ARTÍCULO ORIGINAL

Arquitectura de software para el sistema de visualización médica Vismedic

Software Architecture for the Vismedic medical visualization system

Ing. Alina Dolores Rodríguez Peña,[I] Ing. Luis Guillermo Silva Rojas[II]

I Vertex Entornos Interactivos 3D, Universidad de las Ciencias Informáticas, Carretera a San Antonio, km 2 ½, Torrens, Boyeros, La Habana, Cuba. CP: 19370. E-Mail: alina@uci.cu
II Vertex Entornos Interactivos 3D, Universidad de las Ciencias Informáticas, Carretera a San Antonio, km 2 ½, Torrens, Boyeros, La Habana, Cuba. CP: 19370. E-Mail: lgsilva@uci.cu

RESUMEN

En los últimos años la Arquitectura de Software se ha consolidado como una disciplina que intenta contrarrestar los efectos negativos que pueden surgir durante el desarrollo de un software, ocupando un rol significativo en la estrategia de negocio de una organización que basa sus operaciones en el software. En el presente trabajo se propone una arquitectura de software basada en la integración de los estilos arquitectónicos: Arquitectura basada en componentes, Arquitectura basada en capas y Tuberías y filtros, para el sistema de visualización médica Vismedic, con el objetivo de reducir los problemas de extensibilidad, reusabilidad y dependencias que existían en la arquitectura anterior. Para realizar la propuesta se hizo necesario el estudio de los conceptos relacionados con la Arquitectura de Software, las características arquitectónicas de tres productos establecidos en el campo del procesamiento y visualización de imágenes: Volumen Rendering Engine (Voreen), Visualization Toolkit (VTK) e Insight Toolkit (ITK) y de la especificación OSGi para el desarrollo basado en componentes. La arquitectura propuesta integra las principales características de las bibliotecas antes mencionadas e incorpora el empleo de plugins para extender las funcionalidades. La misma se validó a través de la Técnica de evaluación basada en prototipos y de la aplicación del Método de Análisis de Acuerdos de Arquitectura de Software (ATAM). La evaluación permitió identificar los riesgos presentes en la propuesta realizada y determinar que la arquitectura satisface los atributos de calidad definidos para la presente investigación.

Palabras claves: arquitectura de software, componentes, estilo arquitectónico, plugins, Vismedic.

ABSTRACT

In recent years, Software Architecture has become a discipline that tries to counter the negative effects that may arise during software development, occupying a significant role in the business strategy of an organization that bases its operations on software. This paper proposes a software architecture based on the integration of the architectural styles: Component-based architecture, Layers based architecture and Pipes and Filters, for the Vismedic medical visualization system, with the objective of reducing the problems of extensibility, reusability and dependencies of the previous architecture. In order to develop the proposal was becoming necessary to study the concepts related to Software Architecture, the architectural features of three established products in the image processing and viewing field, they were: Volume Rendering Engine (Voreen), Visualization Toolkit (VTK) and Insight Toolkit (ITK), and the OSGi specification for component-based development. The proposed architecture integrates the main features of the libraries mentioned above and incorporates the use of plugins to extend its functionalities. The architecture was validated through the Prototyping based evaluation technique and the application of the Architecture Tradeoff Analysis Method (ATAM). The evaluation allowed us to identify the risks of the proposal and to determine that the architecture satisfies the quality attributes defined for this investigation.

Key words: architectural style, component, plugins, software architecture, Vismedic.

INTRODUCCIÓN

Las organizaciones actuales, independientemente de su tipo, requieren constantemente de aplicaciones informáticas que contribuyan a solucionar problemas relacionados con su proceso de negocio, automatizar las tareas que realizan frecuentemente y que proporcionen asistencia a la hora de tomar decisiones. Para esto se hace necesaria la construcción de eficientes sistemas de software, estos generalmente requieren la combinación de diferentes tecnologías y plataformas de hardware y software para alcanzar un comportamiento acorde con las necesidades de las organizaciones. Los sistemas de software deben ofrecer un alto nivel de rendimiento, adaptarse a las necesidades específicas de la organización y permitir la adición de nuevas funcionalidades con el menor esfuerzo posible. La consecución de estas características exige a los profesionales dedicados al desarrollo de software poner especial atención y cuidado en el diseño de la arquitectura que soportará el funcionamiento del sistema.

