# Universidad de las Ciencias Informáticas

# Facultad 7


## Título: Implementación de un componente de software para la visualización tridimensional de Neuroimágenes

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

**Autores:** Filiberto López Palenzuela
Luis Javier Vallejo Mursulí

**Tutores:** Ing. Andro Font Hernández
Ing. Samuel Figueredo Pérez

Ciudad de La Habana, Junio 2009

Año del 50 aniversario del triunfo de la Revolución

## DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmamos la presente a los 12 días del mes de junio del año 2009.

Filiberto López Palenzuela

_____
Firma del autor

Luis Javier Vallejo Mursulí

_____
Firma del autor

Ing. Andro Font Hernández

_____
Firma del tutor

Ing. Samuel Figueredo Pérez

_____
Firma del tutor

*"...Utilizaremos la computación no para fabricar armas de destrucción masiva y exterminar vidas, sino para transmitir conocimientos a otros pueblos..."*

*Fidel Castro, 19 de Junio 2008*

## DATOS DE CONTACTO

### Tutores:

Ing. Andro Font Hernández
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: afhernandez@uci.cu

Ing. Samuel Figueredo Pérez
Universidad de las Ciencias Informáticas, Habana, Cuba.
Email: sfigueredo@uci.cu

# AGRADECIMIENTOS

A la Revolución y a Fidel por hacer posible ésta Universidad de excelencia…

A nuestros padres, por su dedicación y apoyo constante.

A nuestras amistades por estar siempre ahí cuando las necesitamos…

A Andro y Samuel por ser nuestros tutores y ayudarnos con sus críticas y consejos.

A Martha por sus revisiones y sugerencias.

A todas las personas que han contribuido de alguna manera a nuestra formación profesional a través de sus enseñanzas y lecciones, a todas gracias por guiarnos siempre por el buen camino…

# DEDICATORIA

***A mis padres.***

Por haberme dado todo lo que soy como persona, por haberme inculcado los principios, los valores, la perseverancia y el empeño, acompañado siempre de una gran dosis de amor, sin pedir nunca nada a cambio.

***A mi novia.***

Lis, a ella especialmente le dedico esta Tesis. Por su paciencia, por su comprensión, por su empeño, por su amor, por ser tal y como es.

***A mis familiares.***

A mi hermana Alys por ser el ejemplo de una hermana mayor.
A mis abuelos por tanta compresión y cariño que siempre me han dado y a mi bella sobrina Marian.

***A mis amigos.***

A Ulises, Alejandro Clavijo, Maikel David, Heliodoro y Luis Javier, por ser mis amigos y hermanos; por darme siempre su apoyo en todo momento.
También quiero dedicárselo a todos mis compañeros de la universidad con los que he compartido todos estos años de clases.

**Filiberto López Palenzuela**

***A mis padres***

Por ser las personas que más me aman, las más lindas, las que más sufren cuando yo sufro, las más felices cuando feliz soy, por haber hecho de mi lo que hoy soy, por siempre brindarme su confianza, su apoyo incondicional, en fin, por ser los mejores padres que la vida me haya podido dar jamás.

***A mi hermana Rosa, a Pedro, a Pedrin***

Por ser la otra parte de mi corazón, la otra parte que me ha dado fuerzas para llegar a donde hoy estoy.

***A mis familiares***

A mis abuelos, por siempre brindarme su amor incondicional y su cariño desmedido; a mi prima Eglis, a Nelson, por ser tan especiales y cuidar de mi mayor tesoro, mis padres, a mí prima Lily, a Pavel, a todos mis tíos y primos.

***A mis amigos***

A Ian Carlos, Henry y Randy, por ser mis amigos eternos, a Kenia, a Clendi y Annalay, por ser mis mejores amigas, a mi amigo y hermano Fily, a Yordanger, por ser el que nunca me ha fallado, el que siempre me enseña, el que siempre está ahí cuando lo necesito, por ser el más grande de mis amigos y al que he aprendido a querer como un hermano.

***A todos mis compañeros, a todas mis amistades***

Por ser los mejores, por ser de los que he aprendido tanto, por compartir estos años tan maravillosos junto conmigo.

**Luis Javier Vallejo Mursulí**

## RESUMEN

La presente investigación surge en el marco de trabajo del Proyecto: “Sistema de procesamiento, fusión y visualización tridimensional de Neuroimágenes”, convenio de colaboración entre el Grupo de Procesamiento de Imágenes del Centro Internacional de Restauración Neurológica, CIREN y el Grupo de Procesamiento de Imágenes de la Universidad de Ciencias Informáticas, UCI. En ésta se realizará la implementación de un componente de software para la visualización tridimensional de Neuroimágenes. La visualización manejará la segmentación de regiones anatómicas de interés, y la reconstrucción tridimensional permitirá visualizar estructuras anatómicas en el espacio y sobre cualquier ángulo. La aplicación será compatible con los formatos de imágenes más comunes y completos para este fin, tales como: DICOM (Digital Imaging and Communications in Medicine), y ANALYZE.



## PALABRAS CLAVE

Componente, neurocirugía, neuroimagen, reconstrucción tridimensional, visualización tridimensional, segmentación de imágenes.

**TABLA DE CONTENIDOS**

AGRADECIMIENTOS ..... I
DEDICATORIA ..... II
RESUMEN ..... IV
PALABRAS CLAVE ..... IV
INTRODUCCIÓN ..... 1
CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA ..... 5
1.1 COMPONENTE DE SOFTWARE ..... 5
1.2 IMAGEN MÉDICA ..... 6
1.2.1 Neuroimagen ..... 6
1.2.2 Modalidades de neuroimágenes ..... 6
1.2.3 Formatos de representación de neuroimágenes ..... 7
1.3 SEGMENTACIÓN DE IMÁGENES ..... 9
1.3.1 Segmentación por umbralización ..... 9
1.3.2 Segmentación por crecimiento de regiones ..... 10
1.4 VISUALIZACIÓN TRIDIMENSIONAL ..... 11
1.4.1 Surface rendering ..... 12
1.4.2 Volume rendering ..... 13
1.4.3 Librerías para la visualización tridimensional ..... 14
1.5 SISTEMAS DE CIRUGÍA ASISTIDA POR ORDENADOR ..... 16
1.5.1 Leksell SurgiPlan ..... 18
1.5.2 Win-Neus 2.0 ..... 18
1.5.3 STASISS ..... 19
1.6 LENGUAJES DE PROGRAMACIÓN ..... 19
1.6.1 Lenguaje de programación C / C++ ..... 20
1.6.2 Microsoft Visual Basic / Microsoft Visual Basic.NET ..... 20
1.6.3 Java ..... 21
1.6.4 C# ..... 21
1.7 METODOLOGÍA DE DESARROLLO ..... 23
1.7.1 Extreme Programming ..... 23
1.7.2 Open Unified Process ..... 24
1.7.3 Rational Unified Process ..... 24
1.8 HERRAMIENTAS A UTILIZAR ..... 25
1.8.1 Enterprise Architect ..... 25
1.8.2 Visual Studio 2008 ..... 26
1.8.3 Mono Migration Analyzer 2.x ..... 27
1.9 PATRONES DE DISEÑO ..... 28
1.9.1 Patrones Creacionales ..... 28
1.9.2 Patrones GRASP ..... 28
1.10 RESULTADOS ESPERADOS ..... 29
CONCLUSIONES ..... 29
CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA ..... 30
2.1 BREVE DESCRIPCIÓN DEL SISTEMA ..... 30
2.2 MODELO DE DOMINIO ..... 30
2.3 ESPECIFICACIÓN DE LOS REQUISITOS DE SOFTWARE ..... 32
2.3.1 Requisitos Funcionales ..... 32
2.3.2 Requisitos No Funcionales ..... 33
2.4 DEFINICION DE LOS CASOS DE USO DEL SISTEMA ..... 34

2.4.1 Actor del sistema ..... 34
2.4.2 Diagrama de Casos de Uso del Sistema ..... 35
2.4.3 Casos de Uso por ciclo ..... 36
2.4.4 Casos de Uso expandidos ..... 36
CONCLUSIONES ..... 43
CAPÍTULO 3 DISEÑO DEL COMPONENTE ..... 44
3.1 ESTILO ARQUITECTÓNICO UTILIZADO ..... 44
3.2 PATRONES UTILIZADOS ..... 45
3.2.1 Patrón SINGLENTON ..... 45
3.2.2 Patrón Experto ..... 45
3.2.3 Patrón Creador ..... 46
3.2.4 Patrón Polimorfismo ..... 46
3.3 DIAGRAMAS DE CLASES ..... 47
3.4 DIAGRAMAS DE SECUENCIAS ..... 54
CONCLUSIONES ..... 62
CAPÍTULO 4 IMPLEMENTACIÓN DEL COMPONENTE ..... 63
4.1 DIAGRAMA DE COMPONENTES ..... 63
4.2 FRAGMENTOS DE CÓDIGO ..... 64
4.2.1 Reconstrucción Tridimensional ..... 64
4.2.2 Segmentación de Imágenes ..... 69
CONCLUSIONES ..... 72
BENEFICIOS ESPERADOS E IMPACTO ..... 73
CONCLUSIONES GENERALES ..... 74
RECOMENDACIONES ..... 75
REFERENCIAS BIBLIOGRÁFICAS ..... 76
BIBLIOGRAFÍA ..... 79
ANEXOS ..... 82
Anexo 1: Estructura de los DataSet que forman la cabecera de un archivo DICOM. ..... 82
Anexo 2: Diagrama de herencia de vtkDataObject. ..... 83
Anexo 3: DataSets que utiliza VTK. ..... 83
Anexo 4: Process Objects. ..... 84
Anexo 5: Leksell SurgiPlan. ..... 85
Anexo 6: Win Neus 2.0. ..... 86
Anexo 7: Stassis ..... 87
Anexo 8: Interfaz Vista principal. ..... 88
Anexo 9: Interfaz Vista de segmentación. ..... 89
Anexo 10: Interfaz Vista tridimensional (1). ..... 90
Anexo 11: Interfaz Vista tridimensional (2). ..... 91
Anexo 12: Objeto tridimensional (1). ..... 92
Anexo 13: Objeto tridimensional (2). ..... 93
Anexo 14: Imágenes segmentadas con el componente de software de esta investigación. ..... 94
Anexo 15: Imágenes combinadas con el componente de software de esta investigación. ..... 94
Anexo 16: Compatibilidad con la plataforma libre Mono ..... 95
GLOSARIO ..... 96

**TABLA DE ILUSTRACIONES**

Fig. 1 Imágenes segmentadas por algoritmo Connected Theshold ..... 11
Fig. 2 Imágenes segmentadas por algoritmo Neighborhood Connected ..... 11
Fig. 3 Funcionamiento de los sistemas CAO ..... 17
Fig. 4 Modelo de Dominio. ..... 31
Fig. 5 Casos de Uso del Sistema. ..... 35
Fig. 6 Diagrama Modelo Vista Controlador. ..... 44
Fig. 7 Patrón Singleton. ..... 45
Fig. 8 Patrón Experto ..... 46
Fig. 9 Patrón Creador. ..... 46
Fig. 10 Patrón Polimorfismo. ..... 47
Fig. 11 CU Aplicar Zoom sobre la Imagen Tridimensional ..... 48
Fig. 12 CU Aplicar filtro sobre la Imagen ..... 49
Fig. 13 CU Reconstruir Imagen Tridimensionalmente ..... 50
Fig. 14 CU Rotar Imagen ..... 51
Fig. 15 CU Segmentar Imagen ..... 52
Fig. 16 CU Visualizar Imagen ..... 53
Fig. 17 Diagrama de interacción (Aplicar Zoom sobre Imagen Tridimensional) ..... 56
Fig. 18 Diagrama de interacción (Aplicar filtro sobre Imagen) ..... 57
Fig. 19 Diagrama de interacción (Reconstruir Imagen Tridimensional) ..... 58
Fig. 20 Diagrama de interacción (Rotar Imagen) ..... 59
Fig. 21 Diagrama de interacción (Segmentar Imagen) ..... 60
Fig. 22 Diagrama de interacción (Visualizar Imagen) ..... 61
Fig. 23 Diagrama de Componentes ..... 64

# INTRODUCCIÓN

La Neurocirugía es una de las especialidades quirúrgicas más exigentes en cuanto a precisión en el procedimiento quirúrgico. Si una estructura anatómica cerebral es equivocadamente dañada debido a un mínimo de error de posicionamiento, las consecuencias para el paciente pueden suponer daños funcionales irreparables y en algunos casos la muerte. [1]

La neurocirugía tradicional era un ejercicio técnico cualitativo, consistente en técnicas manuales basadas en la coordinación mano-vista del cirujano. Por ello, las intervenciones quirúrgicas eran más largas y costosas de lo necesario, por lo que han ocurrido casos muy difíciles por las localizaciones de las lesiones a tratar, y se han presentado secuelas importantes para el paciente, debido a la excesiva manipulación de la anatomía cerebral durante la intervención.

El uso de sistemas computarizados para asistir a la neurocirugía garantiza la realización de cálculos precisos para el abordaje quirúrgico, lo que posibilita la realización de procederes con alto nivel de exactitud y seguridad y la posibilidad de utilizar múltiples facilidades de procesamiento de imágenes. Estas técnicas persiguen el acceso a blancos intracraneales con un mínimo error espacial, de ahí que en las últimas décadas una gran mayoría de los procedimientos en neurocirugía, sean de una u otra forma asistidos por computadoras.

El desarrollo de la neurocirugía estereotáctica[1] ha estado estrechamente ligado a la introducción y desarrollo de las técnicas tomográficas de obtención de imágenes médicas, así como a los avances en el campo de la informática. Las imágenes estructurales de alta resolución como la Tomografía Axial Computarizada (TAC) y la Resonancia Magnética Nuclear (RMN) aportan información sobre la anatomía cerebral y han evolucionado la Radiología y con ello la Neurocirugía. Mediante dichas técnicas, el clínico puede ver la lesión a tratar y su ubicación exacta con relación a la anatomía normal.

Cualquier intervención actual de Neurocirugía comienza con una fase de adquisición de datos con objeto de definir las lesiones patológicas a tratar y su relación espacial con la anatomía normal del paciente. Dicha fase de adquisición incluye ineludiblemente un profundo examen radiológico de la anatomía craneal del paciente, para ello se utilizan estudios de TAC y Resonancia Magnética (RM). De forma ocasional se le practican otros estudios al paciente que ayuden a corroborar los diagnósticos realizados mediante TAC y RM, los mismos incluyen Angiografía Digital de Sustracción (DSA), Tomografía de Emisión de Positrones (PET), Tomografía de Emisión de Fotón Simple (SPECT) y

[1] **Neurocirugía estereotáctica:** técnica tridimensional de Neurocirugía. Consiste en tomar algunas radiografías (u otro tipo de imágenes), basado en las cuales se identifica y se mapea una estructura (núcleo) dentro del cerebro.

Electroencefalografía (EEG).

La Neurocirugía moderna ha asimilado las ventajas tanto de las técnicas de adquisición de imágenes más avanzadas como de los métodos de post-procesamiento. Estos últimos incluyen: reconstrucción de cortes en diferentes planos espaciales, corregistro y fusión de imágenes de diferentes modalidades, y representación tridimensional de imágenes.

En el mundo existen sistemas muy avanzados diseñados para planificar, asistir y simular la neurocirugía, los que se caracterizan por un alto nivel tecnológico. Estos sistemas son conocidos como Cirugía Asistida por Ordenador (CAO), los cuales se dividen en tres fases fundamentales: la de adquisición, la de planificación y la de intervención.

En Cuba existe el STASSIS, un sistema desarrollado por el Centro Internacional de Restauración Neurológica (CIREN) que integra facilidades para la planificación de diferentes técnicas quirúrgicas, haciendo uso de imágenes 2D. Sin embargo la interpretación de imágenes 2D requiere de un mayor entrenamiento para imaginar el carácter tridimensional de las estructuras y lesiones. La posibilidad de usar y combinar modernas modalidades de imágenes, así como la extensión de estos sistemas al tratamiento de imágenes 3D, permiten la obtención de mejores resultados en la aplicación de técnicas como la neurocirugía estereotáctica y terapias de radiación, donde el conocimiento de la forma tridimensional de las lesiones, así como su ubicación espacial exacta en el cerebro resulta indispensable.

Por ello se plantea como **problema científico**: ¿Cómo asistir a la planificación quirúrgica, haciendo uso de las técnicas de reconstrucción tridimensional de neuroimágenes? El **objeto de estudio** son los sistemas de procesamiento, fusión y visualización de neuroimágenes; y el **campo de acción**, las técnicas de reconstrucción tridimensional de neuroimágenes.

Para dar solución al problema científico declarado, esta investigación tiene como **objetivo general** implementar un componente de software que asista a la planificación quirúrgica, haciendo uso de las técnicas de reconstrucción tridimensional de neuroimágenes.

A partir de este objetivo general se trazaron los siguientes **objetivos específicos:**

- Analizar las técnicas y métodos existentes para la reconstrucción tridimensional de neuroimágenes.
- Especificar los requisitos del componente de software.
- Diseñar un componente de software que asista las intervenciones quirúrgicas, a través de las

técnicas de reconstrucción tridimensional de neuroimágenes.

- Implementar el componente de software diseñado.

Los que se cumplirán a través de las siguientes **tareas de la investigación:**

- Selección de los formatos de entrada de las imágenes médicas.
- Selección de los métodos de segmentación de imágenes de TAC y RMN.
- Selección de las técnicas de reconstrucción tridimensional.
- Análisis y descripción del sistema modelado para el procesamiento, fusión y visualización tridimensional de neuroimágenes.
- Definición de los requisitos funcionales y no funcionales del componente de software.
- Descripción de los casos de uso del componente de software.
- Selección de los patrones de diseño.
- Confección del diagrama de clases del diseño.
- Confección de los diagramas de interacción del diseño.
- Implementación del componente de software.

El presente trabajo de diploma está estructurado en 4 capítulos de los que a continuación se describe su contenido:

**Capítulo 1: Fundamentación Teórica.** En este capítulo se aborda el tema del proceso de visualización tridimensional de neuroimágenes, así como en qué contexto se encuentra enmarcado dentro de los sistemas de Cirugía Asistida por Ordenador (CAO). Además se realiza un estudio del estado del arte de los principales sistemas de procesamiento, fusión y visualización tridimensional que asisten la planificación de las intervenciones quirúrgicas en la rama de la neurología. También se analizan las principales técnicas, tecnologías, herramientas y metodologías utilizadas para el desarrollo del componente.

**Capítulo 2: Características del Sistema:** En este capítulo se presenta la propuesta de solución del sistema para la situación problémica. Se muestran los procesos del negocio mediante el modelo de

dominio y las características y funcionalidades que tendrá el sistema a partir de los requerimientos funcionales y no funcionales del componente de software a desarrollar. Se presenta el diagrama de casos de uso del sistema con las correspondientes especificaciones de los casos de uso asociados al componente.

**Capítulo 3: Diseño del Componente.** Este capítulo describe como implementar el componente, a través del diseño, enfocado a cómo el sistema cumple sus objetivos teniendo en cuenta los requisitos funcionales y no funcionales. Se realizan los diagramas de clases y los diagramas de interacción según los casos de uso definidos y se explica además la arquitectura utilizada, así como el empleo de los patrones de diseño.

**Capítulo 4: Implementación del Componente.** En este capítulo se describe cómo fue implementado el componente. Se presenta el diagrama de componentes, que muestra la distribución física de los componentes para la implementación. Se describen los fragmentos relevantes del código.

# CAPÍTULO 1
# FUNDAMENTACIÓN TEÓRICA

En este capítulo se aborda el tema del proceso de visualización tridimensional de neuroimágenes, así como en qué contexto se encuentra enmarcado dentro de los sistemas de Cirugía Asistida por Ordenador (CAO). Además se realiza un estudio del estado del arte de los principales sistemas de procesamiento, fusión y visualización tridimensional que asisten la planificación de las intervenciones quirúrgicas en la rama de la neurología. También se analizan las principales técnicas, tecnologías, herramientas y metodologías utilizadas para el desarrollo del componente.

## 1.1 COMPONENTE DE SOFTWARE

Un componente de software es una unidad de composición con interfaces contractualmente especificadas y explícitas sólo con dependencias dentro de un contexto. Un componente de software puede ser desplegado independientemente y es sujeto a la composición de terceros. [2]

Características claves para que un elemento pueda ser catalogado como componente:

- Identificable: Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.
- Auto contenido: Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.
- Puede ser remplazado por otro componente: Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.
- Con acceso solamente a través de su interfaz: Debe asegurar que estas no cambiaran a lo largo de su implementación.
- Sus servicios no varían: Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.

- Bien Documentado: Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros y adaptarlo.
- Es genérico: Sus servicios debe servir para varias aplicaciones.
- Reutilizado dinámicamente: Puede ser cargado en tiempo de ejecución en una aplicación.
- Independiente de la plataforma: Hardware, Software, S.O.[3]

### 1.2 IMAGEN MÉDICA

Una imagen médica es la representación de distribución espacial de una o más propiedades físicas o químicas dentro del cuerpo humano. [476]

Existen dos parámetros importantes en una imagen médica:

- Contraste: qué es lo que se ve en la imagen.
- Resolución: grado de detalle, puede ser espacial y temporal.

#### 1.2.1 Neuroimagen

La neuroimagen es un conjunto de técnicas utilizadas en Neurología para realizar el diagnóstico mediante la obtención de imágenes de los órganos y tejidos del sistema nervioso. [5]

La neuroimagen es la expresión fisiológica del cerebro. Hoy, la resonancia magnética funcional ofrece una película de la actividad cerebral. [6]

#### 1.2.2 Modalidades de neuroimágenes

Las modalidades de las neuroimágenes se pueden agrupar en dos grupos diferentes según la información que proporcionan al especialista, neuroimágenes anatómicas o estructurales y las neuroimágenes funcionales.

Las neuroimágenes anatómicas o estructurales son las que describen o brindan a los especialistas, información detallada acerca de la anatomía cerebral, dígase tejidos, disposición de órganos, existencia de otras estructuras relevantes, etc.; dentro de las mismas se encuentran modalidades como las TAC y las RM.

Las neuroimágenes funcionales por su parte brindan a los especialistas información detallada del funcionamiento de la anatomía cerebral, al mostrar las estructuras, órganos y áreas cerebrales que se activan ante cierto estímulo; dentro de este grupo se encuentran modalidades como las Positron

Emission Tomography (PET), Magnetoencefalografía (MEG), Electroencefalografía (EEG), Resonancia Magnética funcional (fMRI), y otras.

### 1.2.3 Formatos de representación de neuroimágenes

La totalidad de los sistemas de procesamiento, fusión y visualización tridimensional de neuroimágenes utilizan archivos de formatos especialmente diseñados para imágenes médicas, a continuación se exponen algunos de dichos formatos con sus principales características.

#### *1.2.3.1 DICOM*

El formato DICOM, es el formato contemplado en el estándar DICOM 3.0, para el almacenamiento, procesamiento y transmisión de imágenes médicas, el cual fue creado por ACR (American College of Radiology) y NEMA (National Electrics Manufacturers Association), para lograr una estandarización dentro de los fabricantes de equipos de adquisición de imágenes médicas. Por su uso a nivel mundial por parte de los sistemas informáticos que implementan el estándar DICOM, constituye uno de los formatos de almacenamiento de imágenes médicas más extendidos y populares que existen.

Este estándar define: las estructuras de datos para las imágenes médicas y la información relacionada, ciertos servicios orientados a las comunicaciones (transmisión de imágenes, consulta de un archivo de imagen, etc.) los formatos para el intercambio de los datos almacenados, y los requisitos de compatibilidad para dispositivos y programas. Internamente, un fichero DICOM está formado por dos componentes: [7]

- El primero es una cabecera que contiene un número variable de conjuntos de datos o DataSet. Cada DataSet se compone de varios elementos de datos o DataElement. Ver Anexo 1.

Define la información asociada a la imagen médica, así por ejemplo se tienen elementos de datos tales como: nombre del paciente, edad, espacio entre pixeles, modalidad y dimensión de la imagen y otros, almacenados en un archivo único (.dcm).

- El segundo componente de un fichero DICOM contiene una o más imágenes, es decir, el flujo de bytes donde se codifican las imágenes en sí. No debe confundirse con los datos de la imagen (capas, bits dedicados, filas, columnas, etc.) que se codifican en la cabecera.

Los datos de imagen pueden estar comprimidos usando gran variedad de estándares, incluidos JPEG, JPEG Lossless, JPEG 2000, LZW y Run-length encoding (RLE).

#### *1.2.3.2 ANALYZE 7.5*

El formato ANALYZE 7.5, fue el formato creado para ser usado en el software de procesamiento de imágenes médicas ANALYZE, es un formato ampliamente usado en el campo de las neuroimágenes funcionales, ha sido implementado por numerosos programas comerciales de procesamiento de imágenes médicas como son SPM, FreeSurfer, MRIco, etc.

Entre las principales características del formato ANALYZE 7.5, se encuentran:

- La información se encuentra almacenada en dos archivos, un archivo imagen (.img), que posee datos como son el flujo de vóxeles, tipo de dato y orden que presentan los mismos, los cuales están descritos por un archivo cabecera (.hdr), que posee información acerca de las dimensiones de las imágenes, datos del paciente, y otros.
- Es un formato flexible y extensible, pues permite la inclusión de nuevos tipos de datos si el usuario los necesitara. [8]

#### *1.2.3.3 NIfTI 1.1*

El formato NIfTI 1.1, es el formato propuesto por el DFWG (Data Format Working Group) de NIfTI (Neuroimaging Informatics Technology Initiative), la estructura del mismo está basada en su gran mayoría en la del formato ANALYZE 7.5, pero NIfTI 1.1 introduce una serie de mejoras enmarcadas sobre todo para facilitar el análisis de datos en las fMRI.

Entre las principales características de NIfTI 1.1, se encuentran:

- La información puede ser almacenada ya sea en el formato archivo cabecera-archivo imagen (.hdr y .img) al igual que en el formato ANALYZE 7.5 o también puede ser almacenada en un archivo único (.nii).
- Mantiene la compatibilidad con programas que utilizan como formato de almacenamiento de imágenes médicas el formato ANALYZE 7.5.
- Ha extendido las capacidades que ya existían en el formato ANALYZE 7.5, dígase desde nuevos tipos de datos, códigos y parámetros para lograr una mejor descripción del significado de los datos hasta la modificación del propósito de campos ya existentes en la cabecera del formato ANALYZE 7.5. [9]

## 1.3 SEGMENTACIÓN DE IMÁGENES

La segmentación es la separación de estructuras de interés a partir del fondo de una imagen, además es un elemento esencial para la función de numerosos algoritmos que se han desarrollado en el campo de procesamiento de imágenes. Normalmente, la segmentación de un objeto o bien se logra mediante la identificación de todos los píxeles[2] o voxels[3] que pertenecen al objeto o por la localización de los que forma su límite. [10]

La segmentación es un paso imprescindible en diversos procesos de tratamiento de imagen. Entre otros, es necesaria para tomar medidas sobre una región, para realizar reconstrucciones tridimensionales de una zona de la imagen, para la clasificación o diagnóstico automático o para reducir la información de las imágenes. Si de una serie de imágenes para un determinado estudio sólo interesa una región concreta se puede segmentar, y almacenar sólo las regiones para el análisis posterior.

Existen tres formas de segmentación:

- Manual: el usuario segmenta con una herramienta informática.
- Semi-automática: Hay etapas automáticas, pero el usuario sigue el proceso en todo momento.
- Automática: El proceso se realiza sin interacción por parte del usuario.

### 1.3.1 Segmentación por umbralización

La umbralización (thresholding) es un método que busca segmentar imágenes escalares creando una partición binaria de las intensidades de las imágenes. Una umbralización trata de determinar un valor de intensidad, llamado umbral (threshold), que separa las clases deseadas. La segmentación se logra agrupando todos los píxeles con mayor intensidad al umbral en una clase, y el resto de los píxeles en otra clase. La determinación de más de un valor umbral es un proceso llamado multiumbralización (multithresholding). [11]

La umbralización es una técnica efectiva para obtener la segmentación de imágenes donde estructuras diferentes tienen intensidades contrastantes u otras características diferenciables. Generalmente, es el paso inicial de una secuencia de operaciones de procesamiento de imágenes. Su principal limitación es que la umbralización usualmente no toma en cuenta las características espaciales de la imagen y

[2] **Píxel:** es la menor unidad homogénea en color que forma parte de una imagen digital.
[3] **Vóxel:** es la unidad cúbica que compone un objeto tridimensional.

que es sensible al ruido. Estos factores corrompen el histograma de la imagen, haciendo la separación más difícil.

