UNIVERSIDAD AUTÓNOMA METROPOLITANA – IZTAPALAPA
DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

DESARROLLO CENTRADO EN ARQUITECTURA DE UN SISTEMA DE RECONSTRUCCIÓN 3D PARA VISUALIZACIÓN CONJUNTA DE IRM E IRMF

Tesis que presenta
Jimmy Jesús Durán Ravell
Para obtener el gado de
Maestro en Ciencias de Ingeniería Biomédica

Asesores: M. EN C. ALFONSO MARTÍNEZ MARTÍNEZ
M. EN IB. ÓSCAR YÁÑEZ SUÁREZ

Jurado Calificador:

Presidente: M. EN IB. ÓSCAR YÁÑEZ SUÁREZ

Secretario: ING. LUIS FERNANDO CASTRO CAREAGA

Vocal: M. EN C. MARCO ANTONIO NÚÑEZ GAONA

Ciudad de México, Abril de 2016

Universidad Autónoma Metropolitana
Unidad Iztapalapa
División de Ciencias Básicas e Ingeniería
Posgrado en Ingeniería Biomédica

Desarrollo Centrado en Arquitectura de un Sistema de Reconstrucción 3D para Visualización Conjunta de IRM e IRMf

Tesis

Que para optar por el grado de

Maestro en Ciencias de Ingeniería Biomédica

presenta

Ing. Jimmy Jesús Durán Ravell

Asesores:

M. en C. Alfonso Martínez Martínez

M. en IB. Óscar Yáñez Suárez

Gracias

A mis padres, Ileana y Jaime

El texto en una tesis no bastaría para expresarles mi admiración y agradecimientos por su apoyo incondicional, consejos oportunos y, sobre todo, su cariño constante.

Les quiero con toda mi alma.

Al profesor Alfonso Martínez

Por su inmensa paciencia y su imbatible espíritu docente; por tantas horas de dedicación y tantos consejos que no sólo han hecho salir adelante a este proyecto, sino también a mí como persona.

Al profesor Óscar Yáñez

Por sus atinadas observaciones y recomendaciones en cada etapa del proyecto; por su disposición de colaboración, entusiasmo y jovialidad a pesar del cansancio y el estrés.

A mis sinodales y a la Dra. Angelina Espinoza

Por su muy amable atención y retroalimentación brindadas para la redacción y corrección de esta tesis.

Índice general

Índice de cuadros

Índice de figuras

Resumen

El Laboratorio de Investigación en Neuroimagenología (LINI), establecido en la Universidad Autónoma Metropolitana, Iztapalapa, ha manifestado la necesidad de contar con un sistema de software capaz de integrar información biomédica proveniente de diferentes modalidades de imagen, específicamente imagenología por resonancia magnética, imagenología por resonancia magnética funcional y electroencefalografía, en una única representación tridimensional. Para poder generar un diseño del sistema que no sólo cumpliera con los requerimientos funcionales, sino que también le procurara los atributos de calidad necesarios para los interesados, se empleó un proceso de desarrollo de software fuertemente centrado en arquitectura. Además, las decisiones con respecto al diseño fueron tomadas con base en una revisión sistematizada de la literatura; esta revisión evidenció que hay pocos trabajos que propongan un sistema de visualización tridimensional que presenten de manera conjunta la información provista por las modalidades anteriormente mencionadas, y prácticamente ninguno efectúa el proceso de desarrollo adoptando un enfoque centrado en arquitectura de software.
Entonces, como resultado del diseño arquitectónico dentro de un proceso de desarrollo de software se ha obtenido un sistema de visualización tridimensional con el potencial de seguir expandiendo tanto su funcionalidad como sus atributos de calidad.

Capítulo 1

Introducción

1.1. Visualización tridimensional de información biomédica

Los sistemas de visualización tridimensional (3D) estuvieron por primera vez disponibles comercialmente en la década de los noventa [1], y desde entonces han apoyado a profesionales de la medicina para brindar diagnósticos, planear cirugías, evaluar la recuperación de pacientes, entre otros beneficios. La visualización 3D permite efectuar una inspección más detallada de la información biomédica, a diferencia de métodos más tradicionales de observación como, por ejemplo, las placas radiográficas. Dicha información puede provenir de diferentes modalidades como son resonancia magnética, tomografía computada, ultrasonido, etcétera, y puede presentarse en una única representación con la que el personal médico puede interactuar; así se evita la necesidad de efectuar complicadas abstracciones mentales o disecciones físicas. Esto es posible debido a que las técnicas empleadas en el procesamiento de imágenes y visualización 3D buscan enfatizar determinadas características observables y medibles en las estructuras de interés. Además, suele ser deseable que esas técnicas preserven el realismo y que coadyuven en la percepción de la información biomédica [2].

La visualización 3D tiene un extenso número de aplicaciones en distintas actividades clínicas; atender los requerimientos de cada aplicación conlleva enfrentar retos tecnológicos diferentes, además del (casi siempre) inherente requisito de procesar grandes cantidades de información proveniente de uno o varios equipos médicos. En consecuencia, la comunidad biomédica ha implementado sistemas de visualización muy diversos y complejos.

Una de las especialidades que más ha impulsado los avances en visualización 3D es la Neurocirugía [2]. Una vez que se ha aplicado algún procedimiento de *registro de imágenes*[1] provenientes de una o más modalidades, la representación 3D presenta al cirujano una visión más clara entre estructuras sanas y anormales, así como patrones de actividad cerebral. Toda esta información permite evaluar posibles vías para la aplicación de la instrumentación quirúrgica, efectuar mediciones sobre estructuras o entrever ventajas y desventajas de un determinado tratamiento.

En síntesis, la visualización 3D de información biomédica ofrece soluciones muy convenientes

[1] El registro de imágenes es el proceso de mapear el espacio en el que fue adquirido un conjunto de imágenes al sistema coordenado de otro con el fin de que estén alineados.

para atacar varios problemas en diversas especialidades médicas y, aún cuando su introducción en el ámbito clínico ha sido relativamente lenta, los rápidos avances tecnológicos prometen hacer los sistemas de visualización cada vez más accesibles y confiables.

1.2. Adoptando un enfoque de Ingeniería de Software

En un entorno clínico el personal médico está comprometido con el mejoramiento de la salud de sus pacientes; las herramientas empleadas deben proporcionar un alto grado de seguridad, disponibilidad y rendimiento para poder alcanzar esa meta. Es por ello que el desarrollo de los sistemas de visualización debe considerar una serie de requerimientos encaminados a proporcionar un buen apoyo al personal, incrementando así los beneficios para el paciente. Por otra parte, es probable que en un entorno académico los requerimientos estén enfocados principalmente en la facilidad de modificación del sistema y en el rendimiento, de tal forma que el sistema pueda adaptarse a las necesidades particulares del investigador, docente y alumnos. En general, el proceso de desarrollo de los sistemas de visualización, así como el de cualquier otro sistema de software, debe estar guiado tanto por las necesidades de sus usuarios como por las restricciones del entorno en que se desempeñan.

Efectivamente, todo sistema de software tiene como objetivo satisfacer ciertas necesidades dentro de una organización; el vínculo que se establece entre esas necesidades y la construcción de un sistema que las satisfaga es la arquitectura de software [3]. El proceso de convertir los objetivos, necesidades y restricciones de una organización en un sistema útil puede ser bastante complejo; la ventaja es que las arquitecturas de software pueden ser analizadas, diseñadas y documentadas usando técnicas bien establecidas, lo cual contribuye al cumplimiento de los requerimientos que se tengan para el sistema. Una arquitectura bien diseñada establece cómo el sistema puede cambiar y evolucionar, permite distribuir el trabajo entre los distintos miembros del equipo de desarrollo, ayuda a estimar tiempos y costos en el desarrollo, facilita la comunicación de las características del sistema a los interesados, entre otras ventajas. Todo esto constituye grandes ventajas al momento de construir sistemas complejos, y adquiere especial relevancia cuando dichos sistemas manejarán información importante como son imágenes o señales biomédicas que serán empleados para evaluar, diagnosticar o tratar a pacientes.

El diseño de una arquitectura de software es sólo parte del proceso de desarrollo de un sistema. Existen varias metodologías que pueden guiar a un equipo de trabajo a través del resto de las actividades en dicho proceso. Las metodologías tradicionales son aquéllas que prescriben el proceso de desarrollo marcando una serie de actividades concretas, y es por ello que suele llamárseles metodologías disciplinadas o guiadas por un plan. Incluidas en estas metodologías están: el Proceso Unificado (UP) [4], OpenUP [5], Proceso Unificado de Rational (RUP) [6], Proceso Personal de Software [7], por mencionar algunas. Por otra parte existen aquéllas denominadas metodologías ágiles, que son las que valoran principalmente la comunicación eficaz entre miembros del equipo de desarrollo, que se asegure su motivación constante, que colaboren estrechamente con los clientes, y que se entregue software funcional con regularidad [8]. Entre estas metodologías están Scrum [9], Programación Extrema (XP) [10], Programación guiada por Pruebas (TDD) [11], Kanban [12], entre otras. La dife-

rencia principal con las metodologías disciplinadas es que no prescriben las tareas a seguir durante el proceso, sino que constituyen una serie de recomendaciones y buenas prácticas a seguir durante el proceso de desarrollo. Aunque ambos tipos de metodología tienen como último objetivo la entrega de un sistema de software, la elección de emplear una u otra metodología dependerá en buena medida del tiempo disponible para el proyecto, la disponibilidad de los clientes para participar en el proceso de desarrollo, la cantidad de miembros del equipo, su experiencia y la distancia física existente entre ellos.

1.3. Motivación del proyecto

El presente proyecto fue planteado a partir de la necesidad expresada por el Laboratorio de Investigación en Neuroimagenología (LINI)[2] de contar con un sistema de software capaz de integrar información biomédica proveniente de diferentes modalidades, específicamente imagenología por resonancia magnética (IRM), imagenología por resonancia magnética funcional (IRMf) y electroencefalografía (EEG), así como volúmenes de información generados a partir de soluciones al problema inverso[3] de EEG, en una única representación tridimensional. Algunos de los requerimientos expresados inicialmente incluyen: visualizar de manera conjunta datos anatómicos y funcionales en una única proyección tridimensional, permitir manipulaciones básicas a dicha proyección (como rotación y ampliación/reducción), y asegurar que el sistema permita la incorporación de diferentes métodos de visualización y/o de diferentes bibliotecas de software encargadas de llevarlos a cabo. Luego de haber efectuado una revisión de la literatura en forma sistemática (presentada en el Capítulo 2), se han encontrado pocos trabajos que propongan un sistema de visualización que presente de manera conjunta la información provista por las modalidades anteriormente mencionadas. Por lo tanto, para lograr construir un sistema de visualización con estas características se ha propuesto un objetivo general y varios particulares.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un sistema de software para procesamiento y visualización de imágenes y señales cerebrales, enfocándose en la visualización tridimensional de estructuras anatómicas obtenidas por IRM, el mapeo sobre las mismas de potenciales de EEG, las soluciones al problema inverso y la información provista por la IRMf.

[2] Está ubicado en el laboratorio T-226 de la Universidad Autónoma Metropolitana, unidad Iztapalapa.

[3] El problema directo se refiera a, dada alguna fuente eléctrica o magnética, conocer el potencial eléctrico o campo magnético que se registra en un punto determinado (como, por ejemplo, el cuero cabelludo en el caso del EEG). Por lo tanto, el problema inverso conlleva buscar y ubicar esas fuentes eléctricas o magnéticas mediante el registro y procesamiento de los potenciales eléctricos y campos magnéticos que generan citeSarvas1987. Actualmente, una gran cantidad de estudios proponen y evalúan estrategias eficientes para resolver el problema inverso en EEG.

1.4.2. Objetivos particulares

El objetivo general se alcanzará mediante el cumplimiento de los siguientes objetivos particulares:

- Reunir y priorizar los requerimientos funcionales y no funcionales para el sistema de visualización.
- Efectuar una revisión bibliográfica sistematizada centrada en temas y herramientas de visualización para justificar las decisiones en el diseño del sistema.
- Diseñar una arquitectura de software que cumpla con los atributos de calidad solicitados.
- Verificar que se cumplan los requerimientos funcionales y los principales atributos de calidad.
- Implementar y poner en operación el sistema de visualización.

Capítulo 2

Estado del Arte

2.1. Estudio de Mapeo Sistemático

El proceso de efectuar un Estudio de Mapeo Sistemático (denominado *Systematic Mapping Study* en la literatura en inglés, y que a partir de ahora se abreviará como EMS) ha sido muy abordado en las áreas médicas [13, 14], y ciertos autores en la ingeniería de software han enfocado esfuerzos en aplicarlo a sus áreas de investigación pues constituye un método organizado de buscar y categorizar material bibliográfico. Lo que se ha planteado en este trabajo es una instancia de EMS adaptada a las necesidades del proyecto, y su propósito es presentar de manera clara y justificada el estado del arte en áreas de investigación de interés mediante la identificación y clasificación de los trabajos más relevantes. Dichos trabajos constituyen los *estudios primarios* encontrados como resultado de la realización del EMS. Por su parte, los EMS son *estudios secundarios* que se encargan de resumir e integrar la información contenida en los estudios primarios. También existen los denominados *estudios terciarios* que corresponden a revisiones de varios estudios secundarios. En su conjunto, todos esos estudios conforman la evidencia requerida para sustentar las decisiones en un proyecto. La realización de un EMS se consideró muy conveniente para resumir el estado del arte relacionado a métodos de visualización 3D de imágenes médicas, las herramientas empleadas en el diseño y construcción de sistemas que la emplean, así como los conceptos aplicables de ingeniería de software, dado que estos temas son sumamente amplios y complejos.

2.1.1. Definición del protocolo para el EMS

Una vez reconocida la necesidad de efectuar un EMS el siguiente paso fue la elaboración de un protocolo que guiara las actividades para realizarlo. Éstas actividades son las siguientes:

1. Formular las *preguntas de investigación* a las que se quiere dar respuesta mediante el EMS.

2. Definir una estrategia de búsqueda. Esto es, cómo se llevará a cabo la búsqueda de estudios primarios, incluyendo los recursos (acervos de biblioteca, motores digitales de búsqueda, etc.) y términos de búsqueda propuestos.

3. Determinar los *criterios de selección* para los estudios primarios, es decir, los criterios de inclusión y de exclusión.

4. Establecer los procedimientos de selección de estudios primarios. En otras palabras, las especificaciones sobre cómo se aplicarán los criterios de inclusión y exclusión.

5. Estipular la estrategia de extracción de datos. Esto es el procedimiento que se usará para identificar la información relevante de cada estudio primario.

6. Exponer la estrategia para la síntesis y la presentación de resultados.

Cabe mencionar que varias de estas actividades se realizan iterativamente a pesar de estar especificadas en orden secuencial. Por ejemplo, los criterios de selección deben estar especificados previamente a la selección de los estudios primarios, pero puede suceder que el análisis de algunos de esos estudios promuevan la modificación de los criterios para que éstos propicien una discriminación más adecuada a los requisitos proyecto.
Las siguientes subsecciones describen concretamente las actividades anteriormente enlistadas. Más adelante, en la Sección 2.2 se exponen los resultados de haberlas ejecutado.

2.1.1.1. Formulación de las preguntas de investigación

El punto más importante de los contenidos en el protocolo es el relacionado con proponer las *preguntas de investigación*; esto es porque dichas preguntas guiarán prácticamente toda la ejecución del EMS el cual, una vez recolectada toda la evidencia, les deberá proporcionar una respuesta.
Kitchenham y Charters [13] proponen que las preguntas de investigación deben ser planteadas considerando los siguientes conceptos[1]:

- **Población (population):** puede estar conformada por cualquiera de los siguientes conceptos:
  - Un rol específico: desarrollador, arquitecto de software, jefe de proyecto, etc.
  - Áreas de aplicación concretas: sistemas de cuidado a la salud, sistemas de visualización, telecomunicaciones, etc.
  - Un grupo específico: compañías de telecomunicaciones, laboratorios de neurociencias, empresas de desarrollo de software, etc.

- **Intervención (intervention):** es la metodología, herramienta, tecnología o procedimiento que se emplea para abordar problemas concretos, como pueden ser bibliotecas de software, hardware, metodologías de desarrollo de software, etc.

- **Comparación (comparison):** es la metodología, herramienta, tecnología o procedimiento contra el cual se va a comparar la intervención. También se le conoce como el "control", que en áreas médicas se refiere al tratamiento control.

[1] Kitchenham y Charters proponen estos conceptos para una Revisión Sistemática de la Literatura (*Sistematic Literature Review*, RSL), y no para un EMS. Una RSL también es un proceso organizado para efectuar la revisión bibliográfica pero, a diferencia del EMS, requiere un análisis bastante profundo de cada estudio primario, incluyendo la validez en el diseño del estudio, diferentes tipos de sesgo existentes, generalidad de las conclusiones, etcétera. En consecuencia, una RSL constituye un proceso mucho más complejo y enfocado en un tema, requiriendo un esfuerzo mayor que un EMS. Es por ello que los EMS son útiles para generar un panorama general del estado del arte que permita hallar áreas de investigación en las que, posterirmente, se pudiera aplicar una RSL.

- **Resultado (outcome):** factor relevante que espera obtenerse de la intervención: mejor rendimiento, tiempos de entrega menores, costos reducidos, etc.

- **Contexto (context):** entorno en el cual se suscita la comparación, los participantes en el estudio y la magnitud de las tareas realizadas (por ejemplo, pequeña escala o gran escala).

- **Diseño experimental (experimental design):** forma en que se ha diseñado el experimento o método empleado para efectuar mediciones.

Estos conceptos provienen del área médica, en donde se busca evidencias empíricas en la práctica de la disciplina. Por lo tanto, es necesario adaptarlos al área en donde se pretenden aplicar (que en este caso son visualización 3D e ingeniería de software) y a los objetivos que se persiguen en el proyecto. En el caso del presente trabajo, es complicado pensar en algún diseño experimental buscando evidencias empíricas en herramientas o métodos de visualización en un contexto particular. Además, hallar estudios que hagan la comparación entre herramientas y métodos de visualización es deseable, pero lo más importante es encontrar estudios que indiquen cuáles son esos métodos y herramientas. En consecuencia, en la estructura de las preguntas de investigación se pueden omitir esos conceptos con el afán de encontrar cualquier evidencia reportada respecto a ellos.

2.1.1.2. Estrategia de búsqueda

Básicamente, la idea es automatizar la búsqueda de estudios primarios; para ello se utilizarán bibliotecas digitales que emplean motores de búsqueda. Estos motores utilizan cadenas de caracteres, también llamadas *cadenas de búsqueda*, introducidas por el usuario para seleccionar e incluso filtrar los resultados solicitados. La sintaxis a la que las cadenas de búsqueda deben apegarse varía de un motor a otro, además de que deberán estructurarse con base en los conceptos identificados en las preguntas de investigación. Por la tanto, es preciso documentar en el protocolo las razones detrás de la estructura de las cadenas, cualquier cambio que se suscite en éstas, y la elección de determinados buscadores. La estrategia de estructuración y empleo de las cadenas de búsqueda se describe y ejemplifica de forma muy completa en el reporte técnico de Beecham et al [15]. Básicamente proponen realizar las siguientes tareas:

1. Identificar los conceptos principales de cada pregunta de investigación (ver Sección 2.1.1.1).
2. Identificar todos los sinónimos para cada uno de los conceptos anteriores.
3. Considerar las palabras clave en las publicaciones relevantes con las que ya se cuente.
4. En los buscadores que lo permitan, emplear el operador lógico OR para unir los sinónimos de un concepto.
5. Emplear el operador lógico AND para unir conceptos principales.

Una vez que se han ensamblado las cadenas de búsqueda pueden introducirse en los motores de las bibliotecas digitales. Si el resultado es la obtención de cantidades desmesuradas (en el orden de miles) o muy pequeñas (cercanas a cero) de estudios entonces será necesario analizar y replantear los términos incluidos en la cadena de búsqueda empleada. En el caso de cantidades pequeñas también es posible que la cantidad de evidencia existente sea escasa.

2.1.1.3. Determinación de criterios de inclusión y exclusión

Una vez que se ha obtenido una primera lista de estudios primarios con la ayuda de los motores de búsqueda es momento de aplicar criterios de selección, es decir, *criterios de inclusión y exclusión*; el propósito es seleccionar aquéllos estudios que serán tomados en cuenta para la investigación. Dichos criterios deben estipularse inicialmente en el protocolo del EMS, aunque es muy posible que deban redefinirse en etapas subsecuentes de la ejecución. Es por ello que puede ser conveniente llevar a cabo pruebas piloto con algunos de los estudios primarios y así verificar que los criterios de selección los clasifican correctamente. Kitchenham y Charters [13] listan algunos criterios de selección prácticos que pueden ser empleados en la generalidad de estudios: idioma, revista de publicación, autores, participantes o sujetos de experimentación, diseño de la investigación, método de muestreo y fecha de publicación.

2.1.1.4. Selección de estudios primarios

Inicialmente puede hacerse una revisión rápida de las listas de estudios primarios arrojados por los buscadores pues será evidente que, con sólo leer el título o el resumen, algunos de ellos no estarán ni remotamente relacionadas con las preguntas de investigación. Esta tarea de "refinamiento grueso" reduce significativamente el número de estudios a los que luego deben aplicarse los criterios de selección anteriormente definidos. Si la revisión del título y del resumen no fuera suficiente para hacer una selección concreta entonces consultar las conclusiones puede ser de ayuda.
Una vez aplicados los criterios quedarán únicamente los estudios primarios más relevantes para el trabajo propio; el número de ellos que se obtenga para cada pregunta de investigación puede usarse para determinar:

1. Cuánto esfuerzo se ha realizado en el área de investigación.
2. Si hay o no suficiente apoyo bibliográfico.
3. Si hay oportunidades potenciales para extender la investigación.

2.1.1.5. Estrategia de extracción de los datos

La etapa de extracción de datos consiste en identificar en los estudios primarios la información necesaria para responder a las preguntas de investigación. Para ello es conveniente proponer algún método para registrar la información relevante, y dicho método también debe facilitar la clasificación de cada estudio, sentando así las bases para efectuar el mapeo del estado del arte. Por lo tanto, se determinó que una forma conveniente de registrar la información en los estudios primarios es clasificarlos mediante las *propiedades* que los hacen de interés para el proyecto. Estas propiedades constituyen aquéllos tópicos contenidos en los estudios primarios que dan una respuesta a las preguntas de investigación. La determinación de las propiedades puede efectuarse de forma iterativa como proponen Petersen et al [14] (aquí emplean el término *keywording* para referirse al proceso de crear el esquema de clasificación), revisando las palabras clave y el resumen de cada publicación de tal forma que pueda proponerse un esquema inicial de clasificación; éste irá adaptándose conforme el lector se familiariza con los diferentes contextos de investigación. Otra opción, que es la empleada en este proyecto, es conocer de antemano las propiedades que se pretende identificar en los

estudios primarios; un ejemplo de esta estrategia se presenta en el trabajo de Unterkalmsteiner et al [16]. En cualquier caso, la revisión de cada estudio primario no necesariamente debe ser exhaustiva para lograr clasificarlo adecuadamente.

2.1.1.6. Síntesis y presentación de los resultados

En lo relativo a la síntesis de los datos, el EMS busca exponer lo más clara y brevemente posible la clasificación hecha de los estudios primarios y la información que responde a las preguntas de investigación. En consecuencia es conveniente la inclusión de tablas, gráficas o esquemas que faciliten la comprensión de las conclusiones obtenidas y expongan claramente el panorama general del estado del arte. Una buena opción para esto son las gráficas de burbujas empleadas, por ejemplo, en los estudios presentados por Petersen et al. [14] y Fernández et al. [17].
Por otra parte, la estrategia para la presentación de los resultados conforma el último punto del protocolo. En este trabajo la presentación consistió en la exposición de las tablas y gráficas dentro de este documento. Asimismo, se incluyó una discusión de los estudios primarios con potencial impacto en el diseño del sistema de visualización 3D planteado.

2.2. Ejecución del EMS

Las características de la instancia del EMS llevado a cabo en este proyecto fueron expuestas a lo largo de la Sección 2.1; los resultados de su ejecución se exponen a continuación.

2.2.1. Preguntas de investigación

Las PIs consideradas relevantes para establecer un panorama general del estado del arte en sistemas de visualización 3D de imágenes médicas fueran las siguientes:

- PI 1 – *¿Cuáles son y en qué consisten los métodos de visualización tridimensional que tienen aplicación en neurociencias?*
  Se pretende conocer cuáles son y cómo operan los métodos de visualización 3D empleados en el ámbito de las neurociencias, con el propósito de esclarecer en qué situaciones un determinado método otorga ventajas respecto a los demás.

- PI 2 – *¿Qué herramientas se emplean en el desarrollo de sistemas de visualización tridimensional aplicables a las neurociencias?*
  Se pretende averiguar cuáles son las bibliotecas, marcos de trabajo e incluso paquetes de software comúnmente empleados para desarrollar programas de visualización tridimensional empleados en neurociencias.

- PI 3 – *¿Qué arquitecturas de software se han utilizado en el desarrollo de software de visualización tridimensional para propósitos médicos?*
  Se pretende conocer los detalles de las arquitecturas de software que más se emplean para modelar sistemas de visualización de imágenes médicas, sin limitarse necesariamente a las neurociencias.

ID	Población	Intervención	Resultado
PI 1	Neurociencias	Método de visualización tridimensional	(cualquiera)
PI 2	Desarrollo de software	Herramienta	Desarrollar sistemas de visualización tridimensional para neurociencias.
PI 3	Desarrollo de software	Arquitectura de software	Desarrollar sistemas de visualización tridimensional en medicina.

Cuadro 2.1: Conceptos principales para las preguntas de investigación.

El Cuadro 2.1 lista los tres conceptos principales considerados en el planteamiento de cada PI.

2.2.2. Estrategia de búsqueda

En el protocolo se estableció que la búsqueda de estudios primarios se efectuaría en bibliotecas digitales; éstas fueron las de IEEE Xplore, ACM-DL, Scopus y Springer Link. La primera comprende el motor de búsqueda del IEEE (*Institute of Electrical and Electronic Engineers*). El trabajo relacionado con temas de visualización y con ciencias de la computación es abundante entre los miembros dicha comunidad [18]. La ACM-DL es la biblioteca digital de la ACM (*Association for Computing Machinery*), que es una sociedad específicamente abocada al desarrollo de las ciencias de la computación (incluyendo visualización y computación gráfica) y que reúne el trabajo de profesionales laborando en ese ámbito desde fines de la década de los 40 [19]. Scopus [20] y Springer Link [21] comprenden bases de datos que brindan acceso a una gran cantidad de publicaciones en ciencia y tecnología y artes; varias revistas y conferencias en temas de visualización y computación también publican ahí sus estudios.
Las cadenas de búsqueda estipuladas en el protocolo del EMS, y que se introdujeron en los motores de búsqueda antes mencionados, fueron definidas con base en los conceptos establecidos para las PIs. Estas cadenas, así como los sinónimos empleados para estructurarlas, se presentan en el Apéndice A.

2.2.3. Selección de estudios primarios

La selección de los estudios primarios se determinó mediante la aplicación de criterios de selección, es decir, criterios inclusión y de exclusión. En el protocolo del EMS se establecieron los siguientes:

Criterios de inclusión

Se incluirán aquellas publicaciones cuyo título indique que responden a alguna de las preguntas de investigación. Concretamente, en la inclusión se considerarán los siguientes criterios:

CI1) El título establece que la publicación describe conceptual y/o teóricamente un método (o la optimización de un método) de visualización tridimensional, o bien su aplicación sobre información biomédica.
CI2) El título o el resumen señala que el trabajo evalúa y/o compara métodos o herramientas para la visualización tridimensional de imágenes médicas.
CI3) El título indica que se describirá la aplicación de un proceso de desarrollo, arquitectura de software, herramienta o estándar para generar sistemas de visualización tridimensional de información biomédica.

Cuando el título parezca sugerir que la publicación responde a una pregunta de investigación pero no esté claro si de forma determinante, se procede a leer el resumen (abstract) para luego aplicarle los criterios arriba mencionados.

Criterios de exclusión

Se excluirán de la revisión bibliográfica todas las publicaciones que no respondan a pregunta de investigación alguna. Específicamente, no se considerarán a aquéllas que:

CE1) Tanto título como resumen de la publicación no hacen referencia a algún método o herramienta de visualización tridimensional,
CE2) El título o el resumen mencionan métodos o herramientas de visualización tridimensional pero aplicadas a información no relacionada con medicina o ingeniería biomédica.
CE3) El título hace mención de métodos o herramientas de visualización tridimensional, pero el resumen sugiere que sólo juegan un papel secundario en el trabajo, por lo que se omite todo detalle de su aplicación.
CE4) La publicación es sólo un resumen, un cartel o una presentación de diapositivas.
CE5) La publicación no está escrita en español o en inglés.

Nótese que los criterios de inclusión y de exclusión únicamente involucran a la información contenida ya sea en los títulos o en el resumen de los estudios primarios; esto favoreció que su aplicación fuera más sencilla y, aún así, de utilidad para detectar aquéllos estudios que son potencialmente útiles para el proyecto. Los estudios se registraron empleando la herramienta de software conocida como JabRef v2.9.2, que es de código abierto, multiplataforma y que utiliza archivos en formato BibTeX para presentar y organizar listas de entradas bibliográficas [22].

2.2.4. Extracción de datos

La revisión de cada estudio primario se centró, principalmente, en el análisis de la metodología y herramientas empleadas, así como de las discusiones y conclusiones expuestas. Sólo se dio lectura cuidadosa a los resultados reportados cuando las herramientas y métodos pudieran influenciar el diseño del sistema propuesto en este proyecto.
La clasificación de los estudios primarios se realizó, inicialmente, por pregunta de investigación. Luego, esa clasificación se refinó etiquetando los estudios de acuerdo a las propiedades encontradas. Es decir, cada PI involucra a varias propiedades de interés, y con base en ellas también se clasificó a las publicaciones. La estrategia seguida es similar a la de Unterkalmsteiner et al. [16] en donde las propiedades de interés fueron establecidas previamente a la

revisión de los estudios primarios.
Se propusieron tres propiedades relacionadas con la PI 1. Éstas fueron planteadas para que involucraran a estudios que:

P1.1) Describen métodos de visualización 3D.
P1.2) Exponen aplicaciones biomédicas para métodos de visualización 3D.
P1.3) Efectúan comparaciones cualitativas y/o cuantitativas entre diferentes métodos.

Para la PI 2 se propusieron otras tres propiedades que relacionan a trabajos que:

P2.1) Emplean herramientas de software para efectuar tareas de visualización.
P2.2) Utilizan herramientas de hardware para efectuar tareas de visualización.
P2.3) Realizan comparaciones entre dichas herramientas.

Finalmente, puesto que la PI 3 está centrada en arquitecturas de software para sistemas de visualización 3D, las propiedades relacionadas se plantearon para identificar trabajos que:

P3.1) Hacen uso de patrones arquitectónicos y/o de diseño.
P3.2) Consideran atributos de calidad para el sistema.
P3.3) Siguen alguna metodología de desarrollo de software.

La lista de propiedades y sus respectivas descripciones pueden consultarse en el Cuadro 2.2. Para cada estudio primario revisado se anotaron las propiedades identificadas, los rubros de interés dentro de cada una así como observaciones pertinentes respecto a la misma empleando campos de BibTeX. Por ejemplo, todo trabajo que expusiera un uso interesante de algún método de visualización 3D sobre información biomédica fue marcado como poseedor de la propiedad P1.2, se anotó el nombre de dicho método y, si fuese relevante, alguna observación respecto a la aplicación del mismo. El propósito de esto fue apoyar la generación de conclusiones y la toma de decisiones para el proyecto una vez que la extracción de datos hubiese concluido.

2.2.5. Síntesis de los datos

El Cuadro 2.3 expone las cantidades de estudios primarios obtenidos para cada pregunta de investigación; en él se presentan tanto los arrojados inicialmente por cada biblioteca digital como aquéllos que resultaron luego de aplicar los criterios de inclusión y exclusión. El número de estudios seleccionados incluye a algunos que fueron arrojados por más de un motor de búsqueda, por lo que la parte inferior de la tabla indica tanto el número total de estudios seleccionados como el que resultó luego de eliminar los duplicados.
Una vez descontados los estudios duplicados se encontró que varios de los resultantes también se repetían entre preguntas de investigación. Tomando esto en cuenta, se obtuvo un total de 189 estudios primarios diferentes. De entre éstos se seleccionaron, para cada pregunta de investigación, aquéllos que mediante una rápida revisión del título y resumen parecieran aportar información relevante para por lo menos una de la propiedades. Hecho esto, se procedió a leerlos y clasificarlos con base en las propiedades expuestas en la Sección 2.2.4. Se revisó un total de 39 estudios primarios, y el Cuadro 2.4 expone cuántos y cuáles presentan propiedades de interés. Asimismo, las siguientes secciones presentan las aportaciones de estos estudios relacionadas con cada una de la propiedades encontradas.

	Propiedad	Descripción
PI 1	P1.1 Descripción de métodos de visualización tridimensional.	Esta propiedad incluye a todo estudio primario que describa un procedimiento o algoritmo de visualización 3D, o bien exponga una adaptación al mismo que impacte positivamente al desempeño.
	P1.2 Aplicación original de métodos de visualización tridimensional.	Incluye a los estudios primarios que apliquen métodos de visualización 3D de forma creativa y novedosa, con impacto potencial en aplicaciones biomédicas.
	P1.3 Evaluación cualitativa y/o cuantitativa de métodos de visualización tridimensional.	Considera a los estudios primarios que efectuaron pruebas sobre los métodos de visualización, con el propósito de observar y medir su impacto en algún atributo de calidad.
PI 2	P2.1 Empleo de bibliotecas, componentes o paquetes de software.	Los estudios primarios hacen uso ya sea de bibliotecas y componentes de software específicos, o de algún sistema completo para efectuar visualización de información biomédica.
	P2.2 Comparación cuantitativa entre bibliotecas, paquetes o componentes de software.	Incluye a los estudios primarios que, midiendo el impacto sobre ciertos atributos de calidad, realizan pruebas sobre las bibliotecas, componentes o paquetes de software.
	P2.3 Utilización de hardware gráfico especializado.	Los estudios primarios mencionan el uso de algún tipo de hardware para apoyar las tareas de visualización.
PI 3	P3.1 Empleo de patrones arquitectónicos o de diseño.	El estudio primario emplea uno o más patrones arquitectónicos y/o de diseño en el sistema para lograr atributos de calidad específicos.
	P3.2 Consideración a atributos de calidad.	Aquí se incluyen a los estudios primarios que reportan haber conseguido resultados favorables en algún atributo de calidad, ya sea mediante la aplicación de patrones arquitectónicos o de diseño, o considerando algún otro concepto de diseño.
	P3.3 Empleo de metodologías de desarrollo de software.	El estudio primario describe las actividades llevadas a cabo al seguir determinada metodología de desarrollo de software.

Cuadro 2.2: Lista de propiedades buscadas en los estudios primarios.

	Iniciales			Seleccionados		
	PI 1	PI 2	PI 3	PI 1	PI 2	PI 3
IEEEXplore	108	125	85	32	12	16
ACM-DL	69	12	57	19	5	13
Scopus	449	35	46	62	18	11
Springer	571	213	59	28	15	11
			Total	141	50	51
			Sin duplicados	130	44	44

Cuadro 2.3: Número de publicaciones a considerar inicialmente en el EMS. Las columnas bajo **Iniciales** indican los números obtenidos al insertar las cadenas de búsqueda, mientras que aquéllos bajo **Seleccionados** denotan cuántos quedaron luego de aplicar los criterios de inclusión y de exclusión.

Propiedad	No. publicaciones	Estudios primarios revisados
P1.1	10	[23, 24, 25, 26, 27, 28, 29, 30, 31, 32]
P1.2	17	[33, 34, 24, 25, 27, 29, 30, 31, 35, 36, 37, 38, 39, 40, 41, 42, 43]
P1.3	6	[23, 25, 38, 29, 39, 40]
P2.1	25	[33, 44, 23, 24, 25, 32, 36, 45, 46, 47, 48, 49, 50, 51, 52, 53, 38, 54, 55, 56, 57, 58, 59, 60, 61]
P2.2	1	[45]
P2.3	14	[45, 23, 47, 36, 50, 24, 38, 27, 33, 44, 39, 59, 61, 32]
P3.1	14	[45, 51, 59, 55, 36, 24, 58, 56, 46, 52, 47, 48, 33, 44]
P3.2	13	[45, 59, 23, 36, 58, 56, 46, 52, 47, 57, 48, 49, 50]
P3.3	0	

Cuadro 2.4: Clasificación de las publicaciones revisadas con base en propiedades.

2.3. Resumen de los estudios primarios

Para complementar la presentación de los datos, en esta sección se comentan y discuten los estudios primarios que exhibieron potencial impacto en el diseño del sistema de visualización 3D planteado para este proyecto.

2.3.1. Métodos de visualización tridimensional aplicados en neurociencias

Al dar respuesta a la PI 1 se pretende averiguar qué métodos de visualización tridimensional existen, establecer las ventajas y desventajas generales de esos métodos, y determinar cómo se emplean y desempeñan en aplicaciones médicas, específicamente en neurociencias. De las 39 publicaciones revisadas, 21 tuvieron alguna propiedad relacionada con la PI 1, y el Cuadro 2.4 indica cuántas de esas publicaciones corresponden a cada propiedad. Una vez revisadas fue posible llegar a conclusiones interesantes respecto a los métodos de visualización 3D.

2.3.1.1. Descripción de métodos de visualización tridimensional (P1.1)

Tradicionalmente, dos técnicas principales se han empleado para efectuar visualización 3D de información biomédica, y son representación por superficie (*surface rendering*) y representación por volumen (*volume rendering*, también denominada *direct volume rendering*). Ambas permiten visualizar información biomédica mediante una proyección bidimensional generada a partir de datos tridimensionales, pero los algoritmos que emplean son muy diferentes y cada uno tiene sus ventajas y desventajas [2].
Para efectuar la representación por superficie es muy recomendable aplicar previamente alguna técnica de segmentación o de extracción de contornos en todas las imágenes del conjunto a visualizar; luego los contornos de imágenes adyacentes son ligados mediante polígonos (triángulos por lo general) dando como resultado una representación "en mosaico" de las estructuras tridimensionales. Un claro ejemplo de este proceso se da en el trabajo de Benhamou y Clara [32], en el que se reconstruyen árboles neuronales mediante un método conocido como triangulación Delaunay. En lo que respecta a imágenes del cerebro, más recientemente Bischoff y Kobbelt han hecho propuestas de preprocesamiento empleando operaciones morfológicas, permitiendo que la reconstrucción por superficie arroje resultados más fidedignos respecto a la estructura cerebral real [31]. Con todo lo anterior, la representación por superficie se caracteriza por la cantidad reducida de información que requiere, logrando tiempos de visualización relativamente cortos.
Por su parte, la representación en volumen no requiere de pasos previos de segmentación para llevarse a cabo y otorga resultados visuales muy atractivos, aunque los algoritmos empleados son computacionalmente costosos. Existen varias técnicas consideradas como de representación por volumen y muchas de ellas se basan en el algoritmo de *ray-casting* [62, 63]. Una gran cantidad de métodos de optimización para el ray-casting (no relacionados con el uso de hardware gráfico o de cómputo distribuido) han sido publicados. El trabajo Mueller y Kaufman [64] presenta, en añadidura al panorama general del algoritmo de *ray-casting*, una breve descripción de algunos de esos métodos de optimización. Algunos de ellos como los

denominados “bricking” y “ray-leaping” se presentan en el trabajo de Parker et al. [29], los cuales se emplean para omitir espacios sin información relevante (espacio vacío, por ejemplo) al momento de efectuar la representación. Asimismo, el ray- casting se caracteriza por ser un algoritmo muy flexible puesto que sus parámetros pueden ser definidos para resaltar u omitir determinada información. Por ejemplo, Wenger et al. [30] proponen adaptaciones al algoritmo que permiten resaltar estructuras delgadas en forma de hilo con gran nivel de detalle y realismo; para ello emplean información de tractografía obtenida mediante IRM. Otro caso interesante se describe en el trabajo de Beyer et al. [39] utilizando imágenes de tomografía computada e IRM; en él se estipulan varias condiciones para que el algoritmo de ray-casting omita estructuras de piel y hueso con apoyo de las imágenes de tomografía computada, de este modo logrando visualizar únicamente el cerebro empleando las imágenes de IRM sin necesidad de efectuar segmentación previa.
En resumen, la elección de una u otra técnica de visualización dependerá del tipo de información con la que se cuente y de lo que se quiera representar de ella. Incluso en ocasiones es conveniente emplearlas en conjunto, logrando con ello resultados muy útiles e interesantes, como los que se exponen a continuación.

2.3.1.2. Aplicación original de métodos de visualización tridimensional (P1.2)

Wan et al. [38] exponen una muy interesante aplicación de representación por volumen y por superficie en conjunto, empleando diversos tipos de datos obtenidos por microscopía. Sus propuestas permiten poner en pantalla diferentes tipos de información sin que una entorpezca la visualización de las otras, así como obtener un alto grado de realismo en sombreado y en discernimiento de la distancia de los objetos representados. Por su parte, Jainek et al. [27] proponen una estrategia para visualizar eficientemente y de manera conjunta volúmenes de IRM e IRMf, empleando representación por superficie para pintar una carcasa semitransparente de la corteza cerebral, y representación por volumen para visualizar la información funcional. Emplean imágenes anatómicas en las que únicamente se han segmentado materia gris y materia blanca, pero la estrategia de visualización es escalable a un número mayor de estructuras. Beyer et al. [39] extienden su estrategia a información multimodal, pudiendo visualizar conjuntamente información de IRM, IRMf, tomografía computada, tomografía por emisión de positrones y angiografía digital mediante representación por volumen. También proponen un algoritmo para efectuar la remoción del cráneo “al vuelo” al momento de efectuar el ray-casting. Como último ejemplo, en el trabajo de Jeong et al. [24, 36] aplican representación por volumen a información de microscopía con el objetivo de reconstruir y visualizar cuerpos neuronales. Resulta particularmente interesante que los datos que emplean miden varios terabytes y, a pesar de ello, obtienen tasas de interacción razonables para aplicaciones no críticas.
Cabe mencionar que recientemente han cobrado importancia las aplicaciones que, más allá de simplemente desplegar imágenes en pantalla, buscan mapear la información biomédica al mundo real. El término “realidad virtual aumentada” (*augmented reality*) es empleado para denotar a este proceso [23, 35]. Uno de sus propósitos principales es apoyar el entrenamiento, planeación quirúrgica y diagnóstico, y también brindar apoyo pre e intraoperatorio al médico cirujano mediante la representación de estructuras ocultas o que son difíciles de percibir a simple vista, minimizando así el riesgo para el paciente. Básicamente estos sistemas em-

plean cámaras para efectuar detección y seguimiento de puntos fiduciarios, y luego alinean la representación tridimensional (efectuada mediante alguno de los métodos ya descritos) a las coordenadas del mundo real. En ocasiones el uso de interfaces táctiles acompaña a este tipo de sistemas [23], añadiendo una retroalimentación motriz que es especialmente útil en sistemas de entrenamiento [44].
En resumen, las dos principales técnicas de representación tridimensional, que son representación por volumen y por superficie, tienen una amplia variedad de aplicaciones en el entorno médico y académico; se emplean casi en igual medida en sistemas de visualización. Nuevamente, todo dependerá de aquellas estructuras o mediciones que se quieran visualizar de la información volumétrica. En lo que respecta al presente proyecto, es de interés conocer los trabajos que llevaron a cabo visualización de información multimodal (i.e varias modalidades de imagen visualizadas de forma conjunta). Sólo se encontraron algunos trabajos reportados que exponen estrategias para efectuar este tipo de representación (véase [27, 38, 39]); no obstante, el trabajo relacionado que en ellos se reporta indica que, en efecto, ya hay esfuerzos considerables para representar información multimodal de forma simultánea.

2.3.1.3. Evaluación cualitativa y/o cuantitativa de métodos de visualización tridimensional (P1.3)

Se encontraron pocos estudios primarios que efectuaran algún tipo de evaluación sobre los métodos de visualización. Entre los más relevantes que se revisaron está el trabajo de Parker et al. [29] que hace una evaluación cuantitativa del desempeño logrado al efectuar representación por volumen, empleando el algoritmo de ray-casting con y sin métodos de optimización. De particular interés es su medición de los tiempos de procesamiento logrados al utilizar desde 1 hasta 128 procesadores para la representación; obtuvieron que la mejora de rendimiento deja de ser lineal cuando se emplean más de 64 de ellos. Un trabajo parecido es presentado por Hachaj y Ogiela [23] en el cual también comparan el desempeño del algoritmo de ray-casting empleando técnicas de ray-leaping, así como el uso de una técnica conocida como mapeado de texturas que favorece la generación de proyecciones de apariencia muy realista. Por su parte, el trabajo de Beyer et al. [39] hace el análisis del desempeño obtenido con la representación de volumen efectuada sobre varios conjuntos de imágenes biomédicas registradas y obtenidas de distintas modalidades de imagen, así como de su adaptación del algoritmo de ray-casting para remover el cráneo de la proyección final. Finalmente, el trabajo de Wan et al. [38] lleva a cabo una comparación cualitativa de los resultados visuales obtenidos empleando diferentes modelos de sombreado en imágenes de microscopía; emplean los métodos de representación por volumen y por superficie de manera conjunta sobre varios volúmenes, y documentan los resultados visuales obtenidos al alternar el orden en que dichos volúmenes son procesados y dibujados en pantalla, evaluando así las ventajas y desventajas de cada proyección generada.

2.3.2. Herramientas para la visualización tridimensional

Al responder a la PI 2 se busca conocer, a través de las propiedades definidas, qué bibliotecas se emplean comúnmente en el desarrollo de sistemas de visualización, qué sistemas ya desarrollados existen y cuál es el hardware que emplean. Las propiedades expuestas en el Cuadro 2.2 reflejan estas interrogantes. De las 39 publicaciones revisadas, 27 tuvieron algu-

Herramientas	Tipo	No. publicaciones	Estudios primarios revisados
VTK	Biblioteca	12	[47, 48, 49, 51, 52, 53, 54, 33, 44, 56, 57, 58]
ITK	Biblioteca	6	[47, 48, 52, 33, 56, 58]
Qt	Biblioteca	6	[48, 36, 51, 44, 57, 58]
OpenGL	Biblioteca	6	[45, 36, 50, 25, 38, 60]
CUDA	Biblioteca	4	[47, 36, 50, 24]
Otras bibliotecas	Biblioteca	9	[61, 32, 59, 46, 53, 57, 33, 60, 49]
NVIDIA	Hardware	10	[45, 23, 47, 36, 50, 24, 38, 33, 44, 59]
AMD	Hardware	2	[27, 39]
Silicon Graphics	Hardware	2	[32, 61]

Cuadro 2.5: Resumen de herramientas de software empleadas en sistemas visualización

na propiedad relacionada con la PI 2, y el Cuadro 2.4 indica cuántas de esas publicaciones corresponden a cada propiedad.

2.3.2.1. Herramientas de hardware (P2.3)

La propiedad P2.3 está relacionada con estudios primarios que hagan uso de hardware gráfico; el Cuadro 2.5 indica cuántos y cuáles de ellos contienen esta propiedad. Tres tipos de hardware gráfico marcaron una tendencia: NVIDIA, AMD y Silicon Graphics. Los resultados del EMS indican claramente que las aplicaciones biomédicas se han beneficiado de estas herramientas. En el pasado los sistemas empleados para visualizar información biomédica podían tardar días e incluso meses para completar una representación tridimensional [61, 32]. Sin embargo, muchos sistemas actuales resuelven el problema del desempeño empleando GPUs (*Graphics Processing Units*), convirtiendo a la visualización tridimensional en una alternativa muy atractiva y cada vez más viable para apoyar diferentes actividades tanto clínicas como académicas. Actualmente una GPU es un procesador multinúcleo que contiene cientos e incluso miles de unidades lógicas de aritmética (comúnmente conocidas como ALU) que en conjunto son capaces efectuar operaciones de punto flotante de manera muy eficiente, lo que las hace potencialmente prácticas para muchas aplicaciones. A inicios de la década pasada las GPUs únicamente podían efectuar cálculos para representar gráficos en pantalla, empleando interfaces de programación como OpenGL (ésta se describe más adelante) y DirectX[2] [65]. Sin embargo, su gran desempeño para efectuar cálculos numéricos despertó interés en la comunidad científica, dando como resultado un cambio en la arquitectura electrónica de las GPUs encaminado al cómputo general.
La mayoría de los estudios primarios revisados que emplearon algún hardware gráfico resultó proveniente de NVIDIA. Ésta es una compañía especializada en la manufactura de unidades gráficas de procesamiento o GPUs, así como el desarrollo de herramientas para cómputo paralelo con propósitos científicos. Entre los trabajos más interesantes que se han beneficiado del desempeño aportado por hardware de NVIDIA cabe mencionar el de Wan et al. [38], en el cual obtienen resultados visuales muy atractivos y en el que declaran estar

[2]DirectX es un conjunto de interfaces de programación liberado por Microsoft para manejo y procesamiento de información multimedia; es exclusivo para el sistema operativo Windows.

satisfechos con los tiempos de procesamiento para visualizar estructuras 3D en imágenes de microscopía. Una aplicación similar que hace uso intensivo de los recursos de una GPU de NVIDIA se detalla en los trabajos de Jeong et al. [24, 36], en donde efectúan segmentación y representación 3D de axones de neuronas mediante imágenes de microscopía. También obtienen tiempos de procesamientos relativamente cortos considerando el gran tamaño (orden de gigabytes e incluso terabytes) que pueden tener los conjuntos de imágenes que emplean. Asimismo, un trabajo que hace uso de estas GPU's y que puede servir como referencia a todo el interesado en el ray-casting es el de Hachaj y Ogiela [23], en el cual se describe el desempeño logrado con implementaciones de algunas variantes de dicho algoritmo aplicado a volúmenes adquiridos por tomografía computada. Por su parte, el trabajo de Zhang et al. [50], que también emplea GPU's de NVIDIA, reporta reducciones de por lo menos un orden de magnitud en los tiempos de procesamiento de señales de electromiografía (señales eléctricas musculares) empleadas para controlar el sistema de realidad virtual que proponen, comparado con lo que tomaría hacer el trabajo en un solo procesador.
Los estudios primarios que hacen uso de hardware gráfico de AMD también indican haberse beneficiado de él, aunque únicamente se encontraron dos de ellos; estos son los trabajos de Beyer et al. [39] y de Jainek et al. [27], cuyas propuestas de visualización conjunta de información multimodal reportan buenas tasas de interacción con las representaciones 3D. Por su parte, los estudios que emplearon tecnología Silicon Graphics son trabajos más antiguos que presentan propuestas de visualización que requerían varios días para llevarse a cabo. Los trabajos de Benhamou y Clara [32] y de Carlbom et al. [61] exponen los resultados obtenidos hace más de veinte años empleando estos sistemas.

