UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UN SOFTWARE ORIENTADO A OBJETOS, QUE USA EL MÉTODO DE ELEMENTOS DE FRONTERA, PARA RESOLVER PROBLEMAS BIDIMENSIONALES DE ELASTICIDAD (ESFUERZO PLANO/DEFORMACIÓN PLANA)

REALIZADO POR:

Lara Rojas, Manuel Rafael

Trabajo de Grado presentado como requisito parcial para obtener el Título de
INGENIERO EN COMPUTACIÓN

Barcelona, Abril 2010

UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UN SOFTWARE ORIENTADO A OBJETOS, QUE USA EL MÉTODO DE ELEMENTOS DE FRONTERA, PARA RESOLVER PROBLEMAS BIDIMENSIONALES DE ELASTICIDAD (ESFUERZO PLANO/DEFORMACIÓN PLANA)

ASESOR:

_____

Ing. Carlos Gomes M.Sc., Ph.D.

Barcelona, Abril 2010

UNIVERSIDAD DE ORIENTE

NÚCLEO DE ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UN SOFTWARE ORIENTADO A OBJETOS, QUE USA EL MÉTODO DE ELEMENTOS DE FRONTERA, PARA RESOLVER PROBLEMAS BIDIMENSIONALES DE ELASTICIDAD (ESFUERZO PLANO/DEFORMACIÓN PLANA)

JURADO CALIFICADOR:

El Jurado hace constar que asignó a esta Tesis la calificación de:

APROBADO

_____

Ing. Carlos Gomes M.Sc., Ph.D.

Asesor Académico

_____ _____

Lic. José Bastardo, M.Sc. Ing. Claudio Cortinez

Jurado Principal Jurado Principal

Barcelona, Abril 2010

RESOLUCIÓN

ARTÍCULO Nº 41 Del Reglamento de Trabajo de Grado

"Los trabajos de grado son de exclusiva propiedad de la Universidad y sólo podrán ser utilizados para otros fines con el conocimiento del Consejo de Núcleo respectivo, quién lo participará al Consejo Universitario".

RESUMEN

En la ingeniería mecánica es frecuente plantear problemas elásticos para decidir la adecuación de un diseño. En la actualidad existen múltiples métodos numéricos para la resolución de este tipo de problemas, entre ellos el método de elementos de frontera (BEM, por sus siglas en ingles: *Boundary Element Method*). BEM desempeña un papel importante en cálculos científicos y técnicos, sin embargo, debido a que la mayoría de los códigos computacionales que implementan BEN fueron escritos en lenguaje de programación FORTRAN, estos tienen limitaciones de reusabilidad y expansión ya que el lenguaje de programación FORTRAN es procedimental. En este trabajo se presenta el desarrolló un software orientado a objetos que utiliza el método de elementos de frontera (BEM) para resolver problemas bidimensionales de elasticidad (esfuerzo plano y deformación plana). El software se desarrolló usando los lenguajes: C++ y JAVA, con la intención de aprovechar las características resaltantes de estos dentro de los lenguajes orientados a objeto, como lo son el potencial gráfico de JAVA y la velocidad de cálculo de C++. La conexión de estos lenguajes se hizo mediante Java Native Interface (JNI, por sus siglas en ingles: *Java Native Interface*). Para el desarrollo del proyecto se utilizó la metodología OMT (del ingles: *Object Modeling Technique*), por medio del Lenguaje Unificado de Modelado (UML, por sus siglas en ingles: *Unified Model Lenguage*). Es de destacar que este software puede integrarse con otros métodos como: método de elementos finitos (FEM, por sus siglas en ingles: *Finite Element Method*), método de volumen finito (FVM, por sus siglas en ingles: *Finite Volume Method*), método de diferencias finitas (FDM, por sus siglas en ingles: *Finite Diference Method*) y formar un solo software usando los conceptos de herencia, abstracción y polimorfismo. El software, permite mejorar y experimentar con BEM colaborando así con las investigaciones en esta área.

DEDICATORIA

A mi Madre

INTRODUCCIÓN

El método de elementos de frontera (BEM), es un método numérico que mediante ecuaciones diferenciales parciales puede resolver problemas estructurales, térmicos, y de fluidos de una manera precisa para la mayoría de los fenómenos físicos que se presentan en la naturaleza, pero necesita de la programación para poder realizar los cálculos necesarios para encontrar la solución a cualquier problema físico.

La programación orientada a objetos (POO) ha marcado un hito en la evolución del desarrollo de software, ya que se puede obtener un software robusto de alta calidad y con menos código fuente. Dos de las ventajas más importantes de la POO son: su capacidad de implementar nuevos tipos de datos abstractos (TDA, por sus siglas en ingles), ya sea tipos de datos independientes o como combinación de los ya existentes; y la herencia aunada a la gran abstracción que proporciona este estilo de programación.

La POO puede ser aplicada por diversos lenguajes de programación como: Object Pascal, Python, Smalltalk, Java y C++. Este último lenguaje es uno de los más usados para la programación orientada a objetos, ya que al ser una evolución del lenguaje de programación C, le permite al usuario elegir cuando usar el paradigma funcional nativo de C o el orientado a objetos de C++. En otras palabras es multiparadigma

En función a esto último, el presente trabajo de investigación, tiene como objeto el desarrollo de un software orientado a objetos, que usando el BEM, pueda resolver problemas bidimensionales de elasticidad (esfuerzo plano/deformación plana), los cuales son necesarios para el diseño de estructuras, aeronaves, puentes etc.

El presente informe está estructurado en cinco (5) capítulos que se explican a continuación:

El capítulo 1, presenta la problemática que da origen a la investigación, aquí se muestran los objetivos que se persiguen alcanzar en la investigación, así como también la justificación del trabajo y sus limitaciones.

El capítulo 2, muestra la base de sustanciación documental de la presente investigación, las investigaciones precedentes a este método numérico así como su funcionamiento y previas implementaciones.

En el capítulo 3, se presenta la primera fase de la metodología OMT, conocida como fase de análisis, donde se hace un plan de fases, se identifican los principales casos de uso y riesgos del sistema a desarrollar, permitiendo esto establecer los requisitos que debe cumplir la aplicación y elaborar el conjunto de modelos que describirán el comportamiento de la misma.

El capítulo 4, muestra la segunda y la tercera fase de la metodología OMT, conocida como fase de diseño del sistema y fase de diseño de objetos respectivamente, En esta fase se toman las decisiones de alto nivel sobre la arquitectura del software así como se describen la forma en que el diseño será implementado, y los lenguaje(es) elegidos para la implementación.

El capítulo 5, muestra la fase de implementación de la metodología OMT, en la que se observan los componentes de software, la integración y las pruebas realizadas luego de la misma

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1
PLANTEAMIENTO DEL PROBLEMA

1.1. PLANTEAMIENTO DEL PROBLEMA

En la ingeniería mecánica es frecuente plantear problemas elásticos para decidir la adecuación de un diseño. En la mayoría de estas situaciones de interés eminentemente práctico, basta solo con plantear un modelo simplificado y aplicar la teoría básica de resistencia de materiales, para calcular de forma aproximada las tensiones y los desplazamientos que se están generando en un sólido por la acción de una fuerza, caso contrario al que ocurre cuando la geometría involucrada en el diseño mecánico es compleja, donde se debe recurrir a los métodos numéricos para resolver un problema elástico de manera aproximada.

En la actualidad, existen diversos métodos numéricos para resolver este último tipo de problemas: el método de diferencias finitas (FDM), el método de elementos finitos (FEM) y el método de elementos de frontera (BEM), que es el que aborda esta investigación. Es importante destacar, que cada uno de estos procedimientos tiene sus fortalezas y debilidades, aunque los tres se aplican ampliamente [1].

El BEM, es un algoritmo numérico computacional usado para resolver ecuaciones diferenciales parciales (EDP) que han sido formuladas como ecuaciones integrales [2]. Es necesario destacar, que una de las ventajas más importantes del BEM con respecto a los métodos descritos, se refiere a que la discretización de un modelo se limita solo a la frontera lo que representa, menor coste computacional en la creación y modificación de una malla.

Una de las primeras aplicaciones exitosas de este método, a problemas en mecánica de sólidos, fue formulada para el área de plasticidad por Swedlow and Cruse en el año de 1971 [3]. Seguida por una implementación numérica realizada en el año 1973 por Riccardella [4] en lenguaje de programación FORTRAN. Otras aplicaciones recientes del BEM para la mecánica de sólidos se pueden encontrar en el libro de Aliabadi [5].

Como se puede observar en la literatura científica, la mayor parte de programas de computadora que implementan el BEM fueron escritos en lenguaje FORTRAN [1,6]. Por ejemplo, existen códigos en este lenguaje para problemas de potenciales eléctricos, problemas de elasticidad, etc. Indudablemente, FORTRAN desempeña un papel importante en cálculos científicos y técnicos, sin embargo, tiene sus limitaciones, pues, un código escrito en FORTRAN es básicamente orientado al procedimiento y no orientado a objetos.

En años recientes, las aplicaciones de programación orientada al objeto (POO) al cálculo numérico han despertado mucha atención [1]. Desde 1996, diversas publicaciones [6-8], han introducido una arquitectura orientada a objetos, para ser usada en el desarrollo de programas de computadora basados en métodos como el BEM. Diversas clases han sido desarrolladas para automatizar tareas ordinarias como procedimientos de integración numérica, grados de libertad, entre otros. Siendo importante destacar, que las clases implementadas en estas investigaciones, fueron desarrolladas usando el lenguaje de programación C ++.

Basado en lo anteriormente expuesto, el presente trabajo de investigación, el cual se realiza en el Centro de Métodos Numéricos en Ingeniería (CMNI) de la Universidad de Oriente (UDO), está enfocada en el desarrollo de un software, que implemente BEM y que pueda resolver problemas bidimensionales de la mecánica de sólidos, específicamente en el área de teoría de elasticidad. Este software

proporcionará reusabilidad, extensibilidad y garantizara un fácil mantenimiento, lo que proveerá a la comunidad científica tanto nacional como internacional, una aplicación robusta, que pueda no solo abarcar problemas de elasticidad, sino que con pequeñas adiciones pueda extenderse para resolver otros problemas de ingeniería.

1.2. OBJETIVOS DE LA INVESTIGACIÓN

1.2.1. Objetivo general.

Desarrollar un software orientado a objetos, que usa el método de elementos de frontera, para resolver problemas bidimensionales de elasticidad (esfuerzo plano/deformación plana).

1.2.2. Objetivos específicos.

1. Analizar la estructura usada por el método de elementos de frontera (BEM) para la resolución de problemas bidimensionales de elasticidad.
2. Determinar los requerimientos necesarios para desarrollar un software usando la estructura estudiada.
3. Modelar el software a desarrollar de acuerdo a los requerimientos obtenidos.
4. Codificar el software en función al modelo desarrollado anteriormente.
5. Validar el software desarrollado con problemas bidimensionales típicos de elasticidad.
6. Documentar el software desarrollado.

1.3. JUSTIFICACIÓN DE LA INVESTIGACIÓN

Este software proporcionará reusabilidad, extensibilidad y garantizara un fácil mantenimiento, lo que proveerá a la comunidad científica tanto nacional como internacional, una aplicación robusta, que pueda no solo abarcar problemas de elasticidad, sino que con pequeñas adiciones pueda integrarse con otros métodos como FEM, FVM, FDM y formar un solo software usando los conceptos de herencia, abstracción y polimorfismo y extenderse para resolver otros problemas de ingeniería.

1.4. DELIMITACIÓN DE LA INVESTIGACIÓN

El presente trabajo está dirigido a desarrollar un software que implemente el BEM para resolver solo problemas bidimensionales de elasticidad (deformación plana y esfuerzos planos).

CAPÍTULO 2
BASES TEÓRICAS

2.1. ANTECEDENTES

Los trabajos expuestos a continuación, se usaron como base metodológica y documental para el desarrollo del presente trabajo de investigación.

LAGE CHRISTIAN (1996) [7], diseñó una librería de clases para dar soporte al desarrollo de software para elementos divisorios. En este artículo se discute la extensibilidad y la reutilizabilidad de la librería y da un ejemplo de su aplicación. También se explican todos los pasos seguidos para desarrollar el método y se bosqueja el diseño de la biblioteca de clases.

SALGADO, N. K., ALIABADI, M. H. y CALLAN, R. E. (1997) [8], emplearon POO y las reglas de inferencia para el diseño de mallas del BEM. Este trabajo forma parte de un diseño basado en la tolerancia al daño de los paneles rigidizadores de aeronaves y otros componentes tales como juntas de fuselaje. Un motor de inferencia orientado a objeto es utilizado para aplicar este juego de reglas a la base de conocimiento donde la representación simbólica de la configuración estructural es almacenada. La técnica fue puesta en práctica usando el lenguaje de programación C++.

SALGADO N. K. y ALIABADI M. H., (1999) [9], describen un sistema interactivo, integrado para el diseño de tolerancia de daño de los paneles rigidizadores de aeronaves. Las tareas apoyadas por el sistema se extienden, desde la definición inicial de la configuración de panel rigidizador hasta la elaboración de la fuerza residual y diagramas de crecimiento de grietas; incluyendo el diseño de malla y el

análisis de extensión incremental de grieta. La configuración estructural fue representada del modo orientado a objeto, usando un nivel alto de abstracción y en términos de componentes de objeto, como refuerzos, grietas y límites. Los datos de entrada son realizados interactivamente, por una interfaz Windows fácil de usar y las mallas de elemento divisorias son automáticamente diseñadas usando una estrategia basada en la reglas. El análisis de propagación de grieta es realizado con el BEM que tiene la solución en cuenta de problemas que implican grietas solas o múltiples en condiciones de modo variadas.

BOŘEK, P. y ZDENEK, B. (2001) [10], presentan una estructura general de un código de elemento finito orientado a objeto recientemente desarrollado, así como la razón principal que llevan al desarrollo del nuevo ambiente para unos cálculos de elemento finitos complejos. Las exigencias principales para el nuevo código de FEM son formuladas. El objetivo principal de este trabajo se apoya en la descripción total del diseño de programa y, en particular, en el ambiente desarrollado el cual es comparado con códigos existentes.

CUNHA, M., TELLES J. y COUTINHO, A. (2004) [2] presentaron un código de elemento divisorio en paralelo para la solución de problemas elastoestáticos 2D usando elementos lineales. El autor extrae el código original de un texto de referencia en el área [Boundary elements techniques: theory and applications in engineering, 1984]. El código (realizado en FORTRAN), fue examinado y reescrito para su correcto funcionamiento en sistemas de memoria compartida y distribuida usando bibliotecas estándares y portátiles: OpenMP, LAPACK y ScaLAPACK. El proceso de realización proporciona pautas para desarrollar aplicaciones paralelas del BEM, aplicable a muchos problemas científicos y técnicos. Experimentos numéricos muestran la eficacia del acercamiento propuesto.

MARCZAK R. J., (2006) [11] presenta una plantilla de integración numérica orientada al objeto basada en el lenguaje de programación C++. Su aplicación va dirigida a métodos de elementos finitos y divisorios, el diseño ofrece soporte para integrando de tipo escalar, vector o matriz, de modo que una declaración de programación sola sea capaz de integrar diferentes elementos como matrices y vectores. El integrando puede contener singularidades como las típicamente encontradas en métodos de elemento divisorios, permitiendo la evaluación tanto de integrales regulares como de singulares bajo la misma estructura de programación. El uso del diseño propuesto es ilustrado por algunas aplicaciones elementales así como elemento finito y extractos de código de elemento divisorios.

2.2. BASES TEÓRICAS

2.2.1. Método de elementos de frontera (BEM)

2.2.1.1. Introducción al método de los Elementos de Frontera (BEM)

Las ecuaciones diferenciales que describen los fenómenos físicos (elastomecanica, acústica, conducción de calor, etc.) sólo se pueden resolver analíticamente para una clase limitada de problemas, geometrías y condición de frontera simples, las otras más complejas, requieren de métodos numéricos para su solución aproximada [12].

Cuando el comportamiento de un problema físico puede ser expresado como EDP, existen varios métodos numéricos enfocados a resolverlos, entre los que se mencionan: FEM, FDM, FVM y BEM. Este último método, a pesar de que su desarrollo no fue sino hasta hace pocas décadas, ha recibido especial atención de la comunidad científica, no solo porque se ha convertido en un método importante para obtener la solución computacional de varios problemas físicos; sino, .que sus ventajas son tan considerables que ha generado entusiasmo a la comunidad técnica de obtener

soluciones asistidas por ordenadores flexibles y eficientes para una variedad de problemas técnicos [13].

2.2.1.2. Definición

Anteriormente llamado ecuación integral de frontera (BIE, por sus siglas en ingles), el BEM es un método de elementos divisorios, obtenido por la discretización de una ecuación integral que es matemáticamente equivalente a las EDP originales. La formulación esencial de las EDP y que es la base del BEM, consiste en una ecuación integral que es definida en la frontera del dominio y una integral que relaciona la solución de la frontera con la solución en puntos en el dominio.

En otras palabras, BEM es un método numérico computacional usado para resolver EDP que han sido formuladas como ecuaciones integrales, haciendo este método aplicable a diferentes áreas de ingeniería como: mecánica de fluidos, acústica, electromagnetismo, plasticidad y fracturas mecánicas [14].

2.2.1.3. Enfoque general del método

Aunque en las aplicaciones particulares puede variar, la implementación de BEM sigue el siguiente procedimiento estándar paso a paso:

1. **Discretización:** se genera la malla geométrica que representa el modelo.
2. **Ecuaciones de los elementos:** se calculan las ecuaciones para cada elemento discretizado.
3. **Ensamble:** se ensamblan las ecuaciones calculadas para cada elemento discretizado.
4. **Condiciones de frontera:** se aplican las condiciones de frontera a al sistema ensamblado.

5. **Solución:** se calcula la solución del sistema de ecuaciones.
6. **Procesamiento posterior:** representan los gráficos y/o reportes generados a partir de la solución.

2.2.1.4. Aplicaciones

BEM es empleado para resolver varios problemas físicos, alguno de ellos son: acústica, torsión de barras prismáticas no circulares, deflexión de membranas, flexión de placas simplemente apoyadas, transferencia de calor y flujo de fluidos [15].

2.2.1.5. BEM en el campo de la elasticidad

Con BEM es posible determinar las tracciones, desplazamientos y esfuerzos en la frontera y en los puntos internos de un dominio. Para establecer las ecuaciones integrales de frontera en elasticidad, es necesario indicar previamente la ecuación de equilibrio (Ec.1) y la constitutiva (Ec. 2) que rigen dicho fenómeno, además de las relaciones de deformación (Ec.3) y las tracciones en el contorno (Ec. 4).

$$\sigma_{ij,j} = -b_i \qquad \text{Ec. 1}$$

$$\sigma_{ij} = \frac{2\mu\nu}{(1-2\nu)}\delta_{ij}e_{ii} + \mu e_{ij} \qquad \text{Ec. 2}$$

$$e_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \qquad \text{Ec. 3}$$

$$t_i = \sigma_{ij} n_j \qquad \text{Ec. 4}$$

σB_{ijB} representa el campo de esfuerzos, bB_{iB} son las fuerzas de campo, δB_{ijB} es el delta de Kronecker, eB_{ijB} es la deformación, uB_{iB} es el desplazamiento, tB_{iB} es la tracción y nB_{jB} es el vector normal unitario apuntando hacia afuera del contorno Γ del dominio. Las propiedades μ y ν son el módulo de rigidez y la relación de Poisson, respectivamente.

Combinando la Ec.1 y la Ec..2 y sustituyendo la deformación de la Ec.3, se obtiene la ecuación de Navier (Ec. 5):

$$\mu \frac{\partial u_i}{\partial x_j \partial x_j} + \frac{\mu}{1-2\nu} \frac{\partial u_j}{\partial x_i \partial x_j} = -b_i \qquad \text{Ec. 5}$$

Un planteamiento adecuado de un problema con valores en la frontera, requiere que en cada porción de la frontera sea prescrita cualquiera de las dos condiciones, tracción $t_i = t_i$ sobre el contorno Γ_t o desplazamiento $u_i = u_i$ sobre el contorno Γ_u; donde $\Gamma_t \cup \Gamma_u = \Gamma$ es el contorno de solución del dominio Ω. Finalmente en una formulación del BEM basada en la ecuación integral de Somigliana, queda planteada una relación integral entre los desplazamientos u_i^p en un punto de colocación «*p*» los desplazamientos u_i y las tracciones t_i en todo el contorno Γ. Adicionalmente, las fuerzas de campo b_i, permanecen relacionadas a través de una integral de dominio, como sigue (Ec. 6):

$$c_{ij}^p u_i^p + \int_\Gamma H_{ij} u_i \, d\Gamma = \int_\Gamma G_{ij} t_i \, d\Gamma + \int_\Omega G_{ij} b_i \, d\Omega \qquad \text{Ec. 6}$$

Donde H_{ij} y G_{ij} son las soluciones fundamentales en términos de tracción y desplazamiento, c_{ij}^{p} es una constante geométrica, tal que $c_{ij}^{p} = 1$, si $p \notin \Omega$ y $c_{ij}^{p} = \frac{1}{2}$ en el caso de un contorno suave. En el caso de dos dimensiones las soluciones fundamentales son las expresadas en la Ec. 7 y Ec. 8 [16].

$$G_{ij} = \frac{1}{8\pi\mu(1-\nu)}\left[(4\nu - 3)\delta_{ij} \ln r + \frac{\partial r}{\partial x_i}\frac{\partial r}{\partial x_j}\right] \quad \text{Ec. 7}$$

$$H_{ij} = \frac{-1}{2\pi(1-\nu)r}\left(\frac{\partial r}{\partial n}\frac{\partial r}{\partial x_i}\frac{\partial r}{\partial x_j}\right) + \left[\frac{-(1-2\nu)}{4\pi(1-\nu)r}\left(\delta_{ij}\frac{\partial r}{\partial n} + \frac{\partial r}{\partial x_i}n_j - \frac{\partial r}{\partial x_j}n_i\right)\right] \quad \text{Ec. 8}$$

La distancia *r* está definida como (Ec. 9):

$$r = \left|x_i - x_i^p\right| \quad \text{Ec. 9}$$

2.2.2. Generalidades de ingeniería de software

2.2.2.1. Definición de ingeniería de software

La ingeniería de software es una disciplina que comprende todos los aspectos de la producción de software desde las etapas iníciales de la especificación del sistema hasta el mantenimiento de este después de que se utiliza [17].

La ingeniería de software considera los siguientes cuatro aspectos:

- *Un enfoque de calidad:* la calidad fomenta una cultura de mejora continua del proceso, y esta cultura es la que al final conduce al desarrollo de enfoques muy efectivos para la ingeniería de software.
- *El proceso:* el proceso del software es la base para el control de la gestión de los proyectos de software y establece el contexto en el cual se aplican los métodos técnicos, se generan los productos del trabajo (modelos, documentos, datos, reportes, formatos, etcétera), se establecen los fundamentos, se asegura la calidad, y garantiza que el cambio se manejara de forma apropiada.
- *Los métodos:* los métodos de la ingeniería del software se basan en un conjunto de principios básicos que gobiernan cada área de la tecnología e incluye actividades de modelado y otras técnicas descriptivas.
- *Las herramientas:* estas proporcionan el soporte automatizado o semi-automatizado para el proceso y los métodos. Cuando las herramientas de ingeniería de software se integran de forma que la información que cree una de ellas pueda usarla otra, se dice que se ha establecido un sistema para el soporte del desarrollo del software [18].

2.2.2.2. Objetivos de la ingeniería de software

- Poseer la capacidad para procesar transacciones con rapidez y eficiencia.
- Mejorar la calidad de los productos de software.
- Hacer seguimiento de los costos de mano de obra, bienes y gastos generales.
- Aumentar la productividad y trabajo de los ingenieros del software.

- Facilitar el control del proceso de desarrollo de software.
- Ampliar la comunicación y facilitar la integración de funciones individuales.
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.
- Definir una disciplina que garantice la producción y el mantenimiento de los productos software desarrollados en el plazo fijado y dentro del costo estimado [18].

2.2.2.3. Fases de la ingeniería de software

La ingeniería de software está dividida en las siguientes fases:

- *Análisis y especificación de los requerimientos:* consiste en analizar, entender y registrar el problema que el patrocinante está tratando de resolver. Los requerimientos son una descripción de las necesidades o deseos de un producto. Un conjunto de requerimientos en estado de madurez, debe ser necesario, conciso, completo, consistente, no ambiguo.
- *Diseño:* se crea la interfaz de usuario y la estructura del software.
- *Codificación:* se lleva a cabo la implementación física de las bases de datos, de los programas (codificación), prueba de unidad o procedimientos separados y prueba de subsistemas o integración.
- *Prueba del sistema:* los grupos de procedimientos de unidades probadas como elementos separados en la fase previa son comprobados como sistema durante la integración.
- *Instalación:* corresponde a la puesta en marcha de la aplicación.
- *Mantenimiento del sistema y mejoras:* corresponde al mantenimiento y mejoras futuras que deben contemplarse a la hora de implantar una

aplicación informática, una vez que el sistema está en funcionamiento [18].

2.2.3. Metodología OMT

2.2.3.1. Definición de OMT.

La metodología OMT (Object Modeling Technique) fue creada por James Rumbaugh y Michael Blaha en 1991, mientras James dirigía un equipo de investigación de los laboratorios General Electric.

OMT es una de las metodologías de análisis y diseño, más maduras y eficientes que existen en la actualidad. La gran virtud que aporta esta metodología es su carácter de abierta (no propietaria), que le permite ser de dominio público y, en consecuencia, sobrevivir con enorme vitalidad. Esto facilita su evolución para acoplarse a todas las necesidades actuales y futuras de la ingeniería de software [19].

2.2.3.2. Fases que conforman la metodología OMT.

- **Análisis.**

  El analista construye el modelo de dominio del problema, mostrando sus propiedades más importantes. El modelo de análisis es una abstracción resumida y precisa e lo que debe de hacer el sistema deseado y no de la forma en que se hará [19].

- **Diseño del sistema.**

  El diseñador toma decisiones de alto nivel sobre la arquitectura del software. Durante esta fase el software se organiza en subsistemas basándose tanto en la estructura de análisis como en la arquitectura propuesta [19].

- **Diseño de objetos.**

El diseñador construye el modelo de diseño basándose en el modelo de análisis, pero incorporando detalles de implementación. OMT describe en esta fase la forma en que el diseño puede ser implementado en distintos lenguajes (orientados y no orientados a objetos, bases de datos, entre otros) [19].

- **Implementación.**

  En esta fase se traduce finalmente todo lo desarrollado en las fases anteriores a una implementación concreta. Durante la fase de implementación es importante tener en cuenta los principios de la ingeniería de software de forma que la correspondencia con el diseño sea directa y el software implementado sea flexible y extensible [19].

2.2.4. Programación orientada a objetos (POO)

2.2.4.1. Definición de POO.

POO es una forma de programación que utiliza objetos, ligados mediante mensajes, para la solución de problemas. La programación orientada a objetos puede considerarse como una extensión natural de la programación estructurada en un intento de potenciar los conceptos de modularidad y reutilización del código [20].

2.2.4.2. Algunos conceptos generales en POO.

La idea fundamental en los lenguajes orientados a objetos es combinar en una sola unidad *datos y funciones que operan sobre estos datos.* Tal unidad se denomina *objeto (Figura 2.1).* Por consiguiente dentro de los objetos residen los datos de los lenguajes de programación tradicionales, tales como números, arreglos, cadenas y registros, así como las funciones o subrutinas que operan sobre ellos.

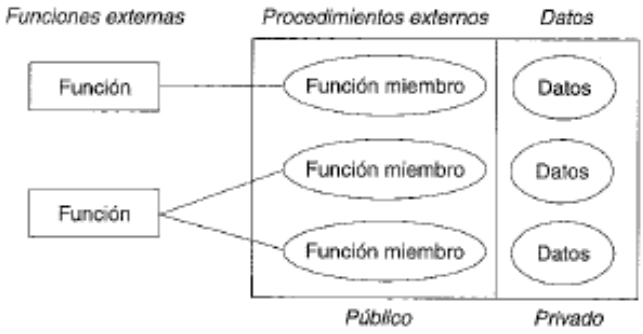

Figura 2.1. El modelo de objeto (Fuente:[20])

Otro concepto relacionado es *clase*, una clase es la descripción de un conjunto de objetos, en otras palabras consta de métodos y datos que resumen características comunes de un conjunto de objetos. Se pueden definir muchos objetos de la misma clase. Dicho de otro modo, una clase es la declaración de un tipo objeto.

La encapsulación es una característica muy potente, y junto con *la ocultación* de la información, representan el concepto avanzado de objeto, que adquiere su mayor relevancia cuando encapsula e integra datos, más las operaciones que manipulan los datos en dicha entidad. Sin embargo, la orientación a objetos se caracteriza, además de por las propiedades anteriores, por incorporar las características de *herencia*, propiedad que permite a los objetos ser construidos a partir de otros objetos. Dicho de otro modo, la capacidad de un objeto para usar estructuras de datos y los métodos previstos en antepasados o ascendientes. El objetivo final es la *reusabilidad o reutilización*, es decir, reutilizar el código anteriormente ya desarrollado.

