Obtener el nombre de todos los campos que se encuentran contenidos en un registro.
- Obtener el nombre de un campo de un registro dado por el usuario.
- Obtener el número de campos (contador acumulado) con los que cuenta un registro.
- Obtener un valor contenido en un campo especificado por el usuario, en una posición dada por el mismo.
- Obtener el contenido de algún campo, indicado por su posición dentro de la matriz de registros, como un valor de tipo numérico.
- Obtener el contenido de un campo indicado por el usuario, en una posición dada por el mismo, como una cadena de caracteres.
- Verificar si un nombre dado por el usuario es o no, un campo de algún registro dentro de una matriz de registros.
- Obtener el argumento de una función que se encuentra contenido en una posición dada por el usuario.
- Obtener la cadena que se va a mandar a ejecutar en Matlab, que representa la función que con todos los argumentos.
- Obtener el contenido del ID de la figura o ventana gráfica en la que se muestra el resultado de la función ejecutada, siempre y cuando la función a ejecutar requiera de una ventana gráfica al momento de arrojar sus resultados tras la ejecución.
- Obtener el nombre de la función que se ha establecido para ser ejecutada.
- Obtener el número total de argumentos contenidos en una función.
- Obtener el valor de la ejecución del llamado a la función de Matlab, es decir, el resultado de la ejecución de dicha función en un tipo de objeto específico.
- Obtener el resultado de la ejecución del llamado de la función a Matlab como una cadena de caracteres.
- Obtener la sesión de Matlab asignada a la variable que encapsule el objeto actual, es decir, sobre la que se está haciendo la manipulación de las diferentes variables utilizadas.

- Obtener el ID de la ventana en la cual se encuentra la figura que se está manipulando, o en la que se desea crear alguna nueva figura dentro de una sesión de Matlab.
- Obtener la sesión de Matlab asignada a la variable sobre la que se está haciendo la manipulación de las diferentes variables utilizadas.
- Leer la propiedad “Visible” para poder ver si la figura de la sesión de Matlab es o no visible en el entorno de trabajo de Windows.
- Obtener cada una de las características correspondientes a una propiedad de un objeto.
- Obtener el contenido de alguna propiedad, indicado por el nombre de la propiedad, como un valor de tipo numérico.
- Obtener un contador de las propiedades con las que cuenta un objeto.
- Obtener el nombre de todas las propiedades que se encuentran contenidos en un objeto en forma de cadena de caracteres.
- Obtener la variable que encapsula a la propiedad de un objeto indicada por su nombre.
- Obtener el nombre de la propiedad de un objeto identificado mediante el número de la propiedad.
- Ver/leer un valor contenido en la propiedad de un objeto especificado por el usuario mediante el nombre de la misma.
- Obtener el contenido de alguna propiedad, indicando el nombre de la propiedad, como un valor de tipo cadena de caracteres.
- Verificar si una cadena de caracteres dada por el usuario e identificada por su nombre es una propiedad de un objeto de alguna clase definida en una toolbox.
- Ver si existe o no una matriz de caracteres dentro del entorno de Matlab.
- Obtener un caracter establecido en una posición determinada dentro de una matriz de caracteres establecida dentro del entorno de Matlab.
- Obtener el contenido de un renglón indicado por el usuario dentro de una matriz de caracteres.
- Obtener el caracter que se está usando como separador de decimales al momento de pasar un valor numérico a Matlab.
- Obtener la cadena que está siendo actualmente utilizada por el componente para saber si Matlab ha devuelto un error tras la ejecución de un comando.
- Obtener el caracter que se está usando como separador de cadenas al momento de pasar un valor string a Matlab.
- Ver la cadena de texto que está siendo actualmente utilizada por el componente, para saber si Matlab ha devuelto una advertencia, tras la ejecución de un comando.
- Ver o conocer el nombre del servicio que se está utilizando por defecto al momento de tratar de conectarse a un servicio remoto mediante SOAP, esto es, a través de las interfaces propias de cada servicio que se maneja dentro del componente (existen métodos individuales para verificar ésta función para cada tipo de servicio de los que se manejan en el componente).
- Obtener la URL base del componente, la cual está siendo usada por el mismo para poder acceder a los servicios remotos mediante SOAP, esto es, mediante las interfaces propias de cada servicio.

- Ver el código de Windows del lenguaje que se encuentra actualmente seleccionado en el componente.
- Ver o mostrar una AboutBox (caja acerca de) para ver las características del componente.

Cada una de las operaciones aquí citadas permite hacer una lectura de algún tipo de dato, de tal manera que podamos estar seguros de lo que se está realizando o se va a realizar con dichas variables.

5.3. Métodos para la escritura de valores en las variables.

En esta sección se encuentran los métodos que hacen posible la escritura de variables al momento de su creación o modificación de su valor después de ser creadas.

- Escribir en la propiedad ClearOnClose para poder establecer si al cerrar la sesión debe limpiarse o no el entorno de trabajo actual de Matlab.
- Modificar la cadena usada como separador de decimales dentro de Matlab.
- Configurar la cadena que sirve como identificador interno de Matlab cuando devuelve un mensaje de error en la ejecución de un comando.
- Establecer el nombre del servicio SOAP que se va a utilizar para comunicarse con el servicio remoto SOAP, que permite manejar cada uno de los diferentes tipos de objetos creados (métodos independientes para cada tipo de objeto).
- Establecer la URL raíz en la cual se encontrarán situados los servicios SOAP a los que se tendrá acceso.
- Establecer el nombre del servicio SOAP que se utilizará para establecer una comunicación con el servicio remoto SOAP.
- Establecer si debe mostrarse o no el último comando que se mandó a ejecutar en Matlab al momento de producirse una excepción en el componente.
- Establecer si debe mostrarse o no el mensaje de error devuelto por Matlab, en caso de que lo haya al momento de producirse una excepción.
- Modificar la cadena usada como delimitador de cadenas dentro de Matlab.
- Definir una variable con un valor de arreglo de caracteres en el entorno actual de la sesión de Matlab.
- Definir una variable con un valor real doble, el entorno actual de la sesión de Matlab.
- Definir el valor de una variable de Matlab que esté definida en el entorno de trabajo actual como un arreglo de valores numéricos dobles (parte real y parte imaginaria).
- Definir una variable como un arreglo bidimensional de valores dobles en el entorno de trabajo actual de Matlab.
- Definir una variable como un valor numérico entero en el entorno actual de la sesión de Matlab.
- Definir una variable como un valor de tipo cadena de caracteres en el entorno actual de la sesión de Matlab.
- Escribir en la propiedad “Visible” para poder establecer que la ventana de la sesión de Matlab debe ser o no visible en el entorno de trabajo de Windows.

- Establecer una cadena de texto la cual se usa para saber si Matlab ha devuelto un mensaje de advertencia.
- Establecer si al momento de eliminar un objeto se desea eliminar también el elemento de Matlab que se encuentra encapsulado por dicho objeto.
- Establecer el identificador del objeto SOAP mediante el cual se identificará el mismo para poder establecer una comunicación entre éste y un servicio remoto SOAP.
- Establecer el nombre del servicio SOAP que se utilizará para establecer una comunicación entre el servicio SOAP y el servicio remoto SOAP.
- Asignar una variable a la sesión de Matlab indicada por el usuario, en la que se manipulará dicha variable, de ésta manera, se puede reutilizar valores e implementar más métodos a la misma variable.
- Asignar un nombre específico, deseado por el programador, a alguna variable en específico.
- Establecer o inicializar una matriz multidimensional con valores del tipo del objeto.
- Establecer el tamaño de un vector de valores numéricos dobles que se desee definir por el usuario.
- Establecer o inicializar una matriz bidimensional con valores del tipo del objeto.
- Establecer o inicializar una matriz tridimensional con valores del tipo del objeto.
- Establecer un arreglo de valores numéricos dobles del tipo del objeto que encapsula este tipo de dato.
- Establecer un valor de tipo complejo (parte real y parte imaginaria), definido por el usuario, dentro de un arreglo de valores dobles en una posición deseada por el usuario.
- Establecer un valor de tipo real, definido por el usuario, dentro de un arreglo de valores dobles en una posición deseada por el usuario.
- Establecer un valor de tipo real, definido por el usuario, dentro de una matriz bidimensional de valores dobles en una posición (x, y) deseada por el usuario.
- Establecer un valor de tipo real, definido por el usuario, dentro de una matriz tridimensional de valores dobles en una posición (x, y, z) deseada por el usuario.
- Inicializar el tamaño de un vector de tipo arreglo de celdas que se desee definir por el usuario.
- Inicializar una variable matriz bidimensional (x, y) con valores del tipo del objeto (arreglo de celdas).
- Inicializar una variable matriz tridimensional (x, y, z) con valores del tipo del objeto (arreglo de celdas).
- Asignar/Escribir valores de tipo cadena de caracteres a una variable deseada, para que pueda ser manipulada como el nuevo tipo de dato asignado.
- Establecer o inicializar una matriz multidimensional con valores del tipo del objeto (arreglo de celdas).
- Establecer/escribir un valor a una variable matriz de celdas (cellarray) de valores de cualquier tipo, en una posición deseada por el usuario.
- Establecer/escribir un valor de tipo numérico doble a una variable matriz de celdas (cellarray) de valores del mismo tipo, en una posición deseada por el usuario.

- Establecer/escribir un valor de tipo cadena de caracteres a una variable matriz de celdas (cellarray) de valores del mismo tipo, en una posición deseada por el usuario.
- Establecer un nuevo valor a un campo específico, en una posición específica (dados por el usuario) dentro de una matriz de registros.
- Establecer/escribir un valor de tipo numérico doble en un campo específico dado por el usuario y en una posición dada por el mismo, dentro de una matriz de registros.
- Establecer/escribir un valor dado por el usuario de tipo cadena de caracteres en un campo y posición específico, dentro de una matriz de registros.
- Establecer el ID de la ventana gráfica a activar, la cual contendrá el resultado arrojado por la ejecución de la función indicada, siempre y cuando, la función ejecutada contenga una salida gráfica, la cual requiera ser representada mediante una figura.
- Asignar a una variable el nombre de la función que se quiere ejecutar.
- Establecer los parámetros o argumentos de tipo numéricos de la función deseada con los que va a ser ejecutado el llamado a Matlab.
- Establecer el nombre en específico de una variable en la que se desea se muestre el resultado de la función ejecutada.
- Establecer la sesión de Matlab dentro de la que se manipulará la variable que encapsule el objeto actual, es decir, sobre la que se va hacer la manipulación de las diferentes variables a utilizar.
- Establecer los parámetros o argumentos de tipo cadena de caracteres de la función deseada con los que va a ser ejecutado el llamado a Matlab.
- Establecer el nombre de una variable en una posición indicada por el usuario, la cual funcionará como argumento de la función para la ejecución de la misma.
- Establecer la propiedad “Visible” para poder determinar si la figura de la sesión de Matlab debe ser o no visible en el entorno de trabajo de Windows.
- Establecer el contenido de alguna propiedad, indicando el nombre de la propiedad como un valor de tipo numérico.
- Establecer un nuevo valor en la propiedad de un objeto especificado por el usuario mediante el nombre de la misma.
- Establecer el contenido de alguna propiedad, indicando el nombre de la propiedad como un valor de tipo cadena de caracteres.
- Establecer una matriz de caracteres dentro del entorno de Matlab.
- Establecer un caracter en una posición determinada dentro de una matriz de caracteres establecida dentro del entorno de Matlab.
- Establecer el contenido de un renglón indicado dentro de una matriz de caracteres.
- Establecer el caracter que se va usar como separador de decimales al momento de pasar un valor numérico a Matlab.
- Establecer cuál será la cadena de error que el componente utilizará para indicar que la ejecución de un comando dentro de Matlab ha producido un error.
- Establecer el caracter que se va usar como separador de cadenas al momento de pasar un valor de tipo cadena a Matlab.

- Establecer cuál será la cadena de advertencia, que el componente utilizará para indicar que la ejecución de un comando dentro de Matlab, ha producido un aviso.
- Establecer o modificar el nombre del servicio que se utilizará por defecto al momento de tratar de conectarse a un servicio remoto mediante SOAP, esto es, a través de las interfaces propias de cada servicio (métodos independientes para cada tipo de servicio).
- Establecer o modificar la URL base del componente la cual va a ser usada por el mismo cuando se trate de acceder a los servicios remotos mediante SOAP.
- Establecer el idioma en que el componente va a devolver los resultados de respuesta esperados tras su uso.
- Establecer si el componente, en los métodos que devuelvan arreglos del tipo OleVariant, debe devolver únicamente arreglos cuyos elementos sean también del tipo Variant, o puede devolver arreglos cuyos elementos sean de otros tipos (cadenas, double, etc.).

Estas operaciones hacen posible la escritura para cada tipo de dato, de tal manera que se pueda corregir su contenido. Al mismo tiempo, son métodos que hacen posible, que el usuario pueda asignar un valor de su preferencia a cada tipo de dato.

6. Características de los usuarios

Los usuarios son todas aquellas personas que se interesen por hacer uso de los componentes COM y/o SOAP, en la realización de nuevas aplicaciones. El número de usuarios que los componentes aceptan son:

- Mediante el uso del componente COM, un usuario a la vez por equipo de cómputo.
- Mediante el uso del componente SOAP, tantos usuarios como se desee mientras puedan acceder al servidor donde se encuentran publicados los servicios WEB.

Para el uso de los componentes es necesario que los usuarios tengan un nivel de experiencia de mínimo a medio en cuanto a desarrollo del software se refiere.

7. Restricciones

Los componentes de software estarán orientados únicamente al soporte de los tipos de datos y variables que maneja Matlab, así como al manejo de sesiones del mismo. No realizarán funciones matemáticas específicas, sin embargo podrán soportar otros componentes que las realicen.

Por otra parte, no existen restricciones en cuanto al alcance de las operaciones contenidas en los componentes, ya que cualquier usuario que desee utilizar los componentes tendrá acceso a todos y cada uno de los métodos.

Aunque sí existen algunas restricciones para el uso de los componentes:

- Componente COM: es necesario contar con un lenguaje de programación que soporte tecnología COM para el correcto funcionamiento del componente. Sin embargo, esto no es gran problema ya que en la actualidad la mayoría de los lenguajes de programación más comunes para el desarrollo de aplicaciones software soportan dicha tecnología.

- Componente SOAP: también es necesario contar con un lenguaje de programación que soporte tecnología SOAP, además, tener una computadora conectada a una red y que ofrezca los servicios de Matlab para el correcto funcionamiento del componente. De la misma manera, esto no es gran problema, ya que en la actualidad la mayoría de los lenguajes de programación para desarrollo de aplicaciones Web cuentan con un soporte para dicha tecnología.

8. Suposiciones y dependencias

Dentro de este apartado se listan las suposiciones y dependencias en cuanto al desarrollo de los componentes.

8.1. Suposiciones

Se da por hecho que los requerimientos descritos en el documento ERS son aceptables una vez que este proyecto ha sido aprobado como tema de tesis. Cualquier petición de cambio en las especificaciones del proyecto será considerada mientras no afecte el curso total o parcial del mismo, de lo contrario, serán propuestas como desarrollos futuros.

8.2. Dependencias

Los componentes en este proyecto son totalmente dependientes de Matlab. Para el componente SOAP, se anexa una nueva dependencia que consiste en la necesidad de tener un equipo de cómputo que ofrezca los servicios de Matlab y que se encuentre conectado a una red.

Requerimientos específicos

En esta sección se presentan los requerimientos funcionales que deberán cumplir los componentes de software. Todos los requerimientos aquí citados son esenciales, esto quiere decir que no es aceptable que el proyecto no cumpla con alguno de los requerimientos presentados.

9. Requerimientos funcionales

9.1. Manejo de sesiones, ejecución de comandos y soporte para el uso de diferentes tipos de datos.

Req (1) Cada vez que el usuario desee obtener los servicios de Matlab, ejecutar comandos y crear diferentes tipos de datos, deberá abrir una sesión, obtener el identificador de la sesión y apoderarse de ella mediante el identificador de la misma, de esta manera podrá trabajar sobre un entorno de Matlab y realizar las operaciones correspondientes.

Req 1.1 Crear una sesión de Matlab y obtener el identificador correspondiente, si la sesión ya se encuentra creada, únicamente obtener el identificador de la sesión.

Req 1.2 Apropiarse de la sesión sobre la cual se desea trabajar mediante su identificador.

Req 1.3 Crear variables y tipos de datos de los diferentes tipos que maneja Matlab (arreglo de caracteres, valor numérico doble, arreglo de valores numéricos dobles (parte real e imaginaria), arreglo plano bidimensional de valores dobles (parte real e imaginaria), entero, cadena, figuras) los cuales serán el objeto a pertenecer en la sesión creada.

Req 1.4 Asignar la sesión a las variables creadas mediante el identificador de sesión.

Req 1.5 Ejecutar comandos de texto dentro de la sesión creada, tales como: guardar entornos de matlab o figuras a disco, copiar figuras, manipular variables, importar objetos de una sesión a otra, interactuar con el ambiente de Windows, generar excepciones cuando ocurra un error en Windows y realizar conversiones de un tipo de dato a otro.

Req 1.6 Manipular las sesiones creadas, esto quiere decir, poder abrir mas sesiones, cerrar las sesiones abiertas, guardar las sesiones deseadas y ver si está activa o no alguna sesión.

9.2. Lectura de valores de las variables.

Req (2) Cada vez que el usuario desee hacer una operación de lectura de alguna variable creada, así como la obtención del contenido de los diferentes tipos de datos, el usuario deberá tener abierta una sesión, obtener el identificador de la sesión en la que desea trabajar y apoderarse de la sesión mediante su identificador. De igual manera, es necesario que haya creado alguna variable o tipo de dato en el entorno de Matlab sobre el cual desea hacer la lectura.

Req 2.1 Crear una sesión de Matlab y obtener su identificador, si la sesión ya está abierta únicamente obtener el identificador de sesión.

Req 2.2 Apropiarse de la sesión sobre la cual se desea trabajar mediante su identificador.

Req 2.3 Haber creado el objeto variable o tipo de dato deseado.

Req 2.4 Asignar al nuevo objeto creado la sesión o entorno de trabajo actual de Matlab mediante el identificador de la sesión.

Req 2.5 El objeto creado podrá hacer cualquier tipo de operación de lectura soportada por el componente.

9.3. Escritura de valores en las variables.

Req (3) Cada vez que el usuario desee hacer una operación de escritura o modificación de alguna de las variables creadas o de los diferentes tipos de datos, el usuario deberá tener abierta una sesión, obtener el identificador de la sesión en la que desea trabajar y apoderarse de la sesión mediante su identificador. De la misma manera, es necesario que haya creado alguna variable o tipo de dato en el entorno de Matlab sobre el cual desea hacer la escritura o modificación del contenido del objeto.

Req 3.1 Crear una sesión de Matlab y obtener su identificador, si la sesión ya se encuentra abierta, únicamente obtener el identificador de sesión.

Req 3.2 Apropiarse de la sesión sobre la cual se desea trabajar mediante su identificador.

Req 3.3 Haber creado el objeto variable o tipo de dato deseado.

Req 3.4 Asignar al nuevo objeto creado la sesión o entorno de trabajo actual de Matlab mediante el identificador de la sesión.

Req 3.5 El objeto creado podrá hacer cualquier tipo de operación de escritura siempre y cuando el componente lo soporte.

10. Interfaces Externos

10.1. Interfaces Software

Los componentes cuentan con una serie de interfaces de software que hacen posible su correcto funcionamiento. Estas interfaces contienen cada una de las funciones propias de la interfaz, de ésta manera, esas funciones son denominadas métodos, los cuales hacen posible el funcionamiento total de los componentes. A continuación se describe cada una de las interfaces, doce en su totalidad, con sus respectivos métodos.

10.1.1. IMOMatLabSession

Esta interfaz es una de las de mayor importancia, debido a que es la encargada de proporcionar las funciones: apertura de sesión en modo exclusivo (single user) de Matlab; ejecución de comandos de texto y otras funciones auxiliares, como recuperación de valores de variables a partir del nombre; comprobación de la existencia de una variable; soporte de diferentes tipos de datos; manipulación de variables; etc. Esta interfaz utiliza la interfaz COM de tipo Ole Automation Server, proporcionada por Matlab para abrir una sesión, ejecutar comandos e intercambiar variables de tipo matriz doble.

Tabla 3.4. Métodos de la interfaz IMOMatLabSesion

▪ Clear	▪ GetRemoteID
▪ ClearVar	▪ GetRemoteNumericArrayServiceName
▪ Close	▪ GetRemoteObjectServiceName
▪ CloseFigure	▪ GetRemoteRootURL
▪ Convert1xNArrayToVector	▪ GetRemoteServiceName
▪ ConvertClassIDToClassName	▪ GetRemoteStringServiceName
▪ ConvertClassNameToClassId	▪ GetRemoteStructArrayServiceName
▪ ConvertDoubleFlatArrayToString	▪ GetRemoteVarServiceName
▪ ConvertFlatArrayToNDims	▪ GetStringDelimiter
▪ CreateCellArray	▪ GetTextOfLastCommandGotError
▪ CreateCharArray	▪ GetTmpAsDouble
▪ CreateFigure	▪ GetTmpAsInteger
▪ CreateFunctionCaller	▪ GetTmpAsString
▪ CreateNumericArray	▪ GetTmpVar
▪ CreateObject	▪ GetVarAsCharArray
▪ CreateString	▪ GetVarAsDouble
▪ CreateStructArray	▪ GetVarAsDoubleArray
▪ CreateVar	▪ GetVarAsDoubleFlatArray
▪ Execute	▪ GetVarAsInteger

▪ ExecuteOrExcept ▪ FigureExists ▪ GetActiveFigure ▪ GetClassIdOf ▪ GetClassOf ▪ GetClearOnClose ▪ GetDecimalSeparator ▪ GetDetailedVarList ▪ GetErrorStringPrefix ▪ GetFigureList ▪ GetFigureToClipboard ▪ GetIMOMatlabInterfaceObject ▪ GetInfinity ▪ GetInstaledToolboxes ▪ GetLastCommandGotError ▪ GetLastErrorMessage ▪ GetLastFullAnswer ▪ GetLastMatlabErrorOnException ▪ GetLastWarningMessage ▪ GetMatLabEngine ▪ GetMatlabVersion ▪ GetNotANumber ▪ GetRemoteCellArrayServiceName ▪ GetRemoteCharArrayServiceName ▪ GetRemoteFigureServiceName ▪ GetRemoteFunctionCallerServiceName ▪ SetRemoteCellArrayServiceName ▪ SetRemoteCharArrayServiceName ▪ SetRemoteFigureServiceName ▪ SetRemoteFunctionCallerServiceName ▪ SetRemoteID ▪ SetRemoteNumericArrayServiceName ▪ SetRemoteObjectServiceName ▪ SetRemoteRootURL ▪ SetRemoteServiceName ▪ SetRemoteStringServiceName ▪ SetRemoteStructArrayServiceName ▪ SetRemoteVarServiceName	▪ GetVarAsSerializedDoubleVector ▪ GetVarAsString ▪ GetVarDims ▪ GetVarList ▪ GetVisible ▪ GetWarningString ▪ IsFinite ▪ IsInfinite ▪ IsLocalSession ▪ IsNotANumber ▪ LastCommandGotWarning ▪ LoadSession ▪ Mat_FigureFormatToString ▪ MatlabFloatToStr ▪ ObtainRandomName ▪ OpenRemote ▪ OpenSession ▪ RaiseMatlabEngineException ▪ SaveFigure ▪ SaveSession ▪ SessionIsOpened ▪ SetActiveFigure ▪ SetClearOnClose ▪ SetDecimalSeparator ▪ SetErrorStringPrefix ▪ SetNewFigure ▪ SetShowMatlabCommandOnException ▪ SetShowMatlabErrorOnException ▪ SetStringDelimiter ▪ SetVarAsCharArray ▪ SetVarAsDouble ▪ SetVarAsDoubleArray ▪ SetVarAsDoubleFlatArray ▪ SetVarAsInteger ▪ SetVarAsString ▪ SetVisible ▪ SetWarningString ▪ VarExists

10.1.2. IMOMatlabToken

La interfaz IIMOMatlabToken es la interfaz base de todos los elementos que se manejan en una sesión de Matlab. La importancia de esta interfaz radica en que se vuelve la base de todos los elementos que se manejan dentro de una sesión de Matlab a través de las interfaces que descienden de ésta, por ejemplo las IMOMatlabVar, IMOMatlabFigure, entre otras que descienden de ella. Esta interfaz soporta los métodos comunes para manipular el nombre del

servicio remoto, el ID remoto, el método Reset, etc. La implementación de ésta interfaz en IIMOCOMMatlabToken, dado que se orienta a manejar objetos locales, éstos métodos no tienen efecto. En cambio, cabe resaltar que al implementarla en IMOSOAPMatlabToken estos métodos sí que tienen funcionalidad.

Tabla 3.5. Métodos de la interfaz IMOMatlabToken

▪ GetClearOnDestroy ▪ GetRemoteID ▪ GetRemoteRootURL ▪ GetRemoteServiceName	▪ Reset ▪ SetClearOnDestroy ▪ SetRemoteID ▪ SetRemoteServiceName

10.1.3. IMOMatLabVar

Esta interfaz encapsula una variable genérica de Matlab, de cualquier tipo. Ofrece los métodos genéricos para crear, importar, averiguar el tipo, dimensiones, etc., de una variable. También contiene métodos para obtener una interfaz descendiente de IMOMatlabVar que encapsule la misma variable de Matlab, pero se oriente a manipular un tipo concreto de variable, por ejemplo AsMatLabNumericArray, AsMatLabCellArray, etc. De esta interfaz descienden las interfaces que definen métodos más concretos, aplicables a tipos determinados de variables (IMOMatlabCellArray, IMOMatlabNumericArray, entre otras). Se necesita tener asociado a cualquier objeto IIMOMatlabVar (o descendientes) una sesión de Matlab, es decir, un objeto IIMOMatlabSession. La variable que se encapsula en ésta interfaz debe tener asignado un nombre. Por defecto se le asigna uno aleatorio. Pero la variable no es realmente creada en el entorno Matlab hasta que se le proporciona un valor. Esto puede hacerse manualmente utilizando comandos de Matlab (por ejemplo a través del método Execute de la interfaz IIMOMatlabSession), o con los métodos adecuados de la interfaz correspondiente. IIMOMatlabVar no ofrece estos métodos, ya que se destina a manipular variables en general, y no ofrece métodos para manipular tipos concretos. En cuanto a las dimensiones de una variable, hay que tener en cuenta que en Matlab se empiezan a numerar desde 1, y que toda variable de Matlab se considera un array, como mínimo de 1x1. Una variable puede tener un número arbitrario de dimensiones de cualquier tamaño.

