CIGET - MATANZAS
IDICT
INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

# Arquitectura de Plug-in para Sistemas de Visualización Médica.

# Plug-in Architecture for Medical Visualization Systems

Ing. Cecilia Valdespino Tamayo
Facultad Regional Granma UCI, Manzanillo, Granma, Cuba
ctamayo@grm.uci.cu

**Resumen**

El presente trabajo pretende conformar la propuesta de una arquitectura de software robusta y confiable que permita a los Sistemas de Visualización Médica ser una aplicación extensible, personalizada y reusable. Se realizó el análisis de los elementos relacionados con la Arquitectura de Software, los principales conceptos, estilos y patrones arquitectónicos.

Se describió la propuesta arquitectónica para los Sistemas de Visualización Médica usando el lenguaje de programación C++. La descripción de la arquitectura se basó en la Metodología de Desarrollo de Software Proceso Unificado de Desarrollo (RUP) usando como herramienta CASE Visual Paradigm. Se propuso y se ejecutó una estrategia de evaluación de arquitectura de software con el objetivo de identificar los riesgos y fortalezas de la propuesta y poder tomar medidas correctivas a tiempo.

Como resultado se obtuvo una Arquitectura de plug-in sobre el framework Qt que permite un alto nivel de funcionamiento a la aplicación desarrollada, siendo así un sistema extensible y reusable, cubriendo las necesidades existentes en el proyecto Vismedic de la Facultad 5.

**Palabras Clave**: Arquitectura, Plug-in, Sistemas de Visualización Médica

**Abstract**

This paper aims to shape the architecture proposed robust and reliable software that allows medical imaging systems is an extensive application, customized and reusable. We performed the analysis of the elements related to software architecture, key concepts, styles and architectural patterns and some systems have a built plug-in architecture. Described the proposed architecture for medical imaging systems using the programming language C + +.

Was proposed and implemented a strategy for evaluating software architecture in order to identify risks and strengths of the proposal and to take corrective measures in time.

The result was a plug-in architecture on the Qt framework that allows a high level of performance to the application developed, making it extensible and reusable system, covering the needs in the project of the Faculty Vismedic 5.

**Keywords**: Architecture, Plug-in, Medical Display Systems

CIGET - MATANZAS
IDICT
INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

## Introducción

Las Tecnologías de la Información y las Comunicaciones (TICs) se han convertido en el vehículo común en el tránsito de la vida social de todos, Influyendo de forma favorable en el desarrollo de cada una de las áreas que rigen la sociedad.

En la actualidad la tecnología en las cirugías mínimamente invasivas está cambiando radicalmente en la manera que los médicos efectúan sus procedimientos quirúrgicos. En Cuba existe una extensa red de instituciones de salud que compone la organización en todo el país, contando con una atención médica de alto nivel científico y tecnológico en un ambiente confortable; destacándose el Centro Nacional de Cirugía de Mínimo Acceso, situado en el municipio 10 de Octubre de Ciudad de la Habana. Este centro lleva a efecto una política científica y planificada de generalización de la Cirugía de Mínimo Acceso a todo el país.

Para que el desarrollo de las intervenciones de mínimo acceso posean un alto nivel científico se requiere de la utilización de sistemas de software donde a partir de una visualización tridimensional del órgano del paciente, los médicos especialistas puedan realizar un diagnóstico de patologías y planificación del acto quirúrgico; estos sistemas poseen un alto costo en el mercado internacional, producto a la situación anterior y al intenso bloqueo económico al que está sometida la nación, son imposibles de adquirir.

Con el objetivo de seguir incrementando el avance tecnológico en la medicina del país, en la Facultad 5 de la Universidad de las Ciencias Informáticas surge un proyecto llamado Vismedic, para el desarrollo de un Sistema de Visualización Médica tridimensional de apoyo al diagnóstico y planeación quirúrgica a partir de imágenes médicas digitales.