Otra característica importante de los sistemas orientados a objetos es el *polimorfismo* y no es más que la capacidad de utilizar funciones virtuales y ejecutar sobrecarga lo cual permite desarrollar sistemas en los que objetos diferentes puedan responder de modo diferente al mismo mensaje [20].

2.2.4.3. Ventajas de la orientación a objetos

Las ventajas de la orientación a objetos se pueden resumir básicamente en [20]:

- La posibilidad de reflejar sucesos del mundo real mediante tipos abstractos de datos extensibles a objetos.
- La división del código en clases, que contribuye a imponer una estructura a medida que crecen los programas.
- La posibilidad de ocultar datos también genera sistemas más seguros.

2.2.5. Lenguaje de programación C++

2.2.5.1. Definición

C++ es un lenguaje de programación de programación general basado en el lenguaje de programación C. El lenguaje de programación C++ ha sido diseñado para: ser mucho mejor que C, soportar abstracción de datos y soportar programación orientada a objetos [21].

2.2.5.2. Características

Las características comunes de las nuevas versiones de C++ son [20]:

- C++ es esencialmente un superconjunto de C ANSI.
- C++ tiene las mismas características de tipificación que C ANSI para propiedades no orientadas a objetos.
- Los compiladores de C++ aceptan normalmente código escrito en la versión original de K&R (Kernighan y Ritchie). Generalmente, los compiladores de C++ proporcionan mensajes de error o advertencia cuando el código C no tiene prototipos.

- Desde el punto de vista especifico de sintaxis, algunas características de C han sido mejoradas notablemente.

2.2.5.3. Definición de Modelo

Un modelo es el que nos proporciona los planos de un sistema, desde los más generales, que proporcionan una visión general del sistema, hasta los más detallados. En un modelo se han de incluir los elementos que tengan más relevancia y omitir los que no son interesantes para el nivel de abstracción que se ha elegido. También se puede decir que, un modelo es una simplificación de la realidad [22].

2.2.5.4. Objetivos de la creación de modelos.

- Los modelos ayudan a visualizar cómo es lo queremos que sea un sistema.
- Los modelos permiten especificar la estructura o el comportamiento de un sistema.
- Los modelos proporcionan plantillas que nos guían en la construcción de un sistema.
- Los modelos documentan las decisiones que hemos adoptado.

En la realidad, no siempre se hace un modelado formal, la probabilidad de que exista un modelado formal para abordar un sistema es inversamente proporcional a la complejidad del mismo, esto es, cuanto más fácil sea un problema, menos tiempo se pasa modelándolo y esto es porque cuando hay de aportar una solución a un problema complejo el uso del modelado nos ayuda a comprenderlo, mientras que cuando tenemos un problema fácil el uso del modelado que hacemos se reduce a representar mentalmente el problema [23].

2.2.6. Lenguaje unificado de modelado (UML)

2.2.6.1. Definición de UML

El lenguaje unificado de modelado (UML) data desde 1994. UML es un lenguaje estándar que sirve para escribir los *planos del software*, puede utilizarse para visualizar, especificar, construir y documentar todos los artefactos que componen un sistema con gran cantidad de software. UML puede usarse para modelar desde sistemas de información hasta aplicaciones distribuidas basadas en Web, pasando por sistemas empotrados de tiempo real.

UML es un lenguaje por que proporciona un vocabulario y las reglas para utilizarlo, además es un lenguaje de modelado lo que significa que el vocabulario y las reglas se utilizan para la representación conceptual y física del sistema.

UML es un lenguaje que nos ayuda a interpretar grandes sistemas mediante gráficos o mediante texto obteniendo modelos explícitos que ayudan a la comunicación durante el desarrollo ya que al ser estándar, los modelos podrán ser interpretados por personas que no participaron en su diseño (e incluso por herramientas) sin ninguna ambigüedad. En este contexto, UML sirve para *especificar*, modelos concretos, no ambiguos y completos.

UML proporciona la capacidad de modelar actividades de planificación de proyectos y de sus versiones, expresar requisitos y las pruebas sobre el sistema, representar todos sus detalles así como la propia arquitectura. Mediante estas capacidades se obtiene una documentación que es válida durante todo el ciclo de vida de un proyecto [22].

2.2.6.2. Elementos de UML

Actor: es una entidad externa (de fuera del sistema) que interacciona con el sistema participando (y normalmente iniciando) en un caso de uso. Los actores pueden ser gente real (por ejemplo, usuarios del sistema), otros ordenadores o eventos externos. Los actores no representan a personas ***físicas*** o a sistemas, sino su ***papel***. Esto significa que cuando una persona interacciones con el sistema de diferentes maneras, estará representado por varios actores.

Clases: en una clase se agrupan todos los objetos que comparten los mismos atributos, métodos y relaciones. Los atributos son características y propiedades comunes en todos los objetos de la clase. Los métodos son operaciones que deben cumplir las instancias de la clase. Las clases se representan como un rectángulo donde figuran el nombre de la clase, sus atributos y sus métodos.

Artefacto: es una información que es utilizada o producida mediante un proceso de desarrollo de software. Pueden ser artefactos un modelo, una descripción o un software. Los artefactos de UML se especifican en forma de diagramas, éstos, junto con la documentación sobre el sistema constituyen los artefactos principales que el modelador puede observar.

Estado: Los estados son los ladrillos de los diagramas de estado. Un estado pertenece a exactamente una clase y representa un resumen de los valores y atributos que puede tener la clase. Un estado UML describe el estado interno de un objeto de una clase particular.

Actividad: es un único paso de un proceso.

2.2.6.3. Diagramas de UML

Un diagrama es una representación gráfica de una colección de elementos del modelo, que habitualmente toma forma de grafo donde los arcos que conectan sus vértices son las relaciones entre los objetos y los vértices se corresponden con los elementos del modelo.

Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibuja de forma que se resaltan los detalles necesarios para entender el sistema [23]**.**

2.2.6.3.1. Diagramas de casos de uso.

Es un diagrama que muestra un conjunto de casos de uso con sus relaciones y los actores implicados. Este diagrama sirve para modelar la vista estática de un programa. La vista estática nos permite visualizar el comportamiento externo del programa; de esta forma se consigue conocer qué es lo que debe hacer el programa independientemente de cómo lo haga y sabremos los elementos que interactúan con el sistema.

Los elementos implicados en un diagrama de casos de uso son los casos de uso, las relaciones y los actores. Las relaciones y los casos de uso ya han sido explicados anteriormente y el papel del actor también ha sido comentado pero merece la pena detallarlo más: un actor es un rol que interactúa con el sistema. Se define como rol porque un actor puede ser tanto un usuario de la aplicación como otro sistema o dispositivos externos.

Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona una respuesta a eventos que se producen en el

mismo. En este tipo de diagrama intervienen algunos conceptos nuevos: un actor es una entidad externa al sistema que se modela y que puede interactuar con él; un ejemplo de actor podría ser un usuario o cualquier otro sistema. Las relaciones entre casos de uso y actores pueden ser las siguientes:

- Un actor se comunica con un caso de uso.
- Un caso de uso extiende otro caso de uso.
- Un caso de uso usa otro caso de uso.

2.2.6.3.1. Diagramas de secuencia.

Estos diagramas muestran la secuencia de mensajes que se van lanzando los objetos implicados en una determinada operación del programa. Dentro del diagrama los objetos se alinean en el eje X respetando su orden de aparición. En el eje Y se van mostrando los mensajes que se envían, también respetando su orden temporal. Cada objeto tiene una línea de vida donde se sitúa su foco de control. El foco de control es un rectángulo que representa el tiempo durante el que un objeto está activo ejecutando una acción. Con este sencillo esquema podemos visualizar la comunicación y sincronización bajo un estricto orden temporal de los objetos implicados en las distintas funcionalidades de un sistema.

2.2.6.3.2. Diagrama de clases de análisis.

Es utilizado por los desarrolladores de software para determinar los requerimientos funcionales, considerando una o varias clases, o sub-sistemas del sistema a desarrollar. Los casos de uso se describen mediante clases de análisis y sus objetos. El diagrama de clases de análisis se construye examinando los casos de usuarios, cerrando sus reacciones e identificando los roles de los clasificadores.

2.2.6.3.3. Diagrama de clases de diseño.

Se emplean para modelar la estructura estática de las clases en el sistema, sus tipos, sus contenidos y las relaciones que se establecen entre ellos. A través de este diagrama se definen las características de cada una de las clases, interfaces, colaboraciones y relaciones de dependencia y generalización**.**

2.2.6.3.4. Diagramas de actividad.

Son similares a los diagramas de flujo de otras metodologías orientadas a objetos. En realidad se corresponden con un caso especial de los diagramas de estado donde los estados son estados de acción (estados con una acción interna y una o más transiciones que suceden al finalizar esta acción, o lo que es lo mismo, un paso en la ejecución de lo que será un procedimiento) y las transiciones vienen provocadas por la finalización de las acciones que tienen lugar en los estados de origen.

Siempre van unidos a una clase o a la implementación de un caso de uso o de un método (que tiene el mismo significado que en cualquier otra metodología OO). Los diagramas de actividad se utilizan para mostrar el flujo de operaciones que se desencadenan en un procedimiento interno del sistema**.**

2.2.6.3.5. Diagramas de implementación.

Se derivan de los diagramas de proceso y módulos de la metodología de Booch, aunque presentan algunas modificaciones. Los diagramas de implementación muestran los aspectos físicos del sistema. Incluyen la estructura del código fuente y la implementación, en tiempo de implementación. Existen dos tipos:

- **Diagramas de componentes:** Muestra la dependencia entre los distintos componentes de software, incluyendo componentes de código fuente, binario y ejecutable. Un componente es un fragmento de código software

(un fuente, binario o ejecutable) que se utiliza para mostrar dependencias en tiempo de compilación.

- **Diagrama de plataformas o despliegue:** Muestra la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un nodo es un objeto físico en tiempo de ejecución, es decir una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formado por otros componentes.

2.2.6.3.6. Modelo de dominio.

Es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea construcción del modelo de dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Similares a los mapas mentales utilizados en el aprendizaje, el modelo de dominio es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir.

- **Diagrama de Paquetes**: se usan para reflejar la organización de paquetes y sus elementos. Cuando se usan para representaciones, los diagramas de paquete de los elementos de clase se usan para proveer una visualización

de los espacios de nombres. Los elementos contenidos en un paquete comparten el mismo espacio de nombre, el hecho de compartir espacios de nombres requiere que los elementos contenidos en un espacio de nombre específico tengan nombres únicos. Los paquetes se pueden construir para representar relaciones tanto físicas como lógicas.

CAPÍTULO 3
FASE DE ANÁLISIS

3.1. INTRODUCCIÓN

En la fase de análisis de OMT se busca crear un modelo general del sistema, con el objeto de comprender el alcance y desarrollar modelos preliminares que representen el comportamiento del software, esto con el fin de obtener un software que sea efectivo, eficiente, flexible, amigable y dinámico al usuario final [23].

En este capítulo se especifica el modelo general del software diseñado.

3.2. REQUISITOS DEL SOFTWARE.

Para definir los requisitos del software, se realizó un estudio de las exigencias de los usuarios finales con respecto al software, y luego se clasificaron en función de establecer prioridades entre ellas.

En la Tabla 3.1 y 3.2 se presentan respectivamente los requisitos funcionales y no funcionales del software diseñado.

Tabla 3.1. Requerimientos funcionales (RF) del software. (Fuente: Propia)

Ref.	Requerimiento
RF1	El sistema debe permitir al usuario cargar un problema de elasticidad por medio de un archivo.
RF2	El sistema debe mostrar en forma grafica el modelo de un problema de elasticidad cargado.
RF3	El sistema debe mostrar una secuencia grafica al usuario sobre el proceso deformativo del modelo geométrico cuando este sea resuelto.

Tabla 3.1. Requerimientos funcionales (RF) del software (Cont).(Fuente: propia)

Ref.	Requerimiento
RF4	El sistema debe permitir al usuario, ampliar, reducir y desplazar el modelo 2D generado a partir del problema cargado.
RF7	El sistema debe permitir al usuario el poder repetir la secuencia de deformativa del modelo sin necesidad de realizar de nuevo de los cálculos.
RF8	El sistema debe permitir al usuario mostrar el modelo original y el modelo deformado.
RF9	El sistema debe ser capaz de generar reportes con información seleccionada por el usuario y mostrarlos por pantalla o exportarlos como archivos.
RF10	El sistema debe indicar al usuario el estatus del cálculo de un problema de elasticidad cargado cuando este último se esté siendo procesando por el sistema.
RF11	El software debe poseer una ayuda amigable al usuario.

Tabla 3.2. Requerimientos no funcionales (RNF) del software. (Fuente: propia)

Ref.	Requerimiento
RNF1	El sistema debe poseer una interfaz gráfica, amigable e intuitiva que permita la fácil interacción con el usuario.
RNF3	El sistema debe estar diseñado con una arquitectura que pueda adaptarse fácilmente a cualquier cambio y mejora.
RNF4	BEM debe estar desarrollado en C++.
RNF5	El sistema debe ser multiplataforma.

3.3. DIAGRAMA DE DOMINIO

En la Figura 3.1 se muestra el diagrama de dominio del software diseñado, en este se establecieron los conceptos principales del dominio y su relación entre ellos. En la Tabla 3.3 se definen cada uno de los términos presentados en el diagrama de dominio de la Figura 3.1.

Tabla 3.3. Glosario de términos del diagrama de dominio. (Fuente: propia)

TÉRMINO	DEFINICIÓN
Usuario	Representa a cualquier persona que utilice el software para resolver problemas 2D de elasticidad usando BEM.
PuntosInternos	Son los lugares no pertenecientes a la frontera donde se quiere conocer información de desplazamientos o esfuerzos. Se diferencia de los nodos por no pertenecer a la frontera.
CondicionesDeFrontera	En un problema de elasticidad, son características definidas de un punto o área del modelo geométrico 2D a evaluar. En otras palabras, representan fuerzas o desplazamientos que existen y para el que se conoce su valor.
Elementos	En un problema elástico, representan agrupaciones de nodos y de condiciones de frontera. Las condiciones de frontera aplicadas a los elementos, deben ser distribuidas entre los nodos que son agrupados por el elemento al cual se le aplican. Los elementos generalmente suelen ser de tres o más nodos formando curvas.

Tabla 3.3. Glosario de términos del diagrama de dominio (Cont.).(Fuente: propia)

TÉRMINO	DEFINICIÓN
Nodos	Son aquellos puntos en el problema elástico que señalan un lugar en el modelo geométrico 2D. Representan el área de menor tamaño analizable.
Material	Representa el (los) material (es) del que está hecho el modelo geométrico 2D real. Ejemplos: Hierro, Plomo, Concreto, Aluminio.
BEM	Es un sistema compacto conformado por procedimientos y métodos numéricos especializados en diferentes áreas de la física. Representa en el software el ente procesador de los datos de un problema 2D de elasticidad.
ArchivosDeResultados	Representa los archivos que son generados como respuesta de un análisis completo del problema 2D de elasticidad cargado en el software. Aquí se especifican en un archivo plano la data generada con proceso de cálculo.
MicrosoftExcel	Es un programa de hojas cálculo desarrollado por la Microsoft Corporation, en donde es cargada la data de los archivos generados por BEM.
Graficos	Son los gráficos generados por la aplicación Excel® en base a la data cargada y las hojas del problema.

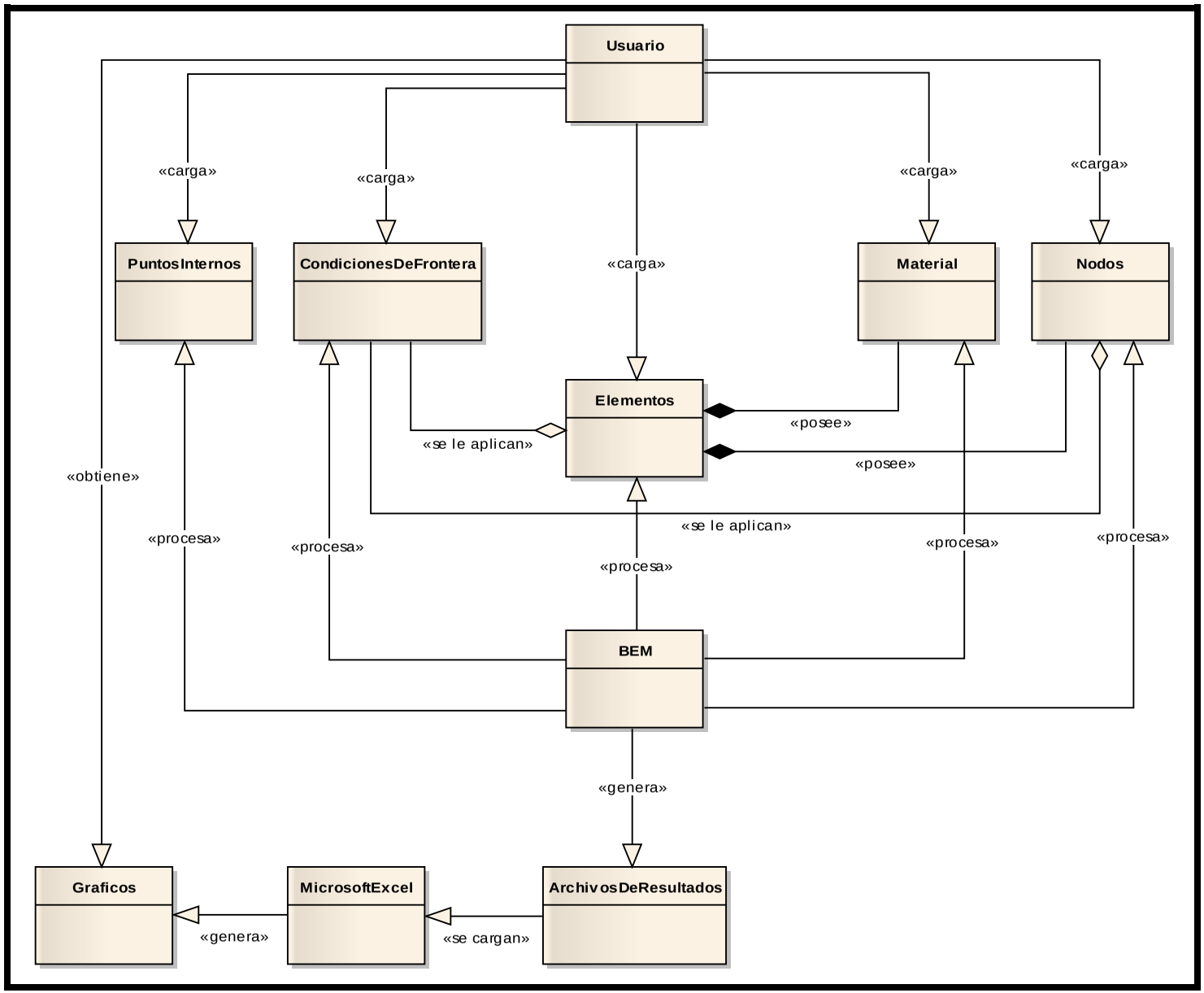

Figura 3.1. Diagrama de dominio. (Fuente: propia)

3.4. MODELO DE CASOS DE USO

3.4.1. Actores del software.

Basado en los requerimientos del software, se determinaron los usuarios del software, obteniéndose dos tipos de actores: **usuario** y **Parametros de BEM**. En la Tabla 3.4 se definen los actores.

Tabla 3.4. Actores del software. (Fuente: propia)

ACTOR	FUNCIÓN
Usuario	Representa cualquier usuario del software, es decir, es quien utiliza el software para resolver problemas 2D de elasticidad, además el usuario tiene la posibilidad de pedir reportes de los resultados obtenidos.
Parametros de BEM	Representa los parámetros estructurados que son pasados al módulo que resuelve el problema de elasticidad, este módulo posee la implementación del método y se encarga de resolver un problema cargado por el usuario, entregando al sistema el resultado de los cálculos.

3.4.2. Diagrama de casos de uso general

Los casos de uso definidos según las funciones principales del software fueron los siguientes:

- **Cargar Datos.**
- **Generar Solución Problema.**
- **Generar Reportes.**

En la Figura 3.2 se presentan el diagrama de casos de uso general del software y en la Tabla 3.5 se muestra la descripción de los tres casos de uso definidos.

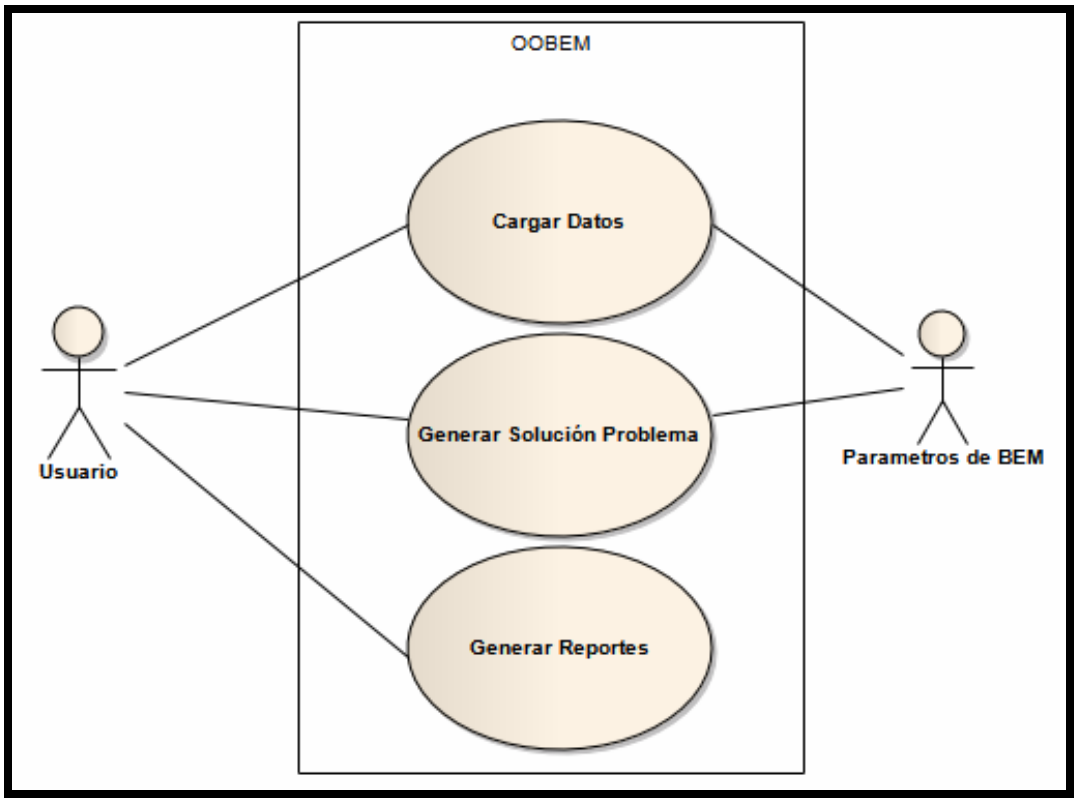

Figura 3.2. Diagrama de casos de uso general del software. (Fuente: propia)

Tabla 3.5. Descripción de los casos de uso generales del software. (Fuente: propia)

CASO DE USO	DESCRIPCIÓN
Cargar Datos	El caso de uso **"Cargar Datos"** es el que hace referencia al flujo de eventos necesarios para cargar un problema en el software OOBEM desde un archivo, el cual contiene todos los componentes de entrada por parte del usuario los cuales serán validados para evitar errores en el cálculo de los resultados. Luego de validarlos se mostrara un gráfico del modelo geométrico del problema elástico 2D planteado.

Tabla 3.5. Descripción de los casos de uso generales del software (Cont.). (Fuente: propia)

CASO DE USO	DESCRIPCIÓN
Generar Solución Problema	El caso de uso "Generar Solución Problema", hace referencia al flujo de eventos que se necesitan para, teniendo los datos del problema elástico 2D, conseguir los resultados aplicando BEM. Luego de tener los resultados se generara una secuencia grafica del proceso deformativo del modelo geométrico 2D inicial.
Generar Reportes	El caso de uso **"Generar Reportes"**, hace referencia al flujo de eventos que se necesitan para solicitar reportes de datos del problema. Los reportes pueden ser solicitados para ser vistos por pantalla o para ser exportados como archivos.

3.4.2.1. Caso de uso "Cargar Datos".

Objetivo en el contexto: el usuario desea cargar un nuevo problema 2D de elasticidad en el software.
Alcance: aplicación.

Actores: usuario

Descripción: Este caso de uso, hace referencia al flujo de eventos necesarios para cargar y validar un problema en el sistema OOBEM desde un archivo de datos. En la Figura 3.3 se observa el diagrama de casos de uso "**Cargar Datos**".

Pre-condiciones:

- Si se desea cargar un nuevo problema y se está trabajando sobre un problema anteriormente cargado, se debe pedir confirmación del usuario sobre su deseo de cargar el nuevo problema 2D de elasticidad.
- No debe existir un problema cargado, de no ser así, la data del problema elástico 2D actualmente cargado deberá eliminarse.

Condición final de éxito:

El software carga en memoria el problema 2D de elasticidad seleccionado y la verificación de errores da satisfactoria dejando el problema listo para resolverse.

Condición Final de Fallo:

El problema no se cargó debido a problemas en la validación.

Disparador:

Se recibe una solicitud para cargar un problema 2D de elasticidad por parte del usuario.

Requisitos Cubiertos (según Tabla 3.1):

- RF1
- RF2

Flujo de eventos:

- ***Flujo principal:***

1. El usuario indica al software su deseo de cargar un problema 2D de elasticidad por medio de un archivo de datos.
2. El software verifica si hay un problema cargado.
3. El software lee el archivo de datos seleccionado por el usuario y lo valida.
4. Se cargan los datos del problema 2D de elasticidad en el software.
5. Se muestra el modelo geométrico 2D planteado en el problema 2D de elasticidad.
6. Finaliza el caso de uso.

- ***Flujo alterno:***

Línea 2: *Existe un problema cargado actualmente en el sistema.*

1. Se le pregunta al usuario si está seguro de querer cargar el nuevo problema.
2. Se elimina toda la data del problema actualmente cargado.
3. Se pasa al paso 3 del flujo principal.

Línea 4: *Los datos no son validos.*

1. Se le notifica al usuario que el archivo presenta inconsistencia de datos.
2. Se cancela la carga del archivo.
3. Se pasa al paso 6 del flujo principal.

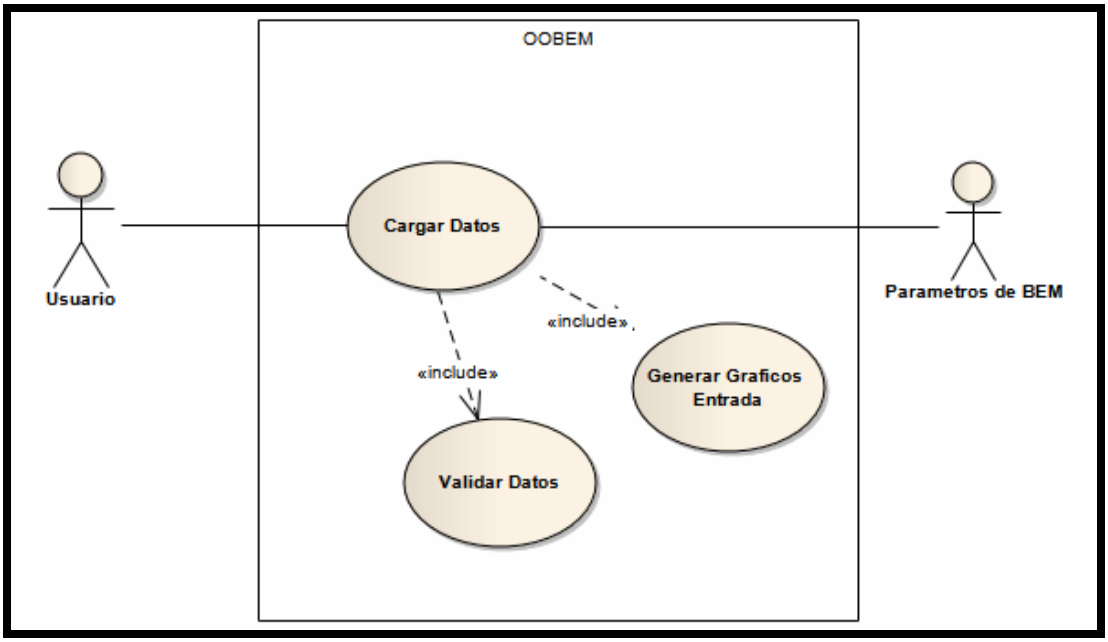

Figura 3.3. Diagrama de casos de uso "Cargar Datos" del software. (Fuente: propia)

3.4.2.2. Caso de uso "Generar Solución Problema".

Objetivo en el contexto: el usuario desea resolver el problema cargado en el sistema.