Tabla 3.6. Métodos de la interfaz IMOMatLabVar

▪ AsMatLabCellArray ▪ AsMatlabCharArray ▪ AsMatLabNumericArray ▪ AsMatlabObject ▪ AsMatLabString ▪ AsMatLabStructArray ▪ ChangeVarName ▪ CopyFrom ▪ GetClassID ▪ GetClassName ▪ GetCopy ▪ GetDimension ▪ GetDimensions	▪ GetVarName ▪ GetXYPos ▪ GetXYZPos ▪ ImportFromSession ▪ IsA ▪ IsCellArray ▪ IsCellOfCellArray ▪ IsCharArray ▪ IsFieldOfStruct ▪ IsNumericArray ▪ IsObject ▪ IsPropertyOfObject ▪ IsScalar

▪ GetDimensionsCount ▪ GetIndexPos ▪ GetLength ▪ GetSessionOwner ▪ GetType	▪ IsString ▪ IsStructArray ▪ SetSessionOwner ▪ SetVarName

10.1.4. IMOMatLabNumericArray

Esta interfaz encapsula una variable que sea una matriz de valores numéricos dobles (reales o complejos) de Matlab. En Matlab este es el tipo por defecto de cualquier valor numérico o matriz de datos numéricos. Puede tener o no una parte imaginaria. Esta interfaz desciende de la IIMOMatlabVar, de la que hereda los servicios genéricos. Aquí se añaden servicios concretos para matrices numéricas como: leer y escribir valores numéricos en una posición de la matriz, inicializar una matriz, entre otros.

Tabla 3.7. Métodos de la interfaz IMOMatLabNumericArray

▪ GetArray ▪ GetRealValueAt ▪ InitMultiDimMatrix ▪ InitXYMatrix ▪ IsComplex ▪ SetComplexValueAt ▪ SetRealValueAtXY	▪ GetComplexValueAt ▪ GetSerializedArray ▪ InitVector ▪ InitXYZMatrix ▪ SetArray ▪ SetRealValueAt ▪ SetRealValueAtXYZ

10.1.5. IMOMatLabString

Esta interfaz encapsula una variable que sea una cadena de caracteres de Matlab. También desciende de la IIMOMatlabVar, de la que hereda los servicios genéricos. Aquí se añaden servicios concretos para leer y escribir valores de cadena.

- GetString
- SetString

10.1.6. IMOMatLabCellArray

Esta interfaz encapsula una variable que sea una matriz de celdas de Matlab (cellarray). Asimismo, desciende de la IIMOMatlabVar, de la que hereda los servicios genéricos. Aquí se añaden servicios concretos para matrices numéricas como leer y escribir valores de diferentes tipos en una posición de la matriz, inicializar una matriz, obtener información detallada de una celda, recuperar todos los valores del arreglo serializados, entre otras.

Tabla 3.8. Métodos de la interfaz IMOMatLabCellArray

▪ GetAllAsNumArrays ▪ GetAllAsSerializedVector ▪ GetCellAsVar ▪ GetDetailedCellInfo ▪ GetStringCellAt ▪ InitVector ▪ InitXYZMatrix ▪ SetNumCellAt	▪ GetAllAsNumVector ▪ GetAllAsString ▪ GetCellAt ▪ GetNumCellAt ▪ InitMultiDimMatrix ▪ InitXYMatrix ▪ SetCellAt ▪ SetStringCellAt

10.1.7. IMOMatLabStructArray

Esta interfaz encapsula una variable que sea una matriz de registros (struct array) de Matlab. De igual manera, desciende de la IIMOMatlabVar, de la que hereda los servicios genéricos. Aquí se añaden servicios concretos para matrices de registros como: ver qué campos tienen los registros, añadir o quitar campos, consultar o modificar el valor de un campo, obtener información detallada de los campos de un registro, recuperar todos los valores de un campo de los registros del arreglo serializados, etc.

Tabla 3.9. Métodos de la interfaz IMOMatLabStructArray

▪ GetAllArrayFieldValues ▪ GetAllNumFieldValues ▪ GetDetailedFieldList ▪ GetFieldList ▪ GetFieldsCount ▪ GetNumFieldAt ▪ IsField ▪ SetFieldValueAt ▪ SetStringFieldValueAt	▪ GetAllAsSerializedVector ▪ GetAllStringFieldValues ▪ GetFieldAsVar ▪ GetFieldName ▪ GetFieldValueAt ▪ GetStringFieldAt ▪ RemoveField ▪ SetNumFieldValueAt

10.1.8. IMOMatlabFunctionCaller

Esta interfaz encapsula una llamada a una función de Matlab. Permite realizar una llamada a una función cualquiera de Matlab, indicando el nombre de la función, sus parámetros, dónde guardar el resultado, etc. El uso general de esta interfaz es: Creamos un objeto con interfaz y le asignamos una sesión matlab (un objeto IMOmatlabSession). Asignamos el nombre de la función de Matlab que queremos ejecutar con el método SetFunctionName. Indicamos los parámetros o argumentos de la función usando: - para valores numéricos: SetNumArgumentAt (position: Integer; value: Double); - para cadenas de texto: SetStringArgumentAt (position: Integer; const value: WideString); - para indicar nombres de variables: SetVarArgumentAt (position: Integer; const value: WideString); - Los argumentos se pueden manipular con métodos como GetArgumentAt, ClearArgumentAt, GetNumOfArgs para contarlos, borrarlos, etc. Podemos indicar el nombre de la variable a la que se asignará el resultado con SetResultName. Lo mismo para una figura si hay salida gráfica con SetFigureResult. Podemos recuperar la cadena que se mandará ejecutar en Matlab con GetExecutionString. Ejecutamos la

llamada a la función en Matlab con ExecuteCall. El resultado lo recuperamos como un objeto IMOMatlabVar a través de GetResult o consultando el valor de la variable.

Tabla 3.10. Métodos de la interfaz IMOMatlabFunctionCaller

▪ GetResultName. ▪ ClearArguments ▪ GetArgumentAt ▪ GetFigureResult ▪ GetNumOfArgs ▪ GetResultName ▪ SetFigureResult ▪ SetNumArgumentAt ▪ SetSessionOwner ▪ SetVarArgumentAt	▪ ClearArgumentAt ▪ ExecuteCall ▪ GetExecutionString ▪ GetFunctionName ▪ Getresult ▪ GetSessionOwner ▪ SetFunctionName ▪ SetResultName ▪ SetStringArgumentAt

10.1.9. IMOMatLabFigure

Esta interfaz permite manipular una figura de Matlab. Debe asignársele una sesión (un objeto IMOMAtlabSession). Permite crear una figura; manipular una figura ya creada; almacenarla en un archivo o copiarla al portapapeles.

Tabla 3.11. Métodos de la interfaz IMOMatLabFigure

▪ Close ▪ CreateFigure ▪ GetSessionOwner ▪ ImportFromSession ▪ SetSessionOwner	▪ CopyToClipboard ▪ GetFigureHandle ▪ GetVisible ▪ SaveFigure ▪ SetVisible

10.1.10. IMOMatlabObject

Esta interfaz encapsula una variable que sea un objeto de una clase definida en una toolbox de Matlab (por ejemplo la clase red de neuronas). Esta interfaz desciende de la IIMOMatlabStructArray, de la que hereda los servicios genéricos para manejar registros, ya que un objeto se comporta como un registro. Aquí se añaden servicios concretos para objetos como ver las propiedades del objeto, leer y escribir valores en propiedad, entre otros.

Tabla 3.12. Métodos de la interfaz IMOMatlabObject

▪ GetDetailedPropertiesList ▪ GetPropertiesCount ▪ GetPropertyAsVar ▪ GetPropertyValue ▪ IsProperty ▪ SetPropertyValue	▪ GetNumProperty ▪ GetPropertiesList ▪ GetPropertyName ▪ GetStringProperty ▪ SetNumProperty ▪ SetStringProperty

10.1.11. IMOMatlabCharArray

Esta interfaz encapsula una variable que sea una matriz de caracteres de Matlab. Se distingue de la IMOMatlabString que ésta última sólo permite manejar cadenas, es decir, matrices de 1xN caracteres. La IIMOMatlabCharArray, en cambio permite manejar matrices de caracteres de cualquier dimensión. Esta interfaz desciende de la IIMOMatlabVar, de la que hereda los servicios genéricos. Aquí se añaden servicios concretos para matrices de caracteres como: leer y escribir caracteres en una posición de la matriz, recuperar o modificar una fila de la matriz, etc.

Tabla 3.13. Métodos de la interfaz IMOMatlabCharArray

▪ GetArray ▪ GetRowAt ▪ SetCharAt	▪ GetCharAt ▪ SetArray ▪ SetRowAt

10.1.12. IMOMatlabInterface

Esta interfaz define métodos y propiedades generales de la librería del componente IMO-COM for Matlab, como el idioma que se usará, las cadenas que identifican un error, una advertencia, el separador decimal, entre otros. También permite mostrar una "Aboutbox". Asimismo, permite definir la URL y los nombres por defecto, donde se localizarán los servicios SOAP, en caso de utilizar interfaces remotas del tipo IMOSOAPMatlabxxxx. IMOMatlabInterface permite definir parámetros que se utilizan en otras interfaces, así como dar un formato a los métodos y propiedades de la librería. Es mediante el uso de esta interfaz como se pueden definir las diferentes cadenas y caracteres necesarios para delimitar y expresar los diferentes resultados devueltos por Matlab, tras la ejecución de los métodos declarados a lo largo de la librería. Los métodos de esta interfaz son:

Tabla 3.14. Métodos de la interfaz IMOMatlabInterface

GetDefaultMatlabDecimalSeparator	GetDefaultMatlabErrorString
GetDefaultMatlabStringDelimiter	GetDefaultMatlabWarningString
GetDefaultRemoteCellArrayServiceName	GetDefaultRemoteCharArrayServiceName
GetDefaultRemoteFigureServiceName	GetDefaultRemoteFunctionCallerServiceName
GetDefaultRemoteNumericArrayService-Name	GetDefaultRemoteObjectServiceName
	GetDefaultRemoteSessionServiceName
GetDefaultRemoteRootURL	GetDefaultRemoteStructArrayServiceName
GetDefaultRemoteStringServiceName	GetLanguage
GetDefaultRemoteVarServiceName	SetDefaultMatlabErrorString
SetDefaultMatlabDecimalSeparator	SetDefaultMatlabWarningString
SetDefaultMatlabStringDelimiter	SetDefaultRemoteCharArrayServiceName
SetDefaultRemoteCellArrayServiceName	SetDefaultRemoteFunctionCallerServiceName
SetDefaultRemoteFigureServiceName	SetDefaultRemoteObjectServiceName
SetDefaultRemoteNumericArrayServiceName	SetDefaultRemoteSessionServiceName
SetDefaultRemoteRootURL	SetDefaultRemoteStructArrayServiceName
SetDefaultRemoteStringServiceName	SetLanguage
SetDefaultRemoteVarServiceName	ShowAboutBox
SetOnlyVariantArraysResult	

Cabe mencionar, que cada uno de los métodos pertenecientes a las interfaces, son soportados tanto por el componente COM como por el SOAP.

10.2. Interfaces de comunicación

Para el componente remoto se contará con una conexión Ethernet proporcionada por la red de la Universidad Tecnológica de la Mixteca, en donde se localizará un equipo de cómputo conectado a ésta red, el cual funcione como servidor de los servicios de Matlab.

11. Requerimientos de rendimiento

Debido a la estabilidad del lenguaje Matlab, se tiene previsto que no existan problemas en cuanto la cantidad de conexiones locales y remotas.

12. Requerimientos de desarrollo

El ciclo de vida elegido para el desarrollo de los componentes será el de espiral y el modelo a utilizar estará basado en el desarrollo orientado a componentes, esto debido a que el producto final de éste proyecto es un par de componentes software, por lo tanto, la mejor tecnología para el éxito del mismo es la orientada a componentes.

13. Requerimientos tecnológicos

Para utilizar el componente local COM, es necesario contar con un equipo de cómputo que cuente con la configuración mínima:

Procesador: Pentium III 750 Mhz.

Memoria: 128 Mb.

Espacio libre en disco: 1 Gb. (para la instalación de Matlab).

Sistema Operativo: Microsoft Windows 98, Me, XP Profesional (recomendado).

Para utilizar el componente remoto SOAP, es necesario contar con un equipo de cómputo que cuente con la configuración mínima siguiente:

Procesador: Pentium III 750 Mhz.

Memoria: 128 Mb.

Espacio libre en disco: 10 Mb.

Tarjeta Ethernet o Módem.

Acceso a Internet.

Sistema operativo: cualquiera que soporte los lenguajes de programación orientados a Web y soporte para la tecnología SOAP

Hasta este momento se han visto las bases teóricas y se ha hecho un análisis estructural de los requerimientos fundamentales que involucra el desarrollo total de los componentes.

3.2. Diseño de los componentes.

En el análisis se mostró los requerimientos de los componentes. Se propone la creación de once interfaces de software para cada uno de los componentes, el COM y el SOAP, y una interfaz de configuración que maneja ambos componentes. Cada interfaz tendrá una función específica y apuntará a la solución de los diferentes problemas que puedan presentarse. De esta manera, las interfaces tendrán un nombre que las ayude a identificarse, mismos que fueron identificados en el análisis y descritos en cuanto a su funcionalidad. Las interfaces para el componentes COM tendrán nombres similares a los que se identificaron en el análisis, con la diferencia que se les agregará las siglas COM para referirnos que forman parte de este componente, los métodos son los mismos al igual que la funcionalidad. Para las interfaces del componente SOAP serán nombres similares a los del COM, con la misma funcionalidad, únicamente se cambian las siglas COM por SOAP para cada una de las interfaces, lo que indica que son parte de dicho componente. Por ejemplo: IMOMatlabSession, para el componente COM será IMOCOMMatLabSession; mientras que, para el componente SOAP será IMOSOAPMatlabSession, y así para cada una de las demás interfaces.

En la Figura 3.1 se muestra una vista general de la funcionalidad y diseño de los componentes. Como se citó en el apartado 2.5.2, SOAP es un protocolo de comunicación mediante el paso de mensajes entre aplicaciones de software, o bien, entre componentes de software. Para esto el componente SOAP servirá como medio para que, el componente COM se comunique con aplicaciones cliente vía web, es decir, mediante servicios web. El componente SOAP hará posible dicha operación mediante el envío de mensajes al COM, mostrando los resultados que le retorne este último componente.

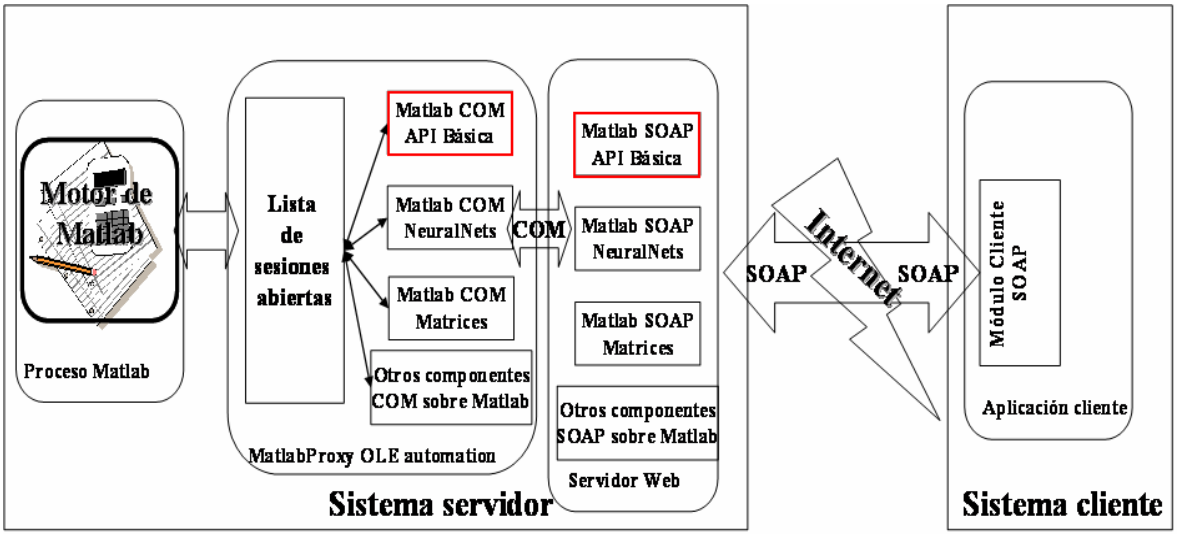


Figura 3.1. Vista General del Diseño de los Componentes

3.2.1. Arquitectura general de los componentes

Así como se cuenta con una arquitectura definida para el análisis y diseño de este proyecto de tesis, se tiene también el diseño de una arquitectura particular para cada uno de los

componentes de software, misma que es de gran importancia para la comprensión del trabajo realizado dentro de este proyecto.

En este apartado se detallaran de manera gráfica cada una de las arquitecturas diseñadas en cuanto al desarrollo de los componentes se refiere, esto con la finalidad de poder dar un amplio panorama sobre el comportamiento de los mismos.

3.2.1.1. Arquitectura para la librería Matlab COM

Esta sección muestra a detalle y de manera grafica la forma en la que interactúan cada uno de los elementos que conforman al componente COM, las interfaces, métodos y funcionalidad básica del mismo. Mediante esta arquitectura, se puede tener una definición y conocimiento general y bien estructurado sobre el comportamiento del componente.

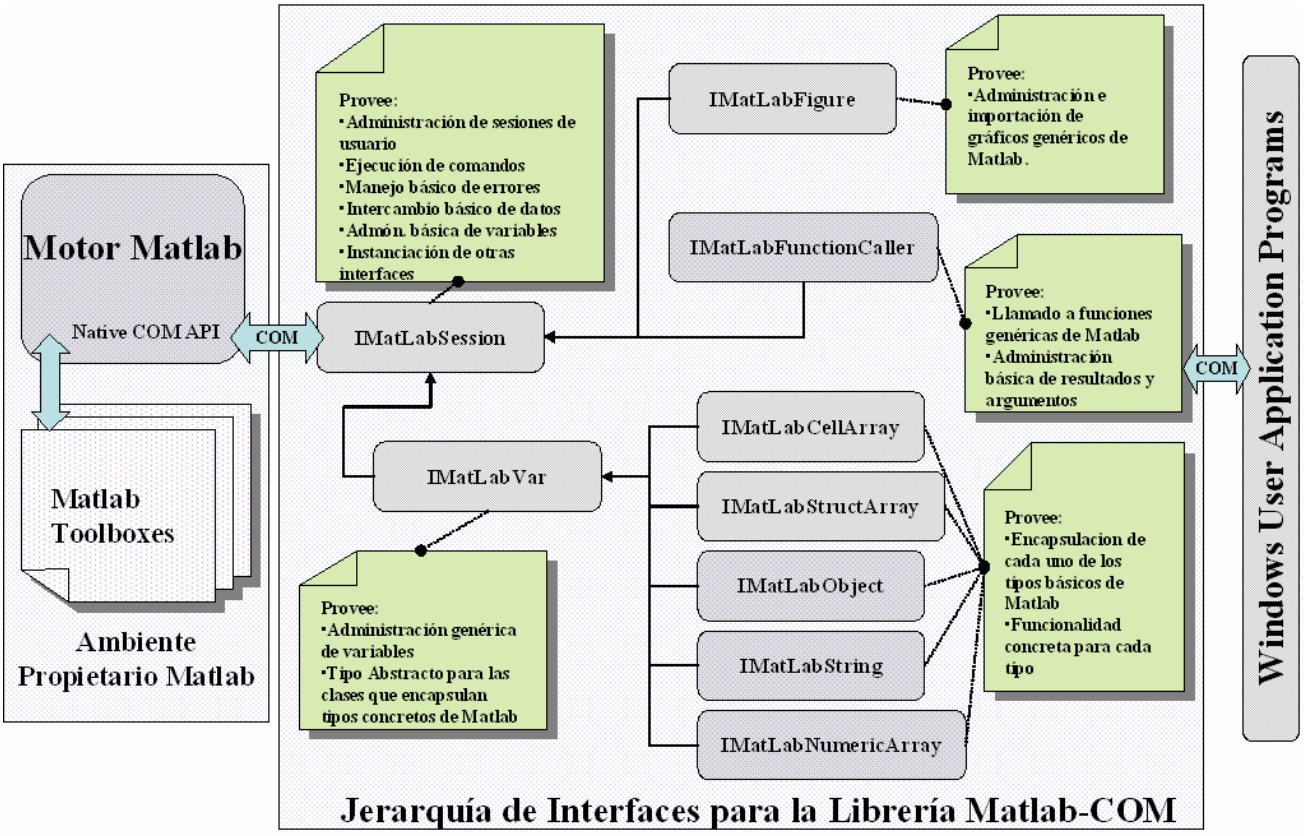


Figura 3.2. Arquitectura de la librería Matlab COM

3.2.1.2. Arquitectura para la librería Matlab SOAP

De la misma manera que con el componente COM, se diseñó un arquitectura específica para el componente SOAP. Esta arquitectura muestra de manera detallada la funcionalidad del componente SOAP así como la forma en la que interactúan los métodos e interfaces con el componente COM, la funcionalidad básica y general de los elementos y la forma en la que el componente se comunica con aplicaciones remotas que requieran su uso.

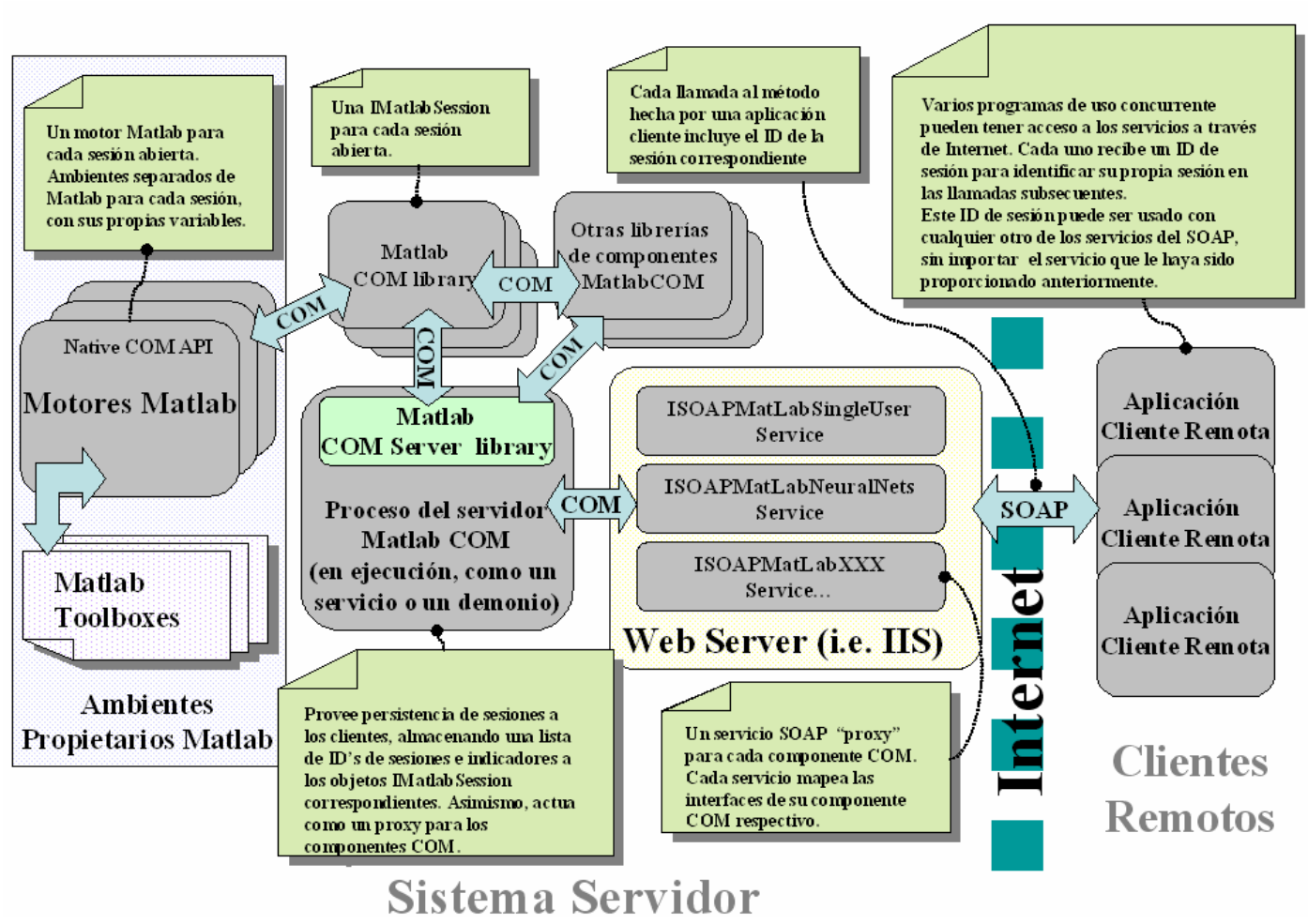


Figura 3.3. Arquitectura de la librería Matlab SOAP

Este diseño se basa en la creación de un sistema servidor que es el encargado de generar una lista de identificadores de sesiones por medio del cual se establece la comunicación entre el componente SOAP y cada sesión creada de Matlab, de manera que el identificador sirve como nuevo parámetro para cada método dentro de cada interfaz y de esta manera poder crear y manipular los objetos en la sesión deseada. Este sistema servidor es creado como un proceso en ejecución o un demonio que se encuentra corriendo al momento de abrir una sesión remota por medio del uso del componente SOAP.

De esta manera la arquitectura nos muestra a detalle la funcionalidad del componente, de aquí la importancia de incluirla en este apartado.

Una vez que se tiene planteada una idea del funcionamiento general y el diseño de los componentes, se tienen que descomponer los componentes en niveles plenamente identificables. Para este punto se utilizará una arquitectura de capas como patrón de diseño, la cual permitirá que estas capas o niveles se descompongan a su vez en tareas más pequeñas que llamaremos paquetes. La arquitectura de capas a utilizar se muestra en la Figura 3.4.

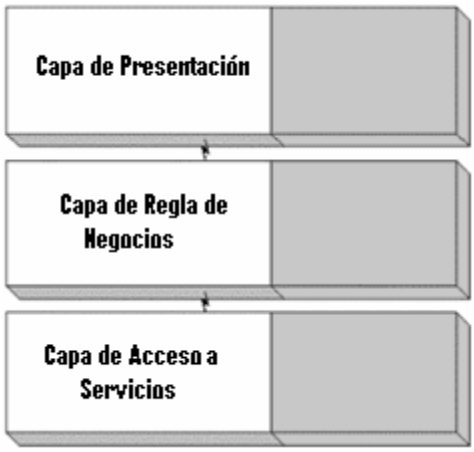


Figura 3.4. Arquitectura General de Capas

El descomponer el sistema en capas se simplifica el proceso de diseño y más tarde la implementación, ya que una capa usa y es usada a lo mucho por otra capa. Ahora bien, estas capas se van a descomponer en tareas más pequeñas, las cuales a su vez se van a convertir en componentes de software y servicios, que muestran cómo se llevará a cabo la solución.