Entre los proyectos de desarrollo de software del centro Vertex, se encuentra el sistema de visualización médica Vismedic, este tiene como objetivo, desarrollar aplicaciones que sirvan de apoyo a los médicos en el proceso de diagnóstico a través de imágenes médicas digitales. Actualmente el proyecto cuenta con un

Visualizador 2D de imágenes médicas digitales en formato DICOM y se encuentra en la fase de pruebas del Visualizador 3D.

Luego de tres años de desarrollo, se comprobó que al adicionar nuevos requisitos funcionales o cambiar sustancialmente los existentes, se necesita realizar modificaciones en la mayor parte del software. Estas modificaciones se deben al fuerte acople, alto nivel de dependencia entre los elementos que componen el sistema y la débil reusabilidad que se puede alcanzar con los mismos. Las deficiencias presentes en la arquitectura actual ralentizan la incorporación de nuevas características a los productos que se desarrollan en el proyecto, lo que conlleva a notables atrasos en el cronograma de desarrollo y aumenta la curva de aprendizaje para los nuevos desarrolladores que se incorporan cada curso.

Teniendo en cuenta la situación problémica anterior, se plantea el siguiente problema científico: ¿Cómo reducir los problemas de extensibilidad, reusabilidad y dependencia para adicionar nuevas funcionalidades al proyecto Vismedic? Para darle solución al problema planteado se estableció como objetivo general: definir y validar una arquitectura de software, que permita reducir los problemas de extensibilidad, reusabilidad y dependencias que presenta la arquitectura actual del proyecto Vismedic.

DESARROLLO

ARQUITECTURA DE SOFTWARE:

Actualmente no existe una definición única para el concepto de arquitectura de software, el término ha sido abordado por un gran número de autores,[1] no obstante, se reconoce como la definición más completa la dada por la IEEE Std 1471-2000: "La arquitectura de software es la organización fundamental de un sistema enmarcada en sus componentes, las relaciones entre ellos, y el ambiente, y los principios que orientan su diseño y evolución".[2]

Al diseñar una arquitectura de software se crean y representan componentes que interactúan entre sí, con responsabilidades específicas y se organizan de forma tal que se logren los requerimientos establecidos. Se puede partir con patrones de soluciones probados que se conocen con el nombre de estilos arquitectónicos, patrones arquitectónicos y patrones de diseño.

Un estilo arquitectónico es un conjunto coordinado de restricciones arquitectónicas que regula las funciones/características de los elementos arquitectónicos y las relaciones permitidas entre estos dentro de cualquier arquitectura que se adapte a ese estilo.[3]

Un patrón arquitectónico, en cambio, expresa una organización estructural para los sistemas de software, proporciona un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y directrices para organizar la relación entre ellos. Los patrones arquitectónicos son plantillas para arquitecturas de software concretas. La selección de un patrón arquitectónico es, por lo tanto, una decisión de diseño fundamental en el desarrollo de un sistema de software.[4]

Los patrones de diseño trabajan a una escala intermedia y son independientes del lenguaje de programación. Definen un esquema de refinamiento de los subsistemas o componentes dentro de un sistema, o las relaciones entre estos. Describe una

estructura común y recurrente de componentes interrelacionados, que resuelve un problema general de diseño dentro de un contexto particular.[5]

Arquitectura basada en componentes:

Un componente es una unidad de composición de aplicaciones, que posee un conjunto de interfaces especificadas contractualmente y dependencias del contexto explícitas, puede ser desplegado de forma independiente y está sujeto a la composición por terceras partes.[6] El estilo de arquitectura basada en componentes, describe un acercamiento al diseño de sistemas como un conjunto de componentes que exponen interfaces bien definidas y que colaboran entre sí para resolver el problema. El uso de este estilo facilita el despliegue, pues permite sustituir un componente por su nueva versión sin afectar a otros componentes o al sistema y favorece la reusabilidad de los componentes independientes del contexto, permitiendo que se empleen en otras aplicaciones y sistemas.