2.3.2.2. Herramientas de software (P2.1 y P2.2)

La propiedad P2.1 involucra a estudios primarios que empleen alguna biblioteca o marco de trabajo de software para efectuar visualización 3D; el Cuadro 2.5 indica cuántas publicaciones emplearon una o más de estas herramientas. Por otra parte, como muestra el Cuadro 2.4 en la fila de la propiedad P2.2, sólo se encontró un estudio que comparara dos o más herramientas para el desarrollo de sistemas de visualización. Las herramientas de software más empleadas en los estudios primarios revisados son VTK, ITK, OpenGL, Qt y CUDA.
VTK (*Visualization Toolkit*) es una biblioteca gratuita y de código abierto dedicada a la visualización y procesamiento de imágenes en 2D y 3D; está escrita en C++ pero también cuenta con bibliotecas de interfaz (*wrappers*) escritas en Java, Tcl/Tk y Python [66]. Además, hace uso de funciones de OpenGL para aprovechar las capacidades de hardware gráfico. Ofrece funcionalidad para procesamiento de escalares, vectores, texturas y métodos de visualización volumétrica. Además de contar con sus propias herramientas gráficas para la interacción también provee interfaces para integrar otros ambientes gráficos, como el de Qt (éste marco de trabajo se describe más adelante). Sin embargo, una de las características más sobresalientes de VTK como biblioteca de software es su diseño, caracterizado por otorgar modificabilidad y portabilidad. Dicho diseño se basa en un modelo de visualización muy completo en el que se pueden representar datos 2D y 3D de cualquier tipo, y en el que se pueden definir objetos que los lean y transformen para finalmente representarlos en pantalla. Dichas transformaciones pueden efectuarse secuencialmente siguiendo un estilo "pipe-and-filter" en el que la información de salida de un objeto que procesa datos (llamados filtros)

puede convertirse en la entrada de otro [67].
VTK es la biblioteca de software más empleada en los estudios primarios revisados; estos describen una gran variedad de aplicaciones que emplean diferentes tecnologías en conjunto. Por ejemplo, Radau et al. [47] presentan un sistema para visualización 4D (representaciones 3D presentadas a lo largo del tiempo) de información cardiaca en tiempo real con VTK, y que puede adquirir conjuntos de imágenes directamente de un equipo de IMR. Otra aplicación interesante es presentada en el trabajo de Buzurovic et al. [51] que describe un sistema para braquiterapia semi-automatizada empleando brazos robóticos. La inserción de las agujas durante el procedimiento es guiada por representaciones 3D de imágenes de ultrasonido generadas con VTK. Por su parte, Mahmoudi et al. [52] describen el desarrollo de una aplicación web para visualización 2D y 3D de imágenes médicas. Esta aplicación emplea VTK para efectuar representaciones 3D que pueden ser visualizadas remotamente mediante un navegador web. Tahmasebi et al. [44] detallan el diseño de un sistema con interfaz háptica para entrenamiento en estudios de ultrasonido. Aquí, una reconstrucción 3D hecha con ayuda de VTK es proyectada en un monitor y se utiliza como modelo de un paciente al cual el dispositivo háptico (que consiste en una sonda de ultrasonido de prueba) debe apegarse, lo cual otorga la sensación de estar escaneando a un paciente real. Finalmente, vale la pena mencionar el trabajo efectuado por Shen et al. [33] en el que hacen uso de VTK en conjunto con ITK para efectuar representaciones 3D de imágenes médicas en un ambiente de realidad virtual. La idea consiste en que un usuario pueda manipular las representaciones desplegadas en el entorno de proyección empleando varios dispositivos de detección de movimiento. Dicho entorno consiste en varias pantallas de gran tamaño en medio de las cuales el usuario se posiciona para obtener la sensación de estar "inmerso" en las estructuras visualizadas. newline Por otra parte, ITK (*Insight Segmentation and Registration Toolkit*) también es una biblioteca de software gratuita y de código abierto que ofrece funcionalidad para efectuar análisis y procesamiento de imágenes enfocados en registro y segmentación. Al igual que VTK, está implementada en C++ y cuenta con bibliotecas de interfaz (*wrappers*) escritas en Java y Python [68]. Desafortunadamente, entre los estudios primarios revisados que mencionan haber usado ITK sólo hubo dos que mencionan explícitamente con qué fines lo hicieron o cómo lo incorporaron al diseño. El trabajo de Rey et al. [48] menciona haberla usado para segmentación de los pulmones en imágenes de tomografía de tórax. Asimismo, Mahmoudi et al. [52] exponen haberlos usado para fines de segmentación y registro de imágenes cerebrales. Aunque se encontraron pocos estudios que la empleen, ITK no es una librería que sea indispensable dentro del diseño del sistema propuesto en este proyecto dado que, como se verá más adelante, el procesamiento de imágenes no figura entre los requerimientos para el ciclo de desarrollo actual.
OpenGL (*Open Graphics Library*) es un estándar propuesto por la compañía Silicon Graphics, con base en el cual definió una interfaz de programación (API) en un esfuerzo por promover la implementación de aplicaciones que empleen gráficos 2D y 3D. El EMS arrojó algunos estudios primarios que presentan aplicaciones interesantes con OpenGL, en ocasiones también empleando otras herramientas de visualización en conjunto. Acosta et al. [45] presentan el diseño de un marco de trabajo para ambientes virtuales (estos son aplicaciones de simulación como los que se emplean, por ejemplo, para entrenamiento de personal médico. La idea es emular con cierto realismo el comportamiento físico de los objetos que se visualizan) que emplea tecnología OpenGL; incluso presentan un caso de estudio en el que desarrollaron un

sistema para simulación de craniotomía. Zhang et al [50] describen la implementación de una interfaz neuromuscular para el control de una prótesis de pierna; parte de su proyecto incluye la creación de un entorno de realidad virtual con OpenGL en el que se despliegua un avatar tridimensional que responde a la actividad muscular registrada por la prótesis. Por otro lado, en su propuesta de solución para visualizar cuerpos neuronales, Jeong et al. [36] emplean OpenGL para dibujar el resultado final en la pantalla. En general, OpenGL es una opción viable para fines de este trabajo, aunque otras bibliotecas de software como VTK ya hacen uso de OpenGL para efectuar visualización.

Compañías como NVIDIA han puesto a disposición del público interfaces de programación capaces de aprovechar los recursos ofrecidos por sus GPUs. El nombre de CUDA (un acrónimo para *Compute Unified Device Architecture*) fue escogido para denominar la arquitectura de sus dispositivos, y su interfaz de aplicación (que lleva el mismo nombre) está disponible como extensiones de los lenguajes C/C++ y Fortran. La diferencia principal entre CUDA y las bibliotecas de software presentadas hasta este punto es que CUDA sirve para propósitos de cómputo en general y, por lo tanto, no está enfocada exclusivamente en visualización. Puesto que antes del advenimiento de las GPUs para cómputo general la única forma de emplear estos recursos era mediante APIs como OpenGL y DirectX, la arquitectura CUDA es la solución provista por NVIDIA para aprovechar el gran potencial de cómputo en sus GPUs [65]. Desde entonces este tipo de hardware ha visto una multiplicidad de aplicaciones, algunas de las cuales se han dirigido a resolver problemas de visualización. Respecto a esto, algunos estudios primarios encontrados mediante el EMS emplean CUDA en los sistemas de visualización que proponen. El trabajo de Jeong et al. [24, 36] hacen pleno uso de la arquitectura multinúcleo de una GPU de NVIDIA para remover ruido y ejecutar el algoritmo de ray-casting sobre conjuntos de imágenes obtenidas por microscopía. A pesar de la inmensa cantidad de información que puede estar contenida en estos conjuntos, los autores reportan obtener buenas tasas de interacción al momento de efectuar la representación 3D. Por su parte, Zhang et al. [50] presentan un uso distinto al de visualización y emplean CUDA para paralelizar la ejecución de sus algoritmos de reconocimiento de patrones aplicados a señales de electromiografía, con lo cual obtuvieron mejoras en el desempeño de un orden de magnitud en los tiempos de procesamiento.

Por otro lado, Qt es un marco de trabajo multiplataforma escrito en C++ para aplicaciones en general, integrando herramientas para interfaces gráficas, información multimedia, comunicación por red, entre otras. Básicamente, el papel que Qt juega en los estudios primarios revisados, y que se listan en el Cuadro 2.4, es en el diseño de la interfaz gráfica y en la recepción de señales de dispositivos externos como ratón o teclado.

El Cuadro 2.5 indica que hay varios estudios primarios que hicieron uso de otras librerías diferentes a las ya descritas. Por ejemplo, el trabajo de Lesage y Raffin [46] emplean un marco de trabajo para aplicaciones interactivas denominado FlowVR. Éste consiste en una serie de módulos genéricos que intercambian información entre ellos siguiendo un estilo de flujo de datos secuencial, es decir, la salida de uno puede constituir la entrada de otro, análogamente a como funcionan los filtros de VTK. Este mismo estudio emplea MPI, que es un sistema de paso de mensajes para cómputo en paralelo, para comunicar módulos de FlowVR que pueden estar ubicados en nodos diferentes. Por su parte, el sistema de visualización 3D propuesto por Engel et al. [59] hacen uso de la funcionalidad provista por Java2D y Java3D, que son APIs de Java empleadas para manipulación de gráficos; Java3D recurre a la funcionalidad

provista por OpenGL si el sistema cuenta con dicha biblioteca. El trabajo de Maurizi et al. [53] emplea módulos de ImageJ para efectuar visualización de imágenes médicas en 2D. ImageJ es un software de dominio público para visualizar y procesar imágenes en general que fue desarrollado en los Institutos Nacionales de Salud, y cuenta con extensiones capaces de efectuar visualización tridimensional de imágenes.
Concluyendo, de acuerdo con los estudios primarios arrojados por el EMS, VTK es la biblioteca de software más popular para efectuar representaciones 3D. Los sistemas que empleen VTK se verán muy favorecidos en su desempeño si cuentan con una GPU, dado que emplea funciones de OpenGL para efectuar varias tareas de visualización. Otro atributo importante de VTK es su diseño, el cual se basa en un modelo de visualización muy general que permite el manejo de muchos tipos de datos, y que también permite definir objetos de procesamiento de datos (filtros) que satisfagan necesidades particulares de visualización. Por su parte, Qt parece ser una buena opción para la interfaz gráfica dado que varios trabajos lo consideraron en el diseño de sus sistemas. ITK también se usó con frecuencia en los trabajos pero, siendo una biblioteca empleada para registro y segmentación de imágenes, es probable que se emplee en trabajo a futuro para fines de este proyecto.

2.3.3. Arquitectura de software en sistemas de visualización tridimensional aplicada a neurociencias

Existen varias definiciones de arquitectura de software, pero aquí se considerará la otorgada por Bass, Clements y Kazman [3] que estipula lo siguiente:

> "La arquitectura de software de un sistema es el conjunto de estructuras necesarias para *razonar sobre el sistema*, considerando elementos de software, las relaciones entre ellos, y las propiedades de ambos."

El razonamiento sobre el sistema al que esta definición alude está relacionado con vislumbrar, analizar y evaluar el impacto que el diseño de la arquitectura tendrá sobre la calidad del sistema. En consecuencia, ese conjunto de estructuras debe constituir la manifestación de las tácticas encaminadas a que el sistema cumpla con atributos de calidad, y por lo tanto toda estructura que permita un análisis y evaluación de la calidad es considerada arquitectónica. Un *atributo de calidad* es una propiedad medible y comprobable de un sistema que indica qué tan bien satisface las necesidades de los interesados; en otras palabras, los atributos de calidad denotan qué tan bien hace algo el sistema, y se manifiestan en situaciones como "qué tan rápido procesa datos" (desempeño), "qué tan amigable es la interfaz gráfica" (usabilidad), "qué tan fácil es añadir nueva funcionalidad" (modificabilidad), entre muchas otras. Es verdad que, como se sugirió en la Sección 2.3.2, los atributos de calidad de un sistema dependen también de las herramientas empleadas para diseñar, implementar y poner en operación al sistema. Sin embargo, es la arquitectura de software lo que inherentemente abre las posibilidades a un sistema para contener atributos de calidad. Puesto que en el objetivo del presente proyecto se plantea cumplir con requerimientos de calidad, se consideró necesario contemplar en la búsqueda bibliográfica la manera que proponen otros autores de abordar, desde una perspectiva de arquitectura de software, esos requerimientos en sistemas de visualización biomédica.

Considerando lo anterior, la PI 3 aborda el tema de arquitectura de software aplicada a sistemas médicos de visualización. A diferencia de las preguntas anteriores, el alcance de ésta no fue acotado a aplicaciones en neurociencias; el objetivo es obtener un panorama de las arquitecturas de software mejor posicionadas en sistemas de visualización en general. Concretamente, se trata de averiguar qué patrones arquitectónicos son empleados en sistemas de visualización, y cómo se adaptan y combinan para obtener atributos de calidad. Las propiedades expuestas en el Cuadro 2.2 exponen las interrogantes anteriores. De los 39 estudios primarios revisados 18 contienen alguna de esas propiedades, y el Cuadro 2.4 clasifica a las publicaciones por propiedad. No se encontraron estudios que contuvieran a la propiedad P3.3, es decir, estudios que empleasen alguna metodología de desarrollo de software.

2.3.3.1. Uso de patrones y atributos de calidad reportados (P3.1 y P3.2)

La propiedad P3.1 está relacionada con trabajos que hagan uso de patrones arquitectónicos o de diseño. La definición formal de patrón arquitectónico se expondrá más adelante en la Sección 3.1.1.2; por el momento basta enfatizar que se trata de una solución bien conocida a un problema recurrente al diseñar arquitectura, y que otorga atributos de calidad al sistema. Los patrones de diseño también son soluciones conocidas, pero aplicadas a clases y objetos que se comunican entre ellos y que, una vez que se hayan personalizado adecuadamente, resuelven problemas de más bajo nivel en el diseño del sistema. El Cuadro 2.6 lista los trabajos que emplean de uno o más patrones arquitectónicos o de diseño. Nótese que los patrones más empleados fueron los de capas y cliente-servidor, y en menor medida se encontraron estudios primarios que recurrieron a estilos como el N-Tier, broker, publish-subscribe y pipe-and-filter, así como unos pocos que aplicaron algún patrón de diseño. Por otra parte, la propiedad P3.2 está contenida en trabajos que reportaron haber obtenido resultados favorables respecto a algún atributo de calidad; estos trabajos se listan en el Cuadro 2.7. Aunque cada patrón arquitectónico inherentemente propicia determinados atributos de calidad, otras decisiones de diseño, como el uso de determinadas herramientas, también son importantes para otorgar dichos atributos a un sistema; en la Sección 2.3.2 se discutieron las principales herramientas indicadas por los resultados del EMS. Por todo lo anterior, los párrafos siguientes exponen cómo algunos de los estudios primarios revisados emplearon patrones y herramientas para la obtención de atributos de calidad.

El patrón arquitectónico de capas es uno de los más empleados en arquitectura de software [3]. Básicamente consiste en dividir al sistema o parte de él en agrupaciones (capas) que conjuntan a unidades de implementación (clases, por ejemplo); estas unidades se caracterizan por ofrecer un servicio en común. En consecuencia, las arquitecturas organizadas en capas suelen promover la modificabilidad y la portabilidad. Por ejemplo, el sistema de realidad virtual propuesto por Shen et al. [33] estructura su diseño en tres capas: interfaz de usuario, lógica de negocio (que ellos denominan kernel) y visualización; cada capa cumple funciones específicas y, concretamente, la capa de interfaz de usuario sólo hace uso de la de lógica de negocio, y ésta a su vez sólo emplea la de visualización. La interfaz de usuario contiene todas las clases para operar desplegar y recibir información de una computadora personal, y del ambiente de realidad virtual; la lógica de negocio se encarga de coordenar la secuencia de acciones necesarias para efectuar tareas relacionadas con visualización; y la capa de visualización encapsula las clases que se encargan de efectuar el procesamiento de la información

médica. Con este diseño los autores logran un buen grado de modificabilidad ya que pueden añadir y modificar funcionalidad desde la capa de lógica de negocio, la cual puede solicitar el servicio de la capa de visualización cuando lo requiera. Se reporta que esta última capa hace uso de bibliotecas de software como VTK e ITK, lo que además favorece al desempeño del sistema. Un ejemplo similar se da en el trabajo de Mahmoudi et al. [52], el cual presenta el diseño para una aplicación web organizado en cuatro capas: interfaz gráfica, comunicación, código gestionado y algoritmos. Las dos primeras capas se emplean para la comunicación cliente-servidor, la capa de código gestionado funciona como "wrapper" para código C++ (dado que parte del código para las páginas web lo escribieron en ASP.NET), y la capa de algoritmos encapsula todo el procesamiento para generar proyecciones 3D empleando VTK, ITK y VRML. Además de la modificabilidad proporcionada por la organización en capas, una aplicación web de este tipo destaca por la disponibilidad; sólo se requiere un navegador web con compatibilidad para VRML para poder efectuar tareas de visualización 3D.

Algunos de los estudios primarios revisados presentan sistemas de estilo cliente-servidor. Como el nombre indica, este estilo de arquitectura maneja dos tipos de componentes (un componente es un elemento de software que existe en tiempo de ejecución del sistema. Los componentes se introducen de forma más completa en la Sección 3.1). El cliente es el solicitante de servicios, y el que los provee es el servidor; en algunos casos los clientes pueden fungir como servidores, o viceversa [3]. Los sistemas que se diseñan al estilo cliente-servidor promueven la modificabilidad (pues todos los servicios comunes están contenidos en el servidor), la escalabilidad y disponibilidad (dado que siempre pueden agregarse nuevos servidores si hiciera falta). El sistema propuesto por Jeong et al. [36] promueve en gran medida el desempeño y la disponibilidad, incluso al tratar con conjuntos de imágenes en el orden de gigabytes y terabytes. El servidor divide cada conjunto en bloques más pequeños y genera versiones de baja resolución de cada uno. De este modo, los clientes que soliciten acceso a las imágenes trabajan con estos bloques que son más fáciles de manipular y cuya transmisión a través de la red es mucho más rápida. La transmisión de datos se lleva a cabo a través de sockets. Por su parte, Gustafson et al. [55] proponen un sistema cliente-servidor para visualización de conjuntos de imágenes médicas en general. El servidor puede efectuar cortes arbitrarios en las imágenes a petición de los clientes, de tal forma que sólo se visualicen ciertas porciones de la información volumétrica. Es decir, el servidor efectúa el procesamiento necesario para simular estos cortes y envía el resultado al cliente para que éste lo despliegue. Además, los autores indican que la transmisión de datos se efectúa mediante sockets que emplean el protocolo TCP/IP, el cual garantiza que todos los datos sean entregados en el orden correcto. Un enfoque similar es adoptado por Engel et al. [59] en su sistema para visualización de imágenes médicas en 2D y 3D. Su propuesta para lo que ellos llaman una "visualización híbridaresulta muy interesante; ésta consiste en que el cliente debe trabajar con una representación tridimensional de baja resolución para luego pasar los parámetros de ésta a un servidor de aplicación, el cual los utilizará para generar una proyección de mayor calidad y finalmente la retransmitirá al cliente. De este modo se favorece significativamente al desempeño del lado del cliente. Finalmente, vale la pena mencionar el trabajo hecho por Kum et al. [56] en el cual se describen el diseño de un sistema para simulaciones de radioterapia, el cual habría de instalarse en zonas rurales donde los radiólogos no siempre estuvieran presentes para capacitar al personal clínico. Un servidor de aplicación con gran capacidad computacional almacena conjuntos de imágenes de tomografía que son enviados por los clientes, y efectúa los cálculos necesarios para efectuar

Patrón	Tipo	No. publicaciones	Estudios primarios revisados
Capas	Arquitectónico	5	[48, 51, 52, 33, 44]
Cliente-Servidor	Arquitectónico	5	[46, 36, 55, 56, 59]
N-Tier	Arquitectónico	1	[47]
Broker	Arquitectónico	1	[47]
Publish-Subscribe	Arquitectónico	1	[58]
Pipe-and-filter	Arquitectónico	1	[45]
Comando	Diseño	1	[48]
Composite	Diseño	1	[46]

Cuadro 2.6: Patrones arquitectónicos y de diseño empleados en sistemas médicos de visualización.

Atributo	No. publicaciones	Consultar
Desempeño	8	[45, 23, 46, 47, 36, 50, 56, 59]
Modificabilidad	6	[46, 47, 48, 52, 33, 58]
Usabilidad	4	[49, 50, 52, 59]
Disponibilidad	1	[52]
Seguridad	1	[56]

Cuadro 2.7: Atributos de calidad reportados.

las simulaciones de radioterapia. Una vez concluida la simulación, los resultados (que básicamente son los parámetros requeridos para efectuar el tipo de tratamiento solicitado) son enviados al cliente que la solicitó. El servidor también puede efectuar segmentación en las imágenes con el fin de calcular el área y volumen de los tumores presentes; para esto emplean las librerías de VTK e ITK, con lo que se favorece al desempeño de este proceso.

2.4. Conclusiones del EMS

Las dos principales técnicas de representación tridimensional son la representación por superficie y la representación por volumen. En los estudios primarios revisados se encontró que ambas se emplean casi en igual medida; la elección de emplear alguna de ellas o la combinación de ambas dependerá del propósito de la visualización y, por lo tanto, de lo que se busque resaltar de la información volumétrica presentada. Dado el alto costo computacional al que los algoritmos de visualización pueden incurrir, es comprensible que los análisis cuantitativos encontrados en los estudios se enfoquen principalmente en el desempeño. La fidelidad visual también parece ser un atributo deseable en la mayoría de los sistemas, ya sea que las representaciones se asemejen a las estructuras originales o que se presenten con la menor cantidad de ruido y artefactos posible.

Al haber ejecutado el EMS se encontró que varios trabajos abordan cuestiones de desempeño, seguido de modificabilidad y usabilidad. No obstante, aquéllos enfocados en desempeño, por ejemplo, se basan en el empleo de ciertos conceptos de diseño (bibliotecas de software, hardware gráfico y algoritmos, principalmente) para otorgar dicho atributo de calidad al sistema. El trabajo de Acosta y Liu [45] fue el único encontrado que aborda el desempeño desde una perspectiva arquitectónica, y lo hace proponiendo componentes genéricos que se ejecutan de

manera concurrente. Por su parte, los estudios primarios que consideran la modificabilidad sí lo hacen desde el diseño arquitectónico. El diseño por capas fue el que más se empleó para procurar este atributo de calidad.

Se encontraron pocos estudios primarios que trataran usabilidad, disponibilidad y seguridad (ver Cuadro 2.7), además de que todos emplean diferentes decisiones de diseño entre sí. En resumen, la evidencia hallada no es suficiente para entrever alguna tendencia a emplear determinados conceptos de diseño para lograr esos atributos. Lo que sí puede observarse, en general, es que no existe mucha tendencia al empleo de patrones y tácticas arquitectónicas en los sistemas y aplicaciones publicadas, aún siendo necesarias para otorgar características de calidad fundamentales en sistemas que manejan información biomédica. De los 39 estudios primarios que se revisaron, únicamente 14 (36 %) reportan haber efectuado algún esfuerzo de diseño arquitectónico (ver Cuadro 2.6).

Sólo el trabajo presentado por Buzurovic et al. [51] reporta explícitamente haber recabado y documentado una serie de requerimientos obtenidos de los interesados en su sistema. Aunque es muy probable que el desarrollo de los sistemas de visualización expuestos en los estudios primarios estuviera, efectivamente, fundamentado en legítimas necesidades de algún usuario o grupo de usuarios, la cuidadosa recopilación de requerimientos y restricciones también es fundamental para diseñar una arquitectura adecuada (sobre esto se abundará en el Capítulo 3). La evidencia encontrada indica que dichos requerimientos y restricciones rara vez se toman en cuenta durante el desarrollo de los sistemas o, cuando mucho, son considerados de forma muy somera.

El mapa expuesto en la figura 2.1 ilustra los resultados del EMS mediante una gráfica de burbuja que relaciona atributos con patrones arquitectónicos y herramientas. Puesto que también se encontró que ningún estudio reporta haber adoptado algún proceso de desarrollo de software, la propiedad que los considera (propiedad P3.3) no figura en dicho mapa.

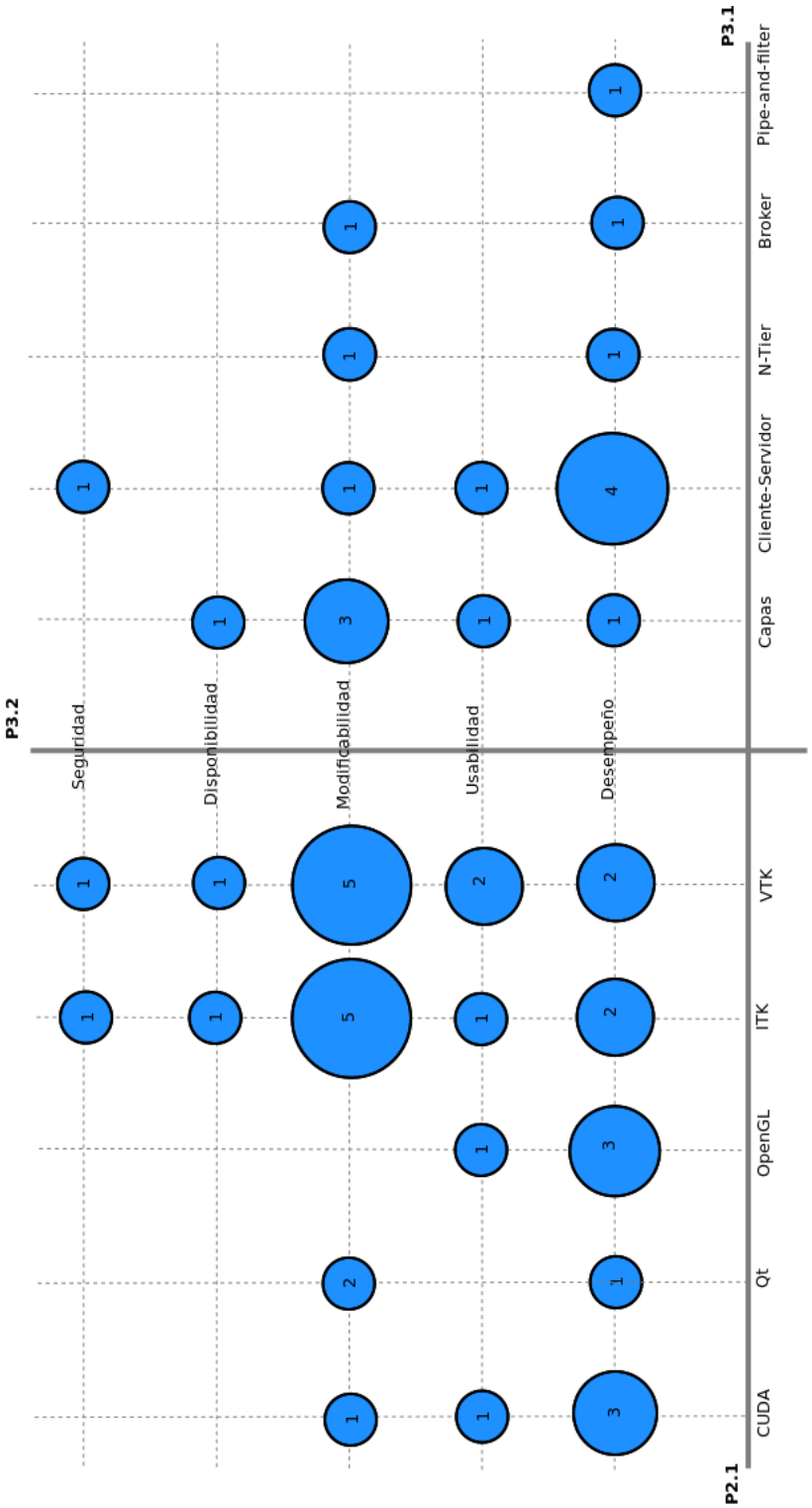


Figura 2.1: Mapa de los estudios primarios que ilustra los atributos de calidad reportados (propiedad P3.2) contra patrones (propiedad P3.1) y herramientas de software utilizadas (propiedad P2.1).

Capítulo 3

Marco de trabajo

Una metodología de desarrollo de software ha sido empleada en este proyecto y, puesto que los objetivos estipulan que deben lograrse atributos de calidad, ésta estuvo centrada principalmente en el diseño de arquitectura de software. En otras palabras, se adoptó un marco de trabajo que permitiera efectuar actividades estrechamente relacionadas al desarrollo arquitectónico. Ese marco de trabajo, las actividades propuestas para el diseño de la arquitectura y las herramientas empleadas para llevarlas a buen término se describen a lo largo de este capítulo.

3.1. Arquitectura de software

En la Sección 2.3.3 se introdujo una definición de arquitectura de software; en ella se hizo hincapié sobre la importancia de determinar cuáles son las estructuras de software que otorgan atributos de calidad al sistema. Retomando esa definición, se considera que varios elementos de software que guardan algún tipo de relación entre sí constituyen una estructura de software, las cuales a su vez se relacionan con otras estructuras de software para conformar finalmente al sistema. Básicamente, se consideran tres tipos de estructuras: *módulos*, *componentes* y *de asignación* [3].
Los *módulos* comprenden las unidades de implementación (código) o de datos que deben ser generadas o reutilizadas. Es decir, los elementos de software que conforman a los módulos pueden ser clases, capas o incluso otros módulos con responsabilidades más especializadas. Una vez que se han establecido las relaciones entre ellos, los módulos conforman entonces una vista estática del sistema. Asimismo, diferentes equipos de trabajo pueden ocuparse en implementar determinados módulos; de este modo, la vista de módulos también puede constituir una base para la repartición del actividades dentro del equipo de desarrollo.
Por su parte, los *componentes* son estructuras definidas en tiempo de ejecución del sistema. Una vez que el sistema está en funcionamiento los componentes interactúan entre ellos para llevar a cabo la funcionalidad del mismo. Los mecanismos mediante los cuales los componentes se comunican se denominan conectores. La correspondencia entre módulos y componentes suele ser varios-a-varios: la instancia de un módulo puede formar parte de varios componentes, ser parte de uno o ser un componente en sí.
Finalmente, las *estructuras de asignación* mapean módulos y componentes a elementos que

no son de software, como pueden ser procesadores (CPUs), sistemas de archivos o equipos de trabajo. En conjunto, conforman el panorama externo en el cual el software es desarrollado y puesto en operación.
La documentación arquitectónica que acompaña al presente proyecto incluye vistas con los tres tipos de estructura, puesto que cada una ilustra determinados atributos de calidad.

3.1.1. Diseño de la arquitectura de software

Las decisiones que se tomen respecto al diseño arquitectónico pueden impulsar u obstaculizar el logro de atributos de calidad, los cuales se ponen de manifiesto a través de los requerimientos y necesidades expresadas por los usuarios y otros interesados en el sistema. Sin embargo, entre esos requerimientos habrá algunos que tendrán un impacto más amplio y profundo en la arquitectura que los demás, y se les conoce como *directrices arquitectónicas*. Develar esas directrices es muy importante para lograr un diseño arquitectónico adecuado, por lo que párrafos siguientes describen el método empleado en este proyecto para lograrlo.

3.1.1.1. Taller de Atributos de Calidad (QAW, *Quality Attribute Workshop*)

Propuesto por el Instituto de Ingeniería de Software (SEI, *Software Engineering Institute*), albergado en la universidad Carnegie Mellon, el Taller de Atributos de Calidad (QAW) es un método que permite involucrar a clientes, usuarios y a otros interesados en el sistema en una etapa temprana del ciclo de desarrollo, de tal modo que con su ayuda las directrices arquitectónicas del sistema puedan ser descubiertas antes de efectuar cualquier diseño de la arquitectura. En otras palabras, representa una oportunidad de reunir a todos los interesados, platicar con ellos y preguntarles respecto a sus necesidades y expectativas relacionadas con los atributos de calidad que más les importan [69]. No obstante, el objetivo de QAW no es efectuar un análisis minucioso de la calidad requerida, sino identificar los escenarios en los que ésta se manifiesta para los distintos interesados en el sistema, es decir, los *escenarios de atributos de calidad*.
Recuérdese que un atributo de calidad es aquélla propiedad medible y comprobable que indica qué tan bien hace algo un sistema. Por lo tanto, un escenario de atributo de calidad es aquél que pone de manifiesto alguna de esas propiedades; se podría decir que es una "historia corta" que relata una situación del sistema en la que se cumple un atributo de calidad. Para ser útiles, los escenarios de atributos de calidad deben denotar, por lo menos, un estímulo y una respuesta. El *estímulo* es el factor que causa que el sistema reaccione de cierta manera; la *respuesta* es, pues, dicha reacción. Adicionalmente, es conveniente que los escenarios también estipulen el *ambiente* o contexto en el cual se suscita el escenario. Por ejemplo, un sistema puede tener varios modos de operación o encontrarse en diferentes situaciones, como alto tráfico de datos, acceso concurrente de usuarios, mantenimiento, o bien modo normal de operación.
Para aclarar lo anterior, considérese el siguiente ejemplo de cómo debe plantearse un escenario de atributo de calidad:

- **Estímulo**: el usuario está efectuando tareas de rutina con el sistema.
- **Respuesta**: el sistema permite efectuar tareas con un máximo de 3 "clicks".

- **Ambiente**: uso cotidiano del sistema en condiciones normales de operación.

El escenario anterior resalta un aspecto relevante para el atributo de calidad conocido como *usabilidad* de un sistema. Cabe mencionar que es muy recomendable que los escenarios estipulen algún parámetro medible como parte de la respuesta, de modo que así se imponga un criterio bien definido que la arquitectura debe cumplir. En el caso del ejemplo, dicho parámetro está determinado por el número de "clicks" necesarios para efectuar cualquier tarea en el sistema.
La participación de los interesados es fundamental para QAW y se espera que se comprometan a estar presentes y activos durante toda la duración del taller. Asimismo, pueden hacer preguntas y emitir comentarios en cualquier momento tomando en cuenta, por supuesto, las restricciones en cuanto a la duración de las actividades. El método se presenta de forma resumida en el diagrama de actividad presentado en la Figura 3.1; las actividades y productos de trabajo presentados en dicho diagrama se detallan a continuación:

1. **Presentaciones e introducción al taller**
   Los organizadores del taller relatan la motivación del mismo y explican brevemente los pasos a seguir. Luego, se introducen a sí mismos y el resto de los participantes hacen lo mismo a la vez que comentan cuál es su perfil, su rol en la organización y su relación con el sistema que se está construyendo.

2. **Presentación del negocio/misión**
   Un representante de los interesados en el sistema expone el contexto del negocio/misión con el que se relaciona el sistema, así como los requerimientos funcionales y no funcionales de alto nivel, y cualquier restricción.

3. **Presentación de la propuesta inicial de arquitectura**
   Alguno de los interesados con el perfil apropiado presenta cualquier artefacto disponible relacionado con la arquitectura, como pueden ser: planes y estrategias sobre cómo cumplir los requerimientos del negocio, requerimientos y restricciones (sistemas operativos, hardware, middleware, estándares, etc.) que guiarán las decisiones de diseño arquitectónico, diagramas de la arquitectura a alto nivel, o cualquier otra descripción escrita de la misma. El resto de los interesados debe anotar cualquier información que consideren útil de esta presentación.

4. **Identificación de directrices arquitectónicas**
   Los organizadores otorgan un descanso de 15-20 minutos al resto del grupo, durante los cuales analizarán el material que se presentó durante los pasos 2 y 3. Una vez hecho, piden a los interesados que hagan aclaraciones y correcciones de tal modo que no haya ambigüedades en los requerimientos de alto nivel, atributos de calidad, reglas del negocio y restricciones.

5. **Lluvia de escenarios**
   Los organizadores exhortan a los participantes a idear y anotar escenarios de atributos de calidad. Deben cerciorarse de que se proponga por lo menos un escenario para cada directriz arquitectónica identificada en el paso 4. Los escenarios no deben ser demasiado generales y no deben constituir requerimientos funcionales. Al final se recaban todos los escenarios anotados por los participantes.

6. **Consolidación de escenarios**
   Se leen y analizan cada uno de los escenarios obtenidos en el paso anterior. Aquellos que sean similares pueden ser replanteados en un único escenario, siempre y cuando las personas que los propusieron estén de acuerdo. De este modo se genera una lista de escenarios únicos.

7. **Establecimiento de prioridades para escenarios**
   La lista de escenarios que resulte del paso anterior se presenta a los participantes; los interesados votan para decidir la prioridad que tiene cada escenario. El número de votos que cada interesado tendrá es igual al 30 % del total de escenarios consolidados, redondeado al entero par más cercano según convenga; se podrá asignar hasta la mitad de los votos a un único escenario. Finalmente, se efectúa el conteo de votos para cada escenario.

8. **Refinamiento de escenarios**
   Se seleccionan los cuatro o cinco escenarios que hayan tenido más votos en el paso anterior. El refinamiento implica identificar los objetivos de negocio y los atributos de calidad que están relacionados con aquellos escenarios; en otras palabras, se identifican los atributos de calidad que se requieren del sistema. Asimismo, en este paso se procede a resolver dudas e inconvenientes que el resto de los interesados puedan tener respecto a esos requerimientos.

3.1.1.2. Diseño Dirigido por Atributos (ADD, *Attribute-Driven Design*)

El Diseño Dirigido por Atributos (ADD) también es un método propuesto por el SEI, y constituye un enfoque para diseñar una arquitectura de software basándose en requerimientos de atributos de calidad. Sigue un proceso de diseño iterativo al ir descomponiendo el sistema o alguno de sus elementos empleando *patrones arquitectónicos* y *tácticas arquitectónicas* que satisfagan esos requerimientos [70].
Un *patrón arquitectónico* es una solución a un problema recurrente en el diseño de arquitecturas de software; especifica un conjunto de elementos de software, así como las relaciones entre éstos, que en conjunto satisfacen varios requerimientos funcionales y no funcionales (estos últimos son los que expresan los atributos de calidad con los que debe cumplir el sistema) [71]. Por ejemplo, uno de los patrones arquitectónicos más socorridos es el de capas, el cual es útil cuando el software debe ser desarrollado en módulos que puedan ser implementados, probados y documentados por separado, con la menor interacción posible entre ellos; de este modo, se favorece la modificabilidad y portabilidad del sistema.
Por su parte, una *táctica arquitectónica* es una decisión de diseño que influencia la respuesta de un sistema para cumplir con un único atributo de calidad [3, 71]. Por ejemplo, para lograr el atributo de calidad de seguridad en un sistema existen tácticas encaminadas a prevenir ataques, detectar ataques y recuperarse de ataques. Como podrá notarse, tanto patrones como tácticas influyen en el diseño de la arquitectura, pero estas últimas lo hacen a menor escala. Los patrones arquitectónicos suelen conllevar la aplicación de una o más tácticas. La descripción de patrones y tácticas arquitectónicas para lograr diferentes atributos de calidad está fuera del alcance de este proyecto, pero se refiere al interesado al libro de Bass, Clements

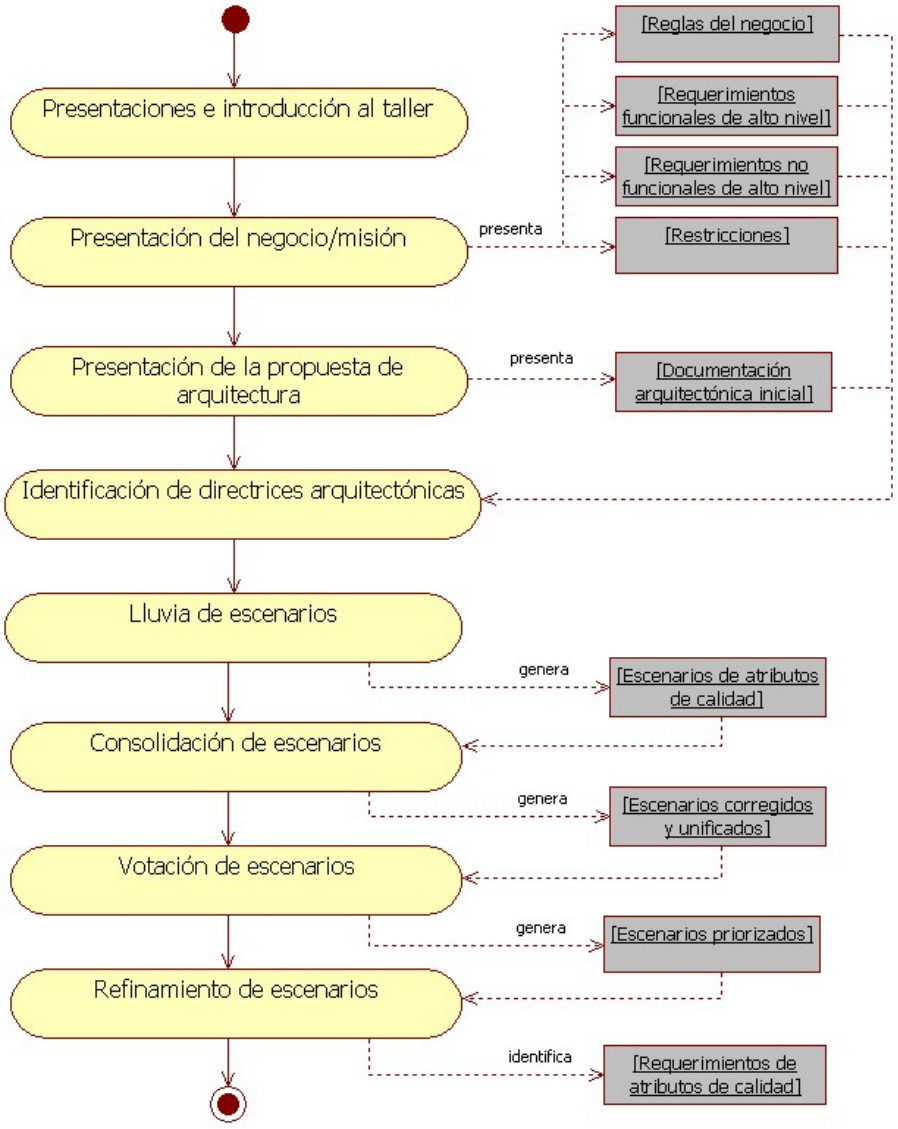


Figura 3.1: Diagrama de actividad para QAW. Los círculos en la parte superior e inferior del diagrama indican el inicio y el final de las actividades, respectivamente. Las actividades son representadas mediante rectángulos ovalados y los productos mediante rectángulos normales. Las flechas entre actividades indican la secuencia en que éstas son realizadas; las flechas punteadas que surgen de una actividad indican que ésta genera el producto apuntado, mientras que una flecha punteada que surge de un producto significa que este se usa como entrada para la actividad apuntada. La notación del diagrama corresponde al estándar UML.

y Kazman [3] para una descripción completa de varias tácticas y de algunos patrones arquitectónicos. Una lista más detallada de estos últimos puede consultarse en la tesis de maestría de Casales Cabrera [72].
Ahora, puesto que el objetivo principal del método ADD es generar un diseño de arquitectura de software, entonces requiere como productos de entrada a los requerimientos funcionales, atributos de calidad y restricciones considerados prioritarios para los interesados en el sistema. Estos son, efectivamente, los productos de salida del método QAW que se describió en la Sección 3.1.1.1. ADD es un método iterativo de descomposición de elementos de software, y el diagrama de actividad en la Figura 3.2 lo presenta de forma general. Los pasos planteados en el diagrama se exponen continuación:

1. **Escoger un elemento a descomponer**
   Se escoge uno de los elementos para dividirlo en elementos hijo. El primer elemento siempre es el sistema completo, por lo que el resultado de la primera iteración será un diseño muy general conteniendo elementos que, muy probablemente, deberán ser descompuestos en iteraciones subsecuentes.

2. **Identificar directrices arquitectónicas**
   En este paso es necesario retomar las restricciones y los requerimientos de atributos de calidad obtenidos como resultado de efectuar QAW, así como los principales requerimientos funcionales solicitados, para luego decidir cuáles están estrechamente relacionados con el elemento en descomposición. En otras palabras, debe estipularse qué requerimientos y restricciones constituyen directrices arquitectónicas para el elemento.

3. **Escoger patrones arquitectónicos**
   Determinar qué patrón o patrones arquitectónicos involucran a las tácticas requeridas para satisfacer las directrices del elemento en descomposición. Asimismo, también deben considerarse otros conceptos de diseño que promuevan los atributos de calidad, como bibliotecas de software o hardware especializado. Dichos conceptos de diseño pueden provenir, por ejemplo, de la revisión del estado del arte.
   El resultado de este paso es una o más vistas arquitectónicas representativas del elemento de software en descomposición; cada vista mostrará las diferentes relaciones entre los elementos hijo identificados. Deberá emplearse por lo menos una de las tres vistas principales: módulos, componentes y asignación (consultar la Sección 3.1). No es necesario documentar completamente las vistas en este momento, sino sólo registrar la información relevante para explicar las decisiones de diseño que se hayan tomado.

4. **Instanciar los elementos y usar múltiples vistas para asignarles funcionalidad**
   Para este paso deben considerarse todos los elementos hijo hallados en el paso anterior. Instanciar los elementos implica asignarles las responsabilidades que le corresponden. En otras palabras, deben enlistarse todas las obligaciones que han de llevar a cabo para cumplir con su funcionalidad, la cual está indicada por los casos de uso (ver Sección 3.2.1). El número total de responsabilidades repartidas entre los elementos hijo deberá ser igual al asignado inicialmente al elemento padre.

5. **Definir las interfaces para los elementos hijo**
   Las interacciones entre los nuevos elementos tendrán ciertas condiciones. Es decir, cada elemento puede requerir y producir cierta información en el proceso de cumplir con sus responsabilidades. Por lo tanto, la documentación para las interfaces debe detallar cuál es esa información requerida y producida por los elementos de software.

6. **Verificar y refinar casos de uso y escenarios de atributos de calidad**
   Aquí simplemente se verifica que los casos de uso y los escenarios están los suficientemente definidos para que los elementos hijo puedan ser descompuestos adecuadamente.

7. **Repetir todos los pasos de arriba para un elemento hijo**
   Aplicar todos los pasos anteriores a los elementos hijo (a aquéllos que lo requieran, por supuesto) hallados en la iteración actual.

Una vez que se ha alcanzado un grado en la descomposición suficiente para efectuar la implementación (estipulada por la vista de módulos), entonces se puede dar por concluida la aplicación del método de ADD. Es entonces cuando se contará con un diseño arquitectónico lo suficientemente completo para cumplir con los requerimientos funcionales y atributos de calidad necesarios para los interesados. Sin embargo, el ciclo de desarrollo de un sistema de software no se limita únicamente al diseño arquitectónico; hay otras actividades que deben efectuarse previa y posteriormente al diseño de la arquitectura, y serán presentadas en las siguientes secciones.

3.1.2. Evaluación de los atributos de calidad

Existen métodos para evaluar arquitecturas de software con respecto a qué tan adecuadas son para otorgar atributos de calidad. El SEI ha propuesto los denominados *Architecture Tradeoff Analysis Method* (ATAM), *Software Architecture Analysis Method* (SAAM) y *Active Reviews for Intermediate Designs* (ARID) para analizar y evaluar arquitecturas, ya sea que estén completamente desarrolladas o en sus fases iniciales de diseño [73]. No obstante, estos métodos deben ser ejecutados por personal calificado y, por lo tanto, aplicarlos en un proyecto como este incurriría en costos insostenibles.
Por otra parte, recientemente se ha propuesto un modelo de referencia para estimar el valor del producto software conocido como RESVEP (acrónimo para *REference model for the Software product Value Estimation Process*) [74], el cual emplea una serie de indicadores para evaluar varios atributos de calidad. Esos indicadores se enlistan en el Cuadro 3.1, exponiendo también el esquema bajo el cual son clasificados. Nótese que la clasificación consiste básicamente en agrupar varios sub-atributos de calidad bajo un atributo más general; el indicador para este último se obtiene al promediar a los primeros. Los detalles concretos sobre cómo calcular cada indicador puede consultarse en el Apéndice C.
En general, la utilización de estos indicadores para determinar qué tanto la arquitectura de software cumple con atributos de calidad prioritarios es una alternativa mucho más viable que los métodos de evaluación anteriormente mencionados, por lo menos para el presente proyecto.

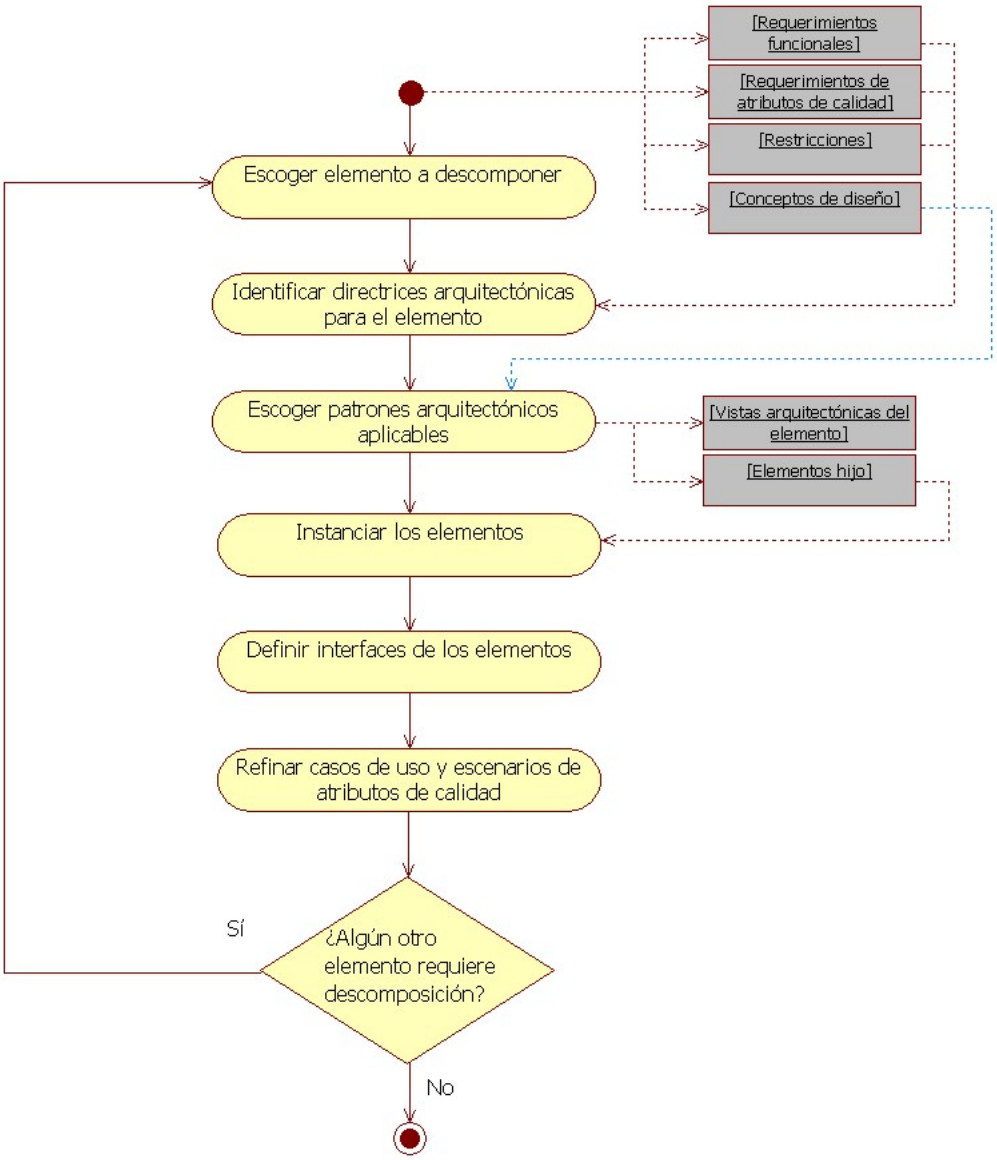


Figura 3.2: Diagrama de actividad para ADD. La notación del diagrama corresponde al estándar UML.