El sistema está desarrollado desde una perspectiva general con el objetivo de cubrir los requerimientos básicos de todos los Sistemas de Visualización Médica tridimensional como cargar, pre-procesar, segmentar y visualizar imágenes médicas; independientemente de ello los especialista médicos presentan exigencias cada vez mayores, donde si el médico desea que la aplicación cumpla con los requerimientos de sus especialidades los desarrolladores tienen que modificar el código previamente programado. Además el sistema no permite el desarrollo por terceras partes (ampliamente utilizado en el mundo para extender aplicaciones informáticas); el usuario final no puede configurar la interfaz gráfica de la aplicación, obligándolo a tener activo todos los módulos que brinda; y las funcionalidades implementadas son altamente dependientes lo que trae consigo altos costos en tiempo y recursos, también posee poca reusabilidad de las funcionalidades para el desarrollo de futuras aplicaciones especializadas en cualquier rama de la medicina o que tengan el mismo fin de Vismedic.

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

## Materiales y métodos

*Definición de Arquitectura de Software*

La Arquitectura de Software es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones (UCI, 2010).

Es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (Reynoso, 2005).

*Estilo Arquitéctonico Plug-in*

Los estilos arquitectónicos definen las reglas generales de organización en términos de un patrón y las restricciones en la forma y la estructura de un grupo numeroso y variado de sistemas de software (Reynoso, 2005).

Una plug-in (extensión) es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúa por medio de la Interfaz Programada de la Aplicación (por sus siglas en inglés API) (Johannes Mayer, 2006).

Dicho estilo no contiene ninguna variante arquitectónica, se ve de forma independiente. Se enfoca en explicar cómo se puede diseñar una aplicación con el fin de soportar varios plug-in permitiendo que la misma se extienda en tiempo de ejecución mediante la carga dinámica de módulos o clases que no conoce durante la compilación.

*Metodologías y herramientas*

Dentro de las herramientas utilizadas para establecer la línea base de la arquitectura de los Sistemas de Visualización Médica se encuentran: la metodología de desarrollo de software RUP con su lenguaje de modelado UML (Lenguaje Unificado de Modelado) y el uso de la herramienta CASE Visual Paradigm para modelar cada uno de los artefactos arquitectónicamente significados que serán generados en la propuesta de solución.

A continuación se detallen sus principales elementos:

La metodología de desarrollo de software *RUP (Proceso Unificado de Rational, Rational Unified Process en inglés)* tiene como objetivo entregar un producto de software. Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Posee una forma disciplinada de asignar tareas y responsabilidades. Sus características

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

fundamentales se basan en un desarrollo iterativo e incremental, guiado por casos de uso y centrado en la arquitectura. Utiliza a *UML* para especificar, visualizar y documentar los artefactos que se crean durante el proceso de desarrollo. Su ciclo de vida está enmarcado en cuatro fases de desarrollo: Inicio, Elaboración, Construcción y Transición por las cuales circulan nueve disciplinas divididas en seis de ingeniería: Modelamiento del negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue, y tres de apoyo: Gestión de Proyecto, Gestión de la Configuración y el Cambio y Ambiente.

Esta metodología propone una guía para establecer la línea base de la arquitectura de un proyecto de desarrollo de software basada en la modelación de las 4 + 1 vistas.

*Visual Paradigm* constituye la herramienta que lleva a cabo la modelación de los artefactos que forman parte de la propuesta de solución. Herramienta profesional multiplataforma que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite modelar los diferentes diagramas generados en el desarrollo de un software, así como generar código desde diagramas, entre otros elementos que pueden ser de ayuda en la documentación de un sistema determinado. Posee una buena integración con Entornos multi-plataforma de Desarrollo Integrado (IDEs), incluye localización en castellano, muy personalizable y soporta varios lenguajes de programación.