Arquitectura en capas:

Este estilo define una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Entre las principales características de este estilo se encuentran la descomposición de los servicios de forma que la mayoría de las interacciones ocurren solo entre capas vecinas. Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces conocidas y cada nivel agrega las responsabilidades y abstracciones del nivel inferior.[7]

Tuberías y filtros:

Una tubería (pipeline) es un conjunto de nodos de procesamiento de datos conectados en serie, la salida de un elemento es la entrada del siguiente. Cada etapa del procesamiento se encapsula en un filtro, los datos se transmiten a través de tubos entre filtros adyacentes,[7] ver figura. 1. El uso de este estilo fomenta la reutilización, el bajo acoplamiento, la concurrencia y facilita, además

Fig 1. Proceso de visualización empleando tuberías y filtros

Arquitectura de proyectos similares:

Para la propuesta de arquitectura de software realizada, se analizaron productos relacionados con el procesamiento de imágenes médicas digitales, que realizan, en cierta medida, funciones homólogas a las de Vismedic. El estudio de los productos Voreen, VTK e ITK tuvo como objetivos fundamentales: determinar las características arquitectónicas que pueden ser reutilizables para la arquitectura de Vismedic y recopilar buenas prácticas de programación empleadas por proyectos consagrados en el campo.

VTK es un sistema de software ampliamente usado para el procesamiento y visualización de datos. Se emplea en cálculos científicos, análisis por imágenes médicas, geometría computacional y visualización de datos volumétricos, entre

otros campos. Su objetivo fundamental es transformar los datos de entrada para hacerlos comprensibles por el sistema visor humano. Por tanto, uno de los requerimientos principales de VTK es la habilidad de crear tuberías de flujo de datos, capaces de adquirir, procesar y visualizar datos. VTK se diseñó como una herramienta altamente flexible, con numerosos componentes intercambiables que pueden combinarse para procesar una amplia variedad de datos.[8]

ITK es una biblioteca de clases para el análisis de imágenes que puede pensarse como una enciclopedia utilizable de algoritmos para procesamiento de imágenes digitales (filtrado, segmentación y registro). Fue desarrollada por un consorcio de universidades, empresas comerciales y numerosos colaboradores individuales de todo el mundo.[9]

Voreen es una biblioteca de código abierto para visualización de volúmenes. Posee alta flexibilidad a la hora de incorporar nuevas técnicas de visualización. Está implementada como una biblioteca multiplataforma (Windows, Linux y Mac.), usando OpenGL como biblioteca gráfica y GLSL como lenguaje de programación para los algoritmos basados en GPU (Graphics Proccesing Unit).[10]

Especificación OSGi para el desarrollo basado en componentes

Para definir el sistema de plugins en Vismedic se estudió la especificación OSGi (del inglés Open Services Gateway Initiative.) para el desarrollo basado en componentes. OSGi, a pesar de ser un sistema (o framework) modular para Java, establece las formas de crear módulos y la manera en que estos interactuarán en tiempo de ejecución, proporciona un entorno orientado a servicios y basado en componentes que ofrece estándares para administrar el ciclo de vida de un componente.[11]

Evaluación de la arquitectura

La evaluación de una arquitectura no define si esta es buena o no, simplemente expresa donde se encuentran los riesgos y fortalezas de la misma. El primer paso para evaluar una arquitectura es conocer qué es lo que se quiere evaluar, de esta forma es posible establecer la base para la evaluación. Los atributos de calidad, a partir de los cuales se evalúa la arquitectura, son requerimientos del sistema que hacen referencia a las características que éste debe satisfacer.

Las técnicas de evaluación de arquitecturas tienen como objetivo, crear especificaciones y predicciones respecto a una propuesta arquitectónica, para saber si satisface las cualidades que el software debe cumplir. Posibilitan, además, establecer comparaciones entre arquitecturas candidatas para determinar cuál satisface mejor un atributo de calidad específico. Estas técnicas pueden clasificarse en dos vertientes fundamentales: cuantitativas y cualitativas. Las técnicas de evaluación cualitativas son usadas cuando la arquitectura se encuentra en construcción, a diferencia de las cuantitativas, que se utilizan cuando la arquitectura ya ha sido implantada. Es más común el uso de las técnicas cualitativas, pues permiten tomar decisiones arquitectónicas en fases tempranas del desarrollo, requieren menor información detallada y pueden conducir a resultados relativamente precisos, además, son menos costosas de realizar que las cuantitativas.[12]

Los métodos de evaluación arquitectónica, evalúan el potencial del diseño arquitectónico para alcanzar los niveles deseados en cuanto a los requisitos de calidad,[13] estos métodos se ven asistidos de las técnicas de evaluación arquitectónica. Actualmente existen diversos métodos para realizar pruebas a la

arquitectura de software, cada uno con características específicas, escoger un método de evaluación requiere tener bien definidos los atributos que se desean evaluar. Algunos de los métodos de evaluación son los siguientes:

- Método de Análisis de Arquitectura de Software (Software Architecture Analysis Method, SAAM).
- Método de Revisión Intermedio de Diseño (Active Reviews for Intermediate Designs, ARID).
- Método de Análisis de Acuerdos de Arquitectura de Software (Architecture Trade-off Analysis Method, ATAM).