En la Figura 3.5 se presenta una descripción más detallada de las capas de los componentes.

Figura 3.5. Descripción Lógica de las capas de los componentes

3.2.2. Diseño Orientado a Objetos

En capítulos anteriores hemos citado que el diseño de los componentes se va a desarrollar en base al diseño orientado a objetos, mediante el uso de la notación del Lenguaje Unificado de Modelado (UML), siguiendo el método general de Diseño Orientado a Objetos [38] que se compone de: Contexto y uso del sistema, Diseño Arquitectónico, Identificación de Objetos y Modelos de Diseño.

3.2.2.1. Contexto y uso del sistema

En esta sección mostraremos el modelo de paquetes que permite comprender de una mejor manera el contexto del sistema. Este modelo de paquetes ayuda a dividir los componentes en subsistemas de tal manera que se puedan repartir los procesos, tareas y responsabilidades a los trabajadores. De manera gráfica, se muestra este modelo en la Figura 3.6.

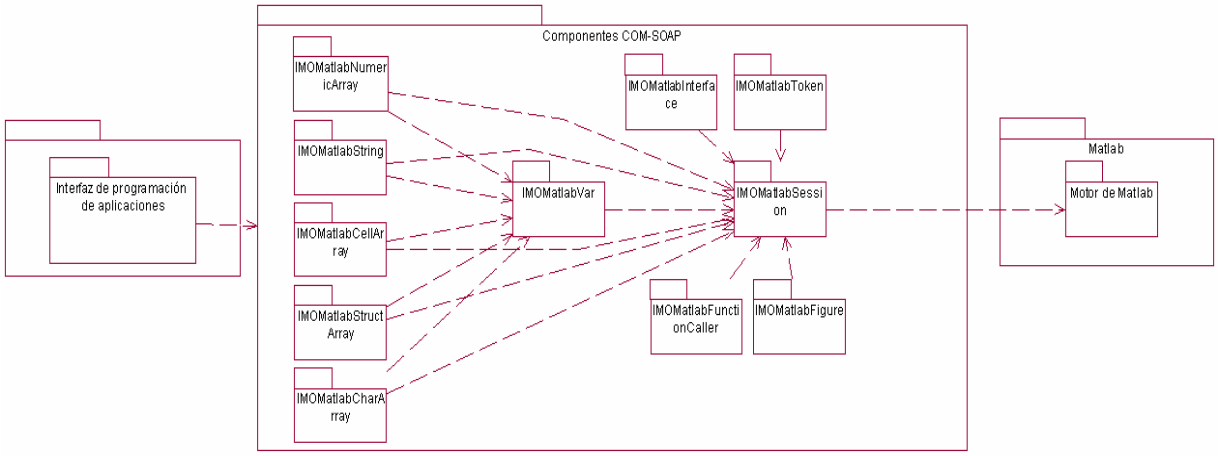


Figura 3.6. Contexto y uso del sistema (Modelo de paquetes para los componentes COM y SOAP)

En la figura se muestra que los componentes (COM y SOAP) mostrarán las siguientes dependencias:

- Las interfaces IMOMatlabNumericArray, IMOMatlabString, IMOMatlabCellArray, IMOMatlabStructArray y la IMOMatlabCharArray, muestran una dependencia doble: serán dependientes de la interfaz IMOMatlabVar y a su vez de IMOMatlabSession, que lleva el manejo de sesiones, esto aplica tanto para el componente COM como SOAP, respectivamente para cada interfaz.
- La interfaz IMOMatlabVar, IMOMatlabFunctionCaller, IMOMatlabFigure, IMOMatlabToken y la IMOMatlabInterface, únicamente serán dependientes de la interfaz IMOMatlabSession, tanto para el componente COM como el SOAP, respectivamente para cada interfaz.

Todas y cada una de las interfaces tienen dependencia de lo que es el motor de Matlab para poder llevar a cabo su función.

A continuación se muestra el diagrama general de casos de uso (Figura 3.7), que tiene como finalidad mostrar la relación que existe entre el usuario, los componentes de software y sus operaciones.

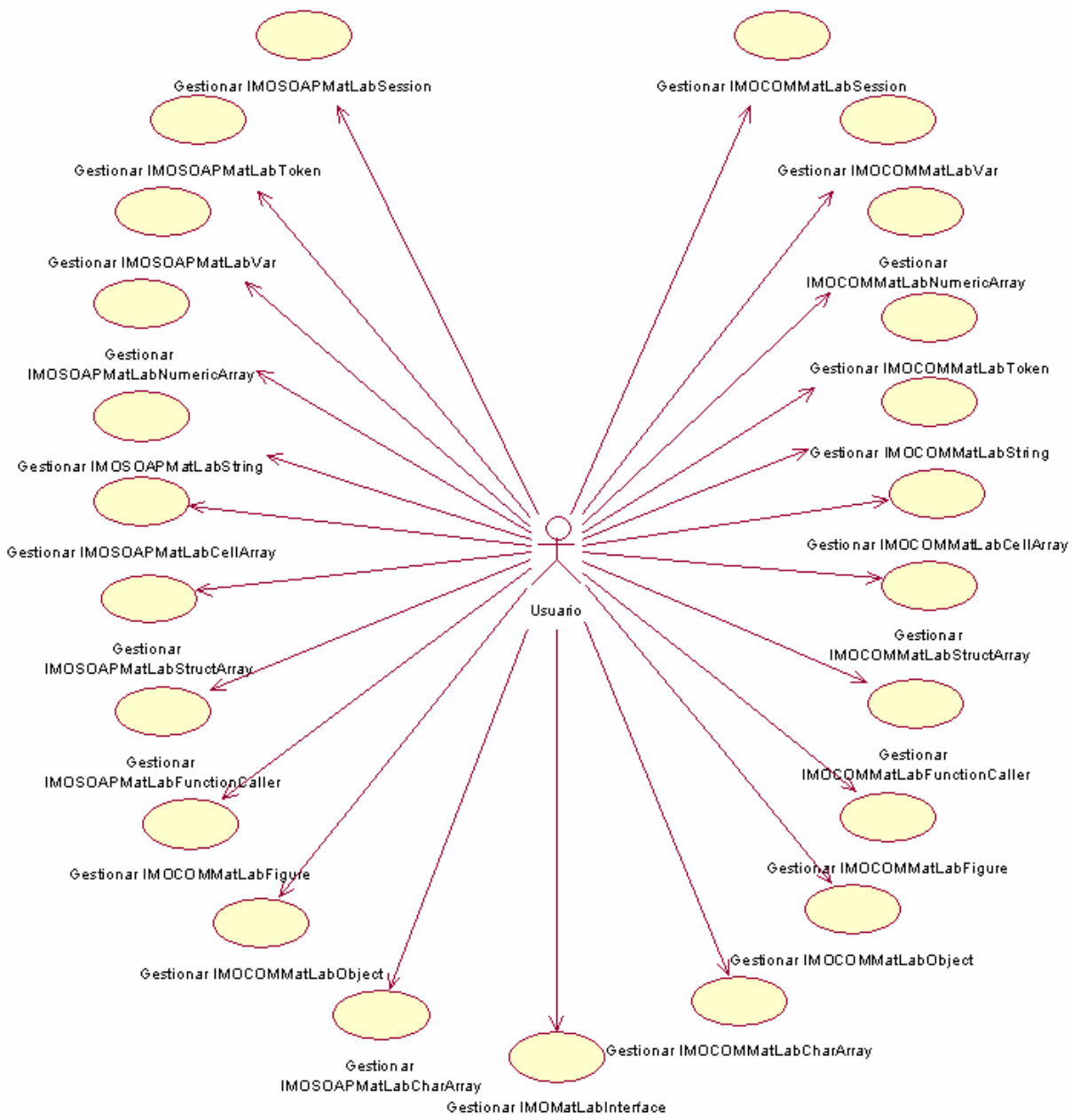


Figura 3.7. Diagrama general de casos de uso para los componentes software

En esta figura, se puede ver claramente que existen veintitrés casos de uso primarios para los componentes. Para ver con mayor detalle cada caso de uso, se utilizará la metodología del Proceso Unificado Rational (RUP) mediante la herramienta Rational Rose para la especificación de cada caso de uso. En el siguiente apartado se detallan sólo algunos de los casos de uso existentes, el resto se listan en el Anexo 3.

3.2.2.1.1. Especificación de Casos de Uso (ECU)

A continuación se describen dos casos de uso de manera expandida para un mejor entendimiento de los procesos y requerimientos.

Especificación de Caso de Uso: Gestionar IMOCOMMatLabSession

Versión 1.2

Historial de revisiones

Fecha	Versión	Descripción	Autor
22/Noviembre/2004	1.0	Creación del Documento	Josefina Flores Flores
26/Noviembre/2004	1.1	Introducción de primeros diagramas	Josefina Flores Flores
30/Noviembre/2004	1.2	Documento completado	Josefina Flores Flores

Escenario: Abrir una sesión de Matlab.

Objetivo

Gestionar la interfaz IMOCOMMatLabSession para abrir una sesión de Matlab.

Breve descripción

El usuario podrá gestionar la interfaz IMOCOMMatLabSession para abrir una sesión o entorno de trabajo de Matlab.

Actores

Usuario

Flujo de eventos

Flujo básico

Actor	Sistema
Este caso de uso comienza cuando el usuario decide instanciar la interfaz IMOCOMMatLabSession, y desea abrir una sesión de Matlab, así como crear una variable.	
El usuario utiliza el objeto el cual contiene la variable creada y ejecuta la función “OpenSession”.	
	El sistema ejecuta la operación, abriendo una sesión de Matlab para ser utilizada por éste objeto.
	El sistema devuelve el identificador de la sesión abierta para poder ser utilizado.
Finaliza el caso de uso.	

Excepción

Si no se puede instanciar la interfaz requerida, se notificará mediante un mensaje de error.
Si la sesión no se puede abrir, se notificará mediante un mensaje de error.

Precondiciones

Haber creado una variable del tipo IMOCOMMatLabSession.

Diagrama de actividades

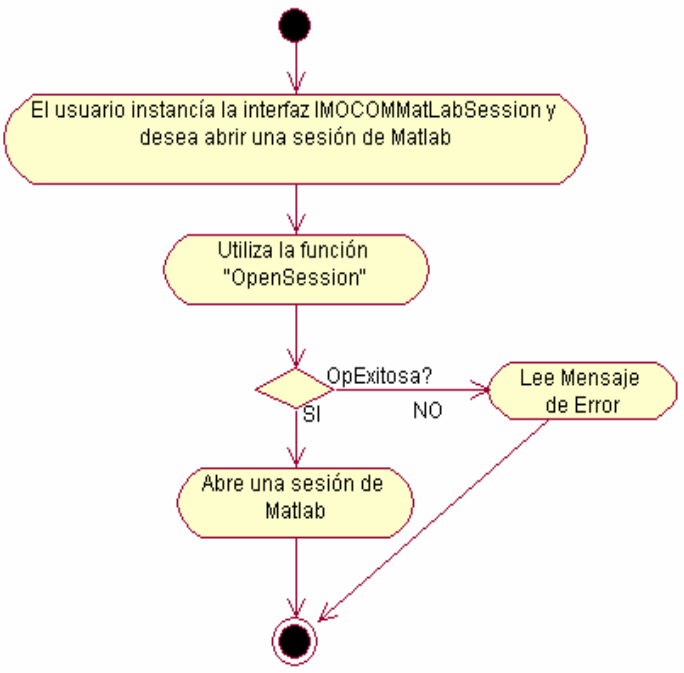


Diagrama de secuencia

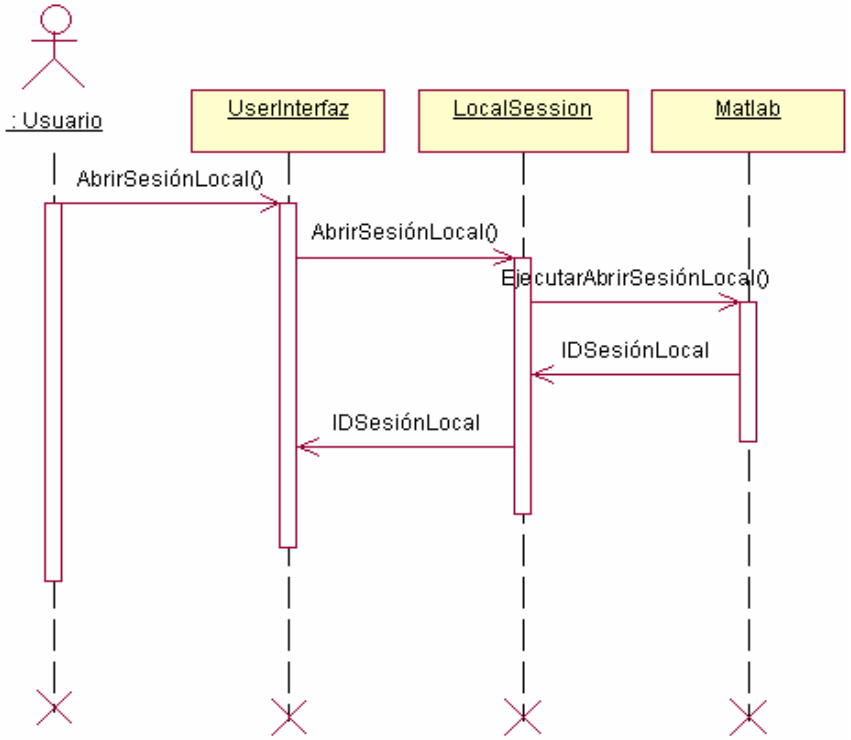
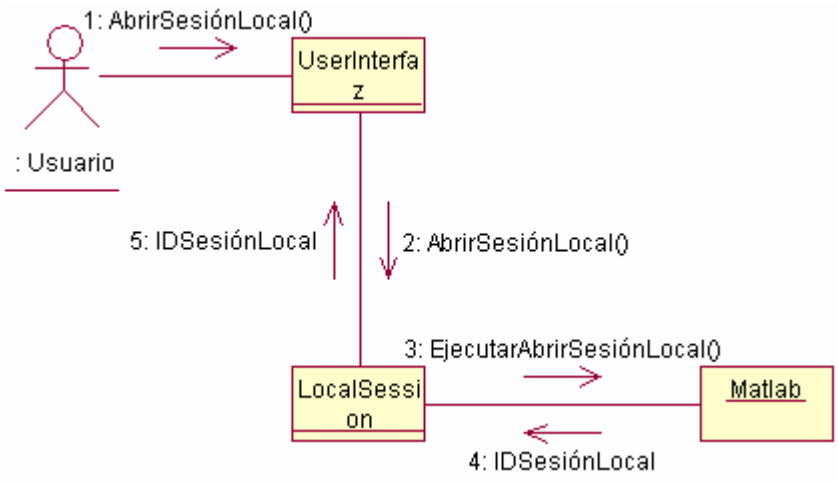


Diagrama de colaboración



Especificación de Caso de Uso: Gestionar IMOCOMMatLabVar
Versión 1.2
Escenario: Asignarle una sesión de Matlab a una variable.

Objetivo

Gestionar la interfaz IMOCOMMatLabVar para asignarle una sesión de Matlab a una variable.

Breve descripción

El usuario podrá gestionar la interfaz IMOCOMMatLabVar para asignarle una sesión o entorno de trabajo de Matlab a una variable definida.

Actores

Usuario

Flujo de eventos

Flujo básico

Actor	Sistema
Este caso de uso comienza cuando el usuario tiene instanciada la interfaz IMOCOMMatLabVar, y decide asignarle una sesión existente a una variable creada.	
El usuario utiliza el objeto que contiene la variable creada y ejecuta la función "SetSessionOwner" con el parámetro de entrada, éste último indica la sesión que quiere asignarle a la variable que encapsule el objeto actual.	
	El sistema ejecuta la operación y asigna la sesión indicada de Matlab a la variable, pudiéndose ver en el entorno de trabajo de Matlab la existencia de dicha variable.
	El sistema devuelve un apuntador a la variable asignada para poder ser utilizada.
Finaliza el caso de uso.	

Excepción

Si no se puede instanciar la interfaz requerida, se notificará mediante un mensaje de error.
Si la operación no pudo ser realizada, se notificará mediante un mensaje de error.

Precondiciones

Tener abierta una sesión de Matlab.
Haber creado e instanciado la variable a utilizar.

Diagrama de actividades

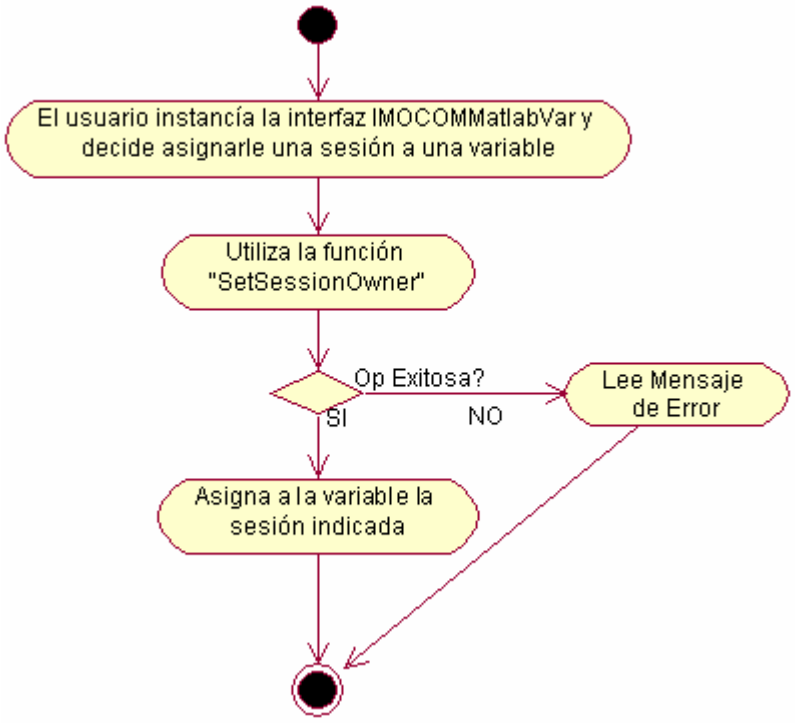


Diagrama de secuencia

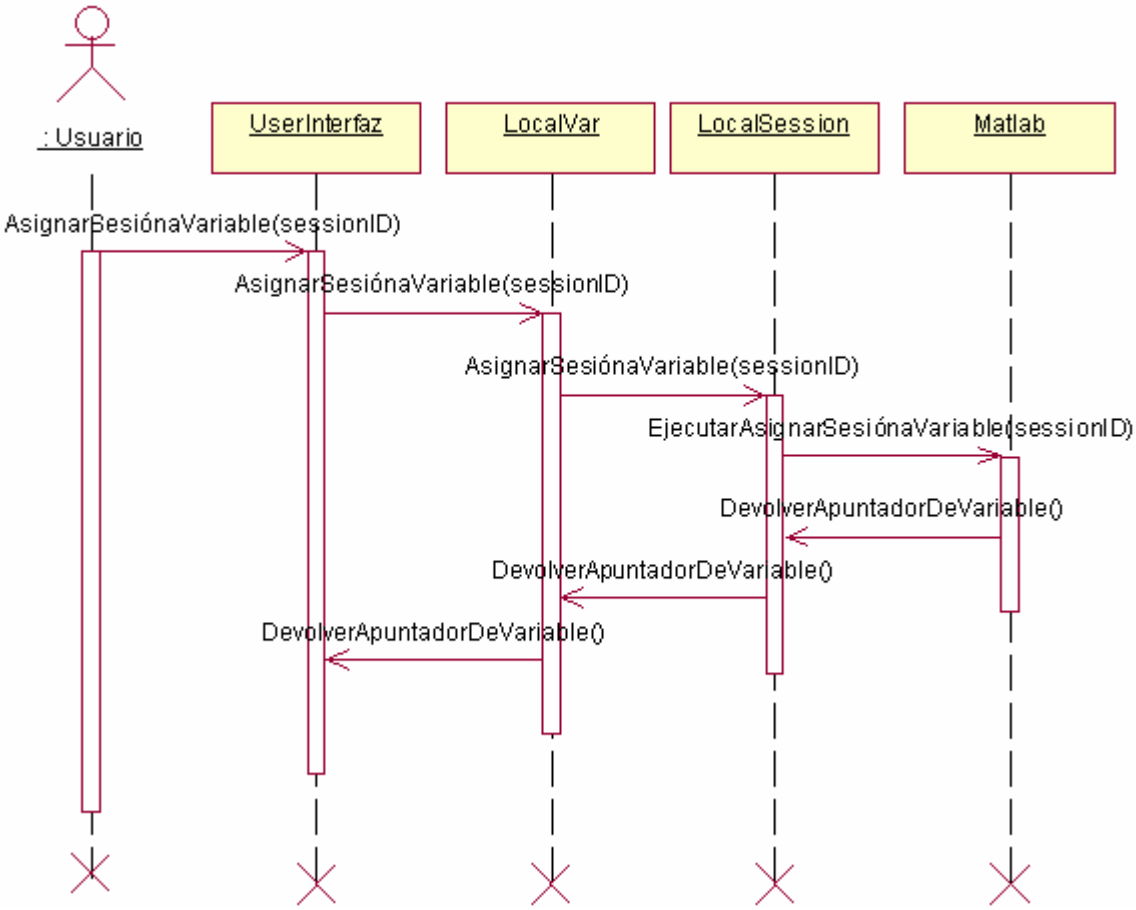
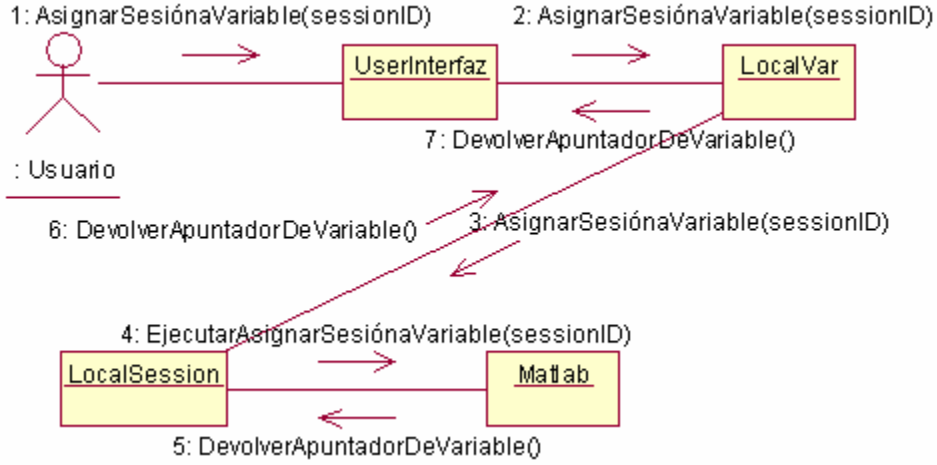


Diagrama de colaboración



Después de ver la manera en la que se diseña el sistema, en las secciones siguientes se mencionarán, de manera general, las fases restantes del diseño del sistema.

3.2.2.2. Diseño Arquitectónico

Con lo anterior, se puede decir que se utilizó como base el modelo arquitectónico de capas (ver Figura 3.5) y dentro de cada una de las capas se utilizó una arquitectura orientada a objetos (implementada con RUP).

3.2.2.3. Identificación de Objetos.

En este apartado se listan las clases encontradas dentro del diseño de los componentes de software. Esto posibilita darnos cuenta de cómo se conforma nuestro diseño.

- **LocalSesion.** Permite manipular y controlar las sesiones de Matlab, sobre las que trabaja el usuario de manera local.
- **LocalVar.** Lleva a cabo la manipulación de variables de cualquier tipo de Matlab, dentro de una sesión local.
- **LocalNumericArray.** Manipula variables de tipo matriz de valores numéricos dobles (reales o complejos) de Matlab, dentro de una sesión local.
- **LocalString.** Manipula variables de tipo cadenas de caracteres dentro de una sesión local.
- **LocalCellArray.** Permite manipular variables de tipo matrices de celdas de Matlab, dentro de una sesión local.
- **LocalStructArray.** Manipula variables de tipo matrices de registros de Matlab, dentro de una sesión local.
- **LocalFunctionCaller.** Encapsula una llamada a una función de Matlab, dentro de una sesión local.
- **LocalFigure.** Manipula una figura o ventana gráfica de Matlab, dentro de una sesión local.
- **LocalObject.** Encapsula una variable que sea un objeto dentro de una clase definida de en una ToolBox de Matlab para una sesión local.
- **LocalCharArray.** Permite manipular variables de tipo matriz de caracteres de Matlab, dentro de una sesión local.
- **RemoteSession.** Lleva un control de las sesiones de Matlab sobre las que trabaja el usuario de manera remota.
- **Token.** Permite el manejo de objetos remotos dentro de una sesión de Matlab remota.
- **RemoteVar.** Lleva a cabo la manipulación de variables de cualquier tipo de Matlab, dentro de una sesión remota.
- **RemoteNumericArray.** Manipula variables de tipo de matriz de valores numéricos dobles (reales o complejos) de Matlab, dentro de una sesión remota.
- **RemoteString.** Manipula variables de tipo cadenas de caracteres dentro de una sesión remota.
- **RemoteCellArray.** Manipula variables de tipo matrices de celdas de Matlab, dentro de una sesión remota

- **RemoteStructArray.** Manipula variables de tipo matrices de registros de Matlab, dentro de una sesión remota.
- **RemoteFunctionCaller.** Encapsula una llamada a una función de Matlab, dentro de una sesión remota.
- **RemoteFigure.** Manipula una figura o ventana gráfica de Matlab, dentro de una sesión remota.
- **RemoteObject.** Encapsula una variable que sea un objeto, dentro de una clase definida en una ToolBox de Matlab, para una sesión remota.
- **RemoteCharArray.** Manipula variables de tipo matriz de caracteres de Matlab dentro de una sesión remota.
- **DefineInterface.** Permite definir todos los métodos y propiedades generales de la librería de los componentes
- **Matlab.** Es provisto por Matlab.
- **UserInterfaz.** Establece la interfaz para los usuarios de los componentes

3.2.2.4. Modelos de Diseño

Para la implementación de este apartado se utilizó el Lenguaje Unificado de Modelado (UML) en base al Proceso Unificado Rational, esto con la ayuda de la herramienta Rational Rose para generar los modelos que rigen a los componentes: desde los casos de uso hasta los diferentes diagramas ilustrados en el apartado anterior.

4. Análisis y diseño de la aplicación de software

Dentro de este proyecto de tesis, además del desarrollo de los componentes de software, se implementará una pequeña aplicación titulada “Solucionador de sistemas de ecuaciones” con el fin de poner en práctica tales componentes. Esta aplicación hará uso de los componentes desarrollados en este proyecto así como de los componentes desarrollados en la tesis con título “Desarrollo de componentes software local y distribuido bajo la plataforma COM-SOAP, que encapsule las operaciones matriciales de Matlab” con esto se demostrará la integración de nuevos componentes software.