El sistema de Visualiazación Médica Vismedic recoge en cada plug-in una funcionalidad ya sea Gestionar paciente, Pre-procesar imagen, Segmentar imagen, etc y pretende a partir de estos personalizar las configuraciones que puede hacer el usuario en el sistema. A esto responde la siguiente especificación de requisitos:

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

Tabla 1. Especificación de requisitos

| No | Funcionalidad | Descripción | Prioridad |
|---|---|---|---|
| RF 1 | Importar plug-in. | El sistema debe permitirle al usuario seleccionar de un directorio el plug-in que desea añadir a la aplicación. | Alta |
| RF 2 | Desinstalar plug-in. | El sistema debe mostrar el listado de plug-in importados para que el usuario pueda seleccionar uno y desinstalarlo. | Alta |
| RF 3 | Habilitar plug-in | El sistema debe permitir al usuario activar las funcionalidades de un plug-in que ha sido deshabilitado. | Alta |
| RF 4 | Deshabilitar plug-in. | El sistema debe permitir al usuario desactivar las funcionalidades de cualquier plug-in activo. | Alta |

Para llevar a cabo la implementación de las funcionalidades especificadas, se propone un conjunto de clases relacionadas de la siguiente forma:

CIGET - MATANZAS
IDICT
INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

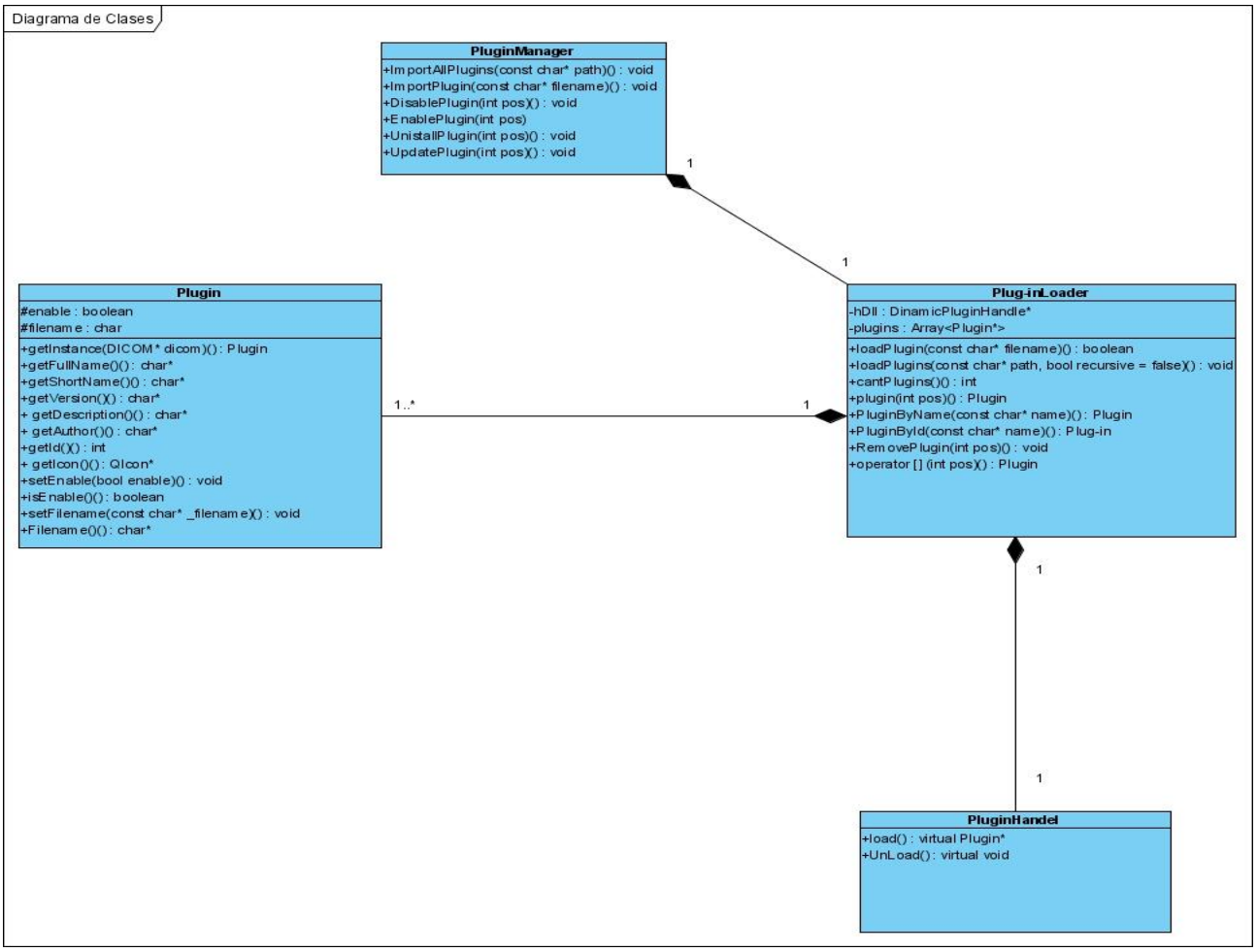

Fig. 1. Diagrama de clases

Breve descripción de las clases ilustradas en la figura 1:

- *Plugin*: Es una clase entidad que define la interfaz de un plug-in dentro de la arquitectura propuesta.