Atributo de calidad	Sub-atributo de calidad	Atributo de calidad	Sub-atributo de calidad
Mantenibilidad	▪ Modularidad ▪ Reusabilidad ▪ Analizabilidad ▪ Modificabilidad ▪ Facilidad de prueba	Seguridad	▪ Confidencialidad ▪ Integridad ▪ No-rechazo ▪ Constancia de acciones ▪ Autenticación
Rendimiento	▪ Comportamiento respecto a tiempos ▪ Consumo de recursos	Usabilidad	▪ Facilidad para aprendizaje ▪ Facilidad de uso ▪ Protección contra errores ▪ Estética de la interfaz gráfica
Confiabilidad	▪ Madurez ▪ Disponibilidad ▪ Tolerancia a fallas ▪ Capacidad de recuperación	Funcionalidad	▪ Funcionalidad correcta ▪ Precisión de la funcionalidad
Compatibilidad	▪ Coexistencia ▪ Interoperabilidad	Portabilidad	▪ Adaptabilidad ▪ Facilidad de reemplazo ▪ Facilidad de instalación
Trazabilidad	▪ (ninguno)		

Cuadro 3.1: Indicadores de atributos de calidad evaluados por RESVEP. El indicador del atributo más general se obtiene al promediar los indicadores de los sub-atributos.

3.2. Proceso de desarrollo de software

Ivar Jacobson declaró hace ya varios años, de manera sucinta y determinante, que un proceso debe definir *quién* hace *qué*, *cuándo* y *cómo* para alcanzar cierto objetivo [4]. También dijo que en ingeniería de software el objetivo es construir un producto software o mejorar uno existente. Por lo tanto, un proceso efectivo debe proveer pautas para el desarrollo eficiente de software de calidad, considerando las mejores prácticas estipuladas por el más actual estado del arte. Se tomó la decisión de adoptar un proceso de desarrollo de software que guiase paso por paso las actividades y tareas en cada fase del proyecto: OpenUP.

3.2.1. OpenUP

OpenUP (abreviación de *Open Unified Process*) constituye un proceso de desarrollo de software dentro del marco de trabajo de Eclipse (Eclipse Process Framework). La página oficial del proceso [5] lo define como una variante ligera del Proceso Unificado que aplica un procedimiento iterativo e incremental, conformando un ciclo de desarrollo bien estructurado. Es independiente de cualquier herramienta, ajustable a las habilidades del equipo de trabajo, y puede ser extendido y/o adaptado a cualquier tipo de proyecto para desarrollar software. Básicamente OpenUP establece que cada integrante del equipo tiene uno o más roles; varios roles se apoyan en la realización de actividades; cada actividad consiste en varias tareas; y las tareas generan productos de trabajo (también conocidos como "artefactos"). Asimismo, ofrece una significativa cantidad de guías, ejemplos, definiciones de conceptos y herramientas para apoyar el correcto seguimiento y ejecución del proceso.
Análogamente al Proceso Unificado, OpenUP divide al ciclo de desarrollo de software en cuatro fases: *concepción*, *elaboración*, *construcción* y *transición* [4]. El trabajo dentro de cada fase debe repartirse en una o más iteraciones, realizando en cada una de éstas las actividades que el proceso dicta o, concretamente, aquéllas con las cuales el proceso haya sido instanciado para el proyecto en cuestión. Es completamente válido, y por lo general necesario, adaptar el proceso a las necesidades particulares de cada proyecto mediante la añadidura o eliminación de actividades y tareas; también pueden incluirse nuevos roles, guías y material de apoyo personalizado. Cada fase se da por concluida cuando se ha alcanzado un hito considerado el objetivo principal de la fase, pudiendo entonces dar paso a la siguiente. El diagrama de actividad expuesto en la Figura 3.3 resume el ciclo de vida del desarrollo de software planteado por OpenUP.
Cada fase de OpenUP consta de una o más iteraciones en las que se pueden incluir varias actividades, según se efectúe la instancia del proceso. Cada actividad se divide a su vez en varias tareas que pueden requerir ciertos artefactos como entrada, y generan otros artefactos como salida. Algunas actividades pueden incluso tener anidadas a otras actividades. Las siguientes subsecciones describen brevemente cada una de las fases de OpenUP; descripciones generales de las actividades para cada fase se exponen en el Apéndice B. Por su parte, la instancia de OpenUP generada para este proyecto se presenta a lo largo de la Sección 3.2.2.

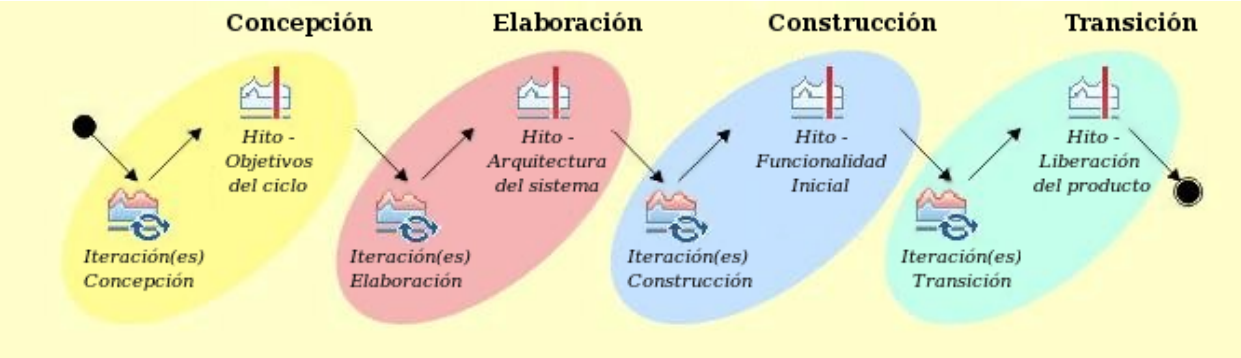


Figura 3.3: Ciclo de vida de desarrollo de software planteado por OpenUP. Ilustración obtenida de [5].

3.2.1.1. Concepción

En esta fase arranca el proyecto; el equipo cuenta con justificaciones sólidas para llevarlo a cabo. Se genera una visión general del sistema y se identifican los principales requerimientos funcionales y no funcionales. Deben establecerse los alcances del sistema, así como los costos y riesgos para el proyecto para luego proponer estrategias para mitigarlos a la brevedad. Se establecen los *casos de uso*, los *actores* externos relacionados con el sistema y el *modelo de casos de uso* inicial con base en los requerimientos funcionales; asimismo, debe elaborarse una *propuesta arquitectónica inicial* que contemple los requerimientos no funcionales identificados. Los *casos de uso* son una manera sistemática e intuitiva de capturar los requerimientos funcionales, y son un elemento clave para guiar todo el proceso de desarrollo pues representan aquello que los usuarios necesitan que el sistema haga, para poder ellos mismos hacer su trabajo y cumplir con los objetivos de negocio. Lo que hace intuitivos a los *casos de uso* es que suelen describirse usando lenguaje natural, sin tecnicismos, y por lo tanto son fácilmente comunicables a todos los interesados en el sistema. Por su parte, los *actores* son cualquier entidad externa que requiera intercambiar información y/o efectuar tareas con el sistema; no son necesariamente personas físicas, por lo que también pueden representar otros sistemas externos. Finalmente, *casos de uso* se relacionan entre sí y con los *actores* mediante un *modelo de casos de uso*, el cual básicamente expone todas las formas posibles en las que el sistema se puede usar [4].
El hito de esta fase se alcanza cuando las primeras versiones de los documentos conteniendo todo lo anterior han sido generadas, y por lo tanto los objetivos para el ciclo de desarrollo en curso han sido estipulados y comprendidos.

3.2.1.2. Elaboración

Durante esta fase los requerimientos funcionales deben terminar de identificarse y comprenderse; la mayoría de los *casos de uso* estarán debidamente priorizados y documentados, y el *modelo de casos de uso* estará casi completo. La *propuesta arquitectónica inicial* proveniente de la fase de concepción se complementa con nuevas decisiones que incorporen los requerimientos funcionales y no funcionales con potencial impacto en la arquitectura. La monitorización de riesgos continúa, y se siguen aplicando estrategias para evaluar su impacto

y mitigarlos en lo posible. Al final de la fase debe contarse con un *modelo arquitectónico* estable que servirá de guía en fases posteriores. También debe contarse con una *arquitectura ejecutable* inicial, la cual constituye una implementación que demuestre que se cumple con los requerimientos prioritarios.

3.2.1.3. Construcción

Se cuenta con una arquitectura estable al inicio de esta fase y, por lo tanto, el énfasis se hace principalmente en diseño, implementación y pruebas de componentes necesarios para realizar los *casos de uso*. La mayoría de los riesgos han sido reducidos al mínimo, lo que permite al jefe de proyecto centrarse en la eficiencia del equipo. Partiendo de la *arquitectura ejecutable* establecida en la fase de elaboración, el equipo genera un producto software listo para operar en el ambiente del usuario; a esto suele llamársele "pruebas beta". La fase concluye cuando una versión funcional del sistema ha sido generada.

3.2.1.4. Transición

El objetivo de esta fase es colocar al sistema en su entorno final de operación [4]. El sistema es operacional aunque no necesariamente está completo; algunos problemas pudieron no haberse detectado al concluir la fase de construcción, tal vez surjan nuevos problemas una vez que el sistema está en su entorno de operación, o bien puede que los interesados descubran nuevas necesidades no vislumbradas en fases previas. Si algo de lo anterior ocurriera, el jefe del proyecto deberá decidir si los cambios pueden efectuarse a estas alturas, o si deben esperar hasta un nuevo ciclo de desarrollo. Si el sistema está dirigido a un público amplio, el equipo distribuye una "versión beta" a sitios donde los usuarios puedan obtenerla; si es para unos pocos o para un solo usuario entonces el sistema se instala en su entorno de trabajo. Luego, los usuarios pueden brindar retroalimentación útil para decidir si el sistema y/o algún artefacto requieren modificaciones; todo cambio que se decida incorporar deberá ser de poco impacto. El equipo de trabajo también debe apoyar en la preparación del entorno de operación y en la capacitación de los usuarios.

3.2.2. Instancia de OpenUP

La Figura 3.4 presenta, a modo de diagrama de actividad, un resumen de las actividades llevadas a cabo para la instancia de OpenUP (esto es, la adapatación del proceso a este proyecto). Las actividades específicas para cada fase de esta instancia se detallan en las siguientes subsecciones.

3.2.2.1. Concepción

Básicamente, la fase de concepción consistió en una única iteración enfocada en recaudar los requerimientos iniciales, y en definir y ejecutar el EMS (ver Sección 2.2). En dicha iteración se incluyeron las cuatro actividades establecidas por OpenUP para esta fase: *iniciar el proyecto*, *planear y gestionar la iteración*, *identificar y refinar los requerimientos*, y *acordar estrategias técnicas*.

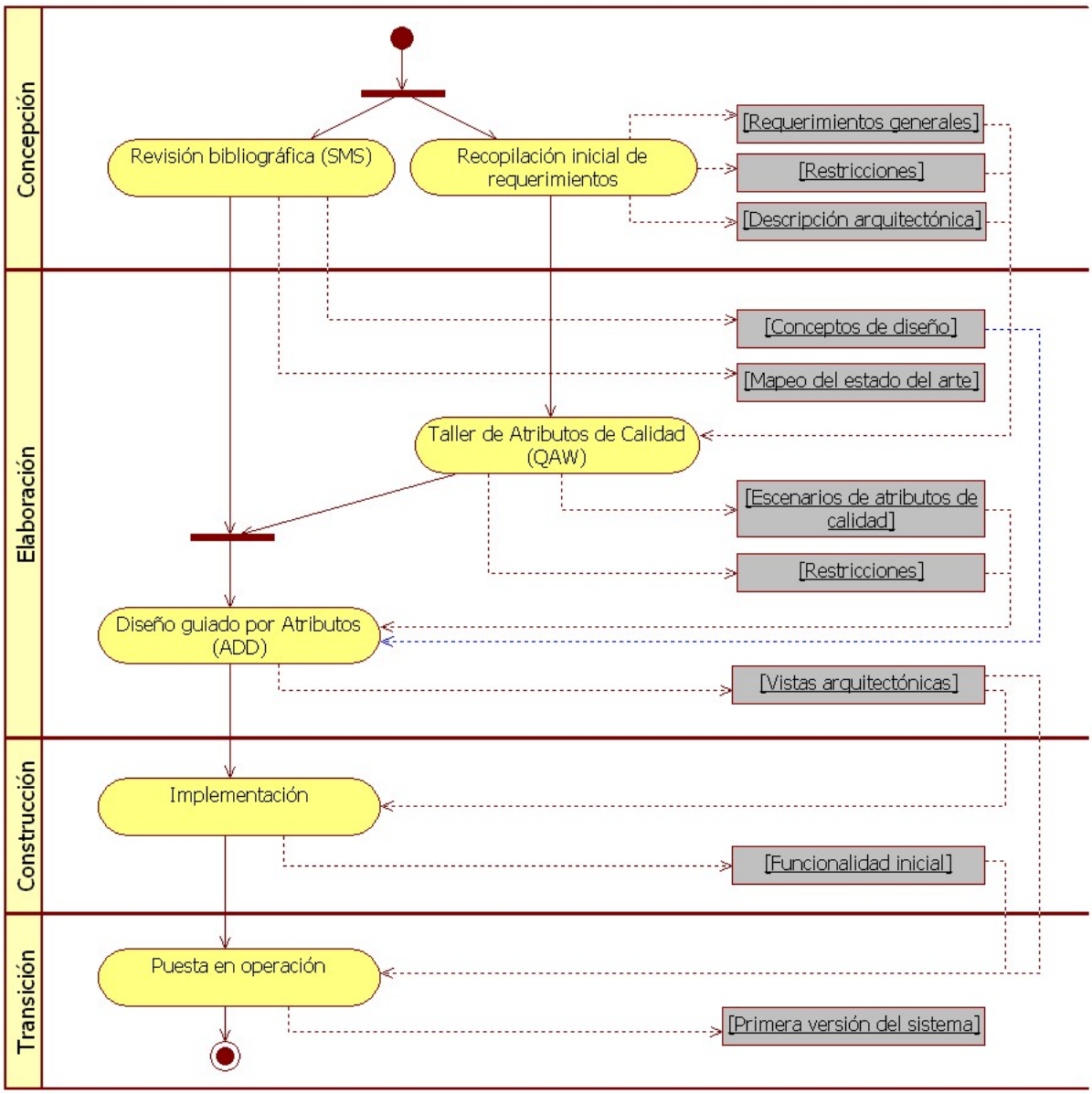


Figura 3.4: Diagrama de actividad que sintetiza la instancia de OpenUP para desarrollar el sistema de visualización 3D. Las fases son representadas mediante carriles horizontales que contienen las actividades correspondientes. Las líneas sólidas horizontales en las que convergen y divergen varias flechas son líneas de sincronización. Dos o más flechas convergiendo en una línea de sincronización implica que las actividades de las que surgen deben completarse antes de continuar el flujo de actividades; dos o más flechas divergiendo de una línea de sincronización significa que las actividades a las que apuntan pueden ser realizadas paralelamente.

- La actividad denominada *Planear y gestionar la iteración* fue común para todas las fases de la instancia del proceso de desarrollo, y se desarrolló constantemente a lo largo de cada una. Su propósito es establecer los tiempos para la realización de cada tarea, así como identificar los riesgos por los que el proyecto podría atravesar y que generarían retrasos en esos tiempos; si los riesgos resultan ser significativos entonces deben establecerse estrategias para mitigarlos a la brevedad. Asimismo, debe estipularse la meta que marca el hito de la fase en cuestión y que será alcanzado mediante la planeación y ejecución de las actividades indicadas por la instancia. Concretamente, en la concepción se propuso una planeación inicial de actividades para la instancia del proceso de desarrollo, aunque esta lista se modificó conforme el proceso avanzaba en su ejecución.
- La actividad nombrada *Iniciar el proyecto* implicó efectuar conversaciones con los usuarios del sistema para escuchar sus ideas y necesidades generales respecto al sistema de visualización. Todos los requerimientos y sus descripciones generales fueron registrados en un *documento de visión*; asimismo, se redactó un *glosario* que define los términos propios de los temas de visualización tridimensional.
- La actividad llamada *Identificar y refinar los requerimientos* inició una vez concluida la de *Iniciar el proyecto*. Se propusieron los principales *casos de uso* y a los *actores* del sistema, así como la definición del *modelo de casos de uso*.
- En la actividad denominada *Acordar estrategias técnicas* deben proponerse maneras en que los requerimientos del sistema pueden ser abordados. Para ello, en la instancia de OpenUP se propuso efectuar la definición y la ejecución del EMS como tarea fundamental de esta actividad; la motivación para esto fue la búsqueda de herramientas, conceptos de diseño y necesidades reportadas que justificaran y apoyaran la realización del proyecto.

Una vez realizadas todas las actividades de la fase de concepción el alcance del ciclo de desarrollo estuvo, finalmente, bien delimitado. Las actividades y tareas establecidas por la instancia de OpenUP para esta fase se presentan en el cuadro 3.2.

3.2.2.2. Elaboración

La mayor parte de las tareas encaminadas al diseño de la arquitectura del sistema deben llevarse a cabo en esta fase. La instancia de OpenUP contempló una sola iteración con cinco actividades: *planear y gestionar la iteración*, *identificar y refinar los requerimientos*, *desarrollar la arquitectura*, *desarrollar incremento de la solución* y *atender tareas pendientes*.

- La actividad de *Planear y gestionar la iteración* se efectuó lo largo de toda la fase; en ella se monitorizaron los principales riesgos para el proyecto, y cuando fue necesario los tiempos para las tareas se ajustaron de tal modo que las fechas de entrega no fuesen afectadas significativamente.
- La actividad denominada *Identificar y refinar los requerimientos* conllevó a cerciorarse que los requerimientos iniciales plasmados, tanto en el *documento de visión* como en el *modelo de casos de uso*, se habían entendido y registrado lo más correctamente posible; los cambios considerados necesarios en esos documentos se hicieron a la brevedad. Además, la tarea de efectuar el Taller de Atributos de Calidad (QAW), descrito en

Actividad	Tarea	Productos de entrada	Productos de salida
Iniciar el proyecto	Desarrollar la visión técnica	-	Documento de visión, glosario
	Planear el proyecto	-	Plan de proyecto
Planear y gestionar la iteración	Planear la iteración	Lista de pendientes	Lista de riesgos, lista de pendientes
	Instanciar el proceso	Proceso definido para el proyecto	Proceso definido para el proyecto
	Poner el proceso en operación	Proceso definido para el proyecto	Proceso definido para el proyecto
	Gestionar la iteración	Lista de riesgos, lista de pendientes	Lista de riesgos, lista de pendientes
	Evaluar los resultados	Lista de pendientes	Lista de pendientes
Identificar y refinar los requerimientos	Identificar los requerimientos	-	Glosario, casos de uso, modelo de casos de uso, lista de pendientes
	Detallar los casos de uso	Casos de uso	Casos de uso, modelo de casos de uso, glosario
Acordar estrategias técnicas	Efectuar revisión de bibliográfica (EMS)	Preguntas de investigación	Mapeo del estado del arte

Cuadro 3.2: Actividades y tareas efectuadas en la fase de concepción.

la Sección 3.1.1, se agregó a la instancia del proceso dentro de esta actividad. Por su parte, la actividad de *Desarrollar la arquitectura* consideró la única tarea de ejecutar el método Diseño Guiado por Atributos (ADD), el cual también se describió en la Sección 3.1.1.

- La actividad *Desarrollar incremento de la solución* empezó posteriormente al inicio de la ejecución de ADD. Básicamente, la actividad consistió en comenzar a implementar y hacer pruebas con los elementos de software que se fueron identificando. Conforme se suscitaron avances en ADD, nuevas implementaciones y pruebas pudieron ser realizadas. No obstante, la mayor parte de las tareas de implementación se llevaron a cabo más adelante, durante la fase de construcción.
- La actividad llamada *atender tareas pendientes*, que también se efectuó a lo largo de toda la fase de elaboración, se enfocó a finiquitar cualquier tarea comenzada desde la fase de concepción. Específicamente, se completó la revisión y extracción de las propiedades en los estudios primarios arrojados por el EMS.

Una vez realizadas todas las actividades de la fase de elaboración el diseño de la arquitectura de software había sido mayormente completado. Algunos aspectos del diseño detallado sufrieron cambios en fases subsecuentes, pero ninguno de esos cambios fue sustancial. Las tareas y actividades que efectuadas durante esta fase se enlistan en el cuadro 3.3.

3.2.2.3. Construcción

La fase de construcción se centró en las tareas de implementación y pruebas para los elementos de software que se identificaron al efectuar ADD. Dos iteraciones se contemplaron para esta fase en la instancia del proceso, y en ambas se especificaron las mismas actividades: *planear y gestionar la iteración*, *desarrollar incremento de la solución*, *generar documentación del producto*, *probar la solución*, y *atender tareas pendientes*.

Actividad	Tarea	Productos de entrada	Productos de salida
Planear y gestionar la iteración	Planear la iteración	Lista de pendientes	Lista de riesgos, lista de pendientes
	Gestionar la iteración	Lista de riesgos, lista de pendientes	Lista de riesgos, lista de pendientes
	Evaluar los resultados	Lista de pendientes	Lista de pendientes
Identificar y refinar los requerimientos	Identificar los requerimientos	Documento de visión, glosario, documentación arquitectónica	Casos de uso, modelo de casos de uso, lista de pendientes
	Efectuar Taller de Atributos de Calidad (QAW)	Documento de visión, documentación arquitectónica	Escenarios de atributos de calidad, documentación arquitectónica
	Detallar casos de uso	Casos de uso	Casos de uso, modelo de casos de uso, glosario
Desarrollar la arquitectura	Refinar la arquitectura - Diseño Guiado por Atributos (ADD)	Escenarios de atributos de calidad	Documentación arquitectónica
Desarrollar incremento de la solución	Diseñar la solución	Modelos de casos de uso, documentación arquitectónica	Diseño
	Implementar pruebas de desarrollador	-	Pruebas de desarrollador
	Ejecutar pruebas de desarrollador	Pruebas de desarrollador	-
Atender tareas pendientes	Efectuar revisión de bibliográfica (EMS)	Preguntas de investigación	Mapeo del estado del arte

Cuadro 3.3: Actividades y tareas efectuadas en la fase de elaboración.

- La actividad de *Planear y gestionar la iteración* se efectuó a lo largo de toda la fase; al igual que en las fases anteriores, los principales riesgos fueron atendidos y mitigados, y los tiempos de trabajo apropiadamente ajustados cuando fue necesario. En la primera iteración se procuró que todas las herramientas (como bibliotecas de software y hardware determinados por los resultados del EMS) estuvieron disponibles y debidamente instalados en el entorno de desarrollo.
- La actividad de *Desarrollar incremento de la solución* engloba las tareas de implementación y prueba de los elementos de software definidos en la arquitectura. Para la primera iteración dicha actividad estuvo enfocada en la implementación y prueba de elementos de software encargados de satisfacer los requerimientos no relacionados con visualización; para la segunda iteración se centró, por lo tanto, en la implementación y prueba de los elementos de software encargados de la visualización.
- Las tareas que conformaron a la actividad denominada *Generar documentación del producto* requirieron efectuar la redacción de los distintos manuales del sistema; asimismo, la *documentación arquitectónica* siguió trabajándose a lo largo de la fase de construcción al punto de estar casi completa al alcanzar el hito.
- Finalmente, la actividad llamada *atender tareas pendientes* se introdujo en la instancia para concluir tareas empezadas en fases previas o que se requerían para llevar a cabo otras actividades de la fase de construcción; por ejemplo, el estudio de las herramientas de software y hardware empleadas en el desarrollo del sistema puede estuvo considerado dentro de esta actividad.

Actividad	Tarea	Productos de entrada	Productos de salida
Planear y gestionar la iteración	Planear la iteración	Lista de pendientes	Lista de riesgos, lista de pendientes
	Configurar herramientas (sólo iteración 1)	Herramientas	Herramientas
	Verificar instalación y configuración de herramientas (sólo iteración 1)	Herramientas	Herramientas
	Gestionar la iteración	Lista de riesgos, lista de pendientes	Lista de riesgos, lista de pendientes
	Evaluar resultados	Lista de pendientes	Lista de pendientes
Desarrollar incremento de la solución	Diseñar la solución	Modelos de casos de uso, documentación arquitectónica	Diseño
	Implementar la solución	-	Implementación
	Implementar pruebas de desarrollador	-	Pruebas de desarrollador
	Correr pruebas de desarrollador	Pruebas de desarrollador	-
	Integrar y crear “build”	-	Build
Generar documentación del producto	Desarrollar documentación del producto	-	Documentación del producto
	Desarrollar documentación de usuario	-	Documentación de usuario
	Desarrollar documentación técnica	-	Documentación técnica
Atender tareas pendientes	Solicitar cambios	-	Lista de pendientes

Cuadro 3.4: Actividades y tareas efectuadas para las dos iteraciones de la fase de construcción.

Una vez concluida la fase de construcción ya se contaba con una versión del sistema que exhibía la funcionalidad solicitada. Las actividades y tareas a efectuadas en la fase de construcción se enlistan en el cuadro 3.4.

3.2.2.4. Transición

Las actividades propuestas para esta fase tuvieron como propósito instalar el sistema en su entorno final de operación. Las actividades se dispusieron en una única iteración, y fueron las siguientes: *planear y gestionar la iteración*, *desarrollar incremento de la solución*, *finalizar documentación y material de entrenamiento*, *prepararse para la entrega* y *poner la entrega en operación*.

- La actividad de *Planear y gestionar la iteración* se contempló a lo largo de toda la fase.
- La actividad de *desarrollar incremento de la solución* sólo contempló hacer cambios menores en la implementación y añadir funcionalidad no prioritaria. Nueva funcionalidad relacionada, por ejemplo, con otras técnicas de visualización no estipuladas inicialmente deberán esperar a un ciclo de desarrollo subsecuente.
- Por su parte, la actividad denominada *finalizar documentación y material de entrenamiento* requiere concluir con la documentación que se empezó en la fase de construcción.

Actividad	Tarea	Productos de entrada	Productos de salida
Planear y gestionar la iteración	Planear la iteración	Lista de pendientes	Lista de riesgos, lista de pendientes
	Gestionar la iteración	Lista de riesgos, lista de pendientes	Lista de riesgos, lista de pendientes
	Evaluar resultados	Lista de pendientes	Lista de pendientes
Desarrollar incremento de la solución	Diseñar la solución	Modelos de casos de uso, documentación arquitectónica	Diseño
	Implementar la solución	-	Implementación
	Implementar pruebas de desarrollador	-	Pruebas de desarrollador
	Correr pruebas de desarrollador	Pruebas de desarrollador	-
	Integrar y crear "build"	-	Build
Finalizar documentación y material de entrenamiento	Desarrollar documentación del producto	-	Documentación del producto
	Desarrollar documentación de usuario	-	Documentación de usuario
	Desarrollar documentación técnica	-	Documentación técnica
Prepararse para la entrega	Desarrollar anuncio para la entrega	-	Anuncio de entrega
Poner la entrega en operación	Empacar la entrega	-	Entrega
	Verificar éxito de la entrega	Entrega	-
	Anunciar la entrega	Anuncio de entrega	-

Cuadro 3.5: Actividades y tareas efectuadas en la fase de transición.

- En lo que se refiere a la actividad de *prepararse para la entrega*, únicamente se contempló informar a los usuarios del LINI (a través de correo electrónico, por ejemplo) que el sistema está finalmente listo para ser utilizado y que su retroalimentación será bienvenida.
- Finalmente, la actividad llamada *poner la entrega en operación* requirió detallar claramente los pasos a seguir para instalar el sistema en su entorno final de operación, para finalmente ejecutarlos. La instalación y ejecución de los elementos de software que conforman el sistema deben ser congruentes con las vistas de asignación de la arquitectura generadas durante la ejecución de ADD.

Las actividades para la fase de transición están enlistadas en el cuadro 3.5.

3.3. Herramientas

3.3.1. Manejo del proyecto

Todas las tareas estipuladas para cada actividad de la instancia de OpenUP, exceptuando aquéllas relacionadas con la planeación de iteraciones, han de ser calendarizadas y deben tener un tiempo límite asignado para su realización. Para llevar la adecuada gestión de fechas y

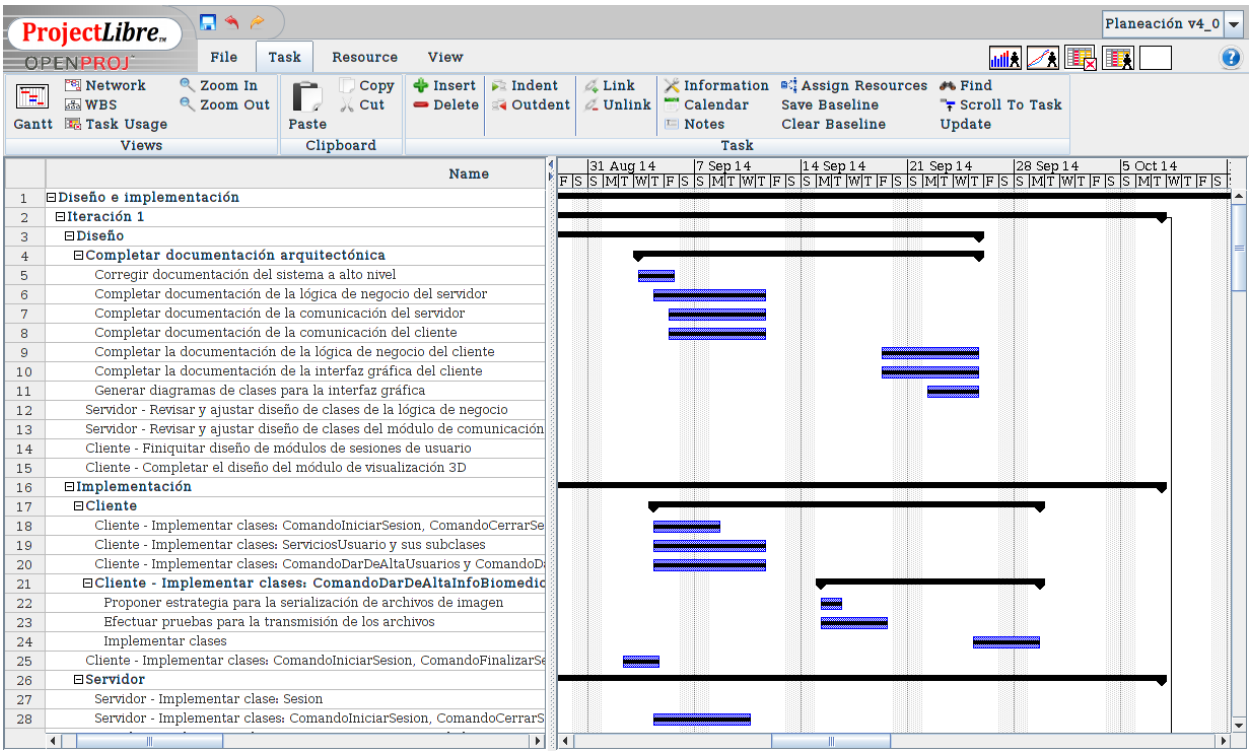


Figura 3.5: ProjectLibre, el software empleado para gestionar fechas y tiempos de cada actividad de la instancia de OpenUP.

tiempos para las tareas se empleo la aplicación ProjectLibre [75], que es gratuita, de código abierto, y muy similar a Microsoft Project. Su uso es bastante simple e intuitivo; permite visualizar la organización de las tareas a modo de diagrama de Gantt, señalar porcentajes de avance, incluir notas y uso de recursos, entre otras funciones. La Figura 3.5 muestra una captura de pantalla mostrando algunas de las actividades de la fase de Construcción gestionadas mediante ProjectLibre.

3.3.2. Modelado

El modelo de casos de uso, las relaciones entre módulos y componentes así como sus propiedades fueron modeladas mediante el estándar UML. UML (*Unified Modeling Language*) es un lenguaje gráfico estandarizado para visualizar, especificar, construir y documentar artefactos de sistemas de software. Fue creado a lo largo de la década de los noventa por Grady Booch, James Rumbaugh e Ivar Jacobson [76], y hasta la fecha sigue vigente como una de las principales herramientas para modelado de software. Para generar los diagramas UML utilizados en este proyecto se empleó el software StarUML en su versión 5.02.1507; la versión más actual de éste puede encontrarse en http://staruml.io/. StarUML es una herramienta bastante completa que permite generar cualquier diagrama definido por UML, y cuenta con funcionalidad para generación de código en varios lenguajes de programación, ingeniería inversa (generación de diagramas a partir de código), entre otras. La Figura 3.6 ilustra la apariencia general de StarUML.

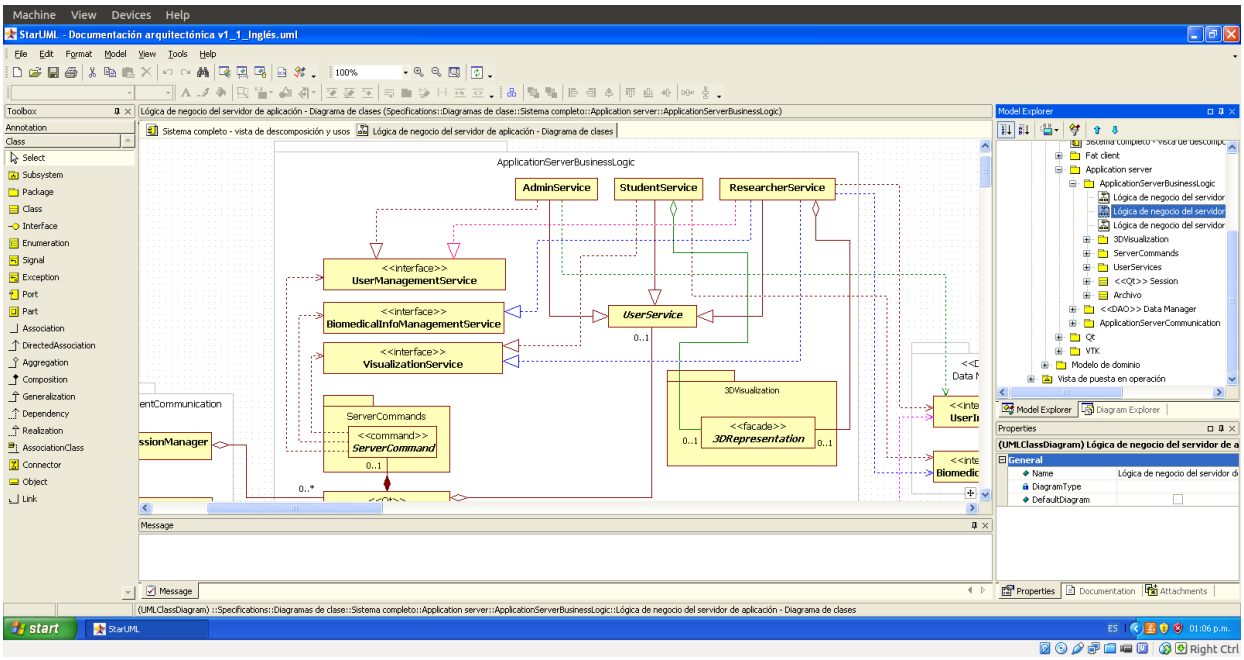


Figura 3.6: StarUML, el software empleado para modelar el sistema de visualización.

3.3.3. Herramientas de implementación

Se empleó la biblioteca de software VTK versión 6.0.0 para efectuar los procesos de visualización, mientras que Qt versión 4.7.4 se utilizó para el diseño de la interfaz gráfica y de los módulos de comunicación; la decisión de emplear estas herramientas se tomó con base en la evidencia arrojada por el EMS, y ambas se describieron en la Sección 2.3.2. La codificación se efectuó en lenguaje C++, mismo en el que están implementadas las bibliotecas mencionadas. Se usó un entorno de desarrollo integrado (comúnmente conocidos como IDE, *Integrated Development Environment*) denominado Qt Creator versión 3.0.1 para administrar todos los archivos de código fuente; este entorno de desarrollo hace uso del compilador de C++ contenido en GCC versión 4.6.3. La Figura 3.7 presenta a Qt Creator mostrando parte de la implementación de una de las clases del sistema de visualización.

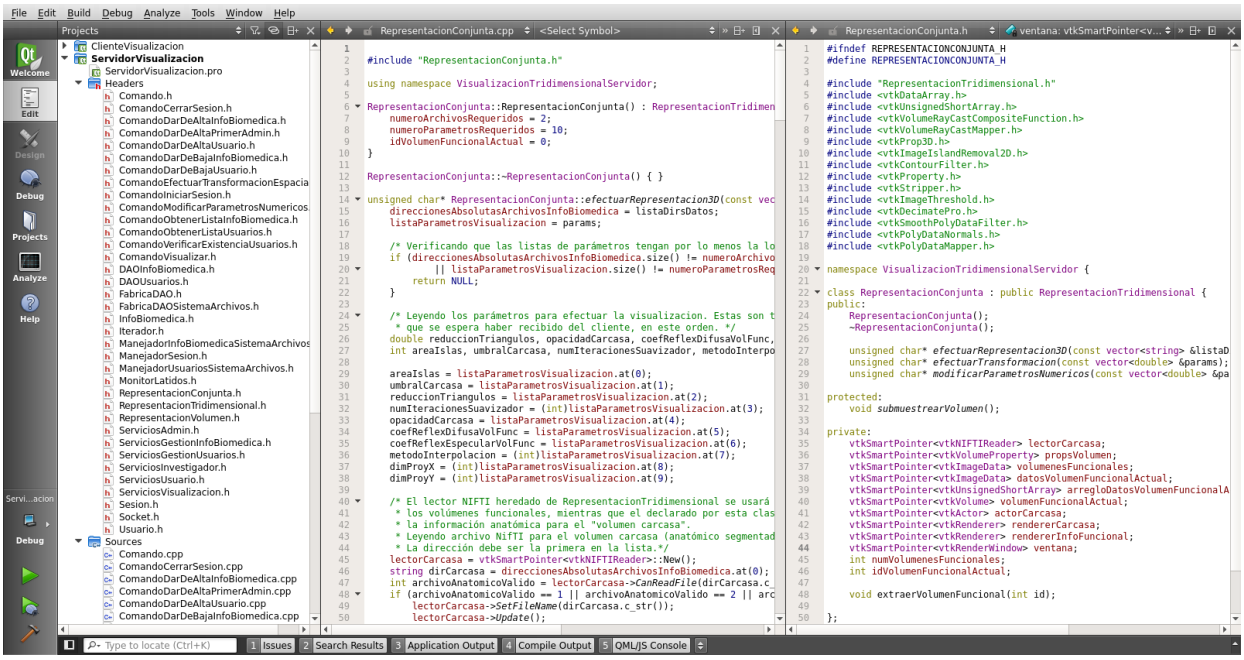


Figura 3.7: Qt Creator, el entorno de desarrollo empleado para la implementación del sistema de visualización.

Capítulo 4

Arquitectura del sistema

En este capítulo se presentan las vistas arquitectónicas del sistema de visualización 3D, así como los requerimientos funcionales y no funcionales que guiaron su diseño.

4.1. Arquitectura inicial

4.1.1. Requerimientos recopilados inicialmente

Las necesidades iniciales que el sistema debe satisfacer fueron registradas en un *documento de visión*, estableciendo de este modo los requerimientos funcionales y no funcionales que, en principio, debían considerarse para estipular una propuesta inicial de arquitectura. La descripción general del sistema plasmada en el documento es la siguiente:

> El sistema de software propuesto será desarrollado con una metodología centrada en arquitectura, de tal forma que se tengan mejores garantías para el cumplimiento de los requerimientos de calidad y de funcionalidad. En cuanto a la última, se requiere la implementación de la visualización tridimensional de imágenes cerebrales, específicamente de la representación volumétrica de información anatómica y funcional del cerebro, así como la representación en superficie de algún parámetro del EEG cuantitivo. Las imágenes cerebrales pueden ser de tipo anatómico (IRM) o funcional (IRMf), pudiendo estar guardadas en formatos convencionales como JPEG, PNG, TIFF, u organizadas en formatos complejos como DICOM o NifTI. Pueden incorporarse funciones básicas como escalamiento de tamaño, rotación o reducción de ruido, además de la representación tridimensional; la mayoría de los parámetros que influyen en dichos algoritmos deberán ser modificables por el usuario. Se espera que las imágenes a emplear ya estén segmentadas, es decir, que ya se hayan etiquetado las estructuras anatómicas de interés, como materia gris, materia blanca, líquido cefalorraquídeo, etcétera; en el caso de las imágenes e información funcional se espera que estén registradas con la información anatómica. [...] Los datos biomédicos y de usuarios estarán siempre disponibles en un servidor, y el sistema deberá brindar la posibilidad de leerlos cuando sea requerido. El sistema debe permitir diferenciar entre usuarios, restringiendo o permitiendo acceso a determinadas opciones y funciones de visualización y manejo de información biomédica según el tipo de usuario en cuestión. Concretamente, la gestión de información de usuarios debe estar bajo observación de un administrador, el cual debe haberse identificado ante el sistema (iniciado sesión) antes de efectuar modificaciones en los registros de dicha información.

El documento de visión expone un resumen de las necesidades generales que el sistema debe satisfacer, y son las siguientes:

1. **Visualización rápida**: la reconstrucción tridimensional debe hacerse en tiempos reducidos y ser manipulada fácil e intuitivamente.
2. **Arquitectura flexible**: el sistema debe permitir la futura incorporación de otras funcionalidades, ya sea de procesamiento de imágenes o de señales, lectura de diferentes formatos de imagen, así como la posibilidad de incorporar y modificar con sencillez las librerías de software empleadas para la visualización.
3. **Visualización de estructuras**: el sistema debe permitir la visualización de las estructuras anatómicas que el usuario escoja, o bien la representación de superficies formadas a partir de algún parámetro particular del EEG.
4. **Privilegios de usuario**: ciertas funciones y configuraciones del sistema sólo podrán ser establecidas y modificadas por el administrador del sistema; es al momento de instalarlo cuando se determina quién es este administrador.
5. **Interfaz gráfica sencilla e intuitiva**: por el bien de la satisfacción de los usuarios, el sistema debe contar con una interfaz gráfica sencilla pero atractiva.
6. **Acceso a datos y funcionalidad remota**: los datos biomédicos siempre deben estar disponibles para sus propietarios. Éstos podrán acceder, modificar y actualizar y trabajar con esos datos contenidos en un servidor remoto.

El entorno de puesta en operación del sistema es el LINI, que es un laboratorio en donde diariamente convergen estudiantes de licenciatura, estudiantes de posgrado y profesores investigadores. Considerando esto y los requerimientos anteriormente enlistados se llegó a la conclusión de que son cuatro los actores que habrán de interactuar con el sistema: *administrador*, *investigador*, *estudiante* y *repositorio de datos*. Los tres primeros son los actores primarios del sistema; el administrador es el que garantiza la adecuada y constante operación del sistema; el investigador emplea el sistema para apoyar su labores académicas y de investigación; y el estudiante lo utiliza con fines didácticos. Por su parte, el repositorio de datos es un actor secundario que almacena y brinda acceso tanto a la información biomédica como a la lista de usuarios que utilizan el sistema. En el Cuadro 4.1 se presentan descripciones detalladas de todos los actores.

4.1.2. Casos de uso principales

Tres casos de uso principales fueron identificados: *manejar sesión*, *gestionar registros* y *visualizar*. Todos son directrices arquitectónicas, pues representan las interacciones básicas que los interesados esperan realizar con el sistema. La Figura 4.1 ilustra la relación entre los actores y los casos de uso; ésta constituye la vista más general del modelo de casos de uso.

4.1.2.1. Caso de uso – Manejar sesión

La Figura 4.2 ilustra las relaciones que este caso de uso tiene con otros. Nótese la relación expresada con una línea punteada y con estereotipo indicado por el término «include». Éste indica una relación de inclusión en la que el caso de uso del que sale la flecha incluye a los casos de uso apuntados por la misma; en otras palabras, estos casos de uso agregados son parte esencial de la secuencia lógica que sigue el caso de uso que los contiene, y sin ellos no estaría bien definido. Esta relación y puede apreciarse mejor en el diagrama de actividad para

Actor	Descripción	Casos de uso relacionados
Usuario	Actor abstracto. Representa los atributos comunes a todos los demás actores humanos	(Ninguno)
Investigador	El investigador requiere observar, medir y analizar la información biomédica necesaria para cumplir con sus responsabilidades académicas y/o docentes. Asimismo, puede compartir dicha información con determinados colegas y estudiantes para que también puedan emplearla con el sistema.	– Manejar sesión. – Gestionar registros. – Visualizar.
Administrador	Es un usuario autorizado para tomar decisiones cuando ocurran eventos inesperados, establecer o modificar las opciones de configuración importantes para el funcionamiento general del sistema, y realizar gestión de registros de usuario e información biomédica en general.	– Manejar sesión. – Gestionar registros.
Alumno	El alumno emplea el sistema para fines didácticos, por lo que el acceso a la funcionalidad del mismo es más restringido. Únicamente tiene acceso a la información biomédica que un investigador haya autorizado.	– Manejar sesión. – Visualizar.
Repositorio de datos	Contiene la información biomédica y los registros de todos los usuarios del sistema. Debe permitir añadir, eliminar y remover información; en el caso de los registros de usuarios, también editar información.	– Manejar sesión. – Gestionar registros.

Cuadro 4.1: Descripción de actores.

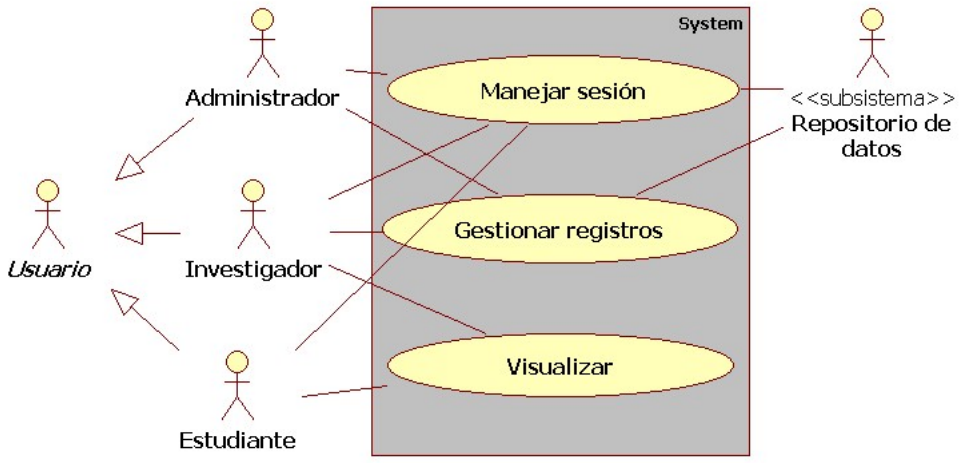


Figura 4.1: Casos de uso de alto nivel

el caso de uso "Manejar sesión", expuesto por la Figura 4.3, en el que también se ilustra el flujo principal y los subflujos (flujos alternos) que indican la secuencia lógica de dicho caso de uso.
A continuación se resume el caso de uso a partir de la información en el documento de modelo de casos de uso generado para este proyecto.

- **Actores involucrados:** usuario, repositorio de datos.
- **Propósito:** describir el proceso del manejo de la sesión de usuario desde que éste comienza a trabajar con el sistema y hasta que concluye.
- **Resumen:** El usuario se identifica ante el sistema para acceder a él. Una vez iniciado el sistema, el usuario se dedicará a sus tareas y en cualquier momento puede elegir finalizar su sesión. El sistema verifica constantemente si hay inactividad; si este es el caso, el sistema se bloquea por lo que el usuario debe restaurar su sesión al reanudar actividades. Si la sesión activa finaliza en algún momento se dará la opción de iniciar otra.
- **Flujo principal:** el sistema inicia e inmediatamente se ejecuta el caso de "Iniciar sesión". Si la sesión se inicia de forma correcta se pone en marcha un temporizador que habrá de monitorizar al sistema para detectar periodos largos de inactividad. Mientras el usuario trabaja puede en cualquier momento elegir dar su sesión por concluida, ante lo cual se ejecuta el caso de uso "Finalizar sesión". Sólo hasta que la sesión abierta se cierre podrá abrirse una nueva.
- **Subflujos:**
  - **Cierre de sesión por inactividad:** si el temporizador detecta inactividad, la sesión actual se bloquea y el sistema ejecutará el caso de uso Restaurar sesión cuando haya actividad nuevamente. Si la sesión se restaura con éxito el temporizador reinicia y se permite el uso del sistema nuevamente, y en caso contrario se negará el acceso al mismo.
- **Excepciones:** ninguna.
- **Requerimientos no funcionales asociados:**
  - El sistema bloqueará una sesión luego de 30 minutos de inactividad (seguridad).
  - El sistema sólo podrá emplearse si hay una sesión activa (seguridad).

4.1.2.2. Caso de uso – Gestionar registros

La Figura 4.4 ilustra las relaciones que este caso de uso tiene con otros. Aquí se hace uso de otra relación denotada por las líneas punteadas con el término «extends» sobre ellas; esta es la relación de extensión entre casos de uso, y establece que los casos de uso de los que parte la flecha extienden a aquellos apuntados por la misma. El caso de uso extendido puede incorporar el comportamiento de otros casos de uso conforme se requiera, pero éstos no son fundamentales para su definición; es decir, expresan comportamiento opcional que puede ser modificado, eliminado o, a su vez, extendido sin repercusiones para el caso de uso base. Esta relación y puede apreciarse mejor en el diagrama de actividad para el caso de uso "Gestionar registros", expuesto por la Figura 4.5, en el que también se ilustra mejor el flujo principal que indica la secuencia lógica de dicho caso de uso.

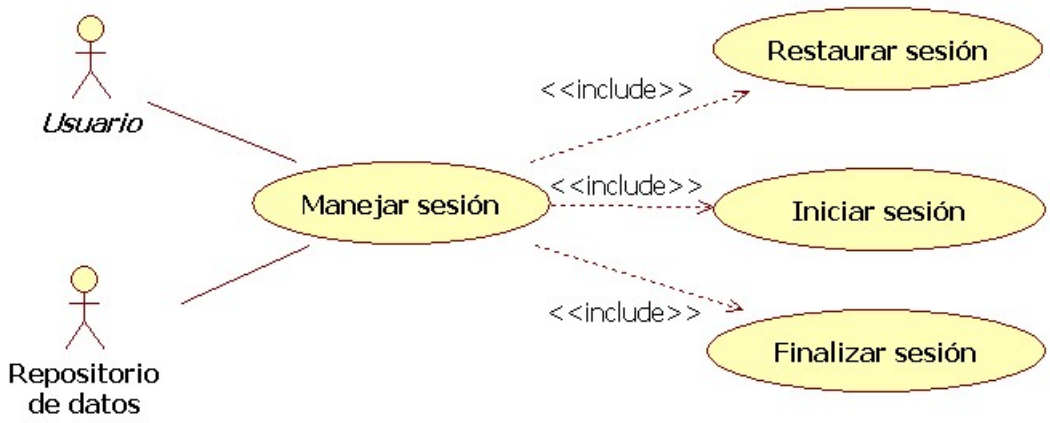


Figura 4.2: Relaciones del caso de uso "Manejar sesión" con los actores y con otros casos de uso.