4.1. Análisis del sistema.

Cualquier aplicación de software debe estar basada en un correcto y funcional análisis para su buen desempeño. El análisis de la aplicación se llevará a cabo de la misma manera que el de los componentes de software, es decir, en el punto 4.1.1 se presentará el Documento de Requisitos del Usuario (DRU) y en el punto 4.1.2 la Especificación de Requerimientos de Software (ERS), del estándar “IEEE Recommended Practice for Software Requirements Specification ANSI/IEEE 830 1998”.

4.1.1. Documento de requisitos de usuario.

1. Introducción.

En este documento se presenta un trabajo de recopilación de requisitos previo en cuanto a la aplicación que se desea desarrollar.

2. Aplicación de prueba “Solucionador de sistemas de ecuaciones”

Se construirá un sistema de software que resuelva sistemas de ecuaciones, el cual permita ver de manera clara la importancia de los componentes de software.

2.1 Requisitos

Mediante la implementación del presente sistema de software se pretende mostrar la funcionalidad y utilidad de los componentes, local y remoto, desarrollados a los largo este

proyecto de tesis. De tal manera, que se vea en su total aplicación la funcionalidad y el objetivo que pretenden cubrir los componentes de software, así como el campo al que estarían orientados para un mejor desempeño. Para tal efecto la aplicación brindará lo siguiente:

1. Crear un sistema de software en el que incluya el par de componentes, el local y el distribuido.
2. El sistema debe de tener la misma funcionalidad tanto para la aplicación local como para la aplicación distribuida.
3. El sistema debe ser capaz de solucionar sistemas de ecuaciones por medio de dos métodos comunes conocidos en las áreas de matemáticas: método de la inversa y método por regla de Cramer.
4. El sistema debe soportar la solución de tantos sistemas de ecuaciones como se deseen, teniendo en cuenta que el tiempo de solución puede aumentar dependiendo de dicho número de ecuaciones que se encuentren creadas al momento de ver la solución de los mismos.
5. El sistema debe tener una interfaz amigable, esto debido a que se orienta al desarrollo de software educativo, sin embargo, podrá ser utilizado por cualquier usuario que así lo requiera.

4.1.2. Especificación de Requerimientos de Software (ERS)

Introducción.

A lo largo de este apartado se detallará la especificación de los requerimientos que el sistema “Solucionador de Sistemas de Ecuaciones” tiene.

1. Propósito.

Este apartado tiene como finalidad delimitar de manera clara el dominio del sistema para poder conocer las funcionalidades y restricciones del mismo. Es también, la base para el diseño del sistema solucionador de sistemas de ecuaciones a implementar, así como para permitir a los revisores de tesis observar de manera clara la implementación y uso correcto y funcional tanto de la aplicación como de los componentes desarrollados.

2. Ámbito del Sistema.

El desarrollo del sistema nace debido a la necesidad de contar con una aplicación que sea capaz de corroborar el funcionamiento de los componentes software desarrollados, de tal manera, que tanto usuarios como desarrolladores puedan ver el impacto de los mismos en diferentes áreas a las que deseen aplicarlos en base a la funcionalidad de los mismos.

El sistema tiene como función principal obtener la solución de sistemas de ecuaciones mediante dos métodos, de tal manera que puede ser utilizado por alumnos y profesores de la UTM, así como de otro nivel que sea de su interés. El sistema puede agilizar éste proceso y ofrecer una solución didáctica que sirva de aprendizaje a los usuarios.

3. Definiciones, Acrónimos y Abreviaturas

3.1. Definiciones

Tabla 4.1. Definiciones

Usuarios	Personas que harán uso del sistema *Solucionador de Sistemas de Ecuaciones*, mediante el cual podrán crear y resolver sistemas de ecuaciones de NxN incógnitas.

3.2. Acrónimos

Tabla 4.2. Acrónimos

ERS	Especificación de Requerimientos de Software
DRU	Documento de Requisitos de Usuario

3.3. Abreviaturas

Tabla 4.3. Abreviaturas

COM	*Component Object Model* modelo de Microsoft que define cómo crear dichos componentes y cómo construir aplicaciones con ellos.
SOAP	*Simple Object Access Protocol* es un protocolo que se utiliza para el intercambio de mensajes en sistemas de cómputo distribuidos.
MatLab	*Matrix Laboratory,* es una herramienta de cálculo numérico orientado al ámbito de la ingeniería e investigación. Creado por la empresa *The MathWorks* y usado en diferentes Universidades e institutos de investigación alrededor del mundo.
UTM	*Universidad Tecnológica de la Mixteca*

4. Referencias del Documento

- IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE std. 830, 1998
- Sommerville, Ian. Ingeniería de software. 6a. edición. Addison Wesley

Descripción General

En este apartado se presenta una descripción de alto nivel de la aplicación de software, las funciones principales a realizar, la información utilizada, así como las limitaciones y restricciones que afecten el desarrollo.

5. Funciones de la Aplicación de Software

El sistema solucionador de ecuaciones contempla las siguientes funciones:

- Crear sistema de ecuaciones.
- Modificar sistema de ecuaciones.
- Resolver sistema de ecuaciones
- Conexión local.
- Conexión remota.

Cada una de estas funciones se detalla en los siguientes puntos para una mejor comprensión.

5.1. Crear Sistema de Ecuaciones

En lo que respecta a esta función el usuario tendrá la posibilidad de crear el sistema de ecuaciones que desea resolver. De ésta manera, se pueden introducir los valores para cada una de las variables del sistema de ecuaciones, así como el valor constante resultante para cada una de ellos. Los sistemas de ecuaciones introducidos son automáticamente cuadrados, esto debido a que los métodos utilizados para la solución se basan en el uso de operaciones matriciales y para esto, la matriz debe ser cuadrada. En un futuro se considera la posibilidad de expandir esta aplicación para la solución de cualquier tipo de sistema mediante métodos que lo soporten.

5.2. Modificar Sistema de Ecuaciones

Esta operación permite al usuario hacer modificaciones sobre un sistema de ecuaciones introducido y almacenado con anterioridad, esto con la finalidad de que pueda cambiar algún valor debido a un error al momento de introducirlo o simplemente porque desea hacerlo.

5.3. Resolver Sistema de Ecuaciones

Uno de los puntos más importantes dentro de la aplicación es la solución del sistema de ecuaciones. En lo que a esto respecta, la solución del sistema introducido se hará mediante el uso del componente para la manipulación de matrices realizado en la tesis antes citada. Para esto, el usuario tendrá la facilidad de elegir entre dos tipos de operaciones para la solución del sistema. Estas son [39] :

- Resolver sistema por el método de la Inversa. Permite al usuario obtener una solución mediante el uso de la función inversa de una matriz, la cual es formada por el sistema de ecuaciones.
- Resolver sistema por medio de Regla de Cramer: Permite al usuario obtener una solución mediante el uso de la regla de Cramer (o método del determinante) que involucra el uso de matrices y una serie de pasos importantes para obtener el resultado.

5.4. Conexión Local

El sistema tendrá la opción de elegir el uso de una conexión local con Matlab, esto quiere decir, una conexión con Matlab dentro de la misma computadora en la que se encuentre la aplicación, para esto debe tenerse instalado Matlab en la computadora actual. Lo anterior permite corroborar la funcionalidad del componente COM desarrollado.

5.5. Conexión Remota

A su vez, el sistema ofrecerá la opción de hacer uso de una conexión remota con Matlab, es decir, una conexión a otra máquina, externa a la aplicación, en esta última es donde tiene que estar instalado Matlab. Esta operación hace posible verificar el funcionamiento del componente SOAP desarrollado.

6. Características de los usuarios

Los usuarios del sistema son todas aquellas personas que requieran resolver sistemas de ecuaciones de una manera automatizada y sencilla.

El sistema ofrece una interfaz amigable e intuitiva que facilita al usuario inexperto el uso de la aplicación.

7. Restricciones

La aplicación únicamente obtendrá la solución de sistemas de ecuaciones mediante los métodos de la inversa y regla de Cramer.

Otra restricción importante es que el sistema estará orientado únicamente a la solución de sistemas de ecuaciones cuadrados. Los sistemas que no cumplan con ésta característica no podrán ser resueltos.

8. Suposiciones y dependencias

8.1. Suposiciones

Se da por hecho que los requisitos descritos en este documento son aceptables una vez que este proyecto ha sido aprobado como tema de tesis. Cualquier petición de cambio en las especificaciones del proyecto serán consideradas mientras no afecten el curso total o parcial del mismo, de lo contrario serán propuestas como desarrollos futuros.

8.2. Dependencias

Los componentes desarrollados en este proyecto son totalmente dependientes de Matlab. Para el componente SOAP, se anexa una nueva dependencia que consiste en la necesidad de tener un equipo de cómputo que ofrezca los servicios de Matlab y que se encuentre conectado a una red.

Requisitos específicos

En esta sección se presentan los requisitos funcionales que deberá cumplir el sistema de software. Todos los requisitos aquí citados son esenciales, esto quiere decir que no es aceptable que el proyecto aquí desarrollado no cumpla con alguno de los requisitos presentados.

9. Requisitos funcionales

9.1. Conexiones locales y remotas

Req (1) El usuario contará con la posibilidad de crear conexiones locales del sistema con Matlab.

Req 1.1 El usuario selecciona la opción de crear una conexión local.

Req 1.2 El sistema establece la conexión local.

Req 1.3 El sistema mandará un mensaje de error en caso de que el intento de crear la conexión local sea fallido.

Req (2) El usuario también contará con la posibilidad de crear conexiones remotas del sistema con Matlab.

Req 2.1 El usuario selecciona la opción de realizar una conexión remota.

Req 2.2 Específica el nombre o la dirección IP del servidor de servicios de Matlab con el que desea realizar la conexión y elige realizar la conexión.

Req 2.3 El sistema establece la conexión remota.

Req 2.4 El sistema mandará un mensaje de error en caso de que el intento de crear la conexión remota sea fallido.

9.2. Creación de sistemas de ecuaciones

Req (3) El usuario podrá crear sistemas de ecuaciones con las características ya citadas, de tal manera que pueda definir el número de variables para el sistema de ecuaciones, así como el nombre que lo va a identificar.

Req 3.1 El usuario selecciona la opción de crear un sistema de ecuaciones.

Req 3.2 Introduce un nombre para el sistema de ecuaciones, así como un número de variables para dicho sistema. En caso de no introducir un nombre para el sistema de ecuaciones, se le asignará uno automáticamente. Si el nombre introducido por el usuario ya existe, el sistema mandará un mensaje de error.

Req 3.3 Se crea el sistema de ecuaciones y se añade a una lista en la que se encuentran almacenados todos y cada uno de los sistemas que se creen.

9.3. Modificación de sistemas de ecuaciones

Req (4) El usuario podrá modificar sistemas de ecuaciones existentes.

Req 4.1 El usuario selecciona la opción de modificar un sistema de ecuaciones.

Req 4.2 Selecciona de una lista de sistemas de ecuaciones el nombre del sistema que desea modificar y acepta la operación

Req 4.3 El sistema muestra en pantalla el sistema de ecuaciones que se desea modificar.

Req 4.4 El usuario introduce el o los valores que desea modificar dentro del sistema de ecuaciones seleccionado y acepta la operación.

Req 4.5 El sistema manda un mensaje de seguridad para verificar que la acción del usuario esté correcta. El usuario acepta o rechaza dicho mensaje.

Req 4.6 En caso de que el usuario haya aceptado el mensaje anterior el sistema realiza la operación, se almacena el sistema de ecuaciones con sus respectivos cambios y se manda un mensaje de confirmación que indica el éxito de la operación. El sistema actualizará la lista en la que se encuentran los sistemas de ecuaciones creados.

Req 4.7 Si el usuario rechaza el mensaje de seguridad sobre las modificaciones realizadas, el sistema de ecuaciones no sufrirá modificación alguna.

Req 4.8 En caso de que la operación de modificación no pueda realizarse por alguna circunstancia el sistema mostrará un mensaje de error.

9.4. Resolver el sistema de ecuaciones

Req (5) El usuario podrá resolver sistemas de ecuaciones mediante cualquiera de los métodos ya citados, así como elegir la forma en la que desea se le muestre la solución (por pasos o automáticamente).

Req 5.1 El usuario selecciona el sistema de ecuaciones que desea resolver de la lista de sistemas creados.

Req 5.2 El usuario selecciona la forma en la que desea ver los resultados. En caso de seleccionar la forma automática, el sistema mostrará la solución de manera secuencial sin la intervención del usuario. Si se selecciona la forma por pasos, el sistema mostrará la solución de manera secuencial con intervención del usuario.

Req 5.3 El usuario da clic derecho sobre el sistema de ecuaciones seleccionado y escoge el método de solución para dicho sistema.

Req 5.4 De acuerdo a lo anterior, el sistema realiza la solución del sistema de ecuaciones y muestra la solución tanto de manera gráfica como en forma de texto.

Req 5.5 En caso de que el sistema no pueda obtener la solución del sistema de ecuaciones indicado, se mostrará un mensaje de error.

9.5. Consulta de Sistemas de Ecuaciones creados

Req (6) El usuario podrá consultar todos los sistemas de ecuaciones que hayan sido generados en la sesión de trabajo actual de Matlab.

Req 6.1 El usuario selecciona el sistema de ecuaciones que desea consultar de la lista de sistemas generados.

Req 6.2 Si el sistema de ecuaciones generado no puede ser visualizado el sistema mandará un mensaje de error.

10. Interfaces Externos

10.1. Interfaces de usuario

La interfaz de usuario va a ser la comúnmente usada para aplicaciones de escritorio. El usuario se desplazará e interactuará con el sistema mediante uso del teclado y el ratón.

10.2. Interfaces de hardware

No han sido definidas.

10.3. Interfaces de Software

Las mismas que se han definido para cada unos de los componentes de software desarrollados y que al mismo tiempo están siendo usadas en el sistema.

10.4. Interfaces de comunicación

Es necesario contar con una conexión de red debido al uso del componente remoto, de ésta manera, se contará con una conexión Ethernet proporcionada por la red de la UTM en donde se localizará un equipo de cómputo conectado a esta red que funcione como servidor de los servicios de Matlab.

11. Requisitos de rendimiento

Debido a la estabilidad del lenguaje Matlab, se tiene previsto que no existan problemas en cuanto a la cantidad de conexiones locales y remotas que se realicen al entorno de programación.

12. Requisitos de desarrollo

El desarrollo de este sistema de software se basa en la tecnología orientada a objetos, siguiendo cada una de las guías que involucra el análisis, diseño e implementación de esta tecnología.

13. Requisitos tecnológicos

Para utilizar el sistema de manera local, es necesario contar con un equipo de cómputo que cuente con la configuración mínima siguiente:

Procesador: Pentium III 750 Mhz.

Memoria: 128 Mb.

Espacio libre en disco: 1 Gb. (para la instalación de Matlab).

Sistema Operativo: Microsoft Windows 98, Me, XP (recomendado).

Para utilizar el sistema de manera remota, es necesario contar con un equipo de cómputo que cuente con la configuración mínima siguiente:

Procesador: Pentium III 750 Mhz.

Memoria: 128 Mb.

Espacio libre en disco: 10 Mb.

Tarjeta Ethernet o Módem.

Acceso a Internet.

Sistema operativo: Windows 98, 2000, Windows XP

4.2. Diseño del sistema de software

En este apartado se trata a detalle lo que es el diseño del sistema. El diseño de cualquier sistema de software nace como consecuencia de un análisis previo de los requisitos funcionales y de usuario que requiere el estudio de la viabilidad de un proyecto. También, el diseño permite ver la manera en la que se va a llevar acabo el desarrollo del sistema de software.

El diseño utilizado para este sistema de software está basado en una arquitectura de capas, misma que se detalló en el apartado de diseño de los componentes de software y que se puede ver gráficamente en la Figura 3.4. Asimismo, el diseño será dividido en subcapas, las cuales nos permitan obtener servicios y procesos que van a ser parte primordial del sistema.

En la Figura 4.1, se puede ver a detalle, la vista lógica del modelo de capas para el sistema de software, de tal manera que se tenga una visión exacta de lo que se está haciendo.

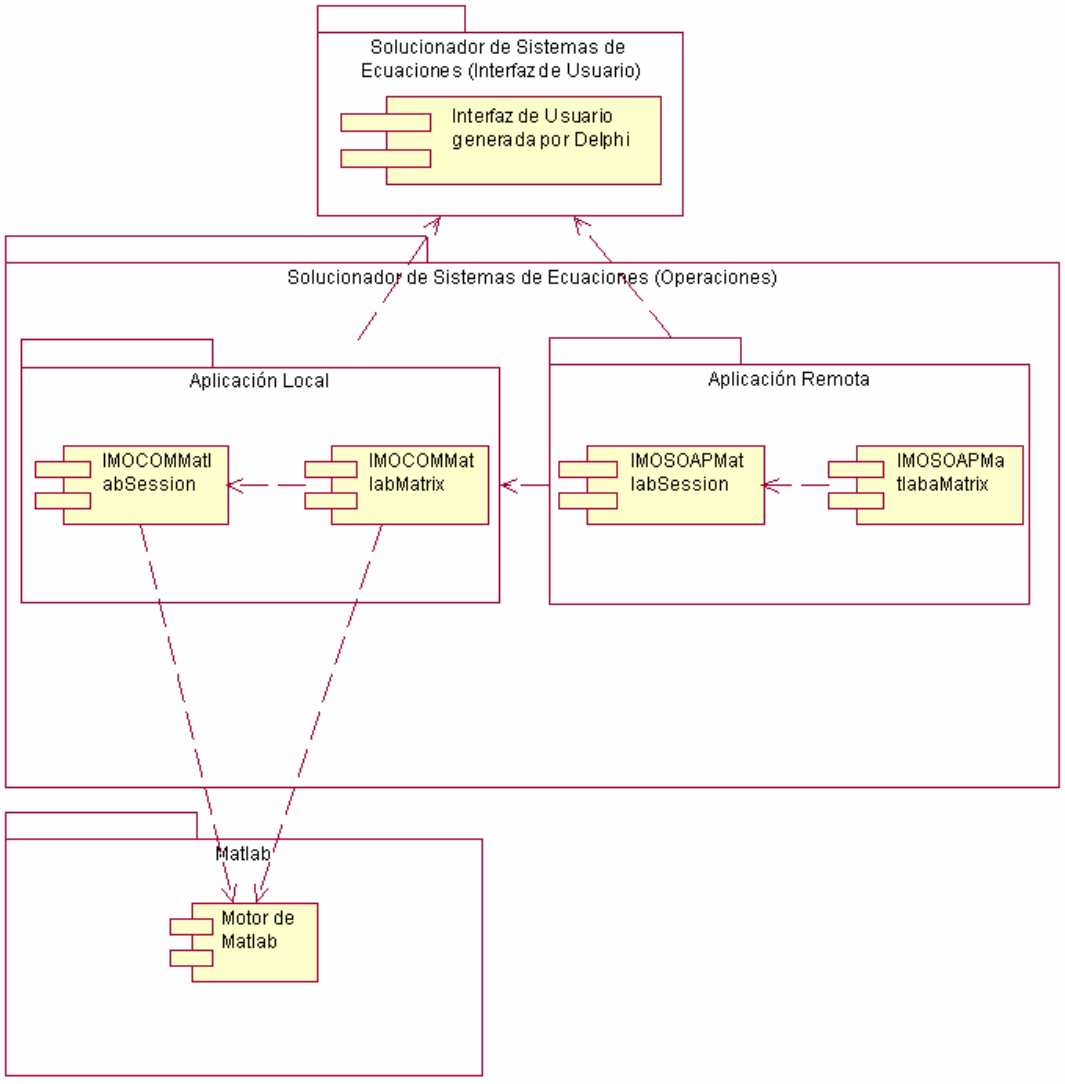


Figura 4.1. Diseño físico de Capas para el Solucionador de Sistemas de Ecuaciones

4.2.1. Diseño Orientado a Objetos

De la misma manera que los componentes de software, el diseño para este sistema se realiza en base al diseño orientado a objetos y mediante el uso de la notación del Lenguaje Unificado de Modelado (UML). De ésta manera, se seguirán las etapas de este modelo que son: Contexto y uso del sistema, Diseño Arquitectónico, Identificación de Objetos y Modelos de Diseño.

4.2.1.1. Contexto y casos de uso del sistema

En esta sección se muestra el modelo de paquetes que permite comprender de una mejor manera el contexto del sistema. Este modelo de paquetes ayuda a dividir el sistema de software en subsistemas más pequeños, de tal manera que se puedan repartir los procesos, tareas y responsabilidades. De manera gráfica, se muestra éste modelo en la Figura 4.2.

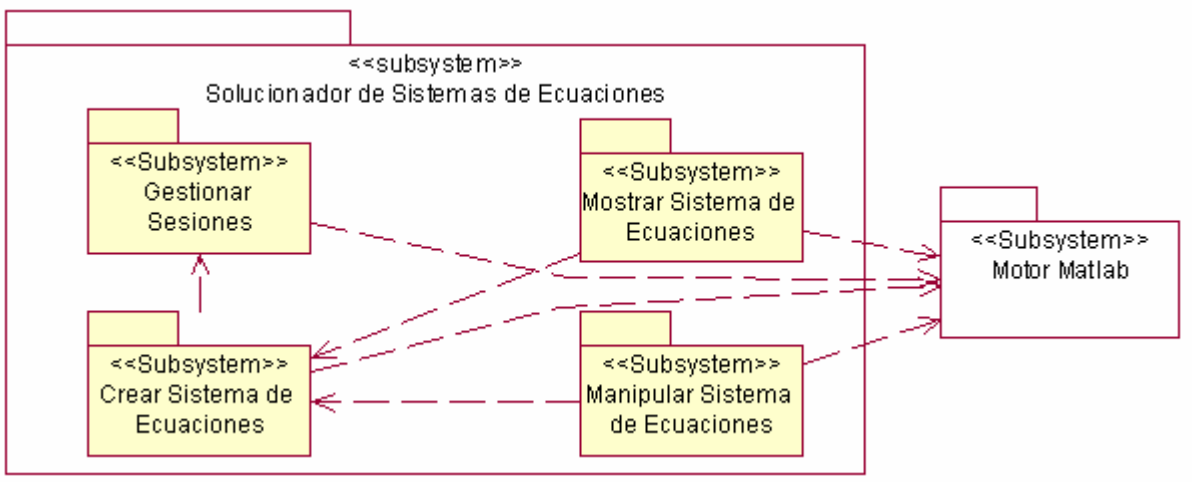


Figura 4.2. Contexto para el Solucionador de Sistemas de Ecuaciones (Diagrama de subsistemas)

En el modelo anterior se muestra la dependencia entre los diversos subsistemas que conforman el sistema, así como la dependencia de este con el entorno de Matlab.

El siguiente punto es poder ver la forma en que el sistema interactúa con su entorno. Para esto, se crean los casos de uso del sistema, mismos que se encuentran concentrados en un diagrama de casos de uso que se muestra en la Figura 4.3.

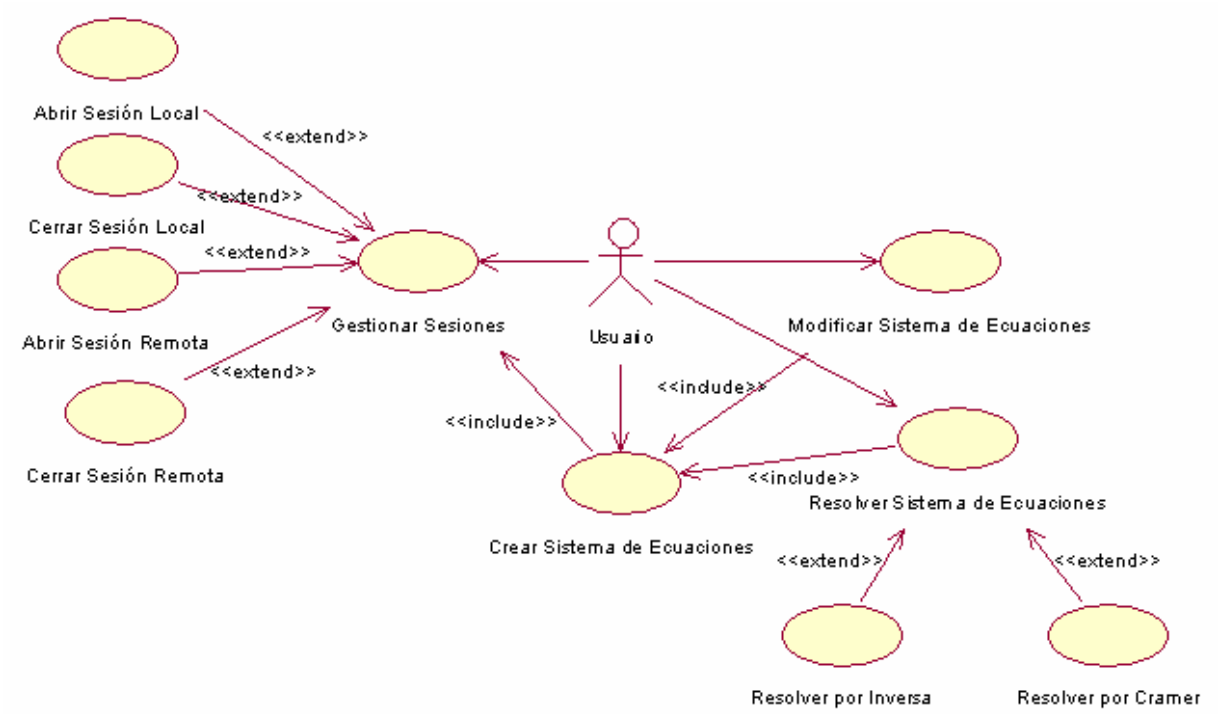


Figura 4.3. Diagrama de Casos de Uso para el Solucionador de Ecuaciones

4.2.1.1.1. Especificación de Casos de Uso (ECU)

A continuación se describen dos casos de uso para el sistema de software mediante el seguimiento de la tecnología RUP. La descripción de los casos de uso será de manera expandida para un mejor entendimiento de los procesos y requerimientos, así como ver de una manera detallada los mismos. El resto de los casos de uso se encuentran en la sección de Anexos.

Especificación de caso de uso: Crear Sistema de Ecuaciones. Versión 1.2

Historial de revisiones

Fecha	Versión	Descripción	Autor
1/Abril/2005	1.0	Creación del Documento	Josefina Flores Flores
18/Abril/2005	1.1	Introducción de primeros diagramas	Josefina Flores Flores
22/Abril/2005	1.2	Documento completado	Josefina Flores Flores

Objetivo

Crear un sistema de ecuaciones en la aplicación, de tal manera que el usuario pueda definir los valores del mismo.