- *PluginHandel*: Es una clase interfaz que posibilita la carga de los ficheros en los cuales está almacenada la implementación del plug-in que se desea importar. En caso de estar en Windows esta clase gestionaría ficheros con extensión (*.dll) y en caso de Linux ficheros con extensión (*.so).
- *PluginLoader*: Es una clase que controla y gestiona el conjunto de objetos correspondientes a los plug-in importados por la aplicación.
- *PluginManager*: Es la clase donde se implementa toda la lógica del trabajo con los plug-in importados en la aplicación.

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

A continuación se muestra la relación entre todos los componentes físicos que deben estar presentes en el desarrollo de la arquitectura y que responden a lo planteado anteriormente:

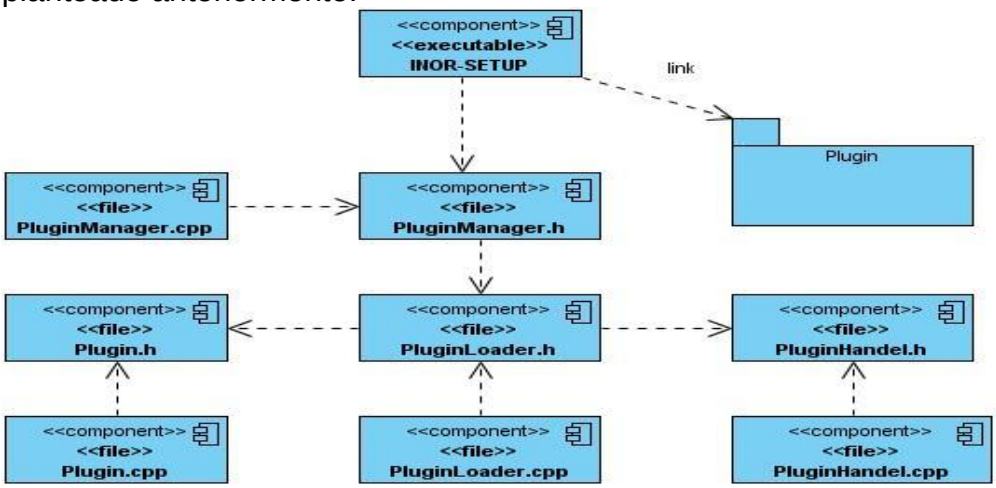


Figura 2 Diagrama de Componentes

El componente INOR-SETUP es el ejecutable que se utilizará para una mejor distribución de la aplicación a los clientes, el cual necesita de cada uno de los archivos que contienen el código fuente. El paquete Plug-in contendrá todos los archivos *dll o *so (con el prototipo “library”) que tendrán también código fuente pero que será cargado por la aplicación en caso que sea necesario o si el propio usuario lo solicita, no se puede definir la cantidad de archivos que contendrá dicho paquete ya que representa un proceso dinámico que permitirá a la aplicación ser extensible y configurable.