### 1.3.2 Segmentación por crecimiento de regiones

Crecimiento de regiones (region growing) es una técnica para extraer regiones de la imagen que están conectadas según cierto criterio predefinido. Este criterio puede estar basado en información de intensidades y/o bordes de la imagen. En su forma más simple, este método requiere un punto(semilla) que es seleccionado manualmente por el usuario, y extrae todos los píxeles conectados a la semilla, que tengan el mismo valor de intensidad.

Al igual que la umbralización, por lo general no se utiliza la región creciente solamente en una imagen, sino que se utiliza como parte de un conjunto de operaciones de procesamiento de imágenes, particularmente en la delineación de pequeñas y simples estructuras como tumores y lesiones. El crecimiento de regiones también puede ser sensible al ruido, causando que las regiones extraídas tengan agujeros e inclusive que se desconecten. La mayor parte de la complejidad algorítmica de los métodos de crecimiento de regiones está dada por la visita a cada píxel vecino.

Esta investigación implementa la forma de segmentación semi-automática, específicamente el crecimiento de regiones mediante los algoritmos Connected Threshold y Neighborhood Connected; esta forma aprovecha la información de intensidad de cada píxel y las regiones homogéneas.

**Connected Threshold**

La selección que utiliza este algoritmo es establecer un criterio para decidir si un píxel debe incluirse en la actual región o no, este criterio es que el píxel se encuentre en el intervalo de los valores de intensidad definido.

$$I(\mathbf{X}) \in [\text{mínino}, \ \text{máximo}]$$

La Fig. 1 muestra el resultado de aplicar el algoritmo Connected Threshold a una imagen a través de diferentes puntos.

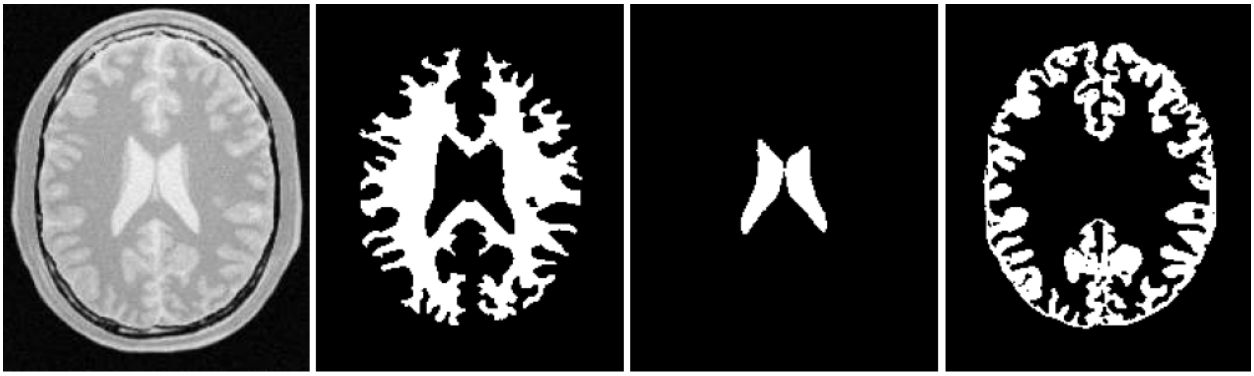


**Fig. 1 Imágenes segmentadas por algoritmo Connected Theshold**

**Neighborhood Connected**

Este algoritmo selecciona el píxel si su valor de intensidad se encuentra en el intervalo definido y si todos los valores de intensidades de sus vecinos se encuentran en dicho intervalo.

La Fig. 2 muestra el resultado de aplicar el algoritmo Neighborhood Connected a una imagen a través de diferentes puntos.

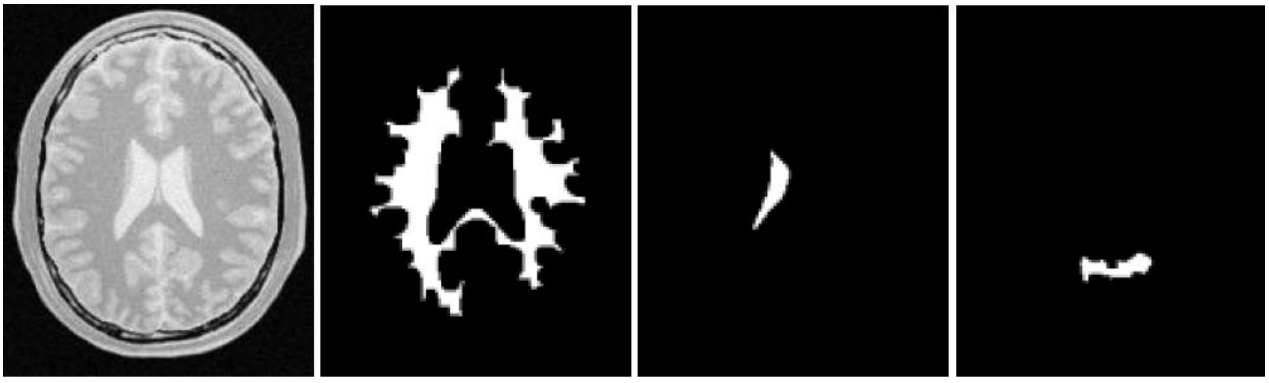


**Fig. 2 Imágenes segmentadas por algoritmo Neighborhood Connected**

La principal desventaja de estos algoritmos es que pueden quedar regiones huecas o no conectadas, por lo que se debe de aplicar antes algún filtro de suavizado.

### 1.4 VISUALIZACIÓN TRIDIMENSIONAL

La visualización tridimensional (3D) de datos científicos o médicos es un área de investigación-desarrollo que ha tenido un acelerado desarrollo en los últimos años. El incremento de las capacidades de los estudios imagenológicos, como TAC y RMN, para adquirir datos volumétricos, unido a la aparición de computadoras personales cada vez más potentes y a menor costo, han beneficiado la introducción en el ambiente clínico de técnicas gráficas para la visualización 3D de imágenes

biomédicas, en particular en el campo de la neurocirugía guiada por imágenes (NGI). Estos datos tridimensionales o volumétricos constituyen la entrada de los diferentes modelos y técnicas de visualización tridimensional.

En el proceso de visualización 3D de volúmenes de datos médicos está implícita la generación de modelos físicamente plausibles, que pueden estar formados por puntos, líneas, superficies, sólidos, y otros. El proceso de creación de imágenes 3D a partir de estos modelos se conoce como *rendering*. De acuerdo al modelo seleccionado, las técnicas de visualización 3D pueden clasificarse en dos grupos: técnicas basadas en la construcción de superficies (*surface rendering*) y técnicas de construcción de volúmenes semitransparentes (*volume rendering*).

### 1.4.1 Surface rendering

En la actualidad las técnicas de surface rendering son las más comunes para visualizar imágenes médicas en 3D. Se caracterizan por aplicar un método para detectar la superficie seleccionada dentro del volumen de datos y luego realizar una representación intermedia de las estructuras encontradas, empleando para esto primitivas geométricas (polígonos, líneas, puntos). Difieren entre ellas principalmente en la selección de las primitivas, la escala en la cual serán definidas y el principio de extracción de las superficies. El modelo resultante puede ser visualizado usando algoritmos convencionales de rendering.

Estas técnicas tienen varias características deseables. Las primitivas geométricas son compactas, lo que hace su almacenamiento y transmisión poco costoso, además tienen un alto grado de coherencia espacial, que garantiza un proceso de rendering eficiente. Por otra parte el poder de estas técnicas es apoyado por la vasta literatura de algoritmos, arquitectura y software que existen para realizar rendering de primitivas geométricas. Se han desarrollado métodos de rendering de superficies muy avanzados, a menudo en asociación con el mapeo de texturas, las cuales generan imágenes de alta calidad.

Estas características la hacen la técnica de elección para la reconstrucción 3D de atlas estereotácticos, donde la información disponible se limita a los contornos de las estructuras, o en situaciones donde es suficiente conocer la representación y ubicación espacial del modelo que describe los objetos de interés para la planificación quirúrgica. Este es el caso de las biopsias y resecciones de tumores, siempre que sea posible una segmentación adecuada de la lesión.

No siempre estas técnicas dan una representación satisfactoria de las estructuras, al no describir el interior del objeto o hacer sólo una representación implícita de éste a partir de su superficie. Con

muchas de ellas se obtienen resultados ambiguos, lo que obliga a la intervención de los usuarios. Por otra parte estas técnicas requieren de una clasificación binaria de los datos (si la superficie pasa o no por las celdas definidas) la cual en presencia de estructuras pequeñas puede provocar errores. Los errores en la clasificación pueden manifestarse en las imágenes generadas como superficies falsas o como agujeros erróneos.

Entre los algoritmos que existen para generar objetos tridimensionales el más utilizado usando esta técnica es el Marching Cubes, el cual difiere del resto de los algoritmos en la forma en que establece los límites del objeto. Para generar estos límites se utiliza generalmente la interpolación lineal.

**Marching Cubes**

La suposición básica de esta técnica es la de que un contorno solo puede atravesar una celda de un número finito de maneras. Se elabora una tabla de casos, que numera todos los estados topológicos posibles de una celda, dadas las combinaciones de valores escalares en los puntos de la celda. El número de estados topológicos depende del número de vértices de la celda, y del número de relaciones de entrada/salida que puede tener cada vértice con respecto a los valores de contorno. Se considera que un vértice se encuentra dentro de un contorno si su valor escalar es mayor que el valor escalar de la línea de contorno. Los vértices con valor escalar menor que el valor de contorno se consideran fuera del contorno. [12]

### 1.4.2 Volume rendering

Las técnicas de volume rendering constituyen una alternativa con múltiples ventajas sobre las de surface rendering, debido a que utiliza información proveniente de todos los vóxel dentro del volumen, esta técnica no está sujeta a limitaciones por pérdida extensiva de información, inherente a la generación de superficies, permitiendo visualizar las no homogeneidades dentro del volumen y ofreciendo una reproducción más realista del objeto. Es por ello, que son más apropiadas en situaciones donde no es posible realizar una segmentación segura, o cuando el contexto es dinámico, es decir, en aquellos casos en que la información contenida en el volumen cambia durante el procedimiento quirúrgico.

Las mayores limitaciones de las técnicas de volume rendering están dadas por su alto costo computacional en términos de memoria RAM (Memoria de Acceso Aleatorio), tiempo de procesamiento y capacidad de almacenamiento. La reconstrucción total de un volumen utilizando volume rendering puede ser varios órdenes de magnitud superior (en términos temporales) al de surface rendering, ya que todos los puntos del volumen de datos participan en la generación de cada imagen, y por tanto el

tiempo de rendering crece linealmente con el tamaño de aquellos. Este hecho ha limitado su aplicación en el ambiente clínico. La aparición de algoritmos más eficientes ha permitido disminuir en parte estas limitaciones. No obstante, para su uso extensivo en aplicaciones neuroquirúrgicas es recomendable utilizar tarjetas gráficas aceleradoras o las más modernas tarjetas gráficas convencionales, cuyo desempeño se acerca al de las tarjetas aceleradoras.

A continuación se describe el algoritmo Ray Casting que implementa dicha técnica.

**Ray Casting**

Técnica que permite encontrar los objetos visibles utilizando 'rayos[4] de luz' que van desde el observador (cámara) hacia la escena. Los rayos pasan a través de cada píxel de la pantalla. De las intersecciones resultantes, se elige aquella que sea la más próxima al observador. [13]

El objetivo básico del Ray Casting es permitir el mejor uso de los datos en tres dimensiones y no tratar de imponer cualquier estructura geométrica en él. Resuelve una de las más importantes limitaciones de las técnicas de extracción de superficies. Las técnicas de extracción de superficie no tiene en cuenta que, particularmente en imágenes médicas, los datos pueden proceder de líquidos y otros materiales que puedan ser parcialmente transparente y debe ser el modelo como tal. Ray casting no sufre de esta limitación. [14]

### 1.4.3 Librerías para la visualización tridimensional

La implementación de funciones y procedimientos gráficos siempre han sido tema de discusión y competencia. Ante este dilema, surgen bibliotecas de código abierto como OpenGl y VTK, las cuales proporcionan al usuario, una herramienta sumamente poderosa para la aplicación de múltiples teorías en la Computación Gráfica. OpenGL y VTK. Se caracterizan esencialmente por su portabilidad y eficiencia en el desarrollo de la graficación, manejando eficazmente texturas, superficies, volúmenes y demás tópicas concernientes a la graficación computacional. Las bibliotecas proveen código básico, lo que permite el desarrollo de sistemas muchos más complejos.

#### *1.4.3.1 OpenGL*

OpenGL es un software de interfaz con el hardware gráfico. Esta biblioteca de rutinas está diseñada para el desarrollo de aplicaciones gráficas interactivas, y es independiente del hardware, por lo tanto,

[4] Rayos: Líneas paramétricas definidas por un origen y una dirección.

OpenGL no hace manejo de ventanas ni de interfaces de usuario. [15]
Es además una biblioteca estilizada de trazado de gráficos de alto rendimiento, hay varias tarjetas gráficas aceleradoras y especializadas en 3D que implementan primitivas OpenGL a nivel hardware.
Todas las aplicaciones OpenGL producen resultados de despliegue visuales consistentes en cualquier (Interfaz de Programación de Aplicaciones) API compleja de OpenGL o hardware, sin tener en cuenta el sistema operativo. OpenGL no es una biblioteca de alto nivel, ya que no posee rutinas que permitan la descripción de objetos complejos, de hecho las primitivas son: puntos, líneas y polígonos. Además no permite el desarrollo de interfaces gráficas por sí solo.

#### *1.4.3.2 Visualization ToolKit (VTK)*

El Visualization ToolKit (VTK) es un software de dominio público, distribuido bajo el modelo Open Source, para computación gráfica en 3D, procesamiento de imágenes y visualización. El diseño de esta biblioteca está fuertemente basado en objetos. [16]
El VTK forma parte de las bibliotecas de alto nivel. VTK contiene una gran variedad de algoritmos de la visualización, incluso métodos escalares, vectoriales, de tensores, de textura, y volumétricos; y técnicas de modelado avanzadas como el modelado implícito, reducción de polígonos, el aplanando de mallas, recorte, contorneando, y triangulación de Delaunay[5]. VTK es un sistema real de visualización ya que incluye capacidades para desplegar primitivas geométricas.
Entender cómo se desarrolla una visualización, es probablemente el aspecto más importante para hacer más eficiente el uso de vtk. La visualización implica dos tipos de clases principales y tres subtipos.

1. Data Objects.
2. Process Objects
    - Sources: archivos, algoritmos, generadores de datos.
    - Filters: convierten un tipo de dato en otro
    - Mappers: mapean datos a primitivas gráficas.

Los Data Objects representan datos de varios tipos, consisten en una estructura geométrica (de puntos y celdas) y topológica, así como en atributos de datos que pueden ser, por ejemplo, escalares o vectores. Los atributos de datos pueden ser asociados con los puntos o celdas del dataset. Las celdas son agrupaciones topológicas de puntos, que forman las unidades del dataset y se usan para interpolar información entre puntos. Ver Anexo 2 y Anexo 3.

[5] **Triangulación de Delaunay:** es una red de triángulos que cumple la condición de Delaunay. Esta condición dice que la circunferencia circunscrita de cada triángulo de la red no debe contener ningún vértice de otro triángulo.

Los Process Objects, también llamados filtros, operan en los Data Objects para generar nuevos Data Objects. Representan los algoritmos del sistema. Process Objects y Data Objects se conectan para formar los pipelines[6] de visualización. Ver Anexo 4.

Para el desarrollo del componente se selecciona la librería VTK debido a que el diseño de esta biblioteca está fuertemente basado en objetos. Además la manera de codificar aplicaciones es más sencilla y rápida que OpenGL, ya que es una biblioteca con rutinas de alto nivel. Es importante destacar que implementa las técnicas de reconstrucción tridimensional como volume rendering y surface rendering, ambas de interés en esta investigación.

### 1.5 SISTEMAS DE CIRUGÍA ASISTIDA POR ORDENADOR

La Cirugía Asistida por Ordenador (CAO) constituye un área tecnológica relativamente reciente que intenta desarrollar y suministrar al cirujano una serie de herramientas que le asistan en la planificación y ejecución de procedimientos quirúrgicos. La idea subyacente en CAO no es la de la sustitución del cirujano en tareas quirúrgicas, hecho imposible de llevar a cabo en la actualidad, sino el desarrollar técnicas y sistemas que permitan ayudar al cirujano en las fases del procedimiento quirúrgico, esto es, diagnóstico, planificación y ejecución.

Los sistemas CAO se dividen en tres fases fundamentales: fase de adquisición, fase de planificación y fase de intervención como se muestra en la Fig. 3.

[6] **Pipelines:** La arquitectura en pipeline consiste en ir transformando el flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior.

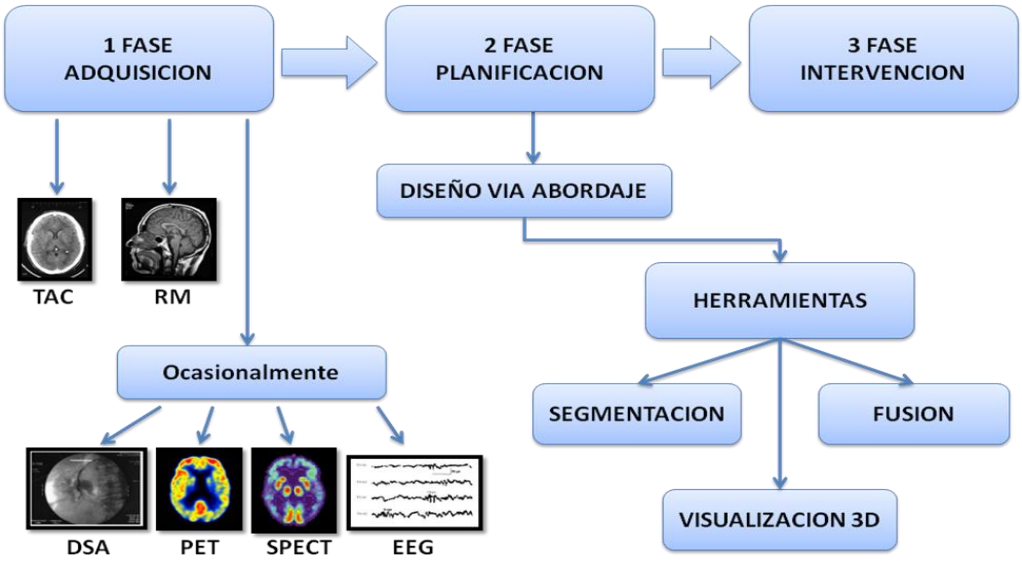



**Fig. 3 Funcionamiento de los sistemas CAO**

Fase de Adquisición: Es la fase donde se adquiere toda la información relevante para emitir un diagnóstico correcto acerca de la patología que afecta al paciente, por ejemplo, la obtención de los estudios imagenológicos como son las TAC, las RM y en ocasiones otros estudios como son las DSA, las PET, las SPECT y las fMRI.

Fase de Planificación: Durante esta fase el especialista apoyado en programas de visualización digital de imágenes médicas planifica la intervención quirúrgica a efectuar posteriormente. Para ello los sistemas de planificación quirúrgica deben disponer de una serie de herramientas entre las que se encuentran:

Visualización Avanzada de Imágenes Médicas: Consiste en permitir al clínico realizar la visualización de varias modalidades a la vez, visión de planos de corte en el espacio en cualquier orientación, filtros de mejoras de imagen, y otras.

Segmentación de Imágenes Médicas: Permite que el clínico reconozca y marque en cada imagen los distintos tejidos y estructuras anatómicas relevantes (huesos, tumor, vasos sanguíneos).

Fusión de Imágenes Médicas: Permite al clínico que pueda fusionar la información de varias modalidades de imágenes diferentes, logrando una representación de la región a estudiar más detallada y rica en información.

Visualización 3D de Imágenes Médicas: Permite al clínico después de haber segmentado las distintas

estructuras, una visualización tridimensional de toda la anatomía del paciente para que de esta manera pueda definir las vías de abordaje óptimas para la intervención quirúrgica.

Fase de Intervención: Es donde el clínico apoyado en la información que obtuvo de las fases previas ejecuta la intervención quirúrgica.

Los sistemas CAO presentan aplicaciones en casi todas las ramas quirúrgicas, una de ellas lo constituyen los sistemas de procesamiento, fusión y visualización tridimensional de neuroimágenes, los cuales tienen su aplicación en la rama de neurocirugía, especialmente en la planificación de cirugía estereotáctica.

En el mundo existen varias empresas que producen estos sistemas de planificación, algunos como línea principal y otros como parte de las soluciones integrales que ofrecen.

### 1.5.1 Leksell SurgiPlan

Es un avanzado programa de planificación neuroquirúrgica basado en neuroimágenes desarrollado por Elekta[7] y que forma parte de la solución integral que brinda dicha empresa para los neurocirujanos. Diseñado especialmente para el marco estereotáctico Leksell Stereotactic System®[8], y dotado de funcionalidades muy útiles para los especialistas, como son vistas en 2D y 3D correlacionadas en tiempo real, señalamiento de forma semiautomática de regiones de interés, entre otras, y apoyado en una agradable y óptima interfaz de usuario, además con la herramienta ImageMerge™[9], la cual brinda funcionalidades de corregistro automático y semiautomático así como de fusión de diferentes modalidades de neuroimágenes. Ver Anexo 5.

### 1.5.2 Win-Neus 2.0

Sistema de planificación para neurocirugía estereotáctica desarrollado por Nuclemed[10], presenta funcionalidades muy útiles para los especialistas que van desde registro y fusión de diferentes modalidades de neuroimágenes, visualización avanzada de neuroimágenes; selección y segmentación de regiones anatómicas relevantes hasta el trazado de trayectorias estereotácticas con el fin de acceder al blanco quirúrgico de la manera que menos afecte al paciente. Este sistema además posee la característica de adaptarse a cualquier sistema estereotáctico. Ver Anexo 6.

[7] **Elekta**: Empresa sueca líder mundial en soluciones integrales para cirugía estereotáctica y radioterapia.
[8] **Leksell Stereotactic System®**: Marco estereotáctico que provee la solución ofrecida por Elekta. Dicho marco estereotáctico fue creado por el profesor Lars Leksell, uno de los fundadores de Elekta.
[9] **ImageMerge™**: Módulo opcional del sistema Leksell SurgiPlan.
[10] **Nuclemed**: Empresa argentina dedicada al desarrollo, producción y distribución de Sistemas de planificación de tratamientos de Radiocirugía-Radioterapia y Sistemas de Planificación para Neurocirugía Estereotáctica.

No obstante las numerosas ventajas que ofrecen estos sistemas, no siempre pueden ser adquiridos por los servicios de neurocirugía debido a los precios que son usualmente altos, lo que sumado a los elevados requerimientos de hardware los convierten en productos significativamente costosos. Utilizando un hardware estándar es posible alcanzar la misma exactitud de estos sistemas, con un rendimiento razonable en condiciones clínicas.

### 1.5.3 STASISS

Durante los años de Revolución el país ha alcanzado un avance sustancial en el campo de la salud. La neurocirugía es una de las disciplinas en la cual se han realizado numerosos avances y que cuenta con un grupo de preparados especialistas que han hecho de ella, una disciplina con un marcado prestigio a nivel mundial.

A raíz de esto, existe como principal solución de planificación de cirugía estereotáctica el STASISS. Este software ha sido creado por especialistas del Grupo de Procesamiento de Imágenes (GPI) del Centro Internacional de Rehabilitación Neurológica (CIREN), y se encuentra desplegado en las principales instituciones médicas con servicio de Neurocirugía del país y en Chile. El mismo posee funcionalidades muy útiles para la planificación neuroquirúrgica, haciendo uso de imágenes 2D, pero carece de herramientas que provean a los especialistas de visualización tridimensional, herramientas que son de vital importancia para este tipo de sistemas. Ver Anexo 7

A partir de esto, y con el propósito de desarrollar una solución que cumpla con las necesidades de los especialistas y que además sea competitiva en el mercado para que así pueda servir como una fuente de ingresos a la economía del país, se ha acordado un proyecto en conjunto entre el GPI del CIREN y el GPI de la Universidad de las Ciencias Informáticas (UCI).

## 1.6 LENGUAJES DE PROGRAMACIÓN

En la actualidad existen diversos lenguajes de programación tales como: C, C++, C#, Microsoft Visual Basic, Microsoft Visual Basic.NET y Java, entre otros.

En ocasiones resulta complejo seleccionar el lenguaje de programación a utilizar para el desarrollo de un proyecto de software determinado. La razón es que elegir un lenguaje de programación depende de muchos factores como lo son: el tipo de programa que se desea realizar, la plataforma requerida para estos programas, incluso siendo poco objetivos, también entra el gusto por un lenguaje en específico.

Una buena elección debería basarse en las fortalezas y debilidades de cada lenguaje para desarrollar una determinada tarea. El problema a resolver debería determinar el lenguaje de programación a

utilizar. [17]

### 1.6.1 Lenguaje de programación C / C++

Los lenguajes de programación C / C++ tienen un soporte muy fuerte en cualquier plataforma o sistema operativo que el programador utilice.

Un programa escrito en C o C++ podrá ser copiado sin dificultad para otra computadora con un entorno diferente, y luego, podrá ser ejecutado con solo algunos cambios, por lo que brindan la posibilidad de escribir los programas solo una vez, y luego modificarlos para que sean capaces de correr en sistemas operativos diferentes.

C / C++ brindan el poder y la flexibilidad de manipular la memoria y acceder directamente al hardware. Esto no solo incrementa la posibilidad de que un error se encuentre alojado en su código, sino que incrementa también la cantidad de tiempo necesaria para depurar el programa con el objetivo de lograr su correcto funcionamiento. [18]

Estos lenguajes, han ido actualizándose lentamente a lo largo de los años pero, en la mayoría de los casos, no ofrecen soluciones integrales para las necesidades actuales, sino tan solo parches para ir resolviéndolas de manera puntual.[19]

La mayor desventaja que presentan estos lenguajes es la dificultad en su aprendizaje, obligan a hacerlo todo manualmente.

### 1.6.2 Microsoft Visual Basic / Microsoft Visual Basic.NET

Visual Basic es uno de los lenguajes de programación más populares para el desarrollo de software. Un programador puede aprender Visual Basic y comenzar a utilizarlo mucho más rápido que si eligiera otro lenguaje de programación.

Visual Basic es fácil de aprender debido a que aísla al programador de los detalles técnicos de la programación de una computadora, a pesar de esto, los programadores profesionales frecuentemente descartan Visual Basic y lo califican como un lenguaje “juguete”, debido a sus limitaciones a la hora de ser utilizado para el desarrollo de aplicaciones. [20]

Su mayor desventaja es en cuanto a su portabilidad, un programa escrito en Visual Basic solamente funciona sobre la familia de sistemas operativos Windows, nunca funcionará en otro tipo de sistema operativo.

Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede

considerar una evolución de Visual Basic implementada sobre el framework .NET. Permite a los desarrolladores centrar el diseño en Windows, el Web y dispositivos móviles. Su introducción resultó muy controvertida, ya que debido a cambios significativos en el lenguaje VB.NET no es compatible hacia atrás con Visual Basic.

### 1.6.3 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Java es un lenguaje de programación completamente portable, por lo que un programa escrito en Java podrá teóricamente correr en cualquier computadora o sistema operativo sin que sea necesaria modificación alguna [21].

Java utiliza un mecanismo llamado recolector automático de basura (automatic garbage collector) para eliminar los objetos creados por el programador en la memoria de la computadora. Esto trae consigo que no existan desbordes de memoria como en C / C++, y que el código sea más limpio y fácil de escribir. La recolección de basura de Java es un proceso prácticamente invisible al desarrollador.

A pesar de sus grandes ventajas, Java también cuenta con un número de desventajas que deberán ser analizadas en el momento de tomar una decisión. Los programas en Java tienden a ser mucho más lentos que sus equivalentes escritos en otros lenguajes de programación. Además, Java es un lenguaje con una curva de aprendizaje bastante elevada.

### 1.6.4 C#

C# es un nuevo lenguaje propuesto por Microsoft para satisfacer las necesidades actuales y de un futuro cercano. C# es, por tanto, una herramienta, como todos los lenguajes de programación, pero adaptada al trabajo actual.

El objetivo de Microsoft ha sido la creación del primer lenguaje orientado a componentes, al estilo de Visual Basic, pero con la flexibilidad y potencia de C++ y sin muchas de sus complejidades. C# ha sido diseñado para una plataforma, la plataforma Microsoft .NET, en la que los servicios son ofrecidos en forma de componentes. Cuenta con construcciones sintácticas nativas para la definición, implementación y consumo de propiedades, métodos y eventos, lo cual le diferencia claramente de C++ o Java.