A continuación se describe textualmente el caso de uso tal cual se reportó en el documento de modelo de casos de uso generado para este proyecto.

- **Actores involucrados:** administrador, investigador, repositorio de datos.
- **Propósito:** permite guardar, eliminar, leer y modificar información en el repositorio de datos.
- **Resumen:** el sistema recupera y enlista los registros que el usuario haya dado de alta en el repositorio de datos. Los registros pueden información biomédica, de usuarios o cualquier otro tipo de información perteneciente o relacionada con el usuario en cuestión. Asimismo, se le permite dar de alta, dar de baja, consultar o modificar aquellos registros.
- **Flujo principal:** el sistema verifica que haya comunicación con el repositorio, y si la hay se despliega una pantalla listando todos los registros dados de alta por el usuario. Asimismo, la pantalla le presentará las opciones para dar de alta, dar de baja, modificar datos y consultar datos del repositorio.
  Si se opta por dar de alta se ejecuta el caso de uso "Dar de alta"; si se elige dar de baja se ejecuta el caso de uso "Dar de baja"; si se selecciona modificar datos se ejecuta el caso de uso "Modificar datos"; finalmente, si se escoge la opción consultar datos se ejecuta el caso de uso "Consultar datos".
- **Subflujos:** Ninguno.
- **Excepciones:** Ninguna.
- **Requerimientos no funcionales asociados:**
  - Si la información a dar de alta es la de un nuevo usuario entonces la información requerida deberá estar conformada por un nombre de inicio de sesión, un nombre de pila, un correo electrónico, y otros dos campos para la contraseña y la repetición de la misma (seguridad).
  - No puede haber usuarios con el mismo nombre de inicio de sesión (seguridad).
  - La contraseña especificada deberá ser de al menos 8 caracteres de longitud y contener un número como mínimo (seguridad).

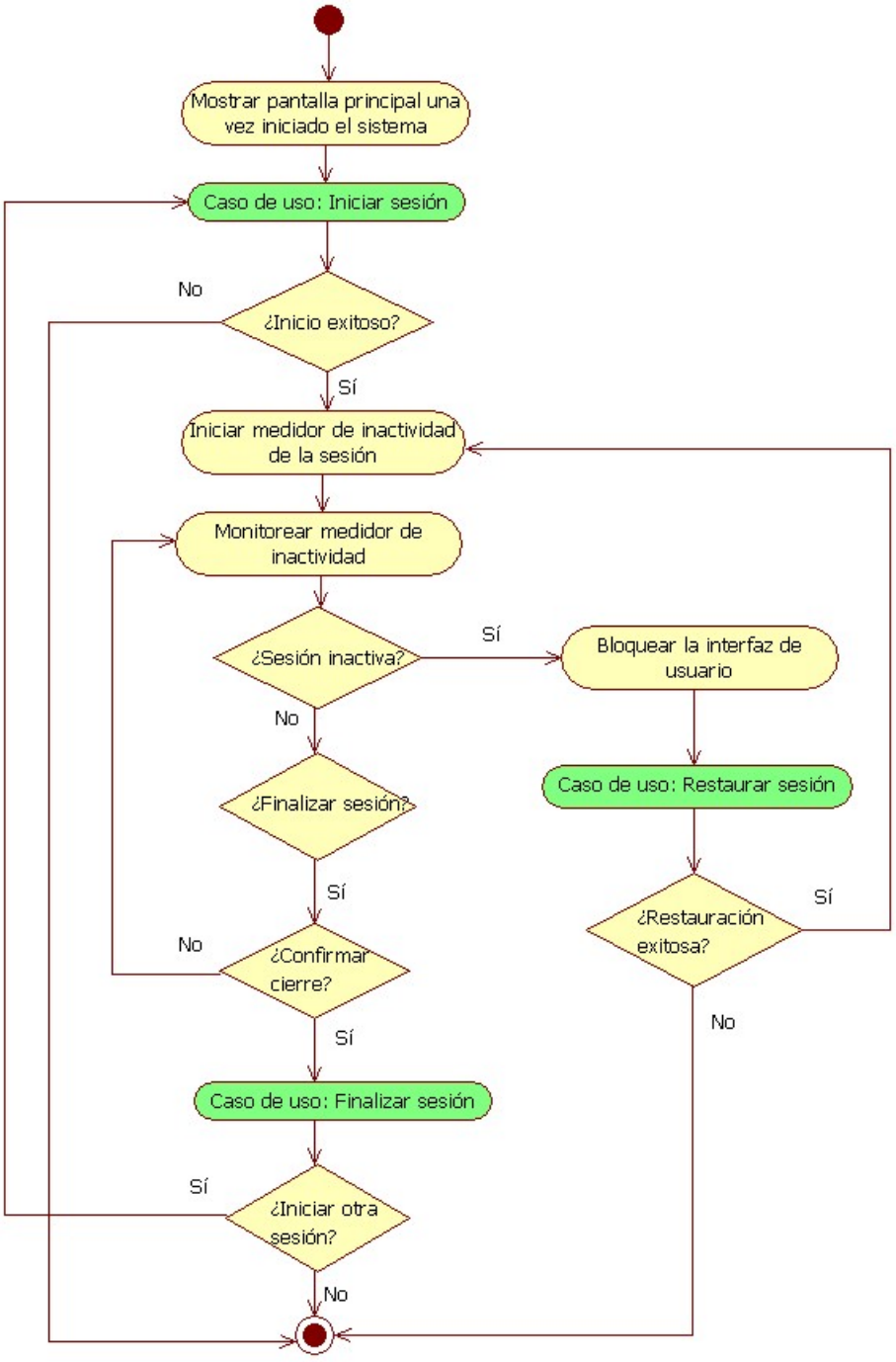


Figura 4.3: Diagrama de actividad para el caso de uso “Manejar Sesion”.

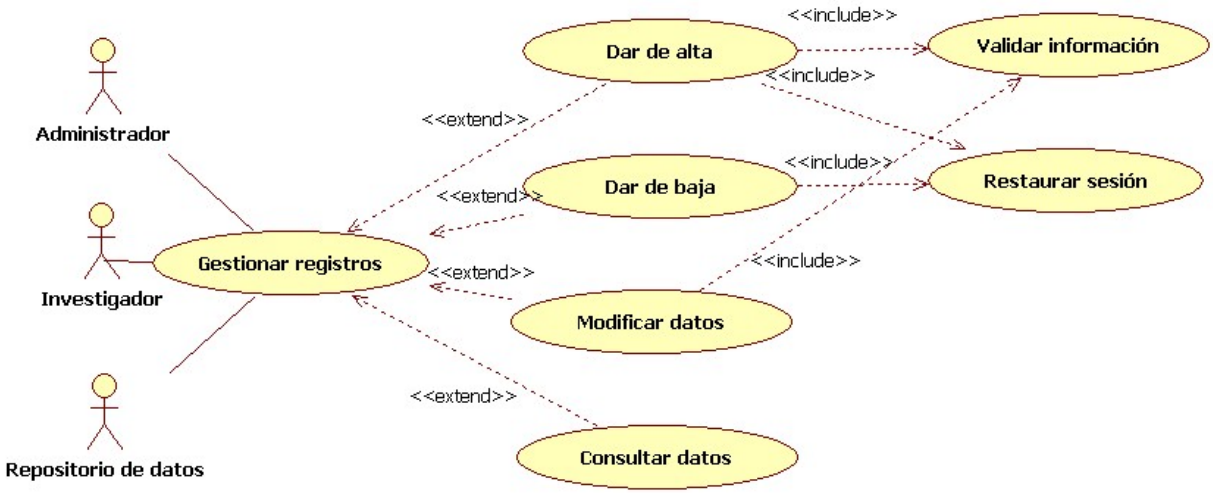


Figura 4.4: Relaciones del caso de uso "Gestionar registros" con los actores y con otros casos de uso.

- El sistema debe indicar al usuario qué tan robusta es la contraseña que propone (usabilidad, seguridad).

4.1.2.3. Caso de uso – Visualizar

La Figura 4.6 ilustra las relaciones que este caso de uso tiene con otros. Por su parte, la Figura 4.7 expone dichas relaciones dentro de la secuencia lógica de actividades. A continuación se describe textualmente el caso de uso tal cual se reportó en el documento de modelo de casos de uso generado para este proyecto.

- **Actores involucrados:** investigador, alumno.
- **Propósito:** generar y desplegar en pantalla la representación 3D de uno o más conjuntos de información biomédica.
- **Resumen:** se presentan al usuario las opciones para efectuar representaciones volumétricas: representación anatómica, representación conjunta y mapeo de EEG; luego, debe escoger qué es lo que el sistema debe realizar.
- **Flujo principal:** en pantalla principal se presentan al usuario las opciones para la visualización tridimensional: efectuar representación anatómica, efectuar representación conjunta, efectuar mapeo de EEG. Si se elige efectuar representación anatómica, se ejecuta el caso de uso "Efectuar representación anatómica"; si se opta por efectuar representación conjunta, se ejecuta el caso de uso "Efectuar representación conjunta"; si se selecciona efectuar mapeo de EEG, se ejecuta el caso de uso "Efectuar mapeo de EEG".
- **Subflujos:** Ninguno.
- **Excepciones:** Ninguna.
- **Requerimientos no funcionales asociados:**

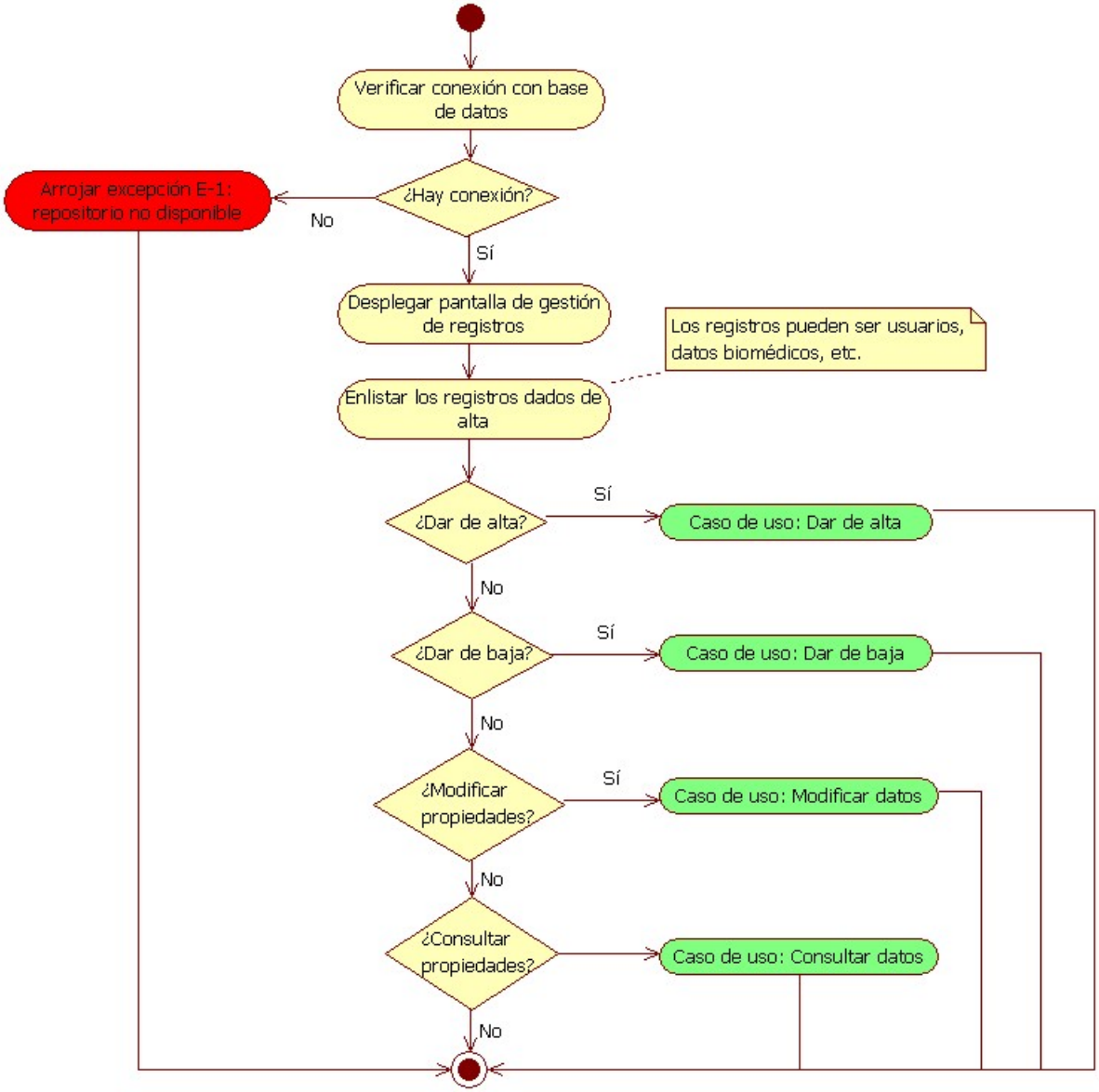


Figura 4.5: Diagrama de actividad para el caso de uso "Gestionar registros".

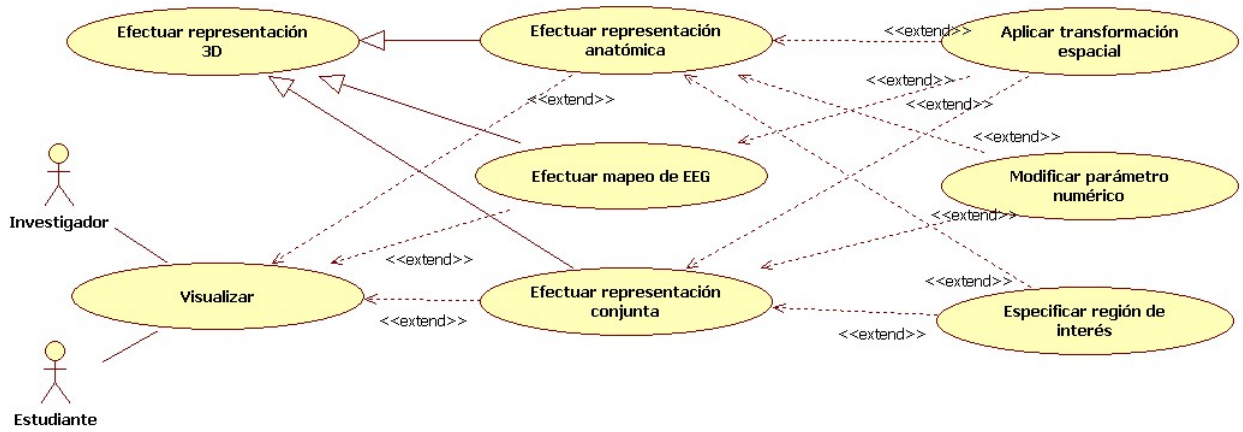


Figura 4.6: Relaciones del caso de uso "Visualizar" con los actores y con otros casos de uso.

- Las opciones para efectuar representaciones 3D deben ser rápidamente accesibles a través de la interfaz gráfica, con un máximo de 3 clicks (usabilidad).

4.1.3. Vista arquitectónica inicial

La Figura 4.8 establece una vista de alto nivel de los módulos que pudieran conformar al sistema. Los puntos importantes respecto a esa propuesta están relacionados con los requerimientos funcionales y no funcionales mencionados anteriormente, y son los siguientes:

- Se propuso que un estilo arquitectónico cliente-servidor debido a que otorga varias ventajas:
  - La funcionalidad y los datos biomédicos siempre estarán a disposición de los usuarios que tengan acceso a un cliente.
  - El servidor puede contar con hardware gráfico para efectuar las tareas de visualización, liberando a los clientes de requerir dicho hardware y favoreciendo al desempeño.
  - El cliente puede emplear la información provista por el servidor como mejor convenga. Esto quiere decir que el cliente puede adaptarse para leer, guardar, transformar o desplegar los datos recibidos como fuese necesario, y dichas adaptaciones serían transparentes para el servidor.
- Al desacoplar la lógica de negocio y la interfaz gráfica (GUICliente) se favorece la modificabilidad solicitada. Lo mismo ocurre al desacoplar la lógica de negocio y el módulo de visualización 3D en el lado del servidor de aplicación, y al desacoplar éste del manejador de datos.
- El módulo de visualización 3D encapsula a las herramientas necesarias para efectuar las tareas de visualización. Dichas herramientas deben favorecer, principalmente, al rendimiento del sistema; todo cambio en ellas se limitará únicamente a ese módulo y no afectará a los demás.

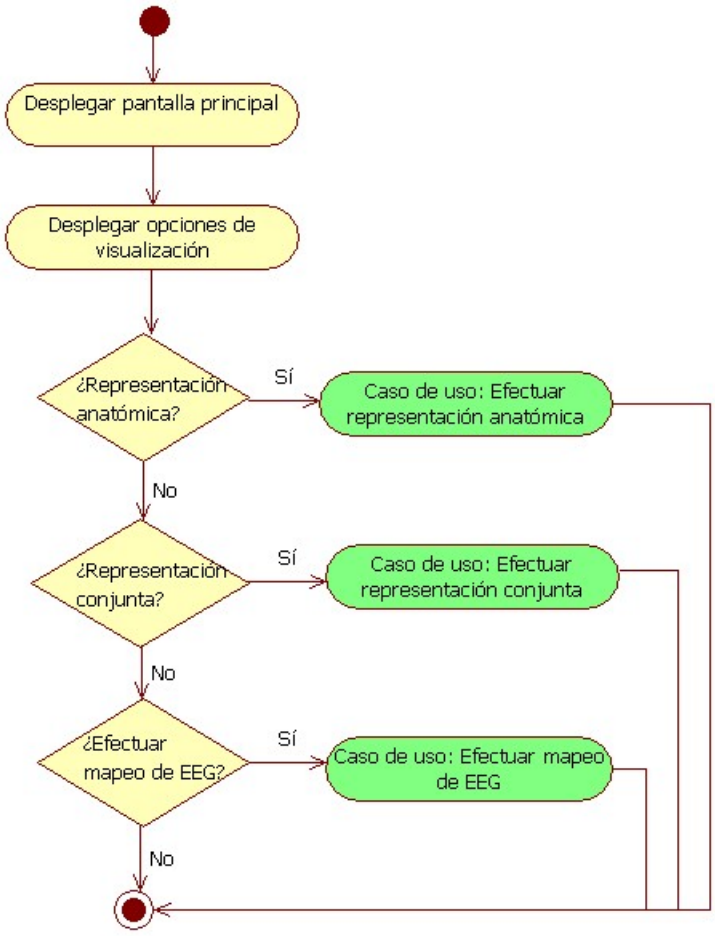


Figura 4.7: Diagrama de actividad para el caso de uso "Visualizar".

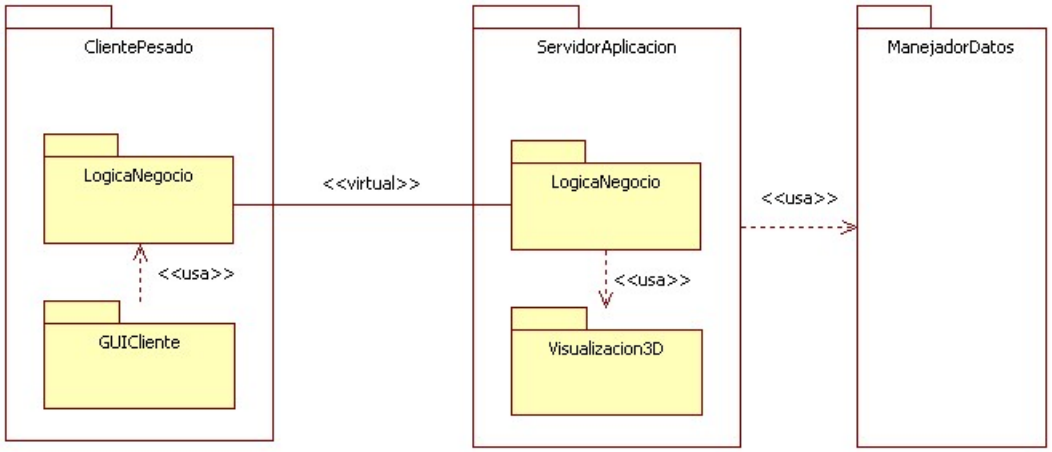


Figura 4.8: Vista de módulos inicial del sistema, propuesta a partir de los requerimientos iniciales.

- El sistema es escalable a varios servidores y, por lo tanto, a un gran número de clientes.

Esta vista resultó útil para comunicar todo lo anterior a los principales interesados en el sistema durante la realización de QAW, del cual se obtuvieron los escenarios de atributos de calidad que complementan los requerimientos del sistema.

4.2. Escenarios de atributos de calidad

Los escenarios de atributos calidad se obtuvieron de la ejecución de QAW (ver Sección 3.1.1.1). Se recibió un total de 18 escenarios de atributos de calidad que constituyeron la entrada para el paso de consolidación de los mismos; la consolidación redujo el número a un total de 16 escenarios. Cada participante contó con 6 votos para distribuir entre los escenarios, teniendo como la restricción de no asignar más de 3 votos a un único escenario. El Cuadro 4.2 presenta los escenarios de atributos de calidad obtenidos del taller, ordenados de mayor a menor con respecto al número de votos asignados por los participantes. Aquéllos escenarios que surgieron de haber fusionado dos o más escenarios contienen el término "(*consolidado*)" antes de su descripción; más adelante se exponen los escenarios individuales que les dieron origen y por qué fue necesaria su fusión. Además del número de votos, en cada escenario se indica el principal actor humano involucrado y el atributo de calidad relacionado. Éste último se determinó con base en las definiciones y ejemplos proporcionados por Bass, Clements y Kazman en [3]. Nótese que los primeros dos escenarios contaron con 7 votos y los siguientes dos con 4, mientras que todos los demás tienen 2 o menos. Es por ello que sólo los primeros cuatro escenarios fueron considerados directrices arquitectónicas.

Cuadro 4.2: Escenarios de atributos de calidad.

#	Descripción	Atributo de calidad	Actor	Número de votos
1	(*Consolidado*) **Estímulo:** surge la necesidad de variantes funcionales en ciertos tipos de procesamiento. **Respuesta:** el sistema permite integrar la variante funcional. **Ambiente:** entorno de desarrollo.	Modificabilidad	Desarrollador	**7**
2	**Estímulo:** se genera un nuevo módulo para el sistema y lo integro al mismo. **Respuesta:** la compilación se restringe al módulo nuevo y una o dos bibliotecas más de interfaz (idealmente ninguna). **Ambiente:** desarrollando código para nueva funcionalidad.	Modificabilidad	Desarrollador	**7**
3	**Estímulo:** interrupción de energía eléctrica o se interrumpe servicio de red. **Respuesta:** no se corrompen los datos de origen. **Ambiente:** operación normal del sistema	Disponibilidad	Investigador	**4**
4	**Estímulo:** se desarrolla un nuevo formato de archivos. **Respuesta:** el sistema reconoce el nuevo formato y lo interpreta cuando es requerido. **Ambiente:** desarrollo de extensiones y su agregación a la aplicación.	Modificabilidad	Desarrollador	**4**
5	**Estímulo:** el sistema se queda inactivo por mas de 10 minutos (sin proceso e interacción con el usuario). **Respuesta:** que bloquee la sesión del sistema. **Ambiente:** operación normal del sistema.	Seguridad	Investigador	**2**
6	(*Consolidado*) **Estímulo:** uso del ratón para interactuar con la visualización. **Respuesta:** el sistema reconoce los comandos del ratón, y actúa adecuadamente. **Ambiente:** operación normal del sistema.	Usabilidad	Alumno	**2**
7	**Estímulo:** el sistema efectúa el proceso para llevar a cabo una reconstrucción. **Respuesta:** el sistema identifica que existen recursos de un chip gráfico y están disponibles. **Ambiente:** operación normal del sistema.	Rendimiento	Alumno	**2**

Continúa...

8	**Estímulo:** el sistema se intenta usar por un usuario con contraseña desconocida en cinco intentos. **Respuesta:** el sistema bloquea la cuenta por un día. **Ambiente:** uso normal del sistema para iniciar sesión.	Seguridad	Administrador	**2**
9	**Estímulo:** el sistema se intenta usar por un usuario desconocido. **Respuesta:** notificación al administrador de intento de ataque. **Ambiente:** uso normal del sistema para iniciar sesión o en estado operativo.	Seguridad	Administrador	**2**
10	**Estímulo:** el administrador solicita prueba de operación y/o configuración del sistema. **Respuesta:** el sistema efectúa la prueba solicitada e informa el resultado de la prueba. **Ambiente:** uso del sistema en sesión de administrador.	Facilidad de pruebas	Administrador	**2**
11	**Estímulo:** el administrador usa el sistema para hacer una alta, bajo o cambio de un usuario. **Respuesta:** el sistema realiza la tarea en un tiempo no mayor a 10 segundos. **Ambiente:** uso del sistema en sesión de administrador.	Rendimiento	Administrador	**2**
12	**Estímulo:** el sistema se usa para configuración de usuarios o del sistema mismo, desde diferentes computadoras. **Respuesta:** la visualización y funcionamiento del sistema es adecuado en cualquier computadora. **Ambiente:** uso del sistema en sesión de administrador.	Disponibilidad	Administrador	**2**
13	**Estímulo:** el administrador usa el sistema para hacer una alta, bajo o cambio de un usuario. **Respuesta:** toda la información referente a un usuario debe visualizarse en una sola pantalla. **Ambiente:** uso del sistema en sesión de administrador.	Usabilidad	Administrador	**2**
14	**Estímulo:** interrupción de energía eléctrica o se interrumpe servicio de red. **Respuesta:** se pueda continuar el proceso desde donde se interrumpió. **Ambiente:** operación normal del sistema	Disponibilidad	Investigador	**1**
15	**Estímulo:** deshacer en un paso la última transformación realizada. **Respuesta:** regresa a la vista anterior. **Ambiente:** operación normal del sistema	Usabilidad	Alumno	**1**

Continúa...

16	**Estímulo:** se conmuta entre visible/invisible los elementos de visualización: superficie, volumen, mapa de potencial, fuentes, etc. **Respuesta:** el sistema demora lo menos posible en quitar y poner los objetos en la escena. **Ambiente:** operación normal del sistema	Rendimiento	Investigador	**0**

El escenario #4 se planteó al fusionar los siguientes escenarios:

#	Descripción
4–a	**Estímulo:** uso del ratón para interactuar con la visualización. **Respuesta:** que permita interacción mediante el ratón para transformaciones espaciales. **Ambiente:** operación normal del sistema.
4–b	**Estímulo:** el usuario utiliza gestos del mouse (clic doble, scroll, ctrl scroll). **Respuesta:** El sistema reconoce los gestos y actúa adecuadamente. **Ambiente:** operación normal del sistema.

Ambos escenarios son similares pues expresan un requerimiento de usabilidad que debe cumplir el sistema al momento de efectuar interacciones con la representación 3D. Por su parte, el escenario #6 se formuló a raíz de los siguientes:

#	Descripción
6–a	**Estímulo:** reutilización de algoritmos o componentes de software que están probados y tienen un estándar de desarrollo (en un lenguaje diferente al nativo del sistema). **Respuesta:** que permita la incorporación de código externo (sin importar si es compilado o interpretado). **Ambiente:** desarrollo del sistema.
6–b	**Estímulo:** surge la necesidad de variantes funcionales en ciertos tipos de procesamiento. **Respuesta:** el sistema permite integrar la variante funcional. **Ambiente:** desarrollo del sistema.

Estos escenarios describen aspectos importantes de modificabilidad en el sistema. Hay varias razones por las que se consideró muy inconveniente añadir funcionalidad que invoque núcleos de otros lenguajes de programación. Primero, ello impactaría negativamente al rendimiento del sistema. Y segundo, asumiendo un rol de desarrollador, probablemente sea más sencillo tomar el código cuya funcionalidad se quiere añadir y traducirlo directamente al lenguaje en que se codificó el sistema; esto en vez de añadir funcionalidad para compilar N distintos lenguajes.

4.3. Vistas arquitectónicas

4.3.1. Sistema completo

Las vistas arquitectónicas presentadas en esta sección son resultado de la ejecución de ADD (ver Sección 3.1.1.2). Puesto que el primer elemento de software a descomponer es el sistema completo, se procedió a determinar las directrices arquitectónicas que impactarían su diseño.

Para asignar las prioridades a cada uno de los requerimientos funcionales y no funcionales hallados se propuso un esquema que contemplara dos factores: la prioridad que el requerimiento tiene para los interesados y el impacto potencial del requerimiento en la arquitectura. El Cuadro 4.5 expone dicho esquema. El impacto potencial de cada requerimiento no funcional se determinó con base en un estimado del número de módulos requerido por cada uno. Por otra parte, la prioridad para los interesados está basada en el número de votos asignado a cada escenario de atributo de calidad. Por su parte, los requerimientos funcionales se priorizaron con base en los expresado por los interesados a lo largo de las conversaciones efectuadas con ellos. En el Cuadro 4.6 se listan todos los requerimientos funcionales (estipulados como casos de uso) y los no funcionales (expresados en escenarios de atributos de calidad) para el sistema completo, así como sus respectivas prioridades. Aquéllos con prioridad media (M) o alta (A) fueron tomados como directrices arquitectónicas; aparecen marcados en negritas dentro del cuadro.

Para el tercer paso de ADD deben escogerse los conceptos de diseño (tácticas y patrones arquitectónicos y de diseño, tecnologías y otras herramientas) que satisfacen las directrices arquitectónicas. El Cuadro 4.7 exhibe los conceptos que se consideraron potencialmente aplicables para cada una de ellas.

El cuarto paso establece que deben instanciarse los elementos encontrados. Con base en los conceptos de diseño propuestos, se identificó que el sistema completo puede descomponerse en los módulos que se enlistan a continuación; el nombre que se les ha puesto para identificarlos en el diseño arquitectónico aparece encerrado entre paréntesis. Asimismo, las responsabilidades que cada uno debe llevar a cabo se listan debajo de sus respectivos nombres.

- **Interfaz gráfica del cliente pesado (GUICliente)**
  - Desplegar los resultados de los procesos de visualización efectuados por el servidor.
  - Recibir entradas de hardware (teclado, ratón, etc.) para acatar las acciones del usuario.
  - Habilitar el tipo de interfaz gráfica apropiado para el permiso de acceso del usuario.
- **Lógica de negocio del cliente pesado (LogicaNegocioClientePesado)**
  - Gestionar registros de usuarios y de datos biomédicos.
  - Captar y enviar al servidor los parámetros para la visualización de la información biomédica.
  - Validación de datos biomédicos.
  - Preparar las peticiones e información biomédica para que puedan ser enviadas al servidor.
- **Comunicación del cliente pesado (ComunicacionClientePesado)**
  - Enviar peticiones e información al servidor de aplicación.
  - Recibir las respuestas enviadas por el servidor de aplicación.

Prioridad cliente	Impacto en arquitectura	Prioridad asignada	Justificación
A	**A**	**A**	Un requerimiento de gran importancia para los interesados y de alto impacto arquitectónico sólo puede ser considerado una directriz arquitectónica de alta prioridad.
A	**M**	**A**	Aquellos requerimientos considerados muy deseables y que requieren considerar ciertos conceptos de diseño deben ser considerados importantes.
A	**B**	**M**	Los requerimientos que son de importancia para los interesados pero que no tienen mayor injerencia en la arquitectura se tomarán en cuenta. Es probable que se este tipo de requerimientos se refieran, por ejemplo, a cuestiones del diseño de la interfaz gráfica del sistema o al empleo de determinado algoritmo dentro de un módulo.
M	**A**	**M**	Debe realizarse un esfuerzo por incluir en el diseño arquitectónico a aquellos requerimientos medianamente importantes para los interesados, pero si el impacto de incluirlos es muy grande debe darse prioridad a los requerimientos de mayor importancia para aquéllos.
M	**M**	**M**	Aquellos requerimientos no tan prioritarios para los interesados y que son de algún impacto arquitectónico (como la utilización de un componente de software o pieza de hardware específica) pueden ser considerados en el diseño, pero definitivamente se desestimarán si hay opciones más convenientes para lograr atributos de calidad.
M	**B**	**B**	Requerimientos de bajo impacto y moderada importancia para los interesados serán relegados; otros de mayor importancia se considerarán primero.
B	**A**	**M**	El diseño arquitectónico debe ser suficientemente flexible para poder satisfacer otros requerimientos en el futuro, aunque de momento hayan resultado de baja importancia para los interesados.
B	**M**	**B**	Estos requerimientos ceden ante otros de mayor relevancia; probablemente se consideren en ciclos de desarrollo subsecuentes.
B	**B**	**B**	Requerimientos de baja relevancia en general pueden mencionarse, cuando mucho, como trabajo a futuro.
Clave: **A** – Alta, **M** – Media, **B** – Baja			

Cuadro 4.5: Esquema para asignación de prioridades a requerimientos.

Requerimiento	Prioridad cliente	Impacto en arquitectura	Prioridad asignada
Visualizar	A	A	A
Manejar sesión	M	M	M
Gestionar registros	M	A	M
Escenario 1	A	A	A
Escenario 2	A	A	A
Escenario 3	M	B	B
Escenario 4	M	A	M
Escenario 14	B	M	B
Escenario 5	B	B	B
Escenario 6	B	B	B
Escenario 15	B	B	B
Escenario 7	B	B	B
Escenario 16	B	B	B
Escenario 8	B	B	B
Escenario 9	B	M	B
Escenario 10	B	M	B
Escenario 11	B	M	B
Escenario 12	B	M	B
Escenario 13	B	B	B
			Clave: **A** – Alta, **M** – Media, **B** – Baja

Cuadro 4.6: Prioridades de los requerimientos para el módulo sistema completo.

- **Lógica de negocio del servidor de aplicación (LogicaNegocioServidorAplicacion)**
  - Interpretar y ejecutar las peticiones enviadas por el cliente.
  - Efectuar tareas de visualización de información biomédica.
  - Lectura de datos biomédicos.
  - Verificar y otorgar permisos de acceso a los servicios y a los datos biomédicos.
  - Preparar la información que el servidor mandará a los clientes.
- **Comunicación del servidor de aplicación (ComunicacionServidorAplicacion)**
  - Recibir las peticiones y la información biomédica que los clientes envíen.
  - Enviar a los clientes la información que soliciten.
- **Manejador de datos (ManejadorDatos)**
  - Gestionar (añadir, eliminar, modificar) los datos persistentes biomédicos y de usuarios.
  - Otorgar acceso a la información persistente del repositorio de datos.
- **Cliente pesado (ClientePesado)**
  Engloba a los siguientes módulos: interfaz gráfica del cliente pesado, lógica de negocio del cliente pesado y comunicación del cliente pesado.
- **Servidor de aplicación (ServidorAplicacion)**
  Contiene a los siguientes módulos: lógica de negocio del servidor de aplicación y comunicación del servidor de aplicación.

La descomposición inicial del sistema completo se presenta en la Figura 4.9. Se presentan tres grandes unidades de implementación: el módulo del cliente pesado, el módulo del servidor de aplicación y el del módulo del manejador de datos. En general, esta vista presenta un estilo cliente-servidor que otorgue las características de modificabilidad, disponibilidad y seguridad de los datos. Cabe mencionar que no todas las decisiones de diseño resultaron evidentes al nivel actual de diseño; el resto de ellas, así como otras nuevas, se hicieron patentes en iteraciones sucesivas del método ADD. Éstas se exponen en las secciones a continuación.

El quinto paso de ADD involucra efectuar la documentación de las interfaces de los elementos; una de las formas más simples de hacerlo es registrar la información que cada módulo espera recibir como entrada, y qué información genera como salida. El Cuadro 4.8 expone esta información para las interfaces de los módulos hijo identificados para el sistema completo. Esta actividad permitió vislumbrar el tipo de datos que los módulos más generales debían recibir y retornar.

El sexto paso de ADD requiere revisar los casos de uso y los requerimientos de atributos de calidad, aunque en este caso no fue necesario un refinamiento exhaustivo de los mismos.

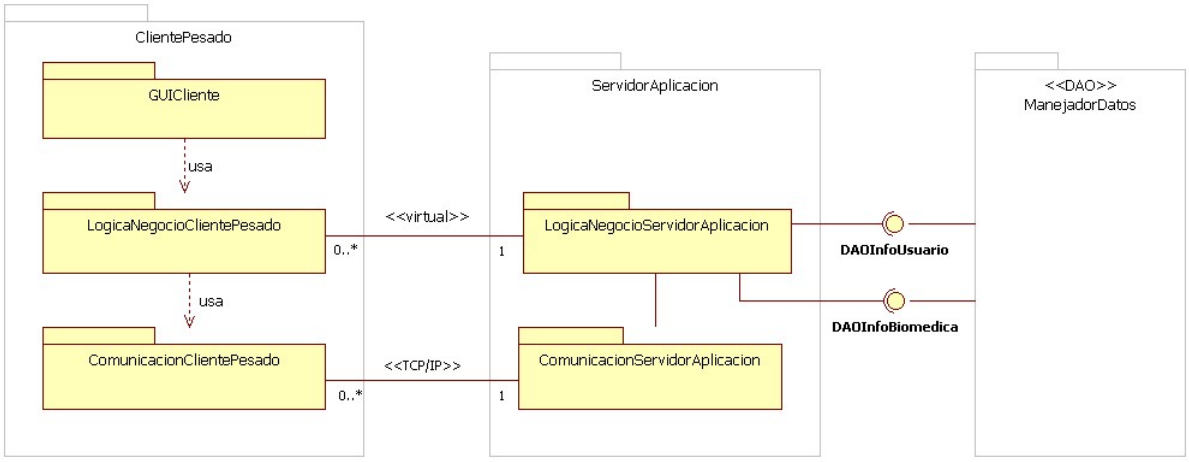
Por su parte, la vista de componentes del sistema completo fue elaborada con base en la de módulos. La Figura 4.10 muestra esta vista, y en ella pueden identificarse tres tipos de componentes: cliente pesado, servidor de aplicación y manejador de datos. Éstos se presentan de forma individual en la Figura 4.11 que constituye la guía para la vista de componentes. Para entender mejor estas figuras debe tenerse en cuenta que los elementos presentados en la vista de componentes son, en realidad, instancias de componentes, y los tipos a los que corresponden dichas instancias son los que exponen en la guía. El mapeo entre la vista de módulos y la de componentes suele ser de varios-a-varios, pero en este caso resultó ser uno-a-uno; por lo tanto, los tipos de componente *ClientePesado*, *ServidorAplicacion* y *ManejadorDatos* corres-

Directriz arquitectónica	Sub-atributo de calidad	Concepto de diseño
Visualizar	Recuperación ante fallas (disponibilidad)	***Tácticas*** – Manejo de excepciones
	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica ***Patrones*** – Cliente-Servidor
	Demanda de recursos (desempeño)	***Tácticas*** – Concurrencia ***Tecnología*** – VTK
	Facilidad de uso (usabilidad)	***Tecnología*** – VTK – QT ***Tácticas*** – Cancelar procesos
Manejar sesión	Seguridad del sistema (seguridad)	***Tácticas*** – Autenticación de usuarios – Limitar exposición – Concurrencia – Perro guardián (watchdog) ***Tecnología*** Hilos
	Seguridad de los datos (seguridad)	***Tácticas*** – Autenticación de usuarios ***Tecnología*** – TCP/IP
Gestionar registros	Resistencia ante ataques (seguridad)	***Tácticas*** – Mantener confidencialidad de los datos
	Seguridad de los datos (seguridad)	***Tecnología*** – TCP/IP – Mecanismos de sincronización de hilos
	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica ***Patrones*** – Data Access Object (DAO)
Escenario 1	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Uso de interfaces ***Patrones*** – Adapter – Facade
Escenario 2	Sencillez para efectuar cambios (modificabilidad)	***Patrón*** – Coherencia semántica – Uso de interfaces
Escenario 4	Sencillez para efectuar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Uso de interfaces ***Patrones*** – Adapter – Facade

Cuadro 4.7: Directrices arquitectónicas identificadas y conceptos de diseño propuestos para el módulo sistema completo

Módulo	Info. entrada	Info. salida
Interfaz gráfica del cliente pesado	– Parámetros de visualización. – Datos para autenticación de usuario. – Otros datos indicados por el usuario mediante el hardware de entrada.	– Resultados de los procesos de visualización, enviados por el servidor. – Resultados de los procesos de visualización o gestión de usuarios. – Mensajes de error provenientes de la lógica de negocio del servidor.
Lógica de negocio del cliente pesado	– Peticiones para acceder a servicios de visualización. – Peticiones para servicios de gestión de datos e info biomédica. – Información biomédica y datos de registros de usuarios que serán enviados al servidor.	– Resultados de los procesos de visualización, enviados por el servidor. – Datos de registros de usuario e información biomédica solicitados al servidor de aplicación. – Mensajes de error relativos al procesamiento de información biomédica, a la validación de información biomédica o registros de usuario. – Mensajes de error provenientes del módulo de comunicación del cliente.
Comunicación del cliente pesado	– Peticiones de servicios para el servidor de aplicación. – Información biomédica y registros de usuarios que serán guardados en el repositorio.	– Información serializada proveniente del servidor. Incluye pero no se limita a: ▪ Resultados del procesamiento efectuado en el servidor de aplicación. ▪ Registros de usuario y datos biomédicos solicitados al servidor de aplicación. ▪ Mensajes de error relativos al procesamiento de información biomédica o a las solicitudes del manejo de datos relacionados con usuarios o info biomédica. ▪ Mensajes de error relativos a la comunicación con el servidor de aplicación.
Comunicación del servidor de aplicación	– Peticiones para la lectura de información recibida por parte los clientes. – Información serializada recibida por parte los clientes. Esta información es equivalente a la de entrada en el módulo de Comunicación del cliente, y también es la que entra al módulo de Lógica de negocio del servidor de aplicación.	– Información serializada proveniente de la lógica de negocio del servidor.
Lógica de negocio del servidor de aplicación	– Información serializada proveniente del cliente. Incluye pero no se limita a: ▪ Peticiones de servicios de visualización. ▪ Peticiones para leer datos sobre registros de usuarios e info biomédica. ▪ Información biomédica y registros de usuario a ser guardados en el repositorio.	– Información serializada que será enviada al cliente. Esta información es equivalente a la de salida en el módulo de Comunicación del cliente pesado.
Manejador de datos	– Peticiones de acceso a registros de usuario e info biomédica. – Información biomédica y registros de usuarios provenientes del cliente.	– Información relativa a registros de usuario y datos biomédicos solicitada por el cliente. – Mensajes de error relativos al intento de acceso a información.

Cuadro 4.8: Documentación de las interfaces de los módulos hijo del sistema completo



.

Figura 4.9: Vista de módulos del sistema completo. La notación de la imagen corresponde al estándar UML.

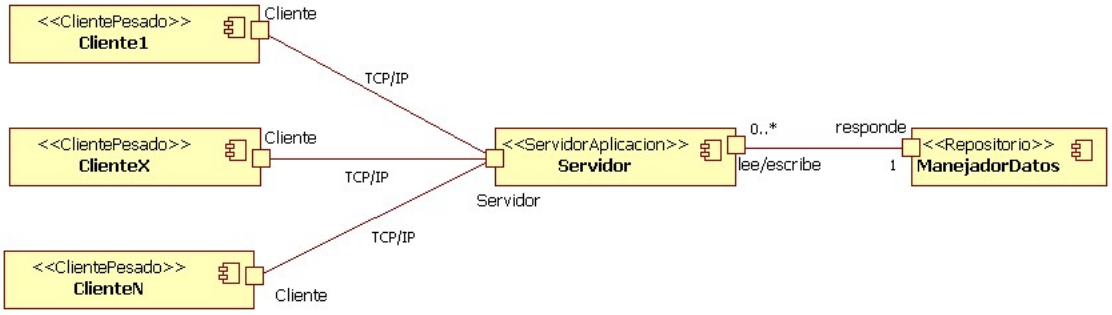


Figura 4.10: Vista de componentes para el sistema completo. La notación corresponde al estándar UML.

ponden a los módulos del mismo nombre (ver Figura 4.9). Tanto en la vista de componentes como en la guía se muestran los puertos de los componentes, pero en la última son representados mediante los símbolos de "paleta" y de "ranura" característicos de las interfaces en UML. De este modo se indica la forma en que el componente se relaciona con su entorno mediante ese puerto en particular, ya sea ofreciendo o requiriendo los servicios de otros.

En lo que respecta a la documentación de la vista de componentes, se registraron las propiedades que brindan (o restringen) atributos de calidad, cómo y qué tanto lo hacen; dicha documentación se efectuó de acuerdo a las pautas establecidas por el método Views and Beyond [77] que también fue propuesto por integrantes del SEI. En la siguiente lista se presentan las propiedades de cada componente; los nombres empleados para identificarlos en el diseño arquitectónico (es decir, los mostrados en la Figura 4.11) aparecen entre paréntesis:

- **Cliente pesado (ClientePesado):**
  - *Concurrencia*: pueden existir varias instancias del componente conectadas a la instancia de *Servidor*.

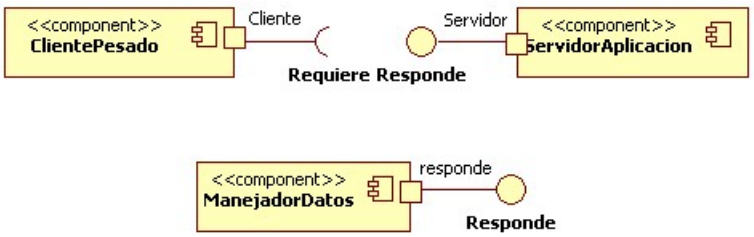


Figura 4.11: Guía de componentes para el sistema completo.

- *Disponibilidad*: el componente intenta periódicamente restablecer contacto con el servidor al detectar problemas en la comunicación.
- *Desempeño*: en el caso de procesos de visualización, las instancias de este componente envían al servidor sólo los parámetros necesarios para efectuar el proceso solicitado. Asimismo, nunca tratan con la información biomédica completa; únicamente se sirven de subconjuntos y/o de versiones submuestreadas de dicha información que son más fácilmente manipulables.
- *Puertos*: a través del puerto del cliente se envían peticiones (parámetros para efectuar alguna tarea) al servidor, y por el mismo puerto se recibe la información de respuesta. Los datos pueden ser texto y/o datos binarios el cual el componente se encargará de interpretar correctamente.

- **Servidor de aplicación (ServidorAplicacion):**
  - *Concurrencia*: el componente puede atender a varias instancias de *ClientePesado*. Cada una es atendida por un hilo (thread) que se genera al establecerse el contacto cliente-servidor. El servidor puede atender a un máximo de 256 instancias de *ClientePesado*.
  - *Desempeño*: el componente procura hacer uso del hardware gráfico disponible en el nodo de puesta en operación; esto redunda en tiempos de procesamiento menores. En caso de efectuar procesos de visualización, el servidor únicamente envía el resultado de la misma al cliente (esto es, una proyección bidimensional para que el cliente presente en pantalla).
  - *Seguridad*: los servicios provistos por este componente se limitarán a aquéllos correspondientes al nivel de acceso del cliente.
  - *Puertos*: el servidor atiende a los clientes a través de un único puerto por el que se reciben las peticiones de éstos; las peticiones constituyen parámetros que la instancia de *ServidorAplicacion* interpreta como comandos a los que responde con información que envía a través del mismo puerto. Además, por cada hilo generado para atender a los clientes también habrá un puerto que se conectará con el de la instancia de *ManejadorDatos* para leer y escribir información.
- **Manejador de datos (ManejadorDatos):**
  - *Concurrencia:* sólo se ejecutará una instancia de este componente para atender a todo el sistema.
  - *Confiabilidad*: se emplean mecanismos de exclusión mutua (mutexes) para evitar la escritura o eliminación simultánea de información almacenada.

  - *Puertos*: el componente recibe solicitudes de lectura y escritura de la información contenida en el repositorio. Devuelve la información solicitada o, en dado caso, los mensajes de error correspondientes.

Los conectores son los elementos de software mediante los cuales los componentes interactúan. En la Figura 4.10 aparecen representados por líneas sólidas que van de puerto a puerto entre componentes, y sus propiedades también deben ser documentadas. Son dos los tipos de conectores que se presentan: el que comunica a las instancias de los clientes con el servidor, y el que comunica a éste con el manejador de datos. Sus propiedades son las siguientes:

- **TCP/IP:**
  - *Confiabilidad*: el protocolo de internet TCP/IP estipula un mecanismo muy confiable de comunicación entre sistemas. Garantiza que un conjunto de datos será enviado de un nodo a otro sin duplicación ni pérdida de datos, además de que serán adquiridos por el receptor en el orden correcto y de manera íntegra [78].
- **Lectura/escritura:**
  - *Concurrencia*: varios puertos de la instancia de *ServidorAplicacion* pueden requerir lectura o escritura de datos a través del único puerto de la instancia de *ManejadorDatos*.

4.3.2. Servidor

4.3.2.1. Lógica de negocio del servidor

Vista de módulos

La asignación de prioridades de requerimientos para este módulo se efectuó de forma análoga al sistema completo. Una vez determinadas las directrices, se procedió a identificar los conceptos de diseño aplicables a cada una; estos se presentan en el Cuadro 4.9. El módulo de la lógica de negocio del servidor de aplicación se descompuso en los módulos hijo enlistados a continuación; el nombre que se les ha puesto para identificarlos en el diseño arquitectónico aparece encerrado entre paréntesis. Asimismo, las responsabilidades que cada uno debe llevar a cabo se listan debajo de sus respectivos nombres.

- **Sesión (Sesion)**
  - Habilitar o deshabilitar los servicios de usuario.
  - Interpretar las solicitudes provenientes del cliente.
- **Servicios para alumno (ServicioAlumno)**
  - Solicitar listas y detalles de datos biomédicos a los que se tiene permiso de acceso.
  - Solicitar alguno de los servicios de visualización.
- **Servicios para investigador (ServicioInvestigador)**
  - Solicitar alguno de los servicios relacionados con la gestión de datos biomédicos: alta, baja y consulta de datos.
  - Solicitar alguno de los servicios relacionados con la gestión de usuarios: alta, baja, modificación y consulta de datos.
  - Solicitar alguno de los servicios de visualización.
- **Servicios para administrador (ServicioAdmin)**

Directriz arquitectónica	Sub-atributo de calidad	Concepto de diseño
Visualizar	Demanda de recursos (desempeño)	***Herramientas*** – VTK – Chip gráfico
	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Generalización de módulos
	Prevenir propagación de cambios (modificabilidad)	***Tácticas*** – Mantener las interfaces ***Patrones*** – Facade
Manejar sesión	Seguridad del sistema (seguridad)	***Tácticas*** – Autenticar usuarios – Concurrencia ***Tecnología*** – Hilos
Gestionar registros	Seguridad del sistema (seguridad)	***Tácticas*** – Autenticar usuarios
Escenario 1	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Generalización de módulos ***Patrones*** – Facade
Escenario 2	Sencillez para efectuar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Uso de interfaces
Escenario 4	Sencillez para efectuar cambios (modificabilidad)	***Tácticas*** – Mantener las interfaces

Cuadro 4.9: Directrices arquitectónicas identificadas y conceptos de diseño propuestos para el módulo de lógica de negocio del servidor de aplicación

  - Solicitar alguno de los servicios relacionados con la gestión de usuarios: alta, baja, modificación y consulta de datos.
- **Visualización tridimensional (Visualizacion3D)**
  - Generar las representaciones tridimensionales: representación anatómica, representación conjunta y mapeo de EEG.
  - Generar las proyecciones 2D a partir de las representaciones 3D.
  - Efectuar la lectura de conjuntos de información biomédica (series de imágenes y señales de EEG).
- **Comandos del servidor (ComandosServidor)**
  - Deserializar los datos y parámetros provenientes del cliente.
  - Invocar métodos en otros módulos que provean la funcionalidad solicitada por el cliente.
  - Proporcionar a otros módulos la información necesaria para efectuar la funcionalidad solicitada.
  - Serializar la respuesta del servidor (datos, parámetros, mensajes de error, etc.) para que sea enviada al cliente.