Alcance: aplicación.

Actores: usuario

Descripción: el caso de uso "**Generar Solución Problema**", representa el proceso por medio del cual el usuario solicita al software que calcule la solución para el problema 2D de elasticidad cuyos datos han sido cargados y genere una secuencia grafica sobre el proceso deformativo del modelo descrito en él. En la Figura 3.4 se observa el diagrama de casos de uso "Generar Solución Problema".

Pre-condición:

Deberá existir un problema en cargado en el sistema.

Condición final de éxito:

Se resuelve el problema 2D de elasticidad, se genera una secuencia que muestra las deformaciones sufridas del modelo geométrico 2D representado por el problema.

Condición final de fallo:

El problema no se resuelve, ni se generan secuencias gráficas de los resultados.

Disparador:

Se recibe una solicitud para resolver un problema 2D de elasticidad por parte del usuario.

Requisitos Cubiertos (según Tabla 3.1):

- RF3
- RF4
- RF7
- RF8
- RF10

Flujo de eventos:

- Flujo principal:

1. El usuario indica a la aplicación que calcule la solución para un problema 2D de elasticidad cargado por medio de un archivo de datos.
2. La aplicación verifica si hay un problema cargado en el sistema.
3. La aplicación estructura la data y prepara las variables de resultados.
4. Se pasan los parámetros a BEM encargado de calcular la solución para el problema cargado.
5. El módulo al terminar los cálculos devuelve a la aplicación los resultados obtenidos.
6. Se muestra una secuencia gráfica del modelo geométrico deformándose según los resultados obtenidos.
7. Finaliza el caso de uso.

- ***Flujo alterno:***

Línea 2: *No existe un problema cargado actualmente en el software.*

1. Se le indica al usuario que debe cargar un problema antes.
2. Se pasa al paso 7 del flujo principal.

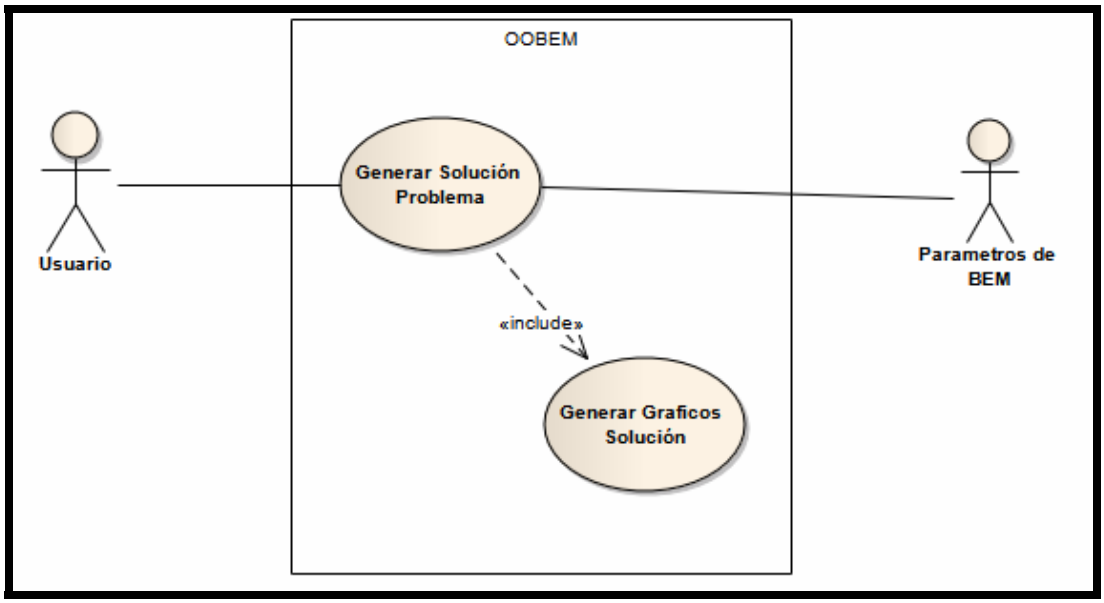

Figura 3.4. Diagrama de casos de uso "Generar Solución Problema". (Fuente: propia)

3.4.2.3. Casos de uso "Generar Reportes".

Objetivo en el contexto: el usuario desea generar reportes de los datos usados para calcular la solución como de los datos de la solución misma. Los datos pueden ser para verlos por pantalla o para exportarlos como archivos.

Alcance: aplicación.

Actores: usuario

Descripción: el caso de uso "Generar Reportes", es el encargado de construir los reportes que son solicitados por parte del usuario. Los informes construidos pueden ser mostrados por pantalla o exportados de forma persistente como archivos. La data contenida en los reportes puede ser variada. En la Figura 3.5 se observa el diagrama de casos de uso "Generar Solución Problema".

Pre-condición:

Al menos, deberá existir un problema en cargado en el software. Aunque exista información que no será posible solicitar en los reportes hasta que se halla calculado la solución del problema elástico cargado.

Condición final de éxito:

El usuario ha seleccionado la información que desea para el reporte y la forma como quiere el reporte y el reporte de ha generado correctamente.

Condición final de fallo:

Ninguna.

Disparador:

Se recibe una solicitud por parte del usuario para generar reporte de un problema 2D de elasticidad.

Requisitos Cubiertos (según Tabla 3.1):

- RF9

Flujo de Eventos:

- ***Flujo principal:***

1. El usuario indica al software su deseo de generar un reporte de un problema 2D de elasticidad cargado por medio de un archivo de datos.
2. El software verifica si hay un problema cargado en el sistema.
3. El software verifica si se ha calculado la solución para el problema 2D de elasticidad cargado.
4. El software adapta la interfaz a las verificaciones anteriores y muestra una interfaz grafica de personalización de reportes.
5. El usuario selecciona el tipo de reporte deseado y la data que desea en él.
6. El reporte es generado.
7. El usuario indica que desea cerrar la interfaz para la construcción de reportes.
8. Finaliza el caso de uso.

- ***Flujo alterno:***

Hasta la línea 6: *El usuario cancela la petición del reporte*

1. Se pasa al paso 7 del flujo principal.

- ***Puntos de Extensión:***

Línea 6: *Se ha seleccionado como el tipo de reporte el que debe mostrarse por pantalla.*

1. La aplicación muestra una interfaz de usuario en donde se muestra de manera tabular la información solicitada por el usuario.
2. El usuario indica que desea cerrar la interfaz de usuario donde se mostro el reporte.
3. Se pasa al paso 7 del flujo principal.

Línea 6: *Se ha seleccionado como tipo de reporte el que debe exportarse como archivos.*

1. La aplicación solicita al usuario el directorio donde desea guardar los archivos generados.
2. La aplicación genera los archivos con la data solicitada.
3. Se pasa al paso 7 del flujo principal.

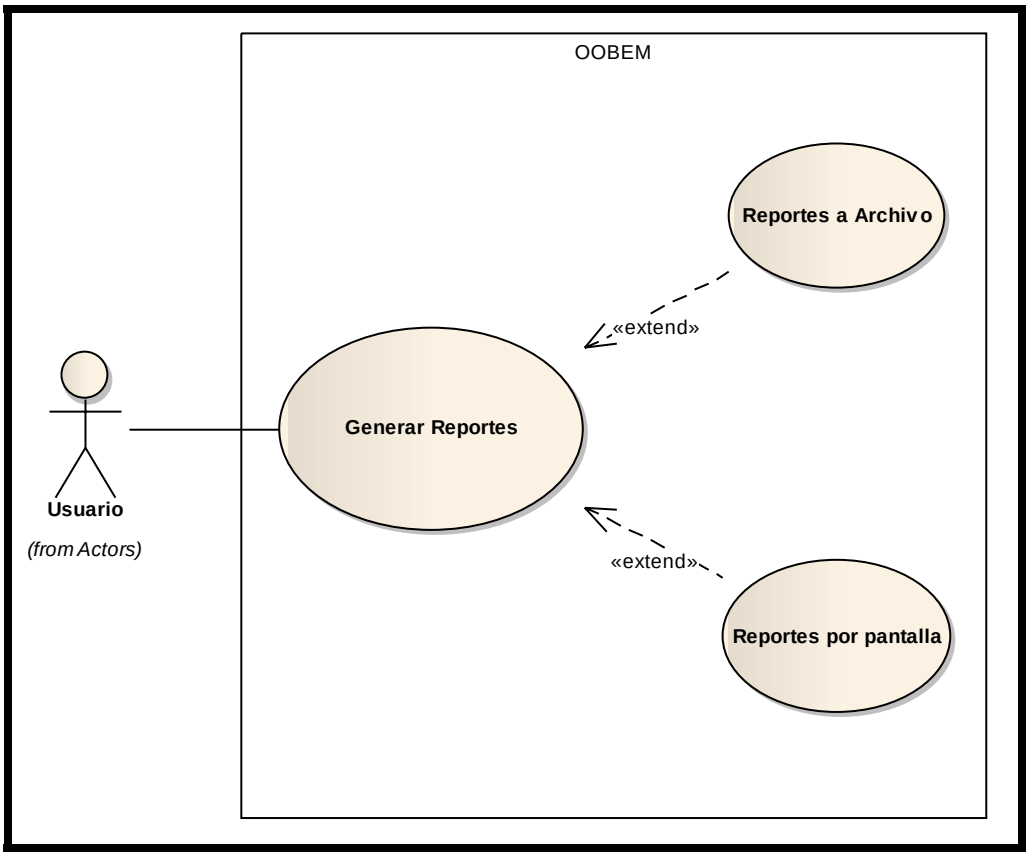

Figura 3.5. Diagrama de casos de uso "Generar Reportes". (Fuente: propia)

3.5. DIAGRAMAS DE CLASES DE ANÁLISIS.

Estos diagramas plasman las posibles clases del diseño, encajándolas en los tres estereotipos básicos: de interfaz, de control y de entidad. Cada una de ellas implica una semántica específica, la cual constituyó un método consistente para identificar y describir cada clase, contribuyendo a la creación de un modelo de objetos y una arquitectura robusta.

3.5.1. Diagrama de clases de análisis de “Cargar Datos”

El proceso **“Cargar Datos”** es el encargado de leer los datos de un problema 2D de elasticidad, validarlos, y en caso de ser validos generar una gráfica del modelo geométrico 2D planteado. De lo anterior se puede determinar que aquí intervienen la interfaz principal del software, que interviene una unidad de control encargada de leer los datos de una entidad que representa el archivo seleccionado por el usuario, luego esta otra unidad de control encargada de la validación de los datos leídos por la unidad de control cargar y la unidad de control que grafica que es común del sistema y que se encarga de realizar gráficos.

En la Figura 3.6, se puede ver el diagrama de clases de análisis de “**Cargar Datos**” y la conexión entre los componentes involucrados.

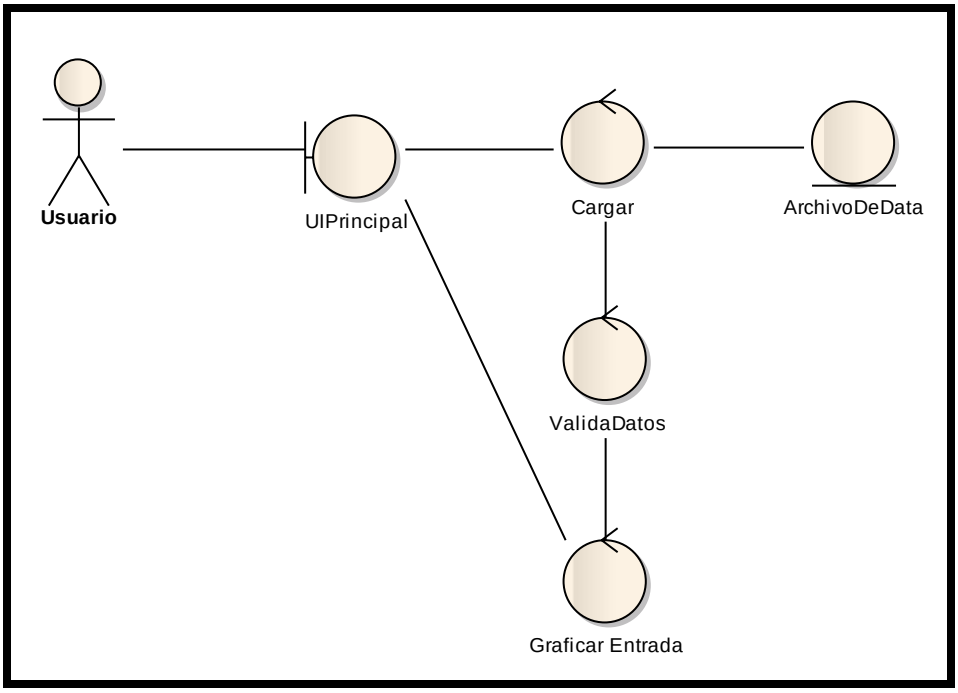

Figura 3.6. Diagrama de clases de análisis de "Cargar Datos". (Fuente: propia)

3.5.2. Diagrama de clases de análisis de “Generar Solución Problema”

En el proceso **“Generar Solución Problema”** es el encardo de resolver un problema 2D de elasticidad cargado mediante el proceso **“Carga Datos”**. En este proceso se invoca a BEM, que es quien abarca todos los procedimientos y métodos que conforman el método de elementos de fronteras. De lo anterior se determino que aquí intervienen la interfaz principal del software, una unidad de control que se encarga de transferir los datos del problema al modulo BEM y una unidad de control que es común al software encargada de graficar.

En la Figura 3.7, se puede ver el diagrama de clases de análisis de “**Generar Solución Problema**” y la conexión entre los componentes involucrados.

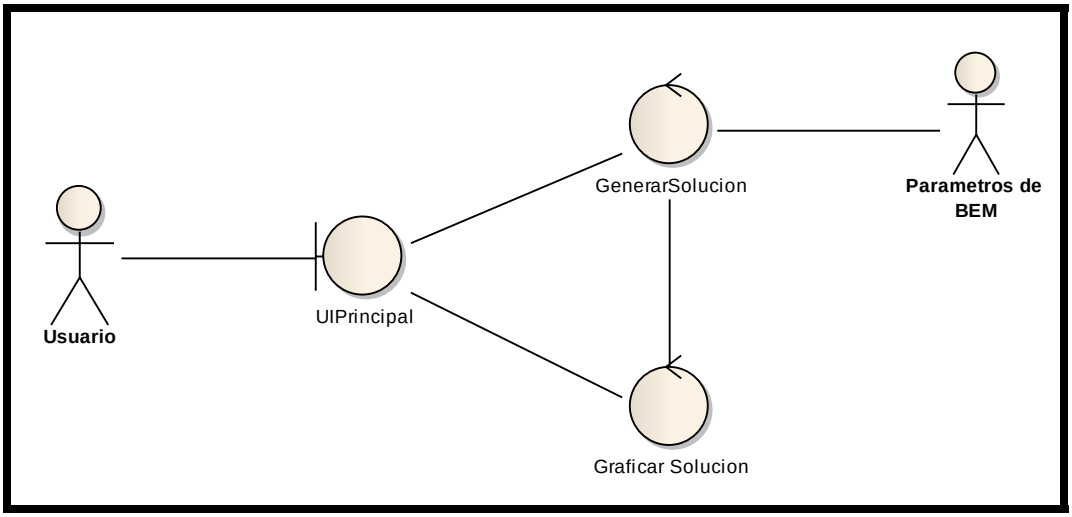

Figura 3.7. Diagrama de clases de análisis "Generar Solución Problema".
(Fuente: propia)

3.5.3. Diagrama de clases de análisis de “Generar Reportes”

El proceso “**Generar Reportes**”, es el encargado de generar los reportes solicitados por el usuario. Los reportes son personalizables, y pueden ser mostrados por pantalla o exportados como archivos. De lo anterior se puede determinar que aquí interviene la interfaz principal del software, una unidad de control que se encarga de

la gestión de los reportes y de mostrar la interfaz para su creación, una unidad de control para gestionar los reportes que son generados para ser mostrados por pantalla y otra unidad de control para los que son generados para exportarse como archivos, se tiene además una entidad que representa los archivos generados y una interfaz que sirve para mostrar los datos cuando se ha elegido mostrar los datos por pantalla.

En la figura 3.8, se puede ver el diagrama de clases de análisis de "**Generar Reportes**" y la conexión entre los componentes involucrados.

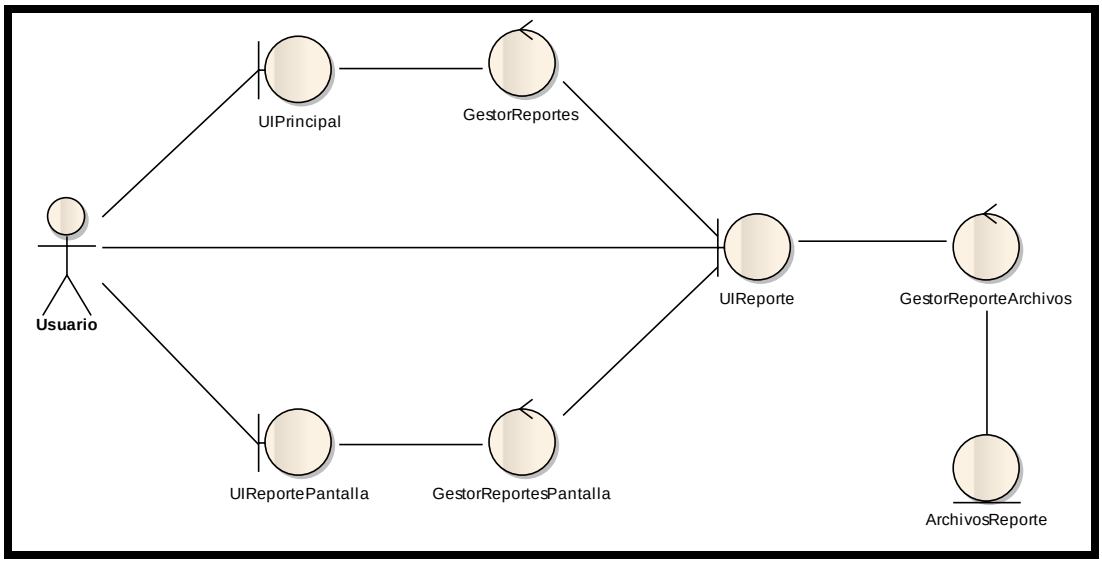

Figura 3.8. Diagrama de clases de análisis "Generar Reportes". (Fuente: propia)

3.5.4. **Diagrama de clases de análisis de "Parametros de BEM"**.

El diagrama de clases de análisis Parametros de BEM, muestra que es lo que existe o a lo que está vinculado el actor Parametros de BEM, esta estructura seria la que instancie los datos transmitidos y calcule la solución del problema. En este diagrama de pueden apreciar: 7 unidades de control, "Domain" que es encargada de estructurar el dominio y de su resolución, "JacobiMethod", "KelvinMethod", "gaussLU", representan métodos numéricos usados en el sistema, y "Logger", que representa un procesador a archivos que generar un log de datos internos, errores o

advertencias. Además de esto se pueden apreciar 5 elementos tipo entidad, estos representan el almacén de data y cada uno el almacén para un componente determinado (condiciones de borde, nodos, elementos, entre otros). La interfaz JNIInterface, permite al actor "Parámetros de BEM" interactuar con los componentes ya descritos anteriormente.

En la figura 3.8, se puede ver el diagrama de clases de análisis de "**Parametros de BEM**" y la conexión entre los componentes involucrados.

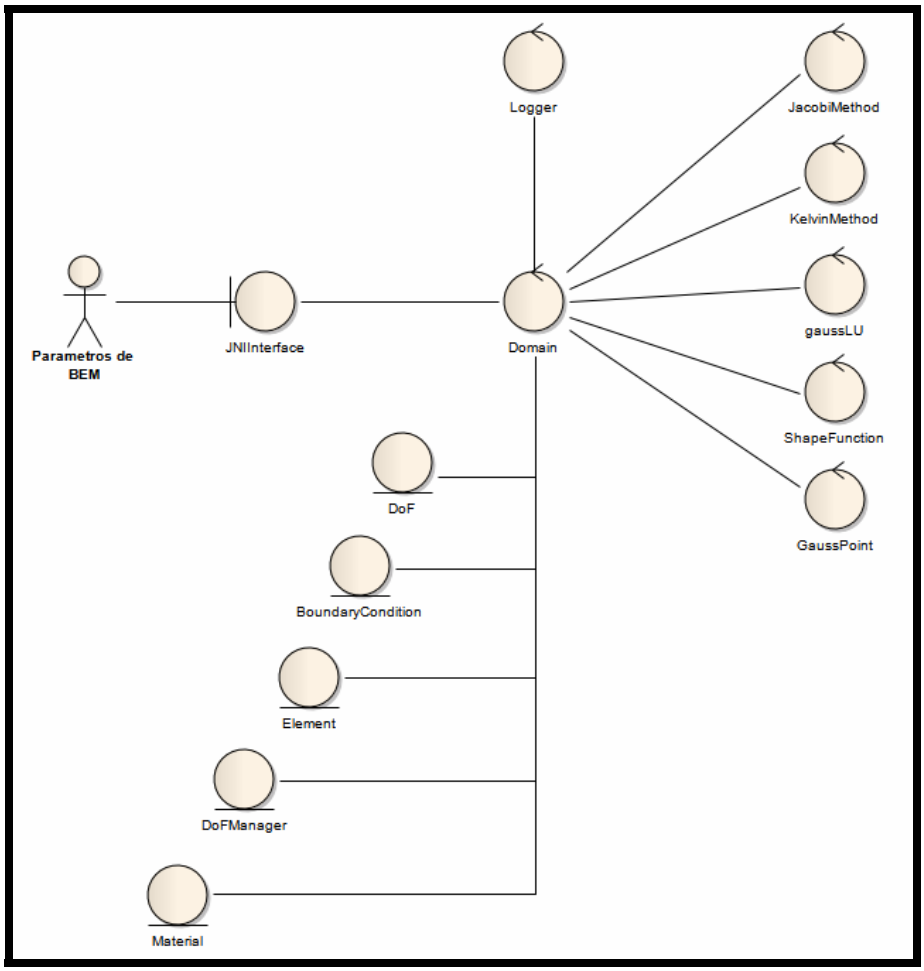

Figura 3.9. Diagrama de clases de análisis de "Parametros de BEM". (Fuente: propia)

3.5.5. **Diagrama de clases de análisis general del software**.

Después de desarrollar y describir el sistema de manera separada, se procedió a, crear el diagrama de clases de análisis del sistema OOBEM, donde se muestra la interacción total de componentes.

En la Figura 3.10, se puede ver el diagrama de clases de análisis del software y la conexión entre los componentes involucrados.

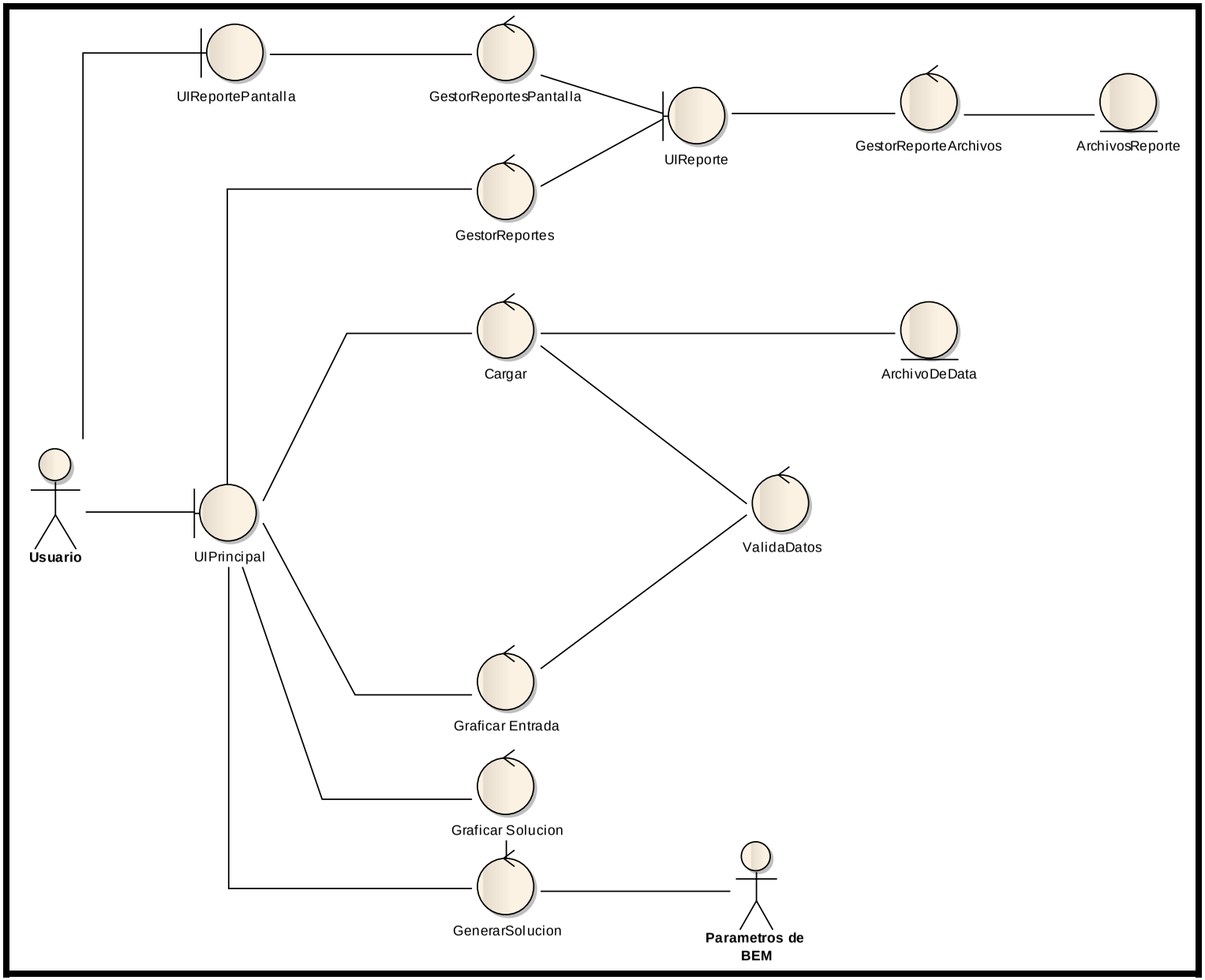

Figura 3.10. Diagrama de clases de análisis del software. (Fuente: propia)

3.6. DIAGRAMAS DE COMUNICACIÓN.

Se desarrollaron 3 diagramas de comunicación:

- Diagrama de comunicación de “Cargar Datos”.
- Diagrama de comunicación de “Generar Solución Problema”.
- Diagrama de comunicación de “Generar Reportes”.

Cada uno de estos diagramas se muestra en las figuras 3.11, 3.12 y 3.13 respectivamente y muestran los patrones de interacción del software, lo cual significa que se puede describir cómo se instancia la realización de un caso de uso.

La descripción de cada uno de los mensajes en los diagramas de comunicación se muestra en las Tablas 3.6, 3.7 y 3.8 respectivamente.