Para construir un componente con C# no es necesario hacer nada especial aparte de crear una nueva clase. Dicho de otro modo, cualquier clase de objeto C# es un componente, sin necesidad de crear GUID (Globally Unique Identifier) para interfaces y clases de componentes.

C# es un lenguaje completamente orientado a objetos con una gran cantidad de características y mejoras sobre sus predecesores. Al igual que Java, trabaja en un entorno manipulado de memoria, aislando al programador de los engorrosos detalles del hardware. A pesar de esto, el programador tendrá la posibilidad de renunciar a esta característica y lidiar por sí mismo con la manipulación de la memoria [22].

Se dice que su competir más cercano es Java, lenguaje con el que guarda un enorme parecido. En este aspecto, es importante señalar que C# incorpora muchos elementos de los que Java carece como: el rendimiento, el cual es mejor, soporta más tipos primitivos, incluyendo tipos numéricos sin signo, compilación condicional, aplicaciones multi-hilo simplificadas; y otros [23].

Las principales características que definen al lenguaje C# son:

- **Sencillez de uso:** C# elimina muchos elementos añadidos por otros lenguajes y que facilitan su uso y compresión, como por ejemplo ficheros de cabecera, o ficheros fuentes IDL.
- **Modernidad:** Al ser C# un lenguaje de última generación, incorpora elementos que se ha demostrado a lo largo del tiempo que son muy útiles para el programador, como tipos decimales o booleanos, así como una instrucción que permita recorrer colecciones con facilidad (instrucción foreach). Estos elementos hay que simularlos en otros lenguajes como C++ o Java.
- **Orientado a objetos:** C# como lenguaje de última generación es orientado a objetos. Además, C# soporta todas las características del paradigma de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo.
- **Orientado a componentes:** La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular.
- **Recolección de basura:** Todo lenguaje incluido en la plataforma .NET tiene a su disposición el recolector de basura.
- **Eficiente:** En C#, todo el código incluye numerosas restricciones para garantizar su seguridad, no permitiendo el uso de punteros.

- **Compatible:** Para facilitar la migración de programadores de C++ o Java a C#, no sólo se mantiene una sintaxis muy similar a la de los dos anteriores lenguajes, sino que también ofrece la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos, tales como las DLLs de la API de Win32.

Se decide utilizar el lenguaje de programación C#, de la plataforma .NET, para generar los artefactos de tipo software, pues C# es simple, eficaz, orientado a objetos, permite desarrollar aplicaciones rápidamente y mantiene la expresividad y elegancia de los lenguajes de tipo C.

### 1.7 METODOLOGÍA DE DESARROLLO

La metodología de desarrollo es un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software [24]. Una metodología puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo [25].

Por tanto, las metodologías de desarrollo representan el camino para desarrollar aplicaciones informáticas de una manera sistemática. Hoy día, existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Entre las más conocidas se tienen: Programación Extrema (Extreme Programming, XP), Proceso Unificado Abierto (Open Unified Process, OpenUP/Basic) y Proceso Unificado de Desarrollo (Rational Unified Process, RUP).

#### 1.7.1 Extreme Programming

Extreme Programming es una de las metodologías más exitosas en la actualidad y forma parte del grupo de las Metodologías Ágiles, está más orientada a la generación de código con ciclos muy cortos de desarrollo, se dirige a equipos de desarrollo pequeños, hace especial hincapié en aspectos humanos asociados al trabajo en equipo e involucra activamente al cliente en el proceso.

XP es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software [26].

Lo aspectos fundamentales en esta metodología son:

- La simplicidad al desarrollar y codificar los módulos del sistema.
- La comunicación entre los usuarios y los desarrolladores.
- La retroalimentación concreta y frecuente del desarrollo, el cliente y los usuarios finales.

### 1.7.2 Open Unified Process

OpenUP/Basic es un Framework de procesos de desarrollo de software de código abierto. Es un proceso modelo y extensible, dirigido a gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo [27].

Este proceso de desarrollo unificado está basado en RUP, desarrollado por IBM y reconocido mundialmente como uno de los procesos de desarrollo de software de mayor calidad. Plantea que sólo se debe usar los procesos que sean necesarios, que no se debe usar muchos artefactos, además debe acoplarse a las necesidades del usuario permitiendo ser modificado y extendido.

OpenUP está caracterizado por cuatro principios básicos, los cuales son:

- Colaboración para unificar intereses y compartir conocimientos.
- Equilibrio de prioridades competentes a maximizar el valor de los involucrados con el resultado del proyecto.
- Enfoque en la articulación de la arquitectura.
- Desarrollo continuo para obtener retroalimentación y realizar las mejoras respectivas.

### 1.7.3 Rational Unified Process

RUP es una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas probadas y una arquitectura configurable [28]. Es un proceso de desarrollo de software que contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de software. Más que un simple proceso, el proceso de desarrollo de software es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de actitud y diferentes tamaños de proyecto [29].

Las características principales de RUP son:

- **Dirigido por casos de uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan, es una facilidad que el software debe proveer a sus usuarios. A partir de aquí los casos de uso guían el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.

- **Centrado en la arquitectura**: La arquitectura involucra los elementos más significativos del sistema. Surge de las necesidades de la empresa y se refleja en los casos de uso. También se ve influida por otros factores como las plataformas de software, los sistemas operativos, protocolos y requerimientos no funcionales. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de Lenguaje Unificado de Modelado (Unified Modeling Language, UML).
- **Iterativo e incremental:** RUP divide el proceso en cuatro fases, las cuales se desarrollan en iteraciones de las actividades principales básicas de cualquier proceso de desarrollo. Las iteraciones hacen referencia a pasos en los flujos de trabajo, y los incrementos, al crecimiento del producto. Las iteraciones están controladas y se ejecutan de una forma planificada.

Se selecciona RUP como metodología de desarrollo porque provee un entorno de proceso de desarrollo configurable, basado en estándares, permite tener claro y accesible el proceso de desarrollo que se sigue, además de ser configurado según las necesidades de la organización y del proyecto. Utiliza mejores prácticas probadas en la industria y provee a cada participante con la parte del proceso que le compete directamente, filtrando el resto.

### 1.8 HERRAMIENTAS A UTILIZAR

Las herramientas de desarrollo son aquellas aplicaciones que tienen cierta importancia en el desarrollo de un programa. A continuación se describen las características principales de las herramientas Enterprise Architect, Visual Studio 2008 y Mono Migration Analyzer 2.x.

#### 1.8.1 Enterprise Architect

Enterprise Architect (EA) es una herramienta flexible, completa y potente de modelado en UML bajo plataforma Windows. Provee lo más nuevo en desarrollo de sistemas, administración de proyectos y análisis de negocio [30].

EA es una herramienta multi-usuario, con bases construidas sobre la especificación UML 2.0. Diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad. Provee una generación poderosa de documentos y herramientas de reporte con un editor de plantilla completo WYSIWYG[11]. Genera reportes detallados

[11] **WYSIWYG: W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et ("lo que ves es lo que obtienes"). Se aplica a los editores de texto que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso.

y complejos de EA con la información que se necesita en el formato que su compañía o cliente demanda [31].

Una de las características más importantes es que EA soporta generación e ingeniería inversa de código fuente para muchos lenguajes conocidos, incluyendo C++, C#, Java, Delphi, VB.Net, Visual Basic y PHP. También es posible vincularla con Visual Studio.NET, permitiéndole modelar en EA y saltar directamente al código fuente en su editor de .NET.

Se decide utilizar Enterprise Architect como herramienta CASE profesional, ya que soporta el ciclo de vida completo, combina la potencia de la última especificación UML 2.1 con un alto rendimiento y una interfaz intuitiva. Además brinda características como: la generación de código en visual C# 2.0; integración con Visual Studio; soporte para pruebas; usabilidad; velocidad; y otras.

### 1.8.2 Visual Studio 2008

.NET es la nueva tecnología desarrollada por Microsoft con los objetivos principales de mejorar los sistemas operativos y de obtener un entorno diseñado para el desarrollo y ejecución del software en forma de servicios que puedan ser accedidos a través de Internet de forma independiente al lenguaje de programación, sistema operativo y hardware utilizados tanto para desarrollarlos como para publicarlos.

Visual Studio .NET es la herramienta de desarrollo multilenguaje más completa para construir e integrar rápidamente aplicaciones y servicios Web XML (Extensible Markup Language, Lenguaje de Marcas Ampliable). Aumenta de un modo extraordinario la productividad de los desarrolladores y crea nuevas oportunidades de negocio. En su diseño se han integrado a fondo los estándares y protocolos de Internet, como XML y SOAP (Simple Object Access Protocol), por lo que Visual Studio .NET simplifica considerablemente el ciclo de vida del desarrollo de aplicaciones [32].

Visual Studio 2008 tiene un elevado aumento de la productividad al escribir aplicaciones dirigidas a la nueva versión de .NET Framework, esto incluye la ampliación de tipos de proyectos, la reducción de tareas mundanas y siempre en evolución y los aspectos de equipo orientados a la ingeniería de software [33].

A las mejoras de desempeño, escalabilidad y seguridad con respecto a la versión anterior, se agregan entre otras, las siguientes novedades:

- **Mejora en las capacidades de Pruebas Unitarias:** son ejecutas más rápido independientemente de si lo hacen en el entorno IDE o desde la línea de comandos.
- **Visual Studio Tools for Office (VSTO):** integrado con Visual Studio 2008 es posible desarrollar rápidamente aplicaciones de alta calidad basadas en la interfaz de usuario de Office que personalicen la experiencia del usuario y mejoren su productividad en el uso de Word, Excel, PowerPoint, Outlook, Visio, InfoPath y Project.
- **LINQ (Language Integrated Query):** nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a Base de Datos.
- **Soluciones multiplataforma:** Visual Studio 2008 ahora permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .Net Framework: 2.0. (Incluido con Visual Studio 2005), 3.0 (incluido en Windows Vista) y 3.5 (incluido con Visual Studio 2008).

Se decide utilizar Visual Studio 2008 como IDE de desarrollo ya que ofrece la visión de las aplicaciones clientes inteligentes, al permitir a los desarrolladores con avanzadas herramientas de desarrollo, y otras características innovadoras, la creación de aplicaciones de manera rápida a través de diversas plataformas.

El sistema se va a desarrollar con el framework 2.0, ya que se considera estable y brinda la posibilidad de migrar a Visual Studio 2005.

### 1.8.3 Mono Migration Analyzer 2.x

El Mono Migration Analyzer (MoMA) es una herramienta que ayuda a identificar los problemas que se encuentran a la hora de portar las aplicaciones realizadas en la plataforma .NET al proyecto MONO[12]. Ayuda a identificar las zonas que aún no están soportadas por el proyecto MONO [34].

Se decide utilizar esta herramienta pues permite a los desarrolladores cuantificar el número de cambios requeridos para ejecutar sus aplicaciones .NET en un entorno Linux.

[12] **Proyecto Mono:** es un proyecto de código abierto impulsado por Novell para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET

## 1.9 PATRONES DE DISEÑO

Un patrón es una solución recurrente para un problema típico y en un contexto determinado. La solución se refiere a la respuesta al problema, por lo que ayuda a resolver las dificultades de diseño en problemas similares. Los patrones deberían comunicar soluciones de diseño a los desarrolladores y arquitectos que los leen y los utilizan. [35]

A continuación se describen los patrones de diseños que serán utilizados en esta investigación.

### 1.9.1 Patrones Creacionales

Los patrones creacionales abstraen el proceso de instanciación y hacen que el sistema sea independiente de la creación de los objetos. Se utiliza de manera práctica el Singleton de este grupo de patrones creacionales.

#### *1.9.1.1 El patrón Singleton*

Este patrón asegura la creación de una única instancia de un objeto y sólo es posible el acceso global a esa única instancia. Tiene dos características fundamentales:

1. Restricción de acceso al constructor: con esto se logra que sea imposible crear nuevas instancias. Solo la propia clase puede crear la instancia.
2. Mecanismo de acceso a la instancia: el acceso a la instancia única se hace a través de un único punto bien definido, que es gestionado por la propia clase y que puede ser accedido desde cualquier parte del código en principio.

### 1.9.2 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.

#### *1.9.2.1 El Patrón Experto*

El patrón experto asigna una responsabilidad al experto en información, es decir, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.

#### *1.9.2.2 El Patrón Creador*

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos,

tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se conecta con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. El patrón Creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear la instancia contenida o registrada.

#### *1.9.2.3 El Patrón Polimorfismo*

Este patrón se utiliza cuando las alternativas o comportamientos relacionados varían según el tipo (clase), asigna la responsabilidad para el comportamiento, utilizando operaciones polimórficas a los tipos para los que varía el comportamiento. Tiene como ventajas que se añaden fácilmente las extensiones necesarias para nuevas variaciones y las nuevas implementaciones se pueden introducir sin afectar a los clientes.

## 1.10 RESULTADOS ESPERADOS

Obtener un componente de software que a partir de imágenes médicas e imágenes segmentadas por los algoritmos de crecimiento de regiones (NeighborhoodConnected y ConnectedThreshold), reconstruya tridimensionalmente dichas imágenes, según los algoritmos surface y volume rendering. Además debe de soportar los formatos de representación de imágenes médicas ANALYZE y DICOM. El componente debe facilitar el posterior trabajo de los especialistas, brindando la posibilidad de aplicar filtros y paletas predefinidas a estas imágenes.

## CONCLUSIONES

El estudio del proceso de reconstrucción tridimensional de neuroimágenes permitió la selección de las técnicas y algoritmos a utilizar en esta investigación; además quedó evidenciada la necesidad de implementar este componente de software. Se establecieron la metodología y herramientas para el modelado y desarrollo del mismo según sus características y se realizó el estudio de los diferentes tipos de patrones de diseño con el objetivo de brindar claridad y organización en la codificación.

# CAPÍTULO 2
# CARACTERÍSTICAS DEL SISTEMA

En este capítulo se presenta la propuesta de solución del sistema para la situación problémica. Se muestran los procesos del negocio mediante el modelo de dominio y las características y funcionalidades que tendrá el sistema a partir de los requerimientos funcionales y no funcionales del componente de software a desarrollar. Se presenta el diagrama de casos de uso del sistema con las correspondientes especificaciones de los casos de uso asociados al componente.

## 2.1 BREVE DESCRIPCIÓN DEL SISTEMA

Teniendo en cuenta que el sistema que existe de planificación quirúrgica es basado en imágenes bidimensionales, se requiere de un mayor esfuerzo por parte de los médicos cubanos para imaginar el carácter tridimensional de las estructuras anatómicas de los pacientes. El Sistema de procesamiento, fusión y visualización tridimensional de Neuroimágenes, dotará a los médicos cubanos de las herramientas necesarias para realizar un estudio profundo de los casos y planificar la intervención quirúrgica a efectuar posteriormente.

El sistema a desarrollar a través de la presente investigación constituirá un componente de software para la visualización tridimensional de neuroimágenes.

## 2.2 MODELO DE DOMINIO

RUP define en su primera fase de desarrollo la realización del modelo de negocio, con el objetivo de comprender el entorno del cliente y detectar las mejoras potenciales en los procesos de la organización. Cuando no es posible identificar claramente los procesos del negocio, RUP propone realizar un modelo de dominio, que es un subconjunto del modelo de negocio.

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los conceptos que existen o eventos que ocurren en el entorno en que trabaja el sistema. El modelo de dominio se describe específicamente mediante diagramas de clases, utilizando el lenguaje de modelado UML.

Teniendo en cuenta que los procesos de negocio no están claramente definidos, se describe el negocio a través del modelo de dominio (Fig. 4).

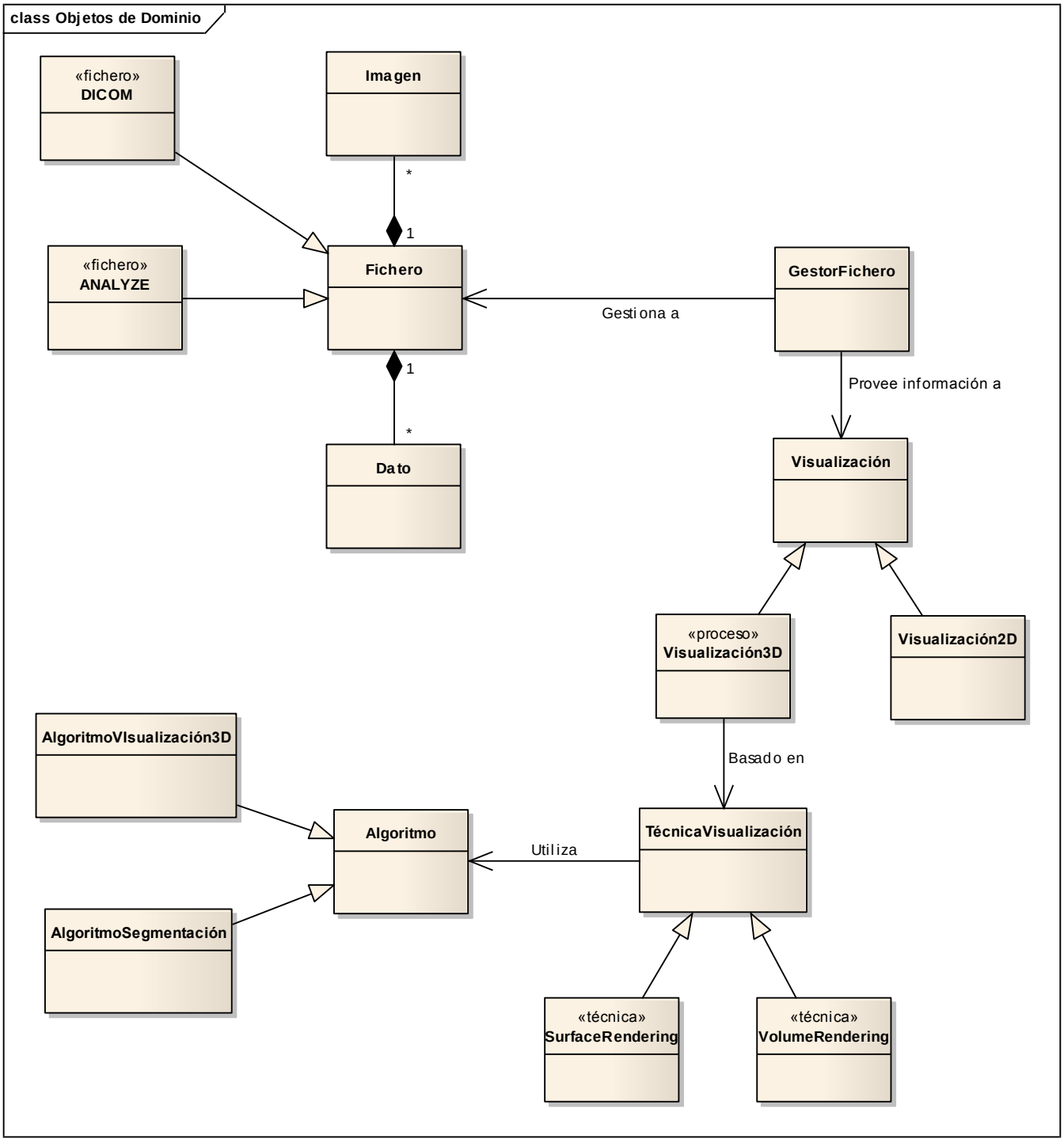



**Fig. 4 Modelo de Dominio.**

A continuación se brinda una descripción de los principales conceptos del modelo de dominio para un mejor entendimiento del contexto que se está describiendo:

**Imagen**: Representa las imágenes por las que están formadas los diferentes ficheros.

**Dato**: Representa los datos que poseen los diferentes ficheros de imágenes médicas tanto del paciente (nombre, edad, sexo) como de las imágenes (cantidad, modalidad).

**Fichero**: Representa una generalización de los ficheros de imágenes médicas y del cual heredan los conceptos que especifican cada uno de los diferentes formatos, dígase **DICOM** y **ANALYZE**.

**GestorFichero**: Concepto que engloba las operaciones necesarias para obtener información (datos e imágenes) de los diferentes ficheros de imágenes médicas.

**Visualización2D**: Representa las operaciones necesarias para la visualización de imágenes en dos dimensiones.

**Visualización3D**: Representa las operaciones necesarias para la visualización de imágenes en tres dimensiones.

**TécnicaVisualización**: Representa una generalización de las técnicas de visualización 3D y constituye una base para los conceptos que describen específicamente cada una de estas técnicas: **SurfaceRendering** y **VolumeRendering**.

**Algoritmo**: Representa una generalización de los diferentes conceptos que describen cada uno de los tipos específicos de algoritmos, ya sean visualización 3D o segmentación.

**AlgoritmoVisualización3D**: Concepto que engloba los diferentes algoritmos utilizados por las técnicas de visualización, para llevar a cabo el proceso de visualización 3D de imágenes.

**AlgoritmoSegmentación**: Concepto que abarca los diferentes algoritmos utilizados para la segmentación de imágenes.

## 2.3 ESPECIFICACIÓN DE LOS REQUISITOS DE SOFTWARE

Los requisitos funcionales y no funcionales muestran las capacidades o condiciones que el sistema debe cumplir y las propiedades o cualidades que el producto debe tener, los cuales en la fase de construcción deben ser posibles de probar o verificar.

### 2.3.1 Requisitos Funcionales

**RF1 VISUALIZAR IMAGEN**

1.1. Visualizar imágenes DICOM simples.

1.2. Visualizar imágenes ANALYZE 7.5.

1.3. Visualizar thumbnails de imágenes.

1.4. Visualizar las imágenes almacenadas en un dispositivo de almacenamiento.

1.5. Visualizar imágenes segmentadas.

1.6. Visualizar imágenes reconstruidas tridimensionalmente.

**RF 2 RECONSTRUIR IMAGEN TRIDIMENSIONAL**

2.1. Reconstruir imágenes por técnica de surface rendering.

2.2. Reconstruir imágenes por técnica de volume rendering.

**RF 3 PROCESAR IMAGEN**

3.1. Aplicar paletas de colores a las imágenes médicas a partir de LUT[13] predefinidos.

3.2. Filtrar imágenes RGB a niveles de grises.

3.3. Filtrar imágenes por filtros de suavizados.

**RF 4 TRANSFORMAR ESPACIALMENTE**

4.1. Rotar imágenes bidimensionales a ángulos fijos.

4.2. Rotar imágenes tridimensionales en cualquier dirección espacial.

4.3. Realizar zoom sobre imágenes tridimensionales.

**RF 5 SEGMENTAR IMAGEN**

5.1. Segmentar imágenes bidimensionales a partir de técnicas de crecimiento de regiones (region growing).

### 2.3.2 Requisitos No Funcionales

**APARIENCIA O INTERFAZ EXTERNA**

El componente de software tendrá una interfaz amigable y similar a las aplicaciones del sistema operativo Windows para brindar facilidades de uso a los especialistas.

[13] **LUT:** LookUp Table, Paleta de colores. Conjunto de colores incluidos en una imagen.

**SOFTWARE**

- Se debe disponer para el uso del componente, del Sistema Operativo Windows XP o superior.
- Se debe instalar la librería MyDicom.NET.
- Debe estar instalado el Net Framework v2.0.

**HARDWARE**

Para el funcionamiento del componente se requiere de una máquina con los siguientes requisitos:

- Pentium IV 3.0 Ghz o superior.
- Memoria principal DDR 1GB.
- Disco rígido: 20GB.
- Memoria de video con más de 128MB.

**RESTRICCIONES EN EL DISEÑO Y LA IMPLEMENTACIÓN**

- El análisis y diseño de la aplicación será basado en la metodología RUP con el uso del lenguaje de modelado UML.
- Se usará como herramienta CASE, Enterprise Architect para el modelado de los artefactos que se generan en cada uno de los flujos de trabajo.
- Se usará como lenguaje de programación C#.
- Se utilizará como IDE de desarrollo Visual Studio 2008.

## 2.4 DEFINICION DE LOS CASOS DE USO DEL SISTEMA

### 2.4.1 Actor del sistema

| Actor | Descripción |
|---|---|
| Especialista | Es la persona que interactúa con el sistema. Procesa, transforma espacialmente, segmenta regiones de interés y reconstruye tridimensionalmente imágenes, con el fin de lograr un mejor entendimiento de la anatomía cerebral. |

### 2.4.2 Diagrama de Casos de Uso del Sistema


**Fig. 5 Casos de Uso del Sistema.**

### 2.4.3 Casos de Uso por ciclo

A continuación se muestran los casos de uso a implementar durante el ciclo de desarrollo.

| Cód. | Nombre caso de uso | Paquete | Justificación |
|---|---|---|---|
| CU 1 | Visualizar Imagen. | Casos de Uso del Sistema (Crítico) | Estos casos de uso describen los procesos a automatizar para el especialista. |
| CU 2 | Segmentar Imagen. | Casos de Uso del Sistema (Crítico) | |
| CU 3 | Reconstruir Imagen Tridimensional. | Casos de Uso del Sistema (Crítico) | |
| CU 4 | Aplicar filtros sobre la Imagen. | Casos de Uso del Sistema (Crítico) | |
| CU 5 | Aplicar zoom sobre Imagen Tridimensional. | Casos de Uso del Sistema (Secundario) | |
| CU 6 | Rotar Imagen | Casos de Uso del Sistema (Secundario) | |

### 2.4.4 Casos de Uso expandidos

Los casos de uso expandidos son muy útiles para alcanzar un conocimiento más profundo de los procesos y de los requerimientos. Constituyen un documento narrativo que describe la secuencia de eventos de un actor que utiliza el sistema para completar un proceso.