El Cuadro 4.10 presenta la documentación básica de las interfaces de cada módulo; en él se lista la información que cada una espera recibir así como la información que retornan. Finalmente, el diagrama de descomposición de módulos se expone en la Figura 4.12.

Módulo	Info. entrada	Info. salida
Sesión	– Datos para iniciar una sesión. – Datos para determinar peticiones solicitadas por los clientes.	– Respuesta a las peticiones de los clientes. En este caso, la respuesta puede consistir en: ▪ Datos de usuarios. ▪ Datos relacionados con información biomédica. ▪ Resultados de tareas de visualización. ▪ Mensajes de error relacionados al inicio de sesión. ▪ Mensajes de error provenientes de otros módulos.
Servicios para alumno	– Parámetros para efectuar tareas de visualización. – Parámetros para la consulta y/o lectura de información biomédica. – Ubicación de información biomédica (imágenes y señales).	– Proyección 2D resultado de los procesos de representación 3D. – Listas de datos biomédicos dados de alta. – Mensajes de error relativos a los procesos de representación 3D. – Mensajes de error relativos a la solicitud de listas de datos biomédicos.
Servicios para investigador	– Parámetros para efectuar tareas de visualización. – Parámetros para efectuar gestión de usuarios. – Parámetros para efectuar gestión de usuarios. – Ubicación de información biomédica (imágenes y señales).	– Proyección 2D resultado de los procesos de representación 3D. – Listas de datos biomédicos y de alumnos dados de alta. – Mensajes de error relativos a los procesos de representación 3D. – Mensajes de error relativos a la solicitud de listas de datos biomédicos. – Mensajes de error relativos a la solicitud de listas de alumno.
Servicios para administrador	– Parámetros para la gestión de usuarios.	– Lista de usuarios dados de alta. – Mensajes de error relativos a la solicitud de listas de datos de usuario.
Visualización tridimensional	– Lista de parámetros para efectuar un tipo de representación 3D particular. – Parámetros para identificar información biomédica (imágenes y señales).	– Proyección 2D resultado de los procesos de representación 3D. – Mensajes de error relativos a los procesos de representación 3D.
Comandos del servidor	– Parámetros serializados para ejecutar la funcionalidad de la lógica de negocio.	– Respuesta serializada a las peticiones de los clientes. – Mensajes de error serializados relacionados con la lógica de negocio.

Cuadro 4.10: Documentación de las interfaces de los módulos hijo de la lógica de negocio del servidor de aplicación

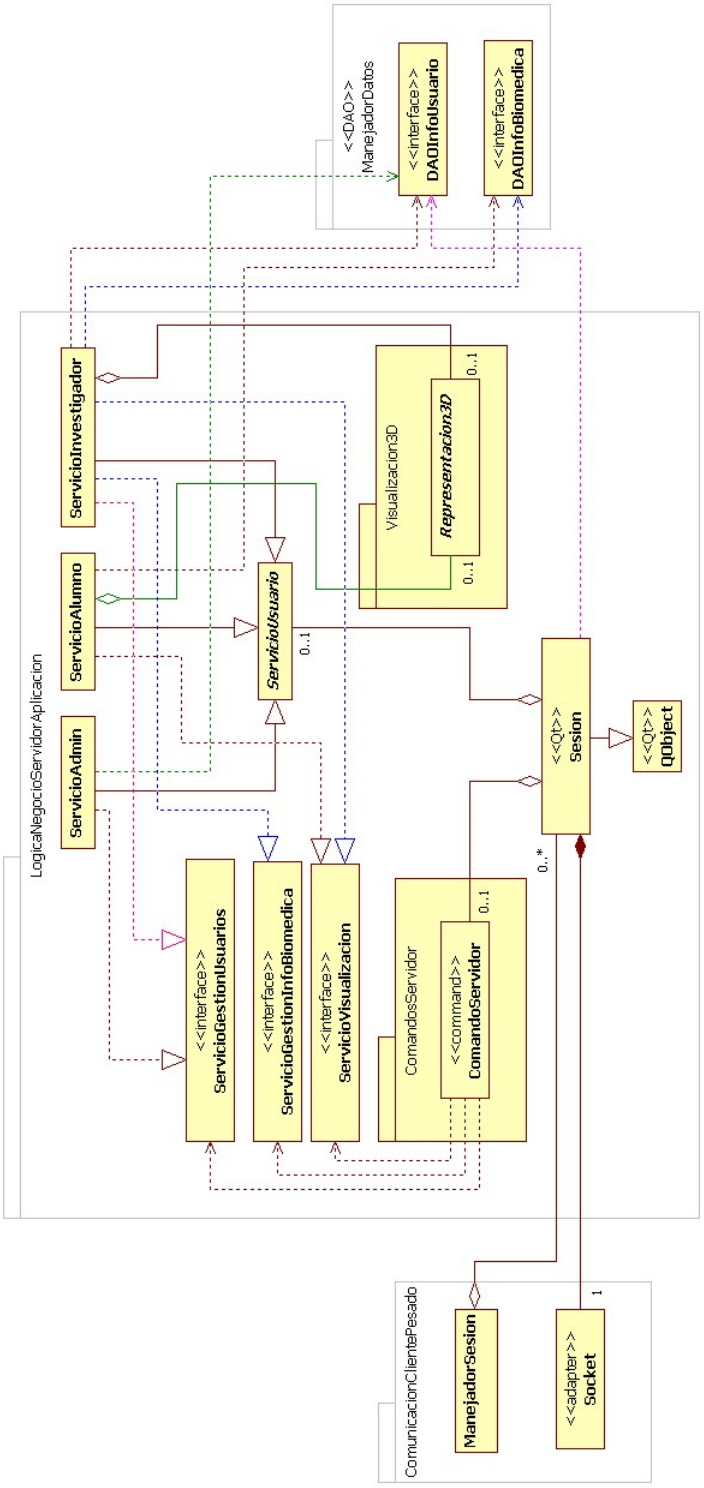


Figura 4.12: Vista de módulos de la lógica de negocio del servidor de aplicación. Las diferencias de color de las líneas sólo son para que puedan ser distinguidas.

Vista de componentes

La vista de componentes y la guía de componentes se ilustran en la Figura 4.13 y en la Figura 4.14, respectivamente. Los componentes del manejador de datos también se muestran en esta vista para exhibir la comunicación con la lógica de negocio. La principal diferencia entre este diagrama y el presentado para la vista de componentes del sistema completo (ver Figura 4.10) es la relación de delegación entre puertos que se muestra como una flecha de línea punteada; esta relación establece que el puerto apuntado por la flecha desempeñará el papel de aquel puerto del que la flecha sale. Por otra parte, el mapeo entre la vista de módulos y la de componentes se expone en el Cuadro 4.11.
Las propiedades de cada tipo de componente se listan a continuación.

- **QThread**
  - *Concurrencia*: cada instancia del componente QThread constituye a un hilo determinado, y en dicho hilo se ejecutan todos los demás componentes aquí listados.
  - *Creación*: una instancia del componente se crea cuando recién se ha conectado un cliente.
  - *Puertos*: presenta un único puerto cuya interfaz se delega al de la instancia del componente Socket.
- **Sesión**
  - *Concurrencia:* puede haber varias instancias del componente en un momento dado, cada una ejecutándose en un hilo generado por una instancia de QThread.
  - *Creación*: una instancia del componente se crea cuando recién se ha conectado un cliente.
  - *Procesamiento de datos*: el componente no modifica la información que recibe de otros componentes; sólo verifica su validez o lee la información que requiera.
  - *Puertos*: mediante su único puerto solicita servicios a otros componentes y redirige la información obtenida de ellos.
- **Socket**
  - *Concurrencia*: puede haber varias instancias del componente en un momento dado, cada una ejecutándose en un hilo generado por una instancia de QThread. Cada instancia atiende sólo a un cliente.
  - *Creación*: una instancia del componente es generada luego de que un componente tipo Sesion ha recibido los datos del cliente que recién se ha conectado.
  - *Procesamiento de datos*: la instancia recibe la información del cliente a través de una conexión TCP/IP, la cual garantiza la integridad de los datos recibidos. Los primeros 4 bytes recibidos indican cuántos bytes están siendo enviados por parte del cliente. El componente no modifica la información recibida.
  - *Buffer*: cada instancia cuenta con un buffer para almacenar la información (que constituye peticiones, principalmente) recibida de cliente. No hay un tamaño máximo definido para este buffer.
  - *Puertos*: la instancia recibe la información del cliente y otros componentes pueden solicitarla a través de su puerto.
- **ServiciosUsuario**:
  - *Concurrencia*: puede haber varias instancias del componente en un momento dado,

cada una ejecutándose en un hilo generado por una instancia de QThread.
- *Creación*: una instancia del componente se genera luego de que el componente tipo Sesion ha validado la información recibida del cliente. La instancia se destruye hasta que se reciba información para finalizar la sesión o el usuario se desconecte del sistema.
- *Procesamiento de datos*: el componente no modifica la información que recibe de otros componentes; sólo verifica su validez o lee la información que requiera.
- *Puertos*: el componente recibe peticiones a través de su puerto de respuesta. Asimismo, con su puerto de llamada solicita servicios a otros componentes (por ejemplo, instancias de Representacion3D) para llevar a cabo la petición solicitada. También puede acceder a la información del repositorio de datos a través de su puerto de lectura/escritura.

- **Comando**
  - *Concurrencia*: puede haber varias instancias del componente en un momento dado, cada una ejecutándose en un hilo generado por una instancia de QThread.
  - *Creación*: una instancia de Comando es generada cada que Sesion recibe una petición pertinente del cliente. La instancia se destruye una vez que la información de respuesta ha sido enviada.
  - *Procesamiento de datos*: las instancias de comando no modifican la información que reciben. Sin embargo, leen dicha información y extraen los parámetros pertinentes para llevar a cabo sus responsabilidades. Asimismo, serializan la respuesta de otros componentes con los que se relacionen para así ensamblar la respuesta que se enviará al cliente.
  - *Buffer*: las instancias de comando cuentan con un buffer capaz de contener la respuesta serializada a una petición del cliente. El tamaño de este buffer puede incrementarse significativamente cuando se envían resultados de tareas visualización (por ejemplo, una proyección a color sin compresión para un monitor de 1600 x 900 puede superar los 4 MB).
  - *Puertos*: el componente recibe la información serializada proveniente del cliente a través de su puerto de respuesta. Se comunica con la instancia del componente ServicioUsuario a través de su puerto de llamada, con el objetivo de solicitar cierto servicio y obtener respuesta al mismo.
- **Representacion3D**
  - *Concurrencia*: puede haber varias instancias del componente en un momento dado, cada una ejecutándose en un hilo generado por una instancia de QThread.
  - *Creación*: una instancia de Representacion3D es generada cuando se recibe una petición de visualización por parte de cliente. La instancia se destruye junto con la del servicio activo.
  - *Procesamiento de datos*: el componente recibe la ubicación del archivo donde se encuentra la información biomédica que se va a visualizar, la lee y efectúa el procesamiento pertinente sin modificar la información original.
  - *Buffer*: cada instancia contará con dos buffers que potencialmente pueden alcanzar grandes tamaños. Uno contiene la proyección serializada resultado del proceso de visualización, y otro contiene una versión submuestreada de la información volumétrica empleada para el mismo. Este último puede alcanzar un tamaño de

Componente	Módulos involucrados
QThread	QThread
Sesion	Sesion
Socket	Socket
ServiciosUsuario	Cualquiera de los siguientes: ▪ ServiciosAdmin ▪ ServiciosInvestigador ▪ ServiciosAlumno
Comando	Cualquier subclase de la clase Comando.
Representacion3D	Cualquier subclase de la clase Representacion3D.

Cuadro 4.11: Mapeo entre la vista de módulos y la vista de componentes para la lógica de negocio del servidor de aplicación

hasta 1 MB, pero el segundo puede superar los 4 MB si el proceso de visualización se efectúa para una representación de pantalla completa y a color.

- *Puertos*: a través de su puerto de respuesta el componente recibe la información necesaria para efectuar el proceso de visualización que le corresponde, y retorna el resultado serializado del mismo o los mensajes de error pertinenes si el proceso no pudo completarse.

- **Conectores**
  - Todos los conectores expuestos en esta vista son constituyen simples llamadas a los métodos implementados en cada componente. Todas las llamadas que ocurren dentro de un mismo hilo se efectúan de forma síncrona.

4.3.2.2. Módulos de comunicación

Las directrices arquitectónicas y los conceptos de diseño son, prácticamente, los mismos tanto para el cliente como para el servidor; se listan en el Cuadro 4.12. Por el momento el módulo de comunicación del lado del cliente consiste en una única clase; se decidió modelarlo de este modo en consideración a futuros diseños de mayor complejidad. La clase contenida en el módulo de comunicación del lado del cliente se denomina Socket, y del lado del servidor son dos: Socket y manejador de sesión. Sus responsabilidades se listan a continuación:

- **Socket, lado del cliente:**
  - Iniciar y terminar la comunicación con el servidor.
  - Enviar peticiones e información biomédica serializada hacia el servidor.
  - Aguardar respuestas del servidor.
- **Socket, lado del servidor:**
  - Escuchar las peticiones del cliente.
  - Enviar respuestas a las peticiones del cliente.
- **Manejador de sesión (ManejadorSesion)**

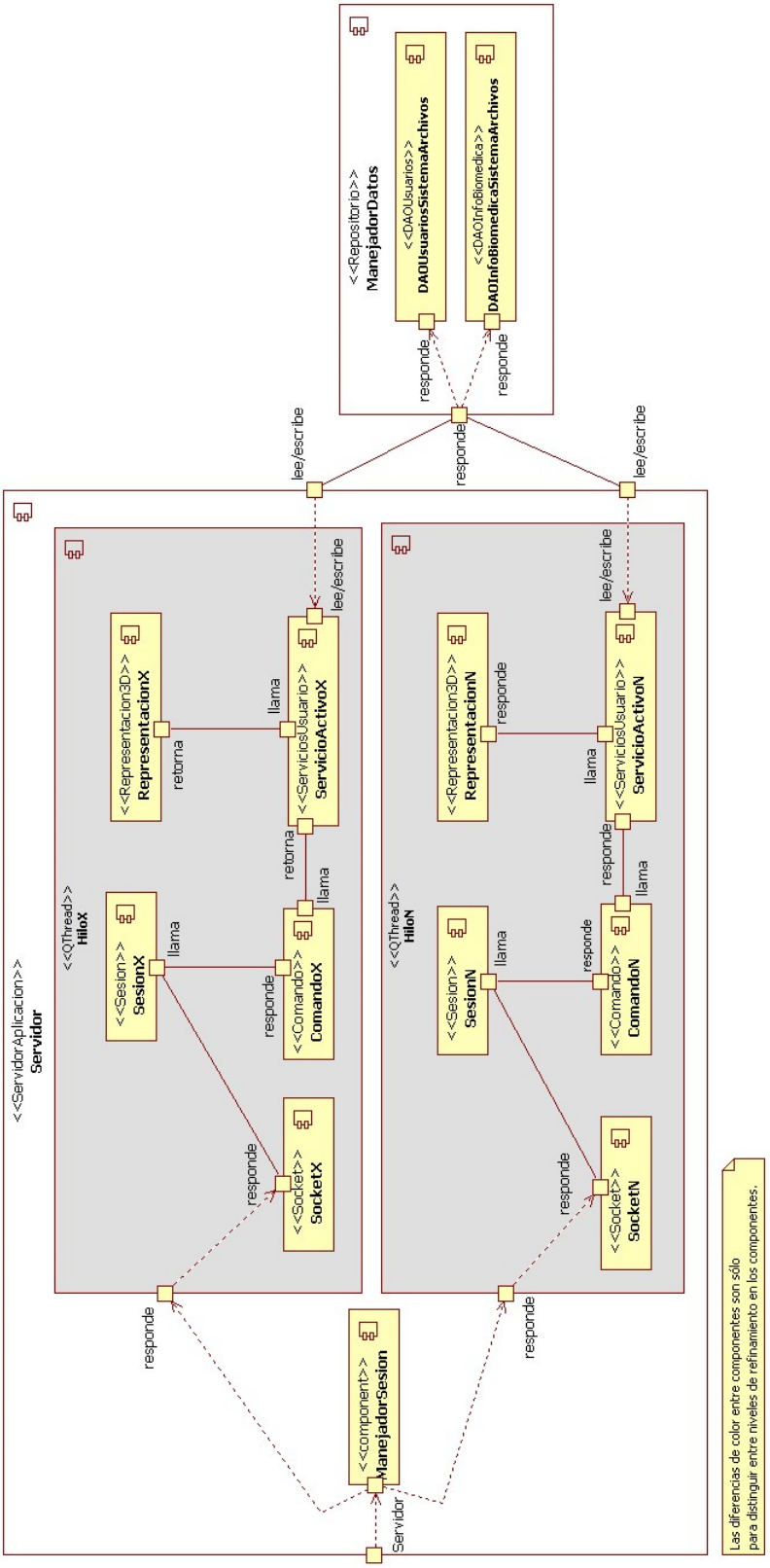


Figura 4.13: Vista de componentes para la lógica de negocio del servidor de aplicación

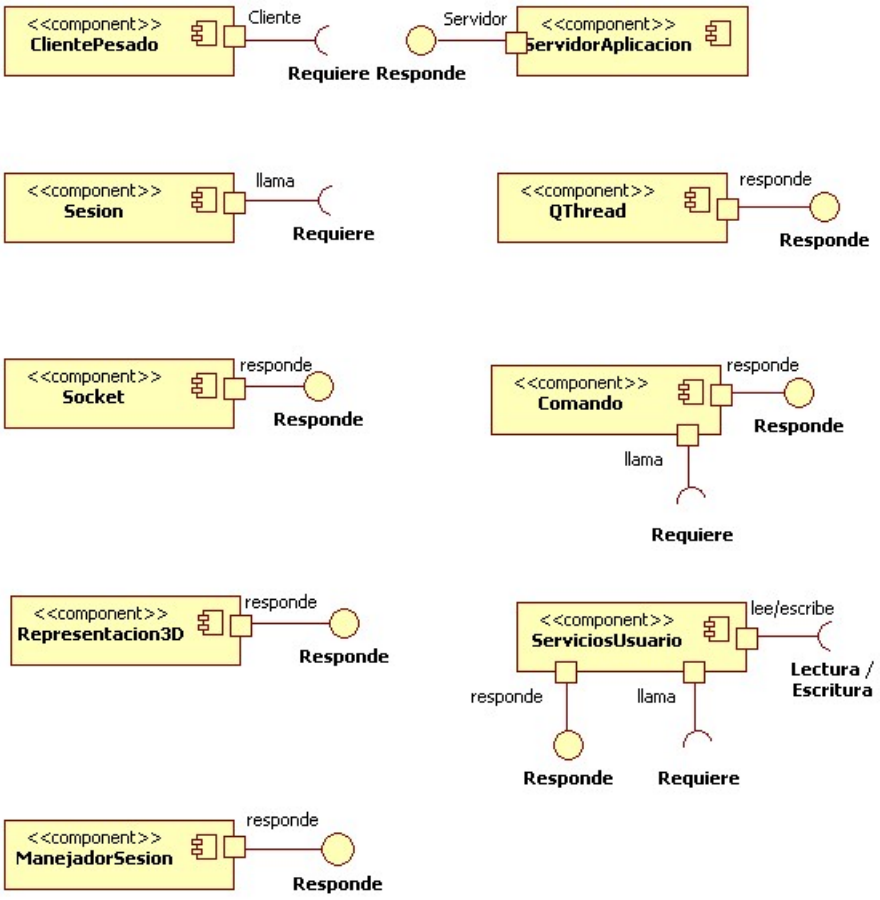


Figura 4.14: Guía de componentes para la lógica de negocio del servidor de aplicación

Directriz arquitectónica	Sub-atributo de calidad	Concepto de diseño
Visualizar Manejar sesión Gestionar registros	Localizar cambios (modificabilidad)	***Patrones*** – Adapter – Facade ***Herramientas*** – TCP/IP
Escenario 1	Localizar cambios (modificabilidad)	***Patrones*** – Adapter – Facade

Cuadro 4.12: Directrices arquitectónicas identificadas y conceptos de diseño propuestos para los módulos de comunicación en el cliente y en el servidor

Módulo	Info. entrada	Info. salida
Socket	– Información serializada. – Parámetros para la conexión con otro nodo.	– Información recibida del nodo contactado. – Estado actual del socket relacionado con la comunicación con el nodo contactado.
Manejador de sesión	*Este módulo no tiene interfaz pública.*	

Cuadro 4.13: Documentación de las interfaces de los módulos hijo de los módulos de comunicación

- Escuchar las conexiones entrantes de un cliente.
- Generar sesiones para cada cliente.

El Cuadro 4.13 presenta la documentación básica de las interfaces de cada módulo. Ahí se enlista la información que cada una espera recibir así como la información que retornan. El diagrama de descomposición de módulos se expone en la Figura 4.15. En lo que respecta a la vista de componentes, éstos se describieron en la Sección 4.3.2.1 en conjunto con la vista de componentes de la lógica de negocio del servidor de aplicación, y sus relaciones con otros componentes puede apreciarse en la Figura 4.13.

4.3.2.3. Módulo de visualización 3D

El módulo encargado de la visualización 3D en el servidor se presenta en la Figura 4.16, en la que se presenta una pequeña familia de clases basada en la clase abstracta llamada *Representacion3D*. Su interfaz se presenta en la Figura 4.17. Las subclases de ella derivadas deben implementar las tareas necesarias para efectuar algún tipo particular de visualización 3D. Se han implementado actualmente dos subclases: *RepresentacionVolumen* y *RepresentacionConjunta*, las cuales hacen uso intensivo de VTK. Algunos detalles sobre las características de VTK, la forma en que se empleó para abordar las tareas de visualización 3D, así como la descomposición para el componente Representacion3D (que es el que se crea en tiempo de ejecución a partir de este módulo), se presentan en el Apéndice D.
Las consideraciones que se tomaron en el diseño de la interfaz para estas clases son las siguientes:

- Toda representación requiere de uno o más conjuntos de información, y de uno o más parámetros numéricos para ser efectuada. La ubicación de la información y los paráme-

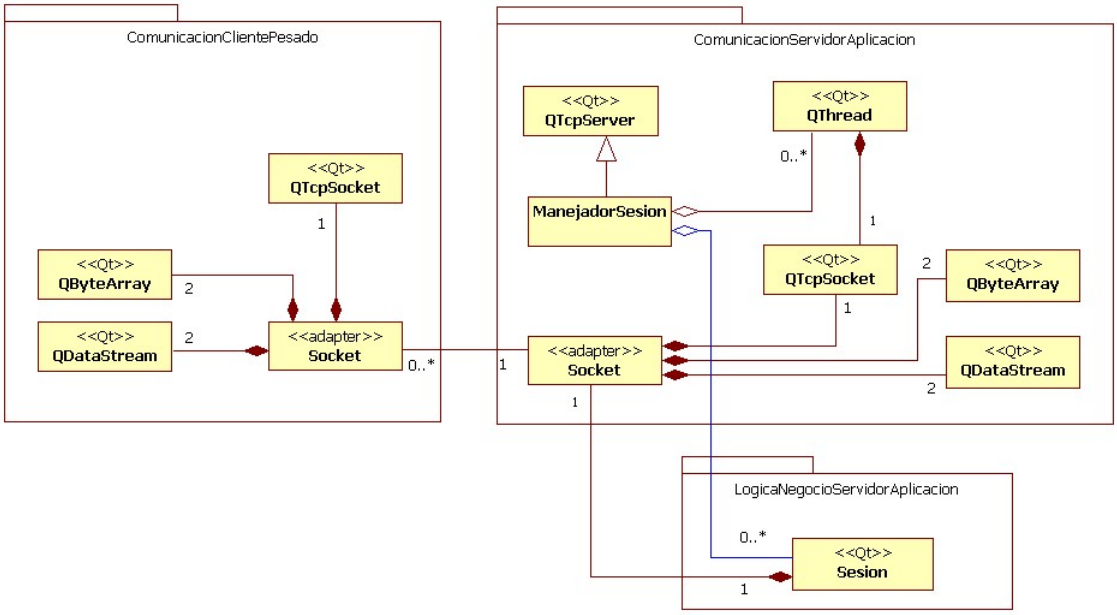


Figura 4.15: Módulos de comunicación.

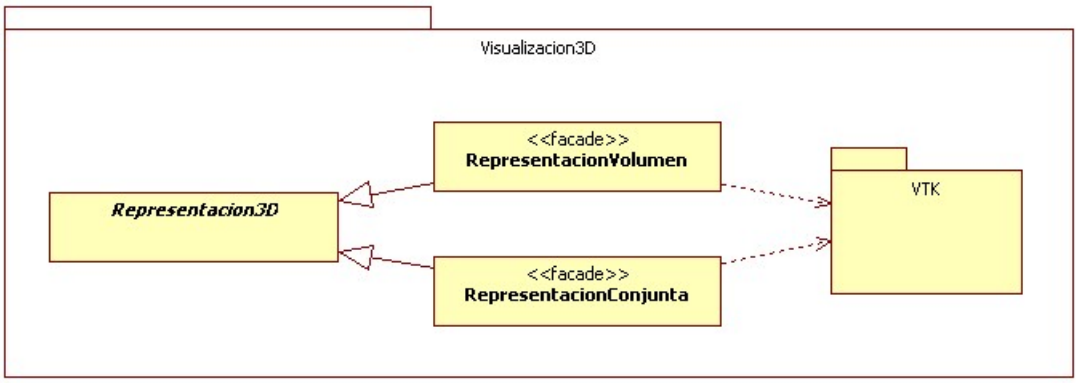


Figura 4.16: Módulo de visualización 3D. El módulo agregado de VTK representa a las clases que proveen la funcionalidad necesaria para las tareas de visualización.

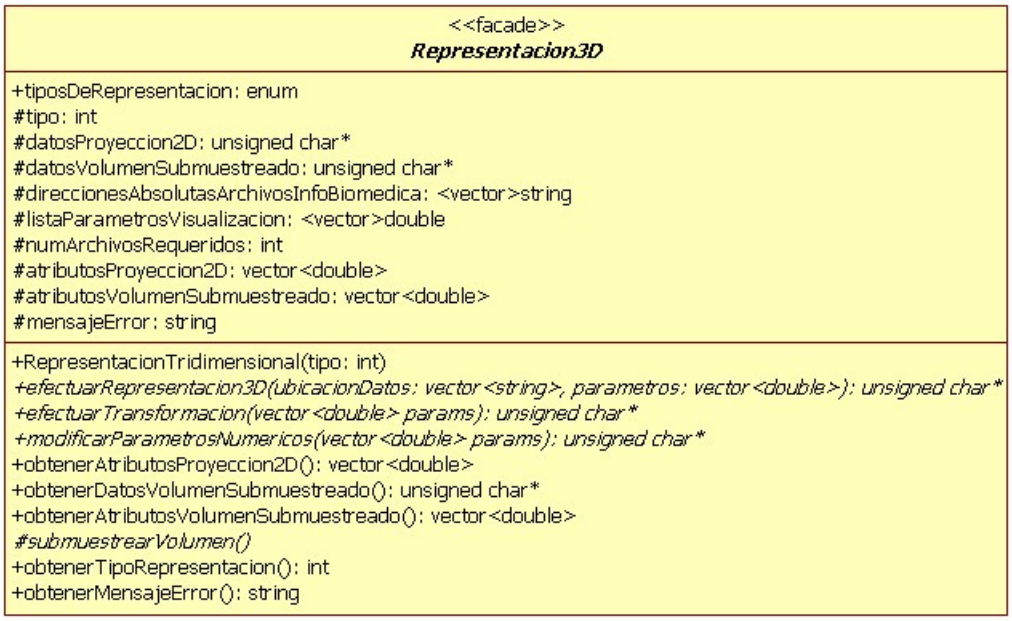


Figura 4.17: Representacion3D. Clase abstracta para todo tipo de representación tridimensional.

tros serán empleados por el método *efectuarRepresentacion*(), que debe ser implementado por cada subclase.

- Toda representación 3D tiene que implementar transformaciones espaciales básicas, como rotación, traslación y escalamiento. Esto puede hacerse en el método *efectuarTransformacion*().
- El método *modificarParametrosNumericos*() se ha definido para que se implementen transformaciones u operaciones sobre la información volumétrica potencialmente más complejas que las transformaciones básicas. A pesar de que el signado de este método es el mismo que el de *efectuarTransformacion*(), se recomienda utilizar este último para implementar únicamente las transformaciones básicas y así evitar cargarlo excesivamente de código. Además, al implementar ahí dichas transformaciones, éstas pueden ser heredadas a otros tipos de representaciones 3D similares, mientras que el método *modificarParametrosNumericos*() puede ser sobreescrito para implementar operaciones particulares.
- Los métodos que ulteriormente generan una proyección 2D a partir de información volumétrica son los tres ya descritos: *efectuarRepresentacion*(), *efectuarTransformacion*() y *modificarParametrosNumericos*(). Todos deben retornar un búffer que contenga la información de imagen de dicha proyección; la forma en que ésta se organice es decisión del programador. En otra palabras, lo que devuelven estos métodos es la proyección 2D serializada; de este modo se facilita que otros componentes puedan leer e interpretar fácilmente la imagen, adaptándola al formato que más les convenga. En el caso de este proyecto, dicha serialización es llevada a cabo enteramente por las herramientas de VTK.

4.3.2.4. Módulos de comandos y protocolo de mensajes

Los módulos ComandosCliente (ver Figura 4.20) y ComandosServidor (ver Figura 4.12) son simplemente familias de clases. Del lado del cliente todas las subclases heredan de la clase ComandoCliente (ClientCommand) y del lado del servidor heredan de ComandoServidor (ServerCommand). Las subclases implementan comportamientos particulares necesarios para efectuar prácticamente toda la funcionalidad del sistema; esto es porque entre sus responsabilidades se encuentran la serialización y deserialización de datos que clientes y servidor utilizan para comunicarse. En otras palabras, en el cliente estas subclases son las encargadas de estructurar los datos en una única cadena de bytes (es decir, serializarlos) de tal forma que el servidor pueda extraer de ella los datos necesarios (es decir, deserializarlos) para entender las peticiones de los clientes. Análogamente, el servidor debe ensamblar una cadena de bytes que conforma la respuesta a las peticiones, y los clientes deberán poder leerla y comprenderla. De este modo, los módulos ComandosCliente y ComandosServidor implementan el protocolo para construir los mensajes que cliente y servidor intercambian, permitiendo así la realización de los servicios del sistema; dicho protocolo se expondrá más adelante.
La Figura 4.18 muestra la interfaz empleada por las clases en ambos módulos, incluyendo la del iterador empleado para la serialización y deserialización de datos. Los puntos importantes sobre la implementación de ambas familias de clases son los siguientes:

- Para cada subclase de ComandoCliente en el cliente hay una subclase análoga de ComandoServidor en el servidor; ambas son responsables de aplicar el protocolo de mensajes que llevará a la ejecución de alguna funcionalidad. Cada par de clases se identifica por un valor numérico entero (un número de ID) que es exactamente el mismo en ambas, y que es especificado desde su constructor.
- Las subclases en el servidor deben estar asociadas con todos los módulos necesarios para llevar a cabo la funcionalidad solicitada; las llamadas a métodos de estos módulos deben efectuarse dentro del método *ejecutar*() y las respuestas de cada uno registradas como más convenga.
- Las subclases en el servidor reciben como parámetro en su constructor un búffer con información serializada; esta información debería, en principio, ser la proveniente del socket y constituye la petición efectuada por el cliente. Los parámetros para la funcionalidad solicitada son extraídos mediante el método *deserializar*(). Una vez que se ha llamado al método *ejecutar*() y la información de los otros módulos ha sido obtenida, ésta es serializada en un búffer mediante en el método *serializar*().
- Las subclases en el cliente no reciben datos serializados en su constructor, sino los parámetros necesarios para solicitar funcionalidad al servidor; estos parámetros son ensamblados en un arreglo de bytes mediante el método *serializar*(). Por otra parte, el método *deserializar*() recibe un búffer con información serializada proveniente, en principio, del socket; este búffer constituye la respuesta del servidor a una petición previa del cliente.
- La clase iterador está agregada a toda subclase de ComandoCliente o ComandoServidor. Es la principal responsable de efectuar la serialización o deserialización de datos, es decir, de estructurar la cadena de bytes que contiene los mensajes y parámetros a intercambiar, y de extraer éstos de dicha cadena.
- Todo arreglo de bytes que conforme un mensaje entre cliente y servidor debe comenzar

con tres bytes específicos: el caracter "#", el número de ID del comando en cuestión, y de nuevo el caracter "#", en ese orden; el resto de los bytes en el arreglo son los parámetros y datos necesarios para la comunicación excepto el último que debe ser, nuevamente, el caracter "#" indicando el final de la información serializada.

El cuadro 4.14 enlista las subclases de ComandoCliente y ComandoServidor que se han implementado para el sistema. Recuérdese que la responsabilidad general de todas ellas es serializar y deserializar información; ergo, la información serializada del lado del cliente es la que se deserializa en el servidor, y viceversa, por lo que el cuadro sólo indica qué datos son los que se serializan y de qué tipo son (indicado entre paréntesis). Para ejemplificar esta serialización, el cuadro 4.15 muestra la manera en que la subclase ComandoDarDeAltaUsuario hace esta tarea.

Cuadro 4.14: Subclases de las clases ComandoServidor y ComandoCliente.

#ID	Subclase	Sintaxis cliente	Sintaxis servidor
1	ComandoVerificar-ExistenciaUsuarios		▪ Hay usuarios (booleano) ▪ Mensaje de error (cad. caracteres)
2	ComandoDarDe-AltaPrimerAdmin	▪ Nombre admin (cad. caracteres) ▪ Contraseña (cad. caracteres)	▪ Alta exitosa (booleano) ▪ Mensaje de error (cad. caracteres)
3	ComandoIniciar-Sesion	▪ Nombre de usuario (cad. caracteres) ▪ Contraseña (cad. caracteres)	▪ Nivel de acceso[1] (entero) ▪ Mensaje de error (cad. caracteres) ▪ Tiempo límite en minutos para requerir restauración de sesión (flotante de doble precisión) ▪ Tiempo límite en minutos para cerrar la sesión automáticamente (flotante de doble precisión)
4	ComandoCerrar-Sesion		▪ Éxito en el cierre (booleano) ▪ Mensaje de error (cad. caracteres)
6	ComandoDarDe-AltaUsuario	▪ Nombre de usuario (cad. caracteres) ▪ Contraseña (cad. caracteres) ▪ Nivel de acceso asignado (entero)	▪ Alta exitosa (booleano) ▪ Mensaje de error (cad. caracteres)
7	ComandoDarDe-BajaUsuario	▪ Nombre de usuario (cad. caracteres)	▪ Baja exitosa (booleano) ▪ Mensaje de error (cad. caracteres)

Continúa...

[1] Los niveles de acceso para los usuarios, que corresponden a los roles que pueden tener, son representados por un valor entero en el sistema: administrador → 0, investigador → 1, alumno → 2.

8	ComandoModificarDatosUsuarios	▪ Nombre de usuario (cad. caracteres) ▪ Nuevo password (cad. caracteres) ▪ Nombre de administrador (cad. caracteres) ▪ Nivel de acceso (entero) ▪ Número de archivos accesibles[2] (entero sin signo) ▪ Tipo de archivo 1[3] (entero) ▪ Nombre de archivo 1 (cad. caracteres) ▪ Tipo de archivo 2 (entero) ▪ Nombre de archivo 2 (cad. caracteres) ▪ ... ▪ Tipo de archivo N (entero) ▪ Nombre de archivo N (cad. caracteres)	▪ Modificación exitosa (booleano) ▪ Mensaje de error (cad. caracteres)
9	ComandoObtenerListaUsuarios	▪ Nombre de usuario (cad. caracteres)	▪ Acceso exitoso (booleano) ▪ Mensaje de error (cad. caracteres) ▪ Total de usuarios a cargo del usuario actual (entero sin signo) ▪ Nombre de usuario 1 (cad. caracteres) ▪ Administrador de usuario 1[4] (cad. caracteres) ▪ Nivel de acceso usuario 1 (entero) ▪ Número archivos accesibles al usuario 1[5] (entero sin signo) ▪ Tipo de archivo 1[6] (entero) ▪ Tamaño de archivo 1 en bytes (largo sin signo) ▪ Nombre de archivo 1 (cad. caracteres) ▪ Tipo de archivo 2 (entero) ▪ Tamaño de archivo 2 en bytes (largo sin signo) ▪ Nombre de archivo 2 (cad. caracteres) ▪ ... ▪ Tipo de archivo N (entero) ▪ Tamaño de archivo N en bytes (largo sin signo) ▪ Nombre de archivo N (cad. caracteres)

Continúa...

[2] Número de archivos a los que el usuario tiene permiso de acceder. Éstos no son los archivos que haya dado de alta sino aquéllos que otros usuarios le han otorgado permiso de acceso.

[3] Al igual que los roles de los usuarios, los tipos de archivos con información biomédica manejados por el sistema también se representan mediante un número entero: NifTI → 0, EDF → 1, PNG → 2

[4] El administrador de un usuario determinado es aquél que le dio de alta por primera vez.

[5] Número de archivos a los que al usuario en la lista se ha otorgado permiso de acceso.

[6] Los atributos de los archivos a los que cada usuario en la lista tiene acceso también viajan serializados como parte de la información en dicha lista.

			▪ Nombre de usuario 2 (cad. caracteres)
			▪ Administrador de usuario 2 (cad. caracteres)
			▪ Nivel de acceso usuario 2 (entero)
			▪ Núm. archivos accesibles al usuario 2 (entero sin signo)
			  • Tipo de archivo 1 (entero)
			  • ...
			▪ ...
			▪ Nombre de usuario N (cad. caracteres)
			▪ ...

10	ComandoDarDe-AltaInfoBiomedica	▪ Tipo de archivo (entero) ▪ Nombre del propietario (cad. caracteres) ▪ Contenido del archivo (datos binarios diversos)	▪ Alta exitosa (booleano) ▪ Mensaje de error (cad. caracteres)
11	ComandoDarDe-BajaInfoBiomedica	▪ Tipo de archivo (entero) ▪ Nombre del propietario (cad. caracteres) ▪ Nombre del archivo (cad. caracteres) ▪	▪ Baja existosa (booleano) ▪ Mensaje de error (cad. caracteres)
12	Comando-ObtenerLista-InfoBiomedica	▪ Nombre de usuario (cad. caracteres)	▪ Acceso exitoso (booleano) ▪ Mensaje de error (cad. caracteres) ▪ Número de archivos dados de alta (entero sin signo) ▪ Tipo de archivo 1 (entero) ▪ Tamaño del archivo 1 en bytes (largo sin signo) ▪ Nombre del archivo 1 (cad. caracteres) ▪ Tipo de archivo 2 (entero) ▪ ... ▪ Tipo de archivo N (entero) ▪ Tamaño del archivo N en bytes (largo sin signo) ▪ Nombre del archivo N (cad. caracteres)

Continúa...

13	Comando-Visualizar	▪ Nombre de usuario (cad. caracteres) ▪ Tipo de representación efectuada[7] (entero) ▪ Número de archivos requeridos[8] (entero sin signo) ▪ Nombre de archivo de info biomédica 1 (cad. caracteres) ▪ Nombre de archivo de info biomédica 2 (cad. caracteres) ▪ ... ▪ Nombre de archivo de info biomédica N (cad. caracteres) ▪ Número de parámetros requeridos[9] (entero sin signo) ▪ Parámetro 1 (flotante de doble precisión) ▪ Parámetro 2 (flotante de doble precisión) ▪ ... ▪ Parámetro N (flotante de doble precisión)	▪ Visualización exitosa (booleano) ▪ Mensaje de error (cad. caracteres) ▪ Número de atributos del volumen submuestreado[10] (entero sin signo) ▪ Atributo 1 del volumen (flotante de doble precisión) ▪ Atributo 2 del volumen(flotante de doble precisión) ▪ ... ▪ Atributo N del volumen (flotante de doble precisión) ▪ Número de atributos de la proyección 2D[11] (entero sin signo) ▪ Atributo 1 de la proyección (flotante de doble precisión) ▪ Atributo 2 de la proyección (flotante de doble precisión) ▪ ... ▪ Atributo N de la proyección (flotante de doble precisión) ▪ Datos del volumen submuestreado (varios bytes sin signo) ▪ Datos de la proyección (varios bytes sin signo)
14	ComandoEfectuar-Transformacion-Espacial	▪ Número de parámetros requeridos (entero sin signo) ▪ Parámetro 1 (flotante de doble precisión) ▪ Parámetro 2 (flotante de doble precisión) ▪ ... ▪ Parámetro N (flotante de doble precisión) ▪ Número de pixeles a lo largo de la proyección 2D (entero sin signo) ▪ Número de pixeles a lo ancho de la proyección 2D (entero sin signo)	▪ Transformación exitosa (booleano) ▪ Mensaje de error (cad. caracteres) ▪ Número de atributos de la proyección 2D (entero sin signo) ▪ Atributo 1 de la proyección (flotante de doble precisión) ▪ Atributo 2 de la proyección (flotante de doble precisión) ▪ ... ▪ Atributo N de la proyección (flotante de doble precisión) ▪ Datos de la proyección (varios bytes sin signo)
15	Comando-Modificar-Parametros-Numericos	▪ Número de parámetros requeridos (entero sin signo) ▪ Parámetro 1 (flotante de doble precisión) ▪ Parámetro 2 (flotante de doble precisión) ▪ ... ▪ Parámetro N (flotante de doble precisión) ▪ Número de pixeles a lo largo de la proyección 2D (entero sin signo) ▪ Número de pixeles a lo ancho de la proyección 2D (entero sin signo)	▪ Transformación exitosa (booleano) ▪ Mensaje de error (cad. caracteres) ▪ Número de atributos de la proyección 2D (entero sin signo) ▪ Atributo 1 de la proyección (flotante de doble precisión) ▪ Atributo 2 de la proyección (flotante de doble precisión) ▪ ... ▪ Atributo N de la proyección (flotante de doble precisión) ▪ Datos de la proyección (varios bytes sin signo)

[7] Los tipos de representación 3D implementados en el sistema también se representan mediante un número entero: representación en volumen → 0, representación conjunta → 1

[8] Cantidad de archivos en los que el sistema guarda la información biomédica necesaria para la representación 3D.

[9] El número de parámetros requeridos para la visualización dependerá ampliamente de la bibliotecas de software empleadas para efectuarla.

[10] Número de atributos requeridos para representar suficientemente al volumen submuestreado.

[11] Número de atributos requeridos para representar suficientemente a la proyección 2D.

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	#	6	#	j	u	a	n	8	6	\0	a	b	c	1	2	3	\0	#

Byte	1	2	3	4	5
	#	6	#	1	#

Cuadro 4.15: Ejemplo de serialización. Los datos son los empleados por la clase ComandoDarDeAltaUsuario; el esquema superior ilustra byte por byte el arreglo de información serializada del lado del cliente para solicitar el alta de un usuario, mientras que el esquema inferior expone la respuesta serializada del lado del servidor ante un alta exitosa. En ambos arreglos el caracter "\0" indica el fin de una cadena de caracteres.

4.3.3. Manejador de datos

En lo relativo al módulo del manejador de datos, las directrices arquitectónicas y los conceptos de diseño considerados para satisfacerlas se listan en el Cuadro 4.16. Los módulos hijo que fueron identificados son los siguientes: Fábrica de DAO, Fábrica de DAO para sistema de archivos, DAO para info de usuarios en sistema de archivos y DAO para info biomédica en sistema de archivos. A continuación se listan las responsabilidades de cada uno:

- **Fábrica de DAO (FabricaDAO)**
  - Generar y otorgar acceso a los DAO's cuando estos sean requeridos.
- **Fábrica de DAO para sistema de archivos (FabricaDAOSistemaArchivos)**
  - Generar y otorgar acceso a los DAO's cuando estos sean requeridos; los DAO's sólo trabajan con sistemas de archivos.
- **DAO para info de usuarios en sistema de archivos (DAOInfoUsuariosSistemaArchivos**
  - Efectuar alta y baja de usuarios.
  - Permitir modificación y consulta de información de usuarios.
  - Generar listas de usuarios dados de alta por un determinado administrador.
- **DAO para info biomédica en sistema de archivos (DAOInfoBiomedicaSistemaArchivos)**
  - Efectuar alta y baja de información biomédica.
  - Verificar existencia de determinados datos y permitir el acceso a ellos.
  - Generar listas de información biomédica dada de alta por un determinado usuario.
- **Usuario**
  - Lectura de información relacionada a un usuario. Esta información incluye, pero no se limita a:
    - Nombre
    - Contraseña
    - Admin a cargo
    - Nivel de acceso
    - Lista de información accesible
- **Información Biomedica (InfoBiomedica)**
  - Lectura de información relacionada a información biomédica. Esta información incluye, pero no se limita a:

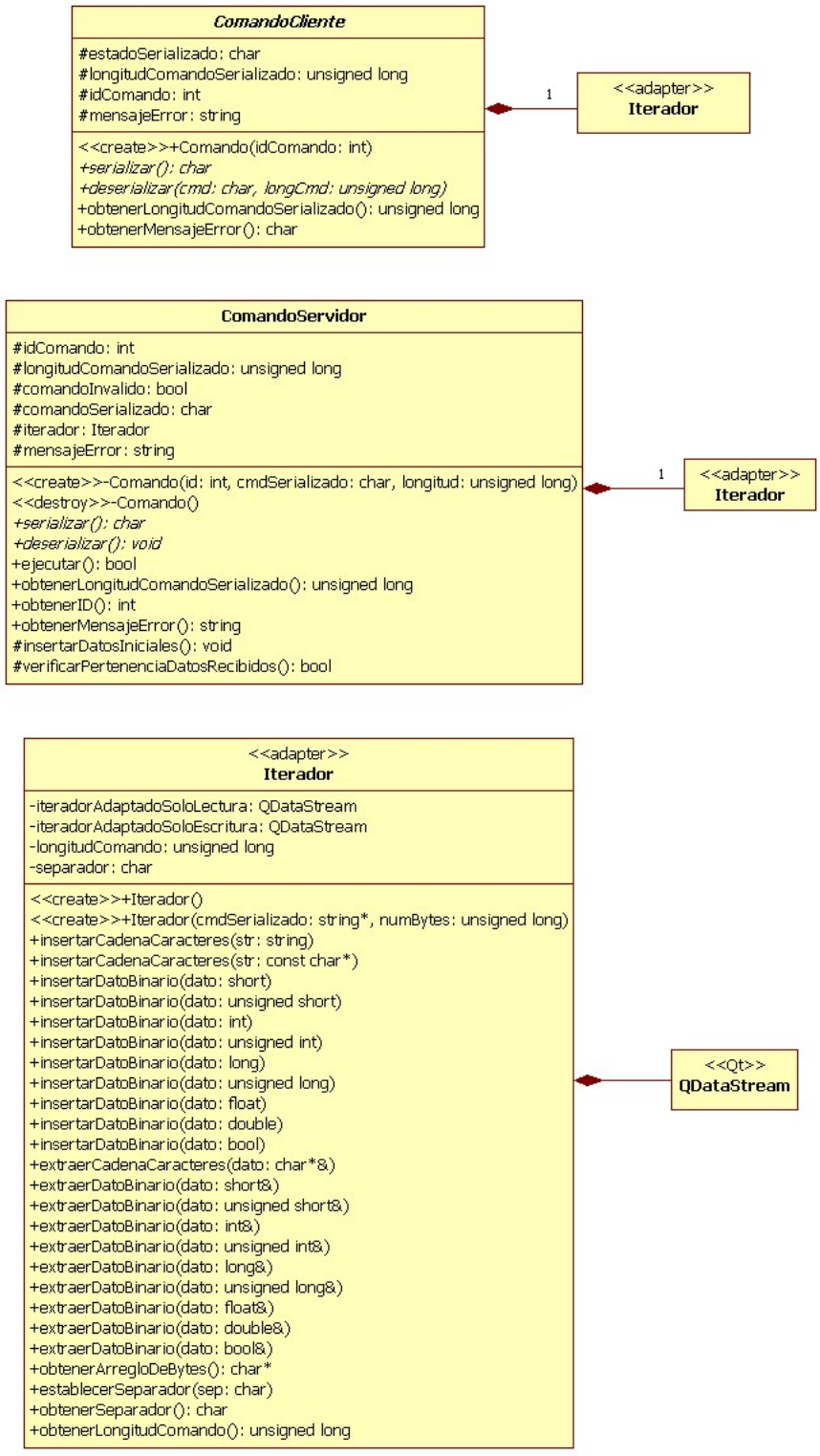


Figura 4.18: Arriba – clase abstracta para la familia de clases en el módulo ComandosCliente; en medio – clase abstracta para la familia de clases en el módulo ComandosServidor; abajo – iterador empleado por todas las subclases en ambos módulos

Directriz arquitectónica	Sub-atributo de calidad	Concepto de diseño
Manejar sesión	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Mantener las interfaces ***Patrones*** – Data Access Object (DAO)
	Mantener la integridad de los datos (confiabilidad)	***Herramientas*** Mecanismos de sincronización de hilos
Gestionar registros	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Mantener las interfaces ***Patrones*** – Data Access Object (DAO)
	Mantener la integridad de los datos (confiabilidad)	***Herramientas*** Mecanismos de sincronización de hilos
Escenario 1	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Mantener las interfaces ***Patrones*** – Data Access Object (DAO)
Escenario 2	Sencillez para efectuar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica – Mantener las interfaces

Cuadro 4.16: Directrices arquitectónicas identificadas y conceptos de diseño propuestos para el módulo del manejador de datos

  - Tipo de información
  - Ubicación (local absoluta)
  - Nombre de usuario del propietario

El Cuadro 4.17 presenta la documentación básica de las interfaces de cada módulo; lista la información que cada una espera recibir así como la información que retornan. Finalmente, el diagrama de descomposición de módulos se expone en la Figura 4.19.

4.3.4. Cliente

4.3.4.1. Lógica de negocio del cliente

Las directrices arquitectónicas para el diseño de este módulo, así como los conceptos de diseño considerados para satisfacerlas, se listan en el Cuadro 4.18. Los módulos hijo que fueron identificados se listan a continuación; el nombre que se les ha puesto para identificarlos en el diseño arquitectónico aparece encerrado entre paréntesis. Asimismo, las responsabilidades que cada uno debe llevar a cabo se listan debajo de sus respectivos nombres.