En la figura 3.11, se presenta el diagrama de comunicación de “Cargar Datos”

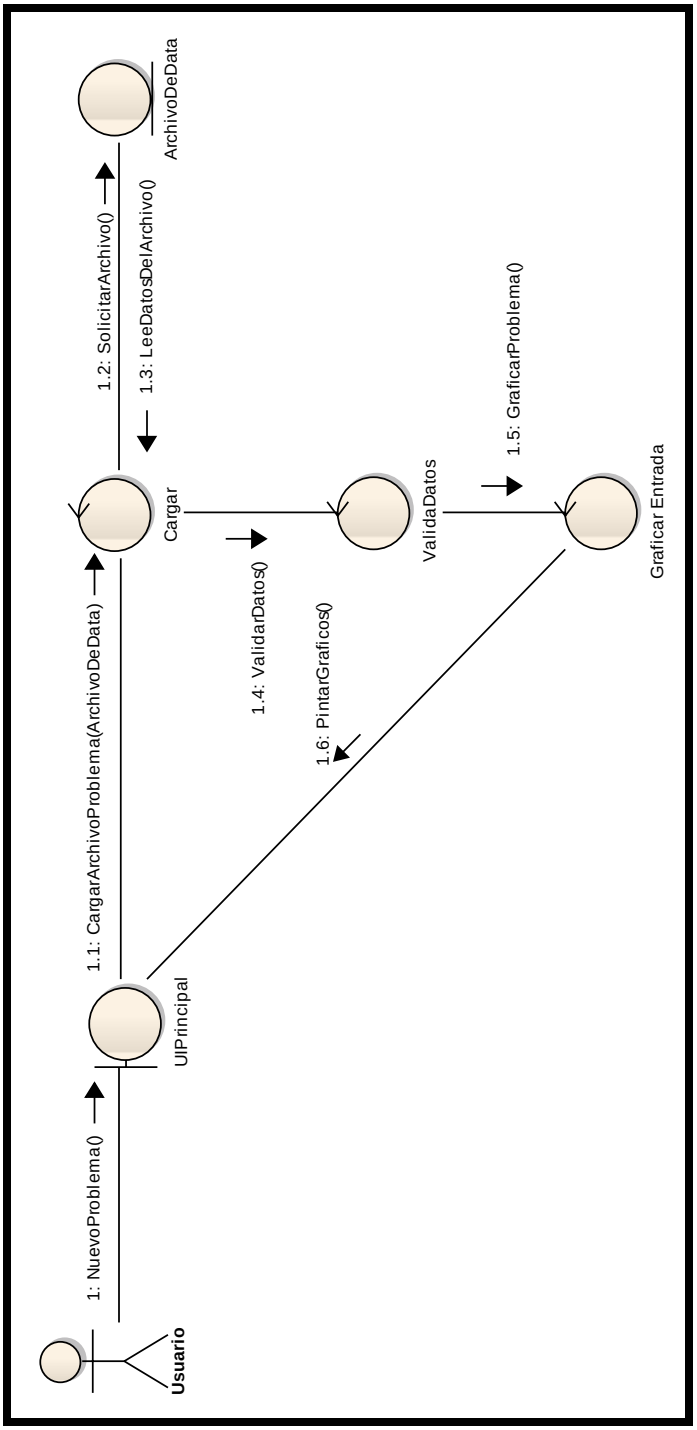

Figura 3.11. Diagrama de comunicación de “Cargar Datos”. (Fuente: propia)

Tabla 3.6. Leyenda del diagrama de comunicación de "Cargar Datos".

(Fuente: propia)

MENSAJES	DESCRIPCIÓN
NuevoProblema()	Es el mensaje que representa la petición del usuario para cargar un problema por medio de un archivo.
CargarArchivoProblema (ArchivoDeData)	Es el mensaje de la interfaz principal a la unidad de control donde se ordena cargar el archivo de datos. El parámetro ArchivoDeData, representa el archivo del cual debe leerse la data.
SolicitarArchivo()	Es el mensaje para solicitar el archivo de data del cual se va a leer la data.
LeeDatosDelArchivo()	Este mensaje representa la data que es leída del archivo de datos por la unidad de control.
ValidarDatos()	Este mensaje es el que indica el comienzo de la validación de la data de entrada, para detectar errores y datos necesarios faltantes.
GraficarProblema()	Es el mensaje que ordena generar los gráficos del modelo inicial.
PintarGraficos()	Es el mensaje que indica la orden de mostrar los gráficos del modelo generado al usuario.

En la Figura 3.12, se presenta el diagrama de comunicación de “Generar Solución Problema”

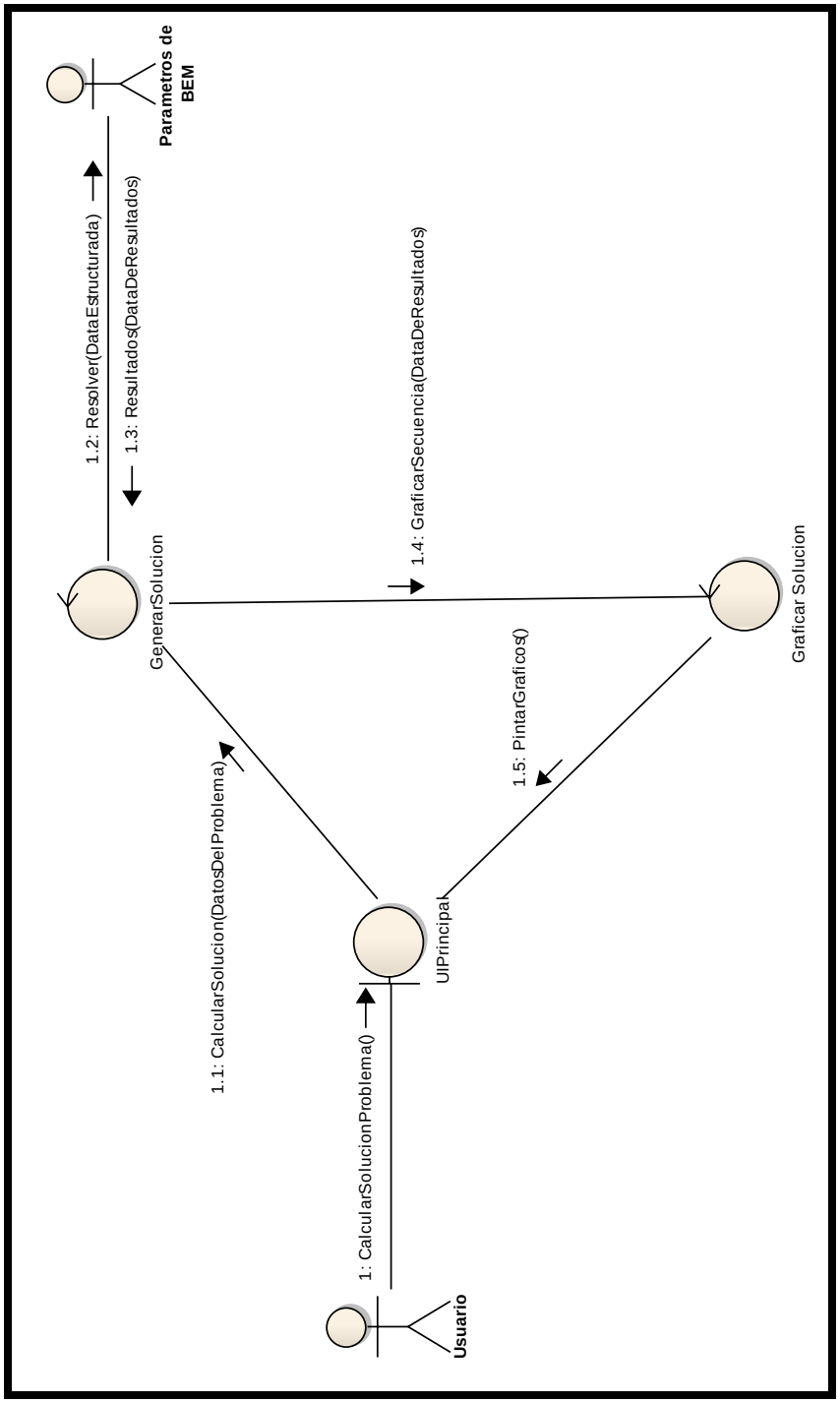

Figura 3.12. Diagrama de comunicación de “Generar Solución Problema”. (Fuente: propia)

Tabla 3.7. Leyenda del diagrama de comunicación de "Generar Solución Problema". (Fuente: propia)

MENSAJES	DESCRIPCIÓN
CalcularSolucionProblema()	Es el mensaje que representa la petición del usuario para resolver un problema cargado en el sistema.
CalcularSolucion (DatosDelProblema)	Es el mensaje de la interfaz principal a la unidad de control donde se ordena resolver el problema actualmente cargado. El parámetro DatosDelProblema, representa la data del problema que está cargada en el sistema sobre el problema actual.
Resolver (DataEstructurada)	Este mensaje representa la orden para resolver el problema al modulo BEM. El parámetro DataEstructurada, representa la misma data del problema pero en el formato necesario para trasferir al modulo de cálculo.
Resultados (DataDeResultados)	Este mensaje indica la salida del modulo de cálculo BEM, que ocurre tan pronto los cálculos sobre el problema han concluido. El parámetro DataDeResultados, representa los resultados obtenidos para el problema planteado.
GraficarSecuencia (DataDeResultados)	Este mensaje ordena que se genere una secuencia grafica con los datos de resultados obtenidos.
PintarGraficos()	Es el mensaje que indica la orden de mostrar la secuencia grafica generada.

En la figura 3.13 se presenta el diagrama de comunicación de "Generar Reportes"

Tabla 3.8. Leyenda del diagrama de comunicación de "Generar Reportes". (Fuente: propia)

MENSAJES	DESCRIPCIÓN
nuevoReporte()	Es el mensaje que representa la petición del usuario de un nuevo reporte.
nuevoReporte (DatosDelProblema)	Es el mensaje de la interfaz y representa la orden de mostrar la interfaz de creación de reportes.
AdaptaryMostrar()	Es el mensaje que ordena a la interfaz de creación de reporte que se adapte a la información disponible y se muestre al usuario.
ConfigurarReporte()	El usuario selecciona la data que contendrá el reporte y la forma como será generado.
ReportePorPantalla (DatosRequeridos)	Es el mensaje emitido por la interfaz de reporte cuando se solicitan reportes para que sean mostrados por pantalla. El parámetro **DatosRequeridos**, hace referencia a la data que debía ser mostrada en el reporte y que había sido seleccionada por el usuario.
ReportesParaArchivo (DatosRequeridos)	Es el mensaje emitido por la interfaz de reporte cuando se solicitan reportes para que sean exportados como archivos. El parámetro **DatosRequeridos**, hace referencia a la data que debía ser mostrada en el reporte y que había sido seleccionada por el usuario.

Tabla 3.8. Leyenda del diagrama de comunicación de "Generar Reportes" (Cont.). (Fuente: propia)

MENSAJES	DESCRIPCIÓN
GenerarReporte (Directorio)	Este mensaje hace referencia a la orden de en el **Directorio**, indicado generar los reportes solicitados. Esto solo ocurre cuando se solicitan reportes para exportar como archivos.
AdaptarInterfazAReporte()	Este mensaje le indica a la interfaz donde se muestran los reportes que según la información requerida se adapte y cargue la data formateada.
MostrarReporte()	Este mensaje indica que el reporte se le muestra al usuario.

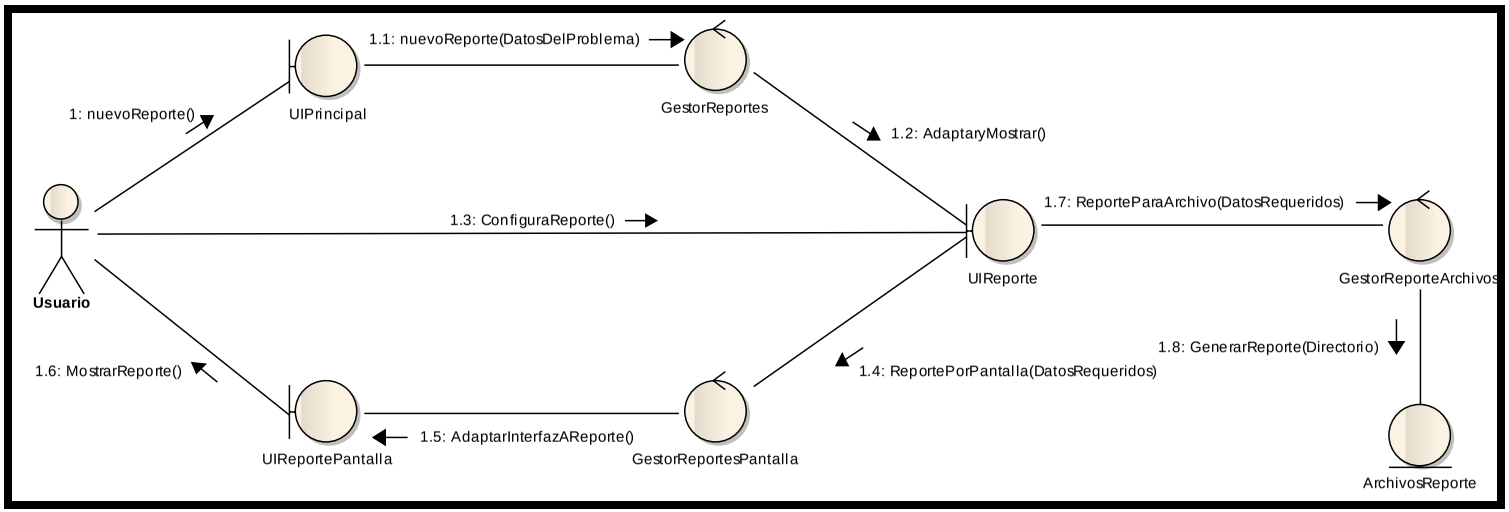

Figura 3.13. Diagrama de comunicación de “Generar Reportes”. (Fuente: propia)

3.7. DIAGRAMAS DE PAQUETES DE ANÁLISIS.

Se desarrollaron 4 diagramas de paquetes de análisis:

- Diagrama de paquetes de análisis del software.
- Diagrama de paquetes de análisis de “Cargar Datos”.
- Diagrama de paquetes de análisis “Generar Solución Problema”.
- Diagrama de paquetes de análisis de “Generar Reportes”.

Cada uno de estos diagramas se muestra en las figuras 3.14, 3.15, 3.16 y 3.17 respectivamente y muestra como el software está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones.

En la figura 3.14 se presenta el diagrama de paquetes de análisis del software

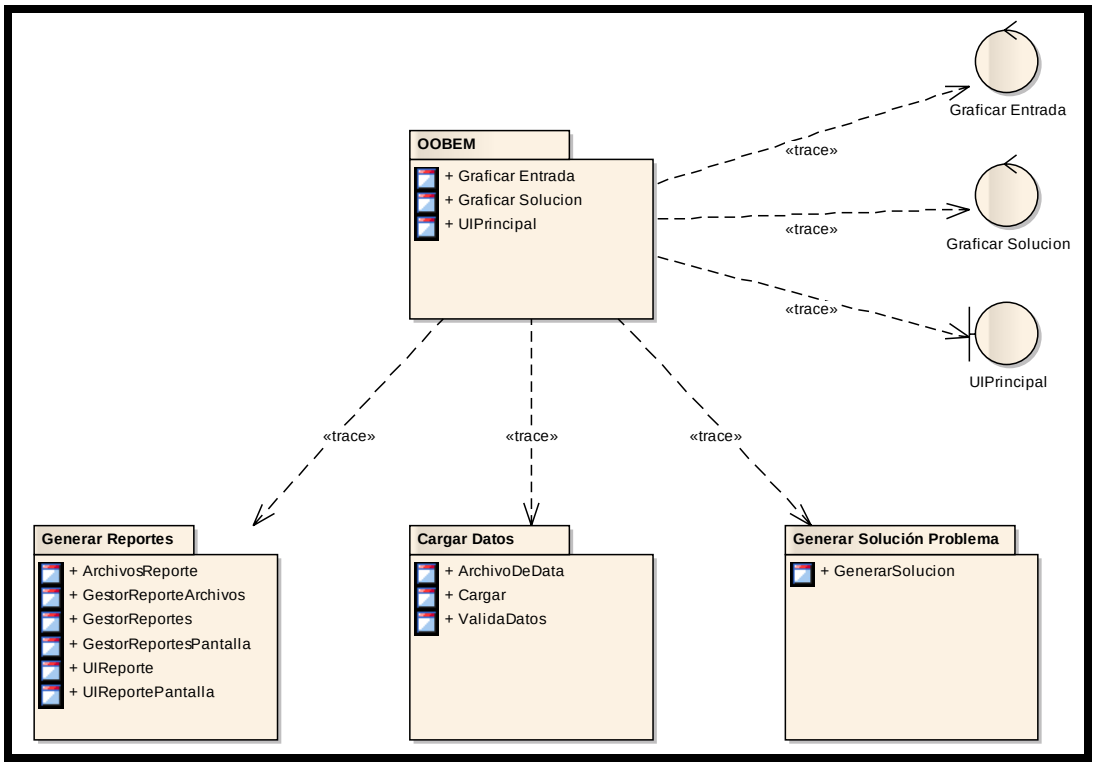

Figura 3.14. Diagrama de paquetes de análisis del sistema OOBEM. (Fuente: propia)

En la figura 3.15 se presenta el diagrama de paquetes de análisis de “Cargar Datos”.

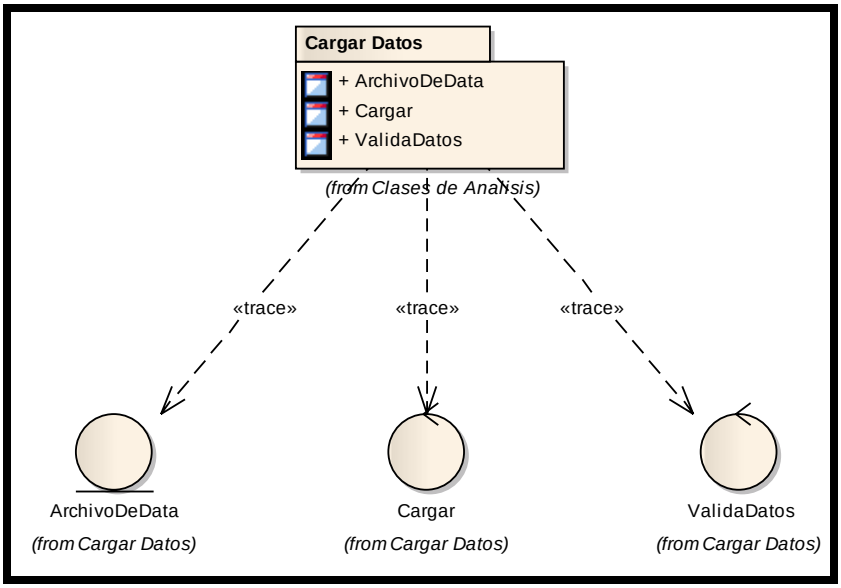

Figura 3.15. Diagrama de paquete de análisis “Cargar Datos”. (Fuente: propia)

En la figura 3.16 se presenta el diagrama de paquetes de análisis de “Generar Solución Problema”.

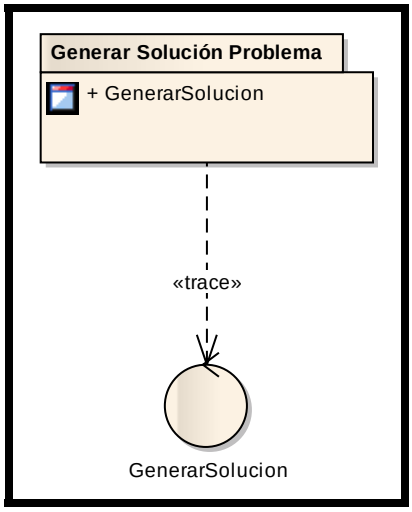

Figura 3.16. Diagrama de paquete de análisis “Generar Solución Problema”. (Fuente: propia)

En la figura 3.17 se presenta el diagrama de paquetes de análisis de “Generar Reportes”.

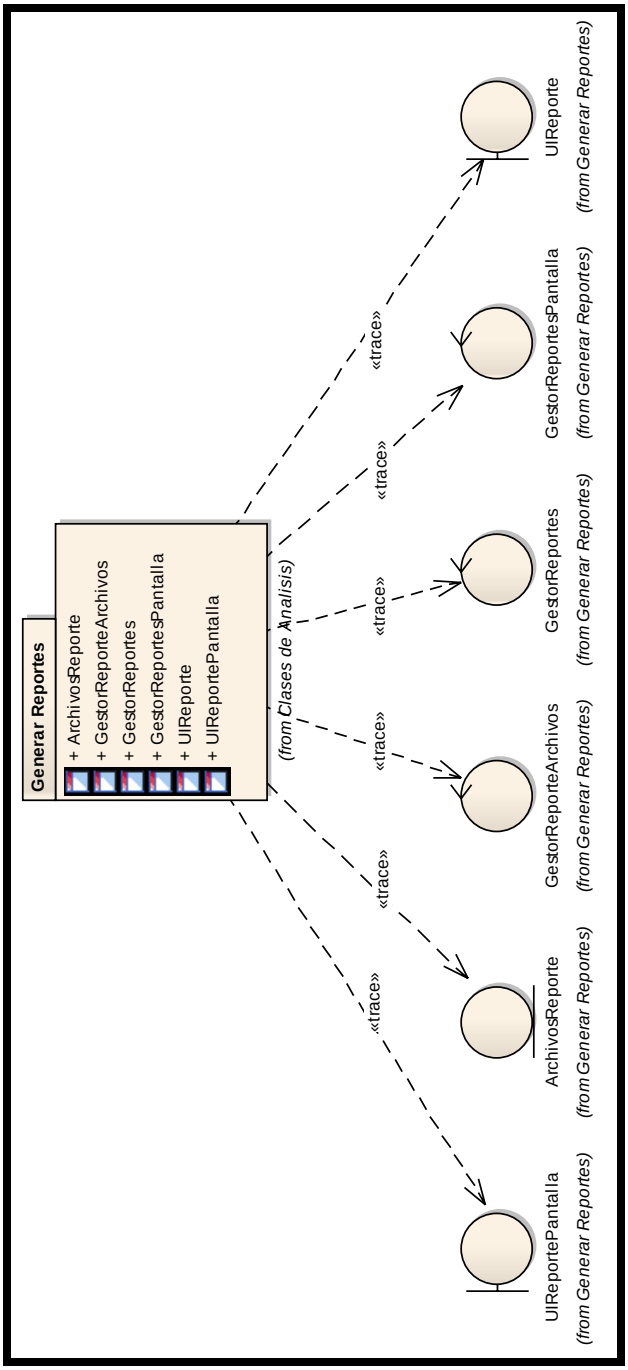

Figura 3.17. Diagrama de paquete de análisis “Generar Reportes”. (Fuente: propia)

3.8. CONCLUSIÓN DE LA FASE DE ANÁLISIS

En este apartado se realizó una investigación preliminar donde se determinaron y clasificaron los requisitos del software, y se estableció una relación de estos con los actores implicados en el sistema mediante los diagramas de casos de uso. Luego, se descompuso el problema en conceptos u objetos individuales con los cuales se pudo, mediante un diagrama de dominio, crear una representación visual del objeto del proyecto; lo que llevo al desarrollo de un glosario de términos.

Luego de completadas las actividades antes mencionadas, se procedió a través del desarrollo de los diagramas de clases de análisis, a describir de una forma más detallada el funcionamiento de los casos de uso. También, a partir de los diagramas de clases de análisis desarrollados, se realizaron los respectivos diagramas de colaboración, para representar las interacciones entre los objetos. Y por último, se construyó el diagrama de paquetes de análisis para encapsular los casos de uso que fueron definidos al realizar el análisis del sistema [23].

En esta fase se obtuvo una buena comprensión del software y después de examinar los objetivos de la fase de análisis, ámbito del sistema, riesgos críticos y arquitectura candidata, se logró establecer que el proyecto es viable y por tanto se decidió continuar con el desarrollo del software **OOBEM**.

CAPÍTULO 4
FASE DE DISEÑO

4.1. INTRODUCCIÓN

En el capítulo 3 se determinó el *qué*, ahora esta fase se encarga de especificar el *cómo*, es decir, se obtuvo un entendimiento más detallado de los requerimientos de la aplicación y se diseñó e implementó y validó la arquitectura del software.

4.2. DISEÑO DE LA ARQUITECTURA DEL SOFTWARE

En este apartado se definieron los componentes que de una forma robusta y optima, forman parte de la base del software en desarrollo, luego se desarrollaron los módulos y se realizaron la integración de los mismos.

4.2.1. Diagrama de clases del software OOBEM

En la Figura 4.1, se muestra el diagrama de clases base del software OOBEM, en el mismo, se muestran las clases relevantes en el desarrollo del software y definen una estructura funcional mínima del sistema.

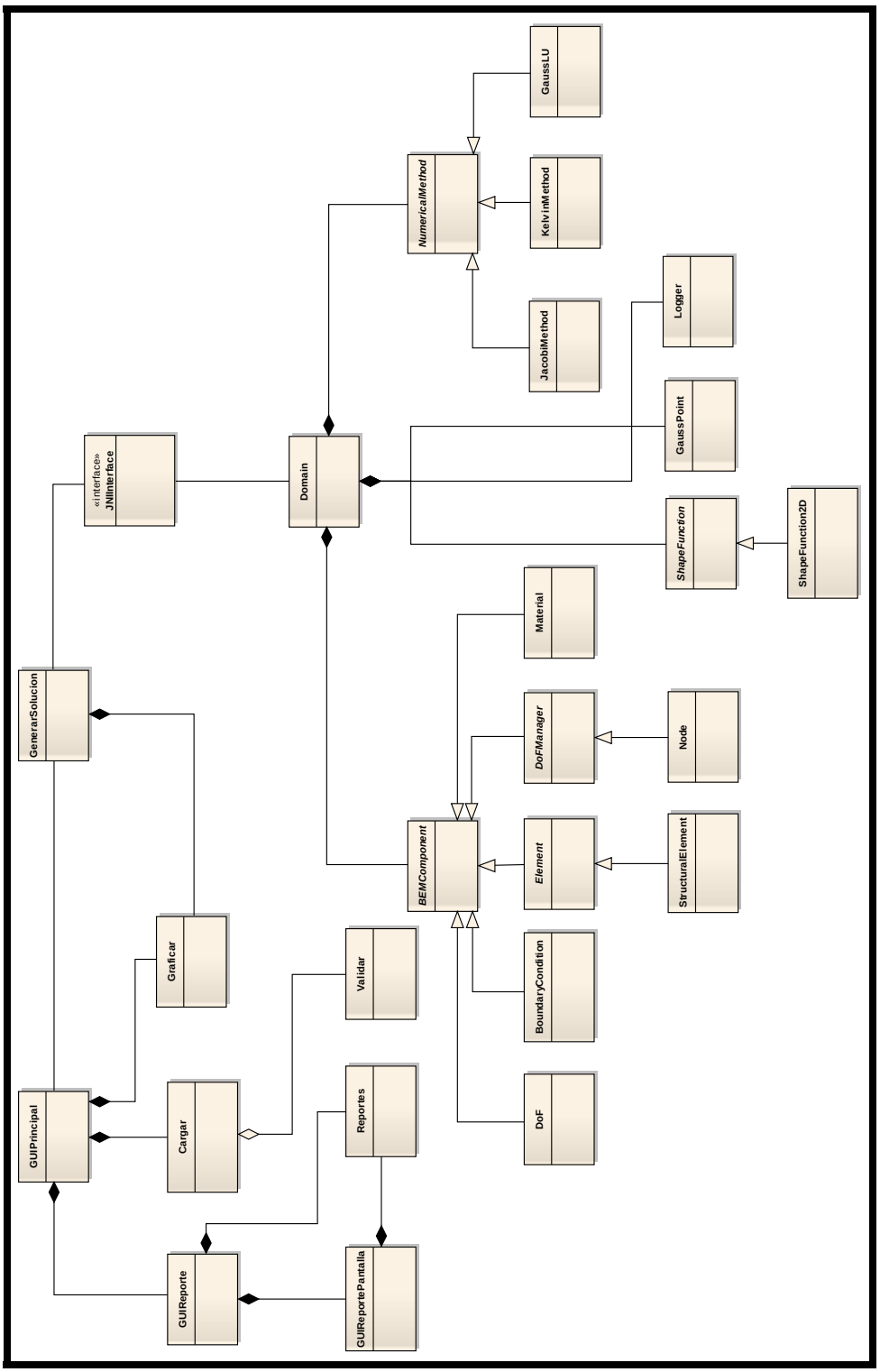

Figura 4.1. Diagrama de clases del software. (Fuente: propia)

4.2.1.1. Clase "Reportes"

Esta clase, es la encargada de la gestión de los reportes tanto de los que son para ser mostrados por pantalla como de los que son exportados en forma de archivos de data. Esta clase debe poseer acceso a los datos del problema 2D de elasticidad como de su solución en caso de existir.

4.2.1.2. Clase "Cargar"

Es la que realiza la carga y procesamiento de los datos de un problema 2D de elasticidad, cuando son leídos de un archivo. Esta clase debe poseer acceso a los datos del problema 2D de elasticidad.

4.2.1.3. Clase "Validar"

Es la que realiza la validación de los datos procesados que son leídos del archivo. Esta clase debe poseer acceso a los datos del problema 2D de elasticidad así como al acceso a la clase "**Graficar**" ya que la clase "**Validar**", emite la orden para que la clase "**Graficar**" del sistema genere y muestre los gráficos del modelo geométrico del problema cuando la validación resulta satisfactoria.

4.2.1.4. Clase "Graficar"

Es la encargada del manejo de gráficos en el sistema, es decir, genera según los datos almacenados en las variables del problema 2D de elasticidad, los gráficos solicitados y la muestra por la interfaz grafica principal del sistema.

4.2.1.5. Clase "GenerarSolución"

Esta clase posee la funcionalidad de permitir la conexión entre la interfaz "**JNIInterface**" y el "**GUIPrincipal**". Posee acceso a los datos del problema 2D de elasticidad que fueron cargados del archivo de entrada y métodos que permiten procesar los resultados y almacenarlos en las variables del problema.

4.2.1.6. Interface "JNIInterface"

Esta interface provee un acceso a las funciones del modulo de cálculo BEM. Posee métodos para el procesamiento de los datos entrantes del problema 2D de elasticidad como por ejemplo, los necesarios para permitir recibir la solicitud de ejecución de los cálculos y devolver los resultados obtenidos.