**Resultados y discusión**

Para evaluar la arquitectura de software que se propone, el método más conveniente a utilizar es el *ATAM,* este método es considerado el más completo porque revela la forma en que una arquitectura específica satisface los atributos de calidad seleccionados y provee una visión de cómo estos interactúan con otros. Este método presenta como principal desventaja que sólo se utiliza después que la arquitectura ha sido diseñada, como es el caso. No sirve para comparar dos arquitecturas candidatas, pero dado a que no es el proceso que se quiere llevar a cabo; no constituye (desventaja) ningún inconveniente para iniciar la evaluación.
Además se propone utilizar la técnica de evaluación basada en *simulación*, su enfoque básico es la implementación a cierto nivel de abstracción del contexto del sistema donde se supone va a ejecutarse (Erika Machado, 2004). Tiene como objetivo evaluar el comportamiento de la arquitectura bajo diversas circunstancias, una vez disponibles estas implementaciones, pueden usarse los perfiles

CIGET - MATANZAS IDICT INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

respectivos para evaluar los atributos de calidad, pudiéndose mezclar con la técnica basada en *escenarios*, la cual es poco costosa para el equipo de desarrollo.
Para darle cumplimiento a las especificaciones del método y las técnicas de evaluación seleccionadas se hace necesario, ejecutar como primer paso, la definición de los escenarios como se muestra a continuación:

Tabla 2. Definición de escenarios

| Atributos de Calidad | Perfil | Escenarios |
|---|---|---|
| Funcionalidad | Operaciones sobre el plug-in. | 1. Operaciones Desinstalar, Habilitar y Deshabilitar Plug-in.<br>2. Operación Importar Plug-in. |
| Portabilidad | Portabilidad | 3. Existe la necesidad de cambiar la plataforma sobre la cual corre el sistema. |
| Reusabilidad | Reusabilidad | 4. Los plug-in deben ser reutilizables en futuras aplicaciones. |
| Integrabilidad | Integrabilidad | 5. Se desea que los plug-in importados funcionen correctamente. |
| Escalabilidad | Ampliación | 6. Se desea que el diseño arquitectónico pueda ser ampliado. |

La validación de cada uno de estos escenarios se realizó a través de un prototipo funcional que provee al usuario interactuar con una interfaz intuitiva y fácil de manejar, puesto que las funcionalidades que pueden ser ejecutadas son de fácil acceso como se muestra en la **Fig. 3**.