- **Sesión (Sesion)**
  - Habilitar o deshabilidar los servicios de usuario.
  - Verificar las condiciones del sistema al momento de que se inicia. Estas incluyen, pero no se limitan a:
    - Disponibilidad del servidor
    - Existencia de usuarios
  - Conectar al cliente con el servidor de aplicación.
- **Servicios para administrador (ServicioAdmin)**

Módulo	Info. entrada	Info. salida
Fábrica de DAO	(*Ninguna*)	DAOs
Fábrica de DAO para sistemas de archivos	(*Ninguna*)	DAO's que operan sobre sistemas de archivos.
DAO para info de usuarios en sistema de archivos	– Información para dar de alta a un nuevo usuario. – Parámetros para dar de baja o para consultar a un usuario determinado. – Parámetros para solicitar una lista de usuarios.	– Información sobre usuarios. – Listas de usuarios. – Mensajes de error relativos a la dada de alta, baja, consulta o modificación de datos de usuarios.
DAO para info biomédica en sistema de archivos	– Parámetros para dar de alta información biomédica. – Parámetros para dar de baja información biomédica. – Parámetros para verificar existencia de información biomédica.	– Ubicación de la información biomédica solicitada. – Lista de información biomédica disponible según los parámetros especificados. – Mensajes de error relativos a la dada de alta, baja o consulta de información biomédica.

Cuadro 4.17: Documentación de las interfaces de los módulos hijo del manejador de datos

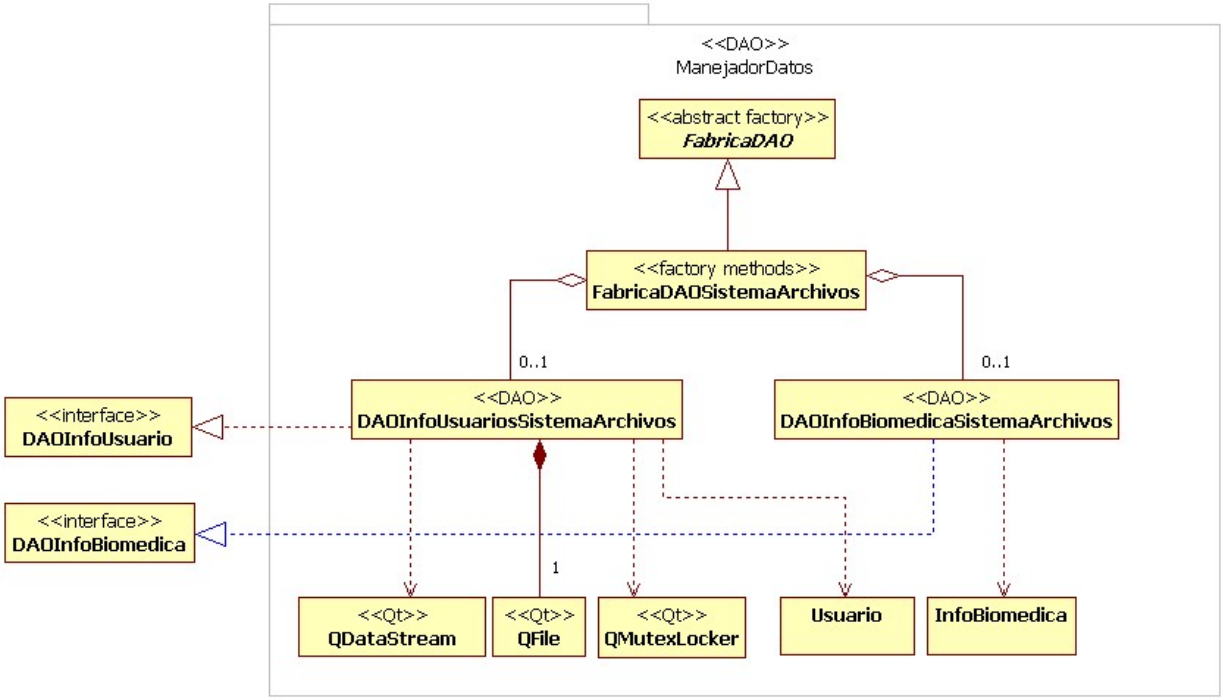


Figura 4.19: Diagrama de clases del manejador de datos. Las diferencias de color de las líneas sólo son para que puedan ser distinguidas.

Directriz arquitectónica	Sub-atributo de calidad	Concepto de diseño
Visualizar	Prevenir propagación de cambios (modificabilidad)	***Tácticas*** – Mantener las interfaces
	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica ***Patrones*** – Adapter
Manejar sesión	Prevenir propagación de cambios (modificabilidad)	***Tácticas*** – Mantener las interfaces
	Seguridad del sistema (seguridad)	***Tácticas*** – Autenticación de usuarios – Latido
Gestionar registros	Seguridad del sistema (seguridad)	***Tácticas*** – Autenticar usuarios
Escenario 1	Prevenir propagación de cambios (modificabilidad)	***Tácticas*** – Mantener las interfaces
	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica ***Patrones*** – Facade – Adapter
Escenario 2	Prevenir propagación de cambios (modificabilidad)	***Tácticas*** – Mantener las interfaces
Escenario 4	Localizar cambios (modificabilidad)	***Tácticas*** – Coherencia semántica ***Patrones*** – Adapter

Cuadro 4.18: Directrices arquitectónicas identificadas y conceptos de diseño propuestos para el módulo de lógica de negocio del cliente pesado.

  - Solicitar alguno de los servicios relacionados con la gestión de usuarios: alta, baja, modificación de datos y consulta de detalles.
- **Servicios para investigador (ServicioInvestigador)**
  - Solicitar alguno de los servicios relacionados con la gestión de datos biomédicos: alta, baja y consulta de detalles.
  - Solicitar alguno de los servicios relacionados con la gestión de usuarios: alta, baja, modificación de datos y consulta de detalles.
  - Solicitar alguno de los servicios de visualización.
- **Servicios para alumno (ServicioAlumno)**
  - Solicitar lista y detalles de datos biomédicos a los que se tiene permiso de acceso.
  - Solicitar alguno de los servicios de visualización.
- **Conversión de datos volumétricos (ModuloConversionDatosVolumetricos)**
  - Convertir archivos de datos volumétricos (que estén en un formato arbitrario) al formato por defecto usado por el sistema: NifTI.
- **Comandos del cliente (ComandosCliente)**
  - Serializar las peticiones (datos, parámetros, mensajes de error, etc.) para que sean enviadas al servidor.
  - Interpretar (deserializar) los datos y parámetros provenientes del servidor.
  - Proveer los datos a los módulos responsables por proveer la funcionalidad solicitada por el cliente.

Módulo	Info. entrada	Info. salida
Sesión	– Datos para la conexión con el servidor de aplicación. – Datos para el inicio de sesión.	– Mensajes relativos al éxito o fracaso del inicio de sesión. – Mensajes relativos a la disponibilidad del servidor.
Servicios para administrador	– Parámetros para la gestión de usuarios	– Lista de usuarios dados de alta. –Mensajes de error relativos a la solicitud de listas de datos de usuario.
Servicios para investigador	– Parámetros para efectuar tareas de visualización. – Parámetros para efectuar gestión de usuarios. – Ubicación de información biomédica (imágenes y señales).	– Proyección 2D resultado de los procesos de representación 3D. – Listas de datos biomédicos dados de alta. – Listas de alumnos dados de alta. – Mensajes de error relativos a los procesos de representación 3D. –Mensajes de error relativos a la solicitud de listas de datos biomédicos. – Mensajes de error relativos a la solicitud de listas de alumno.
Servicios para alumno	– Parámetros para efectuar tareas de visualización. – Parámetros para la consulta y/o lectura de información biomédica. – Ubicación de información biomédica (imágenes y señales).	Proyección 2D resultado de los procesos de representación 3D. – Listas de datos biomédicos dados de alta. – Mensajes de error relativos a los procesos de representación 3D. – Mensajes de error relativos a la solicitud de listas de datos biomédicos.
Conversión de datos volumétricos	– Ubicación de datos biomédicos	– Datos biomédicos en un formato estándar.
Comando del cliente	Parámetros para solicitudes de servicios al servidor.	– Información requerida por la funcionalidad invocada desde el cliente. – Mensajes de error relacionados con la lógica de negocio.

Cuadro 4.19: Documentación de las interfaces de los módulos hijo de la lógica de negocio del cliente pesado.

El Cuadro 4.19 presenta la documentación básica de las interfaces de cada módulo; lista la información que cada una espera recibir así como la información que retornan. Finalmente, el diagrama de descomposición de módulos se expone en la Figura 4.20.

4.3.4.2. Convertidor de datos volumétricos

Las relaciones de este módulo, ubicado dentro del módulo del cliente pesado, con otros se ilustra en la Figura 4.20. Por su parte, la Figura 4.21 presenta las interfaces de las principales clases de este módulo. Básicamente constituye una familia de clases cuya clase base es ConvertidorArchivoDatosVolumetricos (VolumeDataFileConverter), la cual estipula el contrato que debe ofrecer cualquier otro convertidor. Los métodos abstractos se definieron de tal modo que devolvieran valores booleanos; la idea es que el método *convertir*() se implemente de tal modo que retorne verdadero si la operación de conversión se efectuó con éxito, y falso en caso contrario. Por su parte, el método *validarDatos*() puede emplearse para corroborar que el archivo a convertir contenga la información esperada.

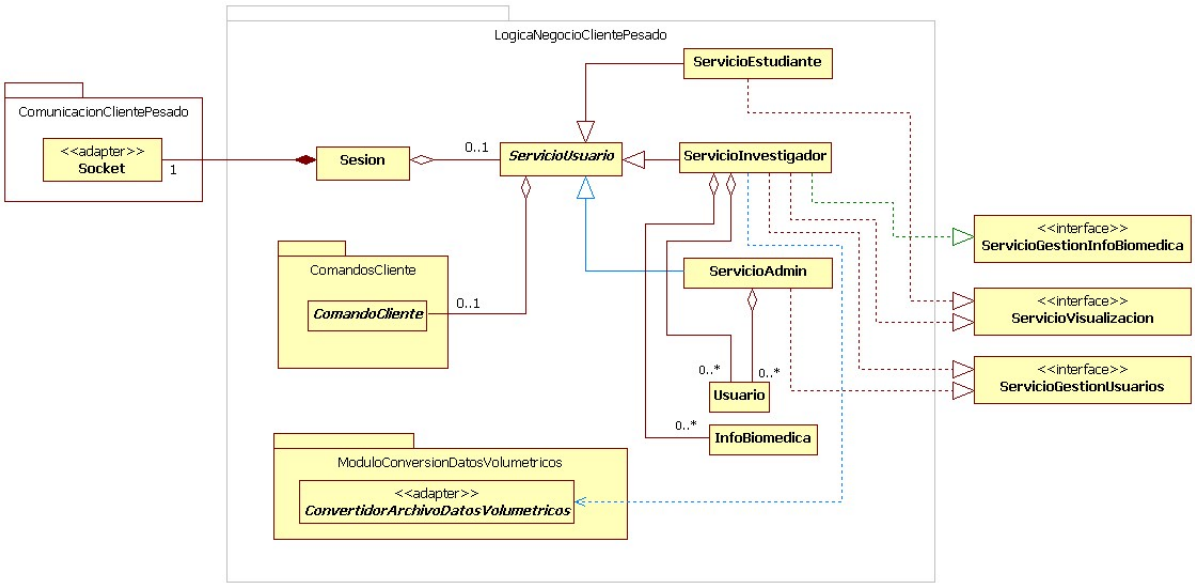


Figura 4.20: Vista de módulos de la lógica de negocio del cliente pesado. Las diferencias de color de las líneas sólo son para que puedan ser distinguidas.

4.3.4.3. Interfaz gráfica del cliente

El diseño de la interfaz gráfica del cliente está fuertemente basado en Qt; está conformada por un manejador de configuración y por dos familias de clases: una que hereda de la clase QMainWindow y otra que hereda de QDialog, ambas pertenecientes al marco de trabajo de Qt. Estas superclases implementan el comportamiento básico de toda ventana: abrirse, cerrarse, cambiar de tamaño y de posición, agregar y remover controles, etcétera. La Figura 4.22 muestra a las familias de clases, sus relaciones entre sí, con el manejador de configuración y con la lógica de negocio.

4.3.5. Vista de puesta en operación

La vista de puesta en operación para este proyecto ilustra cómo los componentes y archivos de instalación son colocados en diversos nodos computacionales. La Figura 4.23 expone a dichas estructuras de asignación (ver inicio de la Sección 3.1). Actualmente se espera que existan dos tipos de nodo para los clientes: nodos con sistemas operativos basados en GNU/Linux y nodos con el sistema operativo Windows. Por su parte, el nodo del servidor estará corriendo con un sistema operativo GNU/Linux. Las características de los nodos y de los archivos necesarios para ejecutar los componentes se describen a continuación.

4.3.5.1. Nodo del servidor

El nodo del servidor ejecutará dos instancias de componente: una de tipo ServidorAplicación y la otra de tipo ManejadorDatos. Sin embargo, la implementación actual de estos componentes previene la portabilidad a sistemas operativos que no estén basados en GNU/Linux, dado que la gestión de archivos que llevan a cabo contempla la organización de directorios

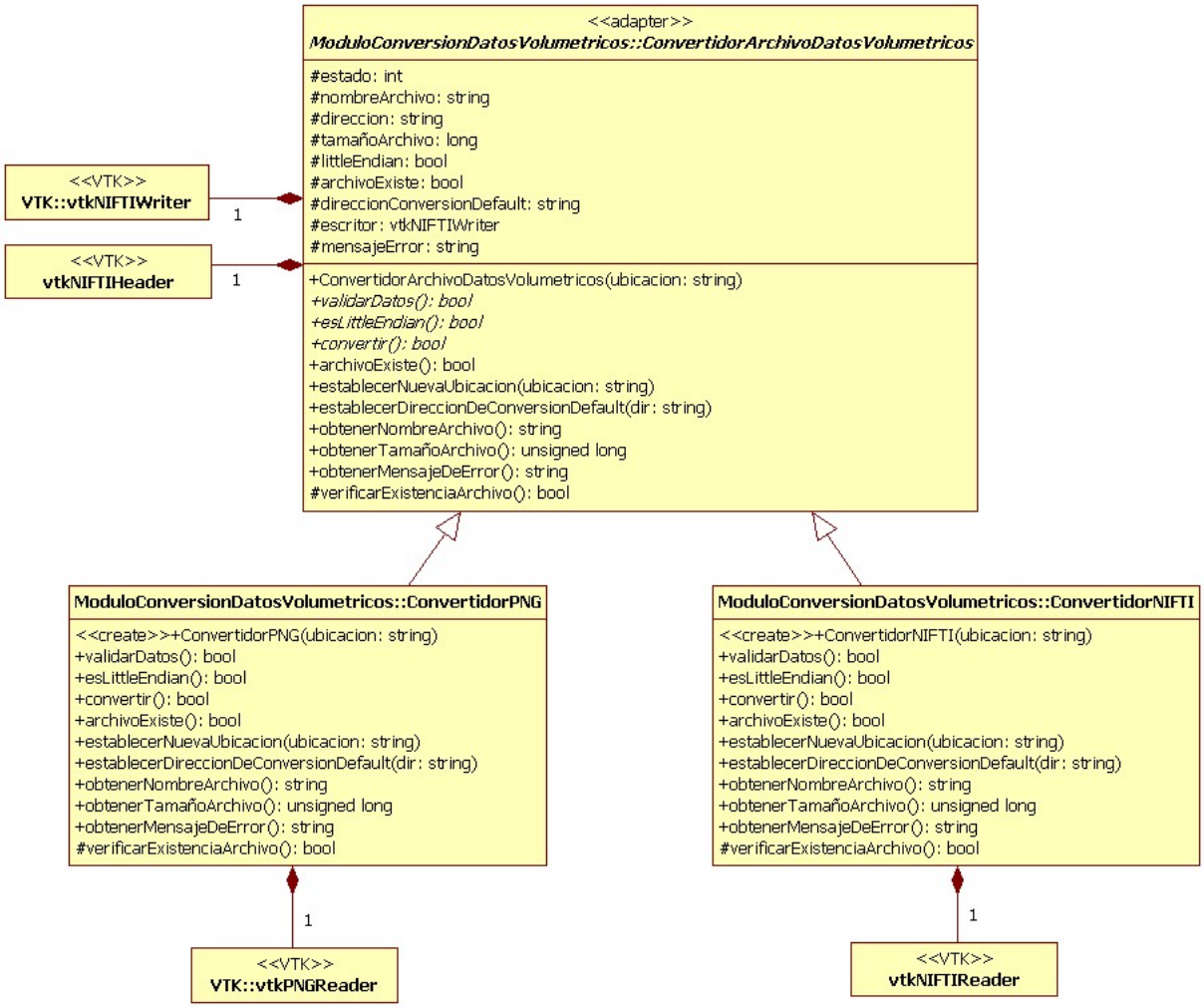


Figura 4.21: Diagrama clases del convertidor de datos volumétricos.

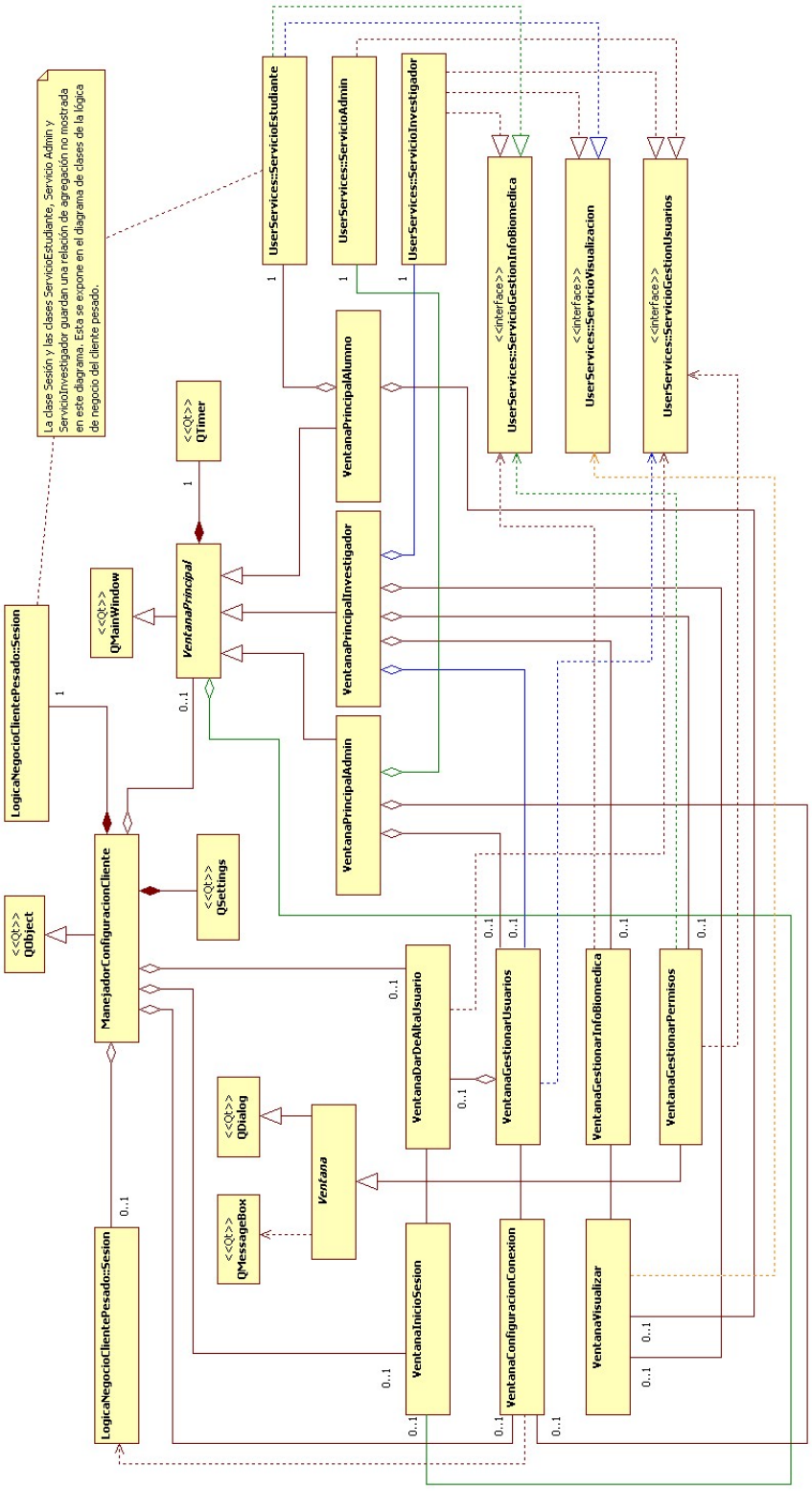


Figura 4.22: Diagrama de clases para la interfaz gráfica del cliente pesado.

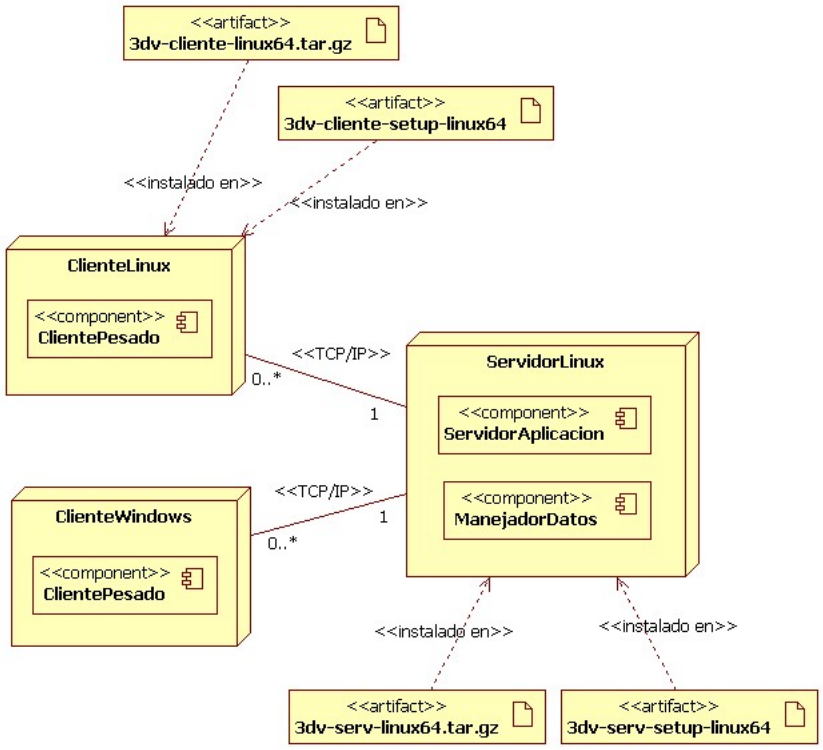


Figura 4.23: Vista de puesta en operación del sistema. Las figuras con forma de cubo simbolizan nodos dentro de los que se ejecutan los componentes del sistema. Dichos componentes aparecen dibujados como rectángulos en el interior de cada cubo. Por otra parte, los archivos requeridos para la instalación de los componentes se ilustran como cuadros fuera de los nodos; de éstos surgen flechas con líneas punteadas que indican en qué nodo deben ser colocados. La notación del diagrama corresponde al estándar UML.

característica de esos sistemas. La correcta ejecución de los componentes ha sido probada en los siguientes sistemas operativos:

- Ubuntu 12.04 (Precise Pangolin), 32 bits
- Ubuntu 14.04 (Trusty Tahr), 64 bits
- CentOS 7, 64 bits

El entorno final de operación del servidor fue el sistema operativo CentOS 7 de 64 bits. Los archivos requeridos para correr los componentes del servidor y el manejador de datos se han empacado en archivo comprimido denominado "3dv-serv-linux64.tar.gz". Para efectuar la instalación de forma automática se propuso emplear un script codificado con comandos de Shell; el archivo denominado "3dv-serv-setup-linux64" corresponde a dicho script. El archivo comprimido contiene tres tipos de archivo: un archivo binario ejecutable, un script de Shell para inicializar al servidor como demonio informático, y varias bibliotecas dinámicas con la funcionalidad requerida por dicho servidor. Los detalles sobre esas librerías dinámicas y sobre las acciones efectuadas por el script de Shell para llevar a cabo la instalación se presentan en el Apéndice E.

4.3.5.2. Nodos del cliente

El único componente que corre en los nodos cliente es de tipo ClientePesado, y su funcionalidad ha sido probada en los siguientes sistemas operativos:

- Ubuntu 12.04 (Precise Pangolin), 32 bits
- Ubuntu 14.04 (Trusty Tahr), 64 bits
- CentOS 7, 64 bits
- Windows XP, 32 bits

Los archivos de instalación para las distribuciones de GNU/Linux (Ubuntu y CentOS en este caso) son "3dv-cliente-linux64.tar.gz" y "3dv-cliente-setup-linux64" para versiones de 64 bits del sistema, y "3dv-cliente-linux32.tar.gz" y "3dv-cliente-setup-linux32" para las versiones de 32 bits. Nótese que la figura 4.23 únicamente hace alusión a la versión de 64 bits del cliente, aunque esto sólo es para fines de brevedad. De forma análoga al servidor, los archivos con extensión .tar.gz contienen a todas las librerías dinámicas de VTK y Qt, y al archivo binario ejecutable. El archivo sin extensión es un script de Shell que debe ejecutarse para realizar la instalación de forma automática. Los detalles sobre las librerías dinámicas y sobre las acciones efectuadas por el script de Shell para llevar a cabo la instalación se presentan en el Apéndice E.

Capítulo 5

Resultados

Una vez presentada la arquitectura para el sistema de visualización 3D, este capítulo expone los resultados de analizar y evaluar el cumplimiento de los atributos de calidad otorgados por la misma.

5.1. Evaluación de los atributos de calidad

Aunque en este proyecto no se aplicó completamente el modelo de referencia RESVEP (ver Sección 3.1.2), sus indicadores de atributos de calidad resultaron útiles para evaluar y analizar el cumplimiento de atributos de calidad por parte de la arquitectura. Los escenarios obtenidos mediante QAW (ver Cuadro 4.2) revelaron cuáles son los atributos de calidad más relevantes para el sistema de visualización 3D. Específicamente, los votos emitidos revelaron que los atributos de calidad tienen el siguiente orden de relevancia:

1. Modificabilidad (3 escenarios, 18 votos)
2. Disponibilidad (3 escenarios, 7 votos)
3. Seguridad (3 escenarios, 6 votos)
4. Usabilidad (3 escenarios, 5 votos)
5. Rendimiento (3 escenarios, 4 votos)
6. Facilidad de pruebas (1 escenario, 2 votos)

No obstante que estos atributos son considerados como sub-atributos de calidad por RESVEP (ver Cuadro 3.1), se procedieron a calcular no sólo sus indicadores correspondientes sino también el de los demás sub-atributos relacionados, para finalmente determinar el indicador del atributo de calidad más general. Los resultados para cada uno de ellos se presentan en las subsecciones a continuación.

5.1.1. Mantenibilidad

Los sub-atributos de calidad considerados por RESVEP como parte del atributo de mantenibilidad son: modularidad, reusabilidad, analizabilidad, modificabilidad y facilidad de pruebas. Los módulos considerados para el cálculo de los indicadores para esos sub-atributos se enlistan en el Cuadro 5.1. No todos los sub-módulos dentro de cada módulo general han sido considerados, y las razones son las siguientes:

Módulo	Sub-módulos
ServidorAplicacion	▪ Sesion ▪ ServicioUsuario ▪ ComandosServidor ▪ Visualizacion3D
ClientePesado	▪ Sesion ▪ ModuloConversionDatosVolumetricos ▪ GUICliente
ManejadorDatos	*Ninguno. Sólo se considera al módulo general.*
ComunicacionServidor-Aplicacion, ComunicacionCliente-Pesado	▪ ManejadorSesion ▪ Socket ▪ Iterador
11 módulos en total	

Cuadro 5.1: Módulos empleados para calcular los indicadores de mantenibilidad.

- Los módulos ServicioAdmin, ServicioInvestigador y ServicioAlumno se han tomado como uno solo: ServicioUsuario, la superclase de todos ellos. Esto es porque todo aspecto relacionado con la mantenibilidad aplica de igual modo a cada servicio.
- Los módulos de comandos, ComandosCliente y ComandosServidor, se contabilizaron como uno solo, pues los aspectos de mantenibilidad aplicables a alguno de ellos son aplicables también al otro.
- El módulo ManejadorDatos también se contó como uno solo, pues se ha estimado que los cambios a cualquiera de sus sub-módulos tienen el mismo impacto en el sistema.
- Sólo se han contado los submódulos del módulo ComunicacionServidorAplicacion, ya que el módulo análogo en el cliente consta de los mismos sub-módulos (a excepción de ManejadorSesion).

Los resultados obtenidos para los indicadores de mantenibilidad son los siguientes:

- **Modularidad (m1)**

$$m1 = 1 - \frac{A}{B} = 1 - \frac{3}{11} \approx 0.73$$

Donde:
A – número de módulos que causan alto impacto al ser modificados.
B – número total de módulos.
Los módulos que generan un alto impacto al ser modificados son:

1. Sesion (servidor)
2. Sesion (cliente)
3. ComandosServidor

- **Reusabilidad (m2)**

$$m2 = \frac{A}{B} = \frac{5}{11} \approx 0.45$$

Donde:
A – número de módulos reutilizables.
B – número total de módulos.
Los módulos que son más reutilizables son:

1. Visualizacion3D
2. ModuloConversionDatosVolumetricos
3. ManejadorDatos
4. Socket
5. Iterador

- **Analizabilidad (m3)**

$$m3 = \frac{A}{B} = \frac{7}{11} \approx 0.64$$

Donde:
A – número de módulos que pueden ser fácilmente diagnosticados de deficiencias o causas de error.
B – número total de módulos.
Los módulos que son más rápidamente analizados en búsqueda de errores y defectos, y cuyo impacto puede ser fácilmente vislumbrado son:

1. ServicioUsuario
2. ComandosServidor
3. Visualizacion3D
4. ModuloConversionDatosVolumetricos
5. Socket
6. Iterador
7. ManejadorDatos

- **Modificabilidad (m4)**

$$m4 = \frac{A}{B} = \frac{5}{11} \approx 0.45$$

Donde:
A – número de módulos que pueden ser modificados sin introducir defectos ni degradar el rendimiento.
B – número total de módulos.
Los módulos cuyo diseño se ha enfocado en minimizar el efecto de posibles cambios son:

1. ModuloConversionDatosVolumetricos
2. ManejadorDatos
3. GUICliente
4. Socket
5. Iterador

- **Facilidad de pruebas (m5)**

$$m5 = \frac{A}{B} = 0.00$$

Donde:
A – número de casos en los que un módulo puede ser adecuadamente probado.
B – número de casos de pruebas.
No hay puntuación para este indicador, dado que no se efectuaron casos de pruebas.

Con todo lo anterior, se obtuvo que:

$$\textbf{Mantenibilidad (M)} = \frac{m1+m2+m3+m4+m5}{5} = 0.45$$

5.1.2. Confiabilidad

Los sub-atributos de calidad considerados por RESVEP como parte del atributo de confiabilidad son: madurez, disponibilidad, tolerancia a fallas y capacidad de recuperación. Sin embargo, dado que los indicadores de madurez y de disponibilidad requieren largos periodos de observación y uso del sistema para ser evaluados (ver Apéndice C), únicamente se calcularon los de tolerancia a fallas y capacidad de recuperación. Dichos cálculos se hicieron considerando los casos de uso identificados; sólo aquéllos que ya están implementados se incluyeron en el análisis de confiabilidad, y son los siguientes:

1. Iniciar sesión
2. Finalizar sesión
3. Restaurar sesión
4. Dar de alta
5. Dar de baja
6. Consultar datos
7. Modificar datos
8. Validar datos
9. Efectuar representación por volumen
10. Efectuar representación conjunta
11. Aplicar transformación espacial
12. Modificar parámetros numéricos

Los resultados obtenidos de calcular los indicadores son los siguientes:

- **Tolerancia a fallas (r3)**

$$r3 = \frac{A}{B} = \frac{0}{12} \approx 0.00$$

Donde:
A – número de funciones que operan correctamente a pesar de fallas en hardware o software.
B – número de funciones evaluadas.
Por el momento, no se han incluido en el diseño arquitectónico mecanismos para manejar fallos en software o hardware.

- **Capacidad de recuperación (r4)**

$$r4 = \frac{A}{B} = 0.00$$

Donde:
A – número de funciones que pueden recuperar información perdida o dañada, o devolver al sistema a un estado correct en caso de fallas.
B – número de funciones que manejan datos relevantes para el sistema.
Por el momento, en el diseño arquitectónico no se han incluido conceptos que permitan la recuperación de información perdida, o que devuelvan al sistema a la normalidad en caso de errores graves.

Con todo lo anterior, se obtuvo que:

$$\textbf{Confiabilidad (R)} = \frac{r3+r4}{2} \approx 0.00$$

5.1.3. Seguridad

Los sub-atributos de calidad considerados por RESVEP como parte del atributo de seguridad son: confidencialidad, integridad, no-repudio, constancia de acciones y autenticación. La evaluación de la seguridad se hizo considerando los casos de uso indentificados; éstos y las relaciones que tienen entre sí se muestran en las Figuras 4.2, 4.4 y 4.6. De entre ellos, los que manejan datos relevantes para la seguridad del sistema son:

1. Dar de alta
2. Dar de baja
3. Consultar datos
4. Iniciar sesión
5. Restaurar sesión

Los resultados obtenidos de calcular los indicadores son los siguientes:

- **Confidencialidad (s1)**

$$s1 = 1 - \frac{A}{B} = 1 - \frac{0}{5} = 1.00$$

Donde:
A – número de funciones (casos de uso) que no están seguros contra revelación de datos, ya sea accidental o deliberada.
B – número de funciones que manejan datos importantes para la seguridad.
La funcionalidad del sistema no revela información sensible de modo alguno. Asimismo, cada usuario sólo tiene acceso a la funcionalidad estipulada por su rol, y a la información de su propiedad y/o a la explícitamente permitida por el propietario de la misma.

- **Integridad (s2)**

$$s2 = \frac{A}{B} = \frac{5}{5} = 1.00$$

Donde:
A – número de funciones que previenen accesos o modificaciones no autorizadas.
B – número de funciones que manejan datos importantes para la seguridad.
Todos los casos de uso que leen o escriben datos relevantes para la seguridad evitan accesos indebidos. Esos casos de uso son:

1. Dar de alta
2. Dar de baja
3. Consultar datos
4. Iniciar sesión
5. Restaurar sesión

- **No-rechazo (s3)**

$$s3 = \frac{A}{B} = 0.00$$

Donde:
A – número de funciones que registran sus eventos en un archivo de historial.
B – número de funciones que requieren un archivo de historial.
Por el momento, el sistema no genera archivo de historial alguno.

- **Constancia de acciones (s4)**

$$s4 = \frac{A}{B} = 0.00$$

Donde:
A – número de funciones cuya actividad puede ser fácil y correctamente identificada en un archivo de historial.
B – número de funciones que requieren un archivo de historial.
Por el momento, el sistema no genera archivo de historial alguno.

- **Autenticidad (s5)**

$$s5 = \frac{A}{B} = \frac{100\,\%}{100\,\%} = 1.00$$

Donde:
A – porcentaje real de verificación de usuarios y/o recursos.
B – porcentaje esperado de verificación de usuario y/o recursos.
Hacer uso de cualquier funcionalidad requiere que el usuario se haya identificado mediante un nombre de usuario y una contraseña (preferentemente) segura. Sin esta identificación simplemente no es posible acceder al sistema.

Con todo lo anterior, se obtuvo que:

$$\textbf{Seguridad (S)} = \frac{s1+s2+s3+s4+s5}{5} \approx 0.60$$

5.1.4. Usabilidad

Los sub-atributos de calidad considerados por RESVEP como parte del atributo de usabilidad son: facilidad para aprendizaje, facilidad de uso, protección contra errores y estética de la IGU (interfaz gráfica de usuario). Para medir este indicador se diseñó una prueba de usabilidad que consistió en varias actividades que guiaran a los usuarios a través de todos

los tres casos de uso principales: manejar sesión, gestionar registros y visualizar (ver Figura 4.1). Se registró el tiempo que cada uno necesitó para efectuar las actividades, el cual se empleó para calcular los indicadores de facilidad de aprendizaje y facilidad de uso.
Los resultados obtenidos son los siguientes:

- **Facilidad de aprendizaje (u1)**

$$u1 = \frac{\sum Te}{\sum Tr} = \frac{765.00}{747.80} \approx 1.02$$

Donde:
Te – tiempo promedio esperado para aprender una unidad de software (en esta ocasión, el tiempo para aprender un caso de uso principal).
Tr – tiempo promedio real requerido para aprender una unidad de software.
Los tiempos promedios requeridos y esperados para aprender a recorrer cada caso de uso principal se presentan en el Cuadro 5.2.

Caso de uso	Tiempo esperado (Te)	Tiempo real (Tr)
Manejar sesión	30	43.2
Gestionar registros	345	338.0
Visualizar	390	366.6

Cuadro 5.2: Tiempos esperados y reales promedio requeridos para aprender a recorrer los casos de uso principales. Las unidades están en segundos.

- **Facilidad de uso (u2)**

$$u2 = \frac{\sum Te}{\sum Tr} = \frac{170.00}{168.4} \approx 1.01$$

Donde:
Te – tiempo promedio esperado para usar correctamente una unidad de software (en este caso, para recorrer completamente un caso de uso principal).
Tr – tiempo promedio real para usar correctamente una unidad de software.
Los tiempos promedios requeridos y esperados para aprender a recorrer cada caso de uso principal se presentan en el Cuadro 5.3.

Caso de uso	Tiempo esperado (Te)	Tiempo real (Tr)
Manejar sesión	30	17.4
Gestionar registros	80	90.4
Visualizar	60	60.6

Cuadro 5.3: Tiempos esperados y reales promedio requeridos para recorrer los casos de uso principales. Las unidades están en segundos.

- **Protección contra errores (u3)**

$$u3 = \frac{A}{B} = \frac{1}{1} = 1.00$$

Donde:
A – número de elementos de interfaz de usuario que protegen al usuario de cometer errores.
B – número de elementos de interfaz de usuario evaluadas.
Todos los controles de la IGU en los que el usuario debe introducir texto, especificar valores o rangos, así como escoger entre opciones evitan que se hagan elecciones inválidas.

- **Estética de la IGU (u4)**

$$u4 = \frac{A_{prom}}{10 \times B} = \frac{\frac{8+9+8}{3}}{10 \times 1} \approx 0.83$$

Donde:
A_{prom} – Calificación (de 0 a 10) promedio asignada a la IGU de acuerdo a los usuarios del sistema.
B – Número de elementos de IGU evaluados.
Tres usuarios emitieron una calificación que expresara su parecer con respecto al atractivo visual de la IGU en general. Las calificaciones fueron promediadas y el resultado empleado para calcular el indicador en cuestión.

Con todo lo anterior, se obtuvo que:

$$\textbf{Usabilidad (U)} = \frac{u1+u2+u3+u4}{4} \approx 0.97$$

5.1.5. Rendimiento

Los sub-atributos de calidad considerados por RESVEP como parte del atributo de rendimiento son: tiempo de procesamiento y empleo de recursos. No obstante, el único indicador que se calculó es el de tiempo de procesamiento, puesto que es considerado más relevante y, por el momento, no se trabaja con información biomédica que ocupe grandes cantidades de memoria, además de que el sistema no espera atender a demasiados usuarios de forma simultánea.

- **Tiempo de procesamiento (pe1)**

Las tareas que más tiempo consumen son las de visualización, por lo que las mediciones de tiempo de procesamiento se centraron en ellas. La representación por volumen se probó sobre varios conjuntos de datos. El Cuadro 5.4 presenta los tiempos promedio obtenidos para varios conjuntos de imágenes que contienen sólo información anatómica. A pesar de las diferencias entre las características de los conjuntos, los tiempos de procesamiento no variaron demasiado.
En lo que respecta a la visualización conjunta, únicamente se probó con un par de datos volumétricos: uno con información anatómica y otra funcional. Ambos conjuntos tienen

# volumen	Largo	Ancho	Num. rebanadas	Num. bits	Espacio requerido (MB)	Tiempo procesamiento (segs.)
1	256	256	181	16	22.26	1.65
2	256	256	200	8	12.50	1.75
3	91	109	91	8	0.88	2.17
4	320	320	180	16	35.15	2.05
5	64	64	45	16	0.36	1.95

Cuadro 5.4: Tiempos de procesamiento obtenidos para la representación por volumen (sólo ray-casting) efectuada sobre varios conjuntos de datos.

dimensiones de $256 \times 256 \times 181$ voxeles codificados a 16 bits. El tiempo de procesamiento promedio fue de 38.38 segundos.
Considerando lo anterior, tenemos que:

$$pe1 = \frac{\sum Te}{\sum Tr} = \frac{10+10}{2.17+38.38} = \frac{20.00}{40.55} \approx 0.49$$

Donde:
Te – tiempo esperado de respuesta de una unidad de software.
Tr – tiempo real de respuesta de una unidad de software.
En la sumatoria se consideró el máximo tiempo real obtenido para la representación por volumen, y el obtenido para la representación conjunta.
El tiempo esperado para la repuesta de ambos tipos de visualización se tomó como 10 segundos. Esto se considera como lo máximo que un usuario mantiene su atención en un cuadro de diálogo. Pasado ese tiempo, el usuario querrá dedicarse a atender otras tareas mientras el sistema completa sus actividades, por lo que un aviso del tiempo estimado para concluirlas es fundamental [79].

Con todo lo anterior, se obtuvo que:

$$\textbf{Rendimiento (PE)} = pe1 = 0.49$$

5.1.6. Idoneidad de la funcionalidad

Este atributo de calidad se refiere al grado en que el sistema provee la funcionalidad que satisface las necesidades de los usuarios. Aunque no fue considerada en la ejecución QAW, se consideró necesario calcular su indicador conforme lo marca RESVEP. Son dos los sub-atributos que deben ser valorados: funcionalidad correcta y precisión. Los cálculos se efectuaron con base en los casos de uso considerados en el diseño arquitectónico, y son los siguientes:

1. Iniciar sesión
2. Finalizar sesión
3. Restaurar sesión
4. Dar de alta
5. Dar de baja
6. Consultar datos
7. Modificar datos
8. Validar datos
9. Efectuar representación por volumen
10. Efectuar representación conjunta
11. Aplicar transformación espacial
12. Modificar parámetros numéricos

Los resultados obtenidos son los siguientes:

- **Funcionalidad correcta (fs1)**

$$fs1 = 1 - \frac{A}{B} = 1 - \frac{1}{12} \approx 0.92$$

Donde:
A – número de funciones faltantes o con errores.
B – número de funciones consideradas en el diseño arquitectónico. Sólo un caso de uso no se ha implementado por completo:

1. Modificar datos

- **Precisión (fs2)**

$$fs2 = 1 - \frac{A}{B} = 1 - \frac{2}{12} \approx 0.83$$

Donde:
A – número de funciones que presentan inconsistencias con respecto a su descripción.
B – número de funciones consideradas en el diseño arquitectónico.
Se detectaron algunas inconsistencias al momento de recorrer los siguientes casos de uso:

1. Modificar datos
2. Efectuar transformación espacial

Con todo lo anterior, se obtuvo que:

$$\textbf{Idoneidad de la funcionalidad (FS)} = \frac{fs1+fs2}{2} \approx 0.88$$

El Cuadro 5.5 expone todos los indicadores de atributos de calidad obtenidos hasta este punto.

Atributo de calidad	Sub-atributo de calidad	Indicador de sub-atributo	Indicador general
Mantenibilidad	Modularidad	0.73	**0.45**
	Reusabilidad	0.45	
	Analizabilidad	0.64	
	Modificabilidad	0.45	
	Facilidad de pruebas	0.00	
Confiabilidad	Tolerancia a fallas	0.00	**0.00**
	Capacidad de recuperación	0.00	
Seguridad	Confidencialidad	1.00	**0.60**
	Integridad	1.00	
	No-rechazo	0.00	
	Constancia de acciones	0.00	
	Autenticación	1.00	
Usabilidad	Facilidad de aprendizaje	1.02	**0.97**
	Facilidad de uso	1.01	
	Protección contra errores	1.00	
	Estética de la IGU	0.83	
Rendimiento	Tiempo de procesamiento	0.49	**0.49**
Idoneidad de la funcionalidad	Funcionalidad correcta	0.92	**0.88**
	Precisión	0.83	

Cuadro 5.5: Resumen de los indicadores de atributos de calidad para el sistema de visualización 3D.

Capítulo 6

Discusión

A lo largo de sus más de 20 años de introducción al mercado, los sistemas de visualización tridimensional han demostrado ser una herramienta muy poderosa para profesionales clínicos y académicos. No es descabellado suponer que, dentro de pocos años, formarán parte del quehacer cotidiano en entornos clínicos, apoyando a diferentes especialidades médicas. Los rápidos avances tecnológicos en hardware computacional capaz de procesar enormes cantidades de datos eficientemente, la reducción sustancial de sus costos, así como la gran cantidad de investigación en interfaces humano-máquina y con aplicaciones prácticas de la representación 3D que actualmente se lleva a cabo, garantizan un papel fundamental para estos sistemas de visualización en la síntesis y el análisis de información biomédica.
El desarrollo de estos sistemas debería estar siempre guiado por métodos centrados en arquitectura de software, pues sólo de este modo puede garantizarse que el sistema cumplirá con requerimientos de calidad. La importancia de incorporar decisiones de diseño encaminadas a cumplir con esos requerimientos no debe ser infravalorada, dada la naturaleza sumamente delicada de la información biomédica y de las fuertes repercusiones que los sistemas biomédicos pueden tener en el desempeño del personal médico y académico. Es por eso que al proceso de desarrollo de software empleado en este proyecto se enfocó fuertemente en arquitectura de software, teniendo presente no sólo el cumplimiento de un conjunto de atributos de calidad sino también contribuir a la discusión de cómo pueden ser abordados. Además, a este proceso se incorporó una sistematización de la revisión bibliográfica que apoyara la recopilación de los estudios primarios relevantes, pues éstas constituyen la evidencia de que las decisiones tomadas para el diseño del sistema son, efectivamente, las idóneas para lograr la calidad buscada. Cabe mencionar que ninguno de esos estudios primarios expone un diseño arquitectónico enfocado en atributos de calidad como el presentado a lo largo de este trabajo.

6.1. Formalización del estado del arte

El EMS, descrito a lo largo de la Sección 2.1, es un proceso de recopilación, clasificación y análisis de material bibliográfico muy útil e interesante que, desafortunadamente, únicamente ha cobrado popularidad en temas de investigación médica y de ingeniería de software. No obstante, luego de haberlo ejecutado en el curso de este proyecto resultó evidente que es plenamente aplicable a cualquier área de investigación.

La principal importancia del EMS presentado en este trabajo radicó en que permitió generar un mapa del estado del arte en los temas principales que atañen al proyecto. Otra ventaja importante es que el EMS marca una serie de actividades que requieren ser documentadas y, por lo tanto, el proceso entero es completamente repetible. En consecuencia, el estudio puede volver a ejecutarse y la documentación actualizarse con relativa facilidad siempre que sea necesario; esto resulta muy conveniente para cualquier proyecto que esté iniciando y/o al cual se pretenda dar continuidad durante un largo tiempo.
La correcta formulación de las preguntas de investigación resultó determinante para los resultados que se obtuvieron en los motores de búsqueda. La inclusión de los conceptos de intervención, población y resultado en cada pregunta, así como la cuidadosa selección de sinónimos y de términos a excluir, fueron importantes para obtener resultados bien centrados en el proyecto. A pesar de todo, ello no evitó que varios resultados desviados (es decir, estudios que no estaban relacionados con el proyecto) fueran incluidos entre los resultados arrojados inicialmente, particularmente en motores como Springer Link. Cabe mencionar que la funcionalidad ofrecida por varios motores de búsqueda para filtrar los resultados por tema, tipo de publicación, fecha, autores, etc., resulta muy conveniente si se obtienen muchos de esos resultados, o simplemente si la cantidad de publicaciones arrojada por el buscador es excesiva.

6.2. Experiencias con QAW

La ejecución de QAW es factible en un entorno académico como en el que se desenvolvió este proyecto, pues los interesados en el sistema suelen ser profesores investigadores y, quizás, alumnos con buena disposición para otorgar parte de su tiempo a las actividades del taller. Sin embargo, en circunstancias como las dadas dentro de un entorno empresarial o clínico, muchos de los interesados podrían no contar con el tiempo ni el entusiasmo para participar. Si este fuera el caso, es posible que deban adoptarse otras estrategias para recolectar escenarios de atributos de calidad, como concertar citas individuales con los principales usuarios o dividir el taller en varias sesiones en diferentes fechas.
La preparación de material de apoyo para QAW, como pueden ser guías, instructivos y presentaciones, debió efectuarse con cuidado y detenimiento. Se concluyó que es importante procurar un estilo de redacción que sea familiar para los participantes del taller, evitando tecnicismos innecesarios o definiéndolos claramente cuando sean indispensables, como en el caso de los conceptos de "arquitectura de software" o de "atributos de calidad", entre otros. Por su parte, la lluvia de escenarios también debe vigilarse en la ejecución del taller, pues una tendencia común entre los participantes fue la de proponer escenarios de funcionalidad en lugar de atributos de calidad.

6.3. Importancia de los escenarios de atributos de calidad

El número de votos asignados a cada escenario de atributo de calidad fue el factor preponderante para determinar su influencia en el sistema; en este sentido, la modificabilidad se

impuso de manera sobresaliente por lo que la mayor parte de los conceptos de diseño arquitectónico se dirigieron a satisfacer este atributo. La lista de estos conceptos para el sistema completo presentada en el Cuadros 4.7 son los que más se aplicaron tanto al más alto nivel de diseño como en el más detallado. Esta tendencia puede observarse en los sub-módulos del servidor (ver Sección 4.3.2) y del cliente (ver Sección 4.3.4). No obstante, la cantidad de escenarios relacionados con otros atributos distintos a la modificabilidad, así como el hecho de que obtuvieron cantidades similares de votos entre sí, fue un indicador de que la arquitectura también debía contemplar conceptos de diseño para satisfacer esos atributos.

6.4. Diseño y evaluación de la arquitectura

La evaluación de la arquitectura para verificar el cumplimiento de los atributos de calidad es una tarea importante. En la Sección 3.1.2 se mencionó que emplear alguno de los métodos más conocidos, como el ATAM del SEI, hubiera resultado sumamente impráctico para este proyecto. Sin embargo, el modelo de referencia RESVEP presenta entre sus actividades una alternativa viable para realizar dicha evaluación. La aplicación de sus indicadores de valor de calidad resultó relativamente sencilla.

6.4.1. Mantenibilidad

La mantenibilidad fue evaluada de acuerdo a las métricas propuestas por RESVEP dado que la modificabilidad, el atributo de calidad prioritario para el sistema de visualización 3D, está estipulada como un sub-atributo de la mantenibilidad. La facilidad de pruebas también está clasificada de esta forma. Los cálculos para todos los indicadores posicionados bajo la mantenibilidad se efectuaron siguiendo las indicaciones de RESVEP.