4.2.1.7. Clase "Domain"

Es la representación de un problema cargado por medio del archivo. Esta clase posee atributos de tipo "**BEMComponent**", "**NumericalMethod**", "**GaussPoint**", "**Logger**" y "**ShapeFunction**", los cuales son usados para conformar un problema elástico y calcular su solución.

4.2.1.8. Clase "BEMComponent"

Es una clase abstracta que representa todo componente de un problema 2D de elasticidad que es resuelto usando el método de los elemento de frontera (BEM).

4.2.1.9. Clase "NumericalMethod"

Es una clase abstracta que representa todos los métodos numéricos usados por la clase "**Domain**", para calcular la solución de un problema 2D de elasticidad.

4.2.1.10. Clase "GaussPoint"

Es la encargada de proporcionar los valores de los puntos de integración gaussianos y los pesos correspondientes para cada punto para los procesos de integración en el cálculo de resultados.

4.2.1.11. Clase "ShapeFunction"

Es una clase abstracta que representa todas las clases que calculan los valores para las funciones de forma según el elemento. Cabe destacar que cada elemento tiene una forma y esa forma una función que la rige.

4.2.1.12. Clase "DoF"

Representa un grado de libertad en el dominio del problema. Esta clase hereda de la clase "**BEMComponent**".

4.2.1.13. Clase "BoundaryCondition"

Representa una condición de frontera en el dominio del problema. Esta clase hereda de la clase "**BEMComponent**".

4.2.1.14. Clase "Element"

Es una clase abstracta que representa todos los tipos de elementos que pueden aplicarse a un problema. Esta clase hereda de la clase "**BEMComponent**".

4.2.1.15. Clase "DoFManager"

Es una clase abstracta que representa todo componente al que se le pueda aplicar grados de libertad, para el caso 2D solo califican los nodos. Esta clase hereda de la clase "**BEMComponent**".

4.2.1.16. Clase "Material"

Es la encargada de almacenar la información del material del que está compuesto el dominio. Esta clase hereda de la clase "**BEMComponent**".

4.2.1.17. Clase "StructuralElement"

Es la clase que representa todos los elementos para problemas con dominios estructurales. Esta clase hereda de la clase "**Element**" y es usada para construir los elementos de problema 2D de elasticidad. Posee atributos que pueden almacenar la matriz del elemento así como una referencia a los nodos que lo conforman.

4.2.1.18. Clase "Node"

Es la clase que representa un nodo del modelo geométrico generado, posee atributos que le permiten almacenar la posición, así como atributos que le permiten diferenciarse entre los puntos internos y nodos de la frontera. Esta clase que hereda de la clase "**DoFManager**".

4.2.1.19. Clase "JacobiMethod"

En esta clase es donde se encuentra la implementación del método de Jacobi tanto para el cálculo de vector unitario, como para el cálculo del jacobiano para los elementos del modelo geométrico en los procesos de integración. Esta clase es una clase que hereda de la clase "**NumericalMethod**".

4.2.1.20. Clase "KelvinMethod"

Esta clase es donde se encontrara implementado el método de kelvin, tiene por objetivo, calcular el valor de la ecuación fundamental para los elementos de un dominio. Esta clase es una clase que hereda de la clase "**NumericalMethod".**

4.2.1.21. Clase "GaussLU"

Esta clase es la encargada de aplicar el método de gauss-crout y resolver el sistema de ecuaciones para obtener los resultados al problema 2D de elasticidad cargado. Esta clase es una clase que hereda de la clase "**NumericalMethod**"

4.2.1.22. Clase "ShapeFunction2D"

Esta clase es una especialización de la clase "**ShapeFunction**", así como su clase padre, está encargada de calcular los valores de las funciones de forma para los elementos del dominio, pero en este caso, esta clase está especializada en calcular solo los valores para las funciones de forma para elementos bidimensionales.

4.3. DIAGRAMAS DE SECUENCIA

Del estudio de los requerimientos para el desarrollo del software, se desarrollaron los casos de usos ya especificados en los diagramas de secuencia que se muestran en las Figuras 4.2, 4.3 y 4.4. Estos muestran el flujo de eventos y la interacción a un mayor nivel que los ya alcanzados en el proceso de desarrollo.

4.3.1. Diagrama de secuencia de “Cargar Datos”

En la Figura 4.2 se muestra el diagrama de secuencia correspondiente al caso de uso: **Cargar Datos**.

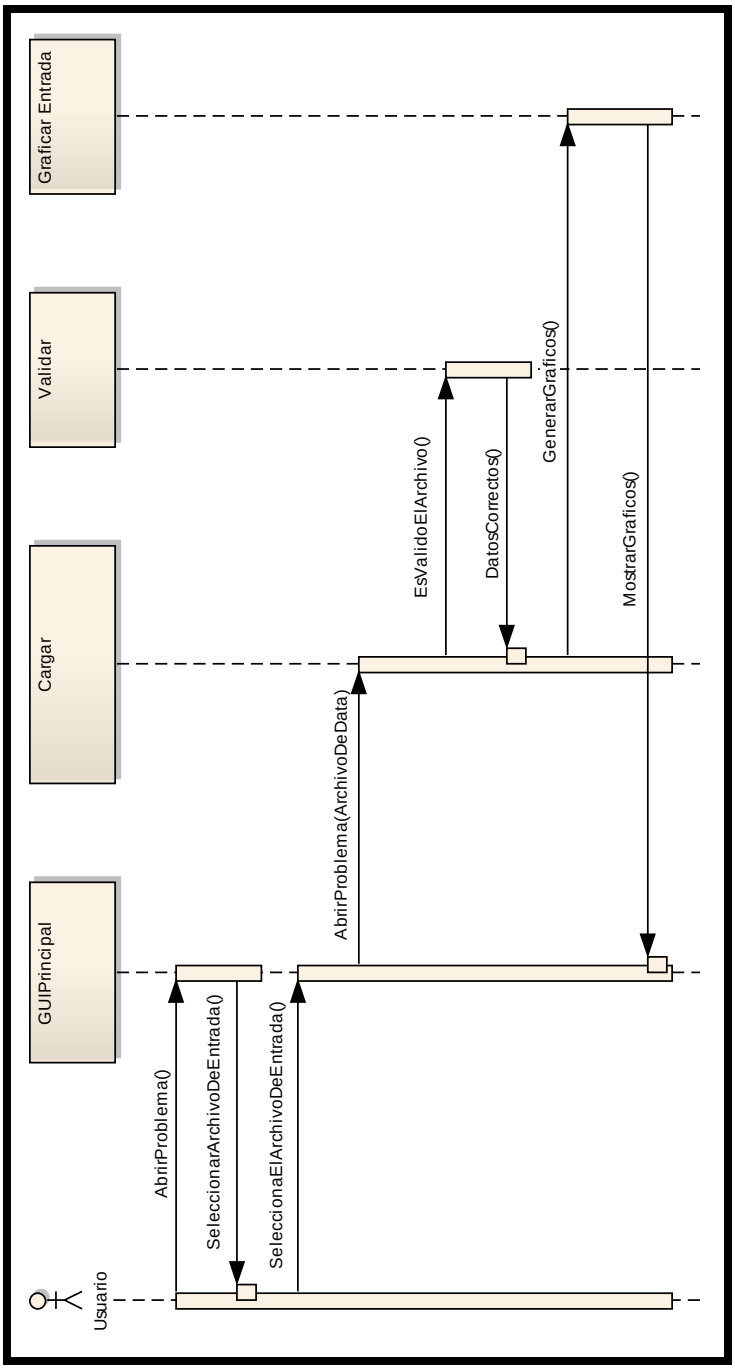

Figura 4.2. Diagrama de secuencia del caso de uso “Cargar Datos”. (Fuente: propia)

4.3.2. Diagrama de secuencia de “Generar Solución Problema”

En la Figura 4.3 se muestra el diagrama de secuencia correspondiente al caso de uso: **Generar Solución Problema**.

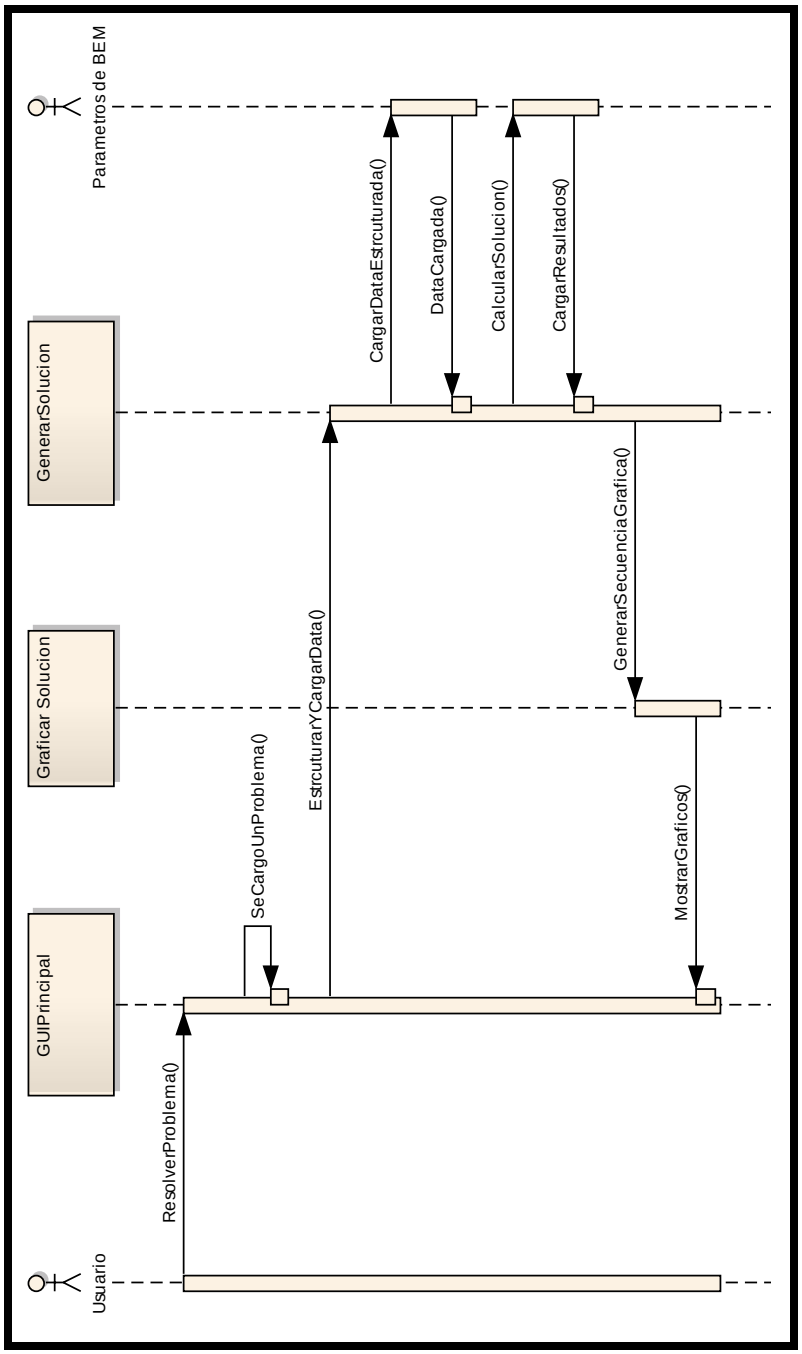

Figura 4.3. Diagrama de secuencia del caso de uso “Generar Solución Problema”. (Fuente: propia)

4.3.3. Diagrama de secuencia de “Generar Reportes”

En la Figura 4.3 se muestra el diagrama de secuencia correspondiente al caso de uso: “**Generar Reportes**”.

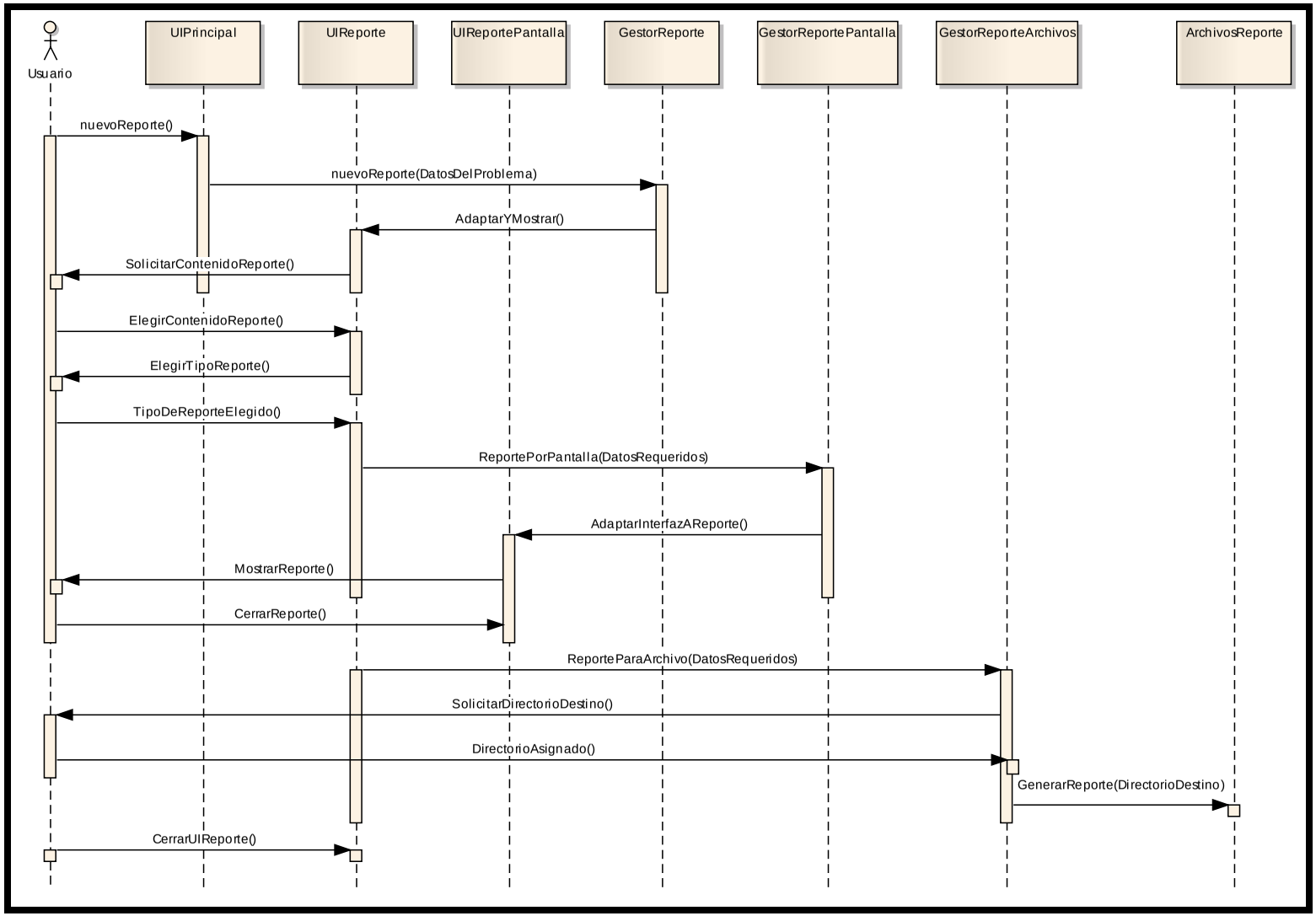

Figura 4.3. Diagrama de secuencia Generar Reportes por pantalla. (Fuente: propia)

4.4. DISEÑO DE LA INTERFAZ DE USUARIO

Uno de los requerimientos especificado en el capítulo 3 fue el que debe poseer una interfaz gráfica, amigable e intuitiva que permita la fácil interacción con el usuario.

En vista de lo anterior se desarrollado las interfaces necesarias con vista de proporcionarle al usuario final las facilidades de interacción con el software final. Las interfaces desarrolladas permiten al usuario realizar todas las tareas necesarias con gran facilidad, aún cuando sea la primera vez que el usuario interactué con él.

En la interfaz principal del software OOBEM se han decidido incluir una serie de menús organizados de tal forma, que el usuario pueda encontrar con gran facilidad las funciones que desea, además, se incluyó una barra de herramienta que contiene una serie de botones a funciones y utilidades a las que el usuario puede acceder directamente.

El panel de reportes es muy intuitivo y fácil de utilizar, permite al usuario no solo emitir los reportes del problema y la solución sino que además le proporciona una forma clara y entendible de seleccionar los datos que quiere para su reporte y en qué forma quiere que sea mostrado.

El panel de muestra de reportes por pantalla, es un panel que es semi-generado por el software dependiendo de la data solicitada en los reportes.

La definición y funcionalidad detallada de los componentes de la interfaces gráficas son descritos en el manual de usuario en el ANEXO A.

4.4.1. Diseño de la interfaz de usuario principal

En la figura 4.5 se muestra la interfaz principal del software.

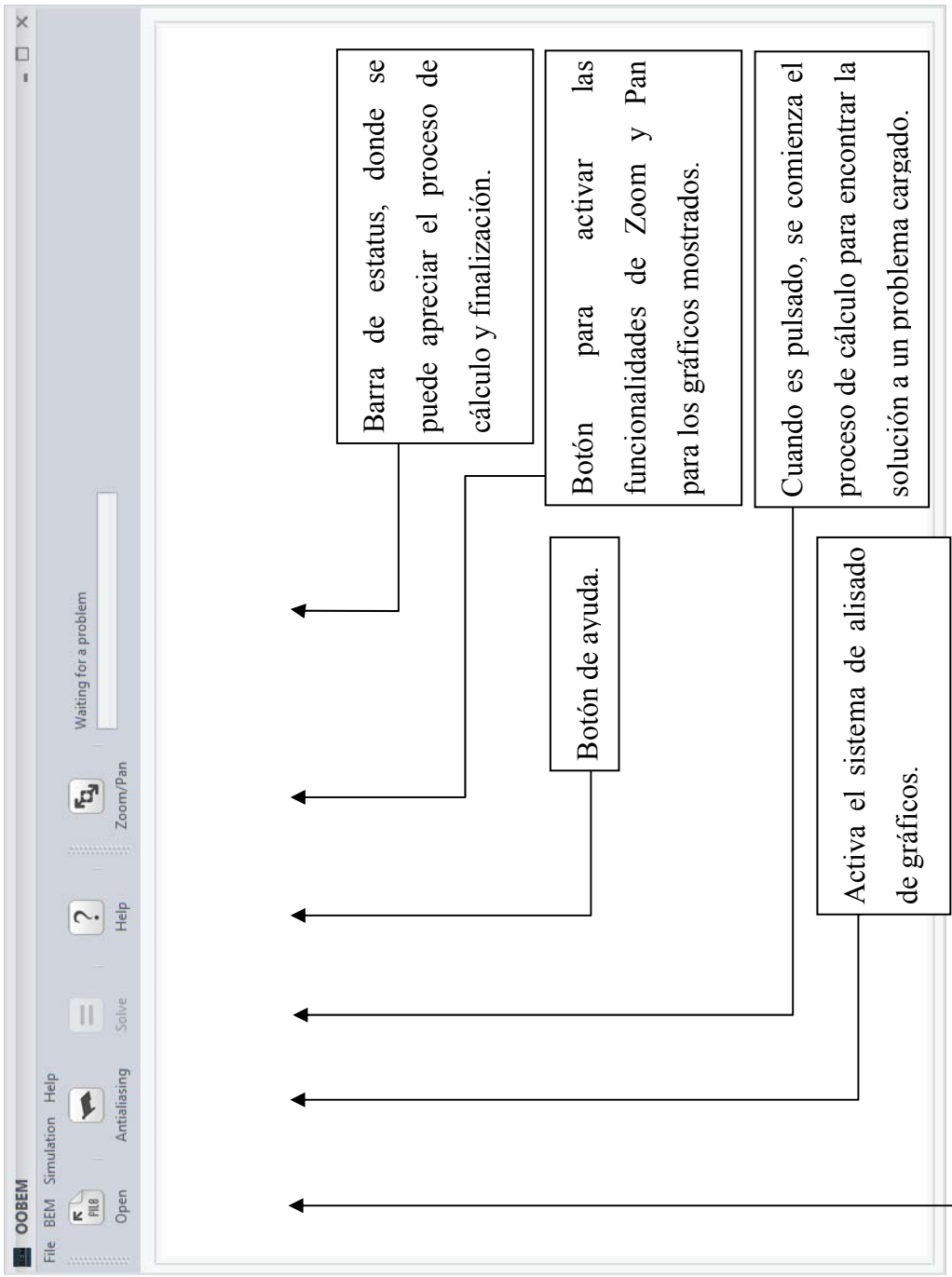

Figura 4.5. Interfaz grafica principal de OOBEM. (Fuente: propia)

4.4.2. Diseño de la interfaz de gestión de reportes

En la figura 4.6 se muestra la interfaz de gestión de reportes del software.

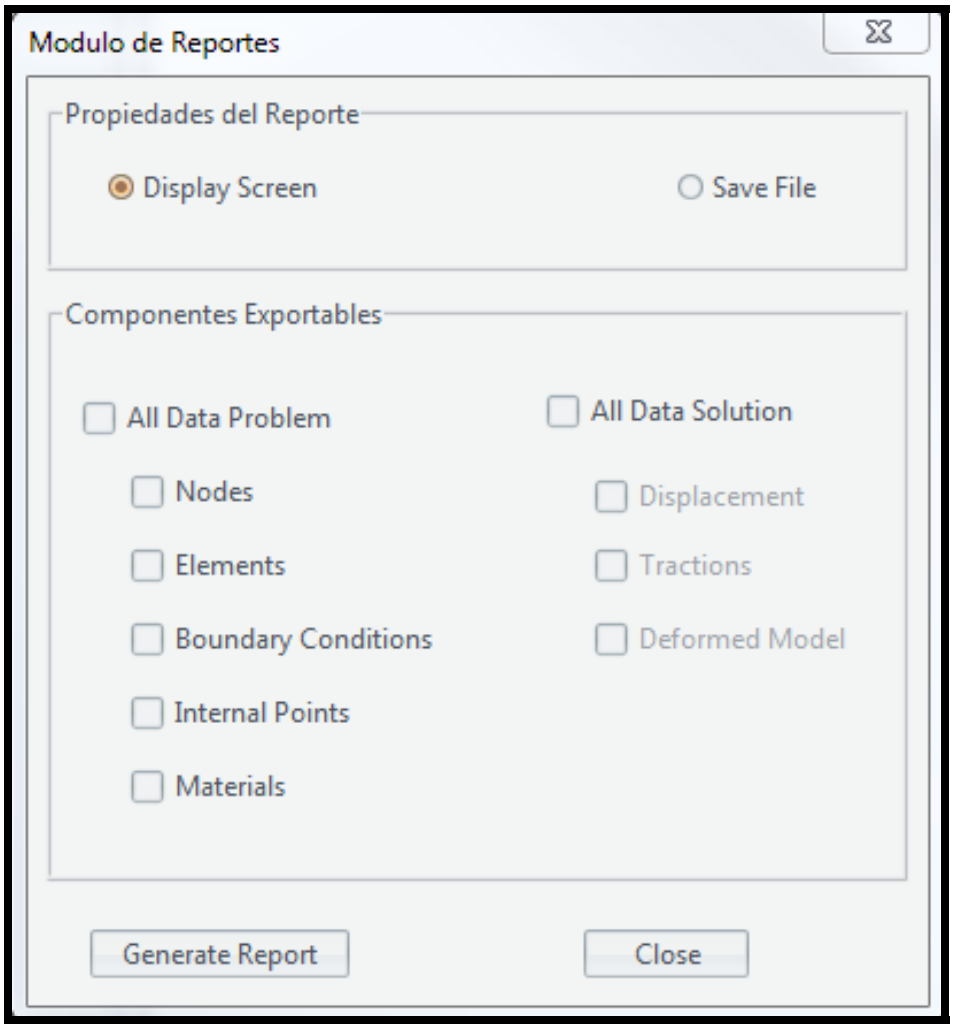

Figura 4.6. Interfaz grafica de gestión de reportes. (Fuente: propia)

4.4.3. Diseño de la interfaz para visualizar reportes por pantalla

En la figura 4.7 se muestra la interfaz para visualizar reportes por pantalla.

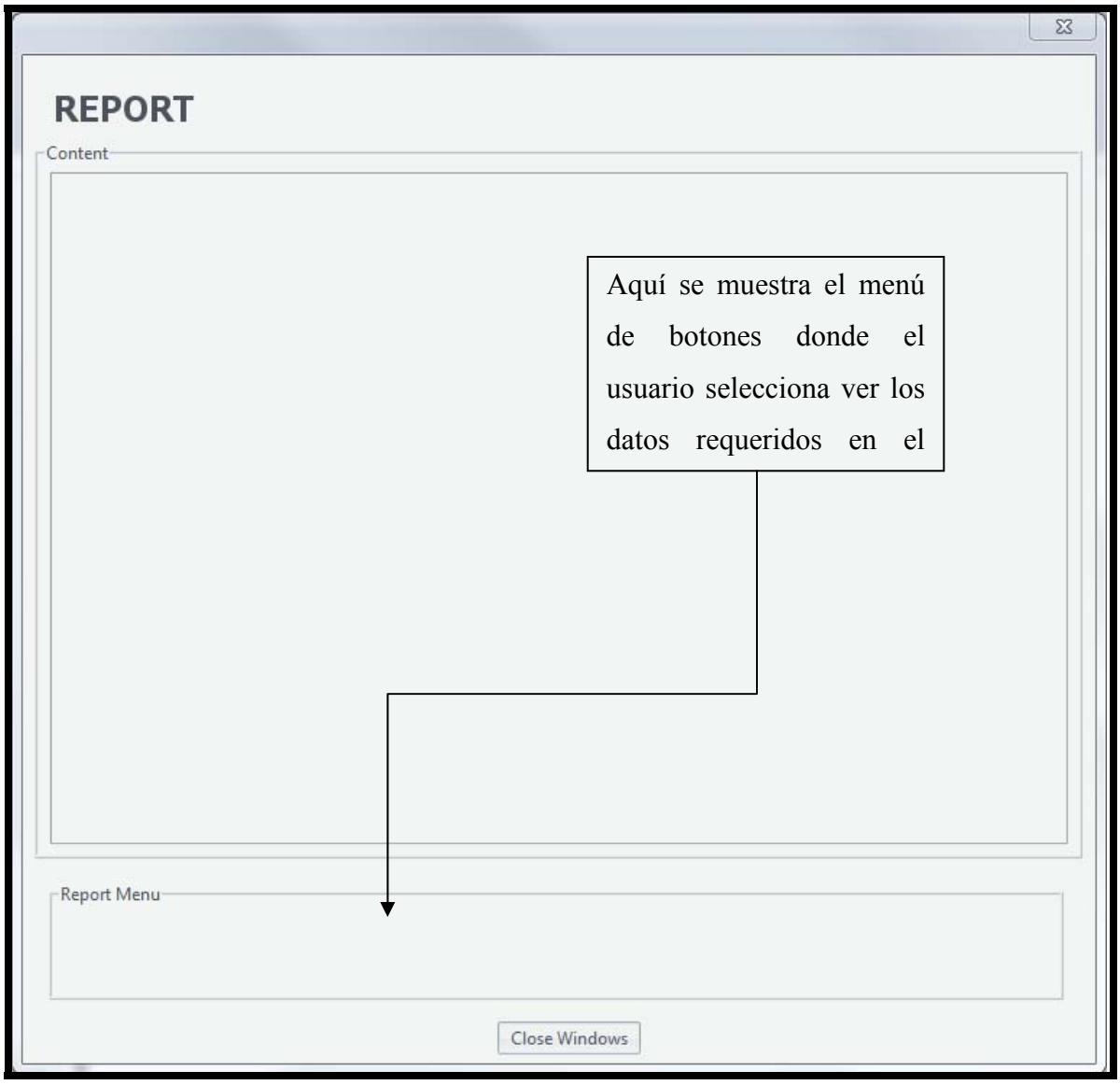

Figura 4.7. Interfaz grafica para visualizar reportes por pantalla. (Fuente: propia)

4.5. ELECCIÓN DE LOS LENGUAJES DE PROGRAMACIÓN.

Para la implementación del sistema y en vista de que uno de los requerimientos indica que el modulo encargado de calcular el resultado del sistema debe

implementarse en C++, el modulo de las interfaces graficas, modulo de graficado y reportes fue implementado en JAVA, esto con el fin de aprovechar las ventajas a nivel de gráficos que posee este lenguaje, mientras que el modulo que realiza los cálculos, es decir, el método de elementos de frontera se implemento en C++.

Una de las novedades de los lenguajes de programación que se ha venido explotando en los últimos tiempos es la de la combinación de los lenguajes de programación para aprovechar sus virtudes y lograr desarrollos modulares garantizando la reutilizabilidad del código y la eficiencia de cálculo.

Para combinar C++ con JAVA se usaron interfaces con métodos nativos. Esta capacidad es provista por JAVA para establecer una conexión por medio de funciones nativas con C++.