**CU Visualizar Imagen**

| | |
|---|---|
| **Caso de Uso:** | Visualizar Imagen |
| **Actores:** | Especialista(Inicia) |
| **Resumen:** | El especialista selecciona la(s) imagen(es) que desea visualizar, mostrándose también la información adjunta a cada imagen. El sistema posibilita además visualizar imágenes previamente segmentadas y |

| | |
|---|---|
| | reconstruidas tridimensionalmente. |
| **Precondiciones:** | |
| **Referencias** | RF 1.1, RF 1.2, RF 1.3, RF 1.4, RF 1.5, RF 1.6 |
| **Prioridad** | Crítico |

| **Flujo Normal de Eventos** | |
|---|---|
| **Sección “”** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista escoge la(s) imagen(es) que desea visualizar, pueden estar almacenadas localmente en la máquina o en un dispositivo de almacenamiento. | 2. El sistema visualiza la(s) imagen(es), el nombre del paciente y la descripción del estudio. |
| | 3. El sistema ofrece la facilidad de navegar por las series de imágenes. |
| ***Prototipo de Interfaz*** | |
| Anexo 8 | |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista desea visualizar una imagen previamente segmentada. | 2. El sistema visualiza la imagen previamente segmentada. |
| ***Prototipo de Interfaz*** | |
| Anexo 9 | |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista desea visualizar una imagen previamente reconstruida tridimensionalmente. | 2. El sistema visualiza la imagen previamente reconstruida tridimensionalmente. |
| ***Prototipo de Interfaz*** | |
| Anexo 10 | |

| Flujos Alternos | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista desea visualizar una imagen previamente procesada o transformada. | 2. El sistema visualiza la imagen. |
| ***Prototipo de Interfaz*** | |
| Anexo 8 | |
| **Poscondiciones** | Una(s) imagen(es) tiene(n) que ser visualizada(s). |

**CU Segmentar Imagen**

| | |
|---|---|
| **Caso de Uso:** | Segmentar Imagen |
| **Actores:** | Especialista(Inicia) |
| **Resumen:** | El especialista selecciona la(s) imagen(es) que desea segmentar y entra los datos necesarios para la segmentación. |
| **Precondiciones:** | Debe de haberse seleccionado una(s) imagen(es) previamente. |
| **Referencias** | RF 5.1, CU Visualizar Imagen |
| **Prioridad** | Crítico |
| **Flujo Normal de Eventos** | |
| **Sección “”** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista selecciona una(s) imagen(es) para segmentar. | 2. El sistema muestra los diferentes algoritmos de segmentación que posee. |
| 3. El especialista selecciona un algoritmo. | 4. El sistema solicita datos de entrada para ejecutar el algoritmo seleccionado. |
| 5. El especialista entra los datos. | 6. El sistema ejecuta el algoritmo.<br>7. El sistema visualiza la imagen obtenida del proceso de segmentación (CUS Visualizar Imagen). |
| ***Prototipo de Interfaz*** | |

| Anexo 9 | |
|---|---|
| **Poscondiciones** | Se obtiene una imagen segmentada. |

**CU Reconstruir Imagen Tridimensional**

| **Caso de Uso:** | Reconstruir Imagen Tridimensional |
|---|---|
| **Actores:** | Especialista (Inicia) |
| **Resumen:** | El especialista selecciona las imágenes para realizar la reconstrucción tridimensional. |
| **Precondiciones:** | |
| **Referencias** | RF 2.1, RF 2.2, CU Visualizar Imagen |
| **Prioridad** | Crítico |

| **Flujo Normal de Eventos** | |
|---|---|
| **Sección “”** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista selecciona una(s) imagen(es) para reconstruir tridimensionalmente. | 2. El sistema realiza la reconstrucción tridimensional utilizando el modelo de renderizado Volume Rendering.<br>3. El sistema visualiza la imagen obtenida del proceso de reconstrucción tridimensional (CUS Visualizar Imagen). |
| ***Prototipo de Interfaz*** | |
| Anexo 10 | |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista desea realizar la reconstrucción de la(s) imagen(es) previamente segmentada(s). | 2. El sistema realiza la reconstrucción tridimensional utilizando el modelo de renderizado Surface Rendering.<br>3. El sistema visualiza la imagen obtenida del proceso de reconstrucción tridimensional (CUS Visualizar Imagen). |

| *Prototipo de Interfaz* | |
|---|---|
| Anexo 11 | |
| **Poscondiciones** | Se obtiene una imagen tridimensional. |

**CU Aplicar filtro sobre Imagen**

| **Caso de Uso:** | Aplicar filtro sobre Imagen |
|---|---|
| **Actores:** | Especialista(Inicia) |
| **Resumen:** | El especialista selecciona la imagen que desea transformar, selecciona el tipo de filtro, el sistema muestra la imagen que queda como resultado de aplicarle el filtro. |
| **Precondiciones:** | |
| **Referencias** | RF 3.1, RF 3.2, RF 3.3, RF 3.4, CU Visualizar Imagen |
| **Prioridad** | Secundario |
| **Flujo Normal de Eventos** | |
| **Sección “”** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista escoge la imagen a la que se quiere realizar la transformación. | 2. El sistema muestra la imagen seleccionada. Brinda la posibilidad de seleccionar el tipo de filtro a aplicar mediante un menú con los siguientes filtros:<br>– Suavizado<br>– Paletas<br>– Invertir<br>– Escala de gris |
| 3. El especialista selecciona el tipo de filtro. | 4. El sistema muestra la imagen transformada por el filtro aplicado. (CUS Visualizar Imagen). |
| *Prototipo de Interfaz* | |
| Anexo 8 | |

| Flujos Alternos | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| | |
| ***Prototipo de Interfaz*** | |
| **Poscondiciones** | Una(s) imagen(es) tiene(n) que ser visualizada(s). |

**CU Aplicar zoom sobre Imagen Tridimensional**

| **Caso de Uso:** | Aplicar zoom sobre Imagen Tridimensional |
|---|---|
| **Actores:** | Especialista(Inicia) |
| **Resumen:** | El especialista puede ampliar toda la imagen tridimensional o disminuirla a conveniencia. |
| **Precondiciones:** | |
| **Referencias** | RF 4.3 |
| **Prioridad** | |
| **Flujo Normal de Eventos** | |
| **Sección “”** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. El caso de uso comienza cuando el especialista escoge una imagen previamente reconstruida tridimensionalmente. | 2. El sistema ofrece la facilidad de realizar zoom a la imagen tridimensional utilizando el Scroll del mouse. Al realizar el movimientos hacia adelante (Zoom In) y hacia atrás (Zoom Out). |
| ***Prototipo de Interfaz*** | |
| Anexo 10 | |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | |
| ***Prototipo de Interfaz*** | |
| **Poscondiciones** | Objeto tridimensional con el zoom aplicado. |

**CU Rotar Imagen**

| | |
|---|---|
| **Caso de Uso:** | Rotar Imagen |
| **Actores:** | Especialista(Inicia) |
| **Resumen:** | El especialista selecciona la imagen que desea transformar, selecciona el ángulo, el sistema muestra la imagen resultante. |
| **Precondiciones:** | |
| **Referencias** | RF 4.1, RF 4.2, CU Visualizar Imagen |
| **Prioridad** | Secundario |

| **Flujo Normal de Eventos** | |
|---|---|
| **Sección “”** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Este caso de uso comienza cuando el especialista escoge la imagen a la que se quiere realizar la transformación. | 2. El sistema muestra la imagen seleccionada. Da la posibilidad de seleccionar el ángulo de rotación. |
| 3. Selecciona un ángulo (90, 270). | 4. El sistema muestra la imagen después de ser rotada (CUS Visualizar Imagen). |
| ***Prototipo de Interfaz*** | |
| Anexo 8 | |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| 1. Este caso de uso comienza cuando el especialista escoge una imagen previamente reconstruida tridimensionalmente. | 2. El sistema ofrece la posibilidad de rotar la imagen tridimensional realizando movimientos horizontales y verticales sobre la imagen con el clic derecho presionado. |
| ***Prototipo de Interfaz*** | |
| Anexo 10 | |

| | |
|---|---|
| **Poscondiciones** | Se visualiza una imagen rotada. |

**CONCLUSIONES**

En el presente capítulo fueron elaborados y descritos brevemente algunos de los artefactos propuestos por RUP para el desarrollo de software, tales como: el modelo de dominio, la definición de los requisitos funcionales y no funcionales, los actores, casos de uso, diagrama de casos de uso del sistema y la descripción textual de los mismos.

# CAPÍTULO 3
# DISEÑO DEL COMPONENTE

Este capítulo describe como implementar el componente, a través del diseño, enfocado a cómo el sistema cumple sus objetivos teniendo en cuenta los requisitos funcionales y no funcionales. Se realizan los diagramas de clases y los diagramas de interacción según los casos de uso definidos y se explica además la arquitectura utilizada, así como el empleo de los patrones de diseño.

## 3.1 ESTILO ARQUITECTÓNICO UTILIZADO

Para el desarrollo del componente se hace uso del patrón arquitectónico Modelo Vista Controlador. Este patrón de diseño, se basa en la separación en tres capas del diseño de las aplicaciones: el modelo de datos, la presentación de los mismos y las acciones de los usuarios. El componente hace uso intensivo de interfaces visuales que requieren de una actualización constante de las propiedades que se modifican. Realiza un diseño que desacopla la vista del modelo, con la finalidad de mejorar la reusabilidad, de esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos. A continuación se muestra este patrón de arquitectura. (Fig. 6)

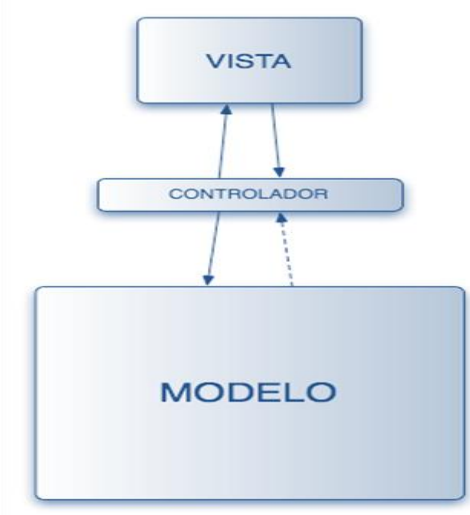



**Fig. 6 Diagrama Modelo Vista Controlador.**

## 3.2 PATRONES UTILIZADOS

En el diseño del componente se utilizaron los siguientes patrones para dar solución a las diferentes problemáticas.

### 3.2.1 Patrón SINGLENTON

**Problema:** Se necesita mantener de manera única la instancia del control de visualización tridimensional de imágenes.

**Solución:** Para esto se aplica el patrón de Creación, a nivel de objetos SINGLETON.

**Aplicación:** Este patrón se utiliza cuando se realiza una construcción tridimensional para garantizar que sólo exista una instancia de este objeto.

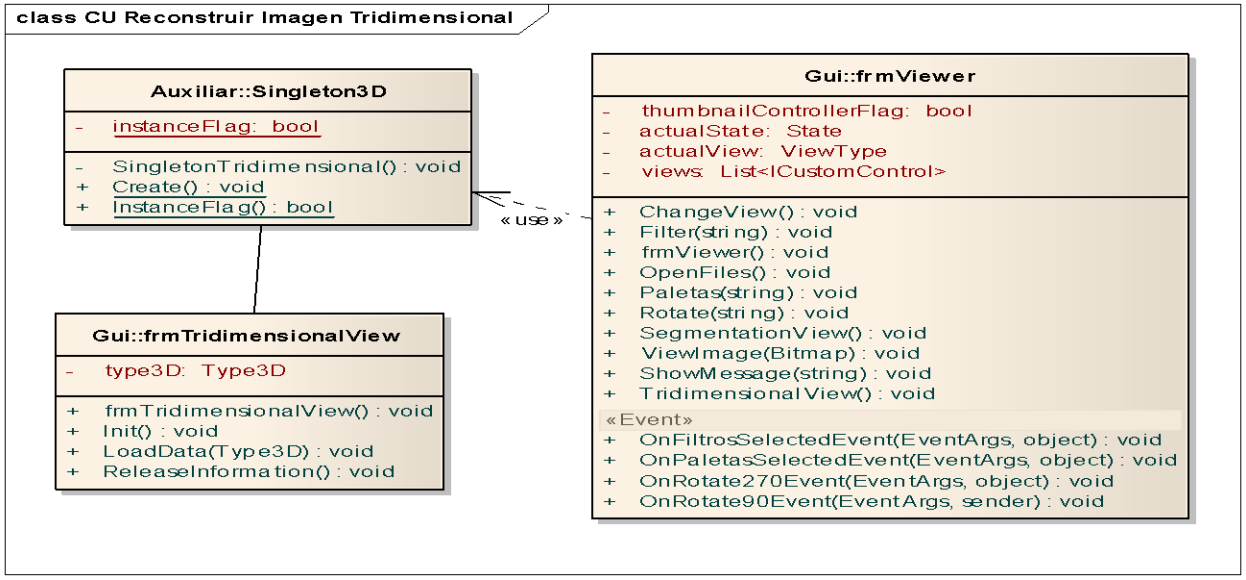



**Fig. 7 Patrón Singleton.**

### 3.2.2 Patrón Experto

**Problema:** Las responsabilidades deben ser acorde a la información con que cuenta cada clase.

**Solución:** Para esto se aplica el patrón GRASP, Experto.

**Aplicación:** La clase Segmentation es la experta en la información, ya que es la que contiene los algoritmos de segmentación utilizados.

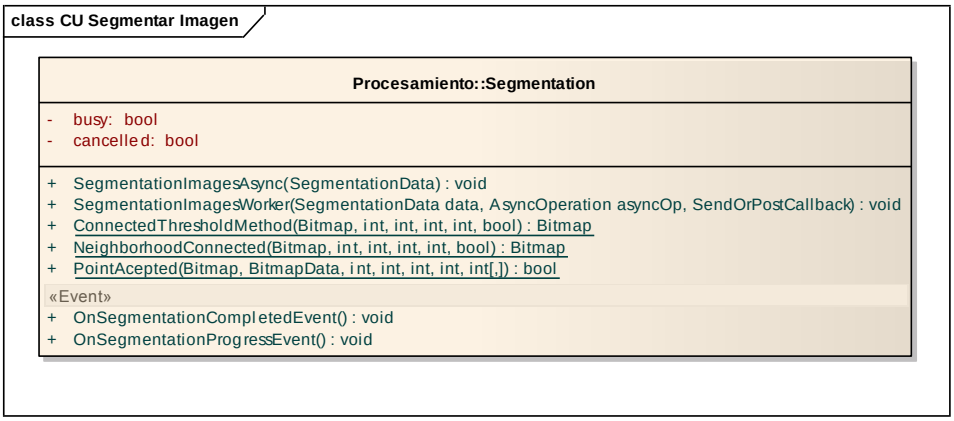



**Fig. 8 Patrón Experto**

### 3.2.3 Patrón Creador

**Problema:** Crea instancias solamente del objeto que lo contiene y lo registra.

**Solución:** Para dar solución a esto se aplica el patrón GRASP, Creador.

**Aplicación:** La clase 2DOperation es la encargada de crear la clase SegmentationData, ya que es la que posee toda la información necesaria.

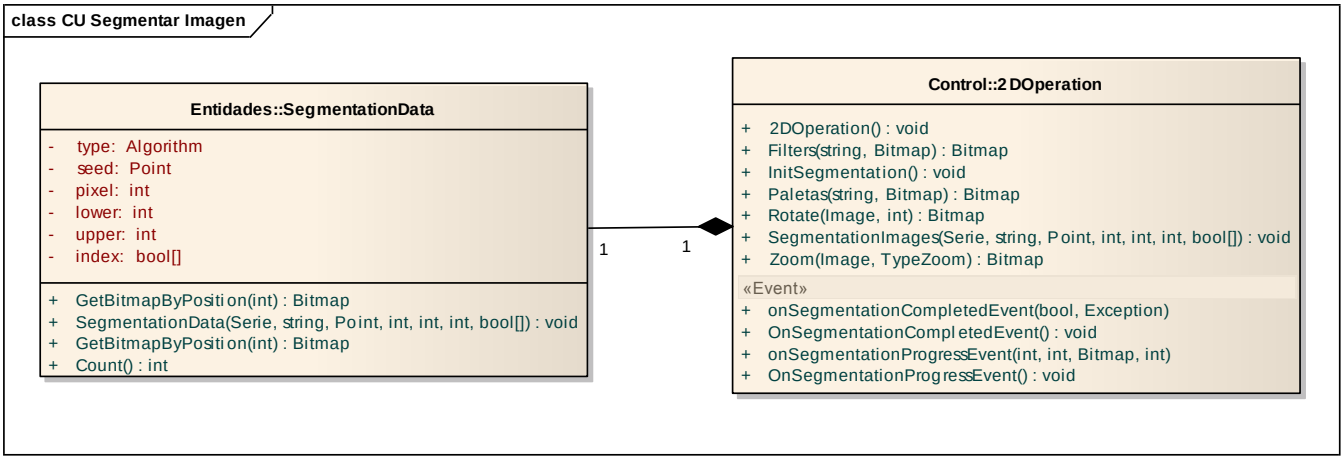



**Fig. 9 Patrón Creador.**

### 3.2.4 Patrón Polimorfismo

**Problema:** Cómo manejar diferentes comportamientos en dependencia del tipo de objeto.

**Solución:** Para dar solución a esto se aplica el patrón GRASP, Polimorfismo.

**Aplicación:** La interfaz ICustomControl define una serie de métodos que después son redefinidos por las clases ucSegmentationView y uc3DView.

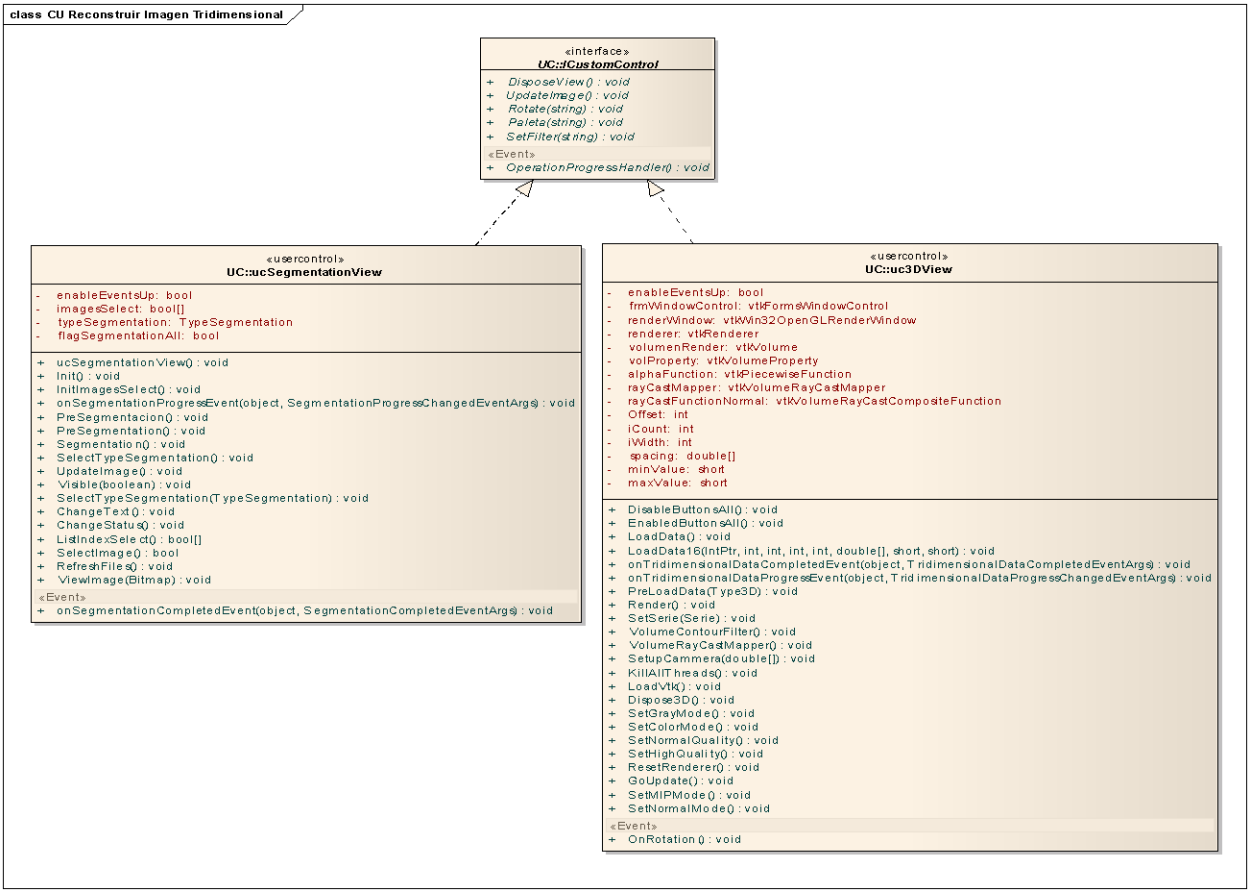



**Fig. 10 Patrón Polimorfismo.**

### 3.3 DIAGRAMAS DE CLASES

Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces de la aplicación. Además visualiza las relaciones entre las clases que se involucran en el sistema.

El diseño de la aplicación por casos de uso se puede apreciar en las figuras: Fig. 11 hasta la Fig. 16.