6.4.1.1. Modularidad

El diseño arquitectónico se desarrolló teniendo muy presentes los aspectos de modularidad. La aplicación de tácticas arquitectónicas como coherencia semántica, uso de interfaces y patrones de diseño como “adapter”, “facade” y DAO favorecieron mucho el bajo acoplamiento entre módulos, reduciendo así el impacto que las modificaciones en ciertos módulos pudieran tener sobre los demás; esto además de que el estilo cliente-servidor naturalmente reduce el acoplamiento entre ambos componentes. La aplicación de todos esos patrones y tácticas llevan al sistema a cumplir los escenarios de atributos de calidad #1 y #4, pues el bajo acoplamiento también facilita la inclusión de nueva funcionalidad. Esto se discute detalladamente más adelante en la Sección 6.5.
El caso más relevante de potencial impacto se da entre las clases Sesion y los módulos de comandos ComandosCliente y ComandosServidor, ya que éstos últimos son susceptibles de sufrir cambios constantes en sus interfaces y es muy probable que, conforme el sistema evolucione, nuevas subclases deban ser añadidas a la familia de comandos. Todo esto requeriría que los módulos Sesión se actualizaran para llamar adecuadamente a los módulos de comandos.

6.4.1.2. Reusabilidad

Las tácticas y patrones de diseño mencionadas anteriormente también favorecen la reusabilidad. Módulos como Visualizacion3D, ModuloConversionDatosVolumetricos y ManejadorDatos presentan muy bajo acoplamiento con otros módulos del sistema, y podrían ser empleados fácilmente en otros proyectos de software. Sin embargo, no todos los módulos presentan esas características; es el caso de Sesion, ServiciosUsuario, ComandoServidor, ComandoCliente y GUICliente, cuyas responsabilidades están muy centradas en controlar las secuencias específicas de actividades en el sistema. Ello ha llevado a un indicador bajo para la reusabilidad pero no por ello poco significativo, pues implica que prácticamente el 40 % de los módulos más generales del sistema son reutilizables.

6.4.1.3. Analizabilidad

La analizabilidad en el sistema es propiciada por la coherencia semántica procurada en el diseño arquitectónico. Al haber propuesto módulos con responsabilidades muy específicas y haber procurado un bajo acoplamiento se ha facilitado el análisis de dónde pueden darse cambios en el sistema.

6.4.1.4. Modificabilidad

El indicador de modificabilidad está relacionado con las decisiones de diseño que permitieron la modularidad y la reusabilidad. Es decir, varios de los módulos pueden sufrir cambios sin afectar a los demás. El escenario de atributo de calidad #2 está relacionado con este sub-atributo; el uso de interfaces y la generalización de las mismas hace posible que módulos como Visualizacion3D, ManejadorDatos y ModuloConversionDatosVolumetricos posibilita que puedan compilarse por separado y ligarse como librerías de software individuales.
Por otra parte, RESVEP incluye a la afectación del rendimiento como un elemento que impacta a la modificabilidad; en ese sentido es evidente que la selección de herramientas de software (como las empleadas para efectuar la visualización 3D) será un factor decisivo. Por tal motivo no se ha incluido al módulo Visualizacion3D entre los "modificables", aunque es claro que la herramienta empleada actualmente para su implementación (VTK) puede ser fácilmente reemplazada por alguna otra. Varios posibles candidatos fueron expuestos y comentados a lo largo de la Sección 2.3.2.2.

6.4.1.5. Facilidad de pruebas

Finalmente, es preciso recalcar que no se establecieron casos de pruebas para los componentes del sistema, por lo que el indicador para el sub-atributo de facilidad de pruebas fue de cero. El trabajo a futuro para el proyecto estipula el diseño de estas pruebas; sólo así podrán tomarse las medidas para un mejor cumplimiento de los atributos de calidad solicitados al sistema, y podrán identificarse las condiciones bajo las cuales dichos atributos dejan de lograrse.

6.4.2. Confiabilidad

La disponibilidad, uno de los sub-atributos de la confiabilidad según RESVEP, es el segundo atributo de calidad más relevante de acuerdo a los resultados de ADD. A pesar de que el indicador de confiabilidad obtenido fue de 0.00, el sistema de visualización cumple con la mayoría de los sub-atributos de calidad establecidos relacionados con este atributo, incluyendo el #3 que fue el que obtuvo más votos. Específicamente, el patrón cliente-servidor aisla los errores que pudieran ocurrir en el cliente de los del servidor, y viceversa; un error que ocasionara el colapso de un nodo cliente no afectaría en lo absoluto al servidor. Asimismo, el riesgo de corrupción de la información contenida en el servidor es muy reducida, puesto que las tareas de visualización únicamente requieren efectuar lectura de datos. Además, en caso de ocurrir fallas eléctricas, el nodo que alberga al servidor cuenta con el apoyo de una unidad de alimentación ininterrumpida (normalmente conocidas por sus siglas en inglés, UPS). Sin embargo, todo lo anterior no se vio reflejado en el indicador para la confiabilidad dado que los sub-atributos de disponibilidad y madurez, que son los que sí consideran todo anterior, no fueron evaluados. La razón de esto es que esos sub-atributos requieren muy largos periodos de observación del sistema para poder ser diagnosticados.
Algunas tácticas arquitectónicas fueron consideradas como posibles de ser incluidas para mejorar el indicador de confiabilidad, aunque éstas no formaron parte del diseño durante este ciclo de desarrollo. Dichas tácticas son: perro guardián, latido, ping/echo y redundancia pasiva o activa. Las primeras dos podrían ser particularmente útiles para vigilar la correcta ejecución de las tareas de visualización; consistirían en un componente adicional corriendo en el servidor como un proceso (un hilo) independiente, verificando constantemente la actividad en los procesos de visualización y reportando anomalías. La táctica de ping/echo serviría para verificar que el servidor está disponible en todo momento y que, por lo tanto, puede seguir atendiendo a los clientes. Todo esto favorecería al indicador de calidad para la tolerancia a fallas. Finalmente, las tácticas de redundancia generarían mejores indicadores para la capacidad de recuperación, pues siempre habría un duplicado de la información biomédica y sobre usuarios lista para ser recuperada en caso de pérdida por parte del servidor.

6.4.3. Seguridad

Los escenarios de atributos de calidad propuestos por los interesados hacen referencia a la seguridad del sistema, y no tanto a la seguridad de los datos. Aunque ningún escenario prioritario se relaciona con la seguridad, el número de escenarios propuestos y la cantidad de votos que obtuvieron permiten entrever que es un atributo relevante para los interesados.

6.4.3.1. Confidencialidad e integridad

El caso de uso Manejar Sesión es el que marca la secuencia de actividades que un usuario debe realizar para acceder a la funcionalidad del sistema y a la información dentro del mismo (ver Sección 4.1.2.1); estas actividades otorgan al sistema cierto grado de calidad en lo que a seguridad del sistema se refiere. En el contexto de tácticas arquitectónicas, lo estipulado por el caso de uso se denomina autenticación de usuarios; la idea es que el sistema garantice que el usuario sea en efecto quien dice ser, y que sólo pueda acceder a información de su

propiedad y/o a la que otro usuario le ha otorgado acceso. Por lo tanto, prácticamente toda la funcionalidad y la información biomédica del sistema está protegida de accesos no autorizados. Además, para cumplir con el escenario de atributo de calidad #5, el sistema también bloquea el uso de la interfaz gráfica cuando ha pasado cierto tiempo de inactividad. Todo lo anterior otorga al sistema puntuaciones elevadas para los indicadores de confidencialidad y de integridad. Además, como se expuso en el estado del arte mediante el EMS, los sistemas de visualización 3D presentados en estudios primarios prácticamente no tienen en cuenta la protección del sistema y sus datos (ver Cuadro 2.7); esto posiciona a este sistema en puntos del mapa del EMS (Figura 2.1) no ocupados por estudio primario alguno.

6.4.3.2. No-rechazo y constancia de acciones

Estos sub-atributos enfatizan la importancia de una táctica de seguridad: la generación de historiales de actividad por parte del sistema. Entre los escenarios de atributos de calidad hubo uno que mencionaba dicha generación, aunque resultó ser de baja prioridad. Es por eso que el empleo de esta táctica de seguridad no fue empleada en este ciclo de desarrollo, aunque su incorporación está contemplada para ciclos subsecuentes. Por el momento, su ausencia ha impactado negativamente al indicador de seguridad estipulado por RESVEP, habiendo obtenido nulas calificaciones en estos sub-atributos.

6.4.4. Usabilidad

La usabilidad recibió poca atención por parte de los interesados en el sistema. Los escenarios de atributos de calidad #6, #13 y #15 se refieren a este atributo de calidad, y el sistema cumple los primeros dos sin ningún problema, pues no requieren el empleo de tácticas arquitectónicas específicas y las herramientas provistas por Qt facilitan mucho la interacción con la IGU. Para cumplir con el escenario #15 sí sería necesario realizar ciertos cambios en el diseño modular; el patrón de diseño denominado Comando facilita la implementación de sistemas que requieran deshacer acciones, por lo que podría emplearse para satisfacer dicho escenario en el futuro.
A pesar de ser un atributo poco prioritario para los interesados, se consideró conveniente obtener los indicadores de calidad estipulados por RESVEP para la usabilidad.

6.4.4.1. Facilidad de aprendizaje y facilidad de uso

Se aplicó una prueba de usabilidad específicamente diseñada para medir los sub-atributos de facilidad para aprendizaje y facilidad de uso. Los resultados fueron bastante buenos dado que los usuarios aprendieron a emplear la funcionalidad del sistema dentro de los tiempos esperados. Sin embargo, la forma en que se determinaron dichos tiempos esperados no es estándar, simplemente es un aproximado del tiempo que le requirió efectuar la prueba a un usuario que ya conocía el sistema. Por lo tanto, queda pendiente emplear una estrategia más formal para calcular los tiempos esperados que los indicadores de usabilidad solicitan.

6.4.4.2. Estética de la IGU

El indicador para la estética de la IGU se determinó mediante las calificaciones otorgadas por los usuarios que hicieron la prueba de usabilidad. En general dichas calificaciones fueron muy buenas, aunque algunos usuarios expresaron que sólo se debe a los pocos controles que se requieren para la funcionalidad implementada hasta ahora. Es decir, en ciclos de desarrollo subsecuentes es factible que se agregue nueva funcionalidad al sistema, por lo que será necesario aplicar estrategias para conservar la apariencia agradable e intuitiva de la interfaz gráfica.

6.4.4.3. Protección contra errores

En lo que respecta al índice de protección contra errores, todos los controles en la IGU que requieren introducción de texto, datos numéricos o la especificación de algún parámetro validan las elecciones del usuario antes de efectuar acción alguna. Por ejemplo, los campos para especificar los parámetros de visualización no permiten que se indiquen valores fuera de un rango específico; las ventanas para gestión de usuarios e información solicitan confirmación al usuario antes de dar de baja cualquier cosa. Muchos controles siguen esta línea de precauciones, y se espera que ciclos de desarrollo subsecuentes continúen implementándolas.

6.4.5. Rendimiento

El rendimiento suele ser un atributo de calidad bastante solicitado en los sistemas de visualización 3D; esto lo corroboran los estudios primarios arrojados por el EMS (ver Cuadro 2.7). Los conceptos de diseño empleados por este sistema de visualización para obtener rendimiento se basaron en las recomendadas por algunos de esos estudios, a pesar de que no resultó ser un atributo relevante de acuerdo al número de votos en los escenarios de atributos de calidad. El cliente pesado y los módulos de visualización del servidor fueron los que concentraron la mayor parte de los conceptos de diseño encaminados a conseguir buen rendimiento.

6.4.5.1. Comportamiento respecto a tiempos

El tiempo de procesamiento fue el indicador clave para este atributo de calidad. La representación volumétrica arrojó tiempos bastante aceptables; esto con respecto a lo máximo considerado razonable de mantener a un usuario en espera, y que es 10 segundos [79]. Con esto, los escenarios de atributos de calidad #11 y #16 se cumplen en el sistema. No sucedió así con la representación conjunta, que rebasó en más de 3 veces aquél tiempo máximo. Cabe mencionar que las imágenes empleadas para este tipo de visualización no fueron las idóneas puesto que el proyecto no contó con imágenes segmentadas para efectuar la representación por superficie, que es uno de los pasos intermedios para completar la visualización conjunta. Los algoritmos empleados para la generación de polígonos pueden volverse ineficientes si las imágenes no están debidamente pre-procesadas. Una vez sorteado este inconveniente, el indicador de calidad para el rendimiento definitivamente se incrementará.

6.5. Cómo integrar nueva funcionalidad

Durante entrevistas y conversaciones con usuarios interesados en el sistema se mencionaron ciertas funcionalidades que el sistema podría efectuar en el futuro. Algunas de ellas son las siguientes:

1. Visualización de isosuperficies en información volumétrica.
2. Selección de regiones de interés en información volumétrica.
3. Visualización de cortes arbitrarios en información volumétrica.
4. Restauración de la interfaz gráfica al estado en que se encontraba al momento de finalizar una sesión de usuario.
5. Permitir el guardado de capturas de pantalla al efectuar tareas de visualización.
6. Generar historial de accesos e intentos de acceso al sistema.

No es difícil notar que, muy probablemente, en futuros ciclos de desarrollo se requerirá añadir nuevas modalidades de visualización. Esto puede efectuarse con relativa sencillez en el módulo Visualizacion3D, en la subclase de ServicioUsuario correspondiente y en el módulo ComandosServidor ubicados en el servidor (en lo que respecta a estos dos últimos, los cambios también deberán hacerse en los módulos análogos del cliente).
Por otra parte, se requerirían modificaciones más sustanciales si se añadiera tipos de funcionalidad diferentes a la visualización 3D. Por ejemplo, una necesidad muy común en el LINI es el procesamiento de señales; supóngase que, en un futuro ciclo de desarrollo, se plantea ese requerimiento para el sistema. Considerando la vista de módulos, los cambios que muy probablemente habría que hacer son los siguientes:

- Agregar al servidor de aplicación un módulo cuyas responsabilidad principal será efectuar procesamiento de señales. Análogamente al módulo de visualización 3D, podría estar constituido por una familia de clases en donde cada subclase sería un tipo de procesamiento particular.
- Complementar la implementación del DAO de información biomédica para que sus métodos puedan gestionar señales biomédicas. El signado de los mismos no se modificaría.
- Modificar la interfaz realizada por la respectiva subclase de ServicioUsuario de tal forma que contenga uno o más métodos para efectuar procesamiento de señales; el cambio tendría que ser efectuado tanto del lado del servidor como del cliente pesado.
- Añadir una nueva subclase al módulo de comandos tanto del lado del cliente como del servidor. Su responsabilidad sería la de encapsular los parámetros necesarios para efectuar el procesamiento, así como el resultado que habrá de ser enviado al cliente.
- Adaptar la interfaz gráfica del cliente para que pueda desplegar los resultados del procesamiento.

Salvo por la inclusión del nuevo módulo para procesamiento y, tal vez, la modificación de la interfaz gráfica del cliente, ninguno de estos cambios es particularmente complejo. En general sólo se requeriría actualizar la implementación de los módulos involucrados, y en el caso de los módulos ComandosServidor y ComandosCliente tendría que estipularse el orden en que los parámetros para el procesamiento serían serializados.
Por otra parte, algunos de los requerimientos en la lista están relacionados con cuestiones de usabilidad. Éstas están acotadas casi exclusivamente al módulo de interfaz gráfica en el clien-

te pesado (GUICliente); cuestiones como el guardado de configuraciones locales, capturas de pantalla y la interacción con la información biomédica requieren efectuar modificaciones únicamente en dicho módulo.
La última funcionalidad de la lista está relacionada con la seguridad. En este ciclo de desarrollo no se implementó la funcionalidad para generar historiales de acceso al sistema. Esta responsabilidad podría asignarse a la superclase ServicioUsuario, de tal forma que cualquier servicio pueda registrar constantemente la actividad suscitada. Por otra parte, sin considerar la realización del caso de uso de Manejar Sesión, durante este ciclo de desarrollo se consideraron pocos conceptos de diseño para la seguridad, especialmente la de los datos. Una solución es la inclusión de mecanismos criptográficos dentro de los módulos de comunicación, como TLS o SSL. Por ejemplo, Qt ofrece implementaciones de estos protocolos que pueden ser fácilmente incluidas en el sistema. Otra situación se da con las propias imágenes y señales biomédicas, pues es frecuente que entre los metadatos haya información que sirva para identificar al paciente. En el caso de las imágenes, esta información podría incluso formar parte de los datos visibles (estos pueden ser etiquetas u "overlays" que se guardan como parte de la imagen por los sistemas de adquisición). En entornos académicos es deseable mantener el anonimato de pacientes y/o sujetos de experimentación, por lo que deben considerarse medidas para ocultar o eliminar esos datos que pudieran revelar sus identidades. Es factible que esta funcionalidad sea implementada dentro de las subclases en el módulo ModuloConversionDatosVolumetricos (ver Sección 4.3.4.2).
Por su parte, es evidente que el desempeño es bastante importante en sistemas como este, a pesar de que no figuró entre los atributos de calidad prioritarios para los interesados. Una visualización que requiere varios minutos, horas o días para llevarse a cabo es muy poco práctica para cualquier fin. La implementación llevada a cabo en este proyecto empleó la librería de VTK para efectuar el cómputo gráfico en el servidor, aunque no es la única opción viable para ello. Como se expuso en la Subsección 2.3.2, hay varios estudios publicados que emplean librerías que sacan partido al hardware gráfico disponible actualmente (GPU's), como es el caso de CUDA y OpenGL. La incorporación de implementaciones que empleen esas herramientas para efectuar los tipos de visualización presentados en este trabajo amerita evaluación.
El sistema puede ser adaptado para trabajar con información organizada por estándares como DICOM. DICOM son las siglas para *Digital Imaging and COmmunications in Medicine* y representa años de esfuerzo para crear el estándar más fundamental y universal en imagenología médica digital [80]. Las imágenes codificadas en este estándar pueden potencialmente ser procesadas por el sistema de visualización; todo lo que se requiere es agregar una subclase en el módulo ModuloConversionDatosVolumetricos que se encargue de convertir la información a otro formato conocido, como puede ser NifTI. O bien, el servidor también puede adaptarse para procesar imágenes DICOM, sin necesidad de convertirlas previamente. Todo lo que se requiere es cambiar el componente de tipo vtkNIFTIReader por uno de tipo vtkDICOMReader; efectivamente, VTK provee las herramientas para leer y manipular información DICOM. Considerando lo anterior futuros ciclos de desarrollo que enfoquen el diseño arquitectónico en otros atributos de calidad (disponibilidad, por ejemplo) podrían generar versiones del sistema para uso en entornos clínicos.

6.6. Protocolo de mensajes cliente-servidor

Los módulos ComandoCliente y ComandoServidor (ver Sección 4.3.2.4) son los encargados de garantizar que los mensajes entre cliente y servidor se adhieran al protocolo que permite un "entendimiento" entre ambos. Esto es, los mensajes que el cliente envía al servidor para solicitar servicios deben apegarse al protocolo para que aquél pueda entender la solicitud; si la solicitud es correcta y puede llevarse a cabo, entonces el mensaje de respuesta del servidor también deberá seguir el protocolo. Dependiendo del tipo de servicio solicitado, los mensajes pueden o no contener datos relacionados con información biomédica o con usuarios.
El protocolo de mensajes está diseñado para ser fácil de comprender y para permitir una comunicación simple y efectiva. No obstante, por tratarse de un protocolo no estándar, es imperativo mantener su documentación actualizada y organizada (parte de esta documentación se presentó en el Cuadro 4.14). En otras palabras, la inclusión de nueva funcionalidad o la modificación de la existente conllevarán cambios en los datos que clientes y servidor deben intercambiar; cada uno de esos cambios debe verse inmediatamente reflejado en la documentación del protocolo.
Si todo lo anterior es debidamente observado, entonces el protocolo de mensajes es extendible para realizar prácticamente cualquier tipo de funcionalidad. El servidor responderá adecuadamente mientras el protocolo sea respetado por el cliente, incluso sin importar cómo este último esté implementado.

6.7. Aportaciones del proyecto

Habiendo considerado la evidencia obtenida mediante un EMS, recolectado requerimientos de atributos de calidad mediante QAW y diseñado una arquitectura mediante ADD, se ha construido un sistema de visualización tridimensional que presenta varios atributos además de aquéllos normalmente reportados en la literatura. Si bien algunos de los estudios primarios revisados emplean patrones y/o tácticas arquitectónicas para satisfacer un atributo de calidad muy específico (particularmente desempeño y modificabilidad), ninguno contempla que sus diseños puedan ser adaptados a otros atributos. La arquitectura presentada en este trabajo es muy flexible, pues permite la incorporación de nueva funcionalidad y, potencialmente, la incorporación de herramientas que potencien aún más los atributos de calidad que se discutieron a lo largo de este capítulo. Todo esto sienta las bases para que el sistema pueda adaptarse a entornos distintos al académico; incluso es factible considerar la continuación de su desarrollo para ambientes clínicos.

Capítulo 7

Conclusiones

A lo largo de este trabajo se ha enfatizado la importancia que tiene el diseño arquitectónico para desarrollar sistemas de software complejos, especialmente aquellos que tratan con información biomédica. En ese sentido, la decisión de centrar el proceso de desarrollo en arquitectura está encaminada a otorgar al sistema los atributos de calidad necesarios para un sistema de visualización 3D de datos biomédicos. Se empleó un método de desarrollo arquitectónico precisamente guiado por atributos de calidad, y que se desempeñó como complemento fundamental de una metodología de desarrollo de software que guió, paso por paso, las actividades que llevaron a construir el sistema actual.

Los principales conceptos de diseño se escogieron con base en una revisión sistematizada de la literatura: el Estudio de Mapeo Sistemático (EMS). Su importancia principal radica en que permite bosquejar un panorama general del estado del arte, posibilitando así tomar decisiones justificadas respecto a herramientas o métodos. En el caso de proyectos como este, la ejecución del EMS debe actualizarse con frecuencia dada la rápida evolución de las herramientas y de los sistemas existentes que contemplan funcionalidad o atributos de calidad similares. El EMS presentado en este trabajo corroboró la necesidad de incorporar un diseño arquitectónico y proporcionó información sobre las herramientas requeridas para llevarlo a cabo. Varios de los conceptos de diseño propuestos por los estudios primarios fueron aplicados en este proyecto, sobre todo en lo que concierne a rendimiento y modificabilidad. Se encontraron muy pocos estudios que se abocaran a otros atributos de calidad (en algunos casos, como seguridad y disponibilidad, prácticamente sólo hubo un estudio, y de poco impacto), por lo que fue necesario considerar decisiones de diseño adicionales que complementaran a las reportadas en la literatura.

La recopilación de requerimientos de atributos de calidad se efectuó mediante el Taller de Atributos de Calidad (QAW). La organización del taller requiere minuciosa planeación, tanto en los tiempos como en el material para efectuarlo. De esta forma se incrementa la posibilidad de que participe un mayor número de los principales interesados en el sistema, cuyas ideas son las que determinarán el diseño arquitectónico.

Las directrices arquitectónicas halladas en el taller sirvieron como entrada al Diseño Dirigido por Atributos (ADD). Este método permitió efectuar de forma muy clara y organizada el diseño de la arquitectura de software, pues propone llevarlo a cabo partiendo desde un nivel muy general de abstracción hasta llegar al diseño detallado. Además, el diseño resultante cumplió con los requerimientos de atributos de calidad solicitados para el sistema.

El resultado de la ejecución de ADD fue una arquitectura que favorece principalmente a la modificabilidad. Un diseño de estilo cliente-servidor, el uso de interfaces y el diseño de módulos con alta coherencia semántica y con poco acoplamiento entre sí, fueron las decisiones de diseño empleadas para promover ese atributo. Sin embargo, varios de los conceptos de diseño incluidos también favorecen a la seguridad, al desempeño y a la usabilidad.
Se utilizaron las métricas estipuladas por el modelo de referencia RESVEP para determinar qué tanto la arquitectura cumple con atributos de calidad. El cálculo de estas métricas es relativamente sencillo si se cuenta con documentación arquitectónica completa y por lo menos una versión funcional del sistema. En lo que respecta al desarrollo del proyecto para fines académicos es factible seguir aplicando este modelo, ya que recurrir a una evaluación mediante métodos como ATAM es sumamente costoso.
Los valores obtenidos para los indicadores de calidad se tomaron como representativos de las fortalezas y debilidades de la arquitectura del sistema. Los indicadores para la mantenibilidad únicamente se vieron afectados por la ausencia de casos de pruebas, por lo que la recomendación es que se considere su ejecución en ciclos de desarrollo posteriores. La confiabilidad resultó ser un atributo de calidad relevante de acuerdo a los interesados en el sistema, aunque el escenario que más votos obtuvo resultó ser uno que se puede cumplirse con facilidad y sin mayores consideraciones en el diseño arquitectónico. Sin embargo, la baja calificación obtenida para este atributo se debe a que RESVEP contempla que el sistema debe llevar registros (historiales) de actividad. Este tipo de funcionalidad no figuró entre los requerimientos, aunque es preciso reconocer su importancia para detectar y sentar evidencia de sucesos que pudieran afectar al sistema. El atributo de seguridad también depende en buena medida de la habilidad del sistema para dejar constancia de lo que ocurre en él. Es por todo lo anterior que el requerimiento de generación de registros de actividad debe ser seriamente considerada para incluirse en el diseño arquitectónico.
VTK y Qt fueron las herramientas que impulsaron los indicadores de desempeño y usabilidad, respectivamente. VTK se empleó para realizar la mayor parte de las tareas de visualización, dado su uso extendido en la literatura y al buen rendimiento que en ella se reporta luego de emplearlo; los resultados obtenidos en el sistema respecto a ese atributo de calidad son satisfactorios siempre y cuando, en el caso de la representación conjunta, los conjuntos de imágenes estén debidamente segmentados. Por otra parte, el marco de trabajo de Qt se utilizó para la transferencia de datos cliente-servidor y, principalmente, para la implementación de la interfaz gráfica; los usuarios indicaron estar bastante conformes con la apariencia y facilidad de uso de dicha interfaz. Qt provee muchas otras herramientas que pueden ser incluidas en prácticamente cualquier proyecto de software, incluyendo pero no limitándose a: manejo de archivos, procesos multi-hebra (*multithreading*), manejo de información multimedia, etcétera. Esto hace a Qt una opción muy versátil para el trabajo a futuro de este proyecto.

7.1. Trabajo a futuro

Se cuenta actualmente con una primera versión funcional del sistema que cumple los requerimientos funcionales y no funcionales prioritarios; además, durante este primer ciclo de desarrollo se ha construido una arquitectura lo suficientemente flexible para permitir no sólo

la inclusión de nueva funcionalidad, sino también de otros atributos de calidad que propiciaría la incursión del sistema a entornos distintos al académico. Por lo tanto, la lista presentada a continuación presenta el trabajo que ha quedado pospuesto para futuros ciclos de desarrollo:

- Añadir funcionalidad para efectuar otros tipos de visualización y procesamiento 3D incluyendo, pero no limitándose a: cortes arbitrarios sobre las representaciones 3D, especificación de regiones de interés y generación de isosuperficies.
- Añadir módulos para procesamiento de imágenes y procesamiento de señales. Existen muchas herramientas disponibles actualmente que pueden reutilizarse para implementarlos.
- Incluir en el diseño actual conceptos que otorguen atributos de seguridad tanto para el sistema como para los datos. Aunque es probable que esto no sea fundamental en el entorno académico, sí es de enorme importancia para hacer que el sistema sea adecuado para operar en entornos clínicos.
- Análogamente a la cuestión de la seguridad, el atributo de confiabilidad es fundamental para los sistemas que operen en un entorno clínico. En futuros ciclos de desarrollo deberán considerarse tácticas para garantizar que el sistema podrá sobreponerse a situaciones de fallos inesperados. Algunas de estas tácticas pueden incluir, pero no se limitan a: mecanismos de redundancia activa y pasiva, monitores de actividad (por ejemplo, latidos o perro guardián).
- Complementar el diseño arquitectónico para que incluya componentes de cliente ligero y un servidor web. La funcionalidad ofrecida por éste último será, muy probablemente, más limitada respecto a la del servidor de aplicación, pero puede ser una opción muy viable para fines didácticos o para efectuar algunas tareas de visualización y procesamiento. Esto podría hacerse mediante VRML, pues es un lenguaje dedicado al despliegue de información tridimensional en clientes ligeros; el procesamiento tridimensional podría seguir siendo responsabilidad de componentes de VTK o ITK. Un buen ejemplo de un sistema de este tipo se presenta en el trabajo de Mahmoudi et al [52].

Apéndice A

Detalles del EMS

A.1. Sinónimos incluidos y conceptos excluidos en la identificación de estudios

La identificación de los sinónimos en inglés para cada concepto en las PIs es necesario para ensamblar las cadenas de búsqueda. Asimismo, deben contemplarse los conceptos a excluir explícitamente, es decir, aquéllos que no son de interés para la investigación pero que pueden estar accidentalmente incluidos en las búsquedas. Los sinónimos y los conceptos excluidos para el EMS efectuado en este proyecto se presentan en la siguiente el Cuadro A.1.

ID	Sinónimos
	* denota cualquier terminación de la palabra (wildcard).
PI 1	▪ **método**: method, procedure, algorithm, technique; **de visualización tridimensional**: three-dimensional visualization, 3D visualization, 3-D visualization, three-dimensional visualisation, 3D visualisation, 3-D visualisation, three-dimensional representation, 3D representation, 3-D representation, volume visualization, volume visualization, volume rendering, surface rendering. ▪ **neurociencias**: neuro* (neuroscience, neurology, neuroimaging, neurosurgery, ...), brain science", brain study.
PI 2	▪ **herramienta**: renderer, tool, toolkit, library, framework, software component. ▪ **desarrollo**: develop* (develop, development, developing, ...), implement* (implement, implementing, implementation), programming, design, construct* (construct, construction, constructing, ...); **de software**: software, application, system. ▪ **desarrollar**: ver **desarrollo** arriba; **sistemas**: ver **de software** arriba; **de visualización tridimensional**: ver PI 1; **para neurociencias**: ver PI 1.

Continúa...

PI 3	▪ **arquitectura de software**: software architecture, architecture model, architectural model, component model, structure model, architecture pattern, architectural pattern. ▪ **desarrollo**: ver PI 2; **de software**: ver PI 2. ▪ **desarrollar**: ver PI 2; **sistemas**: ver PI 2; **de visualización tridimensional**: ver PI 1; **en medicina**: medicine, "medical purpose", "medical application", "medical field", "medical environment", "clinical purpose", "clinical application", "clinical field", "clinical environment", healthcare, health care.

Conceptos excluidos
El proyecto está ideado para su aplicación exclusivamente en humanos, por lo que no es de interés aquéllo relacionado con ciencias que trabajen con animales. Además por el momento tampoco son relevantes las aplicaciones relacionadas con tecnología móvil. Por lo tanto se excluyen de la búsqueda los siguientes términos: ▪ veterinary, zoo* (zoo, zoology, zoological, ...), animal. ▪ mobile, phone, tablet, PDA.

Cuadro A.1: Sinónimos para los conceptos principales y conceptos excluidos.

A.2. Cadenas de búsqueda generales

Una vez identificados los conceptos principales y sus sinónimos se procede a ensamblar las expresiones de búsqueda, agrupando sinónimos con el operador lógico OR y conceptos principales con el operador AND; los conceptos excluidos se especifican mediante el operador NOT. Las cadenas se muestran en el Cuadro A.2.

ID	Cadena
PI 1	("three-dimensional visualization" OR "3D visualization" OR "3-D visualization" OR "volume visualization" OR "volumetric visualization" OR "three-dimensional visualisation" OR "3D visualisation" OR "3-D visualisation" OR "volume visualisation" OR "volumetric visualization" OR "three-dimensional representation" OR "3D representation" OR "3-D representation" OR "volume representation" OR "volumetric representation" OR "three-dimensional reconstruction" OR "3D reconstruction" OR "3-D reconstruction" OR "volume reconstruction" OR "volumetric reconstruction" OR "3D rendering" OR "3-D rendering" OR "volume rendering" OR "surface rendering") AND (method OR algorithm OR procedure OR technique) AND (neuro* OR "brain science" OR "brain study" OR "brain studies") NOT (veterinary OR zoo* OR animal OR mobile OR phone OR tablet OR PDA)

Continúa...

PI 2	(renderer OR tool OR toolkit OR library OR framework OR “software component”) AND (system OR application OR software) AND (develop* OR implement* OR construct* OR programming OR design) AND (“three-dimensional visualization” OR “3D visualization” OR “3-D visualization” OR “volume visualization” OR “volumetric visualization” OR “three-dimensional visualisation” OR “3D visualisation” OR “3-D visualisation” OR “volume visualisation” OR “volumetric visualization” OR “three-dimensional representation” OR “3D representation” OR “3-D representation” OR “volume representation” OR “volumetric representation” OR “three-dimensional reconstruction” OR “3D reconstruction” OR “3-D reconstruction” OR “volume reconstruction” OR “volumetric reconstruction” OR “3D rendering” OR “3-D rendering” OR “volume rendering” OR “surface rendering”) AND (neuro* OR “brain science” OR “brain study” OR “brain studies”) NOT (veterinary OR zoo* OR animal OR mobile OR phone OR tablet OR PDA)
PI 3	(“software architecture” OR “architecture model” OR “architectural model” OR “component model” OR “structure model” OR “architecture pattern” OR “architectural pattern”) AND (system OR application OR software) AND (develop* OR implement* OR construct* OR programming OR design) AND (“three-dimensional visualization” OR “3D visualization” OR “3-D visualization” OR “volume visualization” OR “volumetric visualization” OR “three-dimensional visualisation” OR “3D visualisation” OR “3-D visualisation” OR “volume visualisation” OR “volumetric visualization” OR “three-dimensional representation” OR “3D representation” OR “3-D representation” OR “volume representation” OR “volumetric representation” OR “three-dimensional reconstruction” OR “3D reconstruction” OR “3-D reconstruction” OR “volume reconstruction” OR “volumetric reconstruction” OR “3D rendering” OR “3-D rendering” OR “volume rendering” OR “surface rendering”) AND (medicine OR “medical purpose” OR “medical application” OR “medical field” OR “medical environment” OR “clinical purpose” OR “clinical application” OR “clinical field” OR “clinical environment” OR healthcare OR “health care”) NOT (veterinary OR zoo* OR animal OR mobile OR phone OR tablet OR PDA)

Cuadro A.2: Cadenas de búsqueda resultantes.

Apéndice B

OpenUP

A continuación se exponen sucintamente las actividades contenidas en las fases de OpenUP. Para propósitos de brevedad, no se listan las tareas marcadas por cada actividad pero sus propósitos se resumen a lo largo de cada descripción. Se recomienda al interesado consultar en línea la documentación del proceso [5] para obtener la lista y descripción completa de cada tarea.

- **Acordar estrategias técnicas**
  Básicamente esta actividad se enfoca en generar una propuesta inicial de arquitectura, de tal forma que se demuestre, tanto al equipo de trabajo como a los interesados, que hay una solución viable. La propuesta es a alto nivel, pues en esta etapa algunos de los requerimientos funcionales y no funcionales aún no estarán plenamente especificados.

- **Desarrollar la arquitectura**
  El objetivo de esta actividad es convertir el diseño a alto nivel que se tiene de la arquitectura en elementos de diseño concretos. Es muy probable que no toda la arquitectura se defina en una sola iteración, sino que vaya tomando en varias iteraciones con la prioridad de los requerimientos siempre en mente.

- **Desarrollar incremento en la solución**
  Buena parte del trabajo de diseño e implementación se llevará a cabo en esta actividad. Una parte del sistema es diseñada considerando requerimientos funcionales y/o no funcionales; dentro de ese contexto se implementa algún elemento de software, como puede ser una clase o componente. Para cada unidad de código se planean, ejecutan y registran pruebas para verificar que se comportan como se espera. Si no es el caso, se vuelve a la implementación y se efectúan nuevas pruebas hasta que sean satisfactorias; pueden haber cambios en el diseño de esa parte del sistema si lo aprendido con la implementación y pruebas sugieren su necesidad. Cuando las pruebas son exitosas, las porciones de código son integradas para corroborar su correcto comportamiento en conjunto. Todo esto se repite hasta que el elemento de software cumple satisfactoriamente con los requerimientos.

- **Poner la entrega en operación**
  Las tareas de esta actividad involucran la creación de un paquete de instalación para el sistema; éste debe estar completo y ser compatible con el entorno en el que el sistema

estará envuelto. Luego, debe determinarse si la instalación fue adecuada, y si no lo fue entonces tomar medidas para corregir los desperfectos en la misma. Una vez que se haya hecho todo correctamente, se anuncia la entrega del sistema a los usuarios y demás interesados.

- **Finalizar documentación del producto y material de entrenamiento**
  La documentación que se comenzó en la fase de construcción (manuales de usuario, manual de administrador, manual de desarrollador y material de entrenamiento, principalmente) debe ser concluida en la fase transición. No es infrecuente que el grueso del trabajo se posponga hasta estas alturas del ciclo de desarrollo.

- **Generar documentación del producto**
  Un sistema de calidad requiere estar debidamente documentado, y esta documentación debe estar dirigida a los usuarios e interesados en el sistema. Esta actividad suele ser de las que más se pasan por alto en el ciclo de desarrollo; es fundamental que los usuarios cuenten con información para gestionar el sistema del modo que más les convenga, y que apoye a obtener el máximo provecho de él. La documentación técnica dirigida al personal encargado del mantenimiento del sistema también debe ser considerada. Además, generar material de apoyo como tutoriales para los usuarios favorece mucho la usabilidad.

- **Identificar y refinar requerimientos**
  Actividad encaminada a la recopilación, análisis y validación de los requerimientos del sistema. Durante la fase de concepción lo que se busca con esta actividad es llegar a un consenso sobre lo que debe resolverse con el sistema, recabar las necesidades de los interesados y esclarecer los requisitos de alto nivel. Por su parte, en la elaboración deben identificarse los requerimientos que son más importantes para los interesados, así como los más riesgosos y los que impactarán a la arquitectura. Deben definirse en términos con los que los interesados estén de acuerdo y que todo el equipo de trabajo comprenda; luego, se proponen casos de prueba para cada uno. Finalmente, en la fase de construcción todo lo que queda por hacer es definir los requerimientos menos prioritarios para su subsecuente implementación.

- **Iniciar el proyecto**
  El objetivo de esta actividad es generar una visión general tanto del proyecto como del plan de trabajo. Se registran las necesidades que el sistema debe resolver, y se estipulan tiempos para efectuar las iteraciones en cada fase. Esta actividad se lleva a cabo en la primera iteración de la fase de concepción.

- **Planear y gestionar la iteración**
  A lo largo de cada fase el equipo de trabajo debe reunirse con cierta regularidad para reportar sus avances, discutir percances y plantear soluciones. Debe quedar perfectamente claro qué es lo que se quiere lograr al final de la iteración en curso, cuáles son los riesgos latentes y cómo van a mitigarse. De ser necesario se hacen ajustes al plan de trabajo, siempre con el objetivo de cumplir el objetivo de la iteración o, si el final de ésta coincide con el de la fase, de alcanzar el hito.

- **Prepararse para la entrega**
  Los criterios que el paquete de instalación debe cumplir deben quedar bien corroborados antes de ser liberado. De igual modo, todos los componentes necesarios para que el sistema sea adecuadamente liberado deben estar probados y listos. Si alguno de esos componentes no estuviera disponible, entonces se toman medidas para ponerlo a punto a la brevedad.

- **Probar la solución**
  Una vez que una parte del sistema (*build*) es operacional debe corroborarse que ésta cumple con los requerimientos que aborda. El responsable de las pruebas las ejecuta a nivel de sistema mientras, de forma paralela, los desarrolladores continúan implementando los elementos restantes. Si las pruebas son exitosas, entonces se prueba que se cuenta con una arquitectura robusta y que los requerimientos faltantes se implementarán sobre ella.

- **Tareas pendientes**
  Aunque los requerimientos eventualmente estarán bien definidos y estables, siempre es factible que cambien debido a peticiones de los interesados o por hallazgos de los responsables de pruebas, principalmente. Por lo tanto, es una actividad para la que el equipo debe estar preparado.

Apéndice C

Indicadores de calidad

Los Cuadros C.1 y C.2 enlistan los indicadores de calidad utilizados en el modelo de referencia RESVEP, así como los parámetros empleados para calcularlos. Ambos cuadros fueron extraídos de la tesis de maestría de Óscar Castro López [74], la cual cubre a detalle el modelo mencionado.

Cuadro C.1: Value Quality Indicators definition, measurements and equation for the work products of the *Design Discipline*

Value Quality Indicator	**Measurement and Equation**
x1: Traceability to the requirements	$x1 = \frac{A}{B}$ x1 = Traceability to the requirements. A = Number of traceable design items confirmed in review. B = Number of items checked. Interpretation of the result: 0 <= $x1$ <= 1 The closer to 1, the better.
S: Security	Equation of security: $S = \frac{s1+s2}{2}$
s1: Confidentiality	$s1 = 1 - \frac{A}{B}$ s1 = Confidentiality. A = Number of components that are not secure from having unauthorized disclosure of data or information, whether accidental or deliberate. B = Total number of components that handles data evaluated. Interpretation of the result: 0 <= $s1$ <= 1 The closer to 1, the better.
s2: Integrity	$s2 = \frac{A}{B}$ s2 = Integrity. A = Number of components that prevents unauthorized access to, or modification. B = Total number of components that handles data evaluated. Interpretation of the result: 0 <= $s2$ <= 1 The closer to 1, the better.
M: Maintainability	Equation of Maintainability: $M = \frac{m1+m2+m3+m4+m5}{5}$
m1: Modularity.	$m1 = 1 - \frac{A}{B}$ m1 = Modularity. A = Number of components that have high impact when they are modified. B = Total number of components. Interpretation of the result: 0 <= $m1$ <= 1 The closer to 1, the better.
m2: Reusability.	$m2 = \frac{A}{B}$ m2 = Reusability. A = Number of components that are reusable. B = Total number of components. Interpretation of the result: 0 <= $m2$ <= 1 The closer to 1, the better.

Continued on Next Page...

Cuadro C.1 – Continued

Value Quality Indicator	**Measurement and Equation**
m3: Analyzability.	$m3 = \frac{A}{B}$ m3 = Analyzability. A = Number of components that can be easy diagnosed for deficiencies or causes of failures in the component model. (Helped by comments, version, code standards, etc.) B = Total number of components. Interpretation of the result: 0 <= $m3$ <= 1 The closer to 1, the better.
m4: Modifiability.	$m4 = \frac{A}{B}$ m4 = Modifiability. A = Number of components that can be modified without introducing defects or degrading performance. B = Total number of components. Interpretation of the result: 0 <= $m4$ <= 1 The closer to 1, the better.
m5: Testability.	$m5 = \frac{A}{B}$ m5 = Testability. A = Number of cases in which a component can be tested appropriately. B = Number of cases of component tests. Interpretation of the result: 0 <= $m5$ <= 1 The closer to 1, the better.
P: Portability	Equation of portability: $P = \frac{p1}{1}$
p1: Replaceability	$p1 = \frac{A}{B}$ p1 = Replaceability. A = Number of components that can replace another for the same purpose in the same environment. B = Total number of components in the component model. Interpretation of the result: 0 <= $p1$ <= 1 The closer to 1, the better.
FS: Functional suitability	Equation of functional suitability: $FS = \frac{fs1+fs2}{2}$
fs1: Functional appropriateness	$fs1 = 1 - \frac{A}{B}$ fs1 = Functional appropriateness. A = Number of missing components or with errors detected in design evaluation. B = Number of components described in the requirements. Interpretation of the result: 0 <= $fs1$ <= 1 The closer to 1, the better.
fs2: Accuracy	$fs2 = 1 - \frac{A}{B}$ fs2 = Accuracy. A = Number of inconsistent components detected in design evaluation. B = Number of components described in the requirements. Interpretation of the result: 0 <= $fs2$ <= 1 The closer to 1, the better.
U: Usability	Equation of usability: $U = \frac{u1+u2}{2}$
u1: Learnability	$u1 = \frac{A}{B}$ u1 = Learnability. A = Number of components whose purpose is correctly described in the design. B = Number of components evaluated. Interpretation of the result: 0 <= $u1$ <= N If equal to 1 or higher, the better.
u2: Ease of use	$u2 = \frac{t1}{t2}$ u2 = Ease of use. t1 = Mean real-time taken to learn a component application correctly. t2 = Mean expected-time taken to learn a component application correctly. Interpretation of the result: 0 <= $u2$ <= N If equal to 1 or higher, the better. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.

Cuadro C.2: Value Quality Indicators definition, measure and equation for the work products of the *Construction Process*

Value Quality Indicator	**Measurement and Equation**
U: Usability	Equation of usability: $U = \frac{u1+u2+u3+u4}{4}$

Continued on Next Page...

Cuadro C.2 – Continued

Value Quality Indicator	**Measurement and Equation**
u1: Learnability	$u1 = \frac{t1}{t2}$ u1 = Learnability. t1 = Mean expected-time taken to learn a software unit correctly. t2 = Mean real-time taken to learn a software unit correctly. Interpretation of the result: $0 <= u1 <= N$ If equal to 1 or higher, the better. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.
u2: Ease of use	$u2 = \frac{t1}{t2}$ u2 = Ease of use. t1 = Mean expected-time taken to use a software unit correctly. t2 = Mean real-time taken to use a software unit correctly. Interpretation of the result: $0 <= u2 <= N$ If equal to 1 or higher, the better. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.
u3: User error protection	$u3 = \frac{A}{B}$ u3 = User error protection. A = Number of user-interface units that protects users against making errors. B = Number of user-interface units evaluated. Interpretation of the result: $0 <= u3 <= 1$ The closer to 1, the better.
u4: User interface aesthetics	$u4 = \frac{\frac{A_1+A_2+A_3+\ldots+A_N}{B}}{10}$ u4 = User interface aesthetics. A_i = Rating assigned to the user interface (0 to 10) according to user satisfaction criteria. B = Number of user-interface units evaluated. Interpretation of the result: $0 <= u4 <= 1$ The closer to 1, the better.
PE: Performance efficiency	Equation of performance efficiency: $PE = \frac{pe1+pe2}{2}$
pe1: Time behavior	$pe1 = \frac{t1}{t2}$ pe1 = Time behavior. t1 = Time of response expected. t2 = Time of real response. Interpretation of the result: $0 <= pe1 <= N$ If equal to 1 or higher, the better. If more than one unit of software is evaluated, there will be a summation of t1 of each unit, and this also applies to t2.
pe2: Resource utilization	$pe2 = \frac{A}{B}$ pe2 = Resource utilization. t1 = Expected memory requirement for the software unit. t2 = Real memory requirement for the software unit. Interpretation of the result: $0 <= pe2 <= 1$ If equal to 1 or higher, the better.
C: Compatibility	Equation of compatibility: $C = \frac{c1+c2}{2}$
c1: Co-existence	$c1 = 1 - \frac{A}{B}$ c1 = Co-existence. A = Number of functions that cannot co-exist with other independent software in a common environment sharing common resources without any detrimental impacts. B = Number of functions tested. Interpretation of the result: $0 <= c1 <= 1$ The closer to 1, the better.
c2: Interoperability	$c2 = \frac{A}{B}$ c2 = Interoperability. A = Number of functions that can exchange information and use the information that has been exchanged. B = Number of functions evaluated. Interpretation of the result: $0 <= c2 <= 1$ The closer to 1, the better.
FS: Functional suitability	Equation of functional suitability: $FS = \frac{fs1+fs2}{2}$
fs1: Functional appropriateness	$fs1 = 1 - \frac{A}{B}$ fs1 = Functional appropriateness. A = Number of missing functions or with errors detected in evaluation. B = Number of functions described in design. Interpretation of the result: $0 <= fs1 <= 1$ The closer to 1, the better.

Continued on Next Page...

Cuadro C.2 – Continued

Value Quality Indicator	**Measurement and Equation**
fs2: Accuracy	$fs2 = 1 - \frac{A}{B}$ fs2 = Accuracy. A = Number of inconsistent functions detected in evaluation. B = Number of functions described in design. Interpretation of the result: 0 <= $fs2$ <= 1 The closer to 1, the better.
S: Security	Equation of security: $S = \frac{s1+s2+s3+s4+s5}{5}$
s1: Confidentiality	$s1 = 1 - \frac{A}{B}$ s1 = Confidentiality. A = Number of functions that are not secure from having unauthorized disclosure of data or information, whether accidental or deliberate. B = Total number of functions that handles data evaluated. Interpretation of the result: 0 <= $s1$ <= 1 The closer to 1, the better.
s2: Integrity	$s2 = \frac{A}{B}$ s2 = Integrity. A = Number of functions that prevents unauthorized access to, or modification. B = Total number of functions that handles data evaluated. Interpretation of the result: 0 <= $s2$ <= 1 The closer to 1, the better.
s3: Non-repudiation	$s3 = \frac{A}{B}$ s3 = Non-repudiation. A = Number of functions log file of their events so they can be proven to have taken place. B = Total of functions that need a log file. Interpretation of the result: 0 <= $s3$ <= 1 The closer to 1, the better.
s4: Accountability	$s4 = \frac{A}{B}$ s4 = Accountability. A = Number of functions that their actions can be traced uniquely to them in the log file. B = Total number of functions that need a log file to trace actions. Interpretation of the result: 0 <= $s4$ <= 1 The closer to 1, the better.
s5: Authenticity	$s5 = \frac{A}{B}$ s5 = Authenticity. A = Resource or subject identification (Real percentage of veracity for assuring this identification). B = Resource or subject identification (Expected percentage of veracity for assuring this identification). Interpretation of the result: 0 <= $s5$ <= 1 The closer to 1, the better.
R: Reliability	Equation of reliability: $R = \frac{r1+r2+r3+r4}{4}$
r1: Maturity	$r1 = \frac{A}{B}$ r1 = Maturity. A = Desired system failures (reliability needs) in T. B = Current system failures in T. T = Time period (Time of the maturity test). Interpretation of the result: 0 <= $r1$ <= 1 The closer to 1, the better. Special cases when a variable is equal to 0: 1. If $A = 0$ and $B = 0$, then $r1 = 1$. 2. If $A = 0$ and $B > 0$, then $r1 = 0$. 3. If $A > 0$ and $B = 0$, then $r1 = 1$.
r2: Availability	$r2 = \frac{t1}{t2}$ r2 = Availability. t1 = Time of the component being in an “up state” (operational and accessible when required). t2 = Time period (Time of the availability test). Interpretation of the result: 0 <= $r1$ <= 1 The closer to 1, the better.

Continued on Next Page...