Teniendo en cuenta las dificultades que presenta la arquitectura que se utiliza en el proyecto Vismedic, se determinó evaluar en la arquitectura que se propone los siguientes atributos de calidad:

- Confiabilidad: Medida de la habilidad de un sistema de mantenerse operativo a lo largo del tiempo.[13]
- Escalabilidad: Grado con el que se puede ampliar el diseño arquitectónico, de datos o procedimental.[14]
- Funcionalidad: Habilidad del sistema para alcanzar sus objetivos.[14]
- Integralidad: Medida en que trabajan correctamente componentes del sistema que fueron desarrollados separadamente al ser integrados.[13]
- Mantenibilidad: Alta capacidad de modificación y con bajo costo.[14]
- Portabilidad: Habilidad del sistema para ser ejecutado en diferentes plataformas. Estos ambientes pueden ser hardware, software o una combinación de los dos.[14]
- Reusabilidad: Es la capacidad de diseñar un sistema de forma tal que su estructura o parte de sus componentes puedan ser reutilizados en futuras aplicaciones.[13]

Se aplicará el método de evaluación ATAM, este está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación SAAM.[15] Se concentra en la identificación de los estilos o enfoques arquitectónicos utilizados y ayuda a los involucrados en el proyecto a entender las consecuencias de las decisiones arquitectónicas tomadas con respecto a los atributos de calidad.

Se hará uso además de la técnica de evaluación arquitectónica basada en prototipo, esta consiste en implementar una parte de la arquitectura de software y ejecutarla en el contexto del sistema. Mediante esta técnica se obtiene un resultado de evaluación con mayor exactitud.[12] Entre los aspectos favorables de su uso se destaca la confiabilidad, permite observar que tanto se afecta o no, un atributo de calidad. De acuerdo al nivel de fidelidad con que se desarrolle el prototipo del producto final, puede suponer desventajas el empleo de esta técnica, pues el tiempo de desarrollo del prototipo puede ser largo.

Análisis de la arquitectura actual

El proyecto Vismedic cuenta actualmente con dos productos (Visualizador 2D y Visualizador 3D), derivados de un período de tres años de investigación y desarrollo. Durante este proceso, los cambios frecuentes de los requisitos funcionales y los atributos de calidad, condujeron a realizar numerosas modificaciones a la arquitectura de software del proyecto. La arquitectura sobre la que se desarrollan actualmente los productos se basa fundamentalmente en el empleo de capas y módulos, con el objetivo de agrupar las funcionalidades y delimitar las responsabilidades de los elementos que componen las aplicaciones.

Como se observa en la figura 2. la arquitectura cuenta con una capa que representa el núcleo básico (***Core***) para los productos a desarrollar, esta capa se encuentra dividida en módulos que ofrecen funcionalidades comunes para las aplicaciones de visualización de imágenes médicas digitales. Dentro del módulo Plugins, perteneciente a la capa ***Core***, se definen las interfaces a implementar para extender las aplicaciones mediante ***plugins***, es necesario señalar que, en estos momentos, solo el Visualizador 2D hace uso de estos.