Breve descripción

El usuario podrá crear sistemas de ecuaciones de tal manera que él sea quién defina de forma arbitraria los valores que dicho sistema contendrá para cada variable deseada, así como el número de variables y nombre del mismo; esto para que se cuente con un historial de los sistemas creados.

Actores

Usuario

Flujo de eventos

Flujo básico

Actor	***Sistema***
Este caso de uso comienza cuando el usuario ingresa al sistema y abre una sesión.	
	El sistema realiza el caso de uso Gestionar sesión.
	El sistema abre una ventana para trabajar sobre la sesión recién abierta.
El usuario elige la opción crear sistema de ecuaciones.	
	El sistema muestra una nueva pantalla y solicita la introducción del número de variables y el nombre para el sistema de ecuaciones a crear.
El usuario introduce el número de variables y el nombre del sistema de ecuaciones.	
	El sistema muestra una serie de campos los cuales deben ser llenados por el usuario y que representan los valores para cada una de las variables del sistema de ecuaciones que el usuario solicitó.
El usuario introduce los valores que desea contenga el sistema de ecuaciones.	
	Si no hay errores, el sistema crea el sistema de ecuaciones y manda el nombre del mismo a un historial de sistemas de ecuaciones creados.
Finaliza el caso de uso.	

Excepciones

Si la sesión no se puede abrir, se notificará al usuario mediante un mensaje de error.
Si el sistema de ecuaciones no puede crearse, se notificará al usuario mediante un mensaje de error.

Precondiciones

El usuario deberá haber ejecutado el sistema.

Diagrama de actividades

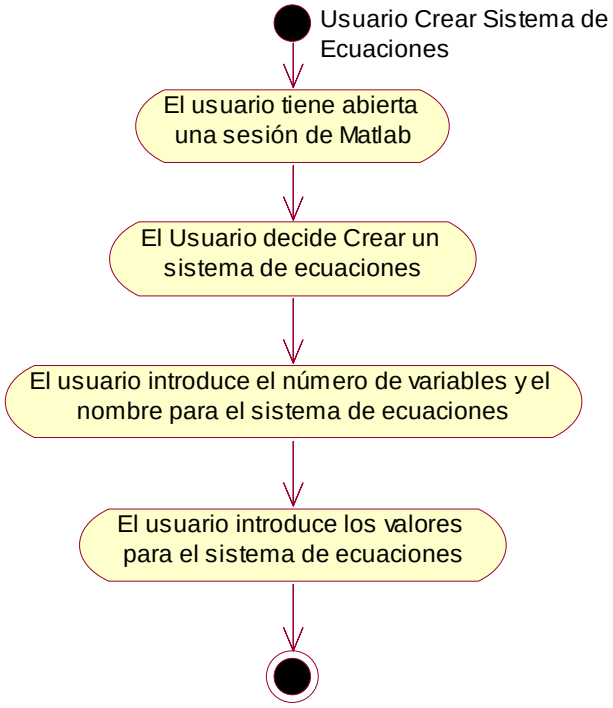


Diagrama de secuencia

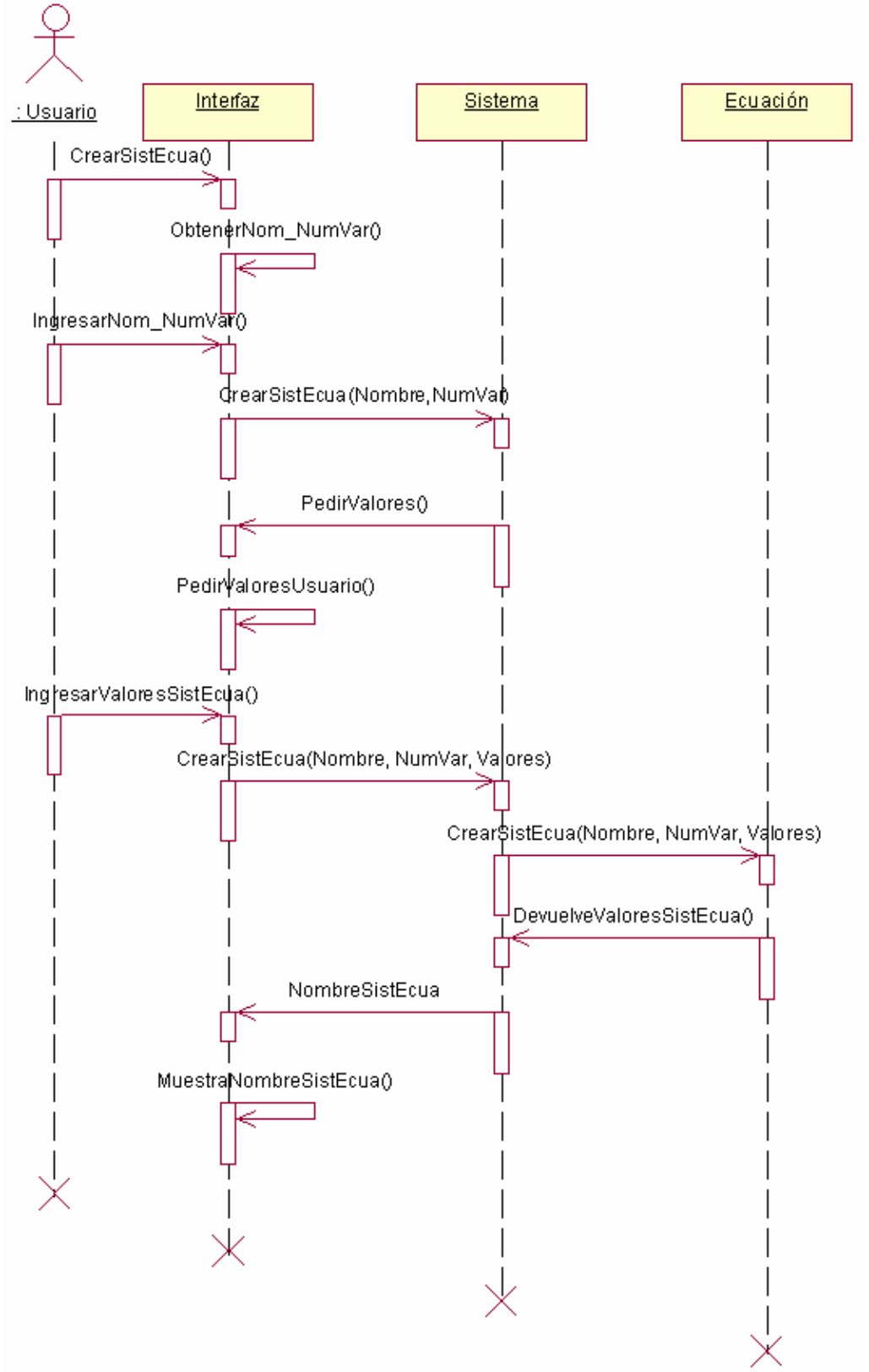
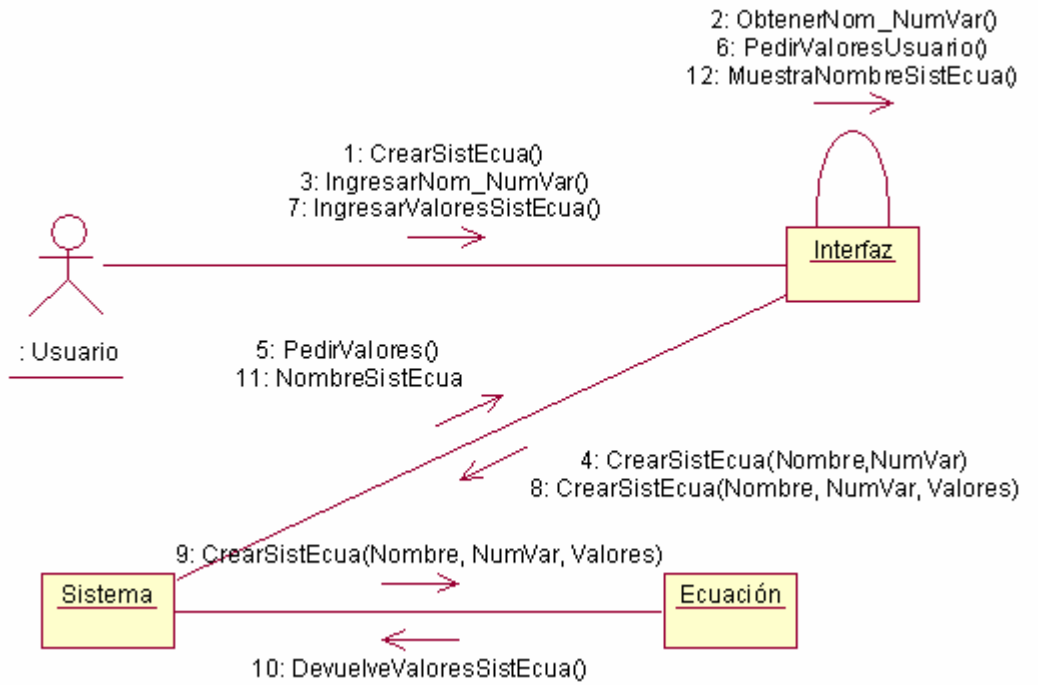


Diagrama de colaboración



Especificación de caso de uso: Modificar Sistema de Ecuaciones.

Versión 1.2

Historial de revisiones

Fecha	Versión	Descripción	Autor
1/Abril/2005	1.0	Creación del Documento	Josefina Flores Flores
18/Abril/2005	1.1	Introducción de primeros diagramas	Josefina Flores Flores
22/Abril/2005	1.2	Documento completado	Josefina Flores Flores

Objetivo

Realizar diversas operaciones sobre los sistemas de ecuaciones existentes.

Breve descripción

El usuario podrá manipular los sistemas de ecuaciones creados, de tal manera que pueda realizar operaciones sobre los mismos.

Actores

Usuario

Flujo de eventos

Flujo básico

Actor	***Sistema***
Este caso de uso inicia cuando el usuario ha ejecutado el sistema y ha creado al menos un sistema de ecuaciones.	
El usuario selecciona la opción Modificar Sistema de Ecuaciones.	
	El sistema muestra una ventana adicional que contiene los sistemas de ecuaciones existentes.
El usuario selecciona el sistema de ecuaciones que desea modificar.	
	El sistema despliega la información del sistema de ecuaciones elegido, permitiendo al usuario modificar los valores.
El usuario hace las modificaciones deseadas y acepta los cambios.	
	El sistema realiza la operación y guarda los cambios realizados.
Finaliza el caso de uso.	

Excepciones

Si la sesión no se puede abrir, se notificará al usuario mediante un mensaje de error.
Si el sistema no se pude crear se notificará al usuario mediante un mensaje de error.
Si la operación no pudo ser realizada se notificará al usuario mediante un mensaje de error.

Precondiciones

El usuario deberá haber ejecutado el sistema.

Diagrama de actividades

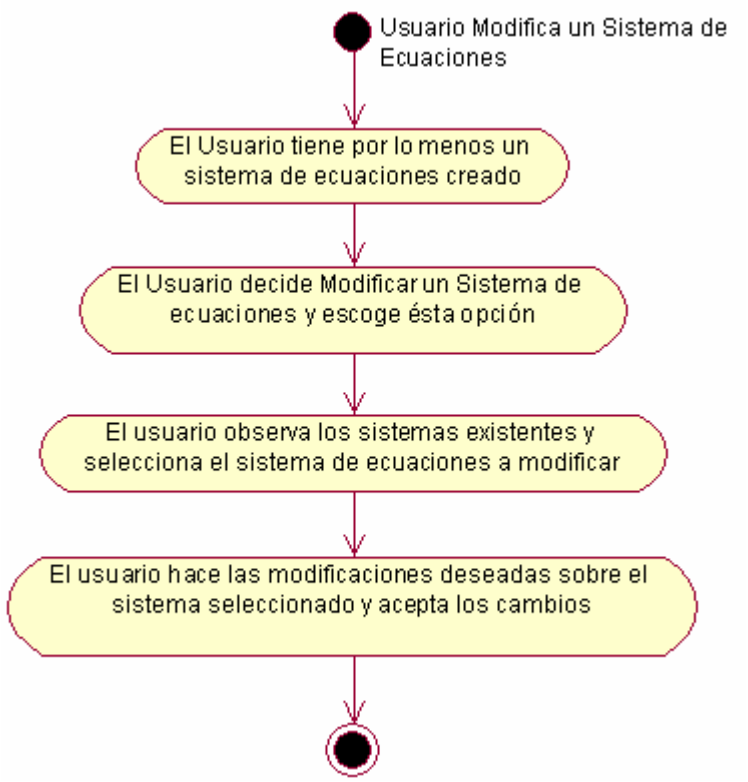


Diagrama de secuencia

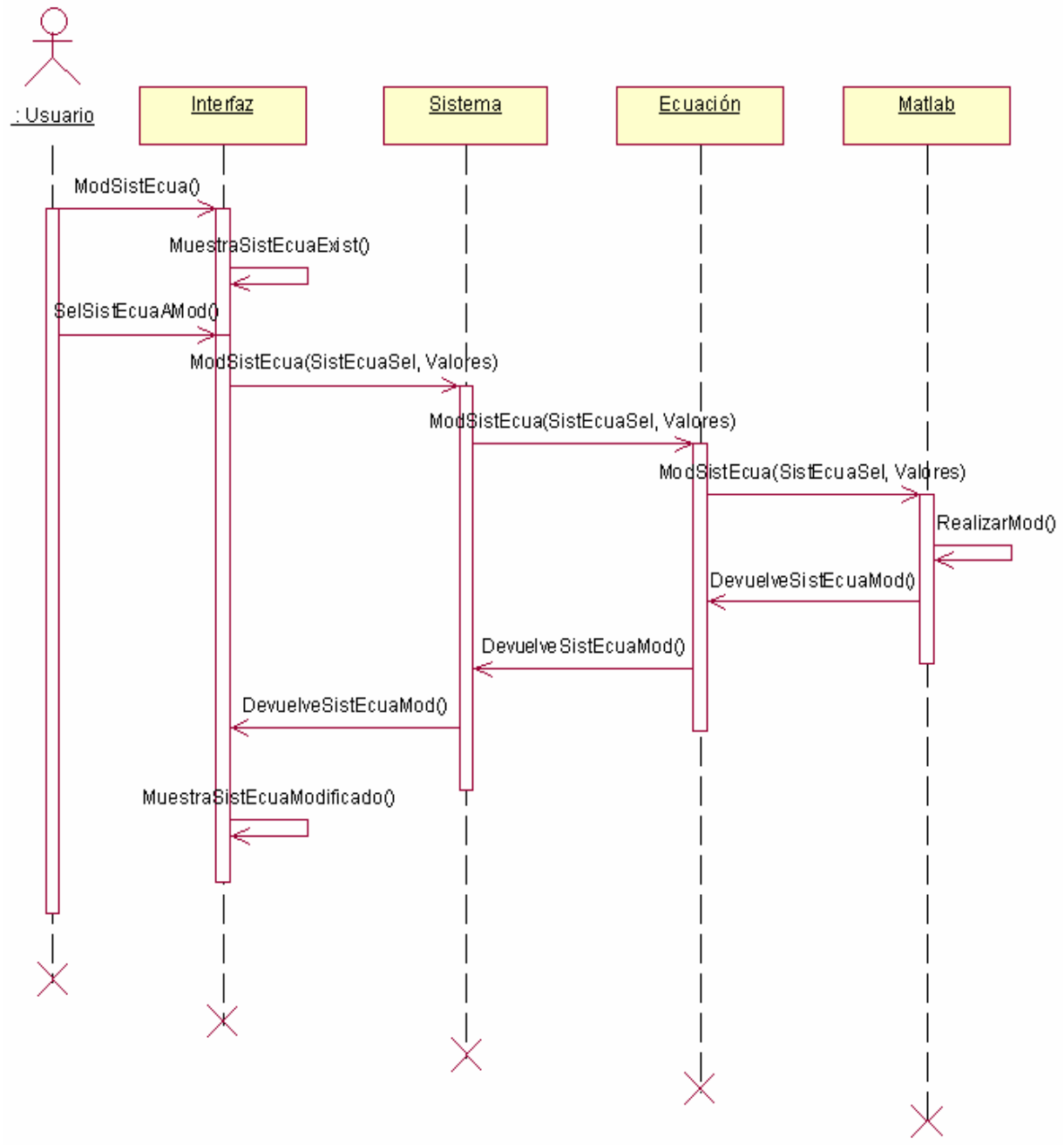
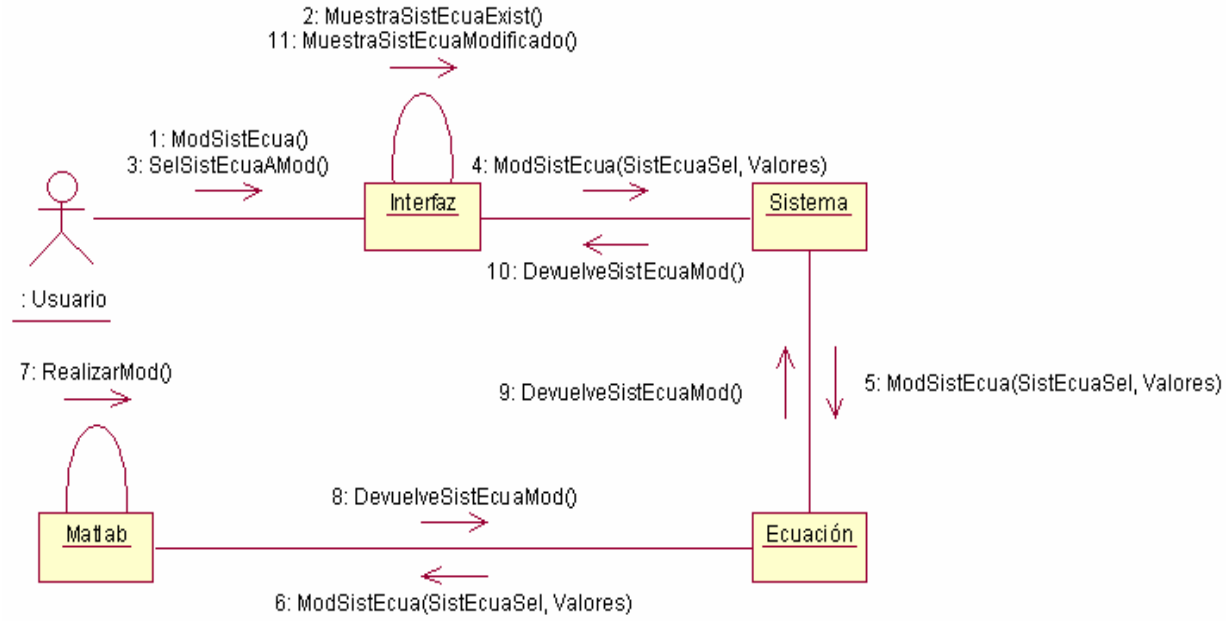


Diagrama de colaboración



Una vez visto la manera en la que se diseña el sistema, en las secciones siguientes se mencionará, de manera general, las fases restantes del diseño del sistema.

4.2.1.2. Diseño Arquitectónico

De la misma manera que en el diseño de los componentes de software y después de haber analizado todo lo anterior, podemos decir que para el diseño del sistema se utilizó como base el modelo arquitectónico de capas (ver Figura 4.1) y dentro de cada una de las capas se utilizó una arquitectura orientada a objetos (implementada con RUP).

4.2.1.3. Identificación de Objetos

En este apartado se definen los objetos responsables de una secuencia establecida en los procedimientos de diseño para cada caso de uso dentro del sistema y son los siguientes:

- **Interfaz.** Permite la representación del entorno de desarrollo en el cual va a ser presentado el sistema ante el usuario, para este punto y en el caso específico de éste sistema se tiene dependencia de las Formas manejadas por el lenguaje de programación Delphi en su versión 7.0 mismas que sirven como este tipo de objeto.
- **Ecuación.** Objeto que nos permite la creación de los sistemas de ecuaciones a resolver.
- **Operación.** Permite llevar a cabo alguna operación sobre el sistema de ecuaciones creado.
- **Sesión.** Se encarga del manejo de las sesiones locales y remotas de Matlab.

- **Sistema.** Es el encargo de proveer la interfaz y las operaciones necesarias para el manejo de los sistemas de ecuaciones creados.

4.2.1.4. Modelos de Diseño

Los modelos de diseño son parte esencial dentro del diseño de cualquier proyecto de software, esto debido a que muestran las clases y los objetos del sistema. Para la implementación de este apartado se utilizó el Lenguaje Unificado de Modelado (UML) en base al Proceso Unificado Rational, esto con la ayuda de la herramienta Rational Rose para generar los modelos que rigen al sistema de software: desde los casos de uso hasta los diferentes diagramas ilustrados en cada uno de los mismos.

5. Implementación y pruebas de los componentes

Al análisis y diseño de cualquier proyecto de software le sigue la descripción de la implementación, así como las pruebas realizadas para dicho proyecto. Por tal razón, en este apartado se describe de manera detallada la implementación y pruebas realizadas para los componentes de software desarrollados en este proyecto de tesis.

5.1. Implementación de los componentes de software

La implementación de los componentes de software COM y SOAP se llevó acabo mediante el uso del lenguaje de programación Object Pascal, teniendo como ambiente de desarrollo el lenguaje Delphi en su versión 7.0. Esto se detalló ampliamente en capítulos anteriores.

5.1.1. Implementación de las interfaces de software

En el capítulo de análisis y diseño de los componentes de software COM-SOAP se mencionó la forma en la que están estructurados dichos componentes, esto es, por interfaces. De aquí que cada componente cuenta con doce interfaces de software, donde a su vez cada interfaz contienen los métodos necesarios para el total y correcto funcionamiento de los componentes.

En éste apartado se detallará cada una de los métodos que envuelve cada interfaz de software, especificando los parámetros de entrada y salida.

La Tabla 5.1 muestra una lista detallada de los métodos que conforman la interfaz IMOMatlabSession.

Tabla 5.1. Métodos de la interfaz IMOMatlabSession

IMOMatlabSession			
Nombre del método	***Parámetros de entrada***	***Tipo de parámetro***	***Salida***
Clear	--	--	--
ClearVar	VarName	String	--
Close	--	--	--
CloseFigura	handle	Double	

IMOMatlabSession			
Nombre del método	***Parámetros de entrada***	***Tipo de parámetro***	***Salida***
Convert1xNArrayToVector	V1xNArray	Variant	Variant
ConvertClassIDToClassName	Classid	Mat_ClassID	String
ConvertClassNameToClassId	ClassName	String	Mat_ClassID
ConvertDoubleFlatArrayToString	doublearray	Variant	String
ConvertFlatArrayToNDims	flatarray	Variant	Variant
	ToDims	Variant	
CreateCellArray	--	--	IMOMatlabCellArray
CreateCharArray	--	--	IMOMatlabCharArray
CreateFigure	--	--	IMOMatlabFigure
CreateFunctionCaller	--	--	IMOMatlabFunctionCaller
CreateNumericArray	--	--	IMOMatlabNumericArray
CreateObject	--	--	IMOMatlabObject
CreateString	--	--	IMOMatlabString
CreateStructArray	--	--	IMOMatlabStructArray
CreateVar	--	--	IMOMatlabVar
Execute	Command	String	Mat_OPResult
ExecuteOrExcept	comando	String	--
	ErrorMessage	String	
	ErrorCode	Mat_ErrorCode	
FigureExists	handle	Double	Mat_Bool
GetActiveFigure	--	--	Double
GetClassIdOf	VarName	String	Mat_ClassID
GetClassOf	VarName	String	String
GetClearOnClose	--	--	Mat_Bool
GetDecimalSeparator	--	--	String
GetDetailedVarList	--	--	Integer
GetErrorStringPrefix	--	--	String
GetFigureList	--	--	Integer
GetFigureToClipboard	handle	Double	--
	format	Mat_FigureFormat	
GetIMOMatlabInterfaceObject	--	--	IMOMatlabInterface
GetInfinity	--	--	Double
GetInstaledToolboxes	--	--	Integer
GetLastCommandGotError	--	--	Mat_Bool
GetLastErrorMessage	--	--	String
GetLastFullAnswer	--	--	String

IMOMatlabSession			
Nombre del método	***Parámetros de entrada***	***Tipo de parámetro***	***Salida***
GetLastMatlabErrorOnException	--	--	--
GetLastWarningMessage	--	--	String
GetMatLabEngine	--	--	Variant
GetMatlabVersion	--	--	String
GetNotANumber	--	--	Double
GetRemoteCellArrayServiceName	--	--	String
GetRemoteCharArrayServiceName	--	--	String
GetRemoteFigureServiceName	--	--	String
GetRemoteFunctionCallerServiceName	--	--	String
GetRemoteNumericArrayServiceName	--	--	String
GetRemoteObjectServiceName	--	--	String
GetRemoteRootURL	--	--	String
GetRemoteServiceName	--	--	String
GetRemoteStringServiceName	--	--	String
GetRemoteStructArrayServiceName	--	--	String
GetRemoteVarServiceName	--	--	String
GetStringDelimiter	--	--	String
GetTextOfLastCommandGotError	--	--	String
GetTmpAsDouble	--	--	Double
GetTmpAsInteger	--	--	Integer
GetTmpAsString	--	--	String
GetTmpVar	--	--	String
GetVarAsCharArray	VarName	String	Mat_Bool
GetVarAsDouble	VarName	String	Double
GetVarAsDoubleArray	VarName	String	Mat_Bool
GetVarAsDoubleFlatArray	VarName	String	Mat_Bool
GetVarAsInteger	VarName	String	Integer
GetVarAsSerializedDoubleVector	VarName	String	Mat_Bool
GetVarAsString	VarName	String	String
GetVarDims	VarName	String	Variant
GetVarList	charDelimiter	Byte	String
	charQuotes	Byte	
GetVisible	--	--	Mat_Bool
GetWarningString	--	--	String
IsFinite	value	Double	Mat_Bool
IsInfinite	value	Double	Mat_Bool

IMOMatlabSession			
Nombre del método	***Parámetros de entrada***	***Tipo de parámetro***	***Salida***
IsLocalSession	--	--	Mat_Bool
IsNotANumber	value	Double	Mat_Bool
LastCommandGotWarning	--	--	Mat_Bool
LoadSession	filename	String	--
	options	String	
Mat_FigureFormatToString	format	Mat_FigureFormat	String
MatlabFloatToStr	value	Double	String
ObtainRandomName	--	--	String
OpenRemote	--	--	--
OpenSession	--	--	--
RaiseMatlabEngineException	mensaje	String	--
	ErrorCode	Mat_ErrorCode	
	CopyMatlabError	Mat_Bool	
SaveFigure	handle	Double	--
	filename	String	
	format	Mat_FigureFormat	
SaveSession	filename	String	--
	options	String	
SessionIsOpened	--	--	Mat_Bool
SetActiveFigure	handle	Double	--
SetClearOnClose	c	Mat_Bool	--
SetDecimalSeparator	dc	String	--
SetErrorStringPrefix	value	String	--
SetNewFigure	handle	Double	Double
SetRemoteCellArrayServiceName	Name	String	--
SetRemoteCharArrayServiceName	Name	String	--
SetRemoteFigureServiceName	Name	String	--
SetRemoteFunctionCallerServiceName	Name	String	--
SetRemoteNumericArrayServiceName	Name	String	--
SetRemoteObjectServiceName	Name	String	--
SetRemoteRootURL	Name	String	--
SetRemoteServiceName	Name	String	--
SetRemoteStringServiceName	Name	String	--
SetRemoteStructArrayServiceName	Name	String	--
SetRemoteVarServiceName	Name	String	--
SetShowMatlabCommandOnException	show	Mat_Bool	--

IMOMatlabSession			
Nombre del método	***Parámetros de entrada***	***Tipo de parámetro***	***Salida***
SetShowMatlabErrorOnException	show	Mat_Bool	--
SetStringDelimiter	sd	String	--
SetVarAsCharArray	VarName	String	--
	CharArray	Variant	
SetVarAsDouble	VarName	String	--
	value	Double	
SetVarAsDoubleArray	VarName	String	--
	RealR	Variant	
	ImagR	Variant	
	IsComplex	Mat_Bool	
SetVarAsDoubleFlatArray	VarName	String	--
	RealV	Variant	
	ImagV	Variant	
	IsComplex	Mat_Bool	
SetVarAsInteger	VarName	String	--
	value	Integer	
SetVarAsString	VarName	String	--
	value	String	
SetVisible	v	Mat_Bool	--
SetWarningString	wString	String	--
VarExists	Name	String	Mat_Bool

En el ANEXO 4[6] se encuentran los métodos para cada una de las interfaces que conforman a los componentes de software. Existe una tabla para cada interfaz en las que se detallan lo métodos de la misma.