Este estilo de programación avanzada permitió cumplir con el requerimiento exigido y se obtuvo un modelo robusto, portable, y con un alto nivel de reutilización a nivel de código fuente.

A continuación se ha desarrollado un diagrama de componentes donde se puede apreciar lo anterior con mayor claridad (Figura 4.8).

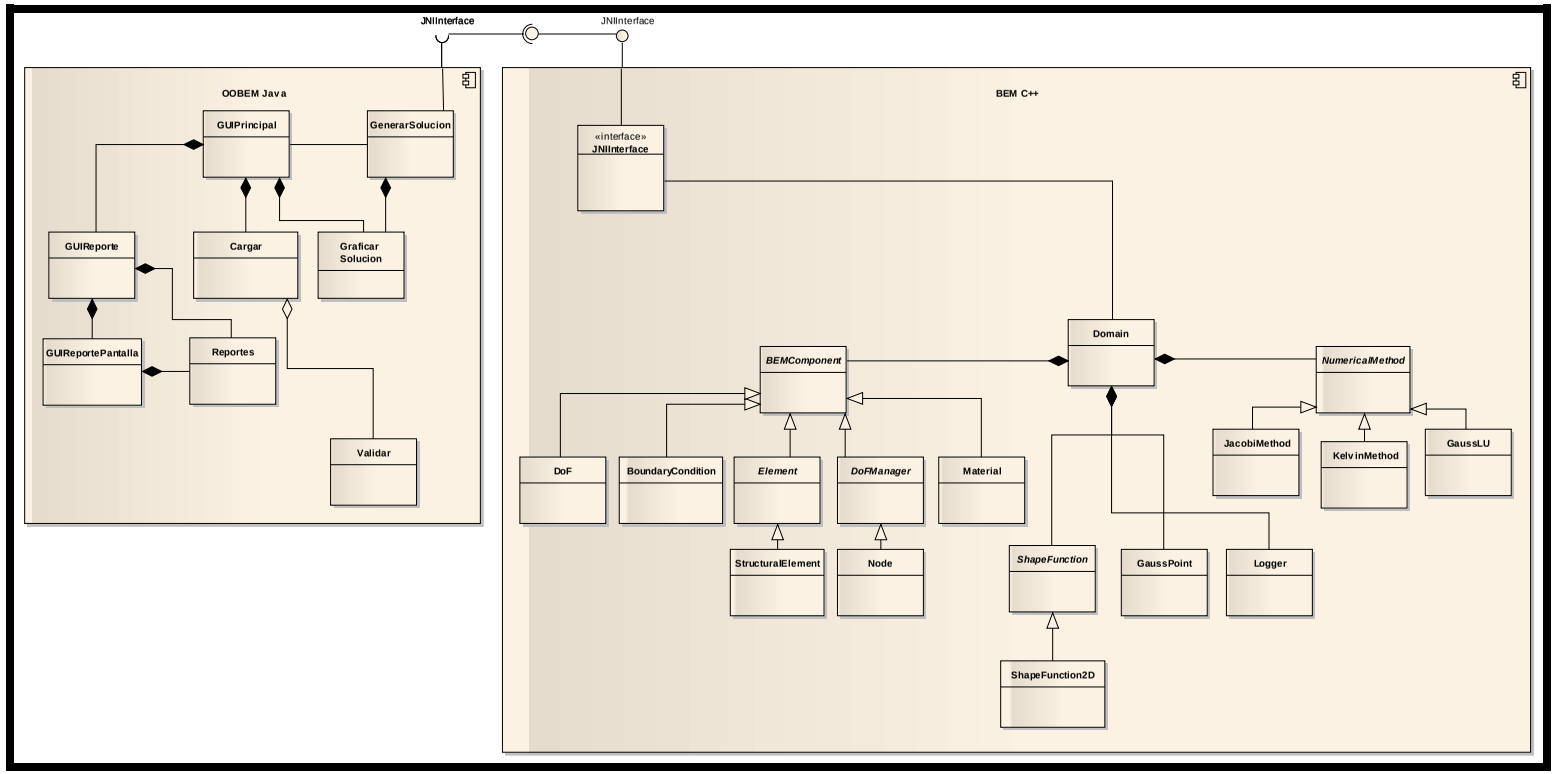

Figura 4.8. Diagrama de componentes generales del software. (Fuente: propia)

CAPÍTULO 5
FASE DE IMPLEMENTACIÓN

5.1. INTRODUCCIÓN

En este capítulo se hace hincapié en la construcción de una versión funcional del sistema y en asegurarse de que el software cumple completamente con las necesidades de los usuarios, esto se logró mediante la implementación del software OOBEM, la realización de pruebas al software OOBEM y en la retroalimentación de los usuarios basada en la utilización del software OOBEM.

5.2. IMPLEMENTACIÓN

En esta sección se implementó la codificación, así como las pruebas al sistema para lograr los reportes de los posibles errores del sistema. El objetivo en esta sección es el lograr tener una versión funcional del producto. Para efectos de la explicación será mostrado el código fuente las interfaces de usuario y de las interfaces de conexión entre los módulos ("**JNIInterface**" y "**BEMConector**").

5.2.1. Implementación de las interfaces de usuario

En esta sección muestra la implementación de las interfaces de usuario del software, las cuales son: "**GUIPrincipal**", "**GUIReportes**" y "**GUIReportesPantalla**".

5.2.1.1. Implementación de "GUIPrincipal"

El código de la interfaz de usuario se presenta a continuación:

```
package ui;

import Listener.AntialiasingListener;
import Listener.GuiBaseListener;
import Listener.ZoomPanListener;
import Listener.AboutBtnListener;
import Listener.CalcaulteBtnListener;
import Listener.UserHelpBtnListener;
import Listener.ExitBtnListener;
import Listener.OpenProblemBtnListener;
import Listener.ReportjmiActionListener;
import Listener.WebpageBtnListener;
import graphics.InteractivePanel;
import java.awt.Dimension;

public class guiBase extends javax.swing.JFrame {
 private static final long serialVersionUID = 1L;

 private InteractivePanel ip = null;

    public guiBase() {
        initComponents();
        this.setPreferredSize(new Dimension(1070,768));
        this.setLocationRelativeTo(null);
        setIconImage(new
javax.swing.ImageIcon(getClass().getResource("/images/32/miniLogo.png")).getIma
ge());
       this.ip = new InteractivePanel(this.canvas_jp);

      this.addComponentListener(new GuiBaseListener(this.ip));
      this.addWindowListener(new GuiBaseListener(null));

      this.antialiasing_tb.addActionListener(new AntialiasingListener(this.ip));
      this.Zoom_tb.addActionListener(new ZoomPanListener(this.ip));
      this.report_jmi.addActionListener(new ReportjmiActionListener(this, true, null));

      OpenProblemBtnListener openFListener =  new OpenProblemBtnListener(this,
this.solve_jb, this.report_jmi,this.calculate_jmi, this.ip, this.status_jl,this.status_jpb);
      this.openProblem_jmi.addActionListener(openFListener);
      this.openProblem_jb.addActionListener(openFListener);
```

```
        this.exit_jmi.addActionListener(new ExitBtnListener());

        CalcaulteBtnListener calcListener = new
    CalcaulteBtnListener(this.status_jl,this.status_jpb, this.ip);
        this.calculate_jmi.addActionListener(calcListener);
        this.solve_jb.addActionListener(calcListener);

        UserHelpBtnListener HelpContentListener = new UserHelpBtnListener();
        this.content_jmi.addActionListener(HelpContentListener);
        this.help_jb.addActionListener(HelpContentListener);

        this.about_jmi.addActionListener(new AboutBtnListener(this));
        this.webpage_jmi.addActionListener(new WebpageBtnListener());

        SimulationListener sim_listener = new SimulationListener(this.ip);
        this.repeatSimulation_jmi.addActionListener(sim_listener);
        this.seeProblemGraphics_jmi.addActionListener(sim_listener);
       }

        private void initComponents() {

         FondoBlanco_jp = new javax.swing.JPanel();
         pizarra_jp = new javax.swing.JPanel();
         canvas_jp = new javax.swing.JPanel();
         BarraDeHerramienta_jtb = new javax.swing.JToolBar();
         estardar_jtb = new javax.swing.JToolBar();
         openProblem_jb = new javax.swing.JButton();
         Separator1 = new javax.swing.JToolBar.Separator();
         antialiasing_tb = new javax.swing.JToggleButton();
         solve_jb = new javax.swing.JButton();
         Separator2 = new javax.swing.JToolBar.Separator();
         help_jb = new javax.swing.JButton();
         Separator3 = new javax.swing.JToolBar.Separator();
         ModificadorPizarra_jtb = new javax.swing.JToolBar();
         Zoom_tb = new javax.swing.JToggleButton();
         jSeparator4 = new javax.swing.JToolBar.Separator();
         status_jp = new javax.swing.JPanel();
         status_jl = new javax.swing.JLabel();
         status_jpb = new javax.swing.JProgressBar();
         BarraDeMenuPrincipal_jmb = new javax.swing.JMenuBar();
         file_jm = new javax.swing.JMenu();
         openProblem_jmi = new javax.swing.JMenuItem();
         exit_jmi = new javax.swing.JMenuItem();
```

```
        bem_jm = new javax.swing.JMenu();
        calculate_jmi = new javax.swing.JMenuItem();
        report_jmi = new javax.swing.JMenuItem();
        simulation_jm = new javax.swing.JMenu();
        repeatSimulation_jmi = new javax.swing.JMenuItem();
        seeProblemGraphics_jmi = new javax.swing.JMenuItem();
        help_jm = new javax.swing.JMenu();
        content_jmi = new javax.swing.JMenuItem();
        webpage_jmi = new javax.swing.JMenuItem();
        about_jmi = new javax.swing.JMenuItem();


setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLO
SE);
        setTitle("OOBEM");
        setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        setLocationByPlatform(true);

        FondoBlanco_jp.setBackground(new java.awt.Color(255, 255, 255));

        pizarra_jp.setBackground(new java.awt.Color(255, 255, 255));
        pizarra_jp.setBorder(javax.swing.BorderFactory.createTitledBorder(""));

        canvas_jp.setBackground(new java.awt.Color(255, 255, 255));
        canvas_jp.setOpaque(false);

        javax.swing.GroupLayout canvas_jpLayout = new
javax.swing.GroupLayout(canvas_jp);
        canvas_jp.setLayout(canvas_jpLayout);
        canvas_jpLayout.setHorizontalGroup(

canvas_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGap(0, 1030, Short.MAX_VALUE)
        );
        canvas_jpLayout.setVerticalGroup(

canvas_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGap(0, 641, Short.MAX_VALUE)
        );

        javax.swing.GroupLayout pizarra_jpLayout = new
```

```
javax.swing.GroupLayout(pizarra_jp);
        pizarra_jp.setLayout(pizarra_jpLayout);
        pizarra_jpLayout.setHorizontalGroup(

pizarra_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addComponent(canvas_jp, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        pizarra_jpLayout.setVerticalGroup(

pizarra_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addComponent(canvas_jp, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        BarraDeHerramienta_jtb.setFloatable(false);

        estardar_jtb.setRollover(true);

        openProblem_jb.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/32/open.png"))); // NOI18N
        openProblem_jb.setText("Open");
        openProblem_jb.setFocusable(false);

openProblem_jb.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

openProblem_jb.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        estardar_jtb.add(openProblem_jb);
        estardar_jtb.add(Separator1);

        antialiasing_tb.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/32/antialiasing.png"))); //
NOI18N
        antialiasing_tb.setText("Antialiasing");
        antialiasing_tb.setFocusable(false);

antialiasing_tb.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

antialiasing_tb.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        estardar_jtb.add(antialiasing_tb);
```

```
        solve_jb.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/32/solve1.png"))); //
NOI18N
        solve_jb.setText("Solve");
        solve_jb.setEnabled(false);
        solve_jb.setFocusable(false);
        solve_jb.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        solve_jb.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        estardar_jtb.add(solve_jb);
        estardar_jtb.add(Separator2);

        help_jb.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/32/help2.png"))); //
NOI18N
        help_jb.setText("Help");
        help_jb.setFocusable(false);
        help_jb.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        help_jb.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        estardar_jtb.add(help_jb);
        estardar_jtb.add(Separator3);

        BarraDeHerramienta_jtb.add(estardar_jtb);

        ModificadorPizarra_jtb.setRollover(true);

        Zoom_tb.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/32/dinamicZoom2.png")));
// NOI18N
        Zoom_tb.setText("Zoom/Pan");
        Zoom_tb.setFocusable(false);
        Zoom_tb.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        Zoom_tb.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        ModificadorPizarra_jtb.add(Zoom_tb);
        ModificadorPizarra_jtb.add(jSeparator4);

        BarraDeHerramienta_jtb.add(ModificadorPizarra_jtb);

        status_jl.setText("Waiting for a problem ");

        status_jpb.setToolTipText("Progress");

        javax.swing.GroupLayout status_jpLayout = new
javax.swing.GroupLayout(status_jp);
```

```
        status_jp.setLayout(status_jpLayout);
        status_jpLayout.setHorizontalGroup(

status_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGroup(status_jpLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(status_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignmen
t.LEADING)
                    .addComponent(status_jpb,
javax.swing.GroupLayout.PREFERRED_SIZE, 200,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(status_jl))
                .addContainerGap(567, Short.MAX_VALUE))
        );
        status_jpLayout.setVerticalGroup(

status_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGroup(status_jpLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(status_jl)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(status_jpb, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(19, Short.MAX_VALUE))
        );

        BarraDeHerramienta_jtb.add(status_jp);

        javax.swing.GroupLayout FondoBlanco_jpLayout = new
javax.swing.GroupLayout(FondoBlanco_jp);
        FondoBlanco_jp.setLayout(FondoBlanco_jpLayout);
        FondoBlanco_jpLayout.setHorizontalGroup(

FondoBlanco_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
            .addComponent(BarraDeHerramienta_jtb,
javax.swing.GroupLayout.DEFAULT_SIZE, 1062, Short.MAX_VALUE)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
```

```
FondoBlanco_jpLayout.createSequentialGroup()
            .addContainerGap()
            .addComponent(pizarra_jp, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addGap(10, 10, 10))
      );
      FondoBlanco_jpLayout.setVerticalGroup(

FondoBlanco_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
         .addGroup(FondoBlanco_jpLayout.createSequentialGroup()
            .addComponent(BarraDeHerramienta_jtb,
javax.swing.GroupLayout.PREFERRED_SIZE, 66,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(11, 11, 11)
            .addComponent(pizarra_jp, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addContainerGap())
      );

      BarraDeMenuPrincipal_jmb.setBackground(new java.awt.Color(255, 255,
255));
      BarraDeMenuPrincipal_jmb.setBorder(null);
      BarraDeMenuPrincipal_jmb.setDoubleBuffered(true);

      file_jm.setText("File");

      openProblem_jmi.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/16/open.png"))); // NOI18N
      openProblem_jmi.setText("Open Problem");
      file_jm.add(openProblem_jmi);

      exit_jmi.setText("Exit");
      file_jm.add(exit_jmi);

      BarraDeMenuPrincipal_jmb.add(file_jm);

      bem_jm.setText("BEM");

      calculate_jmi.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/16/solve1.png"))); //
NOI18N
      calculate_jmi.setText("Solve");
```

```
        calculate_jmi.setEnabled(false);
        bem_jm.add(calculate_jmi);

        report_jmi.setText("Reports");
        report_jmi.setEnabled(false);
        bem_jm.add(report_jmi);

        BarraDeMenuPrincipal_jmb.add(bem_jm);

        simulation_jm.setText("Simulation");

        repeatSimulation_jmi.setText("Repeat Simualtion");
        simulation_jm.add(repeatSimulation_jmi);

        seeProblemGraphics_jmi.setText("See Problem Graphics");
        simulation_jm.add(seeProblemGraphics_jmi);

        BarraDeMenuPrincipal_jmb.add(simulation_jm);

        help_jm.setText("Help");

        content_jmi.setText("Content");
        help_jm.add(content_jmi);

        webpage_jmi.setText("OOBEM Web Page");
        help_jm.add(webpage_jmi);

        about_jmi.setText("About");
        help_jm.add(about_jmi);

        BarraDeMenuPrincipal_jmb.add(help_jm);

        setJMenuBar(BarraDeMenuPrincipal_jmb);

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(FondoBlanco_jp,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(FondoBlanco_jp,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }
    private javax.swing.JToolBar BarraDeHerramienta_jtb;
    private javax.swing.JMenuBar BarraDeMenuPrincipal_jmb;
    private javax.swing.JPanel FondoBlanco_jp;
    private javax.swing.JToolBar ModificadorPizarra_jtb;
    private javax.swing.JToolBar.Separator Separator1;
    private javax.swing.JToolBar.Separator Separator2;
    private javax.swing.JToolBar.Separator Separator3;
    private javax.swing.JToggleButton Zoom_tb;
    private javax.swing.JMenuItem about_jmi;
    private javax.swing.JToggleButton antialiasing_tb;
    private javax.swing.JMenu bem_jm;
    private javax.swing.JMenuItem calculate_jmi;
    public javax.swing.JPanel canvas_jp;
    private javax.swing.JMenuItem content_jmi;
    private javax.swing.JToolBar estardar_jtb;
    private javax.swing.JMenuItem exit_jmi;
    private javax.swing.JMenu file_jm;
    private javax.swing.JButton help_jb;
    private javax.swing.JMenu help_jm;
    private javax.swing.JToolBar.Separator jSeparator4;
    private javax.swing.JButton openProblem_jb;
    private javax.swing.JMenuItem openProblem_jmi;
    private javax.swing.JPanel pizarra_jp;
    private javax.swing.JMenuItem repeatSimulation_jmi;
    private javax.swing.JMenuItem report_jmi;
    private javax.swing.JMenuItem seeProblemGraphics_jmi;
    private javax.swing.JMenu simulation_jm;
    private javax.swing.JButton solve_jb;
    private javax.swing.JLabel status_jl;
    private javax.swing.JPanel status_jp;
    private javax.swing.JProgressBar status_jpb;
    private javax.swing.JMenuItem webpage_jmi;
}
```

5.2.1.2. Implementación de "GUIReporte"

El código de la interfaz de usuario se presenta a continuación:

```
package ui.Report;

import Listener.ReportToScreen;
import Listener.CancelBtnListener;
import Listener.ReportAllDataCheckListener;
import Listener.ReportGoListener;
import Listener.ReportToFile;
import problem.problem;

public class guiReport extends javax.swing.JDialog {

    public guiReport(java.awt.Frame parent, boolean modal) {
        super(parent, modal);
        initComponents();
        setLocationRelativeTo(parent);

        ReportAllDataCheckListener AllDataProblemCheck = new
ReportAllDataCheckListener(new javax.swing.JCheckBox[]{
                    this.Boundary_jcb,
                    this.Elements_jcb,
                    this.InternalP_jcb,
                    this.Materials_jcb,
                    this.Nodes_jcb
                });
        ReportAllDataCheckListener AllDataSolutionCheck = new
ReportAllDataCheckListener(new javax.swing.JCheckBox[]{
                    this.Displacement_jcb,
                    this.tractions_jcb,
                    this.model_jcb
                });
        this.OfScreen_jrb.addActionListener(new ReportToScreen(new
javax.swing.JCheckBox[]{
                    this.Boundary_jcb,
                    this.Elements_jcb,
                    this.InternalP_jcb,
                    this.Materials_jcb,
                    this.Nodes_jcb,
                    this.Displacement_jcb,
                    this.tractions_jcb,
```

```
                this.model_jcb
            }));
        this.ToFile_jrb.addActionListener(new ReportToFile(new
javax.swing.JCheckBox[]{
                this.Boundary_jcb,
                this.Elements_jcb,
                this.InternalP_jcb,
                this.Materials_jcb,
                this.Nodes_jcb,
                this.Displacement_jcb,
                this.tractions_jcb,
                this.model_jcb
            }));
        this.AllDataProblem_jcb.addActionListener(AllDataProblemCheck);
        this.AllDataSolution_jcb.addActionListener(AllDataSolutionCheck);
        this.Cancel_jbtn.addActionListener(new CancelBtnListener(this));
        this.Go_jbtn.addActionListener(new ReportGoListener(this,new
javax.swing.JCheckBox[]{
                this.Nodes_jcb,
                this.Elements_jcb,
                this.Boundary_jcb,
                this.Materials_jcb,
                this.InternalP_jcb,
                this.Displacement_jcb,
                this.tractions_jcb,
                this.model_jcb
            },
            this.OfScreen_jrb));

        if (new problem().getNodes() == null) {
            this.Nodes_jcb.setEnabled(false);
            this.InternalP_jcb.setEnabled(false);
        } else {
            this.Nodes_jcb.setEnabled(true);
            this.InternalP_jcb.setEnabled(true);
        }
        if (new problem().getElements() == null) {
            this.Elements_jcb.setEnabled(false);
        } else {
            this.Elements_jcb.setEnabled(true);
        }
        if (new problem().getBoundaryConditions() == null) {
            this.Boundary_jcb.setEnabled(false);
```

```
    } else {
        this.Boundary_jcb.setEnabled(true);
    }

    if (new problem().getMaterial() == null) {
        this.Materials_jcb.setEnabled(false);
    } else {
        this.Materials_jcb.setEnabled(true);
    }

    if (new problem().getDisplacement() == null) {
        this.Displacement_jcb.setEnabled(false);
        this.model_jcb.setEnabled(false);

    } else {
        this.Displacement_jcb.setEnabled(true);
        this.model_jcb.setEnabled(false);
    }

    if (new problem().getTractions() == null) {
        this.tractions_jcb.setEnabled(false);
    } else {
        this.tractions_jcb.setEnabled(true);
    }
}

private void initComponents() {

    GrupoTipoSalida = new javax.swing.ButtonGroup();
    jPanel1 = new javax.swing.JPanel();
    AllDataProblem_jcb = new javax.swing.JCheckBox();
    Nodes_jcb = new javax.swing.JCheckBox();
    Elements_jcb = new javax.swing.JCheckBox();
    Boundary_jcb = new javax.swing.JCheckBox();
    InternalP_jcb = new javax.swing.JCheckBox();
    Materials_jcb = new javax.swing.JCheckBox();
    AllDataSolution_jcb = new javax.swing.JCheckBox();
    Displacement_jcb = new javax.swing.JCheckBox();
    tractions_jcb = new javax.swing.JCheckBox();
    model_jcb = new javax.swing.JCheckBox();
    jPanel2 = new javax.swing.JPanel();
    OfScreen_jrb = new javax.swing.JRadioButton();
    ToFile_jrb = new javax.swing.JRadioButton();
```

```
        jPanel3 = new javax.swing.JPanel();
        Cancel_jbtn = new javax.swing.JButton();
        Go_jbtn = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Modulo de Reportes");
        setAlwaysOnTop(true);
        setLocationByPlatform(true);
        setResizable(false);

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Componentes
Exportables"));

        AllDataProblem_jcb.setText("All Data Problem");

        Nodes_jcb.setText("Nodes");

        Elements_jcb.setText("Elements");

        Boundary_jcb.setText("Boundary Conditions");

        InternalP_jcb.setText("Internal Points");

        Materials_jcb.setText("Materials");

        AllDataSolution_jcb.setText("All Data Solution");

        Displacement_jcb.setText("Displacement");

        tractions_jcb.setText("Tractions");

        model_jcb.setText("Deformed Model");

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                .addComponent(AllDataProblem_jcb)
                .addGroup(jPanel1Layout.createSequentialGroup()
                   .addGap(21, 21, 21)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                      .addComponent(Elements_jcb)
                      .addComponent(Nodes_jcb)
                      .addComponent(Boundary_jcb)
                      .addComponent(InternalP_jcb)
                      .addComponent(Materials_jcb))))
            .addGap(40, 40, 40)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                .addComponent(AllDataSolution_jcb)
                .addGroup(jPanel1Layout.createSequentialGroup()
                   .addGap(21, 21, 21)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
                      .addComponent(Displacement_jcb)
                      .addComponent(tractions_jcb)
                      .addComponent(model_jcb))))
            .addContainerGap(18, Short.MAX_VALUE))
      );
      jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(16, 16, 16)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
TRAILING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                   .addComponent(AllDataSolution_jcb)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                   .addComponent(Displacement_jcb,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(tractions_jcb)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(model_jcb))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(AllDataProblem_jcb)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(Nodes_jcb)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(Elements_jcb)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(Boundary_jcb)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(InternalP_jcb)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(Materials_jcb)
            .addContainerGap(24, Short.MAX_VALUE))
        );

        jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Propiedades
del Reporte"));

        GrupoTipoSalida.add(OfScreen_jrb);
        OfScreen_jrb.setSelected(true);
        OfScreen_jrb.setText("Display Screen");

        GrupoTipoSalida.add(ToFile_jrb);
        ToFile_jrb.setText("Save File");

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
```

```
                .addGap(16, 16, 16)
                .addComponent(OfScreen_jrb)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 123,
Short.MAX_VALUE)
                .addComponent(ToFile_jrb)
                .addGap(31, 31, 31))
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(OfScreen_jrb)
                    .addComponent(ToFile_jrb))
                .addContainerGap(20, Short.MAX_VALUE))
        );

        Cancel_jbtn.setText("Close");

        Go_jbtn.setText("Generate Report");

        javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(21, 21, 21)
                .addComponent(Go_jbtn, javax.swing.GroupLayout.PREFERRED_SIZE,
113, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(102, 102, 102)
                .addComponent(Cancel_jbtn)
                .addContainerGap(51, Short.MAX_VALUE))
        );
        jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
```

```
            .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                .addComponent(Cancel_jbtn)
                .addComponent(Go_jbtn))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel2,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
    }
    private javax.swing.JCheckBox AllDataProblem_jcb;
    private javax.swing.JCheckBox AllDataSolution_jcb;
    private javax.swing.JCheckBox Boundary_jcb;
    private javax.swing.JButton Cancel_jbtn;
    private javax.swing.JCheckBox Displacement_jcb;
    private javax.swing.JCheckBox Elements_jcb;
    private javax.swing.JButton Go_jbtn;
    private javax.swing.ButtonGroup GrupoTipoSalida;
    private javax.swing.JCheckBox InternalP_jcb;
    private javax.swing.JCheckBox Materials_jcb;
    private javax.swing.JCheckBox Nodes_jcb;
    private javax.swing.JRadioButton OfScreen_jrb;
    private javax.swing.JRadioButton ToFile_jrb;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JCheckBox model_jcb;
    private javax.swing.JCheckBox tractions_jcb;

}
```

5.2.1.3. Implementación de "GUIReportePantalla"

El código de la interfaz de usuario se presenta a continuación:

```
package ui.Report;
import Listener.CancelBtnListener;
import Listener.UIReportOutListener;
import report.FormatoTablaSalida;
```

```
public class UIReportOut extends javax.swing.JDialog {

   public UIReportOut(java.awt.Frame parent, boolean modal,
FormatoTablaSalida[] contenido) {
      super(parent, modal);
      initComponents();
      this.close_jbtn.addActionListener(new CancelBtnListener(this));
       if (contenido != null) {
         for (int i = 0; i < contenido.length; i++) {
            if (contenido[i] != null) {
               System.out.println(contenido[i].getName());
               javax.swing.JButton btnTmp = new
javax.swing.JButton(contenido[i].getName());
               btnTmp.addActionListener(new UIReportOutListener(contenido[i],
this.tablaSalida_jt));
               this.menu_jp.add(btnTmp);
               btnTmp.setVisible(true);
            }
         }
      }
   }

   private void initComponents() {

      content_jp = new javax.swing.JPanel();
      jScrollPane1 = new javax.swing.JScrollPane();
      tablaSalida_jt = new javax.swing.JTable();
      title_jl = new javax.swing.JLabel();
      menuReport_jsp = new javax.swing.JScrollPane();
      menu_jp = new javax.swing.JPanel();
      close_jbtn = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setAlwaysOnTop(true);
      setLocationByPlatform(true);
      setResizable(false);


content_jp.setBorder(javax.swing.BorderFactory.createTitledBorder("Content"));

      tablaSalida_jt.setAutoCreateRowSorter(true);
      tablaSalida_jt.setCursor(new
```

```
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        tablaSalida_jt.setDoubleBuffered(true);
        tablaSalida_jt.setOpaque(false);
        tablaSalida_jt.setSelectionBackground(new java.awt.Color(102, 102, 102));
        tablaSalida_jt.getTableHeader().setResizingAllowed(false);
        tablaSalida_jt.getTableHeader().setReorderingAllowed(false);
        jScrollPane1.setViewportView(tablaSalida_jt);

        javax.swing.GroupLayout content_jpLayout = new
javax.swing.GroupLayout(content_jp);
        content_jp.setLayout(content_jpLayout);
        content_jpLayout.setHorizontalGroup(

content_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addGroup(content_jpLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
695, Short.MAX_VALUE)
                .addContainerGap())
        );
        content_jpLayout.setVerticalGroup(

content_jpLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
            .addGroup(content_jpLayout.createSequentialGroup()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
466, Short.MAX_VALUE)
                .addContainerGap())
        );

        title_jl.setFont(new java.awt.Font("Tahoma", 1, 24));
        title_jl.setText("REPORT");


menuReport_jsp.setBorder(javax.swing.BorderFactory.createTitledBorder("Report
Menu"));

menuReport_jsp.setVerticalScrollBarPolicy(javax.swing.ScrollPaneConstants.VERTI
CAL_SCROLLBAR_NEVER);
        menuReport_jsp.setViewportView(menu_jp);

        close_jbtn.setText("Close Windows");
```

```
        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(21, 21, 21)
                        .addComponent(title_jl,
javax.swing.GroupLayout.PREFERRED_SIZE, 619,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                            .addGroup(layout.createSequentialGroup()
                                .addGap(10, 10, 10)
                                .addComponent(menuReport_jsp,
javax.swing.GroupLayout.PREFERRED_SIZE, 706,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addComponent(content_jp,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
                .addContainerGap())
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(328, Short.MAX_VALUE)
                .addComponent(close_jbtn)
                .addGap(314, 314, 314))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(21, 21, 21)
                .addComponent(title_jl, javax.swing.GroupLayout.PREFERRED_SIZE,
31, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
            .addComponent(content_jp,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(menuReport_jsp,
javax.swing.GroupLayout.PREFERRED_SIZE, 86,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(close_jbtn,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        pack();
    }
    private javax.swing.JButton close_jbtn;
    private javax.swing.JPanel content_jp;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane menuReport_jsp;
    private javax.swing.JPanel menu_jp;
    private javax.swing.JTable tablaSalida_jt;
    private javax.swing.JLabel title_jl;

}
```