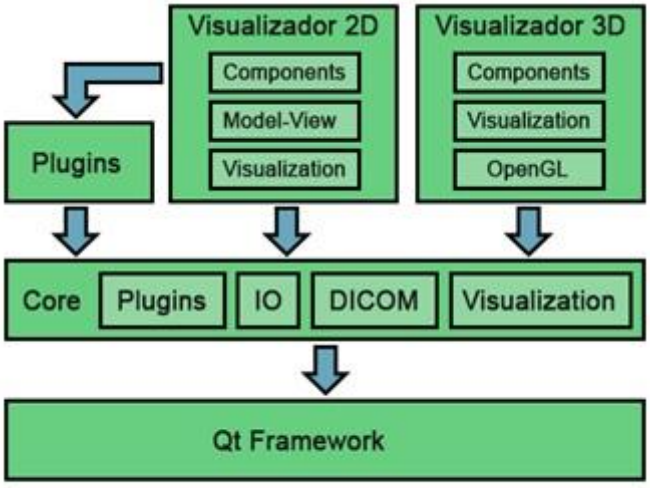

Fig 2. Arquitectura actual del proyecto Vismedic

Luego del análisis realizado, se comprobó que la arquitectura actual presenta un fuerte acople y alto nivel de dependencia entre los componentes y módulos que la integran, esto se evidenció a la hora de adicionar nuevas funcionalidades o modificar las existentes, por ejemplo: para adicionar un nuevo algoritmo de visualización tridimensional, se necesita implementar la interfaz **RenderAlgorithm** (ubicada en el módulo **Visualization** de la capa **Core**), registrar explícitamente el nuevo algoritmo en el contexto de visualización de OpenGL y ubicar en la barra de herramientas principal, en caso de ser necesario, un enlace a la ventana de configuración del nuevo algoritmo. Para realizar el despliegue con el nuevo algoritmo adicionado es necesario recompilar el producto y posteriormente reinstalarlo en cada una de las estaciones de trabajo. La arquitectura actual afecta, además, en cierta medida, los siguientes atributos de calidad:

- **Portabilidad:** la biblioteca de clases DCMTK presenta problemas de portabilidad para el sistema operativo Windows, por tanto, la primera versión de los visualizadores está disponible solo para Linux.
- **Modificabilidad - Mantenibilidad:** la actualización de los componentes conlleva, como mínimo, a repetir el proceso completo de despliegue.
- **Reusabilidad - Escalabilidad:** Las interfaces definidas en el módulo ***Plugins***, perteneciente a la capa ***Core***, solo ofrece soporte para algoritmos de filtrado de imágenes bidimensionales.

RESULTADOS

A partir del estudio realizado se determinó que las bibliotecas Voreen, VTK e ITK tienen en común el empleo de una red de flujo de datos, basada en el estilo arquitectónico tuberías y filtros, para el procesamiento y visualización de las

imágenes. Voreen permite además la edición dinámica de la red de flujo de datos, esto posibilita realizar prototipos rápidos de nuevas aplicaciones, y posee una arquitectura multi-capa que separa limpiamente el sistema de visualización de la capa de interfaz gráfica de usuario. Estas bibliotecas no presentan un uso bien definido de plugins con el objetivo de extender su funcionamiento, por tanto, las modificaciones que se realicen deben ser en tiempo de compilación.

El estudio de la especificación OSGi permitió definir las interfaces que deben implementar los plugins que se deseen adicionar al producto que se desarrolle, así como gestionar de forma eficiente las dependencias entre estos y el versionado.

Por tanto, se propone para el proyecto Vismedic, una arquitectura que combina los estilos arquitectónicos basados en capas y en componentes, y que emplee de una red de flujo de datos (estilo arquitectónico tuberías y filtros) para el proceso de visualización de las imágenes. En la figura 3 se observa la distribución de las capas presentes en la arquitectura propuesta.

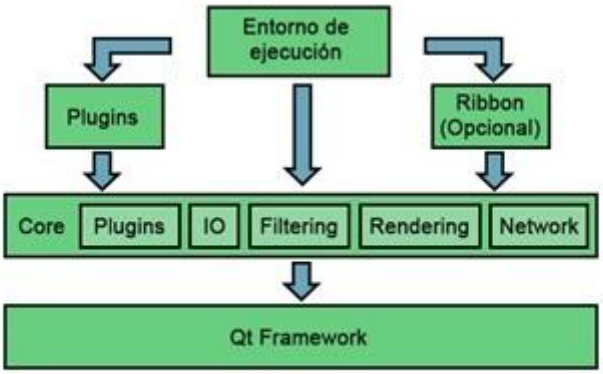

Fig 3. Distribución de las capas presentes en la arquitectura

La capa Qt Framework ofrece el soporte para el manejo de plugins y para la interfaz gráfica de usuario. En la capa Core se encuentran ubicados los módulos y componentes comunes a todas las aplicaciones a desarrollar. Esta capa se encuentra dividida por módulos, dentro de los cuales se pueden mencionar: Plugins (contiene las interfaces a implementar por los plugins concretos) e IO (para entrada y salida de datos). La capa Ribbon permite emplear de forma opcional la biblioteca ribbon desarrollada en el proyecto. Esta biblioteca posibilita crear interfaces gráficas de usuario empleando una cinta de opciones similar a los productos de Microsoft Office 2007 - 2013. La capa Entorno de Ejecución provee el entorno donde se ejecutarán los plugins, así como las funcionalidades de activación y desactivación de estos. Por último, la capa Plugins permite la creación de elementos de extensión para las aplicaciones, el control de su ciclo de vida, así como su versionado y resolución de dependencias internas, las que se gestionarán teniendo en cuenta lo propuesto por la especificación OSGi.