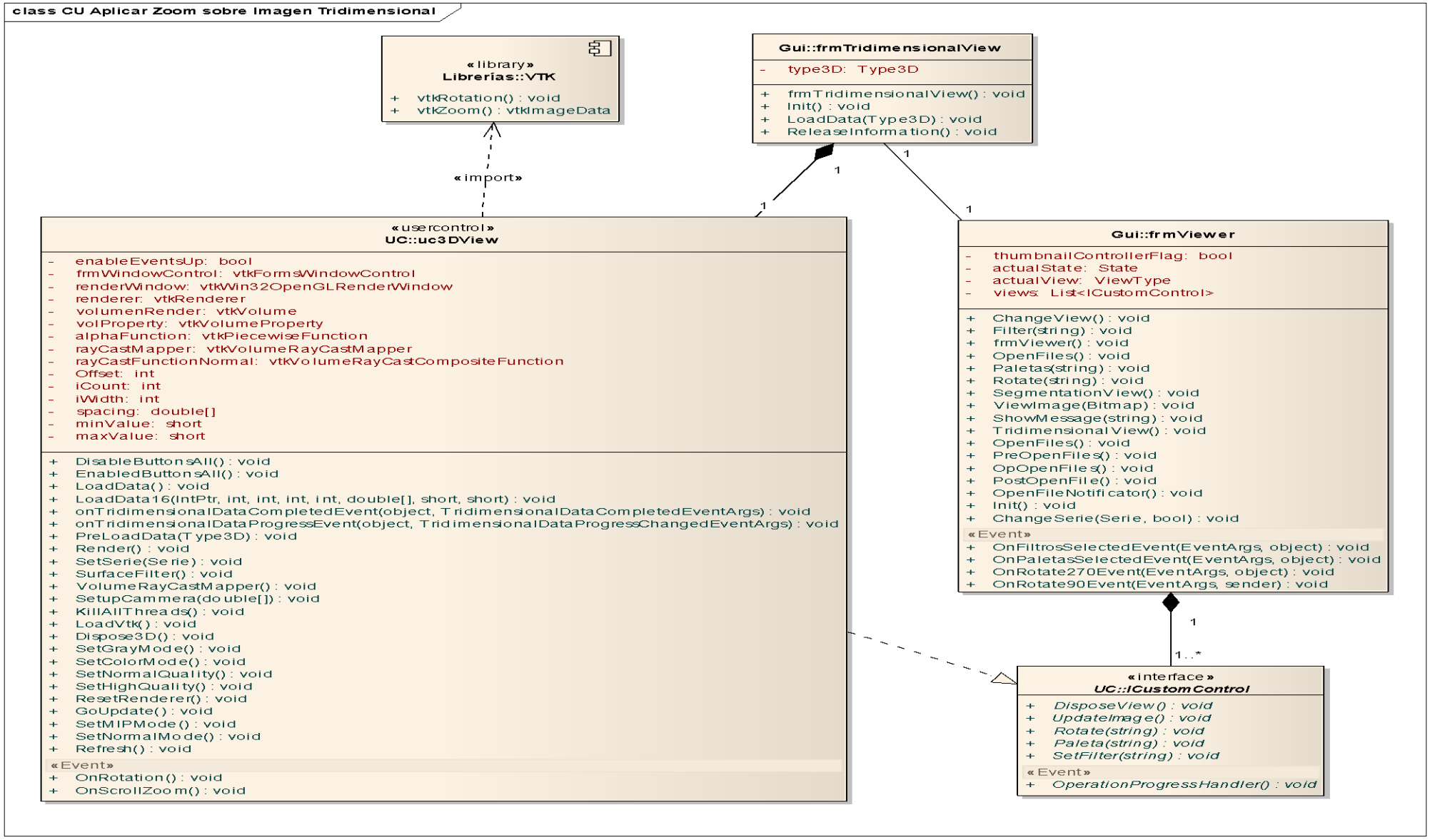



**Fig. 11 CU Aplicar Zoom sobre la Imagen Tridimensional**

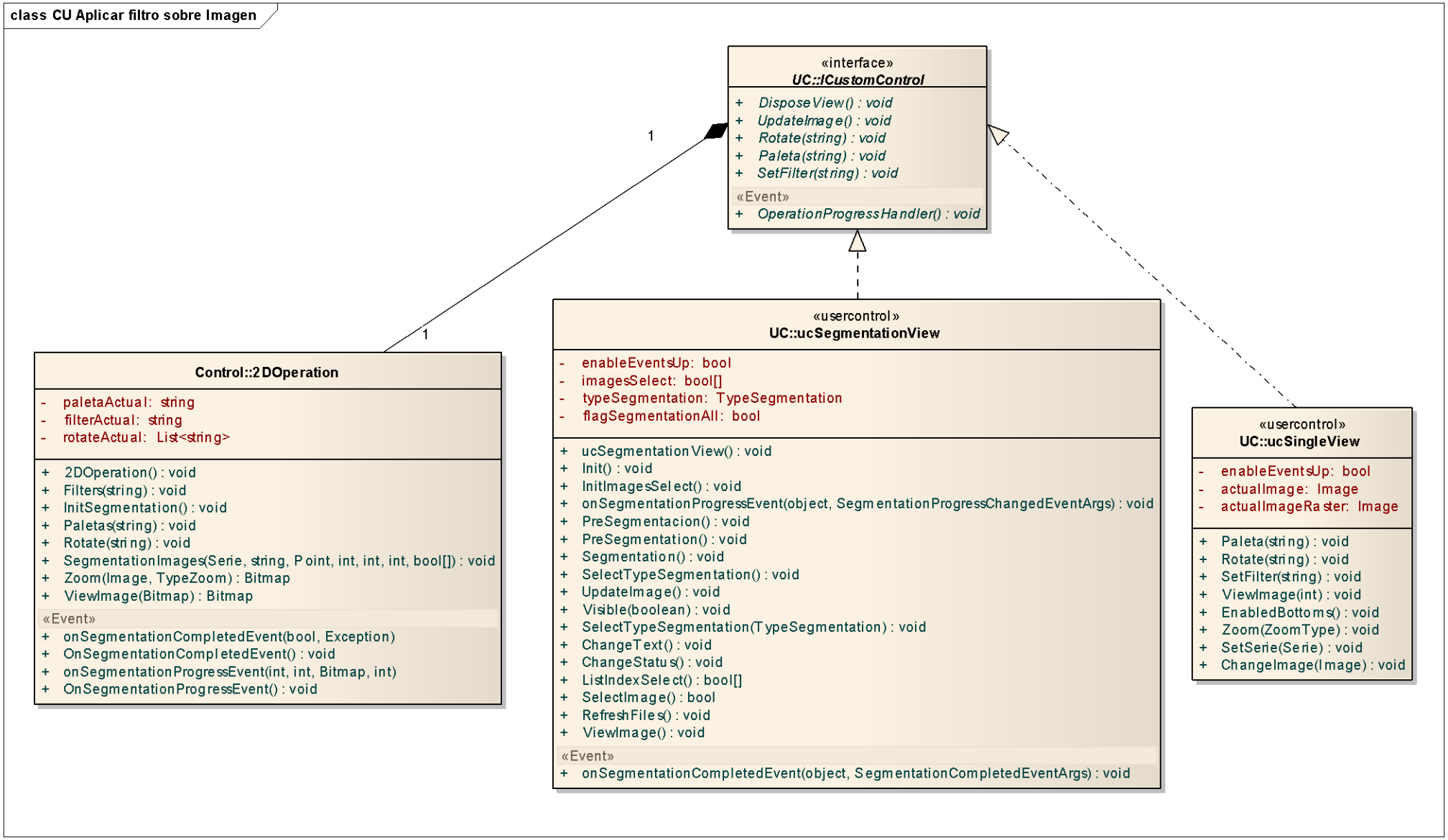



**Fig. 12 CU Aplicar filtro sobre la Imagen**

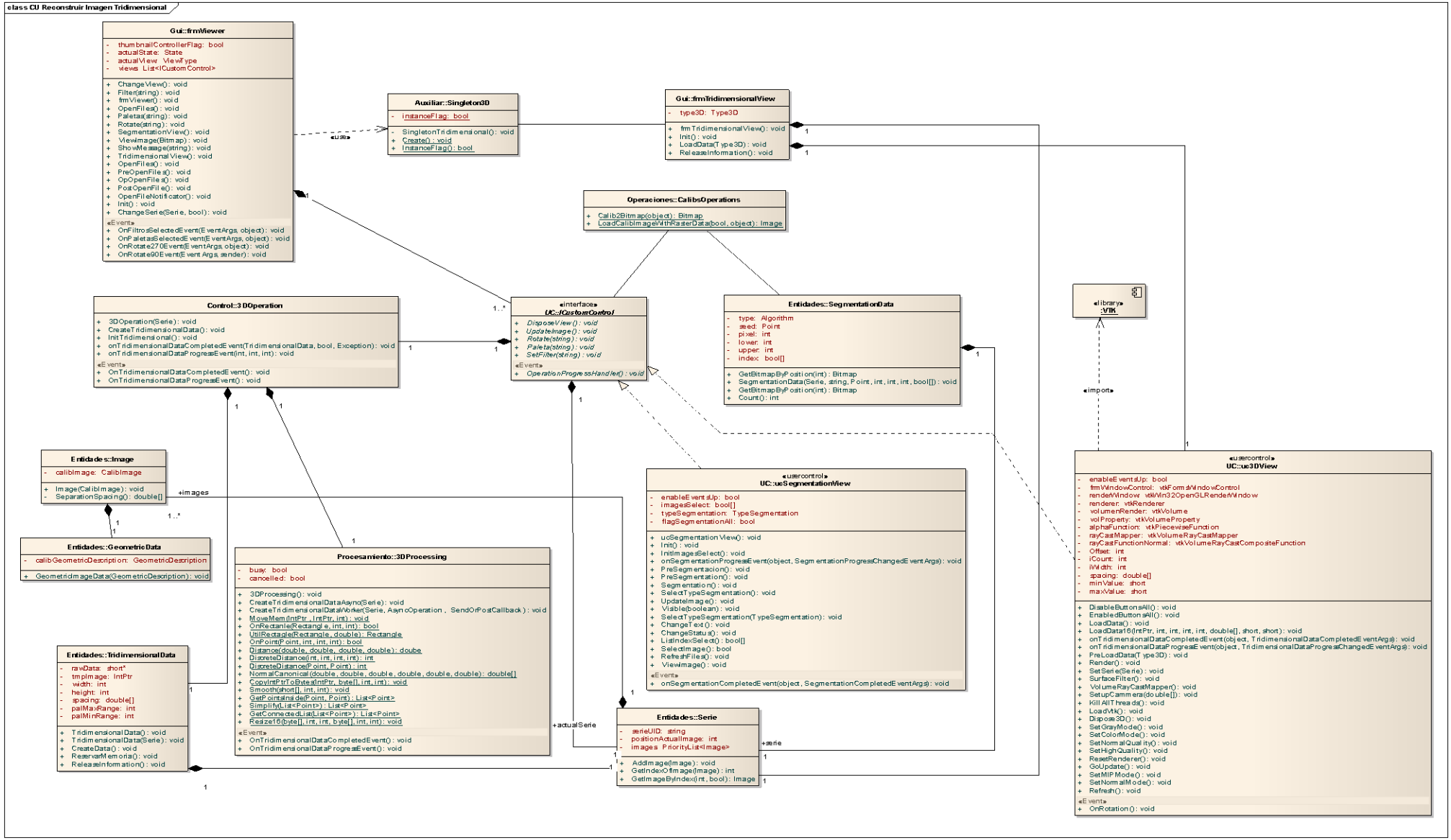



**Fig. 13 CU Reconstruir Imagen Tridimensionalmente**

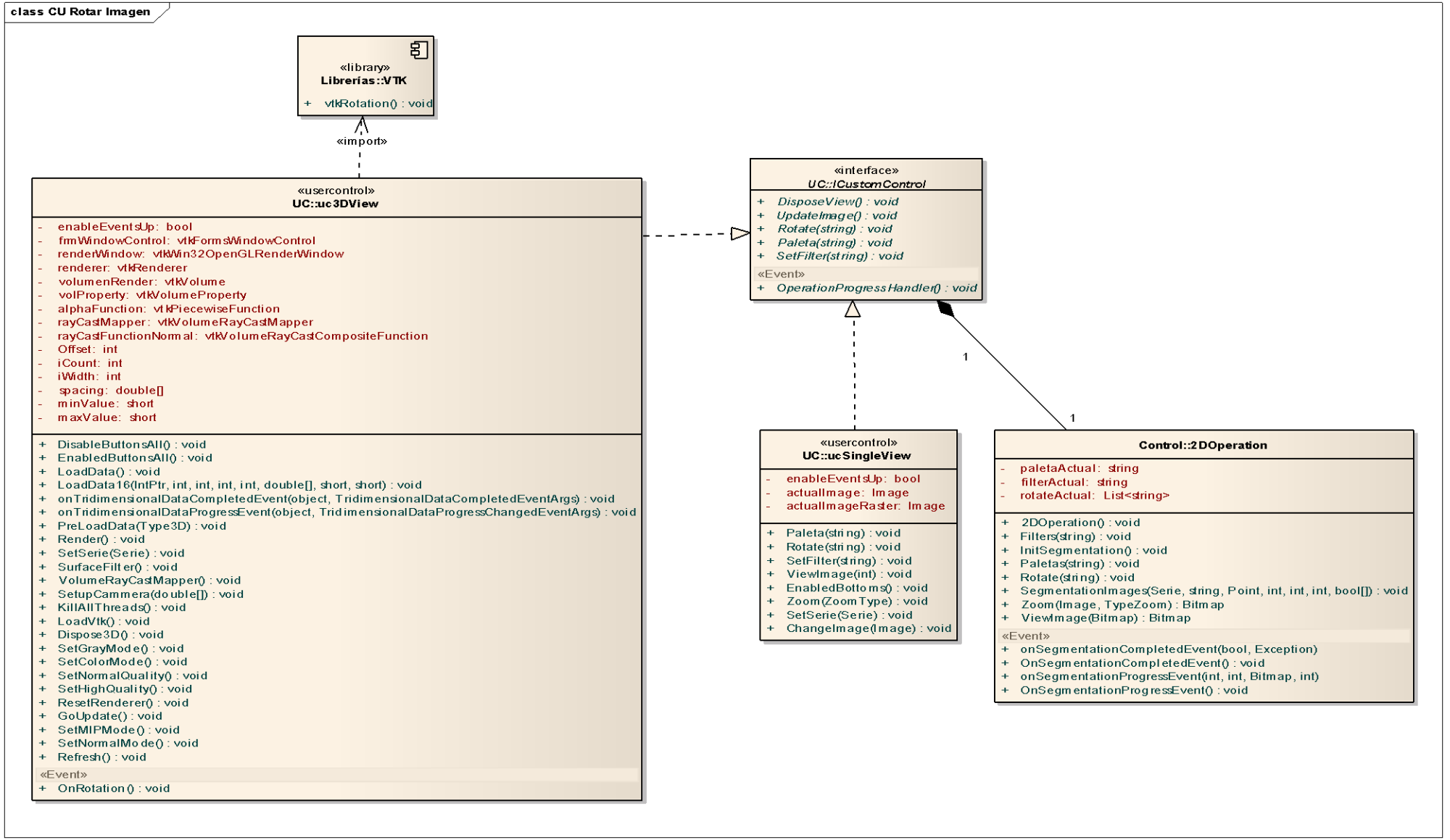



**Fig. 14 CU Rotar Imagen**

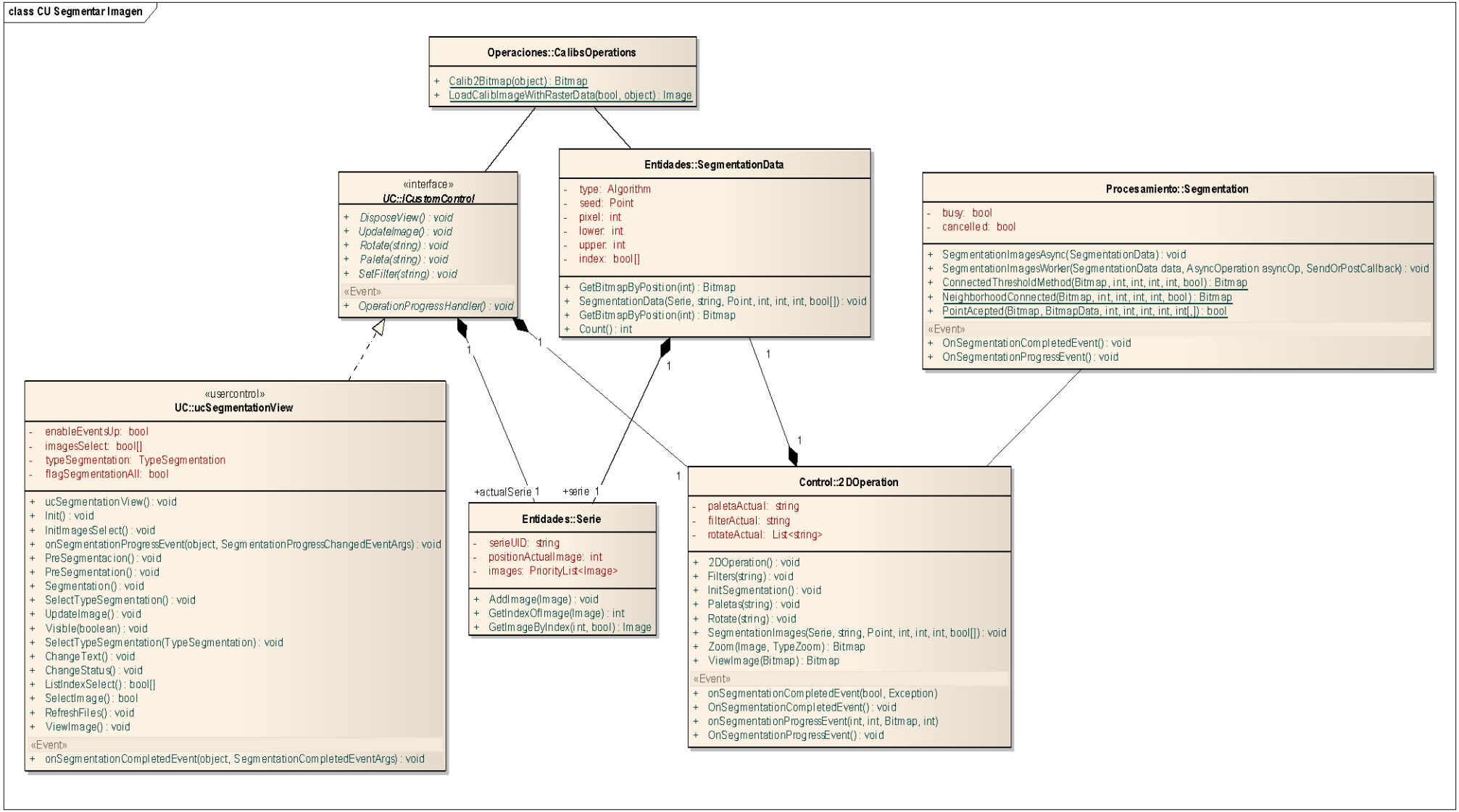



**Fig. 15 CU Segmentar Imagen**

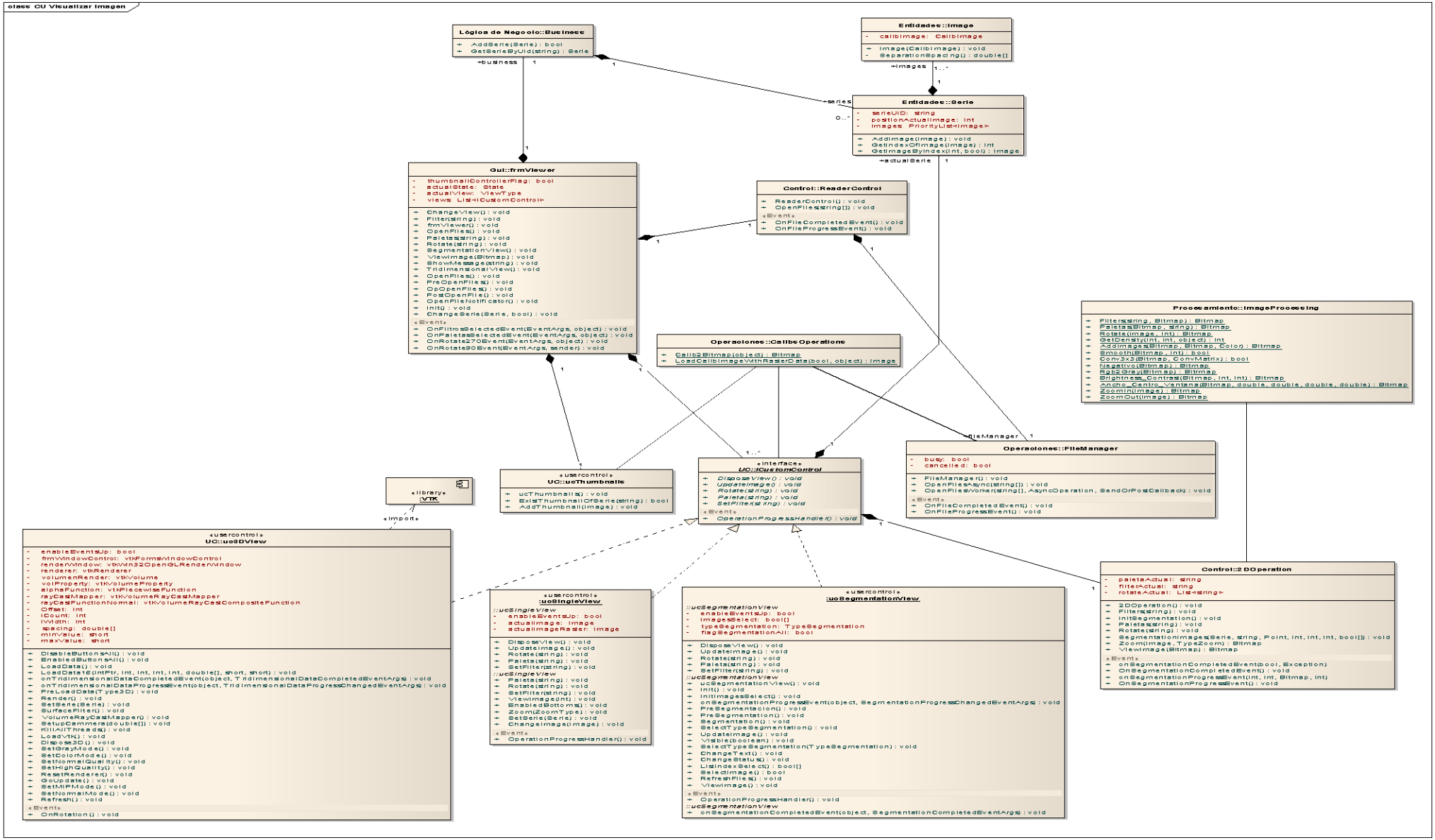



**Fig. 16 CU Visualizar Imagen**

### 3.4 DIAGRAMAS DE SECUENCIAS

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal; muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. [36]
Los diagramas de secuencia guían al programador durante la fase de implementación. A continuación se exponen los mismos:

La Fig. 17 muestra la interacción entre las clases que componen el caso de uso Aplicar Zoom sobre Imagen Tridimensional. En este diagrama se evidencia cómo, después de creado el objeto tridimensional, se le aplica un zoom (ZoomIn o ZoomOut) en dependencia de la selección realizada, finalizando una vez visualizada la imagen.

La Fig. 18 muestra la interacción entre las clases que componen el caso de uso Aplicar Filtro sobre Imagen. En este diagrama se aplica la paleta o el filtro seleccionado una vez cargada la imagen, finalizando con la visualización de la misma.

La Fig. 19 muestra la interacción entre las clases que componen el caso de uso Reconstruir imagen Tridimensional. El mismo comienza creando una instancia única del objeto tridimensional a través de la clase Singleton; si la imagen que va a reconstruir es previamente segmentada utiliza la técnica de Surface Rendering, de lo contrario se utiliza la técnica Volume Rendering, finalizando una vez visualizada la imagen tridimensional.

La Fig. 20 muestra la interacción entre las clases que componen el caso de uso Rotar Imagen. En este diagrama se selecciona el ángulo con el cual se desea rotar la imagen bidimensional, es decir, si la imagen será rotada hacia la derecha o izquierda, y si la imagen es tridimensional puede ser rotada libremente, terminando con la visualización deseada de dicha imagen.

La Fig. 21 muestra la interacción entre las clases que componen el caso de uso Segmentar Imagen. En este diagrama se muestra la vista de segmentación, se selecciona el algoritmo por el cual se desea segmentar así como los datos necesarios para realizar el mismo. Además se brinda la opción de seleccionar si se desea segmentar solo la imagen actual, las imágenes marcadas o el estudio completo, finalizando con la visualización de la(s) imagen(es) segmentada(s).

La Fig. 22 muestra la interacción entre las clases que componen el caso de uso Visualizar Imagen. En este diagrama se muestra como es leído el fichero, éste puede estar almacenado en la máquina o en un dispositivo. La clase FileManager es la encargada de abrir los ficheros. Después se verifica si la imagen tiene alguna paleta, filtro o ángulo de rotación aplicada y se termina visualizando la imagen médica.