5.2. Pruebas de los Componentes de Software

El paso siguiente a cualquier implementación de software es la etapa de pruebas, ésta hace posible verificar el correcto funcionamiento del mismo. En este apartado se detalla la forma en la que fueron probados los componentes de software, es importante mencionar que se hicieron pruebas de caja negra. Se probó cada uno de los métodos de tal manera que se pudo verificar en Matlab la creación y manipulación de datos y variables, verificando que arrojaran los resultados esperados.

[6] Por cuestiones de espacio, este anexo se encuentra en un documento separado de éste llamado anexos.pdf

5.2.1. Pruebas de caja negra

Las pruebas fueron realizadas sobre el lenguaje de software utilizado para el desarrollo de este proyecto de tesis, Delphi 7.0, teniendo como base el archivo de ayuda anexo a este documento. Este proceso de pruebas consiste en probar método a método los componentes de software dentro de un proyecto de software, compilar el proyecto y verificar que la operación deseada haya sido ejecutada correctamente y que los resultados se encuentren dentro del entorno de Matlab actual. Para esto, se realizaron pruebas de caja negra para todos y cada uno de los métodos de cada interfaz, sin embargo, por cuestión de espacio únicamente se mostrará algunos de ellos dentro de este documento.

Prueba No. 1: Prueba para la Interfaz IMOCOMMatlabSession

Objetivo: Abrir una sesión local de Matlab a través del uso de la interfaz IMOMatlabSession.

Procedimiento: Agregar los componentes de software al proyecto en el que se van a realizar las pruebas y ejecutar las instrucciones respectivas (Ver Tabla 5.2).

Resultados esperados: Apertura de una sesión local de Matlab (Ver Figura 5.1).

Tabla 5.2. Operaciones realizadas para abrir una sesión

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  sesion:=COIMOCOMMatlabSession.Create;
  sesion.OpenSession;
  ShowMessage('La sesión ha sido abierta');
end;
```

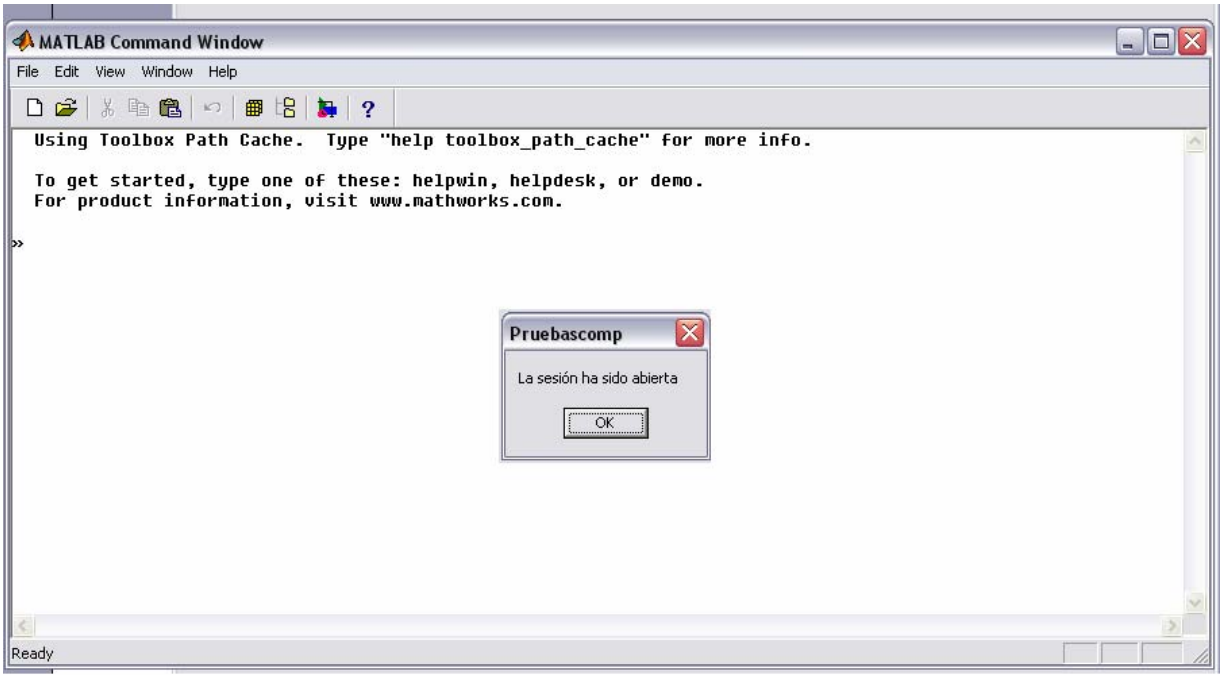


Figura 5.1. Resultados esperados: Apertura de la sesión de Matlab

Prueba No. 2: Prueba para la Interfaz IMOCOMNumericArray

Objetivo: Inicializar una variable de tipo matriz de N x N dimensiones.

Procedimiento: Agregar al proyecto los componentes de software desarrollados para el proceso de pruebas (Tabla 5.3).

Resultados esperados: Inicializar una matriz con los valores indicados dentro del entorno actual de Matlab (Ver Figura 5.3) y mostrar el nombre de la variable matriz creada (Ver Figura 5.2).

Tabla 5.3. Operaciones realizadas para inicializar una matriz de valores numéricos

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  sesion:=COIMOCOMMatlabSession.Create;
  sesion.OpenSession;
  matriz:=COIMOCOMMatlabNumericArray.Create;
  matriz.SetSessionOwner(sesion);
  matriz.InitXYMatrix(5,4);
  ShowMessage(matriz.GetVarName);
End;
```

Figura 5.2. Resultados esperados: Nombre de la matriz creada

Ahora bien, ya que se devolvió el nombre de la variable creada, dentro del entorno de Matlab actual, debe existir una matriz con el nombre mostrado e inicializada con las columnas y renglones indicados, llena de ceros. Esto se muestra en la Figura 5.3.

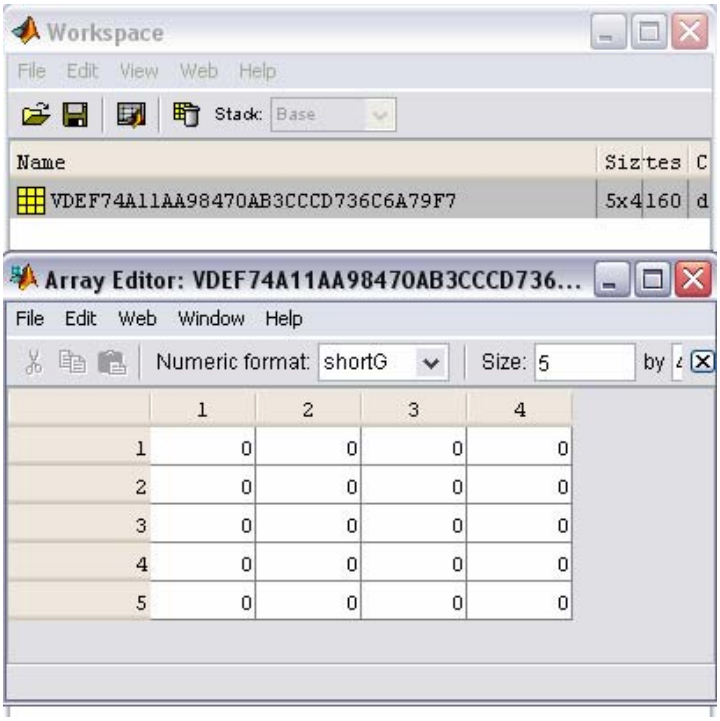


Figura 5.3. Resultados esperados: Matriz inicializada dentro del entorno de Matlab

Es de ésta manera como se realizaron las pruebas para cada uno de los métodos dentro de cada interfaz para los componentes de software (COM y SOAP). Sin embargo, como se menciona al inicio de éste apartado únicamente se describieron las dos pruebas anteriores por motivos de espacio en el documento.

6. Implementación y pruebas de la aplicación de software

Es momento ahora de dar paso a la implementación y pruebas de la aplicación de software. La cual tiene la finalidad de corroborar la funcionalidad de los componentes desarrollados. El desarrollo de esta aplicación fue llevado a cabo con la ayuda de los profesores de matemáticas del instituto de la Universidad, después de haber hecho un análisis previo a las necesidades, se optó por implementar un sistema que ayudara a la solución de sistemas de ecuaciones, que son muy utilizados en el transcurso del estudio de la licenciatura. Asimismo, la aplicación realizada tuvo su fase de análisis y diseño para poder ser realizada de una manera adecuada y funcional.

6.1. Implementación de la aplicación de software

Este apartado muestra la forma en la que se llevó a cabo la implementación de la aplicación. Es importante mencionar que la finalidad de la aplicación es mostrar la utilidad y funcionalidad de los componentes desarrollados en ésta tesis, los cuales, son la parte esencial de este proyecto. La aplicación se desarrolló en el lenguaje de programación Object Pascal, bajo el uso del ambiente de desarrollo Delphi en la versión 7.0.

6.1.1. Distribución física de la aplicación

La distribución física de la aplicación se refiere a la forma en la que se encuentran distribuidas las formas dentro de aplicación de software. La Figura 6.1 muestra de manera gráfica la manera en la que se hizo dicha distribución para las principales formas del sistema.

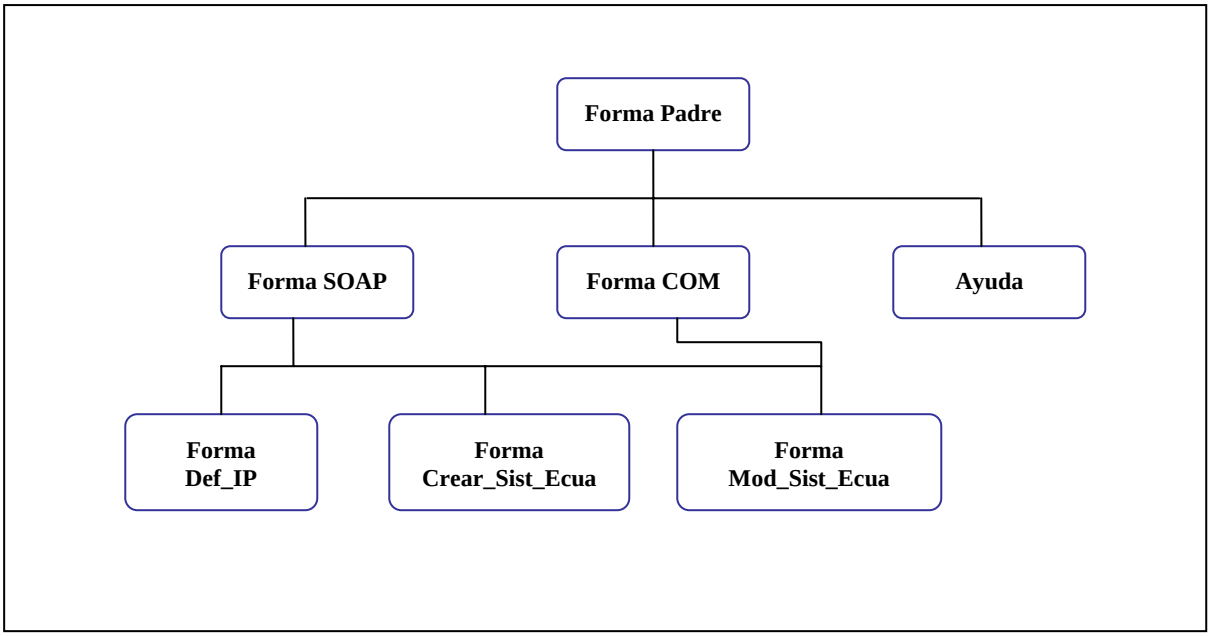


Figura 6.1. Distribución de las formas principales de la aplicación

A continuación se describen de manera detallada cada una de las formas que se muestran en la figura anterior.

Tabla 6.1. Descripción de las formas principales del sistema

Nombre de la forma	Descripción
Forma Padre	Forma que se presenta al usuario inmediatamente después de que el sistema ha sido cargado. Es la forma en la que se encuentran las operaciones principales del sistema: abrir y cerrar sesiones (locales y remotas), ayuda, salir, opciones que son la base para el funcionamiento del sistema.
Forma SOAP	Al momento en que el usuario elige abrir una sesión remota y después de haber indicado la dirección IP o dominio del servidor al que desea conectarse, se muestra la forma SOAP. Esta forma es utilizada por el usuario para crear, modificar y resolver sistemas de ecuaciones de forma remota en un servidor indicado anteriormente.
Forma COM	Al momento en que el usuario elige abrir una sesión local, se muestra la forma COM. Esta forma es utilizada por el usuario para crear, modificar y resolver sistemas de ecuaciones de forma local.
Ayuda	Esta forma va a mostrar la ayuda para todas las funciones del sistema.
Forma Def_IP	Cuando el usuario elige la opción abrir una sesión remota, se mostrará la forma Def_IP. En esta forma el usuario deberá

Nombre de la forma	Descripción
	introducir la dirección IP o dominio del servidor con el cuál se desea establecer la conexión remota y con el que el sistema va a trabajar.
Forma Crear_Sist_Ecua	Al momento en que el usuario decide crear un sistema de ecuaciones se muestra ésta forma. Dicha forma contiene dos campos que indican un nombre y una dimensión para el sistema de ecuaciones que se desea crear.
Forma Mod_Sist_Ecua	Si el usuario decide modificar un sistema de ecuaciones ya creado verá en pantalla ésta forma, que le muestra los sistemas que se encuentran creados. El usuario deberá seleccionar el sistema que desea modificar de tal manera que se mostrará el mismo en la forma COM para poder cambiar los valores existentes por los ahora deseados.

Existen también otras formas que se utilizan dentro del sistema. Las aquí mostradas son las de mayor importancia, motivo por el que se muestran detalladamente.

6.1.2. Diseño e implementación de las interfaces

Después de haber visto la distribución de las formas dentro del sistema, es necesario hacer un diseño de dichas formas de tal manera que el usuario pueda tener una visión fácil de entender y una idea clara del funcionamiento de la aplicación.

Es por lo anterior, que en éste apartado se muestra la manera en la que fueron implementadas las formas del sistema para poder crear interfaces gráficas para el usuario.

6.1.2.1. Pantalla de presentación

La primer pantalla que se muestra al iniciar el sistema es una pantalla de presentación (Véase Figura 6.2), que es mostrada mientras el sistema es cargado en su totalidad, al momento de que la carga del sistema se completa, la pantalla es cerrada automáticamente.

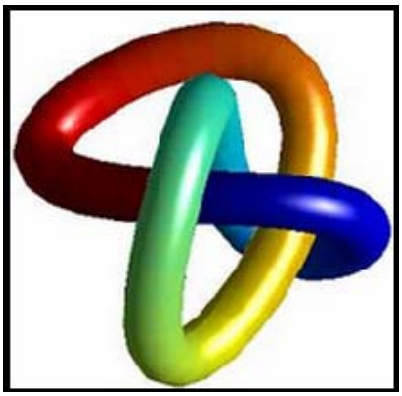


Figura 6.2. Pantalla de presentación

6.1.2.2. Pantalla principal

Al momento en que el sistema está cargado en su totalidad se muestra la pantalla principal (Véase Figura 6.3). Esta pantalla es la que se encarga de llevar el control para el manejo de la aplicación, es aquí en donde se pueden abrir y cerrar las sesiones locales y remotas, así como las opciones para salir de ellas y del sistema en general. También se cuenta con la opción de la ayuda en el menú principal así como una opción para cambiar la manera en la que se muestran las ventanas al momento de tener abierta más de una dentro de la aplicación.

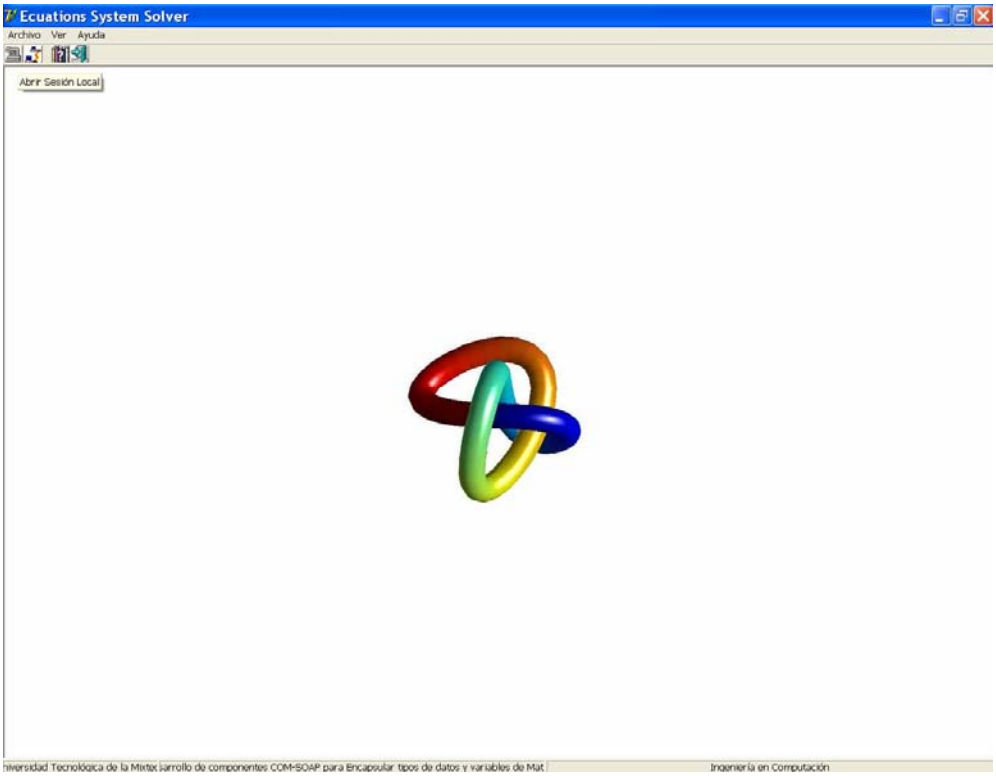


Figura 6.3. Pantalla Principal del Sistema

Esta pantalla contiene también una barra de botones que realizan las funciones mencionadas anteriormente pero de manera directa al momento de dar clic en el botón de la operación a realizar y que son las que encontramos en el menú Archivo.

El espacio central es utilizado para mostrar las ventanas secundarias de la aplicación y por lo tanto, es espacio de trabajo para el sistema.

6.1.2.3. Abrir Sesión Local o Remota

Se puede abrir una sesión local o remota, eligiendo la opción para cualquiera de las tareas en el menú archivo (Figura 6.3) o bien, dando clic en el botón dentro de la barra de botones que indique la operación que se desea realizar.

Supongamos que se desea abrir una sesión remota, en este caso después de ver la pantalla principal y de elegir ésta operación, se mostrará la ventana de la Figura 6.4, que indica al

usuario que debe insertar la dirección IP o nombre del servidor con el cual se desea conectar para el trabajo remoto.

Figura 6.4. Interfaz para definir dirección IP o nombre del servidor remoto

Esta ventana tendrá como valor preestablecido la dirección IP del servidor en el que se harán las pruebas remotas y en que se encuentran instalados los componentes desarrollados. Así como la instalación de los requerimientos para el funcionamiento de este proyecto (Matlab y componente SOAP). Al momento de dar clic en el botón “Aceptar” el sistema muestra otra interfaz la cual permitirá la creación y manipulación de las sesiones dentro del sistema. Esta interfaz se muestra en la Figura 6.5.

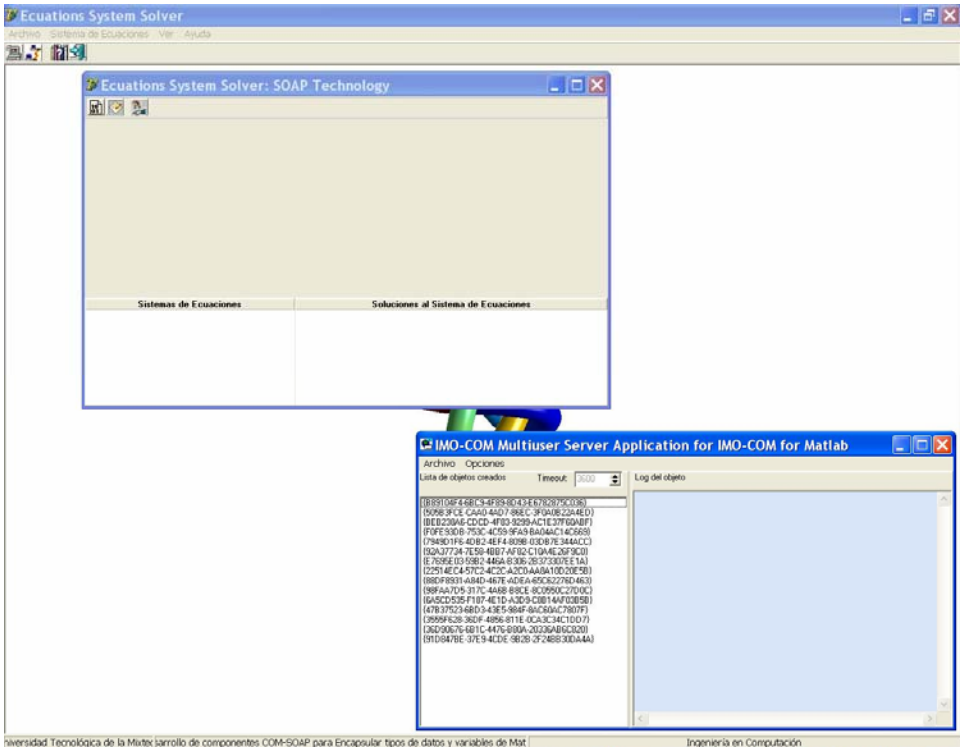


Figura 6.5. Interfaz abrir una sesión remota

Es importante mencionar que si se elige realizar una conexión local, el sistema no desplegará la Figura 6.4, si no que de inmediato lanzará una interfaz casi idéntica a la que se muestra en la Figura 6.5 con la variación del nombre de dicha interfaz y no lanzará la ventana auxiliar, ya que esta es única para el manejo de sesiones remotas.

En esta interfaz se muestran lo que son tres operaciones básicas, el crear un sistema de ecuaciones, modificar sistema de ecuaciones y cerrar la sesión, estas están representadas por botones al igual que se agrega un menú en la ventana principal, el cual contiene estas mismas operaciones. La interfaz se divide en tres partes:

- Se cuenta con dos paneles inferiores, el de la izquierda sirve para mostrar los sistemas de ecuaciones que se han creado, de tal manera que el usuario tenga acceso a ellos en el momento que lo desee. El panel inferior derecho, muestra la solución en forma de texto para el sistema de ecuaciones seleccionado, de tal manera que el usuario la pueda ver de manera simple y rápida.
- En el espacio superior restante, se muestran los campos que se deben llenar para cada sistema de ecuaciones y/o se muestra la solución de manera gráfica.

El sistema puede soportar la creación de tantas sesiones (locales y/o remotas) como el usuario desee, de tal manera que pueda tener varias sesiones abiertas al mismo tiempo en las que desee trabajar (Ver Figura 6.6). Las sesiones pueden ser cerradas en el momento que el usuario lo desee sin ningún problema.

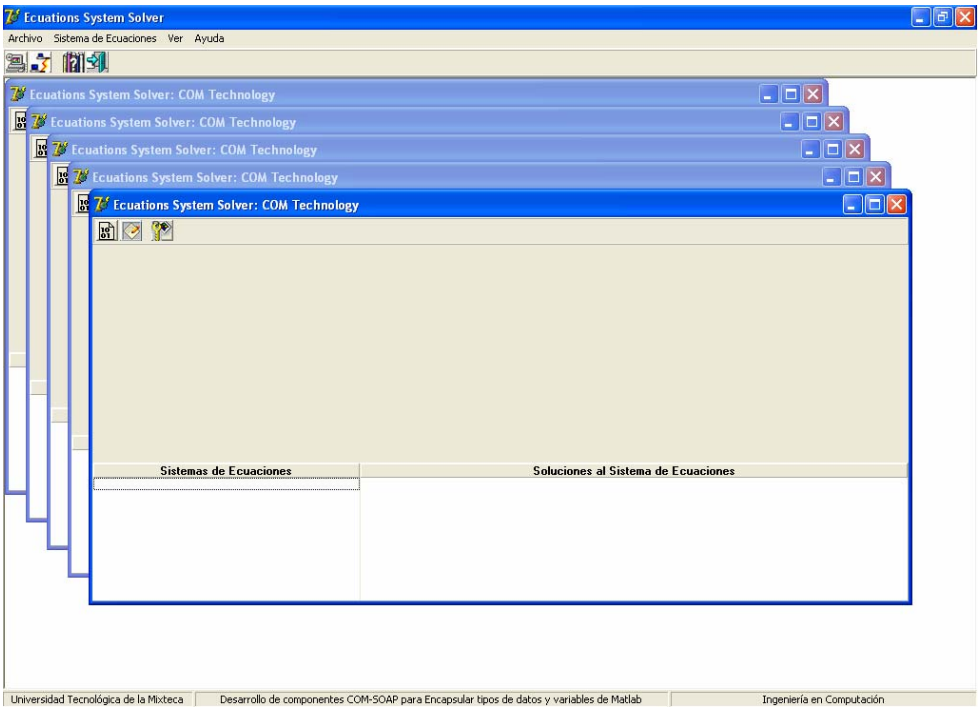


Figura 6.6. Múltiples ventanas abiertas

6.1.2.4. Creación de sistemas de ecuaciones

Para la creación de sistemas de ecuaciones, el usuario tendrá dos opciones: utilizando el menú Sistemas de Ecuaciones con la opción Crear Sistema de Ecuaciones, o dando clic en el primer botón que aparece en la ventana de la sesión. En el momento en que elige alguna de las dos opciones, aparece la siguiente interfaz:

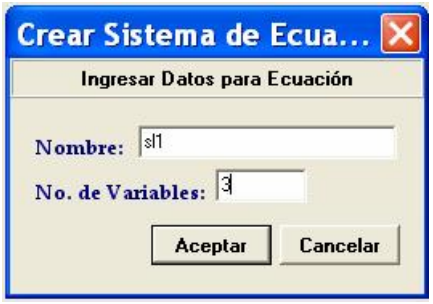


Figura 6.7. Interfaz Crear Sistema de Ecuaciones

La Figura 6.7 muestra una interfaz en la que el usuario debe definir un nombre para el sistema de ecuaciones que desea crear, así como un número de variables para el mismo. El sistema de ecuaciones debe ser cuadrado, por lo que únicamente se pide un valor para el número de variables, esto para que pueda ser resuelto por los métodos elegidos. Si el usuario no da un nombre para el sistema de ecuaciones, Matlab le asigna uno de manera aleatoria, mismo que es manejado por el sistema.