Evaluación de la arquitectura propuesta

La arquitectura propuesta se evaluó utilizando el método ATAM y la Técnica de Evaluación basada en Prototipos, específicamente con la creación de un prototipo funcional. (tabla 1) (tabla 2).

Tabla 1. Escenarios definidos

	Escenarios	Atributos
1	Adicionar, eliminar, deshabilitar o habilitar *plugins*	Funcionalidad (Adecuación - Precisión) Integralidad
2	Estabilidad del sistema ante la ocurrencia de errores	Confiabilidad (Recuperabilidad)
3	Adición de nuevos procesadores o modificación de los existentes	Mantenibilidad (Facilidad de adaptación al cambio)
4	Ejecución del sistema en diferentes sistemas operativos	Portabilidad (Adaptabilidad)
5	Implementación de nuevas funcionalidades a partir de las interfaces existentes	Reusabilidad – Extensibilidad
6	Nivel de adaptación del sistema al incremento de la cantidad de *plugins*	Escalabilidad

Tabla 2. Riesgos detectados

Escenarios	Riesgos
Estabilidad del sistema ante la ocurrencia de errores	El sistema no es capaz de regresar al último estado estable ante la ocurrencia de un error crítico.
Adición de nuevos procesadores o modificación de los existentes	Al ubicar directamente los procesadores en el directorio *plugins*, ubicado en la carpeta de instalación, el sistema no notifica, de forma visual, la no incorporación a la aplicación de los procesadores no compatibles con la misma.

Luego del análisis de los riesgos detectados se determinó que estos no afectan considerablemente el desempeño de las aplicaciones que se pueden desarrollar con la arquitectura propuesta, por tanto, se decide mitigar los mismos en una segunda iteración de la arquitectura.

Prototipo funcional

La creación del prototipo funcional permitió evaluar, de forma práctica, los atributos de calidad de interés, tal es el caso de la integralidad entre los componentes que fueron desarrollados de forma separada y la reusabilidad y extensibilidad que se obtiene al crear los procesadores en forma de plugins.

En la figura 4 se observa, en el prototipo implementado, el resultado de adicionar cinco procesadores a la red, concatenar los mismos y ejecutar la red de procesamiento creada. A la derecha se observa la interfaz "Administrar plugins" con las operaciones que se pueden realizar sobre la misma (Adicionar, Eliminar, Habilitar o Deshabilitar un plugin).

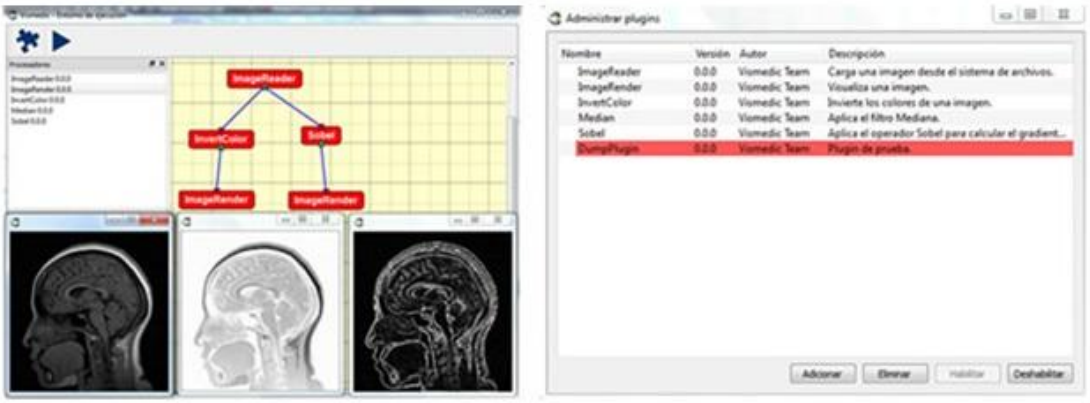

Fig 4. Prototipo funcional

Discusión general de la arquitectura propuesta

Luego de la aplicación del ATAM y la técnica de evaluación basada en prototipo, se comprobó que el empleo de la arquitectura propuesta influye positivamente en los atributos de calidad que se ven afectados con el uso de la arquitectura actual. A continuación, se especifica el efecto de la aplicación de la arquitectura propuesta sobre los principales atributos de calidad definidos:

- **Portabilidad:** la arquitectura propuesta solo depende del framework Qt, una de las fortalezas de este framework es su portabilidad. El empleo de una biblioteca externa como DCMTK se delega hacia la implementación de los plugins concretos, por lo que si esta biblioteca presenta problemas de portabilidad solo se afecta el plugin que la emplea.

- **Modificabilidad - Mantenibilidad:** para la actualización o adición de nuevas funcionalidades, no es necesario repetir el proceso completo de despliegue, solo se necesita adicionar, desde la interfaz visual "Administrar plugins", el plugin que contiene la funcionalidad o copiarlo al directorio ***plugins***.

- **Reusabilidad - Escalabilidad:** las interfaces definidas en el módulo Plugins, pueden emplearse para construir dinámicamente la aplicación, por tanto, la adición de un plugin registra automáticamente sus componentes visuales en la aplicación (desencadenadores, ventanas de configuración y procesadores).

Aplicación práctica de la arquitectura propuesta

Al término de la presente investigación, el software Vismedic - Illustration, encargado de generar ilustraciones a partir de datos volumétricos, se encuentra en fase de adopción de la arquitectura propuesta y presenta una implementación parcial de la misma. En este caso, y como se observa en la figura 5, los algoritmos de filtrado y visualización se cargan en forma de plugins y contienen los procesadores necesarios para construir la red de flujo de datos, de esta forma la aplicación se construye automáticamente a partir de la información contenida en los plugins. La adopción, hasta el estado actual, de la arquitectura propuesta, favoreció el desarrollo basado en componentes y su posterior integración, delegando a los desarrolladores solo la implementación de los procesadores concretos.

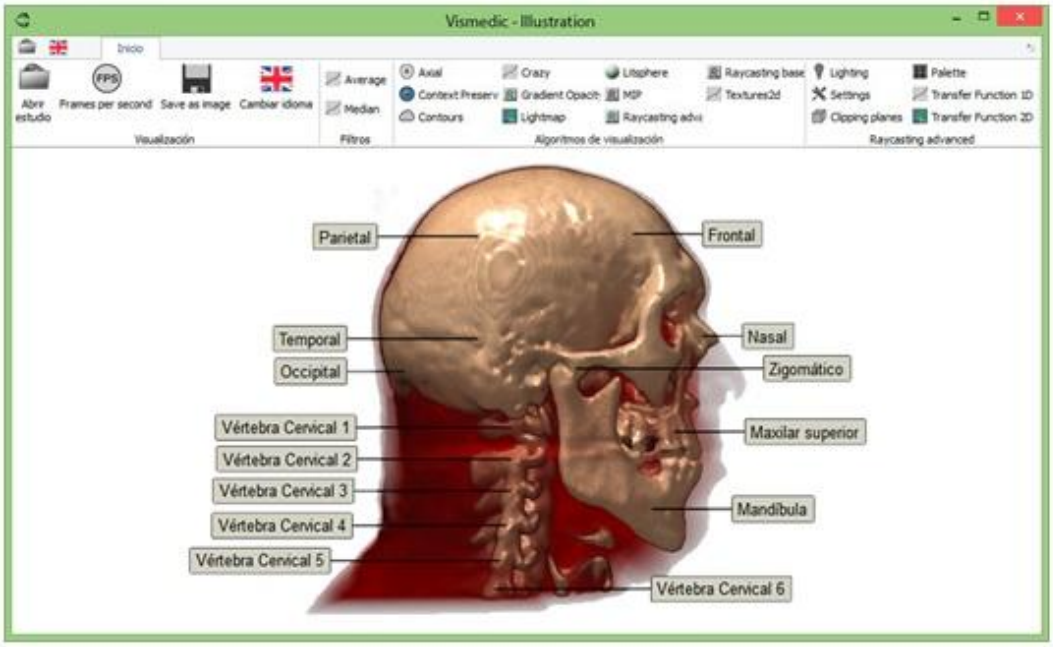

Fig 5. Implementación parcial de la arquitectura propuesta en el software Vismedic - Illustration