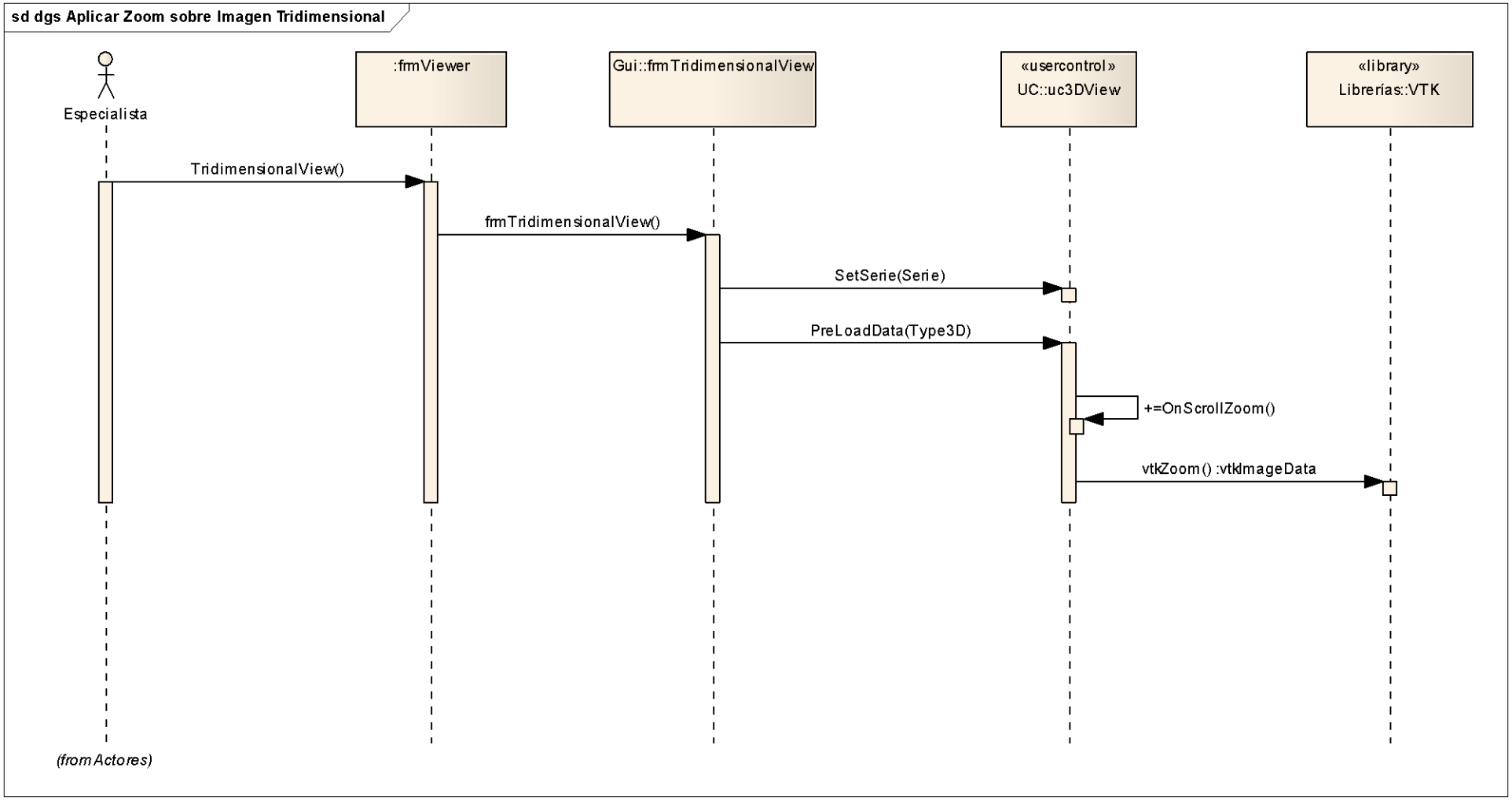



**Fig. 17 Diagrama de interacción (Aplicar Zoom sobre Imagen Tridimensional)**

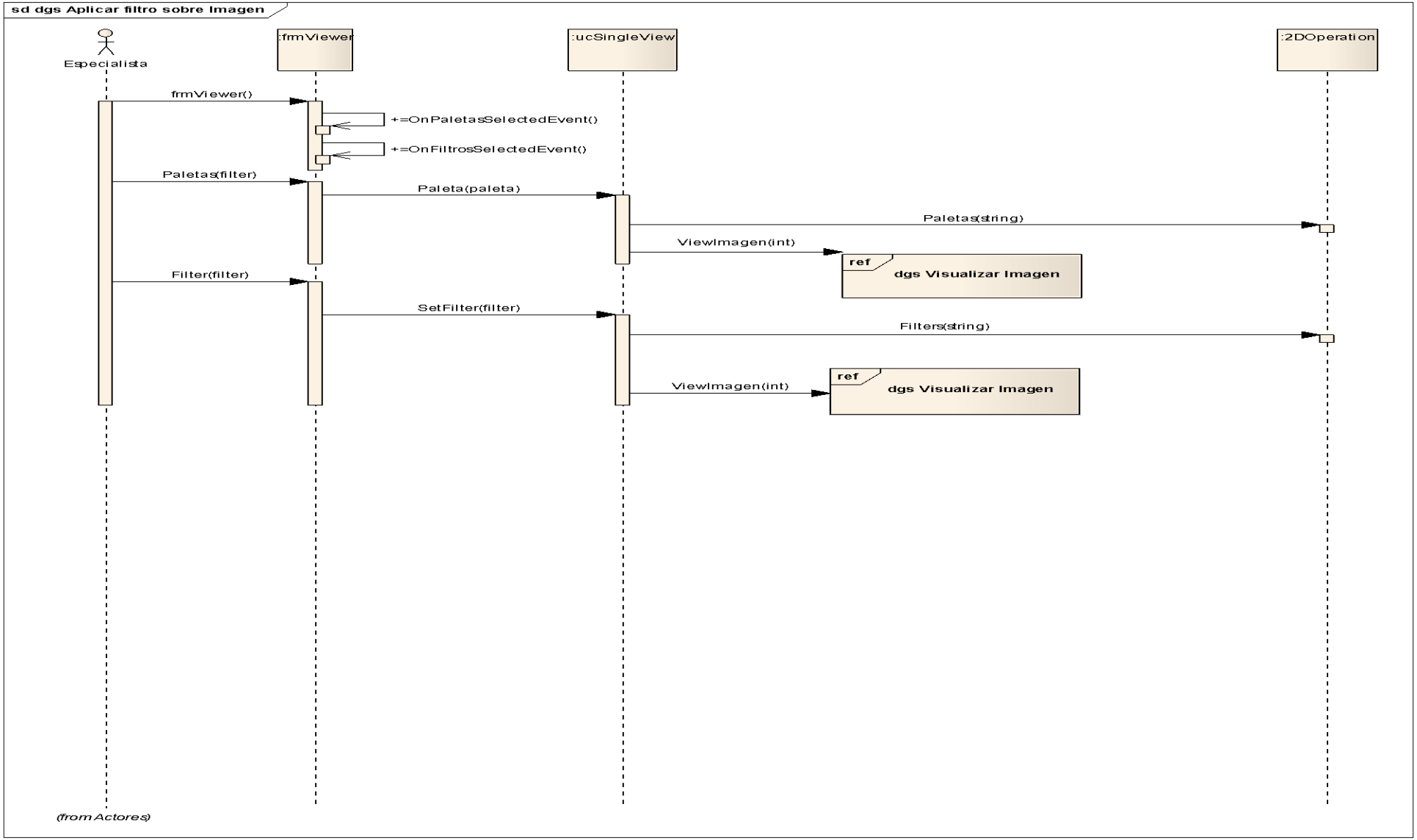



**Fig. 18 Diagrama de interacción (Aplicar filtro sobre Imagen)**

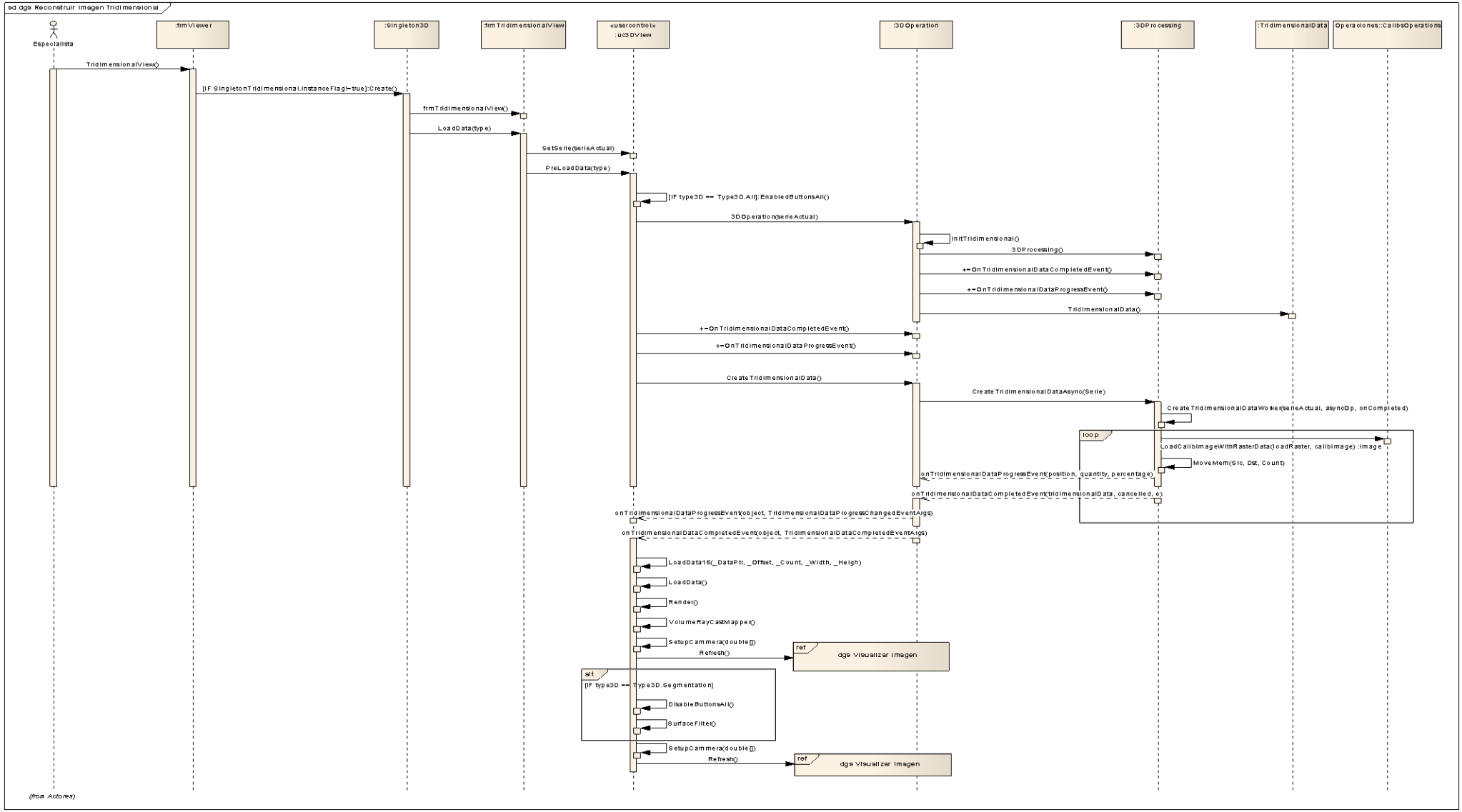



**Fig. 19 Diagrama de interacción (Reconstruir Imagen Tridimensional)**

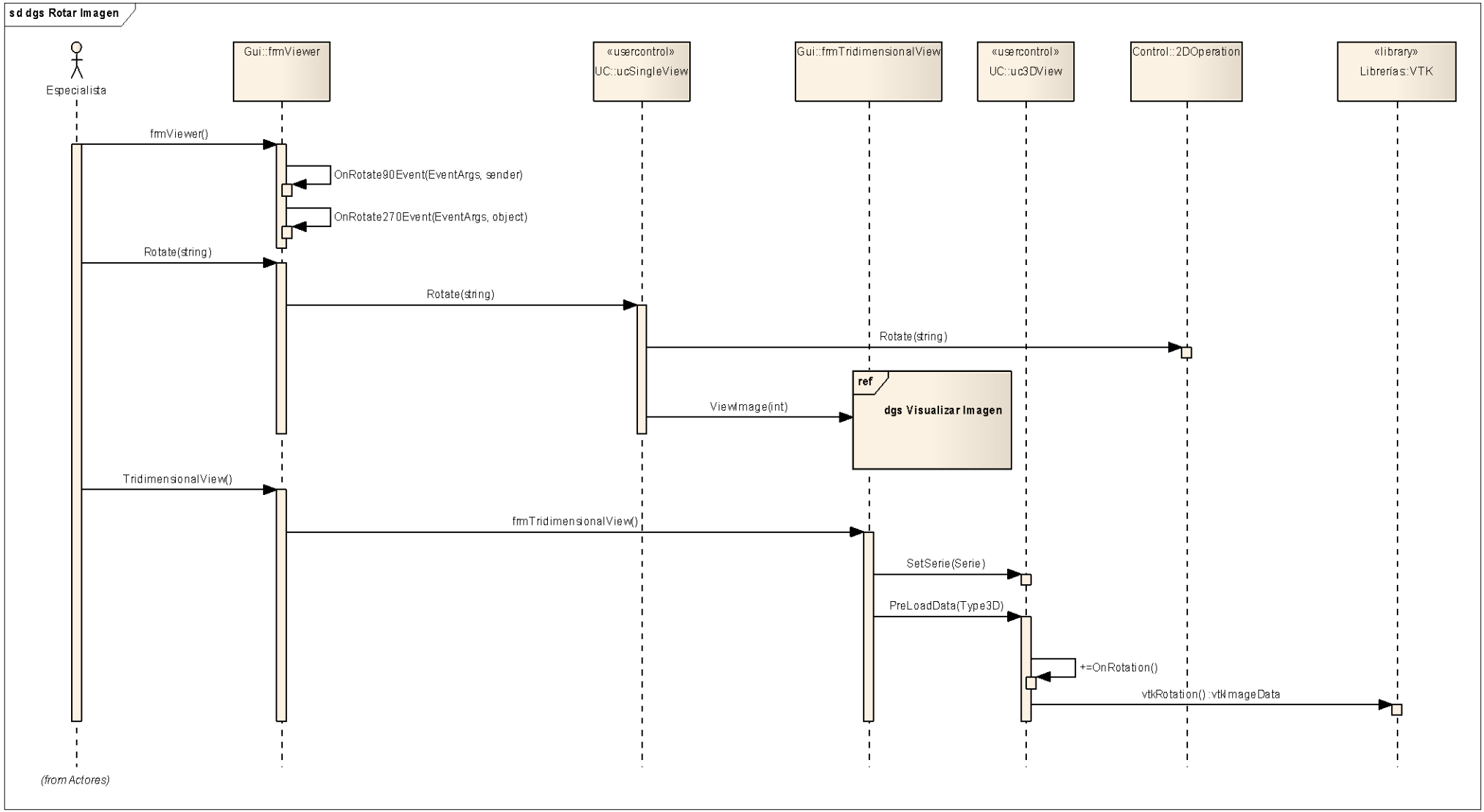



**Fig. 20 Diagrama de interacción (Rotar Imagen)**

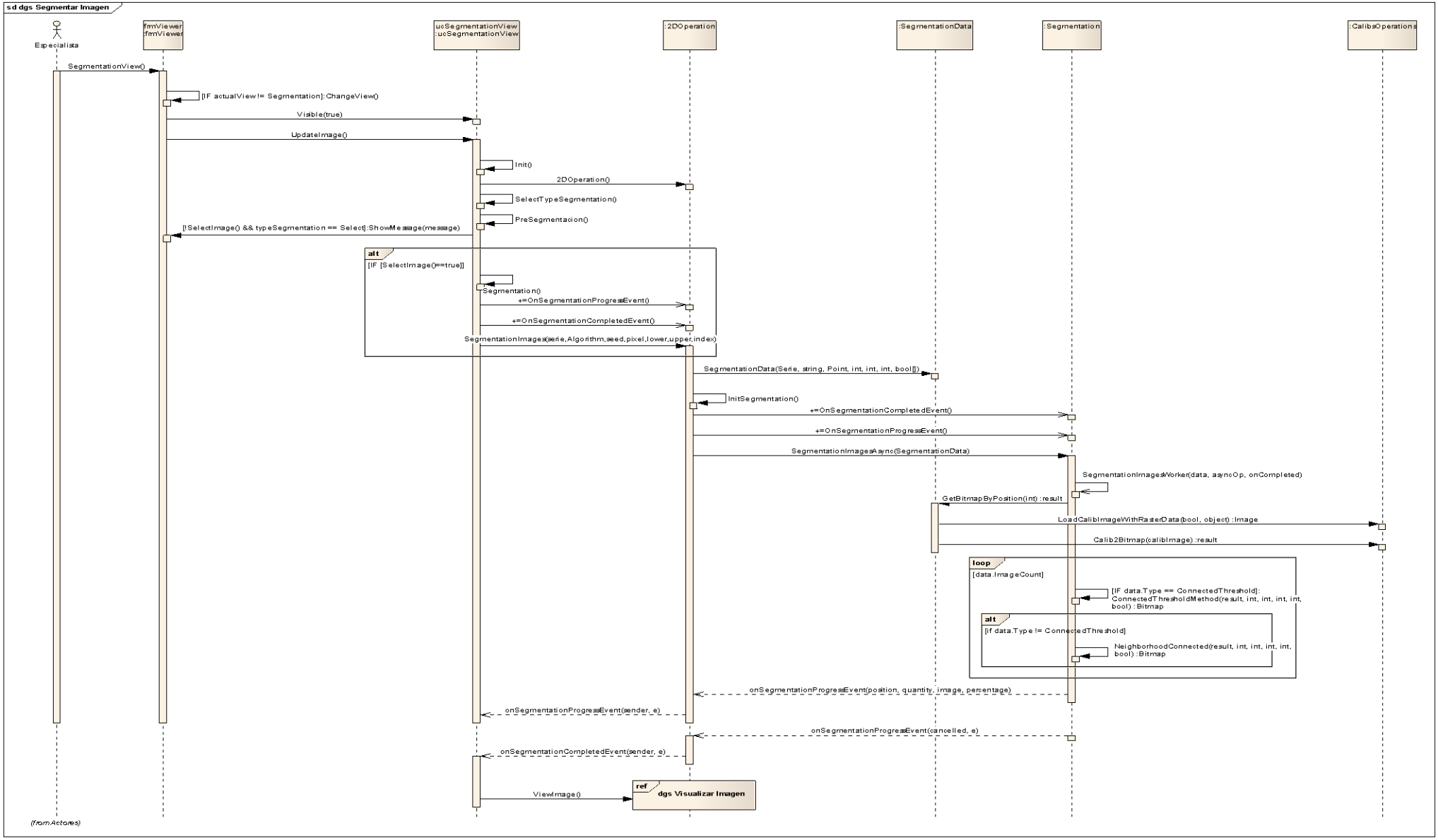



**Fig. 21 Diagrama de interacción (Segmentar Imagen)**

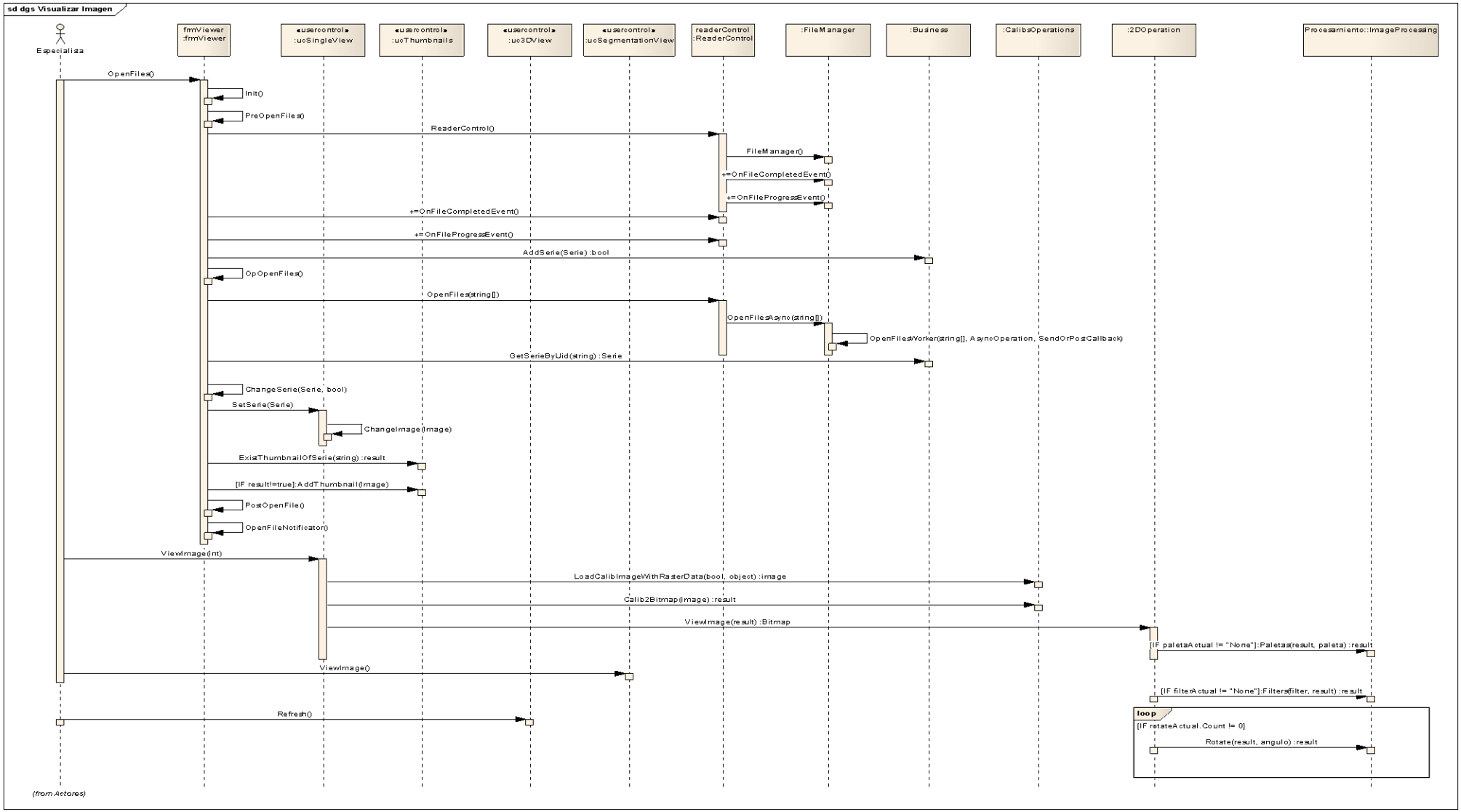



**Fig. 22 Diagrama de interacción (Visualizar Imagen)**

**CONCLUSIONES**

En este capítulo quedó establecido el estilo arquitectónico utilizado para el desarrollo del componente; además se abordó la utilización de los patrones de diseño. Las clases del diseño se representaron a través de los diagramas de clases del diseño, y la interacción de las mismas a partir de los diagramas de secuencias.

# CAPÍTULO 4
# IMPLEMENTACIÓN DEL COMPONENTE

En este capítulo se describe cómo fue implementado el componente. Se presenta el diagrama de componentes, que muestra la distribución física de los componentes para la implementación. Se describen los fragmentos relevantes del código.

## 4.1 DIAGRAMA DE COMPONENTES

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos software que entran en la fabricación de aplicaciones informáticas. [37]

El diagrama de componentes está compuesto por el fichero ComponenteVisualizacion3D.exe que es el ejecutable de la aplicación, los ficheros de paletas necesarios para aplicar paletas a las imágenes médicas y las librerías vtk y calibs que se utilizan para representar los objetos tridimensionales y para la lectura de ficheros respectivamente.

A continuación se muestra el diagrama de componentes de la aplicación desarrollada. Fig. 23.

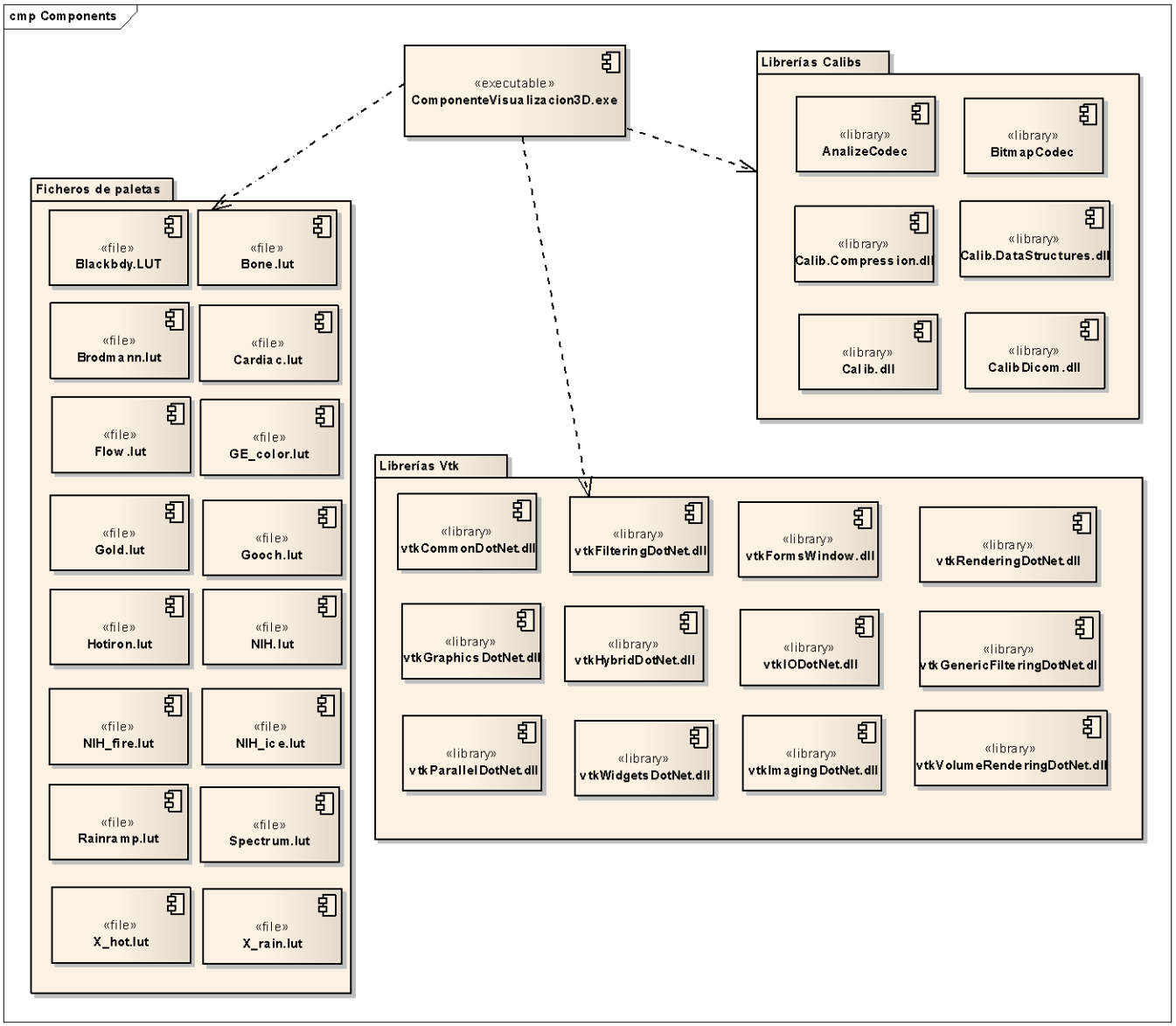



**Fig. 23 Diagrama de Componentes**

## 4.2 FRAGMENTOS DE CÓDIGO

En esta sección se exponen los fragmentos de código críticos en el desarrollo del componente, haciendo énfasis en las técnicas utilizadas para la reconstrucción tridimensional y en los algoritmos implementados para realizar la segmentación de imágenes médicas.

### 4.2.1 Reconstrucción Tridimensional

A continuación se muestra un fragmento de código, el cual genera la estructura (TridimensionalData)

que será utilizada posteriormente para la construcción del objeto tridimensional.

```
private void CreateTridimensionalDataWorker(Serie serie, AsyncOperation asyncOp,
SendOrPostCallback onCompleted)
        {
            TridimensionalData result = new TridimensionalData(serie);
            TridimensionalDataProgressChangedEventArgs e;
            int imageIndex = 0;
            Exception exc = null;

      try
            {
                result.Width = serie.GetImageByIndex(0, false).Width;
                result.Height = serie.GetImageByIndex(0, false).Height;
          1     result.Spacing = serie.GetImageByIndex(0, false).PixelSpacing;
                double OriginalSlope = serie.GetImageByIndex(0, false).RescaleSlope;

                if (OriginalSlope == 0.0) OriginalSlope = 1.0;

                double OriginaOffset = serie.GetImageByIndex(0,
false).RescaleIntercept;
                int BitsPerPixel = serie.GetImageByIndex(0, false).BitsStored;

                result.PalMaxRange = (int)(Pal16bit.Pal2Pow(BitsPerPixel) *
OriginalSlope + OriginaOffset);

                result.PalMinRange = (int)OriginaOffset;

          2     result.ReservarMemoria();

                for (int i = 0; i < serie.ImageQuantity; i++)
                {
                    unsafe
                    {
                        IntPtr Dest = (IntPtr)(&result.RawData[(serie.ImageQuantity
- (i + 1)) * result.Width * result.Height]);
                        MyImage imageWithRaster =
CalibsOperations.LoadCalibImageWithRasterData(serie.GetImageByIndex(i,
false).CalibObject, true);

                        _3DProcessing.MoveMem(imageWithRaster.Raster.PixelData.Data,
Dest, result.Width * result.Height * 2);

                        e = new TridimensionalDataProgressChangedEventArgs(i,
serie.ImageQuantity, ++imageIndex * 100 / serie.ImageQuantity);
                    }
                }
            }
            catch ( Exception exce)
            {
                exc = exce;
            }
```

```
            onCompleted(new TridimensionalDataCompletedEventArgs(result,cancelled,
exc));
        }
```

3

1. En este fragmento es donde se obtienen todos los datos de la serie actual, tales como el ancho, el alto, la separación entre píxeles y otros.
2. Luego se realizan las comprobaciones necesarias y se reserva la memoria para todas las imágenes de la serie actual.
3. Para finalizar se lanza el evento con la información de que fue completada la creación de la estructura (TridimensionalData) y con él si ocurrió alguna excepción, para dar tratamiento posteriormente.

#### 4.2.1.1 Volume Rendering

Con este fragmento de código se crea el objeto tridimensional. Primero se crea el volumen, con las características deseadas, tales como: color, el modo de opacidad, el tipo de interpolación, además de las imágenes a reconstruir (DataSource). Se selecciona además la calidad final del objeto y una vez obtenidos los límites del volumen reconstruido se ajusta la cámara para una buena visualización por parte del usuario.

```
public void VolumeRayCastMapper()
        {
            //Init volumen
            volumenRender.Dispose();
            volumenRender = new vtkVolume();
            InitProgressBar(100);

            //Observers
            volumenRender.AddObserver((uint)EventIds.ProgressEvent,
               new vtkDotNetCallback(ProgressEvent));
            rayCastMapper.AddObserver((uint)EventIds.ProgressEvent,
             new vtkDotNetCallback(ProgressEvent));

            //Create transfer mapping scalar value to opacity
            SetFullRange();

            // Create transfer mapping scalar value to color
            SetGrayMode();

            // The property describes how the data will look
            volProperty.SetColor(colorFunction);
            volProperty.SetScalarOpacity(alphaFunction);
```

```
            volProperty.SetInterpolationTypeToLinear();
            volProperty.DisableGradientOpacityOn();

            //Quality Render
            SetNormalQuality();

            //Ray cast function know how to render the data
            InitProgressBar(100);
            rayCastMapper.SetInput(DataSource);

            SetNormalMode();

            volumenRender.SetMapper(rayCastMapper);
            volumenRender.SetProperty(volProperty);

            renderer.AddVolume(volumenRender);

            try
            {
                double[] Bounds = volumenRender.GetBounds();
                SetupCammera(Bounds);
                Refresh();
            }
            catch { }
            PrgLoad.Visible = false;

        }
```

#### 4.2.1.2 Surface Rendering

En este fragmento de código se crea el objeto tridimensional. Primero se inicializan las variables a utilizar; se crean los eventos para observar el progreso de las acciones, posteriormente son cargadas las imágenes ya segmentadas y se ejecutan una serie de filtros para generar las superficies y es realizado el suavizado de las mismas. Una vez obtenidos los límites del volumen reconstruido, se ajusta la cámara para una buena visualización por parte del usuario.

```
public void SurfaceFilter()
      {

            //Init values
            volumenRender.Dispose();
            System.IO.DirectoryInfo info = new
            System.IO.DirectoryInfo(Application.StartupPath+"\\Cortes\\");
            int count = info.GetFiles().GetLength(0);
            InitProgressBar(count+1);

            //vtk init
            vtkContourFilter filtrocontorno = new vtkContourFilter();
            vtkPolyDataMapper maper = new vtkPolyDataMapper();
            vtkJPEGReader jpg2 = new vtkJPEGReader();
```

```
vtkSmoothPolyDataFilter smooth = new vtkSmoothPolyDataFilter();
vtkPolyDataNormals skinNormals = new vtkPolyDataNormals();
vtkActor actorvolumen = new vtkActor();

//Observers Progress
jpg2.AddObserver((uint)EventIds.ProgressEvent,
    new vtkDotNetCallback(ProgressEvent));
smooth.AddObserver((uint)EventIds.ProgressEvent,
               new vtkDotNetCallback(ProgressEvent));
skinNormals.AddObserver((uint)EventIds.ProgressEvent,
   new vtkDotNetCallback(ProgressEvent));
filtrocontorno.AddObserver((uint)EventIds.ProgressEvent,
    new vtkDotNetCallback(ProgressEvent));

//Create the reader for the data
jpg2.SetFileName(Application.StartupPath+"\\Cortes\\"+"1.jpeg");
jpg2.Update();
int[] exte2 = jpg2.GetDataExtent();
int dato2 = jpg2.GetDataByteOrder();
jpg2.SetFilePrefix(Application.StartupPath + "\\Cortes\\");
jpg2.SetFilePattern("%s%d.jpeg");
spacing = actualSerie.GetImageByIndex(0, false).PixelSpacing;
jpg2.SetDataSpacing(spacing[0], spacing[1], spacing[2]);
jpg2.SetDataByteOrder(dato2);
jpg2.SetDataScalarTypeToShort();
jpg2.SetDataExtent(0, exte2[1] - 1, 0, exte2[3] - 1, 1, count);

//filter that takes as input any dataset and generates on output
isosurfaces and/or isolines.
filtrocontorno.SetInput((vtkDataSet)jpg2.GetOutput());
filtrocontorno.SetValue(0, 45);
filtrocontorno.SetNumberOfContours(1);
filtrocontorno.ComputeNormalsOn();

//Filter smooth
InitProgressBar(100);
smooth.SetInput(filtrocontorno.GetOutput());
smooth.SetNumberOfIterations(1);
smooth.BoundarySmoothingOn();
smooth.SetFeatureAngle(120);
smooth.SetEdgeAngle(90);
smooth.SetRelaxationFactor(.025);

//Take the isosurface  data and create geometry
InitProgressBar(100);
skinNormals.SetInput(smooth.GetOutput());
skinNormals.SetFeatureAngle(60.0);

InitProgressBar(100);
maper.SetInput(skinNormals.GetOutput());
maper.ScalarVisibilityOff();

actorvolumen.SetMapper(maper);
actorvolumen.GetProperty().SetSpecular(0);
actorvolumen.GetProperty().SetDiffuse(1.0);
```

```
            actorvolumen.GetProperty().SetOpacity(1.0);
            actorvolumen.GetProperty().SetDiffuseColor(1.0, 0.95, 0.95);

            renderer.AddActor(actorvolumen);

            Exception exc = null;

            try
            {
                double[] Bounds = actorvolumen.GetBounds();
                SetupCammera(Bounds);

            }
            catch(Exception e)
            {
                exc = e;
            MessageBox.Show("No es posible generar el objeto tridimensional. Partes
            segmentadas desiguales.");
            }

            if(exc ==null)
            Refresh();
            PrgLoad.Visible = false;

        }
```

### 4.2.2 Segmentación de Imágenes

El fragmento de código que a continuación se muestra es el encargado de crear la estructura (SegmentationData) para posteriormente pasar a la segmentación de la(s) imagen(es).

```
public void SegmentationImages(Serie serie, string Algorithm, Point seed, int pixel,
int lower, int upper, bool[] index)
        {
    1       data =  new SegmentationData(serie, Algorithm, seed, pixel, lower,
upper, index);

            try
            {
    2           InitSegmentation();
                segmentation.SegmentationImagesAsync(data);
            }
            catch (InvalidOperationException exc)
            {
                throw new Exception("La aplicación se encuentra ejecutando una tarea
de segmentación de imágenes...");
            }
            catch (Exception exc)
            {
    3           throw exc;
            }
```

```
        }
```

1. Inicialmente se crea la estructura (SegmentationData) con algunos datos, tales como la serie, el algoritmo con el cual se va a realizar la segmentación, la posición del punto, el valor de intensidad del pixel y los umbrales de intensidad a segmentar.
2. Se inician los valores y se invoca el método de segmentación de imágenes con la estructura (SegmentationData) antes creada.
3. Finalmente si ocurre alguna excepción es capturada y lanzada para su tratamiento posterior.

#### 4.2.2.1 NeighborhoodConnected

```
private static Bitmap NeighborhoodConnected(Bitmap b1, int x, int y, int lower, int
upper, bool clonar)
        {
            Bitmap result = (clonar) ? (Bitmap)b1.Clone() : new Bitmap(b1.Width,
b1.Height, b1.PixelFormat);

            int[,] matriz = new int[b1.Height, b1.Width];

            Queue<Point> puntos = new Queue<Point>();
            {
                puntos.Enqueue(new Point(x, y));
                matriz[x, y] = 1;
            }

            BitmapData bmData = b1.LockBits(new Rectangle(0, 0, b1.Width,
b1.Height), ImageLockMode.ReadWrite, b1.PixelFormat);
            BitmapData bmDataR = result.LockBits(new Rectangle(0, 0, b1.Width,
b1.Height), ImageLockMode.ReadWrite, b1.PixelFormat);
            int srcOffSet = bmData.Stride - ((b1.PixelFormat ==
PixelFormat.Format8bppIndexed) ? b1.Width : b1.Width * 3);
```

En este fragmento de código se crea las variables que van a ser utilizadas en el método y además se carga en memoria la imagen que se está segmentando para recorrer los valores de los píxeles con mayor rapidez.

```
if (actual.X + 1 < b1.Height && actual.Y + 1 < b1.Width && matriz[actual.X + 1,
actual.Y + 1] != 1)
                        {
                            if (PointAcepted(ref b1, ref bmData, x + 1, y + 1, ref
lower, ref upper, ref matriz))
                            {
```

```
                    puntos.Enqueue(new Point(x + 1, y + 1));
                    matriz[x + 1, y + 1] = 1;
                }
            }
```

Este fragmento de código inicia con una condición para verificar que los valores de ancho y largo se encuentren en el rango permitido. Con la función PointAcepted se valida que el valor de intensidad de los píxeles vecinos se encuentren dentro del umbral permitido, posteriormente el punto es agregado al resultado final.

#### 4.2.2.2 ConnectedThreshold

```
int total = 0;
            Bitmap result = (clonar) ? (Bitmap)b.Clone() : new Bitmap(b.Width,
b.Height, b.PixelFormat);
            try
            {
                BitmapData bmData = b.LockBits(new Rectangle(0, 0, b.Width,
b.Height), ImageLockMode.ReadWrite, b.PixelFormat);
                BitmapData bmDataR = result.LockBits(new Rectangle(0, 0, b.Width,
b.Height), ImageLockMode.ReadWrite, b.PixelFormat);
                int srcOffSet = bmData.Stride - ((b.PixelFormat ==
PixelFormat.Format8bppIndexed) ? b.Width : b.Width * 3);
```

De esta manera son creadas las variables que se van a utilizar de manera similar al algoritmo NeighborhoodConnected.

```
if (actual.Y + 1 < b.Width && matriz[actual.X, actual.Y + 1] != 1)
{
 B = src[(actual.Y + 1) * bmData.Stride + 3 * actual.X];
 G = src[(actual.Y + 1) * bmData.Stride + 3 * actual.X + 1];
 R = src[(actual.Y + 1) * bmData.Stride + 3 * actual.X + 2];
 v = (byte)(0.299f * R + 0.587f * G + 0.114f * B);
      if (v >= lower && v <= upper)
      {
            puntos.Enqueue(new Point(actual.X, actual.Y + 1));
            matriz[actual.X, actual.Y + 1] = 1;
      }
}
```

Este fragmento verifica de igual manera al algoritmo anterior, la diferencia radica en que sólo se valida que el punto actual se encuentre en el rango del umbral permitido, posteriormente este punto es agregado al resultado final.

**CONCLUSIONES**

En este capítulo se presentó el diagrama de componentes para la representación en componentes físicos; además fueron descritos algunos fragmentos de la implementación del código del componente de software para brindar claridad y entendimiento a los desarrolladores que requieran hacer uso del mismo.

# BENEFICIOS ESPERADOS E IMPACTO

El componente tiene un significativo impacto en el área neurológica, ya que constituye una herramienta para asistir a la planificación quirúrgica. La misma facilita el trabajo de los especialistas al no tener que imaginar el carácter tridimensional de las estructuras y lesiones a abordar posteriormente. Esta herramienta le permite a los especialistas realizar un estudio previo del caso antes de ir al quirófano y realizar su planificación quirúrgica.

Permite a los especialistas de la salud cubana contar con una herramienta desarrollada en Cuba, que puede ser mejorada; tiene garantizado su soporte técnico y ahorra al país recursos financieros al no tener que adquirirla a precios cada día mayor en el mercado.

# CONCLUSIONES GENERALES

El estudio de las técnicas y métodos existentes para la reconstrucción tridimensional de neuroimágenes y la especificación de los requisitos funcionales del componente de software posibilitaron la definición de las funcionalidades de la aplicación informática para la visualización tridimensional de neuroimágenes.

La utilización de los diferentes patrones de diseño y arquitectura permitieron diseñar una solución considerando las buenas prácticas de desarrollo de software.

Con el desarrollo se implementaron las clases del diseño definidas y se obtuvo un componente de software capaz de asistir a las planificaciones quirúrgicas a través de las diferentes técnicas de reconstrucción tridimensional de neuroimágenes.