Posteriormente, el sistema muestra una ventana con una serie de campos vacíos a llenar (Ver Figura 6.8), estos campos deben ser llenados con los valores del sistema de ecuaciones. Del lado izquierdo al signo igual se tendrán los valores para cada una de las variables y, del lado contrario, el resultado para cada ecuación.

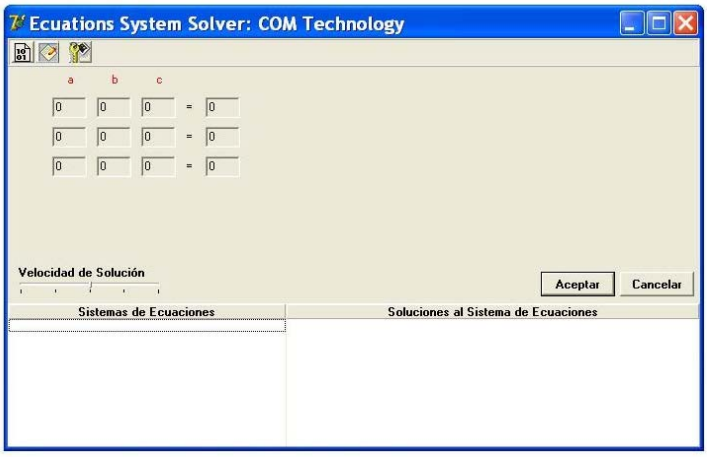


Figura 6.8. Llenar sistema de ecuaciones

Al momento de que el usuario presione clic en el botón de Aceptar, el sistema limpia el espacio en el que se encontraban los campos a llenar para el sistema de ecuaciones y manda el nombre de la variable creada al panel de “Sistemas de Ecuaciones” así como la variable se encuentra ahora alojada en Matlab. El sistema de ecuaciones se encuentra ahora creado y listo para ser manipulado.

6.1.2.5. Modificar Sistema de Ecuaciones

Cuando el usuario se ha equivocado, o desea modificar el valor o los valores de un sistema de ecuaciones creado anteriormente, podrá hacerlo sin ningún problema. Esta opción se encuentra dentro del menú Sistemas de Ecuaciones o directamente dando clic en el segundo botón de la barra de botones para la sesión abierta. En este momento se mostrará una ventana, la cual le listará al usuario los sistemas de ecuaciones que se encuentran creados en el sistema, y podrá modificar el que desee (Ver Figura 6.9).

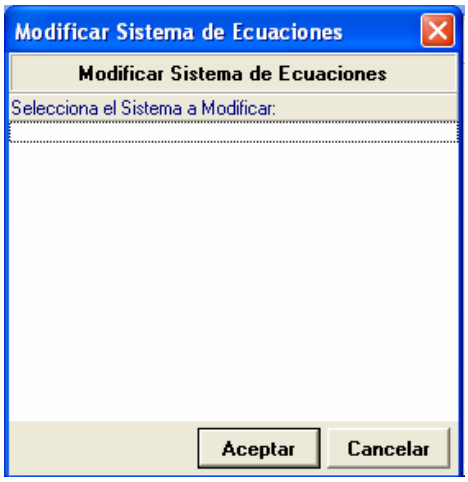


Figura 6.9. Modificar Sistema de Ecuaciones

Al momento de escoger el sistema de ecuaciones a modificar, éste será desplegado en la pantalla, permitiendo editar los campos de cada valor y cambiarlos si así lo requiere por los nuevos valores. Después, al dar clic en el botón "Modificar" los valores serán almacenados tanto en el sistema como en Matlab, permitiendo resolver y visualizarlo nuevamente en el momento que se desee.

6.1.2.6. Interfaces auxiliares

En este apartado se muestran las interfaces que sirven como avisos al usuario en el momento de realizar cualquier operación, estas interfaces tienen la finalidad de hacer que el usuario esté seguro de lo que va a realizar, así como para advertir al usuario de alguna confirmación; de haber introducido un dato erróneo o haber realizado una operación no válida.

A continuación se muestran algunas de esas interfaces auxiliares:

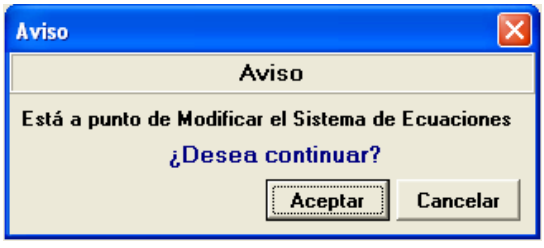


Figura 6.10. Aviso Modificar Ecuación

La Figura 6.10 muestra un aviso al momento de que el usuario va a hacer una modificación a algún sistema de ecuaciones, de tal manera que si no está seguro de la operación pueda cancelarla o, de lo contrario, realizarla si así lo desee confirmando dicha operación.

Cuando un error ocurre dentro del sistema por razones antes mencionadas, el sistema le indicará al usuario lo que ha sucedido.

Así mismo, cuando se realiza una operación correctamente, el sistema también lo notificará con un mensaje como el de la Figura 6.11.

Figura 6.11. Mensaje de Error del sistema

Este mensaje como puede verse, es cuando se ha modificado un sistema de ecuaciones correctamente.

6.2. Pruebas de la aplicación de software

Al haber terminado la implementación del sistema es necesario realizar una etapa de pruebas de manera que se pueda corroborar que la aplicación se encuentra funcionando de la manera esperada. En este apartado se muestran las pruebas realizadas a la aplicación de software, las cuales fueron realizadas de la misma manera que para los componentes de software.

Las pruebas realizadas son pruebas de caja negra, estas pruebas tienen la finalidad de mostrar que los resultados arrojados después de resolver el sistema indicado son persistentes comparados con los resultados dados para la solución del sistema de ecuaciones que se desea resolver.

6.2.1. Pruebas de caja negra

Para la realización de estas pruebas se tomó como referencia el manual de usuario que se encuentra como documento anexo a este trabajo de investigación. Las pruebas fueron realizadas a todas las operaciones que ofrece la aplicación, sin embargo, por cuestiones de espacio únicamente se muestran algunas de ellas.

Es importante mencionar que para este ejemplo y para otros, se constataron los resultados con la referencia [39]

Con el sistema de ecuaciones siguiente como ejemplo:

$$2x + 4y + 6z = 18$$
$$4x + 5y + 6z = 24$$
$$3x + y - 2z = 4$$

Sol. (4, -2, 3)

Prueba 1. Crear Sistema de Ecuaciones.

El usuario debe tener arrancado el sistema y debe haber abierto una sesión local o remota. Debe escoger la opción “Crear sistema de ecuaciones” (Figura 6.12).

Figura 6.12. Crear Sistema de Ecuaciones

Después de dar clic en la opción crear sistema de ecuaciones se lanzará una ventana nueva en la que se piden algunos datos para un sistema de ecuaciones, estos campos son: nombre del sistema y número de variables. Esto se muestra en la Figura 6.13. En este ejemplo el sistema se llama sl1 y tiene 3 variables, lo que quiere decir que se está creando un sistema de tres ecuaciones con tres incógnitas, que se va a resolver por cualquiera de los métodos dados por el sistema.

Figura 6.13. Datos para Sistema de ecuaciones

Una vez que el usuario ha decido la magnitud del sistema a crear, debe dar clic en el botón “Aceptar” y en ese momento se mostrarán unos tantos campos vacíos como el número

de ecuaciones y variables dados por el usuario, en este caso, serán 3 sistemas con 3 incógnitas, en total 9 campos vacíos para ser llenados y una columna de resultados para cada ecuación del sistema de ecuaciones, esto se muestra gráficamente en la Figura 6.14.

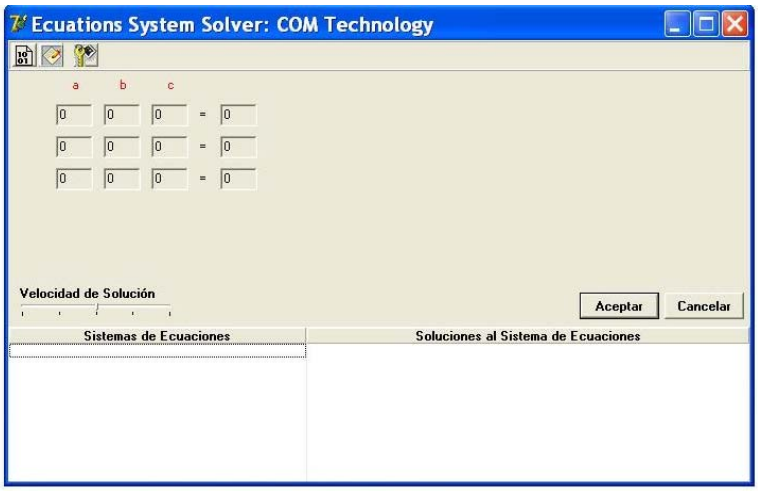


Figura 6.14. Sistema de ecuaciones de 3x3

El usuario deberá dar valores a estos campos vacíos para que el sistema pueda contener datos reales (Figura 6.15), y después dar clic en el botón de aceptar. Con esto, el sistema es almacenado en Matlab y el nombre de éste se envía a la columna “Sistemas de Ecuaciones” para después poder ser manipulado (Figura 6.16).

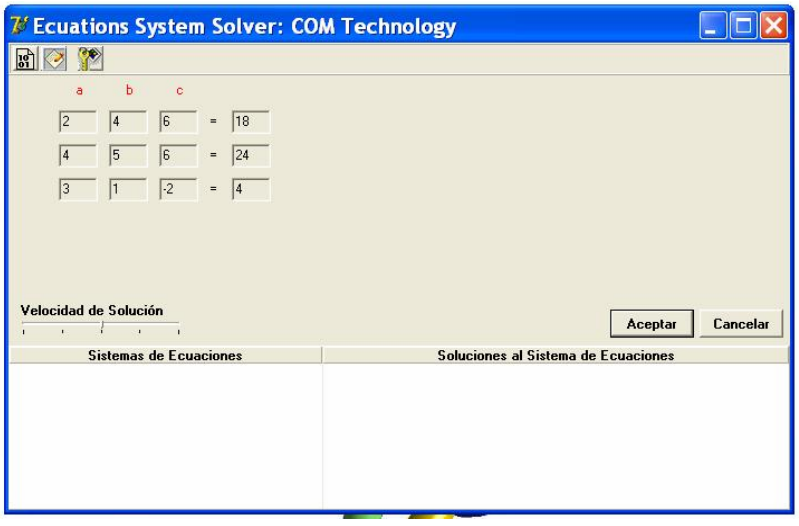


Figura 6.15. Datos dados por el usuario

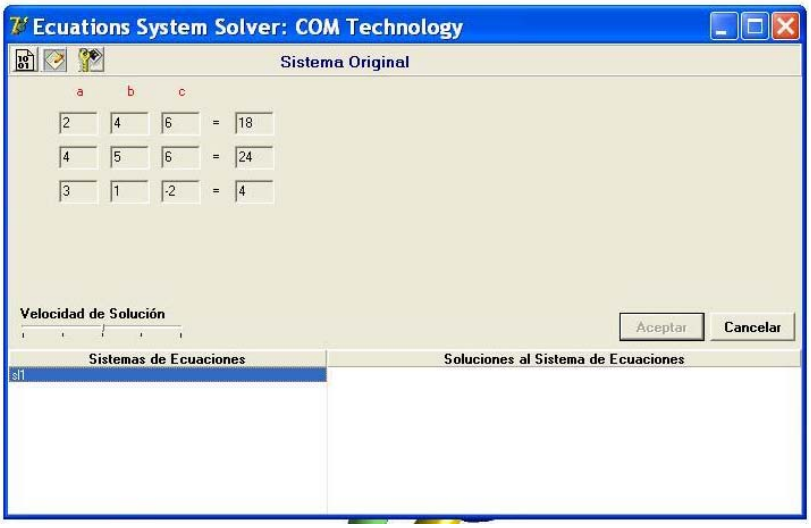


Figura 6.16. Sistema de ecuaciones almacenado en Matlab y listo para su manipulación

El sistema no está siempre en la ventana principal, es mostrado cada vez que el usuario lo desee, al momento en el que da clic sobre el nombre del mismo dentro del campo "Sistemas de Ecuaciones".

Ahora bien, para solucionar el sistema el usuario deberá dar clic derecho sobre el nombre del sistema que quiere solucionar, de tal manera que aparecerá un sub-menú con dos opciones de solución: por Regla de Cramer (determinante) o por la Inversa (Ver Figura 6.17).

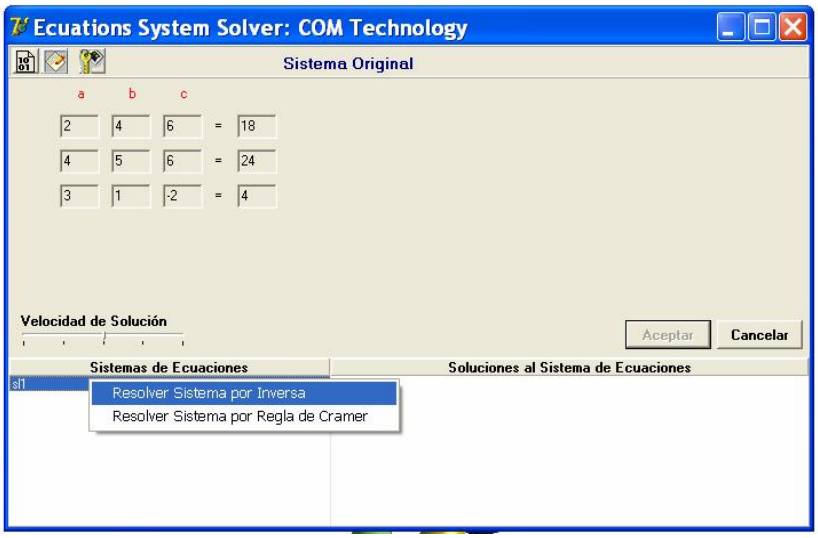


Figura 6.17. Métodos de Solución

El usuario debe seleccionar el método de solución que desee y de inmediato la solución será mostrada en el campo "Soluciones al Sistema de Ecuaciones" y en el espacio principal de la ventana (Ver Figura 6.18). En este caso el usuario escoge la solución por el método de la inversa y es la que se muestra en la figura.

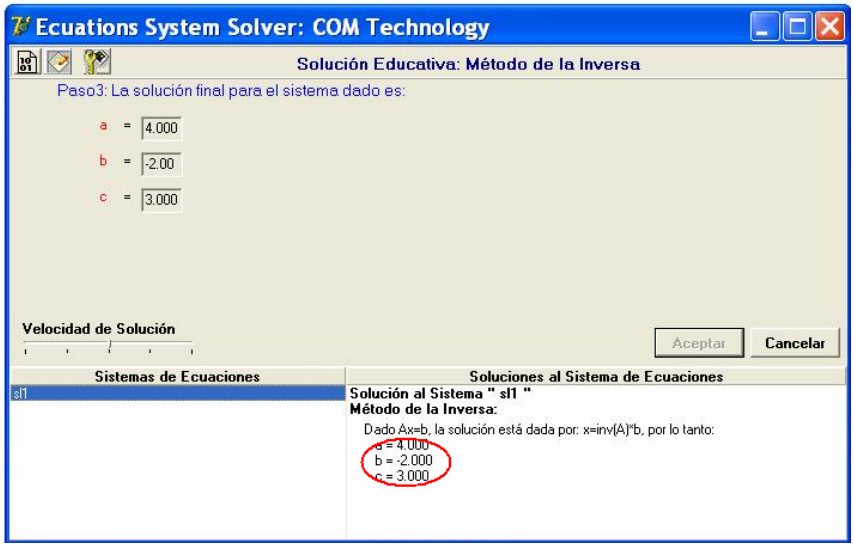


Figura 6.18. Solución del sistema de Ecuaciones (Método de la Inversa)

Como podemos ver en la Figura 6.18 la solución coincide con la dada por el autor del libro y la mencionada al inicio de este apartado, de tal manera que podemos comprobar que el sistema es confiable para su uso y su funcionalidad es la esperada.

Este es sólo un ejemplo con un método de solución, sin embargo, se puede hacer con ambos métodos y los resultados serán los mismos (Figura 6.19).

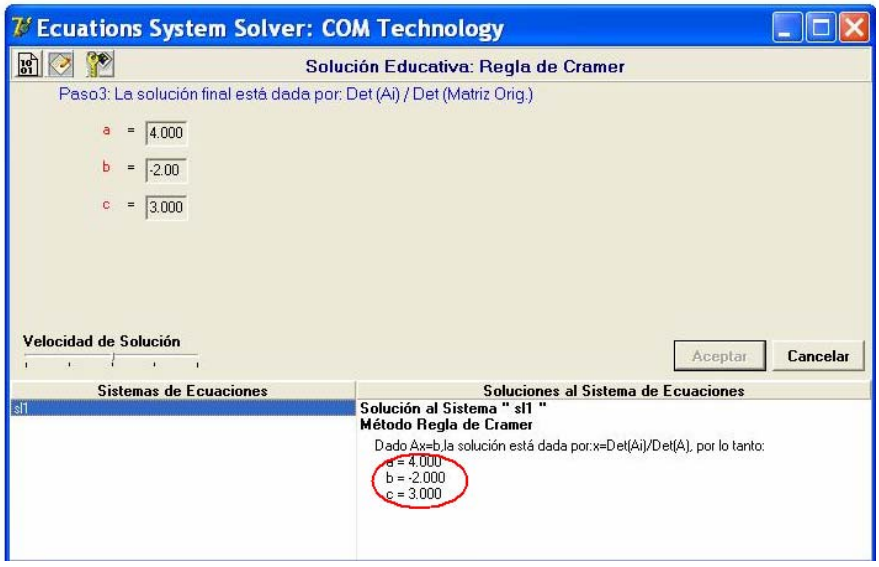


Figura 6.19. Método regla de Cramer

De la misma manera, existen dos posibles maneras de mostrar la solución de forma gráfica, esto es: Solución educativa o Solución automática, que el usuario podrá elegir desde el menú Sistemas de ecuaciones. Cada uno tiene diferente funcionalidad:

1. Solución Automática: Muestra únicamente el resultado final para el sistema de ecuaciones seleccionado.
2. Solución por Educativa: Esta es una solución por pasos, que muestra el resultado paso a paso hasta llegar a la solución. Este método tiene la finalidad de ayudar a los estudiantes a entender como se realiza el proceso de solución.

Estas pruebas fueron aplicadas de manera exhaustiva con más sistemas de ecuaciones, algunos de los cuáles se incluyen en el ANEXO 5[7] con su respectiva bibliografía, y cabe mencionar que el resultado fue el esperado. Los resultados, de forma detallada, no se muestran de manera gráfica debido a cuestiones de espacio como se mencionó desde el inicio.

Cabe señalar que este software fue probado con éxito por los profesores: F.M. Hilario Flores y M. Miguel Tirso, ambos profesores pertenecen al Instituto de física y matemáticas de la UTM y dentro de sus actividades académicas se encuentra el uso recurrente de MATLAB.

[7] Por cuestiones de espacio, este anexo se encuentra en un documento separado de éste llamado anexos.pdf

7. Conclusiones y expectativas

En la actualidad nos encontramos en un mundo gobernado por los sistemas y la electrónica, por ello, se exige crear sistemas de cómputo cada vez más robustos, confiables y de menores costos de desarrollo. Por tal motivo, los sistemas de cómputo se vuelven demandantes y, con ellos, el software para acrecentar el poder de los mismos sistemas.

Es en este momento, en el que se tiene la necesidad de desarrollar sistemas más robustos y poderosos, sistemas que no sean de gran costo por su desarrollo, sino que sean capaces de tomar en cuenta lo ya desarrollado para crear nuevas propuestas que sean capaces de integrar desarrollos elaborados previamente para encontrar una nueva forma de adaptarse al constante incremento de la tecnología. Por lo anterior, surgen nuevos y poderosos paradigmas cuya principal función se basa en la reutilización de software. Paradigmas orientados a objetos y orientados a componentes mediante los cuáles se puede desarrollar software de menor costo y muy poderoso, con una sola premisa, utilizar lo que ya está construido y hacer de nuestros desarrollos lo más abierto posible para que puedan ser parte de otro sistema.

Es así, como en este trabajo de tesis se observó la necesidad de desarrollar componentes de software que sean reutilizados en sistemas futuros o ya existentes. En este caso en particular, se desarrollaron componentes de software que encapsulan las variables, funcionalidad y tipos de datos de Matlab; así como una aplicación de software que verifique la funcionalidad de dichos componentes. La importancia del desarrollo de estos componentes radica en que formarán la base para el desarrollo de otros componentes de software, es decir, estos componentes permitirán entablar una comunicación con Matlab, encapsular las variables y tipos de datos de dicho lenguaje y, principalmente, el manejo de sesiones que realicen la comunicación entre un lenguaje de programación cotidiano y el leguaje matemático. Mediante esta comunicación y teniendo ya los tipos de datos de manera programable se podrán crear componentes independientes que encapsulen otra funcionalidad de Matlab sin preocuparse del manejo de sesiones y tipos de datos.

A lo largo del presente capítulo se detallan las conclusiones, aportaciones y limitaciones que se presentaron a lo largo de este desarrollo.

7.1. Conclusiones finales.

Este proyecto contempló la realización de dos componentes de software que sean capaces de encapsular operaciones y tipos de datos de Matlab, de tal manera que el objetivo fue

cumplido en su totalidad. Los componentes desarrollados, tienen como finalidad proporcionar a los desarrolladores una interfaz que sea de fácil comprensión al momento de que se vean en la necesidad de integrar los componentes en el uso de sistemas que requieran la manipulación de variables y tipos de datos de Matlab. La principal característica de los componentes es que son independientes del lenguaje de programación (componente COM) y de la plataforma de desarrollo (componente SOAP). La justificación principal del desarrollo de estos componentes es que están orientados no sólo a la población Universitaria, como se pensaba en un principio, sino a una comunidad mayor como son investigadores, profesores y cualquier otra persona que tenga interés en el desarrollo de aplicaciones de software, por tal motivo, se tiene pensado que a partir de este momento y en el futuro los componentes sean una herramienta de gran utilidad muy utilizada por diferentes personas y en diferentes lugares convirtiéndose así, en una herramienta poderosa y reusable que sea capaz de aminorar los problemas de un desarrollador y acrecentar las posibilidades de solución para un problema. Al ser este componente una interfaz directa para la interacción con Matlab, ofrece a la comunidad de desarrolladores una nueva manera de hacer software matemático, prueba de ello, son los desarrollos que se han estado haciendo en la Universidad de Vigo en España y dentro de la misma UTM, mismos que se basan en el uso de estos componentes para la realización de software matemático hecho a la medida y componentes diversos.

De la misma manera se desarrolló una aplicación de software basada en la solución de sistemas de ecuaciones mediante el uso de matrices. Dicha aplicación cumplió sus cometidos: Corroborar la utilidad y funcionalidad de los componentes de software desarrollados mediante la integración de los componentes en la misma aplicación, resolver un problema matemático como es el encontrar solución a sistemas de ecuaciones de una manera rápida, con lo cuál, se beneficia a universitarios, profesores, investigadores y demás personas relacionadas con el tema y, por último, dar inicio a la formación de una librería de componentes mediante la integración de otros componentes desarrollados actualmente o en un futuro, en combinación de los desarrollados en esta tesis que son la base de dicha librería. La funcionalidad se vio incrementada debido a que el uso de los componentes puede ser local y/o remoto dependiendo de las necesidades y gusto del usuario.

Por lo anterior, podemos concluir que los objetivos planteados para este proyecto se cumplieron satisfactoriamente. En esta primera versión, debido a la complejidad del sistema y principalmente de los componentes desarrollados, se presentan algunas limitaciones, las cuales se plantean como posibles trabajos futuros.

7.2. Aportaciones realizadas.

Las aportaciones principales de esta tesis son las siguientes:

- Dos componentes de software poderosos capaces de ayudar tanto a alumnos como profesores-investigadores universitarios en el desarrollo de software relacionado. Se espera sean de gran utilidad y ayuda a los usuarios tanto en el trabajo local y remoto, permitiéndoles reducir tiempos y costos de desarrollo.
- Una aplicación de software que ayude tanto a alumnos como profesores- investigadores y demás usuarios en la solución de sistemas de ecuaciones de una manera automatizada y exacta, que hagan procesos rápidos y seguros al momento de la obtención de las soluciones.

- Dar paso a la posible formación de una librería de componentes futura mediante la integración de nuevos componentes realizados dentro de la UTM, abriendo con esto líneas de investigación para estudiantes y profesores.
- Difundir los componentes a otras universidades, lo que dará valor agregado a la UTM haciendo notar la capacidad con la que se cuenta, así como el nivel y prestigio.
- Por último, este trabajo abre las puertas a un sin fin de líneas de investigación para la realización de investigaciones, temas de tesis y publicación de artículos, en base a las líneas de investigación que esta tesis propone o generando nuevas líneas en base a las que aquí se postulan, no solo a nivel UTM sino mas allá, como un principio, la Universidad de Vigo España, ya se encuentra haciendo difusión de este trabajo e implementando su uso para nuevos desarrollos.