CONCLUSIONES

Con la realización de este trabajo, se definió y validó una arquitectura de software para el proyecto Vismedic, que permite reducir los problemas de extensibilidad, reusabilidad y dependencias que presenta la arquitectura sobre la que se desarrollan actualmente los productos del proyecto. De esta forma se dio cumplimiento al objetivo propuesto al inicio de la investigación, además:

- La combinación de los estilos arquitectónicos: Arquitectura basada en capas, Arquitectura basada en componentes y Tuberías y filtros, permitió desarrollar una arquitectura que satisface los atributos de calidad reutilización y mantenibilidad.
- El análisis de los productos Voreen, VTK e ITK permitió reutilizar los conceptos fundamentales asociados a una red de flujo de datos (procesadores, puertos, tipos de datos). La incorporación de plugins en la creación de la red de flujo de datos constituye el principal aporte de la investigación.
- La configuración dinámica de la red de flujo de datos por parte del usuario, posibilita crear prototipos rápidos de aplicaciones sin necesidad de conocer el código fuente de la aplicación, lo que incrementa la versatilidad de los productos del proyecto.

REFERENCIAS BIBLIOGRAFICAS

1. SEI | CARNEGIE MELLON. Community Software Architecture Definitions. [Citado: enero 10, 2013] Disponible en: http://www.sei.cmu.edu/architecture/start/glossary/community.cfm

2. ISO/IEC/IEEE 42010: Defining architecture. [Citado: enero 10, 2013]. Disponible en: http://www.iso-architecture.org/ieee-1471/defining-architecture.html

3. Fielding R.T. Architectural styles and the design of network-based software architectures. Ph.D. dissertation. University of California, California, 2000.

4. Buschmann F. Pattern oriented software architecture: a system of patters. Ashish Raut, 1999. p. 25 - 26.

5. Gamma E, Helm R, Johnson R, Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, 1995. 13, 94, 155, p. 249.

6. Szyperski A. Component software: Beyond Object-Oriented programming. Addison-Wesley, 2002, 2nd

7. De la Torre C, Zorrilla U, Calvarro N.J, Ramos M. A, Manteiga Ch, Cortés F, García I. Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0. Krasis Press, Marzo 2010: p. 9-31

8. Gevenci B, Schroeder W. VTK. En: Brown A.E, Wilson G. The Architecture of Open Source Applications: Elegance, Evolution, and a Few Fearless Hacks. CreativeCommons, 2011. p. 315 - 330. [Citado: marzo 15, 2013] Disponible en: http://aosabook.org/en/vtk.html

9. Ibáñez L, King B. ITK. En: Brown A.E, Wilson G. The Architecture of Open Source Applications. Volume II: Structure, Scale, and Few More Fearless Hacks. CreativeCommons, 2012. p. 100 - 126. [Citado: marzo 15, 2013] Disponible en: http://www.aosabook.org/en/itk.html

10. VOREEN - VOLUME RENDERING ENGINE. Voreen - Volume Rendering Engine (Official Project Website). Visualization & Computer Graphics Research Group, University of Münster, Germany. [Citado: enero 29, 2013.] Disponible en: http://voreen.uni-muenster.de/

11. OSGi ALLIANCE. OSGi Core Release 5. Tech. rep. OSGi Alliance. [Citado: diciembre 20, 2012]. Disponible en: http://www.osgi.org/Specifications/HomePage

12. Bosh J. Design & Use of Software Architectures. Adopting and evolving a product-line approach. s.l. : Pearson Education, 2000. pp: 57-62

13. Bass L, Clements P, Kazman R. Software Architecture in Practice, Third Edition. 3. s.l. : Addison-Wesley Professional, Septiembre 2012. 4, p. 18

14. Pressman R.S. Software Engineering. A Practitioner´s Approach. 7th Edition. s.l. : McGraw-Hill. Higher Education, 2010. p. 397 - 415

15. Clements P, Kazman R, Klein M. Evaluating software architectures: methods and case studies. s.l.: Addison-Wesley, 2002. p. 28.

Recibido: 15 de diciembre de 2015.
Aprobado: 8 de abril de 2016.