# RECOMENDACIONES

Realizar una segunda versión del componente en la que se utilice de una manera más eficaz el hardware a través de un procesador GPU (Unidad de Procesamiento Gráfico) integrado en tarjetas gráficas para agilizar los algoritmos de renderizado de las imágenes y que los mismos ocurran en tiempo real.

Implementar el componente de software en otros lenguajes de programación que sean multiplataforma.

Integrar este componente de software al Sistema de procesamiento, fusión y visualización tridimensional de neuroimágenes.

# REFERENCIAS BIBLIOGRÁFICAS


1. Mariano Alcañiz, Vicente Grau, Carlos Monserrat, Carmen Juan. Sistema de neurocirugía asistida por ordenador mediante computación de altas prestaciones. Desarrollo y validación clínica. [En línea] [Citado el: 15 de Octubre de 2008.] Disponible en: http://users.dsic.upv.es/~cmonserr/Articulos/AA160.pdf
2. Clemens Szyperski. Component Software Component Software Beyond Object – Oriented Programming, 1998. [Citado el: 15 de Octubre de 2008.]
3. Jonás A. Montilva C., Nelson Arapé y Juan Andrés Colmenares. Desarrollo Basado en Componentes, p 3. 2003 [Citado el: 15 de Octubre de 2008.]
4. Salvador Olmos. Imagen médica: información estructural y funcional del cuerpo humano en vivo. [En línea] 22 de octubre de 2007. [Citado el: 20 de Octubre de 2008.] Disponible en: http://www.ibercajalav.net/img/imagen_medica.pdf
5. Diccionario en línea. Medciclopedia. [En línea] 2009. [Citado el: 20 de Octubre de 2008.] Disponible en: http://diccionario.babylon.com/Neuroimagen
6. Fundación San Juan de Dios. [En línea] 2005. [Citado el: 20 de Octubre de 2008.] Disponible en: http://www.fsjd.org/cas/ic2_noticies.php?art_id=106&idioma=2
7. Cesar J. Acuña, Esperanza Marcos, Valeria de Castro, Juan A. Hernández, Marcos López Sanz. Gestión de Imágenes Médicas a Través de la Web. [En línea] [Citado el: 20 de Octubre de 2008.] Disponible en: http://caribdis.unab.edu.co/pls/portal/docs/PAGE/REVISTACOLOMBIANACOMPUTO/RCC_ESPANOL/NUMEROSANTERIORES/JUNIO2007/R81_ART1_C.PDF
8. The Analyze data format. [En línea] 1 de septiembre de 2006. [Citado el: 20 de Octubre de 2008.] Disponible en: http://imaging.mrc-cbu.cam.ac.uk/imaging/FormatAnalyze
9. NIfTI: Neuroimaging Informatics Technology Initiative. [En línea] 25 de Septiembre de 2007. [Citado el: 25 de Octubre de 2008.] Disponible en: http://nifti.nimh.nih.gov/nifti-1
10. Bankman, I. Handbook of Medical Imaging. Processing and Analysis, 2000. [Citado el: 25 de Octubre de 2008.]
11. Métodos de Segmentación de Imágenes Médicas. [En línea] Mayo de 2003. [Citado el: 24 de Octubre de 2008.] Disponible en: http://lcg.ciens.ucv.ve/~ernesto/nds/CotoND200305.pdf
12. Moreno Berzal Ignacio. Desarrollo de algoritmos de procesamiento de imágenes con VTK. [En línea]. [Citado el: 27 de Octubre de 2008.] Disponible en:

http://www.elai.upm.es/spain/Investiga/GCII/personal/iberzal/PFC_I_Berzal.pdf

13. Inostroza Patricio. Computación Gráfica. [En línea]. [Citado el: 27 de Octubre de 2008.] Disponible en: http://www.dcc.uchile.cl/~cc52b/Apuntes/Clase13-RayTracing.pdf
14. Pawasauskas John. 18 de febrero 1997. [En línea] [Citado el: 27 de Octubre de 2008.] Disponible en: http://web.cs.wpi.edu/~matt/courses/cs563/talks/powwie/p1/ray-cast.htm
15. Carranza Athó Fredy, Florian Cruz Laura. OPENGL Y VTK. [En línea] 2006. [Citado el: 25 de Octubre de 2008.] Disponible en: http://www.seccperu.org/files/OPENGL%20Y%20VTK.pdf
16. Javier Gutiérrez, Violeta Rangel, Rubén Medina, Luis A. Núñez. Reporte de Búsqueda de Bibliotecas de funciones Gráficas, de Alto Rendimiento y de Dominio Público en INTERNET. [En línea] Junio 2000. [Citado el: 26 de Octubre de 2008.] Disponible en: http://ecotropicos.saber.ula.ve/db/ssaber/Edocs/centros_investigacion/cat/publicaciones/papers/luisnunez/libvisu.pdf
17. Radeck, K. C# and Java: Comparing Programming Languages. Microsoft Corporation, Febrero de 2003. [Citado el: 31 de Octubre de 2008.]
18. Oualline, S. Practical C++ Programming. O'Reilly and Associates, Inc. Pag: 3-7. 1995. [Citado el: 31 de Octubre de 2008.]
19. Charte,F. Visual C#.Net. Anaya Multimedia, SA. 2002. [Citado el: 31 de Octubre de 2008.]
20. Smith, E. A., Whisler, V., et al. Visual Basic 6 Bible. IDG Books WorldWide, Inc. USA, 1998. [Citado el: 31 de Octubre de 2008.]
21. Deitel, H. M. y Deitel, P. J. Java How to Program. Prentice Hall. 2002. [Citado el: 5 de Noviembre de 2008.]
22. Robinson, S., Cornes, O., et al. Professional C#. Wrox Press Ltd. 2001. [Citado el: 5 de Noviembre de 2008.]
23. URRIELLUnet. [En línea] [Citado el: 5 de Noviembre de 2008.] Disponible en: http://urriellu.net/es/articles-software/csharp-advantages.html
24. eduangi telecom – Proyectos de Open Source y Telecomunicaciones. [En línea] 4 de Agosto de 2003[Citado el: 25 de Noviembre de 2008.] Disponible en: http://web.madritel.es/personales3/edcollado/ingsw/tema2/2-4.htm
25. Metodologías de Desarrollo Software [En línea]Febrero de 2008 [Citado el: 25 de Noviembre de 2008] Disponible en: http://www.scribd.com/doc/2050925/metodologias-de-desarrollo-software
26. Beck, K. Extreme Programming Explained. Addison-Wesley Professional, 1999.
27. Proceso de Desarrollo OpenUP – Córdova Software Factory [En línea] 2 de Septiembre de 2008 [Citado el: 25 de Noviembre de 2008.] Disponible en:

http://cbasqa.wordpress.com/2008/09/02/proceso-de-desarrollo-openup/

28. IBM Rational Unified Process – Grupo Soluciones Innova [En línea]2007 [Citado el: 25 de Noviembre de 2008] Disponible en: http://www.rational.com.ar/herramientas/rup.html
29. Jacobson, Ivar; Booch, Grady; Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. [Citado el: 10 de Diciembre de 2008.]
30. APEXNET - Enterprise Architect. [En línea] 22 de Octubre de 2008 [Citado el: 10 de Diciembre de 2008] Disponible en: http://www.apexnet.com.ar/index.php/product/viewProducts/24/sl=0
31. Enterprise Architect - Herramienta de diseño UML. [En línea] 2007 [Citado el: 10 de Diciembre de 2008] Disponible en: http://www.sparxsystems.com.ar/products/ea.html
32. Vitter,D; Templeman J. Visual Studio .NET, 2002. [Citado el: 10 de Enero de 2009]
33. Powers, R; Snell, M. Microsoft Visual Studio 2008 UNLEASHED, 2008. [Citado el: 12 de Enero de 2009]
34. Mono. [En línea] [Citado el: 12 de Enero de 2009] Disponible en: http://www.mono-project.com/MoMA
35. Lago Torres, Manuel. Introducción al diseño con patrones [En línea] Diciembre de 2002. [Citado el: 13 de Enero de 2009.] Disponible en: http://www.elrincondelprogramador.com/default.asp?id=29&pag=articulos/leer.asp
36. Vilas Fernández Ana [En línea] marzo de 2001 [Citado el: 15 de Enero de 2009.] Disponible en: http://www-gris.det.uvigo.es/~avilas/UML/node42.html
37. Ingeniería del Software (3º I.T.I.S., I.T.I.G.) Módulo 2. Tema 12: Modelo de Implementación [En línea] [Citado el: 15 de Enero de 2009.] Disponible en: http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf

# BIBLIOGRAFÍA


1. Acharya T, Ray Ajoy K. Image Processing, Principles and Applications. Wiley- Interscience, 2005. 428 páginas. [Citado el: 19 de Octubre de 2008]
2. Adams R, Bischof L. Seeded region growing, Patter Recognition, 1994, Vol. 16(no.6), 641-647. [En línea] [Citado el: 16 de Octubre de 2008] Disponible en: http://portal.acm.org/citation.cfm?id=628615
3. Ancuta Paul-Nicolae. 3D Object Modeling and Visualization Software for Surgery Preoperative Plan, 2008. [En línea] [Citado el: 16 de Octubre de 2008] Disponible en: http://www.cefin.ro/docs/conferinta/3.%203D%20Object%20Modeling%20and%20Visualization%20Software%20for%20Surgery%20Preoperative%20Plan.pdf
4. Bankman Isaac N. Handbook of Medical Imaging. Processing and Analysis. Estados Unidos, Academic Press, 2000. 901 páginas. [Citado el: 16 de Octubre de 2008]
5. Foley J. D, van Dam A, Feiner S. K, Hughes J. F, Phillips R. L. Introduction to Computer Graphics. Addison-Wesley, 1994. 632 páginas. [Citado el: 16 de Febrero de 2009]
6. Gonzáles Rafael C, Woods Richard E. Tratamiento Digital de Imágenes. Addison-Wesley Iberoamericana S.A, 1996. 800 páginas. [Citado el: 6 de Enero de 2009]
7. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. s.l.: PEARSON EDUCACIÓN S.A, 2000. [En línea] [Citado el: 12 de Diciembre de 2008] Disponible en: http://bibliodoc.uci.cu/pdf/reg00060.pdf
8. Jahne B. Practical Handbook on Image Processing for Scientific Applications. CRC Press, 1997. 608 páginas. [Citado el: 23 de Octubre de 2008]
9. Kaufman A, Cohen, D, Yagel R. Volume Graphics, IEEE Computer, 1993, Vol. 7. 51-64. [En línea] [Citado el: 13 de Enero de 2009] Disponible en: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&isnumber=&arnumber=274942
10. Kaufman Arie, Mueller Klaus. Overview of Volume Rendering, 2005. [En línea] [Citado el: 22 de Octubre de 2008] Disponible en: http://www.cs.sunysb.edu/~mueller/papers/volvisOverview.pdf
11. Knittel, G, Strasser, W. A compact Volume Rendering Accelerator, Volume Visualization Symposium Proceedings, 1994, 67-74. [En línea] [Citado el: 20 de Enero de 2009] Disponible en:

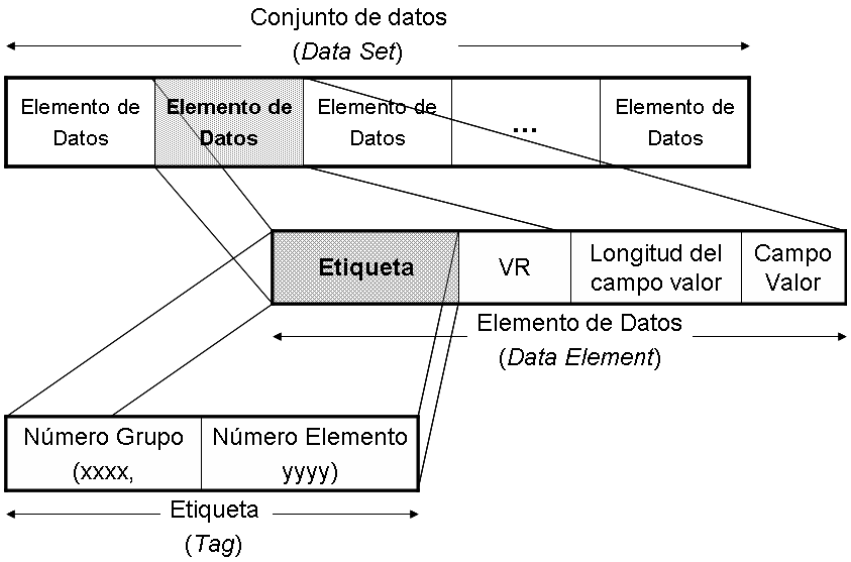
http://portal.acm.org/ft_gateway.cfm?id=197968&type=pdf&coll=GUIDE&dl=GUIDE&CFID=33717943&CFTOKEN=19978288

12. Larman, C. UML y Patrones: Introducción al análisis y programación orientada a objetos. México, Prentice Hall, 1999, 536 p.; (MON-001311) [En línea] [Citado el: 23 de Octubre de 2008] Disponible en: http://bibliodoc.uci.cu/pdf/reg00061.pdf
13. Miralles Pechuán Luis. Uso de ontologías para guiar el proceso de segmentación de imágenes, mayo del 2008. [En línea] [Citado el: 15 de Noviembre de 2008] Disponible en: http://digitum.um.es/xmlui/bitstream/10201/861/1/memoria.pdf
14. Moreno Berzal Ignacio. Desarrollo de algoritmos de procesamiento de imágenes con VTK, 2004. [En línea] [Citado el: 23 de Octubre de 2008] Disponibe en: www.elai.upm.es/spain/Investiga/GCII/personal/iberzal/PFC_I_Berzal.pdf
15. Pressman, Roger S. Ingeniería del Software: un enfoque práctico. Parte I y II / Madrid, McGraw-Hill, 2002, ed. 5ta. (MON-002581) 601p. [En línea] [Citado el: 18 de Octubre de 2008] Disponible en: http://bibliodoc.uci.cu/pdf/reg02689.pdf
16. Russ John C. Handbook The Image Processing. CRC, 2006. 832 páginas. [Citado el: 18 de Octubre de 2008]
17. Schroeder Will, Martin Ken, Lorensen Bill. Visualization toolkit: an object-oriented approach to 3D graphics. Kitware, 2006. 528 páginas. [Citado el: 19 de Octubre de 2008]
18. Schroeder William J, Martin Kenneth M, Lorensen William E. The Design and Implementation Of An Object-Oriented Toolkit For 3D Graphics And Visualization. [En línea] [Citado el: 18 de Octubre de 2008] Disponible en: http://vtk.org/VTK/img/dioot.pdf
19. Semerano Dave. VTK Tutorial. [En línea] [Citado el: 19 de Octubre de 2008] Disponible en: http://cherokee.ncsa.uiuc.edu/~semeraro/PPT/VTK_TUTORIAL/v3_document.htm
20. Stytz M. R, Frieder G, Frieder O. Three Dimensional Medical Imaging: Algorithms and Computer Systems, ACM Computing Surveys, 1991, 421-499. [En línea] [Citado el: 5 de Febrero de 2009] Disponible en: http://portal.acm.org/ft_gateway.cfm?id=125155&type=pdf&coll=GUIDE&dl=GUIDE&CFID=32700681&CFTOKEN=70086800
21. Vidholm Erik, Agmund Jonas. Fast surface rendering for interactive medical image segmentation with haptic feedback. [En línea] [Citado el: 29 de Octubre de 2008] Disponible en: http://www.ep.liu.se/ecp/013/008/ecp01308.pdf

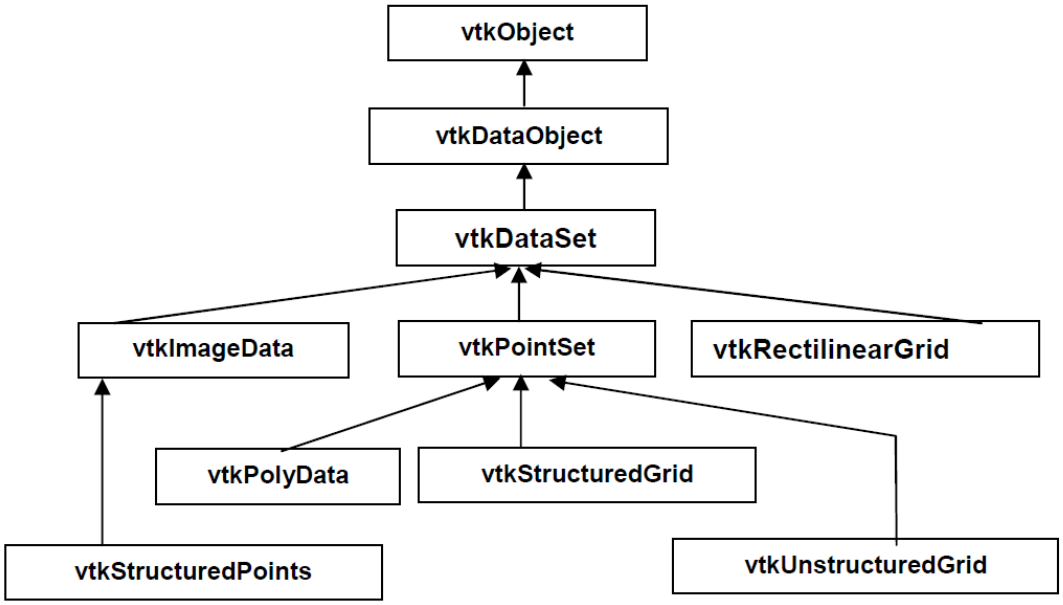
22. VTK 5.4.0 Documentation, marzo 2009. [En línea] [Citado el: 19 de Octubre de 2008] Disponible en: http://www.vtk.org/doc/release/5.4/html/index.html
23. Xu Chenyang ,Pham Dzung L , Prince Jerry L. Current Methods in Medical Image Segmentation, Annual Review of Biomedical Engineering, Annual Reviews, 2000, Vol. 2, 315-337. [En línea] [Citado el: 24 de Noviembre de 2008] Disponible en: http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.bioeng.2.1.315

# ANEXOS

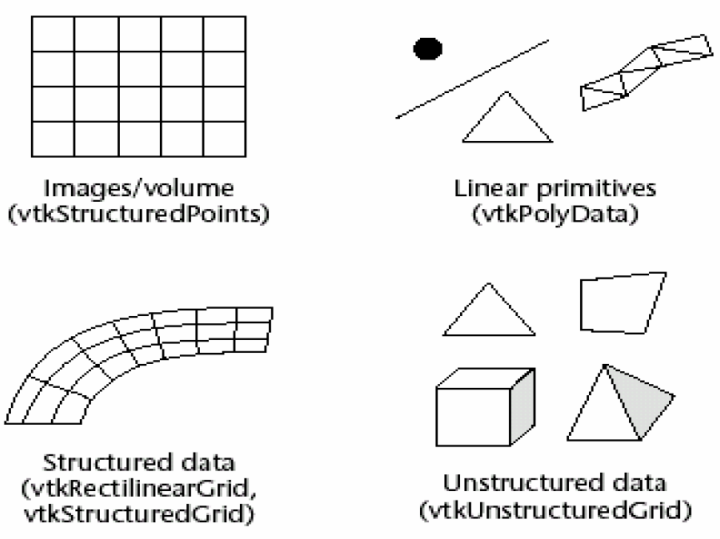
**Anexo 1: Estructura de los DataSet que forman la cabecera de un archivo DICOM.**



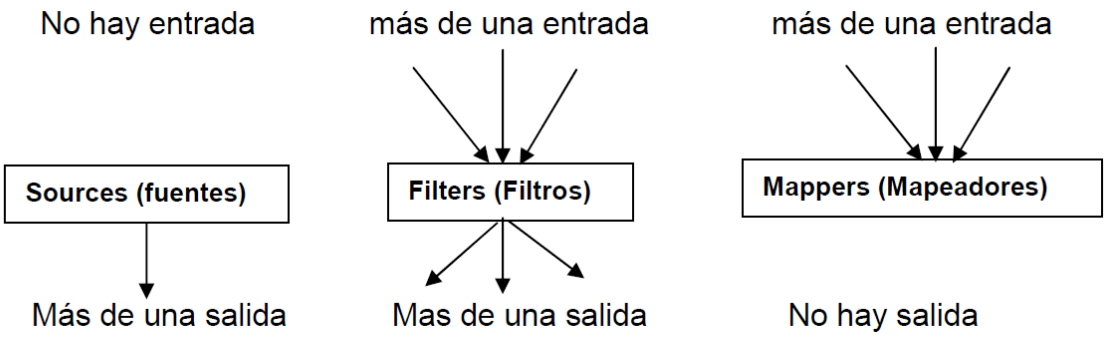
**Anexo 2: Diagrama de herencia de vtkDataObject.**




**Anexo 3: DataSets que utiliza VTK.**



**Anexo 4: Process Objects.**

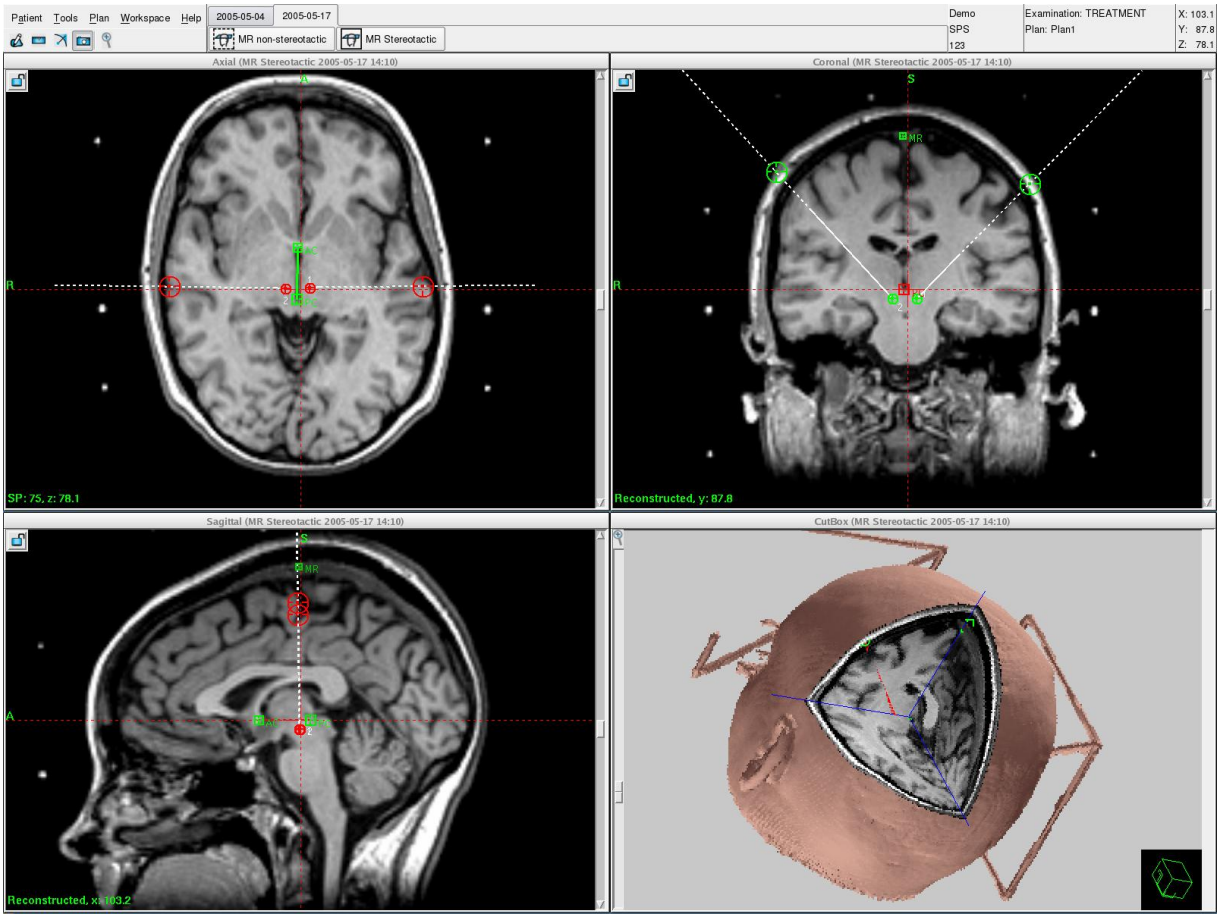



Los **Sources** (fuentes) son objetos que leen o generan algún tipo de dato, este es el objeto inicial de la visualización Pipeline.

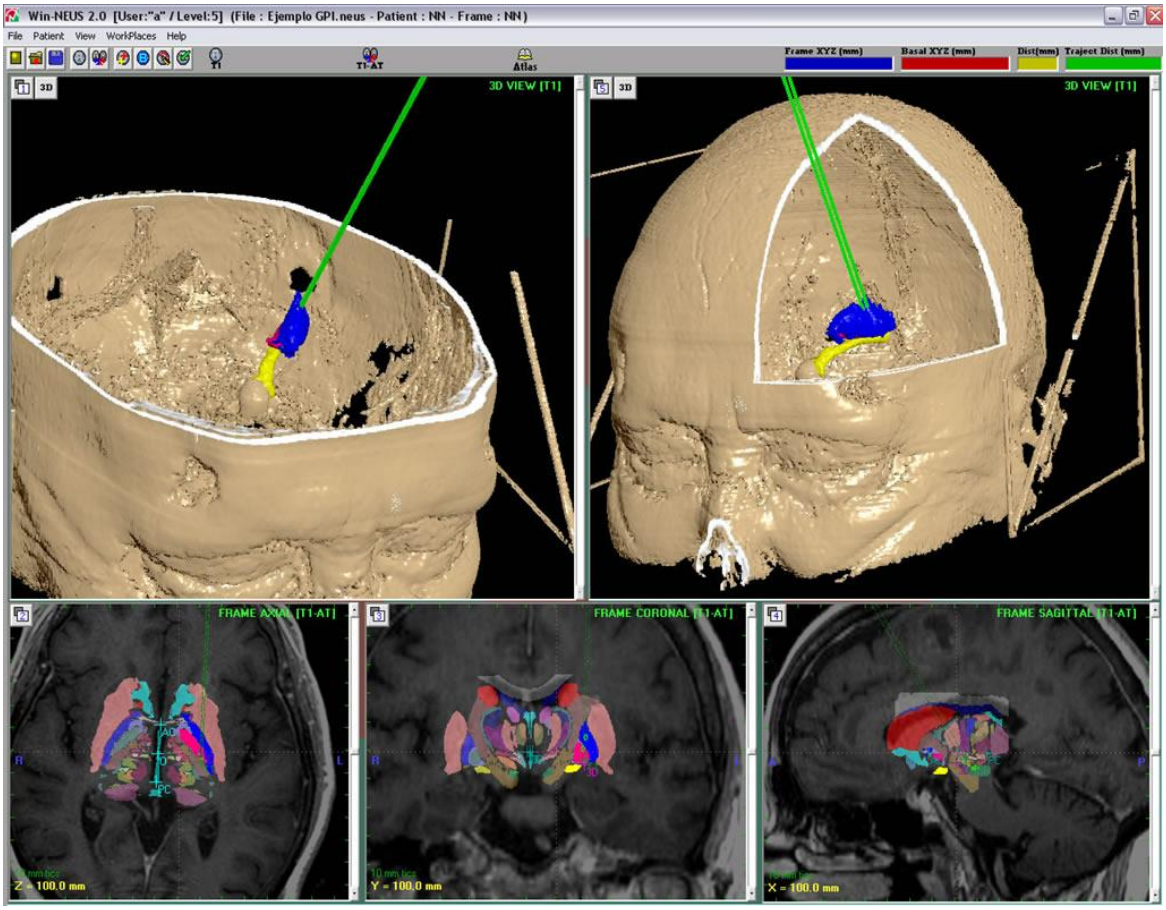
Los **Filters** (filtros) son objetos que pueden tener varios Data Objects de entrada, y generar uno ó más Data Objects en la salida.

Los **Mappers** (mapeadores) transforman los Data Objects en datos gráficos.