5.2.2. Implementación de las Interfaces de conexión de módulos

5.2.2.1. Implementación de BEMConector

El código de la interfaz de usuario se presenta a continuación:

```
package jni;

import problem.problem;
import problem.result;

public class bem {
```

```
    public bem() {
    }

    public native void Calculate();

    public native void ChargerGeneralDataProblem(int quadratureOrder, String
type);

    public native void ChargerNode(String[] ID, double[] x, double[] y, boolean[]
isInternalPoint);

    public native void ChargerElement(String ID[], long[] node1, long[] node2,
long[] node3);

    public native void ChargerBC(String type[], boolean toElement[], long indice[],
String direction[], double valor[]);

    public native void ChargerMaterial(double young, double poisson);

    public native String getProgress();

    public native void InstanciateProblem();

    public void ChargerResult(double[] displacementX, double[] displacementY,
double[] tractionX, double[] tractionY) {
        System.out.println(displacementX.length);
        problem p_tmp_static = new problem();
        result[] disp_tmp = new result[displacementX.length];
        result[] trac_tmp = new result[tractionX.length];

        for (int idNodo = 0; idNodo < displacementX.length; idNodo++) {

            disp_tmp[idNodo] = new result(idNodo, displacementX[idNodo],
displacementY[idNodo]);
            trac_tmp[idNodo] = new result(idNodo, tractionX[idNodo],
tractionY[idNodo]);
            System.out.println(displacementX[idNodo]);
        }

        p_tmp_static.setDisplacement(disp_tmp);
        p_tmp_static.setTractions(trac_tmp);
    }
```

```
    public void ChargerAllProblem() {

        problem p_tmp_static = new problem();

       ChargerGeneralDataProblem(p_tmp_static.getQuadratureOrder(),
p_tmp_static.getType());
        String ID[] = new String[p_tmp_static.getNodes().length];
        double x[] = new double[p_tmp_static.getNodes().length];
        double y[] = new double[p_tmp_static.getNodes().length];
        boolean isInternalPoint[] = new boolean[p_tmp_static.getNodes().length];

        for (int i = 0; i < p_tmp_static.getNodes().length; i++) {
           ID[i] = String.valueOf(i);
           x[i] = p_tmp_static.getNodes()[i].getX();
           y[i] = p_tmp_static.getNodes()[i].getY();
           isInternalPoint[i] = p_tmp_static.getNodes()[i].isIntPoint();
        }

        ChargerNode(ID, x, y, isInternalPoint);

        ID = new String[p_tmp_static.getElements().length];
        long n1[] = new long[p_tmp_static.getElements().length];
        long n2[] = new long[p_tmp_static.getElements().length];
        long n3[] = new long[p_tmp_static.getElements().length];

        for (int i = 0; i < p_tmp_static.getElements().length; i++) {
           ID[i] = String.valueOf(i);
           n1[i] = Long.parseLong(p_tmp_static.getElements()[i].getIDGlobalNode(0));
           n2[i] = Long.parseLong(p_tmp_static.getElements()[i].getIDGlobalNode(1));
           n3[i] = Long.parseLong(p_tmp_static.getElements()[i].getIDGlobalNode(2));
        }

        ChargerElement(ID, n1, n2, n3);

        String type[] = new String[p_tmp_static.getBoundaryConditions().length];
        boolean toElement[] = new
boolean[p_tmp_static.getBoundaryConditions().length];
        long indice[] = new long[p_tmp_static.getBoundaryConditions().length];
        String direction[] = new String[p_tmp_static.getBoundaryConditions().length];
        double valor[] = new double[p_tmp_static.getBoundaryConditions().length];

        for (int i = 0; i < p_tmp_static.getBoundaryConditions().length; i++) {
```

```
            type[i] = p_tmp_static.getBoundaryConditions()[i].getType();
            toElement[i] = p_tmp_static.getBoundaryConditions()[i].isApllyElement();
            indice[i] = p_tmp_static.getBoundaryConditions()[i].getIndice() - 1;
            direction[i] = p_tmp_static.getBoundaryConditions()[i].getDir();
            valor[i] = p_tmp_static.getBoundaryConditions()[i].value();
        }

        ChargerBC(type, toElement, indice, direction, valor);
        ChargerMaterial(p_tmp_static.getMaterial().getYoung(),
p_tmp_static.getMaterial().getPoisson());

        Calculate();

    }
```

5.2.2.2. Implementación de JNIInterface

El código de la interfaz de usuario se presenta a continuación:

```
#include "bemJni.hpp"
#include "Node.hpp"
#include "Element.hpp"
#include "dirBCType.hxx"
#include "BoundaryCondition.hpp"
#include "Material.hpp"
#include "Logger.hpp"
#include "StructuralElement.hpp"
#include "util.hpp"


#include<string>
#include<iostream>
#include<stdlib.h>

using std::string;
using std::cout;

/*
 * Class:     jni_bem
 * Method:    getProgress
 * Signature: ()Ljava/lang/String;
```

```
 */
JNIEXPORT jstring JNICALL Java_jni_bem_getProgress
(JNIEnv *env, jobject jobj) {
   if(domainProblem != NULL){
     return env->NewStringUTF(domainProblem->getStatus().c_str());
   }else{
     return env->NewStringUTF("");
   }

}

/*
 * Class:     jni_bem
 * Method:    InstanciateProblem
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_jni_bem_InstanciateProblem
(JNIEnv *, jobject) {
  domainProblem->clearComponents();
}

/*
 * Class:     jni_bem
 * Method:    Calculate
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_jni_bem_Calculate
(JNIEnv *env, jobject jobj) {

   domainProblem->SolveYourself();

   jclass javaclass = env->GetObjectClass(jobj);
   jmethodID ChargerResultID = env->GetMethodID(javaclass, "ChargerResult",
"([D[D[D[D)V");
   if (ChargerResultID == NULL) {
      return;
   }
   long VectorSize = domainProblem-
>getNumbersofComponentOfType(DoFManagerClass);
   jboolean isCopy1;
   jboolean isCopy2;
   jdoubleArray displCoordinate_x = env->NewDoubleArray(VectorSize);
   jdoubleArray displCoordinate_y = env->NewDoubleArray(VectorSize);
```

```
    jdouble *displacements_x = env->GetDoubleArrayElements(displCoordinate_x,
&isCopy1);
    jdouble *displacements_y = env->GetDoubleArrayElements(displCoordinate_y,
&isCopy2);

    for (int i = 0, j = 1, k = 0; k < VectorSize; i += 2, j += 2, k++) {
        displacements_x[k] = domainProblem->getResults()[i];
        displacements_y[k] = domainProblem->getResults()[j];
    }

    if (isCopy1 == JNI_TRUE) {
        env->SetDoubleArrayRegion(displCoordinate_x, 0, VectorSize,
displacements_x);
        env->ReleaseDoubleArrayElements(displCoordinate_x, displacements_x, 0);
    }
    if (isCopy2 == JNI_TRUE) {
        env->SetDoubleArrayRegion(displCoordinate_y, 0, VectorSize,
displacements_y);
        env->ReleaseDoubleArrayElements(displCoordinate_y, displacements_y, 0);
    }
    //devolvemos resultados a JAVA para simulación.
    env->CallVoidMethod(jobj, ChargerResultID, displCoordinate_x,
displCoordinate_y, displCoordinate_x, displCoordinate_y);
}

/*
 * Class:     jni_bem
 * Method:    ChargerGeneralDataProblem
 * Signature: (ILjava/lang/String;)V
 */
JNIEXPORT void JNICALL Java_jni_bem_ChargerGeneralDataProblem
(JNIEnv *env, jobject jobj, jint quadratureOrder, jstring type) {

    const char *result;
    domainProblem->setQuadratureOrder(quadratureOrder);
    result = env->GetStringUTFChars(type, NULL);
    domainProblem->setTypeOfProblem(result);
    env->ReleaseStringChars(type, NULL);

#ifdef _DEBUG_JNI
    string salida = "Problem:\n"
            "Type = ";
    salida += domainProblem->getTypeOfProblem();
```

```
    salida += "\n";
    salida += "Quadrature Order = ";
    salida += toString<int>(domainProblem->getQuadratureOrder());
    Logger log(salida, "Function: ChargerGeneralData - bemJNI.cc");
#endif
    return;
}

/*
 * Class:     jni_bem
 * Method:    ChargerNode
 * Signature: ([Ljava/lang/String;[D[D[Z)V
 */
JNIEXPORT void JNICALL Java_jni_bem_ChargerNode
(JNIEnv *env, jobject jobj, jobjectArray ID, jdoubleArray x, jdoubleArray y,
jbooleanArray isInternalPoint) {

    int size = env->GetArrayLength(ID); //obtengo la cantidad de los nodos...
    //variables temporales
    double *X, *Y;
    bool *isInternal;

    //cargamos los nodos por medio de un for...
    X = (env->GetDoubleArrayElements(x, NULL));
    Y = (env->GetDoubleArrayElements(y, NULL));
    isInternal = (bool*)env->GetBooleanArrayElements(isInternalPoint, NULL);
    DoFManager *nod_temp;
    for (int i = 0; i < size; i++) {
        jstring strID = (jstring) env->GetObjectArrayElement(ID, i);
        const char *str = env->GetStringUTFChars(strID, NULL);
        double xx = (double) X[i];
        double yy = (double) Y[i];
        bool ii = (bool)isInternal[i];
        nod_temp = new Node(atol(str), domainProblem, xx, yy, ii);

        domainProblem->setComponentOfType(DoFManagerClass, nod_temp);

    }

#ifdef _DEBUG_JNI
    string salida = "Nodes List:\n"
            "ClassName \t ID \t X \t Y \t isInternal"
            "\n";
```

```
    BEMComponent **tmp_n = domainProblem-
>getAllComponentOfType(DoFManagerClass);
    for (int i = 0; i < domainProblem-
>getNumbersofComponentOfType(DoFManagerClass); i++, salida += "\n") {
        Node *n = dynamic_cast<Node*> (tmp_n[i]);

        salida += n->getClassName();
        salida += "\t\t";
        salida += toString<long>(n->getIDComponent());
        salida += "\t";
        salida += toString<double>(n->giveXGeometricsPoc());
        salida += "\t";
        salida += toString<double>(n->giveYGeometricsPoc());
        salida += "\t";
        salida += toString<bool>(n->isInternalPoint());
    }
    Logger log("Function: ChargerNode - bemJNI.cc", salida);
#endif


}

/*
 * Class:     jni_bem
 * Method:    ChargerElement
 * Signature: ([Ljava/lang/String;[J[J[J[Z)V
 */
JNIEXPORT void JNICALL Java_jni_bem_ChargerElement
(JNIEnv *env, jobject jobj, jobjectArray ID, jlongArray node1, jlongArray node2,
jlongArray node3) {
    int size = env->GetArrayLength(ID);
    jlong *nod1 = env->GetLongArrayElements(node1, NULL);
    jlong *nod2 = env->GetLongArrayElements(node2, NULL);
    jlong *nod3 = env->GetLongArrayElements(node3, NULL);

    long id;
    Node *n1,
         *n2,
         *n3;
    BEMComponent **nodos = domainProblem-
>getAllComponentOfType(DoFManagerClass);
    GaussPoint *gp = new GaussPoint(domainProblem->getQuadratureOrder());
```

```
    for (int i = 0, j = 0; i < size; i++, j = 0) {
        jstring strID = (jstring) env->GetObjectArrayElement(ID, i);
        const char *str = env->GetStringUTFChars(strID, NULL);
        id = atol(str);
        do {
            n1 = dynamic_cast<Node*> (nodos[j]);
            j++;
        } while (n1->getIDComponent() != (nod1[i] - 1));
        j = 0;


        do {
            n2 = dynamic_cast<Node*> (nodos[j]);
            j++;
        } while (n2->getIDComponent() != (nod2[i] - 1));
        j = 0;

        do {
            n3 = dynamic_cast<Node*> (nodos[j]);
            j++;
        } while (n3->getIDComponent() != (nod3[i] - 1));

        Element *e_tmp;
        e_tmp = new StructuralElement(id, domainProblem, 3, gp);
        e_tmp->addNode(n1, 1);
        e_tmp->addNode(n2, 2);
        e_tmp->addNode(n3, 3);
        e_tmp->CalculateFunctionalXY();

        domainProblem->setComponentOfType(ElementClass, e_tmp);

    }
#ifdef _DEBUG_JNI
    string salida = "Element List:\n"
            "ClassName \t ID \t NofNodes \t N1 \t N2 \t N3 ..."
            "\n";
    BEMComponent **tmp_n = domainProblem-
>getAllComponentOfType(ElementClass);
    cout << "voy por aqui 9" << endl;
    for (int i = 0; i < domainProblem-
>getNumbersofComponentOfType(ElementClass); i++, salida += "\n") {
        Element *n = dynamic_cast<Element*> (tmp_n[i]);
        cout << "voy por aqui 10" << endl;
```

```
        salida += n->getClassName();
        salida += "\t\t";
        salida += toString<long>(n->getIDComponent());
        salida += "\t";
        salida += toString<int>(n->giveNumbersOfNodes());
        salida += "\t";

        Node **tmp = n->giveNodesList();
        for (int j = 0; j < n->giveNumbersOfNodes(); j++) {
           Node *aux = tmp[j];
           salida += toString<long>(aux->getIDComponent());
           salida += "\t";
        }
     }
     Logger log("Function: ChargerElement - bemJNI.cc", salida);
#endif
}

/*
 * Class:     jni_bem
 * Method:    ChargerBC
 * Signature: ([Ljava/lang/String;[Z[J[Ljava/lang/String;[D)V
 */
JNIEXPORT void JNICALL Java_jni_bem_ChargerBC
(JNIEnv *env, jobject jobj, jobjectArray type, jbooleanArray toElement, jlongArray
indice, jobjectArray direction, jdoubleArray value) {

    int size = env->GetArrayLength(type);
    boundaryType b;
    dirBCType dBc;
    jboolean *to_e = env->GetBooleanArrayElements(toElement, NULL);
    jlong *idTarget = env->GetLongArrayElements(indice, NULL);
    jdouble *values = env->GetDoubleArrayElements(value, NULL);
    BoundaryCondition *bc;
    BEMComponent *target;

          for (int i = 0; i < size; i++) {
        jstring strType = (jstring) env->GetObjectArrayElement(type, i);
        const char *strtype = env->GetStringUTFChars(strType, NULL);
        string T = strtype;

        if (T.compare("DESP") == 0) {
           b = DISPLACEMENT;
```

```
        } else if (T.compare("TRAC") == 0) {
            b = TRACTION;
        }
        jstring strDir = (jstring) env->GetObjectArrayElement(direction, i);
        const char *strdir = env->GetStringUTFChars(strDir, NULL);
        string dir = strdir;

        if (dir.compare("N") == 0) {
            dBc = NORMAL_DIR;
        } else if (dir.compare("X") == 0) {
            dBc = X_DIR;
        } else if (dir.compare("Y") == 0) {
            dBc = Y_DIR;
        } else if (dir.compare("Z") == 0) {
            dBc = Z_DIR;
        }

        int isElement = to_e[i];
        if (isElement != 0) {
            BEMComponent **elements = domainProblem-
>getAllComponentOfType(ElementClass);
            Element *e_tmp;
            int j = 0;
            do {
                e_tmp = dynamic_cast<Element*> (elements[j]);
                j++;
            } while (e_tmp->getIDComponent() != idTarget[i]);
            target = e_tmp;
        } else {
            BEMComponent **nodes = domainProblem-
>getAllComponentOfType(DoFManagerClass);
            DoFManager *n_tmp;
            int j = 0;
            do {
                n_tmp = dynamic_cast<DoFManager*> (nodes[j]);
                j++;
            } while (n_tmp->getIDComponent() != idTarget[i]);

            target = n_tmp;
        }

        bc = new BoundaryCondition(i, domainProblem, b, target, dBc, values[i]);
```

```
        domainProblem->setComponentOfType(BoundaryConditionClass, bc);

    }
#ifdef _DEBUG_JNI
    string salida = "BC List:\n"
            "ClassName \t ID \t Type \t TargetType \t TargetID \t DIR \t Value ..."
            "\n";
    BEMComponent **tmp_bc = domainProblem-
>getAllComponentOfType(BoundaryConditionClass);
    for (int i = 0; i < domainProblem-
>getNumbersofComponentOfType(BoundaryConditionClass); i++, salida += "\n") {
        BoundaryCondition *bc = dynamic_cast<BoundaryCondition*> (tmp_bc[i]);
        salida += bc->getClassName();
        salida += "\t\t";
        salida += toString<long>(bc->getIDComponent());
        salida += "\t";
        salida += toString<int>(bc->getType());
        salida += "\t";
        salida += bc->getTarget()->getClassName().c_str();
        salida += "\t";
        salida += toString<long>(bc->getTarget()->getIDComponent());
        salida += "\t";
        salida += toString<int>(bc->getDirection());
        salida += "\t";
        salida += toString<double>(bc->getValue());
        salida += "\t";
    }
    Logger log(salida, "Function: ChargerBC - bemJNI.cc");

#endif


}

/*
 * Class:     jni_bem
 * Method:    ChargerMaterial
 * Signature: (DD)V
 */
JNIEXPORT void JNICALL Java_jni_bem_ChargerMaterial
(JNIEnv *env, jobject jobj, jdouble young, jdouble poisson) {
    double YoungValue = young;
    double PoissonValue = poisson;
```

```
    if (domainProblem->getTypeOfProblem() == "STRESS") {
        PoissonValue /= (1 + PoissonValue);
        YoungValue *= (1 - pow(PoissonValue, 2));
    }
    Material *mat = new Material(1, domainProblem, YoungValue, PoissonValue);

    BEMComponent **elements = domainProblem-
>getAllComponentOfType(ElementClass);
    for (int i = 0; i < domainProblem-
>getNumbersofComponentOfType(ElementClass); i++) {
        Element *e_tmp = dynamic_cast<Element*> (elements[i]);
        e_tmp->setMaterial(mat);
    }

    domainProblem->setComponentOfType(MaterialClass, mat);

#ifdef _DEBUG_JNI
    string salida = "Element List (Data Material):\n"
            "ClassName \t ID \t Poisson Ratio \t Young Mod \t ShearMod"
            "\n";
    BEMComponent **tmp_n = domainProblem-
>getAllComponentOfType(ElementClass);
    for (int i = 0; i < domainProblem-
>getNumbersofComponentOfType(ElementClass); i++, salida += "\n") {
        Element *n = dynamic_cast<Element*> (tmp_n[i]);
        salida += n->getClassName();
        salida += "\t\t";
        salida += toString<long>(n->getIDComponent());
        salida += "\t";
        salida += toString<double>(n->giveMaterial()->getIDComponent());
        salida += "\t";
        salida += toString<double>(n->giveMaterial()->givePoissonRatio());
        salida += "\t";
        salida += toString<double>(n->giveMaterial()->giveYoungMod());
        salida += "\t";
        salida += toString<double>(n->giveMaterial()->giveShearModulus());
        salida += "\t";
    }
    Logger log("Function: ChargerMaterial - bemJNI.cc", salida);
#endif

}
```

5.3. PRUEBAS

En esta sección se muestran los resultados de las pruebas de unidad de todos los componentes del sistema. Los resultados obtenidos en estas pruebas son cruciales para la culminación del sistema OOBEM, ya que esto nos permitirá obtener como resultado una versión confiable del sistema.

5.3.1. Archivo de entrada

El archivo utilizado por OOBEM es un archivo plano con extensión ".oobem", en el cual en un formato especificado se describe un problema bidimensional de elasticidad (esfuerzo plano/deformación plana).

El formato de este archivo, permite un contenido variable en orden de los datos pero en una estructura de cada grupo de datos restringida, es decir, se tienen un conjunto de etiquetas las cuales son:

- **[TITLE]:** Específica que luego de esta etiqueta se proporcionara el titulo del problema definido en el resto del archivo.
- **[DESCRIPTION]:** Específica que luego de esta etiqueta se proporcionara la descripción del problema definido en el resto del archivo. La descripción es una breve reseña de lo que trata el problema, de lo que se quiere averiguar. Se puede especificar una cantidad ilimitada de información en esta etiqueta así como puede ignorarse;
- **[TYPE]:** Específica que luego de esta etiqueta se proporcionara el tipo del problema, que especificaría entre muchas cosas lo que se busca calcular en el problema. Es un dato importante en la descripción del problema.

- **[MATERIAL]:** Específica que luego de esta etiqueta se proporcionara el material que define al objeto geométrico en la realidad. Es obvio que cada objeto real está compuesto de uno o más materiales, por ejemplo, hierro, plástico, entre otros.
- **[QUADRATURE]:** Específica que luego de esta etiqueta se proporcionara el orden de la cuadratura gaussiana, la cual servirá para especificar la cantidad de puntos de integración a usar para resolver el problema.
- **[NODES]:** Específica que luego de esta etiqueta se proporcionara una lista tabulada de los nodos que componen al modelo geométrico generado a partir de un modelo real.
- **[INTERNAL POINTS]:** Específica que luego de esta etiqueta se proporcionara una lista tabulada de los nodos fuera de la frontera que son usados para hacer evaluaciones en secciones particulares del modelo geométrico.
- **[ELEMENTS]:** Específica que luego de esta etiqueta se proporcionara una lista tabulada de los elementos geométricos que conforman la frontera del modelo total.
- **[BOUNDARYS]:** Específica que luego de esta etiqueta se proporcionara una lista tabulada de las condiciones de frontera aplicadas tanto a elementos como a nodos particulares.
- **[END PROBLEM]:** Indica el final de la carga del archivo, debe colocarse al final lo que esté por debajo de esta etiqueta se omitirá de la lectura.
- **[END]:** Esta etiqueta va a la final de cada etiqueta de grupo de datos, específica el final de la data de esa etiqueta, eso nos permite una movilidad de los grupos de data en el archivo, pueden colocarse en orden variable mientras se respeten las etiquetas.

Dentro de cada etiqueta, la estructura de los datas está bien especificada y esta debe mantenerse. Cabe destacar que en el proceso de lectura, las líneas sin texto y las tabulaciones al inicio y final de las líneas son ignoradas en la lectura, también, el texto que es escrito en el archivo de entrada y no se encuentre dentro del alcance de una etiqueta, será igualmente ignorado.

Tabla 5.1 Descripción de formato del archivo de entrada. (Fuente: propia)

ETIQUETA	ESTRUCTURA
[TITLE]	[TITLE] *Cadena de caracteres que indican el titulo del problema* [END]
[DESCRIPTION]	[DESCRIPTION] *Cadena de caracteres que indican la descripción del problema* [END]
[TYPE]	[TYPE] Cadena de caracteres que representa el tipo de problema. [END]
[MATERIAL]	[MATERIAL] *YOUNG: (Double Number)* *POISSON: (Double Number)* [END] ***Leyenda de Datos:*** ***YOUNG:*** *Representa el Modulo de Elasticidad que define al material que se piensa describir, el nombre de la etiqueta no debe retirarse solo colocar el valor luego de los dos puntos.* ***POISSON:*** *Representa el valor de la razón de poisson que define al material que se piensa describir, el nombre de la etiqueta no debe retirarse solo colocar el valor luego de los dos*

puntos.

Tabla 5.1 Descripción de formato del archivo de entrada (Cont.). (Fuente: propia)

ETIQUETA	ESTRUCTURA
[QUADRATURE]	*[QUADRATURE]* *ORDER: (Int Number)* *[END]*
[NODES]	*[NODES]* *(long Number)<tab>(Double Number)<tab>(Double Number)* *[END]*
[INTERNAL POINTS]	*[INTERNAL POINTS]* *(long Number)<tab>(Double Number)<tab>(Double Number)* *[END]*
[ELEMENTS]	*[ELEMENTS]* *(long Number)<tab>(long Number)<tab>(long Number)<tab>(long Number)* *[END]*

Tabla 5.1 Descripción de formato del archivo de entrada (Cont.).

(Fuente: propia)

ETIQUETA	ESTRUCTURA
[BOUNDARYS]	*[BOUNDARYS]* *(String Type)<tab>(TARGET)<tab>(long Number) <tab>* *(DIRECTION) <tab>(Double Value)* *[END]* ***Leyenda de Datos:*** ***TARGET:*** *Es el dato que representa si la condición de borde es aplicada a un nodo o a un elemento. Esta especificación en el archivo se logra por medio de las cadenas* ***"NODE"*** *y* ***"ELEM";*** *las dos cadenas hacen referencia a su tipo específico y debe ser usada esta notación de manera obligatoria cuidando no equivocarse al escribirla.* ***DIRECTION:*** *Similar a* ***TARGET****, esta etiqueta es definida por caracteres cada uno representando el valor una dirección posible para la condición de borde. Los valores que definen la dirección de la condición de borde son:* ***"X y "Y"****, siendo la primera para indicar que va en dirección de* ***X,*** *la segunda para indicar que va en dirección de* ***Y.***

5.3.2. Partición equivalente

Una partición equivalente es un método de caja negra que divide el dominio de entrada de un programa en clases de datos. El diseño de casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

5.3.3. Identificación de las clases de equivalencia

Debido a que la forma estructurada del archivo de entrada es restrictiva y presenta un alto riesgo de errores en su construcción, las clases de equivalencia serán definidas como cada formato de la tabla para la cual existe una restricción, esto quiere decir, que existirán tantas clases de equivalencia como posibles bloques de entrada existan.

Al ser las clases de equivalencia un factor importante en la realización de pruebas, estas clases serán especificadas y definidas con el mayor detalle posible con el objetivo de evitar confusiones con el bloque estructural que represente cada clase de equivalencia.

Clase de equivalencia: TITULO
Numero: 1
Descripción:
[TITLE]
Titulo del problema
[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser TITLE.
b. La etiqueta TITLE debe existir antes de que se escriba el titulo y debe estar encerrada entre corchetes.
c. En la línea donde está escrita la etiqueta TITLE no debe existir ninguna otra cadena.

d. El titulo del problema puede ser cualquier valor alfanumérico o una cadena vacía.
e. Luego del título del problema en la línea siguiente debe existir la etiqueta END
f. La etiqueta END debe estar encerrada también entre corchetes.
g. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta TITLE es minúscula o tiene algún carácter que lo sea.
b. La etiqueta TITLE este incompleta.
c. Que la etiqueta TITLE no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta TITLE o END existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END
f. Que la etiqueta END no esté encerrada entre corchetes.

Clase de equivalencia: DESCRIPCION

Numero: 2

Descripción:

[DESCRIPTION]

Descripción del problema

[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser DESCRIPTION.
b. La etiqueta DESCRIPTION debe existir antes de que se escriba la descripción y debe estar encerrada entre corchetes.

c. En la línea donde está escrita la etiqueta DESCRIPTION no debe existir ninguna otra cadena.
d. La descripción del problema puede ser puede ser cualquier valor alfanumérico o una cadena vacía.
e. Luego de la descripción del problema en la línea siguiente debe existir la etiqueta END
f. La etiqueta END debe estar encerrada también entre corchetes.
g. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta DESCRIPTION es minúscula o tiene algún carácter que lo sea.
b. La etiqueta DESCRIPTION este incompleta.
c. Que la etiqueta DESCRIPTION no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta DESCRIPTION o END existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END.
f. Que la etiqueta END no esté encerrada entre corchetes.