CIGET - MATANZAS
IDICT
INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

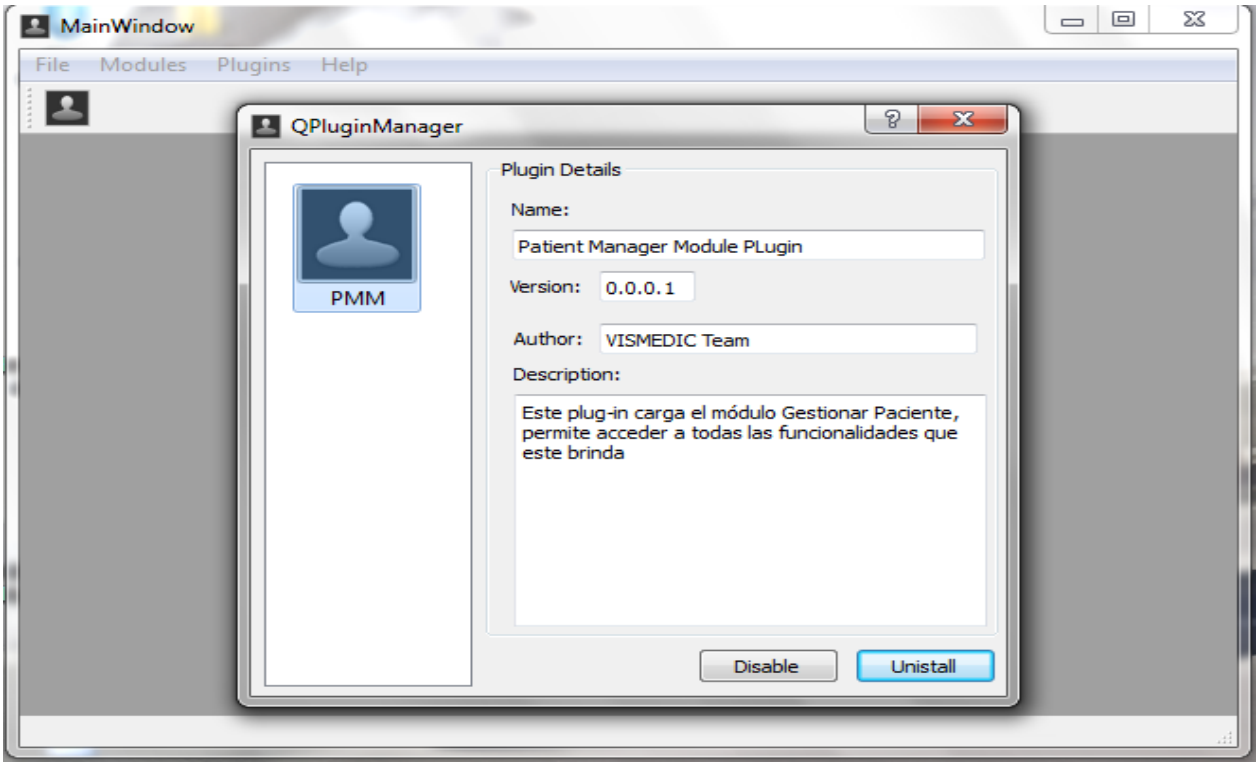


Fig. 3. Prototipo funcional

A partir de la especificación y evaluación de cada uno de los atributos de calidad definidos en la propuesta de solución se puede concluir:

*Riesgos:* A pesar de que la aplicación en cuestión no requiere de módulos que dependan de otro, constituye un riesgo y una desventaja el hecho de que no se pueda ejecutar un plug-in que tenga dependencia con otro, aspecto identificado en los escenarios: Integrabilidad y Reusabilidad.

*Táctica y recomendación***:** Investigación sobre cómo llevar a cabo la implementación y la gestión de la dependencia entre plug-in.

No riesgos:

- En cuanto a las funcionalidades previstas en el diseño de la arquitectura propuesta la simulación realizada no presenta ningún inconveniente, permitiendo *Importar, Desinstalar, Deshabilitar y Habilitar* un plug-in, mientras se cumpla con las restricciones arquitectónicas descritas en el capítulo anterior dando respuesta rápida a cada uno de estos eventos, demorando un tiempo aproximado de 2 segundos, obteniendo de esta forma un mayor rendimiento de la aplicación.
- El diseño arquitectónico está previsto para que pueda ser ejecutado tanto en Linux como en Windows sólo sería necesario reutilizar el código de la clase DinamycPluginHandle implementada para Windows y cambiarle la

extensión (*.so para Linux y *.dll para Windows) del archivo que se desea cargar.

- La extensibilidad del sistema demuestra que la necesidad de agregar funcionalidades en tiempo de ejecución al sistema, queda solventada. Se brinda la posibilidad de cargar los módulo que se desean utilizar e incluso tenerlos activos o no, según las necesidades del usuario.
- Cada uno de los plug-in que utiliza la aplicación puede ser reutilizado por otros sistemas con fines similares a Vismedic.