## Anexo 5: Leksell SurgiPlan.

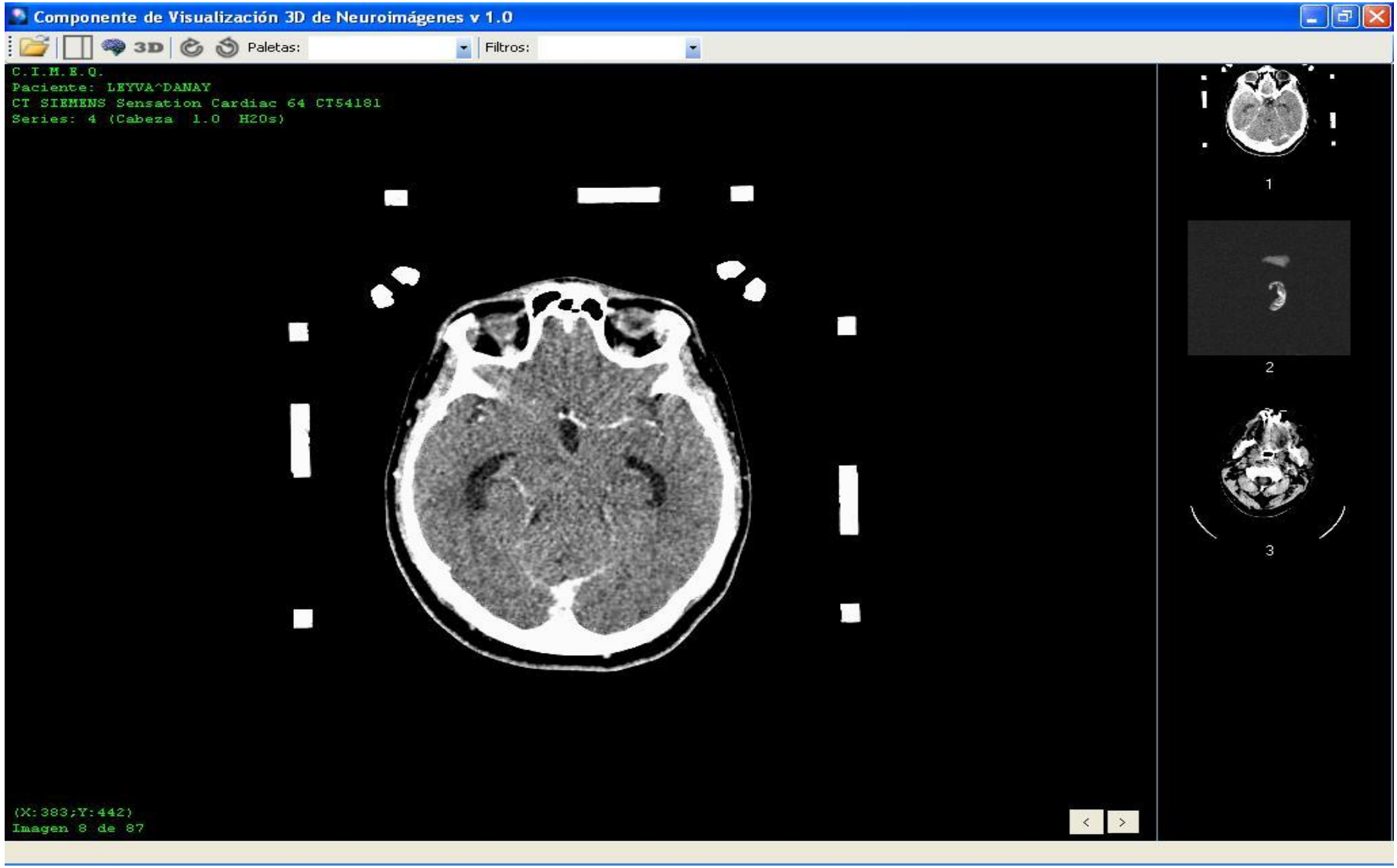


## Anexo 6: Win Neus 2.0.



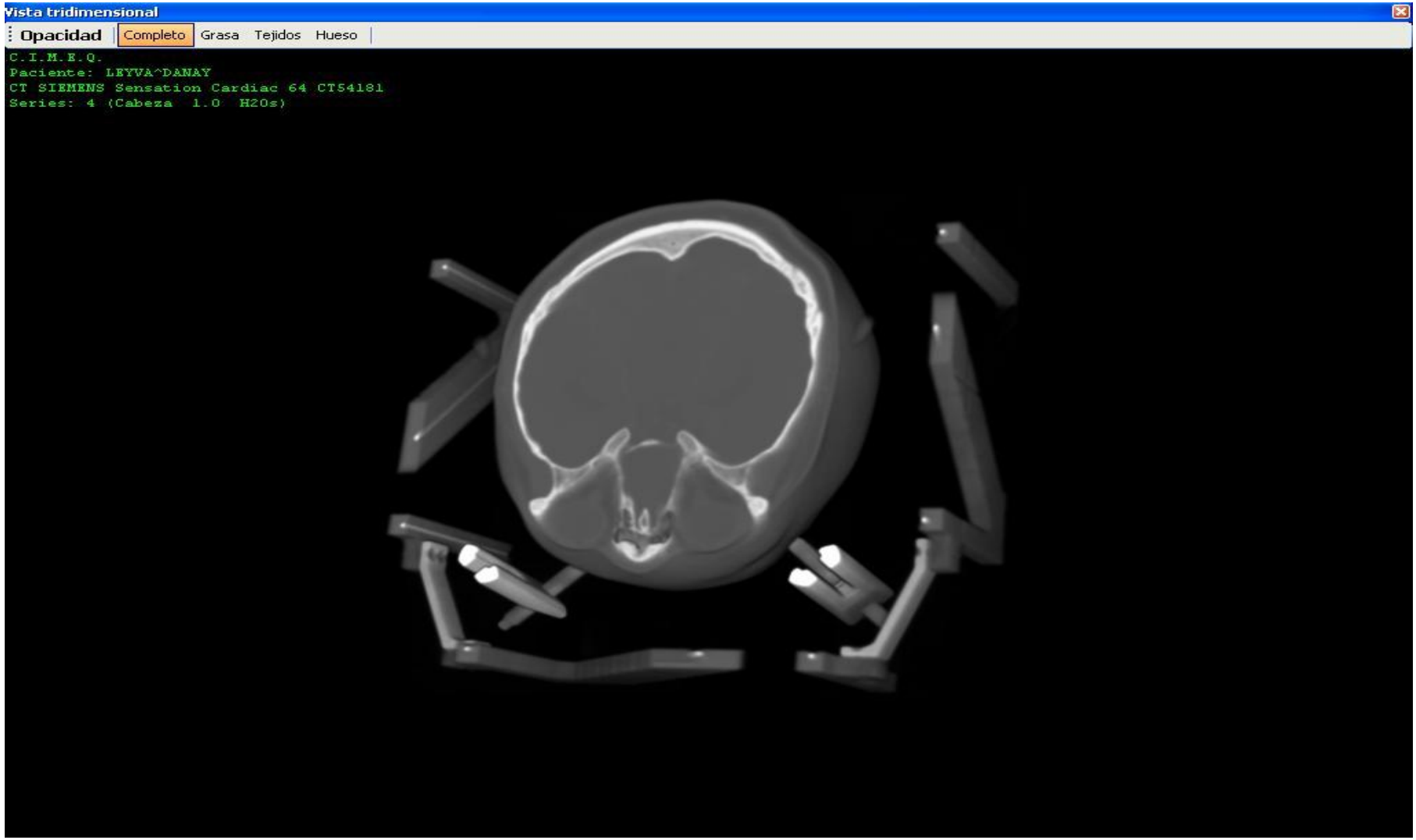
**Anexo 7: Stassis.**

**Anexo 8: Interfaz Vista principal.**



**Anexo 9: Interfaz Vista de segmentación.**

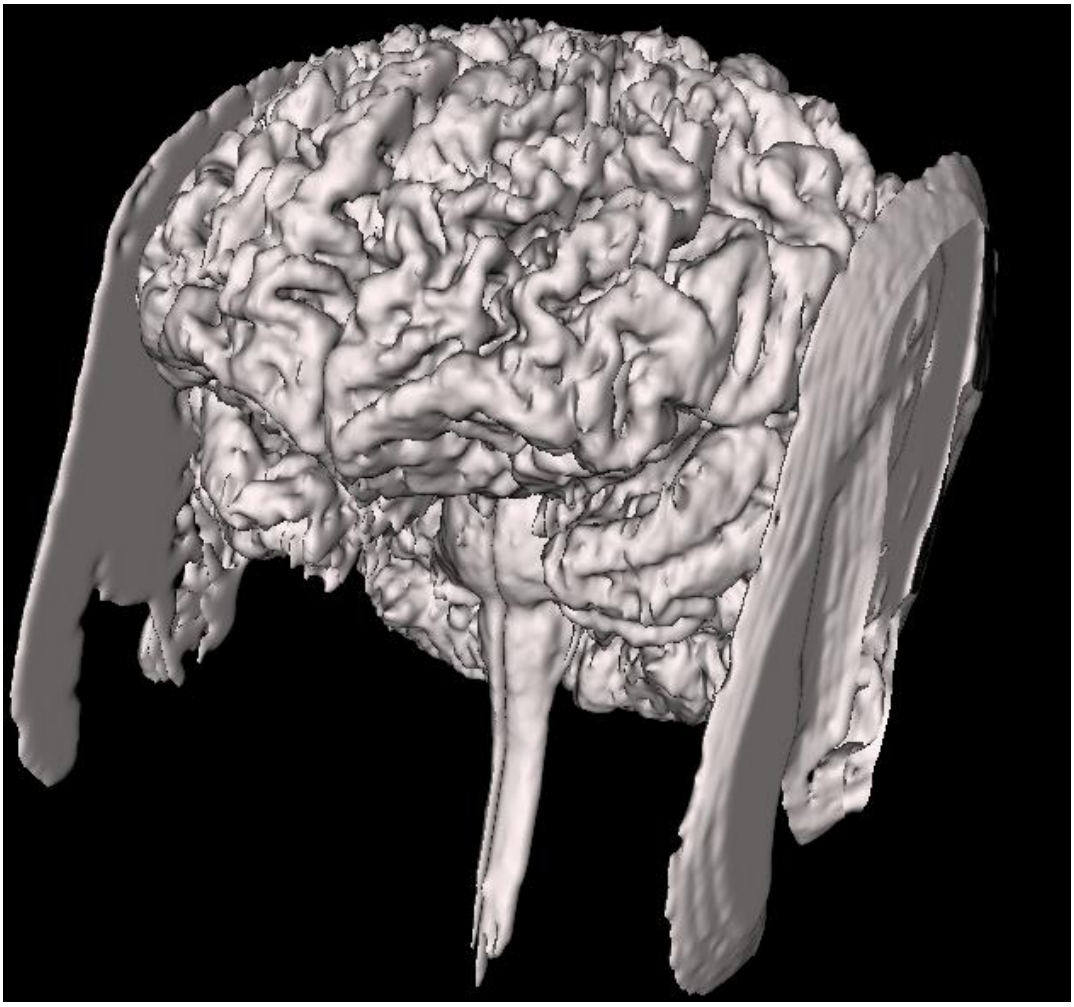
**Anexo 10: Interfaz Vista tridimensional (1).**



**Anexo 11: Interfaz Vista tridimensional (2).**

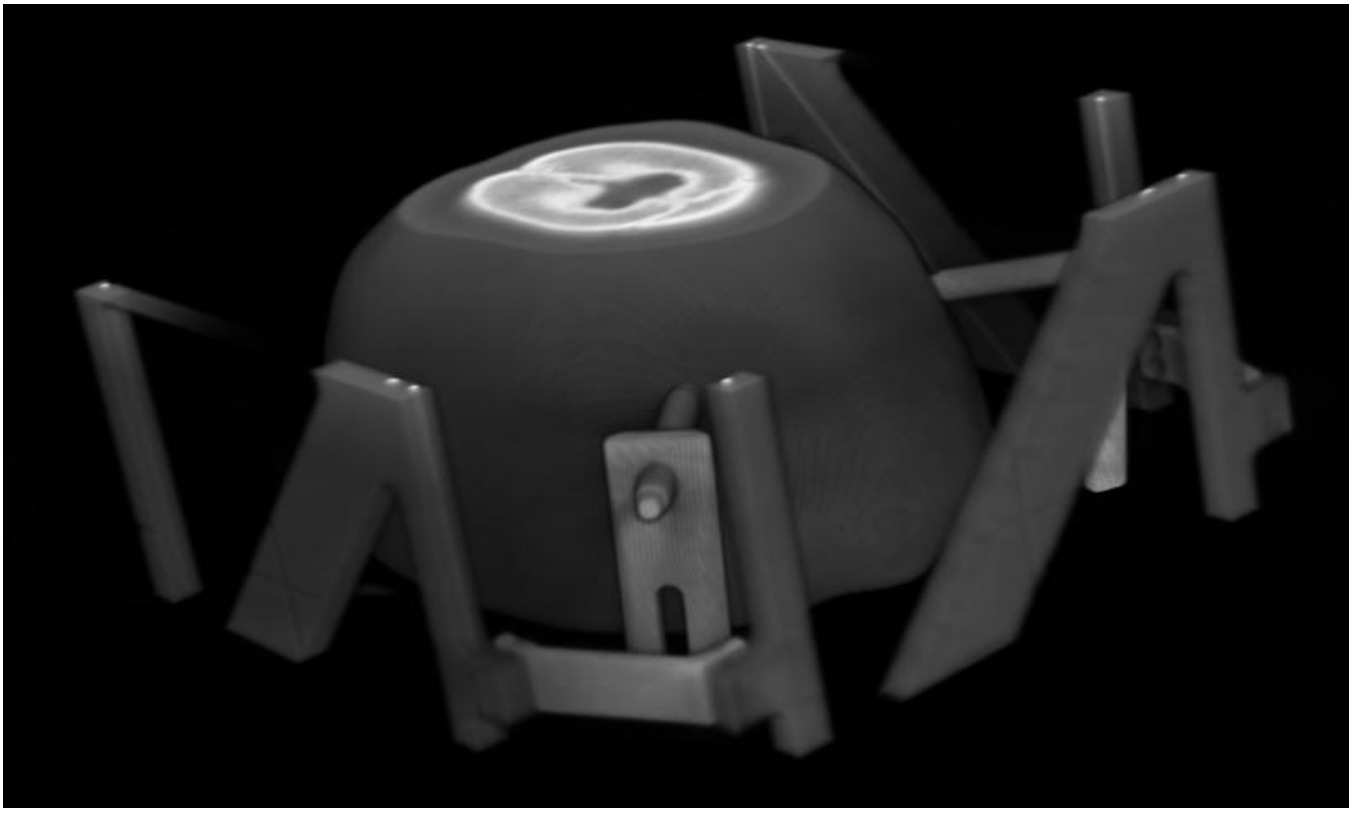
**Anexo 12: Objeto tridimensional (1).**

Objeto tridimensional obtenido con el componente de software de esta investigación utilizando la técnica de Surface Rendering.

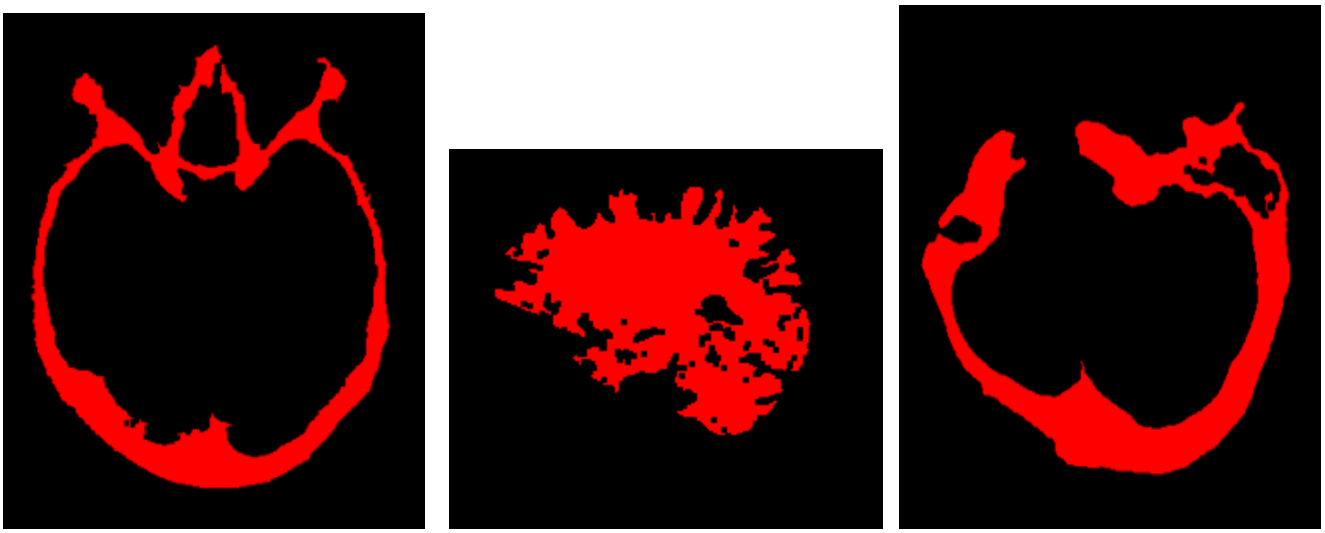


**Anexo 13: Objeto tridimensional (2).**

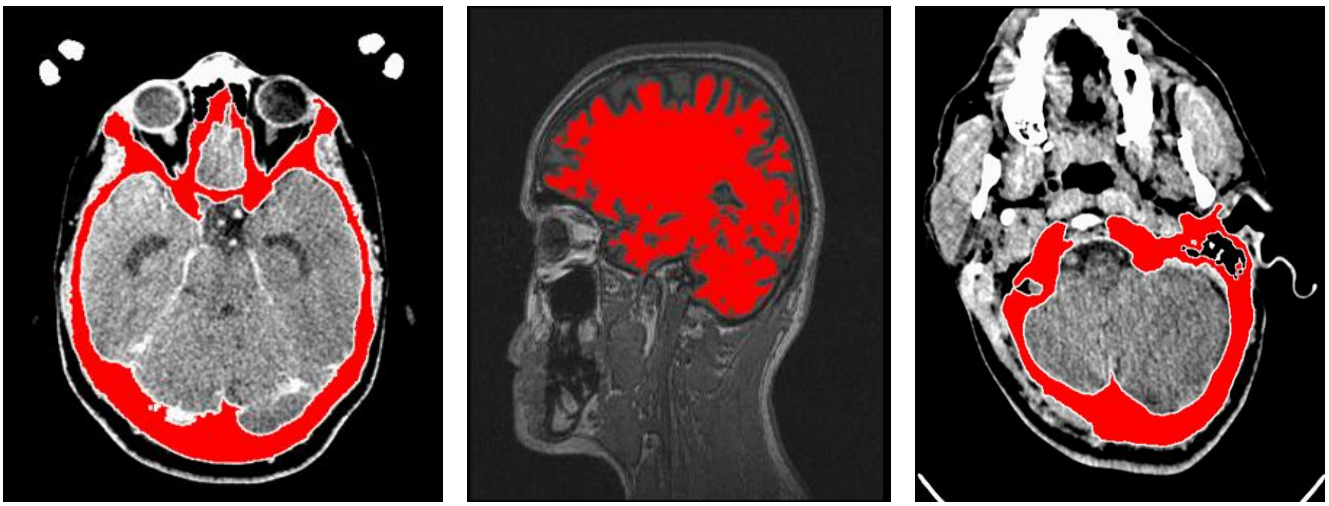
Objeto tridimensional obtenido con el componente de software de esta investigación utilizando la técnica de Volume Rendering.



**Anexo 14: Imágenes segmentadas con el componente de software de esta investigación.**

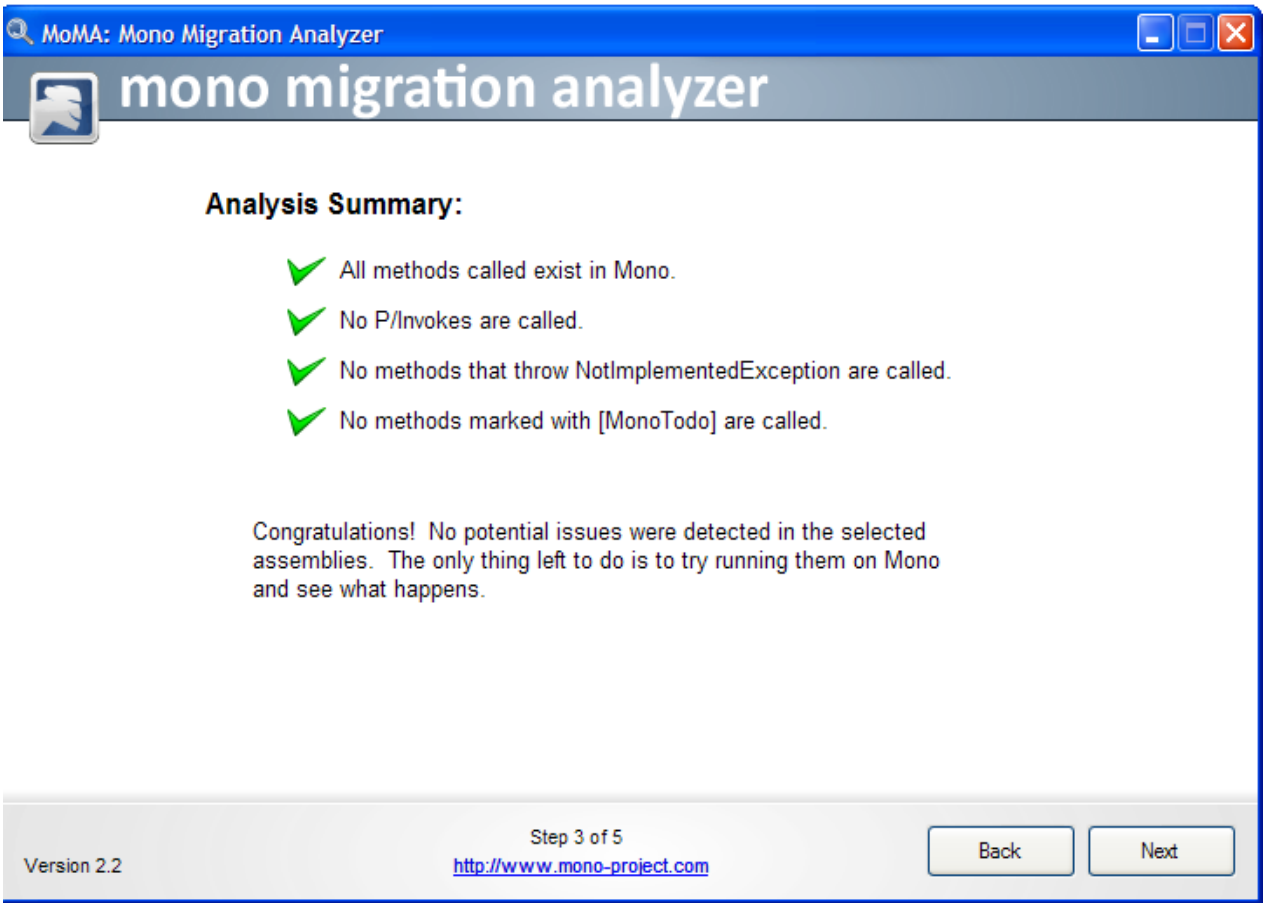


**Anexo 15: Imágenes combinadas con el componente de software de esta investigación.**



**Anexo 16: Compatibilidad con la plataforma libre Mono**

La herramienta Mono Migration Analyzer (MoMA) permite analizar si una aplicación .NET puede ser ejecutada satisfactoriamente en la plataforma Mono. La compatibilidad del componente Visualizacion3D.exe y los ensamblados de vtk(vtkCommonDotNet.dll, vtkFilteringDotNet.dll, vtkFormsWindow.dll, vtkGenericFilteringDotNet.dll, vtkGraphicsDotNet.dll, vtkHybridDotNet.dll, vtkImagingDotNet.dll, vtkIODotNet.dll, vtkParallelDotNet.dll, vtkRenderingDotNet.dll, vtkVolumeRenderingDotNet.dll y vtkWidgetsDotNet.dll), con la plataforma Mono, fue comprobada utilizando la herramienta MoMA v.2.2. Los resultados obtenidos demuestran que el componente propuesto en esta investigación es completamente portable hacia la plataforma libre.



# GLOSARIO

**2D:** 2 dimensiones, bidimensional.

**3D:** 3 dimensiones, tridimensional.

**ACR:** American College of Radiology.

**Analyze:** Formato de imágenes médicas donde se guarda los datos de la imagen en un archivo (*.img) y los datos del título (información sobre el nombre del paciente, el tipo de examen, dimensiones de la imagen, entre otros) en otro archivo (* .hdr).

**API:** Conjunto de funciones y métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

**Atlas estereotácticos:** Mapa por cortes sucesivos del cerebro en el que se identifican límites y estructuras asociados a un sistema de coordenadas estándar.

**CAO:** Cirugía Asistida por Ordenador.

**CIREN:** Centro Internacional de Restauración Neurológica.

**Corregistro:** Consiste en alinear geométricamente los sistemas de coordenadas de dos o más volúmenes de imágenes, de tal forma que exista correlación espacial entre la información contenida en los vóxel que representan una misma estructura anatómica.

**DataElement:** Elemento de datos. Estructura fundamental de datos en un sistema de procesamiento de datos.

**DataSet:** Colección de datos.

**DICOM:** Digital Imaging and Communications in Medicine, estándar reconocido mundialmente para el intercambio de imágenes médicas.

**DSA:** Angiografía Digital de Sustracción.

**EEG:** Electroencefalografía.

**Elekta:** Empresa sueca líder mundial en soluciones integrales para cirugía estereotáctica y radioterapia.

**fMRI :** Resonancia Magnética funcional.

**Fusión:** Implica la integración de información proveniente de dos o más modalidades en una nueva imagen. Esta imagen fusionada hereda todas o algunas de las características más representativas de las imágenes originales. En el caso particular de la fusión TAC/RMN para neurocirugía, la TAC aporta la fidelidad espacial (imágenes libres de distorsiones) y la RMN un alto nivel de detalle anatómico.

**ImageMerge:** Módulo opcional del sistema Leksell SurgiPlan para el corregistro y fusión de imágenes médicas.

**JPEG 2000:** Mejora del algoritmo de compresión de imágenes JPEG.

**JPEG Lossless:** Norma de compresión de imágenes.

**JPEG:** Joint Photographic Experts Group. Es la norma de compresión de imágenes más utilizada a nivel mundial.

**Leksell Stereotactic System:** Marco estereotáctico que provee la solución ofrecida por Elekta. Dicho marco estereotáctico fue creado por el profesor Lars Leksell, uno de los fundadores de Elekta.

**LUT:** LookUp Table, Paleta de colores. Conjunto de colores incluidos en una imagen.

**MEG:** Magnetoencefalografía.

**NEMA:** National Electrics Manufacturers Association.

**Neurocirugía estereotáctica:** Técnica tridimensional de Neurocirugía. Consiste en tomar algunas radiografías (u otro tipo de imágenes), basado en las cuales se identifica y se mapea una estructura (núcleo) dentro del cerebro.

**Neurocirugía:** Especialidad que estudia las enfermedades del sistema nervioso, tratables mediante intervención quirúrgica. (Incluye la prevención, diagnóstico, evaluación, tratamiento, cuidados intensivos, y rehabilitación).

**Neuroimagen:** Imagen médica realizada especialmente sobre estructuras anatómicas y funcionales estudiadas por la disciplina de neurología, generalmente imágenes de la anatomía cerebral.

**NIfTI:** Neuroimaging Informatics Technology Initiative. Formato de imágenes médicas.

**Nuclemed:** Empresa argentina dedicada al desarrollo, producción y distribución de Sistemas de planificación de tratamientos de Radiocirugía-Radioterapia y Sistemas de Planificación para Neurocirugía Estereotáctica.

**OpenGL:** Open Graphics Library. Es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.

**PET:** Positron Emission Tomography.

**Pipelines:** La arquitectura en pipeline consiste en ir transformando el flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior.

**Píxel:** Menor unidad homogénea en color que forma parte de una imagen digital.

**Proyecto Mono:** Proyecto de código abierto impulsado por Novell para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET

**Rendering:** Proceso de generar una imagen desde un modelo.

**RLE:** Run-length encoding.

**RMN:** Resonancia Magnética Nuclear.

**Segmentación:** Proceso mediante el cual es posible extraer o clasificar los objetos de interés dentro de una imagen o volumen.

**SOAP:** Simple Object Access Protocol. Protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

**SPECT:** Tomografía de Emisión de Fotón Simple.

**Surface Rendering:** Técnicas basadas en la construcción de superficies.

**TAC:** Tomografía Axial Computarizada.

**Triangulación de Delaunay:** Red de triángulos que cumple la condición de Delaunay. Esta condición dice que la circunferencia circunscrita de cada triángulo de la red no debe contener ningún vértice de otro triángulo.

**Volume rendering:** Técnicas de construcción de volúmenes semitransparentes.

**Vóxel:** Unidad cúbica que compone un objeto tridimensional.

**VTK:** The Visualization ToolKit. Librería de código abierto para generar gráficos 3D.

**WYSIWYG:** What You See Is What You Get ("lo que ves es lo que obtienes"). Se aplica a los editores de texto que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso.

**XML:** Extensible Markup Language, Lenguaje de Marcas Ampliable. Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).