Clase de equivalencia: TIPO DE PROBLEMA

Numero: 3

Descripción:

[TYPE]

Tipo de problema.

[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser TYPE.

b. La etiqueta TYPE debe existir antes de que se escriba el tipo del problema y debe estar encerrada entre corchetes.
c. En la línea donde está escrita la etiqueta TYPE no debe existir ninguna otra cadena.
d. El tipo de problema debe ser una cadena de caracteres que corresponda con "STRESS" o "STRAIN".
e. Luego de especificar el tipo de problema en la línea siguiente debe existir la etiqueta END.
f. La etiqueta END debe estar encerrada también entre corchetes.
g. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta TYPE es minúscula o tiene algún carácter que lo sea.
b. La etiqueta TYPE este incompleta.
c. Que la etiqueta TYPE no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta TYPE existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END
f. Que la etiqueta END no esté encerrada entre corchetes.
g. Que el tipo de problema no sea "STRAIN" o "STRESS" o que no se encuentran en mayúscula.
h. Que además del tipo adecuado (STRAIN o STRESS) existan otros valores en la misma línea.

Clase de equivalencia: MATERIAL

Numero: 4

Descripción:

[MATERIAL]

YOUNG: (Double Number)

POISSON: (Double Number)
[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser MATERIAL.
b. La etiqueta MATERIAL debe existir antes de que se escriba la descripción y debe estar encerrada entre corchetes.
c. En la línea donde está escrita la etiqueta MATERIAL no debe existir ninguna otra cadena.
d. Los valores del material se colocan uno por línea identificados con el nombre del valor que representa seguido de ":" un espacio en blanco y un valor numérico con o sin punto decimal.
e. Los valores registrables en este campo son "YOUNG" y "POISSON".
f. Luego de especificar el los valores del material en la línea siguiente debe existir la etiqueta END.
g. La etiqueta END debe estar encerrada también entre corchetes.
h. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta MATERIAL es minúscula o tiene algún carácter que lo sea.
b. La etiqueta MATERIAL está incompleta.
c. Que la etiqueta MATERIAL no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta MATERIAL existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END
f. Que la etiqueta END no esté encerrada entre corchetes.
g. Los títulos para los valores del material no sean del tipo permitido.

h. Los títulos para los valores del material no estén completamente en mayúscula.
i. Que los títulos y valores del material no estén uno por línea.
j. Que falte el titulo de un valor para el material.
k. Que el valor de un material no sea un valor numérico.
l. Que el titulo para un valor no esté seguido de ":".
m. Que entre el titulo y el valor no exista un espacio en blanco.

Clase de equivalencia: CUADRATURA

Numero: 5

Descripción:

[QUADRATURE]

ORDER: (Int Number)

[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser QUADRATURE.
b. La etiqueta QUADRATURE debe existir antes de que se escriba la información de la cuadratura y debe estar encerrada entre corchetes.
c. En la línea donde está escrita la etiqueta QUADRATURE no debe existir ninguna otra cadena.
d. El orden la de la cuadratura se coloca de la siguiente forma: primero se coloca en mayúscula la palabra ORDER seguido de ":", luego un espacio y luego un numero entero sin punto decimal.
e. Luego de especificar el valor del orden de cuadratura, en la línea siguiente debe existir la etiqueta END.
f. La etiqueta END debe estar encerrada también entre corchetes.

g. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta QUADRATURE es minúscula o tiene algún carácter que lo sea.
b. La etiqueta QUADRATURE está incompleta.
c. Que la etiqueta QUADRATURE no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta QUADRATURE existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END
f. Que la etiqueta END no esté encerrada entre corchetes.
g. El título para el valor de la cuadratura no sea válido.
h. El titulo para el valor de la cuadratura no esté completamente en mayúscula.
i. Que falte el titulo del valor para la cuadratura.
j. Que el valor del orden de la cuadratura no sea un valor numérico entero.
k. Que el titulo para el valor no esté seguido de ":".
l. Que entre el titulo y el valor no exista un espacio en blanco.

Clase de equivalencia: NODOS

Numero: 6

Descripción:

[NODES]

(long Number)<tab>(Double Number)<tab>(Double Number)

[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser NODES.

b. La etiqueta NODES debe existir antes de que se escriba la información de los nodos y debe estar encerrada entre corchetes.
c. En la línea donde está escrita la etiqueta NODES no debe existir ninguna otra cadena.
d. Los datos de un nodo son cargados por filas de datos separadas por 8 espacios en blanco entre dato y dato.
e. Los datos de un nodo deben estar uno por cada línea.
f. Los datos de un nodo son los siguientes: ID, posición en X y posición en Y. Los valores deben ir como sigue: un valor entero que sea mayor que 1, un valor de punto flotante que corresponde a la posición en X, y por ultimo otro valor de punto flotante que corresponde con la posición Y.
g. Luego de especificar la lista de nodos, en la línea siguiente debe existir la etiqueta END.
h. La etiqueta END debe estar encerrada también entre corchetes.
i. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta NODES es minúscula o tiene algún carácter que lo sea.
b. La etiqueta NODES está incompleta.
c. Que la etiqueta NODES no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta NODES existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END
f. Que la etiqueta END no esté encerrada entre corchetes.
g. Que el campo ID sea menor que 1
h. Que el ID se repita para diferentes nodos.
i. La cantidad de nodos sea menor que 4
j. Que entre los valores no se exista el espaciado correcto.

k. Que falte un valor sobre un campo.
l. Que los valores contengan caracteres excepto por el que representa exponente (e).

Clase de equivalencia: PUNTOS INTERNOS

Numero: 7

Descripción:

[INTERNAL POINTS]

(long Number)<tab>(Double Number)<tab>(Double Number)

[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser INTERNAL POINTS.
b. La etiqueta INTERNAL POINTS debe existir antes de que se escriba la información de los puntos internos y debe estar encerrada entre corchetes.
c. En la línea donde está escrita la etiqueta INTERNAL POINTS no debe existir ninguna otra cadena.
d. Los datos de un punto interno son cargados por filas de datos separadas por 8 espacios en blanco entre dato y dato.
e. Los datos de un punto interno deben estar uno por cada línea.
f. Los datos de un punto interno son los siguientes: ID, posición en X y posición en Y. Los valores deben ir como sigue: un valor entero que sea mayor que 1, un valor de punto flotante que corresponde a la posición en X, y por ultimo otro valor de punto flotante que corresponde con la posición Y.
g. Luego de especificar la lista de puntos internos, en la línea siguiente debe existir la etiqueta END.
h. La etiqueta END debe estar encerrada también entre corchetes.

i. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta INTERNAL POINTS es minúscula o tiene algún carácter que lo sea.
b. La etiqueta INTERNAL POINTS está incompleta.
c. Que la etiqueta INTERNAL POINTS no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta INTERNAL POINTS existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END
f. Que la etiqueta END no esté encerrada entre corchetes.
g. Que el campo ID sea menor que 1
h. Que el ID se repita para diferentes puntos internos.
i. Que el ID sea igual a algún ID de algún nodo
j. Que entre los valores no se exista el espaciado correcto.
k. Que los valores contengan números no validos.

Clase de equivalencia: ELEMENTOS

Numero: 8

Descripción:

[ELEMENTS]

(long Number)<tab>(long Number)<tab>(long Number)<tab>(long Number)

[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser ELEMENTS.
b. La etiqueta ELEMENTS debe existir antes de que se escriba la información de los elementos y debe estar encerrada entre corchetes.

c. En la línea donde está escrita la etiqueta ELEMENTS no debe existir ninguna otra cadena.
d. La información de los elementos son cuatro valores separados por ocho espacios en blanco.
e. Los datos de un elemento son los siguientes: ID, ID del nodo 1, ID del nodo 2 e ID del nodo 3. Los valores de cada elemento deben ir uno por cada línea como sigue: un valor entero que sea mayor que 1, luego un valor entero que corresponde el ID del nodo 1, un valor entero que corresponde el ID del nodo 2 y un valor entero que corresponde el ID del nodo 3.
f. Luego de especificar la lista de elementos, en la línea siguiente debe existir la etiqueta END.
g. La etiqueta END debe estar encerrada también entre corchetes.
h. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta ELEMENTS es minúscula o tiene algún carácter que lo sea.
b. La etiqueta ELEMENTS está incompleta.
c. Que la etiqueta ELEMENTS no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta ELEMENTS existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END
f. Que la etiqueta END no esté encerrada entre corchetes.
g. Que en el campo ID sea menor que 1
h. Que el ID se repita para diferentes elementos.
i. Que entre los valores no se exista el espaciado correcto.
j. Que falte un valor sobre un campo
k. Que los valores contengan números no validos.
l. Que los valores no correspondan a IDs de nodos validos.

m. Que existan menos de 2 elementos.

Clase de equivalencia: CONDICIONES DE FRONTERA

Numero: 9

Descripción:

[BOUNDARYS]

(String Type)<tab>(TARGET)<tab>(long Number) <tab> (DIRECTION)

<tab>(Double Value)

[END]

Condición de validez: Las condiciones de validez para este bloque son las siguientes:

a. El valor de la etiqueta superior debe ser BOUNDARYS.
b. La etiqueta BOUNDARYS debe existir antes de que se escriba la información de las condiciones de frontera y debe estar encerrada entre corchetes.
c. En la línea donde está escrita la etiqueta BOUNDARYS no debe existir ninguna otra cadena.
d. Las condiciones de frontera están conformadas por cinco valores secuenciales separados por 8 espacios en blanco.
e. El primer valor del conjunto de izquierda a derecha es una cadena de caracteres que puede ser o “DESP” o “TRAC”.
f. El segundo valor del conjunto de izquierda a derecha es una cadena de caracteres que puede ser “ELEM” o “NODE”.
g. El tercer valor del conjunto de izquierda a derecha es un numero entero que representa un ID de un nodo o elemento.
h. El cuarto valor de izquierda a derecha es un carácter que puede ser “X” o “Y”.
i. El quinto valor de izquierda a derecha es un valor numérico con punto flotante.

j. Cada condición de borde es declara en una línea.
k. Luego de especificar la lista de condiciones de borde, en la línea siguiente debe existir la etiqueta END.
l. La etiqueta END debe estar encerrada también entre corchetes.
m. En la línea donde está la etiqueta END no debe existir otra cadena.

Condiciones de no validez: las condiciones que hacen inválido el bloque son las siguientes:

a. La etiqueta BOUNDARYS es minúscula o tiene algún carácter que lo sea.
b. La etiqueta BOUNDARYS está incompleta.
c. Que la etiqueta BOUNDARYS no esté encerrada entre corchetes.
d. Que en la línea donde está la etiqueta BOUNDARYS existan otros valores aparte de la misma.
e. Que el bloque no termine con la etiqueta END
f. Que la etiqueta END no esté encerrada entre corchetes.
g. Que los valores de una condición no estén separados correctamente.
h. Que el primer campo no sea ni "DESP" ni "TRAC"
i. Que el segundo campo no sea "ELEM" o "NODO".
j. Que el tercer valor sea un numero pero que no corresponda a un ID de un nodo o elemento.
k. Que el cuarto valor no sea el carácter "X" ni "Y".
l. Que el quinto valor no sea un valor numérico valido.

5.3.4. Resumen de las clases de equivalencia

1. TITULO
2. DESCRIPCIÓN
3. TIPO DE PROBLEMA
4. MATERIAL
5. CUADRATURA
6. NODOS

7. PUNTOS INTERNOS
8. ELEMENTOS
9. CONDICIONES DE FRONTERA

5.3.5. Casos de prueba

Los casos de prueba fueron creados a través de usuarios potenciales a los cuales se les solicito, dándoles un modelo de archivo de entrada, la modificación de la información contenida en el con la intención de provocar errores en el sistema. Las modificaciones no fueron supervisadas por los desarrolladores para mantener la realidad de un error que pudiera producirse en el proceso de creación o modificación de un archivo.

Los casos de prueban serán representados cada uno por un archivo afectado, para efectos de explicación se describen aquí solo tres archivos de prueba.

5.3.5.1. Caso de Prueba correspondiente al primer archivo de entrada.

Tabla 5.2 Caso de prueba 1. (Fuente: propia)

BLOQUE DE ENTRADA	RESPUESTA	DETALLES DE ERROR	CLASE
[TITLE] **EL TITULO DEL PROBLEMA** **[END]**	**VALIDO**	NO APLICA	**1**
[DESCRIPTION] **DESCRIP DEL PROBLEMA** **[END]**	**VALIDO**	NO APLICA	**2**
[TYpE **STRES** **END**	**NO VALIDO**	a, c, f, g	**3**
[MATERIaL] **YOUN: 200.0e9** **POISSON 0.300** **[ED]**	**NO VALIDO**	a, e, f, g, l	**4**
QUADRATURE **ORDER A** **[end]**	**NO VALIDO**	c, e, j, k	**5**
[NODES] **1 0.0 0.0** **2 1.0 0.0** **3 2.0 0.0** **4 3.0 0.0** **5 4.0 0.0** **6 5.0 0.0** **7 6.0 0.0** **[END]**	**VALIDO**	NO APLICA	**6**
[INTERNAL POINTS] **[END]**	**VALIDO**	NO APLICA	**7**
[ELEMENTS] **1 1 2 3** **2 3 4 5** **3 5 6 7** **[END]**	**VALIDO**	NO APLICA	**8**
[BOUNDARYS] **DESP NODE 1 X 0** **DESP NODE 0 Y 0** **DESP NODE 8 X 0** **DESP NODE 6 Y 0** **TRAC ELEM 1 Y -100000**	**NO VALIDO**	j	**9**

[END]

5.3.5.2. Caso de Prueba correspondiente al segundo archivo de entrada.

Tabla 5.3 Caso de prueba 2. (Fuente: propia)

BLOQUE DE ENTRADA	RESPUESTA	DETALLES DE ERROR	CLASE
[TITLE] **[END]**	**VALIDO**	NO APLICA	**1**
[DESCRIPTION **DESCRIPCIÓN DEL PROBLEMA** **[END]**	**NO VALIDO**	c	**2**
[TYPE] **STRAIN** **[END]**	**VALIDO**	NO APLICA	**3**
[MATRIAL] **YOUNG: 2Q0.0e9** **POISSON: 0..300** **[end]**	**NO VALIDO**	b, e, k	**4**
[QUADRATURE] **0RDER: 10** **{END]**	**NO VALIDO**	f, g	**5**
[NODES] **1 0.0 0.0** **2 1.0 0.0** **3 2.0 0.0** **0 ..0.5 0.0** **5 4.0 Aa** **2 5.0 0.0** **7 6.0 0.0** **[END]**	**NO VALIDO**	g, h, l	**6**
[INTERNAL POINTS] **3 9.0 -.e** **2 5.0 0.0** **67 6.0 0.0** **[END]**	**NO VALIDO**	i, j, k	**7**
[ELEMENTS] **1 1 2 3** **2 3 4 5** **3 5 6 7** **[END]**	**NO VALIDO**	l	**8**
[BoUNDARYS] **DESP NODE 1 X 0** **DESP NODE 2 Y node** **DEPL NODE 5 X 0**	**NO VALIDO**	a, f, h, j, k, l	**9**

DESP	**NODE**	**6**	**H**	**0**
TRAK	**ELEM**	**1**	**Y**	**-100000**
[END				

5.3.5.3. Caso de Prueba correspondiente al tercer archivo de entrada.

Tabla 5.4 Caso de prueba 3. (Fuente: propia)

BLOQUE DE ENTRADA	RESPUESTA	DETALLES DE ERROR	CLASE
[TITL] **[END]**	**NO VALIDO**	b	**1**
[DESCRIPTION} DESCRIPCIÓN DEL PROBLEMA **[END]**	**NO VALIDO**	c, d	**2**
[TYPE] **ESFUERZO** **[END]**	**NO VALIDO**	NO APLICA	**3**
[MATRIAL] **YOUNG: 200.0e9** **POISSON: 0.300** **[END]**	**VALIDO**	NO APLICA	**4**
[QUADRATURE] **0RDER: 10** **[END]**	**VALIDO**	NO APLICA	**5**
[NODES] **1 0.0 0.0** **2 1.0 0.0** **3 2.0 0.0** **5 0.5 0.0** **5 4.0** **2 5.0 0.0** **7 6.0 0.0** **[END]**	**NO VALIDO**	h, j, k	**6**
[INTERNAL POINTS] **40 9.0 0.0** **77 5.0 0.0** **67 6.0 0.0** **[END]**	**VALIDO**	NO APLICA	**7**
[ELEMENTS] **1 1 2 3** **2 3 4 5** **3 5 6 7** **[END]**	**NO VALIDO**	l	**8**
[BoUNDARYS] **DESP NODE 1 N 0**	**NO VALIDO**	a, i, f, j, k, l	**9**

DESP	**NODE**	**2**	**Y**	**-e**
DESP	**NODe**	**5**	**X**	**0**
DESP	**NODE**	**6**	**Y**	**0**
TRAC	**E1EM**	**1**	**Y**	**-A**
[END				

5.4. CONCLUSIÓN DE LA FASE DE IMPLEMENTACIÓN

En esta fase del ciclo de vida se ha asegurado que el software cumple completamente las necesidades de los usuarios. Y se ha puesto en manos de la comunidad de usuarios la versión final del software con su manual de usuario donde se describe como instalar y utilizar correctamente la aplicación.

CONCLUSIONES

- El formato de archivo desarrollado para la creación de problemas bidimensionales de elasticidad, permite un contenido variable en el orden de los datos pero en una estructura de cada grupo de datos restringida y facilita la documentación de los archivos de problema de una forma sencilla cómoda y segura al usuario.
- El desarrollo del modulo BEM en C++ y las interfaces gráficas en JAVA, permitirá un grado de reutilizabilidad satisfactorio y una nivel de cálculo veloz para la resolución de problemas bidimensionales de elasticidad 2D.
- JNI resulto ser un modo adecuado para realizar el enlace entre las interfaces graficas y el modulo de cálculo de OOBEM, ya que esta modalidad permitió una comunicación segura y transparente entre C++ y JAVA.
- La implementación de las interfaces graficas ofrecen una nueva forma de resolución y visualización de resultados cuando se resuelven problemas bidimensionales de elasticidad con el método de elementos de frontera, esto permitirá compartir resultados y evita la dependencia de otros programas
- El sistema de reportes es una unidad importante en este tipo de software porque permite al usuario extraer datos más allá de las simulaciones graficas. OOBEM al implementar un sistema se reportes que permite la creación de reportes como archivos HTML y por pantalla, proveerá comodidad a los usuarios a la hora de extraer, comparar y compartir datos con otros usuarios sobre sus investigaciones.
- Mediante la metodología OMT, se diseñaron clara y eficientemente los módulos que conforman el software, lo que permitió desarrollar una arquitectura sólida del mismo.
- Los diagramas UML permitieron representar de una manera clara y con diferentes niveles de abstracción el software desarrollado, esto permitió tener clara la estructura del software desde sus inicios hasta su finalización.

- OOBEM facilitará las investigaciones en el área de propagación de grietas, ya que permitirá experimentar con procedimientos numéricos con mayor facilidad.

RECOMENDACIONES

- Adaptar nuevos procedimientos para el diseño y modelado de problemas bidimensionales de elasticidad para obtener así un software de análisis más completo.
- Adaptar nuevos procedimientos para la resolución de problemas tridimensionales así como el desarrollo e integración de otros métodos numéricos que resuelven EDP.

BILIOGRAFÍA CITADA

[1] QIAO H., (2006) **"Object-oriented Programming for the Boundary Element Method in Two-Dimensional Heat Transfer Analysis",** *Advances in Engineering Software*, vol. 37, pp 248–259.

[2] CUNHA, M.T.F., TELLES J.C.F., COUTINHO, A.L.G.A. (2004) "**A Portable Parallel Implementation of a Boundary Element Elastostatic Code for Shared and Distributed Memory Systems"** *Advances in Engineering Software*, vol. 35, pp. 453–460.

[3] SWEDLOW, J. L. AND CRUSE, T. A. (1971) **"Formulation of the Boundary Integral Equation for Three-dimensional Elastoplastic Flow"**, *International Journal of Solids and Structures*, vol. 7, pp. 1673-1681.

[4] RICCARDELLA, P. (1973) **"An Implementation of the Boundary Integral Technique for Plane Problems of Elasticity and Elastoplasticity"**, PhD Thesis, Carnegie Mellon University, Pittsburgh, PA.

[5] ALIABADI, M.H., (2001) **"The Boundary Element Method, vol II: Application to Solids and Structures"**, JOHN WILEY & SONS, LTD., 2da edición.

[6] WANG W., XING J., WANG Y. (1999) **"Object-Oriented Programming in Boundary Element Methods Using C++."** *Advances in Engineering Software*, vol. 30, pp 127-132.

[7] LAGE C. (1996) **"The Application of Object-Oriented Methods to Boundary Elements".** *Comput. Methods Appl. Mech. Engrg.*, vol. 157, pp. 205-213.

[8] SALGADO, N. K., ALIABADI, M. H. and CALLAN, R. E. (1997) **"Rule Inferencing and Object-Orientation for Boundary Elements Mesh Design",** *Artificial Intelligence in Engineering*, vol. 11, pp. 183-190.

[9] SALGADO, N. K. and ALIABADI, M. H. (1999), **"An Object Oriented System for Damage tolerance Design of Stiffened Panels"**, **Engineering** *Analysis with Boundary Elements*, vol. 23, pp 21–34.

[10] BOŘEK, P., ZDENEK, B. (2001) **"Design of Object Oriented Finite Element Code"**, Advances in Engineering Software. vol. 32, pp 759 – 767.

[11] MARCZAK R.J., (2006) **"Object-Oriented Numerical Integration a Template Scheme for FEM and BEM Applications"**, *Advances in Engineering Software*, vol. 37, pp. 172-183.

[12] UNIVERSITÄT STUTTGART **"Boundary Element Methods - Numerical Solutions for Engineering Problems."** *[Pagina Web en línea disponible en : http://www.iam.unistuttgart.de/bem/home_bem_introduc.html] [Fecha de consulta: 05/06/2009]*

[13] KIRKUP, S., (2007) **"The boundary element method in acoustics"**, 1 Ed. Integrated sound software.

[14] DR. MIRANDA, J. C., DR. MUCI, K. H. **"Boundary Elements and Their Application in Engineering."** *[Pagina Web en línea disponible en:http://dim.tol.itesm.mx/proyectos/bem/contents.htm]. [Fecha de consulta: 05/06/2009]*

[15] KATSIKADELIS, J., (2002) **"Boundary elements: theory and applications"**, 1 Ed. El Selvier Science. Ltd.

[16] GAMEZ, B., DIVO, D., KASSAB, A., *et al*. (2007) **"Análisis de problemas elásticos 2d utilizando la técnica de descomposición de dominio paralelo-iterativa y el método de los elementos de contorno."** *Rev. Fac. Ing. UCV*, 2007, vol.22, no.1, p.107-113. ISSN 0798-4065.

[17] SOMERVILLE, I., (2005) **"Ingenieria de software"**, 7 Ed. Adisson Wesley

[18] PRESSMAN, R., (2005) **"Ingenieria de software: un enfoque práctico"**, 6 Ed. McGraw-Hill. Mexico.

[19] RUMBAUGH, J., BLAHA, W., HEDÍ F., LORENSEN W., (1996) **"Modelado y diseño orientados a objetos Metodología OMT"**, Pearson Prentice Hall. Mexico.

[20] AGUILAR, L., (1998) **"Programación Orientada a Objetos."** 2ª Edición. Mcgraw Hill Interamericana,

[21] DEITEL, H. M. y DEITEL, P. J., (2003) **"Como Programar C++."** 4ta Edición. Pearson Prentice Hall.

[22] RUMBAUGH J. (2007) **"El Lenguaje Unificado de Modelado. Manual de Referencia",** 2da Edición. Pearson Prentice Hall.

[23] LARMAN, C., (2006) **"UML y Patrones"** Prentice-Hall. México.

BIBLIOGRAFÍA ADICIONAL

- ROJAS G., C.I. (2007). **"Manual para la Elaboración del Proyecto y de la Tesis de Pregrado en Ingeniería".** Comisión de grado. Escuela de Ingeniería. UDO, Puerto la Cruz.

- FEDUPEL. (2006). **"Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales".** 4ª Edición, FEDUPEL Caracas.

- Universidad Pedagógica Experimental Libertador. (2006). **"Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales".** Editorial Pedagógica de Venezuela. 4ª Edición.

- JACOBSON, I., BOOCH, GRADY y RUMBAUGH, J., (2000) **"El Proceso Unificado de Desarrollo de Software"**, *Pearson Addisson-Wesley.*

UMETADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	"DESARROLLO DE UN SOFTWARE ORIENTADO A OBJETOS, QUE USA EL MÉTODO DE ELEEMNTOS DE FRONTERA, PARA RESOLVER PROBLEMAS BIDIMENSIONALES DE ELASTICIDAD (ESFUERZO PLANO/DEFORMACIÓN PLANA)"
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CULAC / E MAIL
Lara Rojas Manuel Rafael	**CVLAC:** 18.228.072 **E MAIL:** m.lara.rojas@gmail.com
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**

PALÁBRAS O FRASES CLAVES:

MÉTODO DE ELEMENTOS DE FRONTERA
JAVA NATIVE INTERFACE
JAVA
C++
POO
OMT

UMETADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÀREA	SUBÁREA
INGENIERÍA	Ingeniería de Computación

RESUMEN (ABSTRACT):

En la ingeniería mecánica es frecuente plantear problemas elásticos para decidir la adecuación de un diseño. En la actualidad existen múltiples métodos numéricos para la resolución de este tipo de problemas, entre ellos el método de elementos de frontera (BEM, por sus siglas en ingles: *Boundary Element Method*). BEM desempeña un papel importante en cálculos científicos y técnicos, sin embargo, debido a que la mayoría de los códigos computacionales que implementan BEN fueron escritos en lenguaje de programación FORTRAN, estos tienen limitaciones de reusabilidad y expansión ya que el lenguaje de programación FORTRAN es procedimental. En este trabajo se presenta el desarrolló un software orientado a objetos que utiliza el método de elementos de frontera (BEM) para resolver problemas bidimensionales de elasticidad (esfuerzo plano y deformación plana). El software se desarrolló usando los lenguajes: C++ y JAVA, con la intención de aprovechar las características resaltantes de estos dentro de los lenguajes orientados a objeto, como lo son el potencial gráfico de JAVA y la velocidad de cálculo de C++. La conexión de estos lenguajes se hizo mediante Java Native Interface (JNI, por sus siglas en ingles: *Java Native Interface*). Para el desarrollo del proyecto se utilizó la metodología OMT (del ingles: *Object Modeling Technique*), por medio del Lenguaje Unificado de Modelado (UML, por sus siglas en ingles: *Unified Model Lenguage*). Es de destacar que este software puede integrarse con otros métodos como: método de elementos finitos (FEM, por sus siglas en ingles: *Finite Element Method*), método de volumen finito (FVM, por sus siglas en ingles: *Finite Volume Method*), método de diferencias finitas (FDM, por sus siglas en ingles: *Finite Diference Method*) y formar un solo software usando los conceptos de herencia, abstracción y polimorfismo. El software, permite mejorar y experimentar con BEM colaborando así con las investigaciones en esta área.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Gomes A., Carlos J.	**ROL**	**CA**	**AS X**	**TU**	**JU**
	CVLAC:	10.305.208			
	E_MAIL	carlos.j.gomes@gmail.com			
	E_MAIL				
Cortinez, Claudio	**ROL**	**CA**	**AS**	**TU**	**JU X**
	CVLAC:	12.155.334			
	E_MAIL	Cl_cortinez@cantv.net			
	E_MAIL				
Bastardo, José	**ROL**	**CA**	**AS**	**TU**	**JU X**
	CVLAC:	6.890.832			
	E_MAIL	josebastardo@gmail.com			
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2010	04	23
AÑO	**MES**	**DÍA**

LENGUAJE. SPA

UMETADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
TESIS. OOBEM.doc	Application/msword

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL:

TEMPORAL:

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:
Ingeniero en Computación

NIVEL ASOCIADO CON EL TRABAJO:
Pregrado

ÁREA DE ESTUDIO:
Departamento de Computación y Sistemas.

INSTITUCIÓN:
Universidad de Oriente. Núcleo de Anzoátegui

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

De acuerdo al Artículo 41 del Reglamento de trabajos de grado:

"Los trabajos de grado son de exclusiva propiedad de la Universidad de Oriente y sólo podrán ser utilizados a otros fines con el consentimiento del consejo de núcleo respectivo, el cual lo participará al consejo universitario".

Manuel Rafael Lara Rojas
AUTOR

Ing. Carlos J. Gomes. A., M.Sc, PhD.
TUTOR ACADEMICO

Ing. Claudio Cortinez
JURADO

Lic. José Bastardo, M.Sc.
JURADO

POR LA SUBCOMISIÓN DE TRABAJO DE GRADO

Lic. José Bastardo, M.Sc.