*Seguridad en la Arquitectura propuesta***.**

Los Sistemas de Visualización Médica manejan un conjunto de datos de vital importancia para el diagnóstico perteneciente a los pacientes que son atendidos por el médico. Entre los principios de la Seguridad Informática se encuentra la integridad, es la propiedad que busca mantener los datos libres de modificaciones no autorizadas. La Arquitectura de plug-in propuesta garantiza cumplir con lo que plantea este principio a partir de que un usuario sólo tendrá acceso a los módulos que necesita y a los que puede acceder, avalando que el personal no autorizado no pueda atentar a la seguridad de los datos que se manipulan en el sistema.

*Proceso de Integración*

Como se especificó anteriormente, la arquitectura propuesta es tan genérica que pudiera ser reutilizada por otro sistema que no persiga como objetivo la visualización médica tridimensional, por ejemplo: Laboratorios Virtuales, HMI (módulo de los sistemas SCADA) y Realidad Aumentada, por sólo mencionar tres. El proceso de integración está en dependencia de la arquitectura de software que sustenta el sistema y las características particulares de mismo, pero a través de un diagrama de clases se puede mostrar cómo cualquier aplicación puede reutilizar la propuesta realizada. En el caso específico de Laboratorios Virtuales se refleja de la siguiente forma:

CIGET - MATANZAS
IDICT
INFORMACION Y GESTION TECNOLOGICA
CITMA
ISSN 1029-3450
REVISTA AVANZADA CIENTÍFICA
RNPS 1843

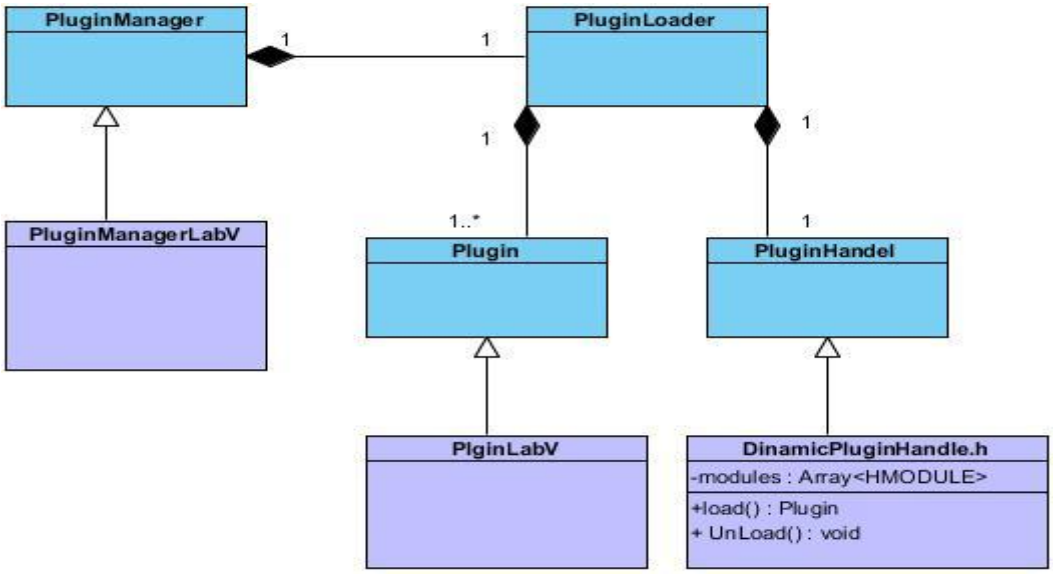


Fig. 4. Diagrama de clases de integración

Solo sería necesario realizar un PluginManager específico para el sistema que desee reutilizar la arquitectura, el mismo hereda las mismas funcionalidades del PluginManager que define la propuesta y el otro paso es definir todos los plug-in que se quieren cargar en la nueva aplicación con sus características específicas (nombre, autor, descripción, etc), en el caso de la clase PluginHandel si se mantiene igual, cambia según el sistema operativo en el cual se quiera ejecutar la aplicación.