Cuadro C.2 – Continued

Value Quality Indicator	**Measurement and Equation**
r3: Fault tolerance	$r3 = \frac{A}{B}$ r3 = Fault tolerance. A = Number of functions that operate as intended despite the presence of hardware or software faults. B = Number of functions evaluated. Interpretation of the result: 0 <= $r1$ <= 1 The closer to 1, the better.
r4: Recoverability	$r4 = \frac{A}{B}$ r4 = Recoverability. A = Number of functions that can recover data directly affected and re-establish the desired state of the system in the case of an interruption or a failure. B = Total number of functions that handles data to be evaluated. Interpretation of the result: 0 <= $r1$ <= 1 The closer to 1, the better.
X1: Traceability to the requirements	$x1 = \frac{A}{B}$ x1 = Traceability to the requirements. A = Number of traceable software units confirmed in review. B = Number of items checked. Interpretation of the result: 0 <= $x1$ <= 1 The closer to 1, the better.
P: Portability	Equation of portability: $P = \frac{p1+p2+p3}{3}$
p1: Adaptability	$p1 = \frac{Hi+Si}{2}$ p1 = Adaptability. 1. Hi = Hardware independence $Hi = 1 - \frac{A}{B}$ A = Number of functions of which tasks were not completed or not enough resulted to meet adequate levels during combined operating testing with environmental hardware. B = Total number of functions evaluated. Interpretation of the result: 0 <= Hi <= 1 The closer to 1, the better. 2. Si = Software independence $Si = 1 - \frac{A}{B}$ A = Number of functions of which tasks were not completed or were not enough resulted to meet adequate level during combined operating testing with operating system software or concurrent application software. B = Total number of functions evaluated. Interpretation of the result: 0 <= Si <= 1 The closer to 1, the better.
p2: Replaceability	$p2 = \frac{A}{B}$ p2 = Replaceability. A = Number of functions that can replace another for the same purpose in the same environment. B = Total number of functions evaluated. Interpretation of the result: 0 <= $p2$ <= 1 The closer to 1, the better.
p3: Installability	$p3 = \frac{A}{B}$ p3 = Installability. A = Number of environments where each function can be successfully installed and/or uninstalled. B = Number of environments specified in the quality scenarios. Interpretation of the result: 0 <= $p3$ <= 1 The closer to 1, the better.

Apéndice D

Visualización 3D con VTK

VTK emplea un modelo que emula a una escena de cine; cada representación requiere de un actor con "vestuario y maquillaje", cámaras, luces, un escenario sobre el que desempeñarse y un director que organice la escena. El resultado final será que el actor estará correctamente visualizado en pantalla.
Son muchos los tipos de objetos que VTK emplea para presentar imágenes en un monitor, pero los principales son los siguientes:

- vtkProperty
- vtkActor
- vtkCamera
- vtkLight
- vtkMapper
- vtkRenderer
- vtkRenderWindow

La Figura D.1 expone cómo se ven en pantalla estos componentes cuando están en ejecución.

D.1. Lectura y procesamiento de datos

Los actores en VTK son la representación visual de algún conjunto de datos. El tipo y la estructura de esos datos depende de muchos factores, pero la forma de generarlos y procesarlos es, en general, la misma.
VTK define dos tipos de componentes para trabajar con los datos que se emplean para crear a los actores: componentes de datos y componentes de procesamiento [67].

- Los *componentes de datos* representan a algún tipo de información, y proveen métodos para crear, leer y eliminar dicha información; su modificación directa, por otra parte, no está permitida en estos componentes.

- Los *componentes de procesamiento* son los que generan o transforman la información de los componentes de datos basándose en una serie de parámetros específicos. Se subdividen en tres tipos de acuerdo a las operaciones que efectúan: lectura, filtro y mapeo:

    - Los *componentes de lectura* generan nuevos componentes de datos a partir de alguna fuente (por ejemplo, archivos de imágenes).

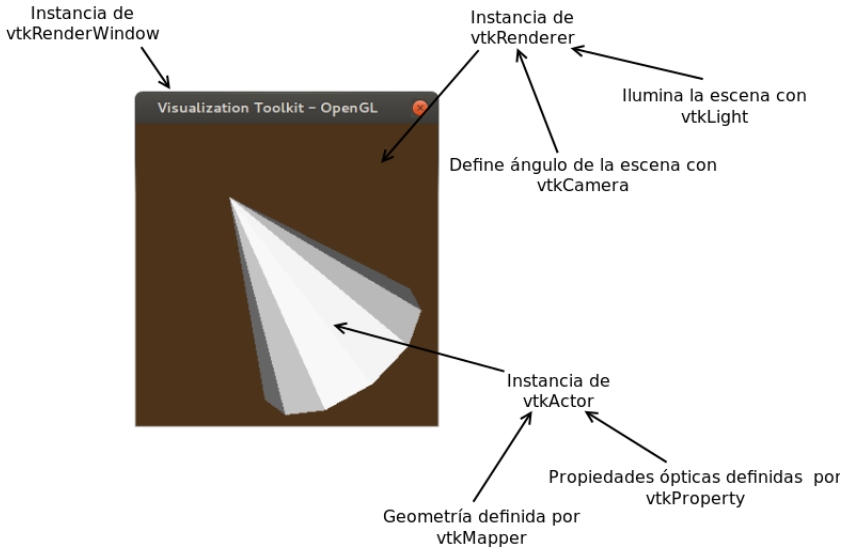


Figura D.1: Apariencia visual de los componentes de VTK en tiempo de ejecución. Las flechas negras apuntan hacia la parte de la ventana en donde se despliega el respectivo componente, o bien hacia un componente que depende de los otros en la manera descrita.

- Los *filtros* transforman la información que reciben y con ella crean uno o varios nuevos componentes de datos.
- Los *componentes de mapeo* son los encargados de generar la representación final de los datos, ya sea polígonos en la pantalla, un archivo de imagen, datos en un archivo de texto, etcétera.

De este modo, VTK estipula un patrón de diseño de red de flujo de datos (comúnmente conocida como *pipe-and-filter*) en la cual los componentes de procesamiento pueden ser añadidos e intercambiados a conveniencia, siempre y cuando el tipo de componente de datos entre la salida y la entrada sean los correctos. La Figura D.2 ilustra este patrón de forma muy general; consiste en varios componentes de procesamiento comenzando con uno de lectura al que pueden seguir uno o más filtros para, finalmente, concluir con un componente de mapeo.

D.2. Descomposición del componente Representacion3D

Actualmente se han implementado dos tipos de representación tridimensional para este proyecto, y son representación por volumen y representación conjunta (ver la Sección 2.3.1.1). El componente Representacion3D es el encargado de efectuar esos tipos de representación en el servidor durante el tiempo de ejecución; sus principales características se enlistaron en las vistas de componentes de la Sección 4.3.2.1, mientras que sus relaciones con otros componentes se ilustraron en la Figura 4.13. Dependiendo del tipo de representación tridimensional efectuada, los sub-componentes contenidos en Representacion3D variarán de acuerdo a lo expuesto en los párrafos siguientes.

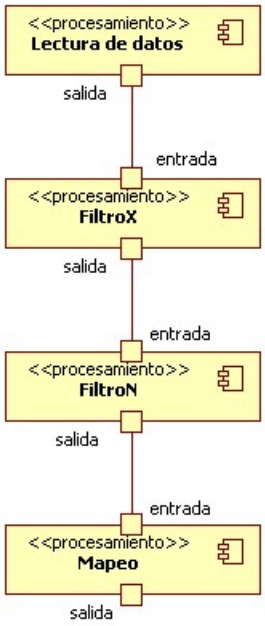


Figura D.2: Red de flujo de datos para la lectura y procesamiento de información en VTK.

D.2.1. Representación anatómica

La secuencia de componentes de procesamiento que se requiere para efectuar la representación por volumen se presenta en la Figura D.3. Éstos componentes son instancias de clases de VTK que se relacionan entre ellas mediante dependencias simples. La red de flujo de datos para realizar una representación por ray-casting, ilustrada dentro de un rectángulo en la figura, es relativamente sencilla; únicamente requiere de tres componentes de procesamiento. No obstante, las propiedades del volumen, contenidas en el componente de tipo vtkVolumeProperty, requiere que previamente se especifiquen algunos parámetros a cargo de otros componentes que no se presentan en el diagrama. El tipo de estos componentes y sus respectivas responsabilidades son los siguientes:

- *vtkColorTransferFunction*: se requiere un componente de este tipo para representar la función de transferencia de color para los "rayos" del ray-casting. Lo usual es que ésta sea una función identidad con respecto a la intensidad de los voxeles, aunque puede especificarse a conveniencia.
- *vtkPiecewiseFunction*: se emplean dos componentes de este tipo. Uno para modelar la función de transferencia para la opacidad, y otro para modelar el gradiente de la misma con respecto a la intensidad de los voxeles.

Finalmente, la Figura D.3 también muestra los componentes de VTK necesarios para generar la escena en pantalla (esto es, el actor con sus propiedades ópticas y geométricas, el renderer y ventana), pero con una única diferencia: el filtro de tipo vtkWindowToImageFilter toma como entrada la información del componente vtkRenderWindow, que es el que despliega la ventana en pantalla. Ese filtro se encarga de generar una captura de pantalla (screenshot) a partir de la información deplegada por vtkRenderWindow. Básicamente, vtkWindowToImageFilter es capaz de serializar la captura de pantalla para que ésta sea guardada en el

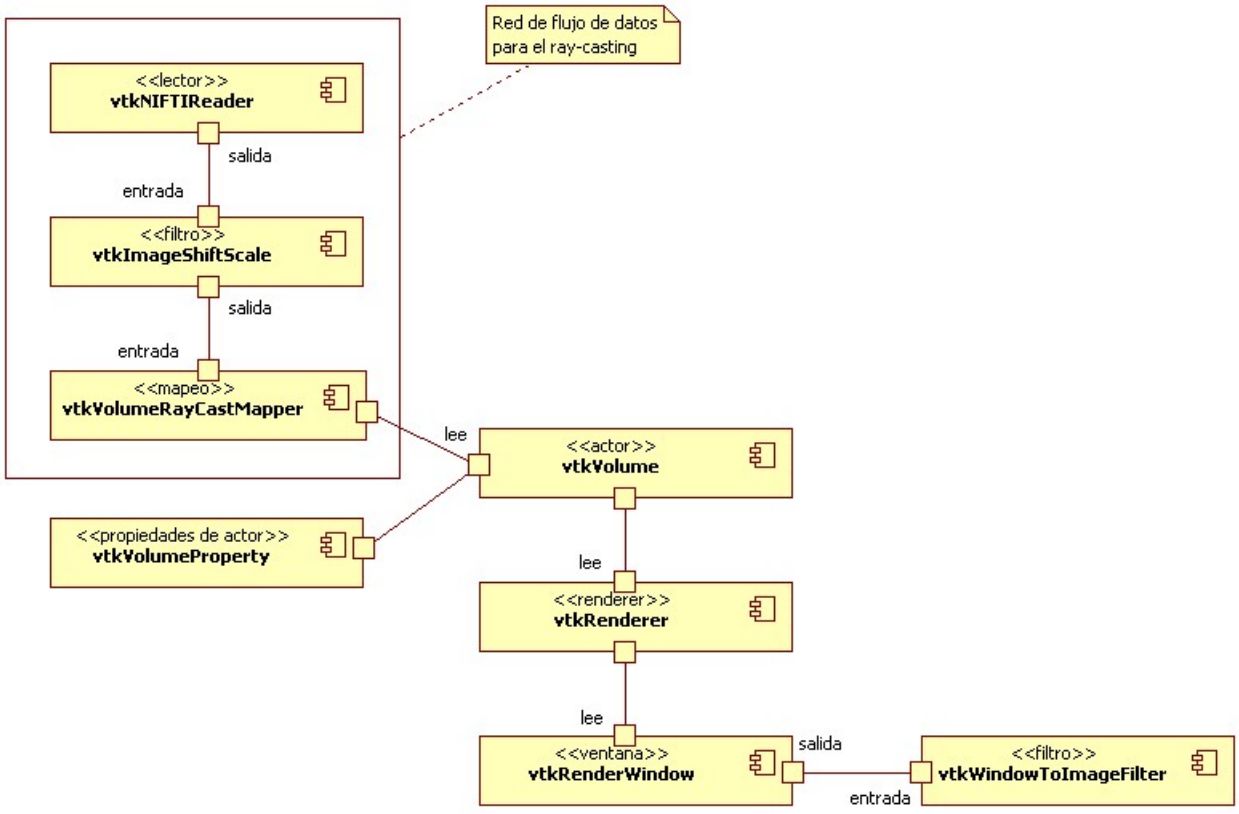


Figura D.3: Descomposición del componente Representacion3D para la representación anatómica.

buffer *datosProyeccion2D* (ver Figura 4.17); este buffer es finalmente devuelto por el método *efectuarRepresentacion3D*().

D.2.2. Representación conjunta

Dada la manera en que se ha implementado en este proyecto, la representación conjunta requiere de dos tipos de representación básicos: representación por volumen y representación por superficie. Básicamente, la idea es emplear la primera para visualizar la información funcional, y la última para sobreponer una "silueta" de la información anatómica que sirva como referencia para posicionar la funcional; esta es una estrategia similar a la empleada en el estudio presentado por Janeik et al. [27]. La Figura D.4 muestra la interacción entre componentes de VTK que se emplearon para la representación conjunta. En la figura se exponen dos redes de flujos de datos. La del rectángulo de la izquierda corresponde al de la representación por superficie, y el más pequeño de la derecha al de la representación por volumen. Este último corresponde a la misma red de la representación por superficie ilustrada la Figura D.3, salvo por la omisión del componente vtkImageShiftScale, dado que éste se emplea para convertir el tipo de datos inicial a entero sin signo; esto no es deseable al tratar con información funcional, pues podría conllevar una modificación inadecuada de

la misma. Por su parte, el rectángulo de la izquierda presenta la red de flujo de datos para la representación por superficie. Las responsabilidades de cada componente para este tipo de representación son las siguientes:

- *vtkImageIslandRemoval2D*: reclasificación de conjuntos de un número determinado de pixeles. Estos conjuntos se considerarán como pertenecientes a la estructura anatómica utilizada para generar la malla de polígonos (triángulos en este caso).
- *vtkImageThreshold*: umbralización de intensidades de voxeles. Valores superiores o iguales al umbral se considerarán dentro de la estructura para generar la malla poligonal; valores inferiores se tomarán como fondo.
- *vtkContourFilter*: filtro generador de isosuperficies. Se espera que los datos de entrada a este filtro correspondan a una estructura superficial segmentada de la cabeza (piel o hueso preferentemente) y registrada con la información funcional. Emplear otro tipo de datos, datos no segmentados y/o no registrados no ocasionará errores, pero los resultados visuales no serán idóneos y los tiempos de procesamiento podrían verse seriamente afectados.
- *vtkDecimatePro*: filtro para reducir el número de polígonos empleados en la malla. El porcentaje de reducción es configurable.
- *vtkSmoothPolyDataFilter*: filtro para suavizado de mallas poligonales. Opera en vértices y líneas para brindar una apariencia más "relajada". Mayor número de iteraciones del algoritmo de suavizado generan mallas más suaves a costa de un mayor tiempo de procesamiento.
- *vtkPolyDataNormals*: filtro para cómputo de vectores normales en las celdas de la malla poligonal.
- *vtkStripper*: este filtro genera una representación más compacta de un conjunto grande de triángulos, al describirlos únicamente mediante una serie de puntos en el espacio tridimensional.

Las redes de flujo de datos para cada tipo de representación generarán a su actores correspondientes, con propiedades y mapeo independientes. Una vez definidos los actores, estos son pasados a un componente de tipo vtkRenderer para que generar la escena. Finalmente, los componentes vtkRenderWindow y vtkWindowToImageFilter se emplean para generar la serialización de la proyección 2D que será retornada por el método *efectuarRepresentacion3D*().

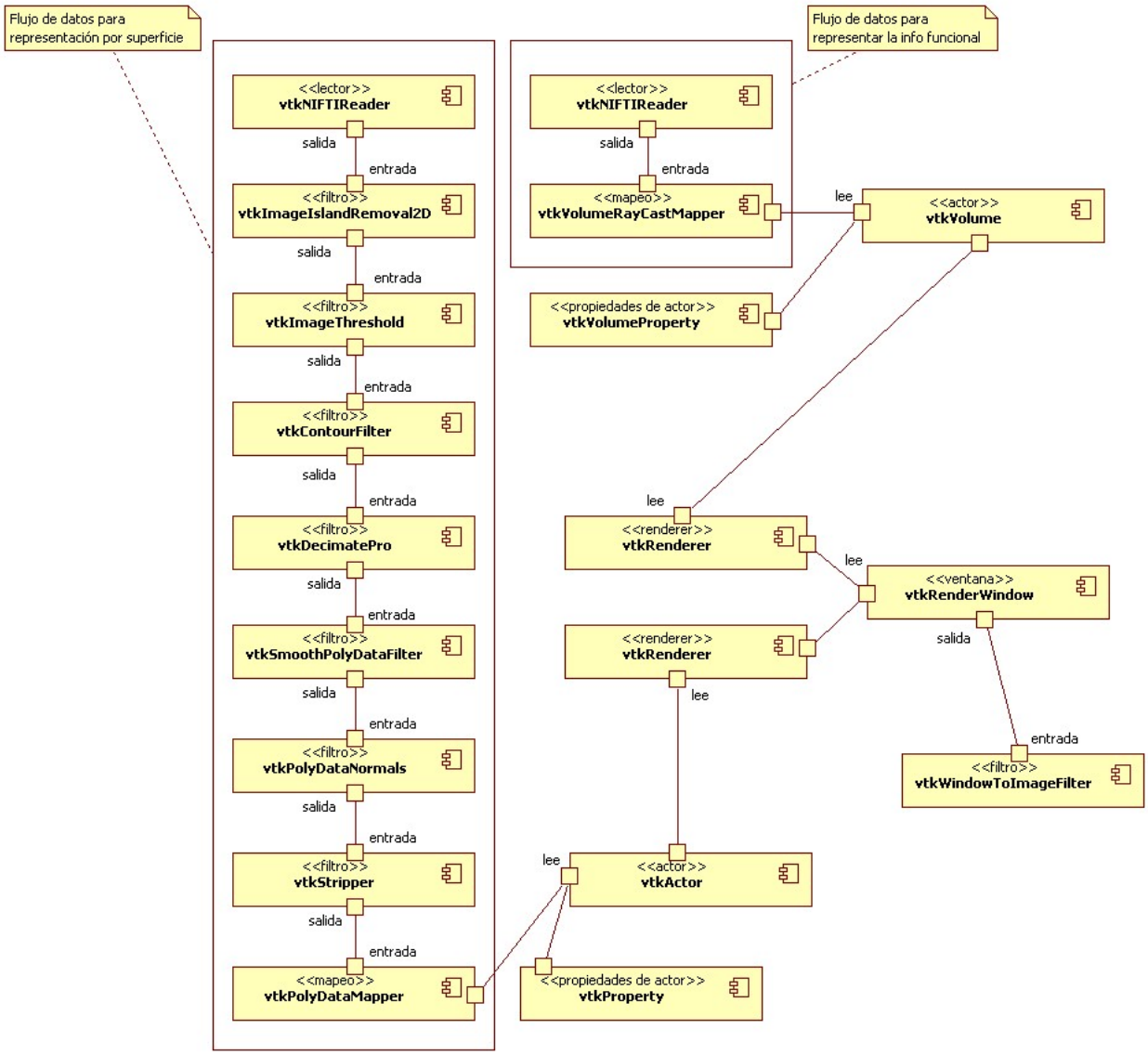


Figura D.4: Descomposición del componente Representacion3D para la representación conjunta.

Apéndice E

Puesta en operación

E.1. Archivos requeridos en el nodo del servidor

Los archivos requeridos para ejecutar los componentes en el nodo del servidor se empacaron en un archivo comprimido denominado "3dv-serv-linux64.tar.gz". Éste contiene tres tipos de archivo: un archivo binario ejecutable, un script de Shell para inicializar al servidor como demonio informático, y varias bibliotecas dinámicas.
Las bibliotecas dinámicas que contienen la funcionalidad relacionada con visualización corresponden a la versión 6.0.0 de VTK. Los archivos que corresponden a dichas bibliotecas son los siguientes:

- libvtkalglib-6.0.so.1
- libvtkCommonComputationalGeometry-6.0.so.1
- libvtkCommonCore-6.0.so.1
- libvtkCommonDataModel-6.0.so.1
- libvtkCommonExecutionModel-6.0.so.1
- libvtkCommonMath-6.0.so.1
- libvtkCommonMisc-6.0.so.1
- libvtkCommonSystem-6.0.so.1
- libvtkCommonTransforms-6.0.so.1
- libvtkDICOM-6.0.so.0.7 [1]
- libvtkDICOMParser-6.0.so.1
- libvtkexpat-6.0.so.1
- libvtkFiltersCore-6.0.so.1
- libvtkFiltersExtraction-6.0.so.1
- libvtkFiltersGeneral-6.0.so.1
- libvtkFiltersGeometry-6.0.so.1
- libvtkFiltersSources-6.0.so.1
- libvtkFiltersStatistics-6.0.so.1
- libvtkfreetype-6.0.so.1
- libvtkftgl-6.0.so.1
- libvtkGUISupportQt-6.0.so.1
- libvtkImagingCore-6.0.so.1
- libvtkImagingFourier-6.0.so.1
- libvtkImagingGeneral-6.0.so.1
- libvtkImagingHybrid-6.0.so.1
- libvtkImagingMorphological-6.0.so.1
- libvtkImagingSources-6.0.so.1
- libvtkInteractionStyle-6.0.so.1
- libvtkIOCore-6.0.so.1
- libvtkIOImage-6.0.so.1
- libvtkIOSQL-6.0.so.1
- libvtkIOXMLParser-6.0.so.1
- libvtkjpeg-6.0.so.1
- libvtkmetaio-6.0.so.1
- libvtkpng-6.0.so.1
- libvtkRenderingCore-6.0.so.1
- libvtkRenderingFreeType-6.0.so.1
- libvtkRenderingFreeTypeOpenGL-6.0.so.1
- libvtkRenderingOpenGL-6.0.so.1
- libvtkRenderingVolume-6.0.so.1
- libvtkRenderingVolumeOpenGL-6.0.so.1
- libvtksqlite-6.0.so.1
- libvtksys-6.0.so.1
- libvtktiff-6.0.so.1
- libvtkzlib-6.0.so.1

[1] Esta librería no forma parte de la distribución oficial de VTK. El crédito por la misma corresponde a David Gobbi, y el código fuente puede ser obtenido en la siguiente dirección: https://github.com/dgobbi/vtk-dicom/

Por otra parte, la funcionalidad relacionada con la comunicación remota, serialización y deserialización de datos, y la gestión de directorios y archivos es otorgada por las bibliotecas de Qt versión 4.7.4. Los archivos que contienen dicha funcionalidad son los siguientes:

- libQtCore.so.4.7.4
- libQtNetwork.so.4.7.4

Finalmente, el archivo "3dv-serv-linux64.tar.gz" contiene otros dos archivos:

- 3DV-serv
- 3dvserv

"3DV-serv" es el archivo ejecutable, mientras que "3dvserv" es otro script de bash empleado para correr a dicho archivo ejecutable como un demonio informático (en la jerga computacional en inglés, a estos archivos se les conoce como *daemons*. Son aplicaciones que se ejecutan discretamente en segundo plano del sistema cada vez que éste arranca. La ejecución del archivo se mantiene de manera sileciosa y constante a la espera de que ocurra algún evento [81], que en este caso es la conexión de los clientes al servidor).

E.2. Archivos requeridos en los nodos cliente

Los archivos requeridos para ejecutar los componentes en nodos cliente con sistemas operativos basados en GNU/Linux se empacaron en los archivos comprimidos "3dv-cliente-linux64.tar.gz" y "3dv-cliente-linux32.tar.gz". El primero contiene los archivos para sistemas de 64 bits, y el segundo para sistemas de 32 bits; ambos contienen a todas las librerías dinámicas de VTK y Qt indispensables, y al archivo binario ejecutable.
Las librerías de VTK empleadas por los clientes en distribuciones de GNU/Linux son las siguientes:

- libvtkalglib-6.0.so.1
- libvtkCommonComputationalGeometry-6.0.so.1
- libvtkCommonCore-6.0.so.1
- libvtkCommonDataModel-6.0.so.1
- libvtkCommonExecutionModel-6.0.so.1
- libvtkCommonMath-6.0.so.1
- libvtkCommonMisc-6.0.so.1
- libvtkCommonSystem-6.0.so.1
- libvtkCommonTransforms-6.0.so.1
- libvtkDICOM-6.0.so.0.7 [2]
- libvtkDICOMParser-6.0.so.1
- libvtkexpat-6.0.so.1
- libvtkFiltersCore-6.0.so.1
- libvtkFiltersExtraction-6.0.so.1
- libvtkFiltersGeneral-6.0.so.1
- libvtkFiltersGeometry-6.0.so.1
- libvtkFiltersSources-6.0.so.1
- libvtkFiltersStatistics-6.0.so.1
- libvtkfreetype-6.0.so.1
- libvtkftgl-6.0.so.1
- libvtkGUISupportQt-6.0.so.1
- libvtkImagingCore-6.0.so.1
- libvtkImagingFourier-6.0.so.1
- libvtkImagingHybrid-6.0.so.1
- libvtkInteractionStyle-6.0.so.1
- libvtkIOCore-6.0.so.1
- libvtkIOImage-6.0.so.1
- libvtkIOSQL-6.0.so.1
- libvtkIOXMLParser-6.0.so.1
- libvtkjpeg-6.0.so.1
- libvtkmetaio-6.0.so.1
- libvtkpng-6.0.so.1
- libvtkRenderingCore-6.0.so.1
- libvtkRenderingFreeType-6.0.so.1
- libvtkRenderingFreeTypeOpenGL-6.0.so.1

[2] Esta librería no forma parte de la distribución oficial de VTK. El crédito por la misma corresponde a David Gobbi, y el código fuente puede ser obtenido en la siguiente dirección: https://github.com/dgobbi/vtk-dicom/

- libvtkRenderingOpenGL-6.0.so.1
- libvtkRenderingVolume-6.0.so.1
- libvtkRenderingVolumeOpenGL-6.0.so.1
- libvtksqlite-6.0.so.1
- libvtksys-6.0.so.1
- libvtktiff-6.0.so.1
- libvtkzlib-6.0.so.1

En lo que respecta a Qt, las librerías empleadas por los clientes son:

- libQtCore.so.4.7.4
- libQtGui.so.4.7.4
- libQtNetwork.so.4.7.4

Finalmente, el archivo comprimido contiene al archivo ejecutable denominado "3DV-cliente". A diferencia del servidor, el cliente no requiere un script de Shell que corra al archivo ejecutable como un demonio informático.
En lo que concierne a la versión del cliente para el sistema operativo Windows, ésta ha sido probada en un entorno de desarrollo con todas las librerías de Qt y VTK previamente instaladas. No obstante, aún está pendiente proponer una forma versátil de empacar e instalar esos archivos para que sean fácilmente distribuidos.

E.3. Instalación en sistemas GNU/Linux

En lo que respecta al nodo del servidor, la automatización de la instalación está a cargo del archivo "3dv-serv-setup-linux64", mientras que del lado del cliente los archivos "3dv-cliente-setup-linux64" o "3dv-cliente-setup-linux32" tienen esta tarea. Todos son scripts que contienen una serie de comandos para el intérprete conocido como Shell, e implementan un algoritmo que ejecuta los pasos necesarios para efectuar una instalación correcta. Específicamente, escanean el sistema para determinar cuáles de las bibliotecas dinámicas contenidas en el archivo .tar.gz deben ser instaladas y cuáles no, actualizan el registro de bibliotecas dinámicas en el sistema para que puedan ser utilizadas, copian el archivo ejecutable, y en el caso del servidor también lo configuran como un demonio para que se ejecute cada vez que el sistema inicia. Puede suceder que en el sistema existan versiones previas o más recientes de las bibliotecas que se van a copiar, por lo que el script también verifica estas condiciones para determinar si la instalación de dichos archivos tendrá lugar.
Antes de exponer las actividades realizadas por el script de Shell que automatiza la instalación, es necesario exponer la convención empleada por las distribuciones del sistema operativo GNU/Linux para nombrar y emplear bibliotecas dinámicas. El nombre de éstas bibliotecas suele comenzar con el prefijo "lib" y la extensión de archivo es ".so" (abreviación para shared object); lo más común es que luego de la extensión se presenten uno o más dígitos, separados por puntos, que denotan las versiones de dicha biblioteca [82]. Entonces, el nombre genérico para cada biblioteca es el siguiente:

```
lib<nombre>.so.<version-mayor>.<version-menor>.<release>
```

Donde <nombre> es el nombre de la biblioteca, y <version-mayor>, <version-menor> y <release> son números enteros que denotan las versiones de la biblioteca; el significado de cada una de éstas es la siguiente:

- <version-mayor> estipula la versión de la interfaz de la biblioteca. Es decir, versiones mayores diferentes de una misma biblioteca también tendrán interfaces diferentes.
- Aplicaciones que empleen determinada versión muy probablemente no podrán emplear una distinta, dada la diferencia entre interfaces.
- <version-menor> indica las variaciones en implementación existentes dentro de una versión mayor; no hay diferencias de interfaz entre versiones menores distintas.
- Aplicaciones que empleen cierta versión menor de una biblioteca podrán usar otras versiones menores dado que no hay cambios en la interfaz, o bien dichos cambios no rompen la compatibilidad con versiones previas.
- <release> establece sub-versiones de las versiones menores. Estos cambios suelen denotar correcciones de errores menores; tampoco hay cambios en la interfaz entre diferentes valores de <release>, por lo que las aplicaciones pueden emplearlas indistintamente.
- Un nombre con la forma `<nombre>.so` es aquél que el enlazador (linker) emplea para solicitar una determinada biblioteca. Por lo tanto, se le conoce como "nombre de enlazador" (*linker name*).
- El nombre con la forma `lib<nombre>.so.<version-mayor>` se le conoce cotidianamente como *soname* en inglés. Éste es el nombre que los archivos ejecutables emplean internamente para solicitar bibliotecas; básicamente corresponde a la versión mayor de la biblioteca que el enlazador empleó para generarlos.
- `lib<nombre>.so.<version-mayor>.<version-menor>.<release>` es el nombre real (es decir, el nombre completo) de la biblioteca.

Las bibliotecas en GNU/Linux no siempre especifican sus versiones hasta <release>. Aquéllas que especifican a partir de <version-menor> suelen contar con ligas simbólicas que apuntan al verdadero archivo (conocido como *hard link* en inglés) de la biblioteca. Por ejemplo, la biblioteca "libQtCore.so.4.7.4" especifica su versión hasta <release>, y puede tener una liga simbólica denominada "libQtCore.so.4" o "libQtCore.so.4.7" (o ambas) que apunte hacie ella; la librería "libvtkCommonCore-6.0.so.1" sólo especifica su versión hasta <version-mayor>, y puede tener una liga simbólica llamada "libvtkCommonCore.so" apuntando hacia ella. Tanto el enlazador como los archivos ejecutables pueden hacer uso de estas ligas simbólicas para encontrar las bibliotecas que requieren.
La Figura E.1 presenta al diagrama de actividad seguido por el script de Shell para la instalación del servidor. La parte derecha de este diagrama expone las condiciones relacionadas con las librerías dinámicas anteriormente expuestas, donde "A", "B" y "C" son la versión mayor, menor y el release, respectivamente, de una biblioteca que se quiere instalar en el sistema; "X", "Y" y "Z" son la versión mayor, menor y el release, respectivamente, de una biblioteca existente en el sistema.

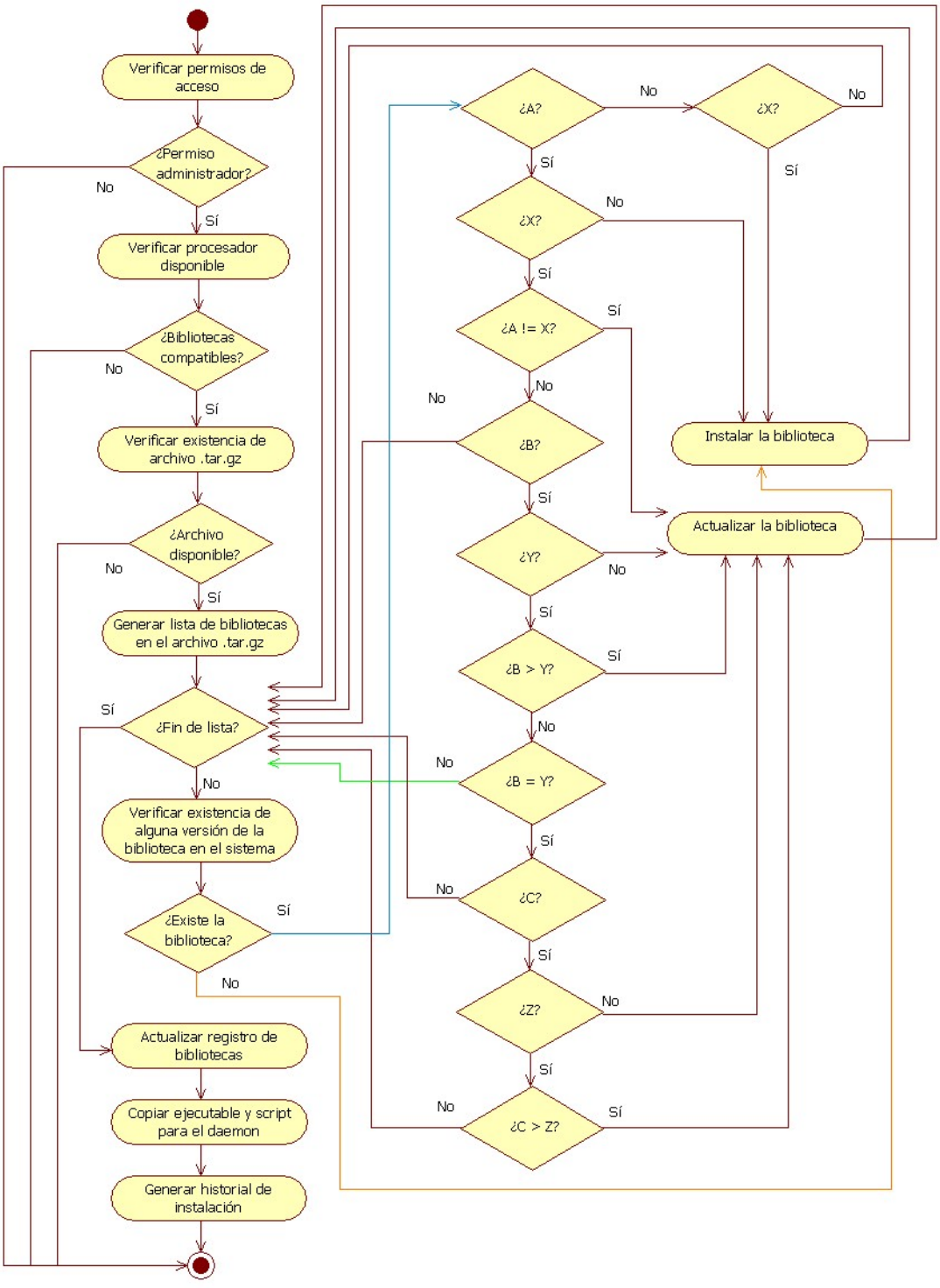


Figura E.1: Diagrama de actividad para el script de Shell empleado en la instalación del servidor. "A", "B" y "C" son la versión mayor, menor y el release, respectivamente, de una biblioteca que se quiere instalar en el sistema; "X", "Y" y "Z" son la versión mayor, menor y el release, respectivamente, de una biblioteca existente en el sistema. Cuando estas variables estén encerradas entre signos de interrogación, la condición se refiere a si la versión en cuestión está o no especificada.

Bibliografía

[1] P. S. Calhoun, B. S. Kuszyk, D. G. Heath, J. C. Carley, and E. K. Fishman, "Three-dimensional Volume Rendering of Spiral CT Data: Theory and Method," *RadioGraphics*, vol. 9, pp. 745–764, 1999. [Online]. Available: http://radiographics.rsna.org/content/19/3/745.full

[2] D. Hanson and R. Robb, "Three–Dimensional Visualization in Medicine and Biology," in *Handbook of Medical Image Processing and Analysis*, 2nd ed., I. N. Bankman, Ed. Burlington: Academic Press, 2009, pp. 755–784.

[3] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed., ser. SEI Series in Software Engineering. Addison Wesley, 2013.

[4] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Addison Wesley, 1998.

[5] OpenUP. Eclipse Public License V1.0. Eclipse Foundation. [Online]. Available: http://epf.eclipse.org/wikis/openup/

[6] P. Kroll and P. Kruchten, *The Rational Unified Process Made Easy. A Practitioner's Guide to the RUP*. Addison Wesley, 2003.

[7] W. Humphrey, *PSP: A Self-improvement Process for Software Engineers*. Addison Wesley, 2005.

[8] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. (2001) Manifesto for software agile development. [Online]. Available: http://agilemanifesto.org/

[9] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004.

[10] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison Wesley, 2005.

[11] K. Beck, *Test Driven Development by Example*. Addison Wesley, 2003.

[12] D. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[13] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," Keele University and University of Durnham, Tech. Rep., July 2007.

[14] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE'08. Swinton, UK: British Computer Society, 2008, pp. 68–77. [Online]. Available: http://ewic.bcs.org/content/ConWebDoc/19543

[15] S. Beecham, N. Badoo, T. Hall, H. Robinson, and H. Sharp, "Protocol for a Systematic Literature Review of Motivation in Software Engineering," School of Computer Science, University of Hertfordshire, Hatfield, Herts AL10 9AB, Tech. Rep. 453, September 2006. [Online]. Available: https://uhra.herts.ac.uk/dspace/bitstream/2299/992/1/S73.pdf

[16] M. Unterkalmsteiner, T. Gorschek, A. Islam, C. K. Cheng, R. Permadi, and R. Feldt, "Evaluation and measurement of software process improvement - a systematic literature review," *IEEE Transactions on Software Engineering,*, no. 2, pp. 398–424, March 2012.

[17] A. Fernandez, E. Insfran, and S. AbrahÃ£o, "Usability evaluation methods for the web: A systematic mapping study," *Information and Software Technology*, vol. 53, no. 8, pp. 789 – 817, 2011, advances in functional size measurement and effort estimation - Extended best papers. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584911000607

[18] (2015, Nov) Ieee xplore digital library. Institute of Electrical and Electronics Engineers. [Online]. Available: http://ieeexplore.ieee.org/Xplore/home.jsp

[19] (2015, Nov) Acm digital library. Association of Computing Machinery. [Online]. Available: http://dl.acm.org/

[20] (2015, Nov) Scopus. Elsevier. [Online]. Available: https://www.elsevier.com/solutions/scopus

[21] (2015, Nov) Springer link. Springer. [Online]. Available: http://link.springer.com/

[22] O. Patashnik and L. Lamport. (2015, Oct) JabRef reference manager. [Online]. Available: http://jabref.sourceforge.net/

[23] T. Hachaj and M. R. Ogiela, "Novel Applications of VR: Visualization of perfusion abnormalities with GPU-based volume rendering," *Comput. Graph.*, vol. 36, no. 3, pp. 163–169, May 2012.

[24] W.-K. Jeong, J. Beyer, M. Hadwiger, A. Vazquez, H. Pfister, and R. T. Whitaker, "Scalable and Interactive Segmentation and Visualization of Neural Processes in EM Datasets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1505–1514, Nov. 2009.

[25] N. Razzali, M. Rahim, D. Daman, and M. Kasmuni, "A design consideration of neuron 3D reconstruction visualization for neuronal morphology," in *2009 IEEE 9th Malaysia International Conference on Communications (MICC)*, 2009, pp. 84–89.

[26] Y. Assaf and O. Pasternak, "Diffusion Tensor Imaging (DTI)-based White Matter Mapping in Brain Research: A Review," *Journal of Molecular Neuroscience*, vol. 34, no. 1, pp. 51–61, 2008.

[27] W. M. Jainek, S. Born, D. Bartz, W. Strasser, and J. Fischer, "Illustrative hybrid visualization and exploration of anatomical and functional brain data," in *Proceedings of the 10th Joint Eurographics / IEEE - VGTC conference on Visualization*, ser. EuroVis'08. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 855–862.

[28] B. Csebfalvi, "Prefiltered gaussian reconstruction for high-quality rendering of volumetric data sampled on a body-centered cubic grid," in *Visualization, 2005. VIS 05. IEEE*, 2005, pp. 311–318.

[29] S. Parker, M. Parker, Y. Livnat, P.-P. Sloan, C. Hansen, and P. Shirley, "Interactive ray tracing for volume visualization," in *ACM SIGGRAPH 2005 Courses*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005.

[30] A. Wenger, D. F. Keefe, S. Zhang, and D. Laidlaw, "Interactive volume rendering of thin thread structures within multivalued scientific data sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 6, pp. 664–672, 2004.

[31] S. Bischoff and L. P. Kobbelt, "Isosurface Reconstruction with Topology Control," in *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, ser. PG '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 246–.

[32] M. Benhamou and F. Clara, "Advances in 3D computer vision for neuron analysis," in *Systems, Man and Cybernetics, 1993. 'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on*, 1993, pp. 508–512 vol.3.

[33] R. Shen, P. Boulanger, and M. Noga, "MedVis: A Real-Time Immersive Visualization Environment for the Exploration of Medical Volumetric Data," in *BioMedical Visualization, 2008. MEDIVIS '08. Fifth International Conference*, 2008, pp. 63–68.

[34] A. Neubauer, S. Wolfsberger, M.-T. Forster, L. Mroz, R. Wegenkittl, and K. Buhler, "Advanced Virtual Endoscopic Pituitary Surgery," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 5, pp. 497–507, Sep. 2005.

[35] S. Drouin, M. Kersten-Oertel, S. J.-S. Chen, and D. L. Collins, "A realistic test and development environment for mixed reality in neurosurgery," in *Proceedings of the 6th international conference on Augmented Environments for Computer-Assisted Interventions*, ser. AE-CAI'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 13–23.

[36] W. Jeong, J. Beyer, M. Hadwiger, R. Blue, C. Law, A. Vazquez, C. Reid, J. Lichtman, and H. Pfister, "SSECRETT and NeuroTrace: Interactive Visualization and Anaysis

Tools for Large-Scale Neuroscience Datasets," *IEEE Computer Graphics and Applications*, vol. PP, no. 99, pp. 1–1, 2011.

[37] N. Schimke, M. Kuehler, and J. Hale, "Preserving privacy in structural neuroimages," in *Proceedings of the 25th annual IFIP WG 11.3 conference on Data and applications security and privacy*, ser. DBSec'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 301–308.

[38] Y. Wan, H. Otsuna, C.-B. Chien, and C. Hansen, "An interactive visualization tool for multi-channel confocal microscopy data in neurobiology research," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1489–1496, Nov. 2009.

[39] J. Beyer, M. Hadwiger, S. Wolfsberger, and K. Bühler, "High-Quality Multimodal Volume Rendering for Preoperative Planning of Neurosurgical Interventions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1696–1703, Nov 2007.

[40] F. A. Cardillo, M. Cesaraccio, A. Starita, and E. Neri, "A 3-D CAD Tool for CT Colonography," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, 2007, pp. 3757–3760.

[41] C. R. Johnson and D. M. Weinstein, "Biomedical computing and visualization," in *Proceedings of the 29th Australasian Computer Science Conference - Volume 48*, ser. ACSC '06. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2006, pp. 3–10.

[42] S. Pallapotu and N. Pizzi, "Volumetric display of magnetic resonance images using Scopira and OpenGL," in *Canadian Conference on Electrical and Computer Engineering 2004*, vol. 4, 2004, pp. 1885–1888.

[43] S. K. Warfield, F. Talos, C. Kemper, L. O'Donnell, C.-F. Westin, W. M. Wells, P. M. Black, F. A. Jolesz, and R. Kikinis, "Capturing brain deformation," in *Proceedings of the 2003 international conference on Surgery simulation and soft tissue modeling*, ser. IS4TM'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 203–217.

[44] A. Tahmasebi, K. Hashtrudi-Zaad, D. Thompson, and P. Abolmaesumi, "A Framework for the Design of a Novel Haptic-Based Medical Training Simulator," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, no. 5, pp. 658–666, 2008.

[45] E. Acosta and A. Liu, "A pipeline virtual environment architecture for multicore processor systems," *The Visual Computer*, vol. 28, no. 11, pp. 1099–1114, 2012.

[46] J.-D. Lesage and B. Raffin, "A hierarchical component model for large parallel interactive applications," *The Journal of Supercomputing*, vol. 60, no. 3, pp. 389–409, 2012.

[47] P. E. Radau, S. Pintilie, R. Flor, L. Biswas, S. O. Oduneye, V. Ramanan, K. A. Anderson, and G. A. Wright, "VURTIGO: visualization platform for real-time, MRI-Guided cardiac electroanatomic mapping," in *Proceedings of the Second international conference*

on Statistical Atlases and Computational Models of the Heart: imaging and modelling challenges, ser. STACOM'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 244–253.

[48] A. Rey, A. Castro, J. C. Dafonte, and B. Arcay, "A novel visualizer of medical images by integrating an extensible plugin framework," in *Proceedings of the 4th international conference on Ambient Assisted Living and Home Care*, ser. IWAAL'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 41–48.

[49] A. Vemuri, J.-H. Wu, K.-C. Liu, and H.-S. Wu, "Deformable three-dimensional model architecture for interactive augmented reality in minimally invasive surgery," *Surgical Endoscopy*, vol. 26, no. 12, pp. 3655–3662, 2012.

[50] F. Zhang, W. DiSanto, J. Ren, Z. Dou, Q. Yang, and H. Huang, "A Novel CPS System for Evaluating a Neural-Machine Interface for Artificial Legs," in *2011 IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS),*, 2011, pp. 67–76.

[51] I. Buzurovic, T. Podder, L. Fu, and Y. Yu, "Modular Software Design for Brachytherapy Image-Guided Robotic Systems," in *BioInformatics and BioEngineering (BIBE), 2010 IEEE International Conference on*, 2010, pp. 203–208.

[52] S. E. Mahmoudi, A. Akhondi-Asl, R. Rahmani, S. Faghih-Roohi, V. Taimouri, A. Sabouri, and H. Soltanian-Zadeh, "Web-based interactive 2D/3D medical image processing and visualization software," *Comput. Methods Prog. Biomed.*, vol. 98, no. 2, pp. 172–182, May 2010.

[53] A. Maurizi, D. Franchi, and G. Placidi, "An optimized Java based software package for biomedical images and volumes processing," in *Medical Measurements and Applications, 2009. MeMeA 2009. IEEE International Workshop on*, 2009, pp. 219–222.

[54] X. Li-qiang, P. Fang, L. Shu-yu, L. De-yu, and F. Yu-bo, "Three-dimensional reconstruction and analysis of human central sulcus based on visualization toolkit," in *The 2nd International Conference on Bioinformatics and Biomedical Engineering 2008*, 2008, pp. 2397–2400.

[55] C. Gustafson, W. Bug, and J. Nissanov, "NeuroTerrain - A client-server system for browsing 3D biomedical image data sets," *BMC Bioinformatics*, vol. 8, 2007.

[56] O. Kum, "Telematics-based online client–server/client collaborative environment for radiotherapy planning simulations," *Medical & Biological Engineering & Computing*, vol. 45, no. 11, pp. 1053–1063, 2007.

[57] O. Ratib and A. Rosset, "Open-source software in medical imaging: development of OsiriX," *International Journal of Computer Assisted Radiology and Surgery*, vol. 1, no. 4, pp. 187–196, 2006.

[58] E. Keeve, T. Jansen, B. von Rymon-Lipinski, Z. Burgielski, N. Hanssen, L. Ritter, and M. Lievin, "An open software framework for medical applications," in *Proceedings of the 2003 international conference on Surgery simulation and soft tissue modeling*, ser. IS4TM'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 302–310.

[59] K. Engel, P. Hastreiter, B. Tomandl, K. Eberhardt, and T. Ertl, "Combining local and remote visualization techniques for interactive volume rendering in medical applications," in *Visualization 2000. Proceedings*, 2000, pp. 449–452.

[60] P. St-Jean, A. Sadikot, L. Collins, D. Clonda, R. Kasrai, A. Evans, and T. Peters, "Automated atlas integration and interactive three-dimensional visualization tools for planning and guidance in functional neurosurgery," *Medical Imaging, IEEE Transactions on*, vol. 17, no. 5, pp. 672–680, 1998.

[61] I. Carlbom, D. Terzopoulos, and K. Harris, "Computer-assisted registration, segmentation, and 3D reconstruction from images of neuronal tissue sections," *Medical Imaging, IEEE Transactions on*, vol. 13, no. 2, pp. 351–362, 1994.

[62] M. Levoy, "Display of surfaces from volume data," *IEEE Comput. Graph. Appl.*, vol. 8, no. 3, p. 29 37, May 1988.

[63] ——, "Efficient ray tracing of volume data," *ACM Trans. Graph.*, vol. 9, no. 3, pp. 245–261, Jul 1990.

[64] K. Mueller and A. Kaufman, "Volume Visualization in Medicine," in *Handbook of Medical Image Processing and Analysis*, 2nd ed., I. N. Bankman, Ed. Academic Press, 2009, ch. 46, pp. 785–816.

[65] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Addison Wesley, 2011.

[66] Kitware. (2015, May) Visualization toolkit. Kitware. [Online]. Available: http://www.vtk.org/

[67] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, 4th ed. Kitware, 2006.

[68] Kitware. (2015, May) Insight segmentation and registration toolkit. Kitware. [Online]. Available: www.itk.org

[69] M. Barbacci, R. Ellison, A. Lattanze, J. Stafford, C. Weinstock, and W. Wood, "Quality Attribute Workshops (QAWs), Third Edition," Software Engineering Institute, Tech. Rep., August 2003. [Online]. Available: http://resources.sei.cmu.edu/asset_files/technicalreport/2003_005_001_14249.pdf

[70] R. Wojcik, F. Bachman, L. Bass, P. Clements, P. Merson, R. Nord, and B. Wood, "Attribute-Driven Design," Software Engineering Institute, Tech. Rep., November 2006. [Online]. Available: http://resources.sei.cmu.edu/asset_files/TechnicalReport/2006_005_001_14795.pdf

[71] N. Harrison and P. Avgeriou, "How do architecture patterns and tactics interact? A model and annotation," *The Journal of Systems and Software*, vol. 83, pp. 1735–1758, 2010.

[72] M. Casales-Cabrera, "Análisis de arquitecturas de software en ambientes de desarrollo Ágil," Master's thesis, Universidad Nacional Autónoma de México, 2011.

[73] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*. Addison Wesley, 2002.

[74] Óscar Castro-López, "Modelo de referencia para la estimación del valor del producto software durante el proceso de desarrollo," Master's thesis, Universidad Autónoma Metropolitana, unidad Iztapalapa, May 2013. [Online]. Available: http://mcyti.izt.uam.mx/archivos/Tesis/Generacion2010/ICR_OscarCastro.pdf

[75] M. O'Brien and L. Chretienneau. (2015, Mayo) ProjectLibre. GeekPolis. [Online]. Available: http://www.projectlibre.org/

[76] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*. Addison Wesley, 1999.

[77] P. Clements, F. Bachman, L. Bass, D. Garland, J. Ivers, R. Little, P. Merson, R. Nord, and J. Stafford, *Documenting Software Architectures*, 2nd ed., ser. SEI Series in Software Engineering. Addison Wesley, 2011.

[78] D. Comer, *Internetworking with TCP/IP. Principles, protocols and architecture*, 3rd ed. Prentice Hall, 1995, vol. 1.

[79] J. Nielsen, *Usability Engineering*. Academic Press, 1993.

[80] O. Pianykh, *Digital Imaging and Communications in Medicine (DICOM)*. Springer-Verlag, 2008.

[81] (2015, Nov) The Jargon File, version 4.4.7. [Online]. Available: http://www.catb.org/jargon/html/D/daemon.html

[82] J.-D. Dodin, S. Pawlowicz, D. Lawyer, R. Moen, S. Chukov, S. Shannigrahi, G. Fergusson, S. Gjoen, and J. Ditchburn. (2015, Nov) The Linux Documentation Project. [Online]. Available: http://tldp.org/HOWTO/Program-Library-HOWTO/shared-libraries.html