7.3. Líneas de investigación abiertas.

Este proyecto deja como principal línea de investigación abierta, la posibilidad de construir una librería de componentes basados en los desarrollados en esta tesis. Dicha librería pretende integrar componentes realizados en tesis individuales dentro de la Universidad que abran las oportunidades a los estudiantes de la misma y den valor agregado tanto a la Universidad como a los profesores-investigadores.

La librería de componentes se encuentra ya en sus inicios, se probó su funcionalidad mediante la integración de los componentes desarrollados en una tesis dentro de la misma UTM. [8]

El siguiente paso es el desarrollo de otros componentes que encapsulen otras funciones de Matlab e integrarlas para formar una librería de componentes completa y funcional.

Es importante mencionar, que debido al impacto que tuvo este desarrollo, no solo dentro de la UTM sino a nivel internacional, es importante abrir una línea de investigación que continúe en trabajo de componentes COM-SOAP, ya sea con Matlab mismo o con algún otro sistema cerrado.

7.4. Trabajos futuros.

La principal finalidad de la realización de esta tesis y, después de los resultados obtenidos, se pretende dar una gama de posibilidades a los alumnos de la Universidad en cuanto al desarrollo de componentes de software bajo tecnologías como las aquí utilizadas. Asimismo, darles a conocer los grandes beneficios que trae consigo para motivarlos en la contribución de nuevos productos.

El desarrollo del proyecto que aquí se presentó se encuentra debidamente documentado de manera exhaustiva, motivo por el cuál se considera existe mayor posibilidad de poder des-

8 Felipe de Jesús García Pérez (2005) "Desarrollo de componentes de software local, y distribuido bajo la plataforma COM-SOAP, que encapsule las operaciones matriciales de MATLAB". Tesis de Licenciatura, UTM, Huajuapan de León, Oaxaca, México

arrollar software similar debido a la investigación realizada, al mismo tiempo, se dan algunos puntos que podrán ser implementados como desarrollos futuros:

- Crear otros componentes de software basados en los desarrollados en este trabajo de tesis pero enfocados a las diferentes herramientas matemáticas que maneja Matlab, es decir, cada una de las operaciones que el lenguaje es capaz de manipular y resolver.
- Crear una librería de componentes completa y funcional en base a los componentes individuales que pueden ser creados por diferentes universitarios.
- Crear una aplicación más robusta que de solución a problemas que aquejan a la comunidad universitaria.
- Crear una aplicación que sea capaz de resolver sistemas de ecuaciones no cuadrados para dar un valor agregado a la utilidad de la misma.
- Eliminar la dependencia del lenguaje Matlab a un lenguaje libre de igual o mayor funcionalidad para evitarse los costos de software y desarrollo.

Es importante aclarar que los puntos aquí citados pueden ser de un grado de complejidad mayor, sin embargo, se consideran dignos de tratarse como proyectos futuros debido a que parten del desarrollo de este proyecto y podrían dar solución a muchos problemas que nos encontramos en la actualidad. Por último, este trabajo de tesis cumple de manera exitosa con cada uno de los objetivos planteados al inicio del desarrollo del proyecto, así como da solución a la hipótesis planteada. De aquí surgen nuevas líneas de investigación y posibles trabajos futuros para el aprovechamiento de lo desarrollado.

Bibliografía

[1] Bass, L., P. Clements y R. Kazman “Software Architecture in Practice”, Addison-Wesley, 1998.

[2] Boehm, B. W. “Software Engineering”, IEEE Transactions on Computers, C-25, no. 12. Diciembre.

[3] Boehm, Barry W. "Software and Its Impact: A Quantitative Assessment," Datamation May, 1973.

[4] Booch, G. (1988) “Object Oriented Development”. Trans. of Soft. Eng. Vol. SE-12. Num. 2. Feb. 1986. pp. 211-221.

[5] Booch, G. (1994), “Object-Oriented Analysis and Design with Applications”. Benjamin-Cummings, Redwood City, CA.

[6] Booch, G., Rumbaugh, J., Jacobson, I., “El Lenguaje de Modelado Unificado”. Addison-Wesley, Iberoamericana, 1999.

[7] Box, D. (1998). “Essential COM”. Addison-Wesley.

[8] Bredemeyer, Dana & Malan, Ruth (2001) “Architecture Definitions”. Artículo.

[9] Budd, T. “Introducción a la Programación Orientada a Objetos”. Addison-Wesley Iberoamericana.1994

[10] Castaño I., Florencio, Lafuerza G., Bernardo, Martínez G., Pedro. "Análisis numérico de ecuaciones diferenciales con MATLAB". Disponible en: http://www.ual.es/personal/estmatap/numerico.htm

[11] Clements, P.C., “From Subroutines to Subsystems: Component-Based Software Development”, American Programmer, vol. 8, no. 11, Noviembre 1995.

[12] Coad P. y Yourdon E. (1991). “Object-Oriented Analysis”. Prentice-Hall.

[13] Freire, José Luis (2001) “Tutorial de Delphi”. Disponible en: http://www.elrinconcito.com/delphi/tutorial/tutorial.htm

[14] Fuentes L. (2000), “Desarrollo de Software basado en componentes”. Universidad de Málaga.

[15] Gamma, E., Helm, R. et al. (1995) “Design Patterns: Elements of Reusable Object-Oriented Software”. Adison-Wesley.

[16] Gamma, E., Helm, R., Johnson, R. & Vlissides, John (1993) “Design Patterns: Abstraction and Reuse Object-Oriented Design” Artículo, Proceedings of ECOOP'93, volume 707 of Lecture Notes in Computer Science. Springer-Verlag.

[17] García E., Gonzáles J., Pino D., Baltasar J., Martínez D., Pérez M., “Software educativo basado en componentes reutilizables: una propuesta para la integración de entornos propietarios”. In proceedings of the 10mo. Congreso Internacional de Investigación en Ciencias Computacionales. Octubre 20-22, 2003 — Cuernavaca, México.

[18] Garlan, D. and M. Shaw (1993), “An Introduction to Software Architecture”, in Advances in Software Engineering and Knowledge Engineering, vol. 1, World Scientific Publishing Company, 1993.

[19] Greiff W. R. “Paradigma vs Metodología; El Caso de la POO (Parte II)”. Soluciones Avanzadas. Ene-Feb 1994.

[20] Jacobson, I., “Object-Oriented Software Engineering, A use case driven approach”, Addison-Wesley, ACM press, 1992.

[21] Jacobson, I., Booch G., Rumbaugh J. (1998). “The Unified Software. Development Process”. Reading, MA: Addison-Wesley Longman Inc.

[22] Jones, A. K. (1993), “The Maturing of Software Architecture,” Software Engineering Symposium, Software Engineering Institute, Pittsburgh, Pa., August 1994.

[23] Microsoft y Rational 1998. “A White Paper on the Benefits of Integrating Microsoft Solutions Framework and The Rational Process”. Rational Software Corporation y Microsoft Corporation. Documento. Disponible en: http://www.rational.com/uml/papers.

[24] Nierstrasz, O. (1995). “Requirement for a Composition Language”. En ciancarini, P., Nierstrasz, O., y Yonezawa, A. (eds),

[25] Object Management Group (1996). “The Common Object Request Broker: Architecture and Specification”, July 1996. Proc. of the ECOOP'94 Workshop on Object-Based Models and Languages for Concurrent Systems.

[26] Perry, D.E. and A. L. Wolf (1992), “Foundations for the Study of Software Architecture”, Software Engineering Notes, ACM SIGSOFT, vol. 17, no. 4, October 1992.

[27] Pressman, Roger. (2002) “Ingeniería de Software un enfoque práctico”, Ed. Mc Graw Hill/Interamericana de España, España.

[28] R. Pressman, Roger (1998) “Ingeniería del Software un enfoque práctico”, ed. Cuarta, Ed. McGraw Hill, México.

[29] Rogerson, D. (1997). “Inside COM: Microsoft’s Component Object Model”. Microsoft Press.

[30] Roschelle J., Kaput J., Stroup W., Kahn T.M. (1998). “Scaleable Integration of Educational Software: Exploring the Promise of Component Architectures. Journal of Interactive Media in Education”.

[31] Roschelle, J., DiGiano C., Chung M. (2000). “Reusability and Interoperability of Tools for Mathematics”, Learning: Lessons from the ESCOT Project. En: Proceedings of Intelligent Systems & Applications at University of Wollongong, NSW Australia. ICSC Academic Press, Wetaskiwin, AB, Canadá, pp. 664-669.

[32] Rose, A., Eckard, D. Rubloff, G. (1998), “An Application Framework for Creating Simulation-Based”. Learning Environments. University of Maryland Technical Report CS-TR-3907, College Park, MD 20782. Disponible en: http://citeseer.nj.nec.com/116687.html

[33] Rumbaugh, J., et al. (1991) “Object Oriented Modeling and Design”. Prentice-Hall.

[34] Salinas C., Patricio, Histchfeld K., Nancy “Tutorial de UML”. Disponible en: http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html

[35] Sametinger, J. (1997), “Software Engineering with Reusable Components”. Springer, New York, 1997.

[36] Samuel Garrido, Daniel. Disponible en: http://computacion.cs.cinvestav.mx/~sgarrido/cursos/ing_soft/Componentes/node16.html

[37] Schneider, Jean-Guy (1999) “Component, Scripts and Glue: A conceptual framework for software composition”, Tesis Doctoral, Universidad de Bern, Suiza, 18 de Octubre de 1999. Págs. 27.

[38] Sommerville, Ian (2002) “Ingeniería de Software”, ed. Sexta, Ed. Pearson Educación, México. Pág. 230, 231, 261.

[39] Stanley I. Grossman. Algebra Lineal. 5ª Ed. ED. Mc. Graw hill.

[40] Sun MicroSystems (1999). Enterprise JavaBeans Specification. August, 1999.

[41] Szypersky, C. y Pfister, C. (1997). "Summary of the Workshop on Component Oriented Programming (WCOP'96)". En Mühlhäuser, M. (ed.), Workshop Reader of ECOOP'96. Dpunkt Verlag.

[42] The Mathworks, Inc., "Matlab (2004)". Disponible en: http://www.mathworks.com

[43] The MathWorks, Inc., “MATLAB User's Guide”, Massachusetts, 1995.

[44] Vallecillo M., A. (1999) “Un modelo de componentes para el desarrollo de aplicaciones distribuidas.” Tesis de Doctorado, Universidad del Área de Lenguajes y Sistemas informáticos, Málaga, España Pág. 3, 64.

Sitios de internet

[URL1] http://www.webopedia.com "Webopedia: Online Computer Dictionary for Computer and Internet Terms and Definitions" Internet.com, Jupitermedia, Corporation 2004.

[URL2] http://www.borland.com Delphi 6.0 (2001). "Prog. Computacional". Borland Inc.

[URL3] http://buscon.rae.es/diccionario/drae.htm Diccionario de la lengua española. "Software, Sesión".

[URL4] http://www.microsoft.com/Com/default.mspx Microsoft (2004). "Component Object Model". Microsoft Corporation.

[URL5] http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/art51.asp Microsoft (2004). "SOAP y WebServices". Microsoft Corporation.

[URL6] http://java.sun.com/products/javabeans/ Sun Developer Network (SDN) (2006). "JavaBeans". Sun Microsystems, Inc. (1994-2006).

[URL7] http://mixtli.utm.mx/~casi/ Universidad Tecnológica Mixteca (2004), "Cuerpo Académico de Ingeniería de Software".

[URL8] http://www.microsoft.com Visual Basic 6.0 (2000). "Prog. computacional."

[URL9] http://www.w3.org/TR/2000/NOTESOAP-20000508 W3C (2000) "Simple Object Access Protocol (SOAP)".

[URL10] http://www.xml.com/pub/a/98/10/guide0.html?page=2#AEN58 "What is XML" O'Reilly Media, Inc.

Anexo 1. Glosario

Application Program Interface (API). Conjunto de rutinas, protocolos, y herramientas para la construcción de aplicaciones software. Un buen API hace que sea fácil desarrollar un programa que provea todos los bloques de construcción [URL1] .

Component Object Model (COM). Arquitectura e infraestructura para construir software basado en componentes, software rápido, robusto y expandible. Es una tecnología creada por la familia Microsoft-Windows para permitir la comunicación entre componentes de software. COM es usado por desarrolladores para crear componentes de software reusables, interconectar componentes para construir aplicaciones y obtener beneficio de los servicios de Windows. La familia de tecnologías COM incluye COM+, COM Distribuido (DCOM) y controles ActiveX [URL4] .

Delphi. Entorno de programación muy completo, propiedad de Inprise-Borland, que se basa en el lenguaje Object Pascal. Completamente basado en los principios de Programación Orientada a Objetos (POO), con acceso a cualquier base de datos existente, con un compilador y una ejecución muy rápida [URL2] .

Simple Object Access Protocol (SOAP). Protocolo ligero para el intercambio de información en entornos descentralizados y distribuidos, basado en XML. Puede ser usado con otros protocolos [URL9] .

XML. Lenguaje de marcado para documentos que contienen información estructurada. La especificación XML define un estándar para agregar etiquetas a documentos. Estas etiquetas se denominan mensajes [URL10] .

Anexo 2. Acrónimos

Acrónimo	Definición
API	Application Program Interfaz (Interfaz de Programación de Aplicaciones)
CASI	Cuerpo Académico de Ingeniería de Software
COM	Component Object Model
ERS	Especificación de Requerimientos de Software
MatLab	Matrix Laboratory
MOO	Métodos Orientados a Objetos
MVC	Modelo Vista Controlador
OO	Object Oriented
OOSE	Object Oriented Software Engineering
POC	Programación Orientada a Componentes
POO	Programación Orientada a Objetos
RUP	Rational Unified Process (Proceso Unificado Rational)
SOAP	Simple Object Access Protocol
TOC	Tecnología Orientada a Componentes
TOO	Tecnología Orientada a Objetos
UML	Unified Modeling Language
UTM	Universidad Tecnológica de la Mixteca

Anexo 3. Extensión de especificación de casos de uso para los componentes de software.

Especificación de caso de uso: Gestionar IMOCOMMatlab-Session
Versión 1.2

Historial de revisiones

Fecha	Versión	Descripción	Autor
22/Noviembre/2004	1.0	Creación del Documento	Josefina Flores Flores
27/Noviembre/2004	1.1	Introducción de primeros diagramas	Josefina Flores Flores
1/Diciembre/2004	1.2	Documento completado	Josefina Flores Flores

Especificación de casos de uso: Gestionar IMOCOMMatlab-Session

Escenario: Cerrar una Sesión de Matlab.

Objetivo

Gestionar la interfaz IMOCOMMatLabSession para Cerrar una sesión de Matlab.

Breve descripción

El usuario podrá gestionar la interfaz IMOCOMMatLabSession para poder Cerrar una sesión o entorno de trabajo de Matlab.

Actores

Usuario

Flujo de eventos

Flujo básico

Actor	Sistema
Este caso de uso comienza cuando el usuario decide instanciar la interfaz IMOCOMMa-tLabSession y desea cerrar una sesión abierta de Matlab.	
El usuario utiliza el objeto el cual contiene la variable creada y ejecuta la función “Close”.	
	El sistema ejecuta la operación y cierra una sesión de Matlab abierta que estaba siendo utilizada por el objeto.
Finaliza el caso de uso.	

Excepción

Si no se puede instanciar la interfaz requerida se notificará mediante un mensaje de error.
Si la operación no pudo ser realizada se notificará mediante un mensaje de error.

Precondiciones

Tener abierta una sesión de Matlab.

Diagrama de actividades

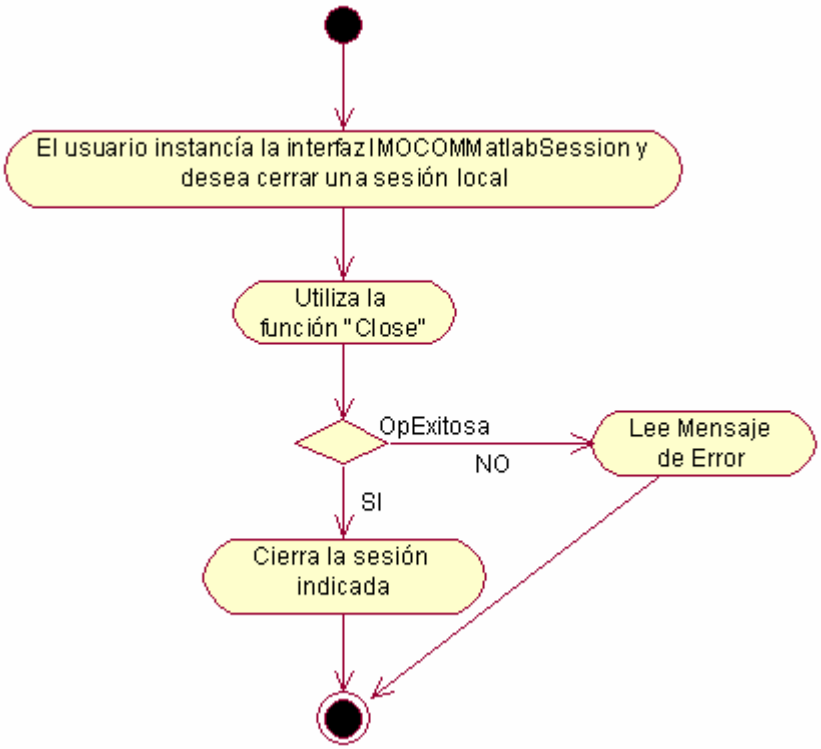


Diagrama de secuencia

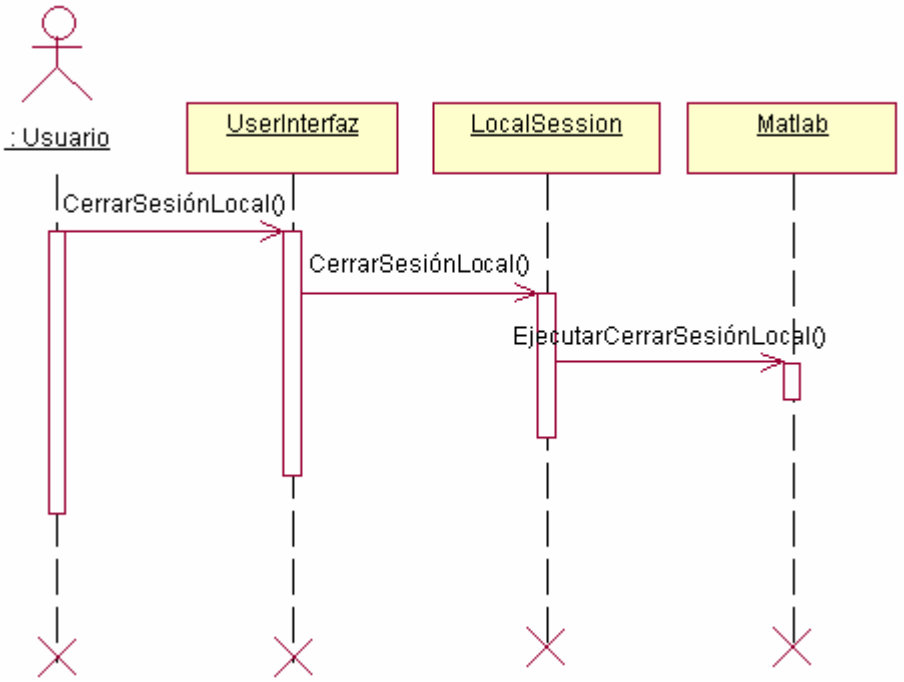
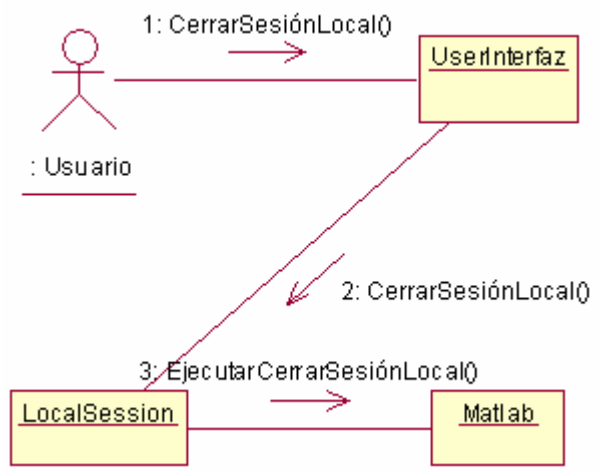


Diagrama de colaboración



Especificación de caso de uso: Gestionar IMOCOMMatlab-Var
Versión 1.2

Especificación de casos de uso: Gestionar IMOCOMMatlabVar

Escenario: Copiar el contenido de una variable a otra.

Objetivo

Gestionar la interfaz IMOCOMMatLabVar para copiar el contenido de una variable a otra.

Breve descripción

El usuario podrá gestionar la interfaz IMOCOMMatLabVar para poder copiar el contenido de una variable a otra.

Actores

Usuario

Flujo de eventos

Flujo básico

Actor	Sistema
Este caso de uso comienza cuando el usuario tiene instanciada la interfaz IMOCOMMatLabVar y decide copiar el contenido de una variable a otra.	
El usuario ejecuta la función "CopyFrom" con el parámetro de entrada que indica la variable de la cual se va a copiar el contenido a la variable que encapsule el objeto actual.	
	El sistema ejecuta la operación y ahora la variable encapsulada por el objeto indicado, va a contener el nuevo valor de la variable indicada.
	El sistema devuelve un apuntador a la variable que contiene el nuevo valor.
Finaliza el caso de uso.	

Excepción

Si no se puede instanciar la interfaz requerida se notificará mediante un mensaje de error.
Si la operación no pudo ser realizada se notificará mediante un mensaje de error.

Precondiciones

Tener abierta una sesión de Matlab.

Haber creado e instanciado las variables a utilizar.
Tener asignada una sesión a las variables a utilizar.

Diagrama de actividades

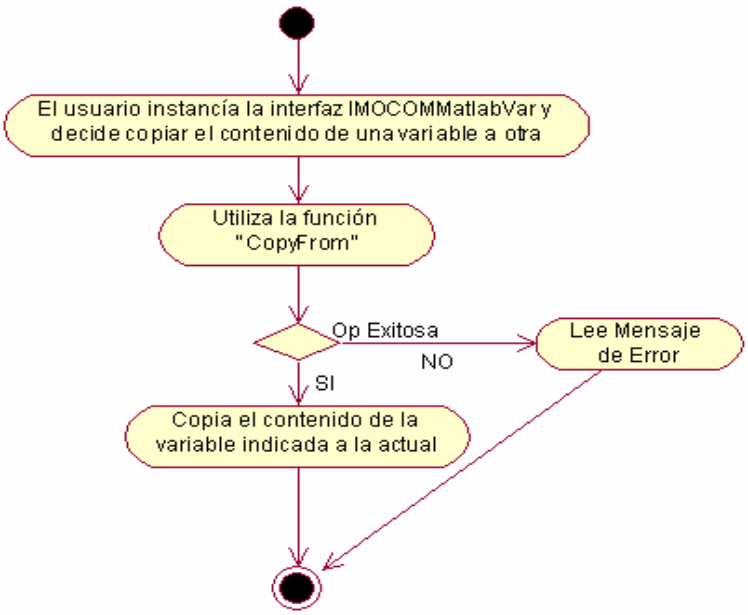


Diagrama de secuencia

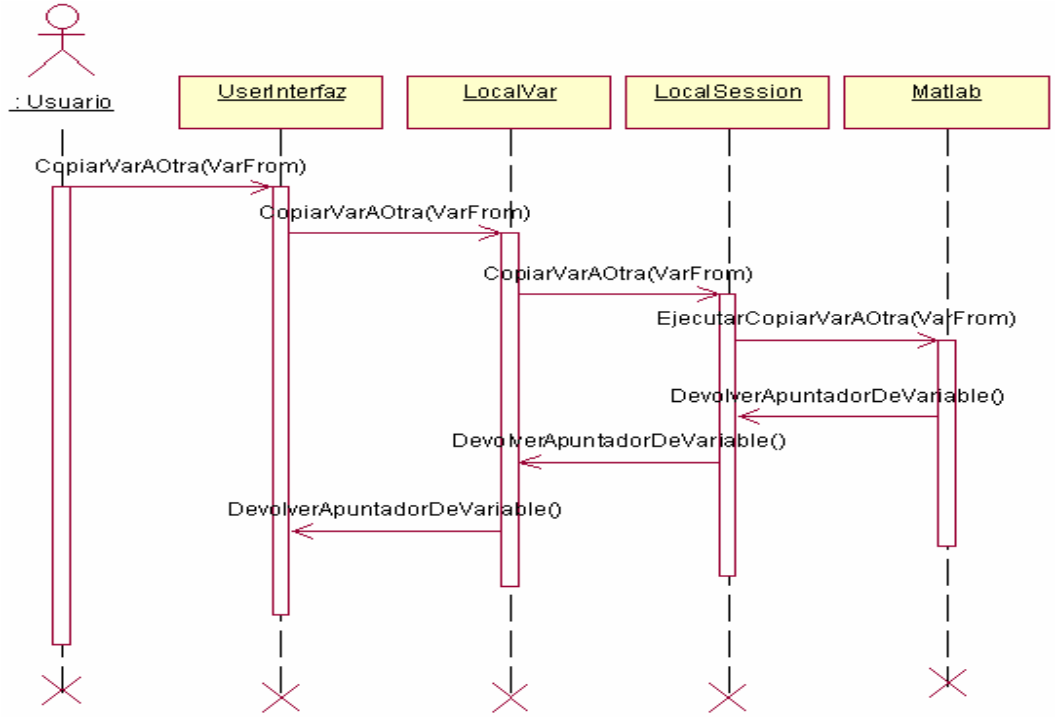
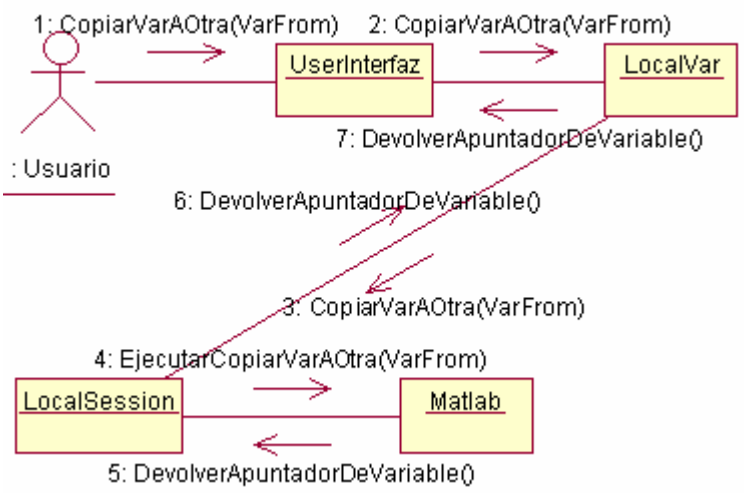


Diagrama de colaboración



Por cuestiones de espacio, el resto de casos de uso se encuentran en el archivo Anexos.pdf