## Conclusiones

Al finalizar la investigación y después de haber valorado cada uno de los resultados obtenidos se pudo arribar a las siguientes conclusiones:

1. A partir del estudio realizado se logró definir la arquitectura permitiendo obtener un sistema extensible y configurable, mediante la carga en tiempo de ejecución de las funcionalidades que el usuario, que interactúa con la aplicación, necesita en un momento determinado.
2. La validación de la arquitectura permitió conocer cómo se comportará el sistema ante las acciones del usuario, permitiendo afirmar que la arquitectura propuesta cumple con los requerimientos funcionales y no funcionales que debe poseer el Sistema de Visualización Médica, Vismedic.
3. La idea a defender planteada como propuesta de solución inmediata al problema científico de la investigación, ha quedado validada de manera satisfactoria a través de los resultados de la descripción de la arquitectura y el comportamiento de los atributos de calidad de la solución propuesta, dándole cumplimiento de esta manera al objetivo general de la investigación.

**Referencias bibliográficas**

Arias, R. (2009). *Propuesta de arquitectura e un subsistema de modelado de Diagramas de Flujo de Información.*
Brooks, F. (1975). *The mytical man-month.* Addison-Wesley.
Dayami Ayala Chávez. (2010). *Arquitectura de la plataforma de Transmisión Abierta para Radio y Televisión.* Ciudad Habana.
Direccón Nacional de Servicios Académicos Virtuales. (2011). *Direccón Nacional de Servicios Académicos Virtuales*. Retrieved 1 7, 2011, from Direccón Nacional de Servicios Académicos Virtuales.
Erika Machado, F. C. (2004). *Arquitectura de Software.*
Gutierrez, J. (2066). *Lenguajes y Sistemas Informáticos.*
Johannes Mayer, I. M. (2006). *Ligthweight Plug-in Based Aplication Development.*
Kiccillof, C. R. (2004, 3). *Willidev.* Retrieved 11 11, 2010, from http://www.willydev.net/descargas/prev/Estiloypatron.pdf.
Kruchten. (n.d.). *Arquitectural Blueprints.* Retrieved 2 17, 2011, from http://www.computer.org/portal/web/csdl/doi/10.1109/52.469759.
Lasso, A. (2010). *MVP Scribd.* Retrieved 12 12, 2010, from http://www.scribd.com/doc/210452/Arquitectura-de-Software-Adrian-Lasso
Nokia Corporation. (2010). *QT*. Retrieved 3 21, 2011, from http://doc.qt.nokia.com/4.6/plugins-howto.html.
Perry, D. W. (1992). *Foundations for the study of software architecture.*
Rerynoso, C. (2004). *willydev.* Retrieved 11 12, 2010, from http://www.willydev.net/descargas/prev/Estiloypatron.pdf.
Reynoso, C. B. (2005). *Introducción a la Arquitectura de Software.*
Reynoso, C. B. (2004). *Introducción a la Arquitectura de Software.*
Reynoso, K. N. (2004). *Lenguaje de descripción de Software.*
Salanitri, S. (2009). *ssallanitry-blog.* Retrieved 12 13, 2010, from http://ssalanitri.blogspot.com/2009/07/plugin-pattern.html.
UCI. (2010). *EVA*. Recuperado el 10 de 10 de 2010, de http://eva.uci.cu/mod/resource/view.php?id=14075
Vela, J. A. (2008). *Eventos.* Retrieved 2 10, 2011, from http://www.cimat.mx/Eventos/seminariotecnologias08/javd.pdf-.
Woodwin. (2011). *Woodwin*. Retrieved 4 4, 2011, from http://www.woodwin.nl/es/enterprise-publishing-system/extensibility
Yoan Arlete Carrasco Puebla, E. C. (2009). *Gestiopolis.* Retrieved 4 5, 2001, from http://www.gestiopolis.com/administracion-estrategia/procedimiento-para-la-evolucion-de-las-arquitecturas-de-software.htm.