UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

Título: "Plugin para la integración del Sistema Gestor de Procesos de Media v2.0 con la aplicación Vantage."

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

Autor: Arianny Sánchez Medina

Tutor: Ing. Eliani Varen Caballero

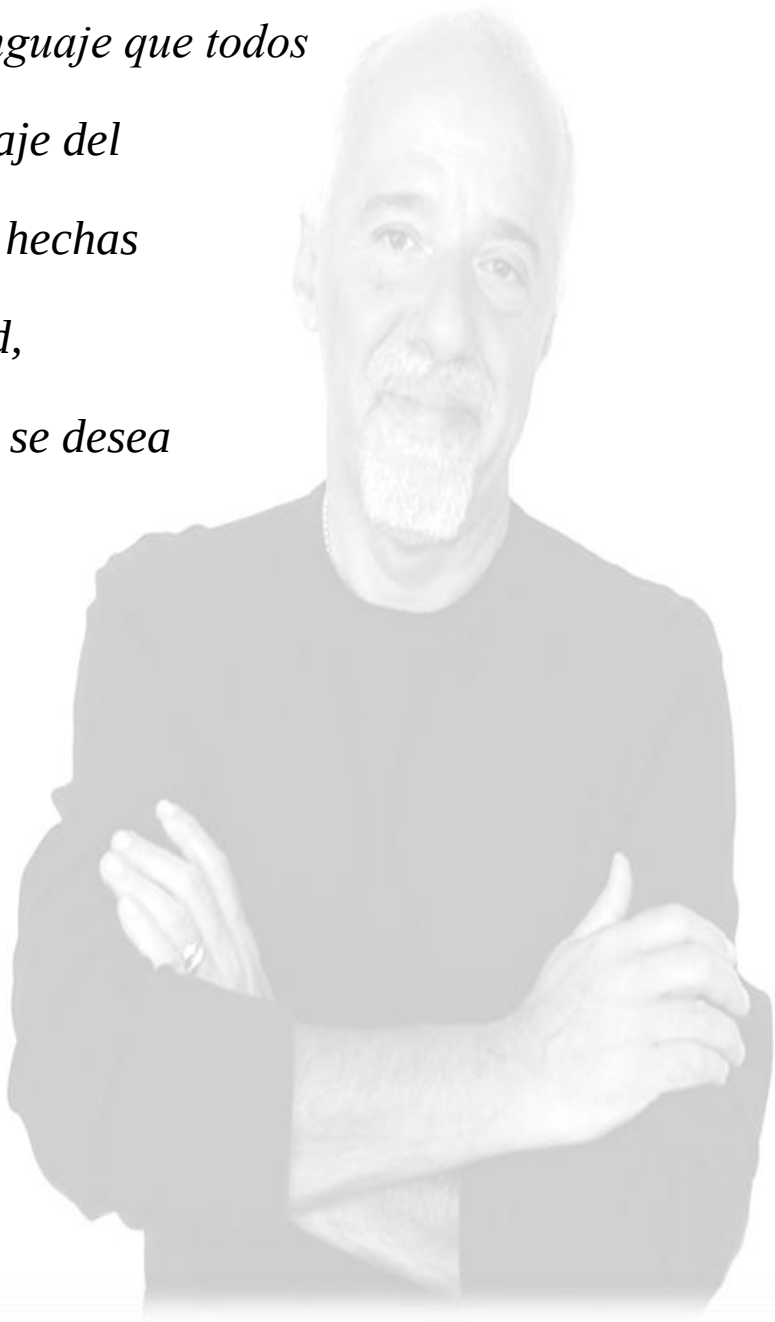
Co-Tutor: Ing. Adnan Fuentes Díaz

La Habana, junio de 2012.

"Año 54 de la Revolución"

"Hay en el mundo un lenguaje que todos comprenden: es el lenguaje del entusiasmo, de las cosas hechas con amor y con voluntad, en busca de aquello que se desea o en lo que se cree."

Paulo Coelho



Dedicatoria

Dedico este triunfo a todas las personas que estuvieron a mi lado durante estos cinco años, principalmente a mis padres Nancy y Juan Alberto por confiar en mí y estar presente en cada momento, por ser los líderes de mi vida, por complacerme tanto como pudieron y llamarme la atención siempre que fue necesario, por quererme y respetar mis decisiones, para ustedes son mis victorias porque merecen eso y mucho más.

Agradecimientos

Agradezco infinitamente a mis padres, por enseñarme a luchar por mis sueños, apoyarme y ser los principales protagonistas en esta aventura que ha sido la vida. Por criarme en el seno de una familia unida y bajo los principios de que nada en la vida importa más que la salud y la felicidad de los que amamos. Mamita, hoy te agradezco cada día en el que me dijiste ¨Mente positiva mi niña, verás que puedes¨, este es mi regalo para ustedes por ser las personas que mas me quieren y que más quiero en el mundo.

A mi hermanita Lary, por ser el regalo de mis padres, por estar a mi lado, incluso cuando discutimos, por apoyarme, por sufrir cuando yo sufro, por compartirlo todo conmigo, te quiero mucho gordi.

A mis abuelitos Mace y Paprimo, por darme la dicha de tener dos padres más que son las personas más tiernas y buenas que he conocido. Por darme la educación y el amor suficiente para ganarse mi cariño, admiración y respeto, gracias por ser tan especiales en mi vida y gracias a Dios por darle vida y permitirles ver cómo me hice una profesional.

A mis tíos Raudel y Yamila, ustedes han sido un factor muy importante en esta etapa, me han apoyado y han estado a mi lado durante toda mi vida, y a mis primitos Roci y Pedri por darme la felicidad de tener dos hermanos mas.

A mi novio Luis y su familia por animarme, mi bebé gracias por existir y por ser la persona que ha estado a mi lado este último año, soportándome en esta etapa tan difícil, por tu amor y entrega, te amo.

A toda mi familia por cada granito que aportaron en mi formación como mujer, mis tíos, mis primos, abuelos. A los que ya no se encuentran y quice mucho, mis abuelos Pello, Petra y Felicia.

A mis vecinos, todos, por su preocupación y cariño.

A mi madrina Yeney, porque aunque estás lejos nunca faltó un consejo y confiaste en mi, a ti también va mi triunfo.

A mis amigas de la universidad Yoarys, Dimelsa y Mahelis, gracias por los buenos ratos, por cada consejo, por tolerarme y estar a mi lado durante mis momentos difíciles, nunca las voy a olvidar. A Yuya, porque durante estos 5 años nos prometimos amistad incondicional a pesar de la distancia, nunca me falló y la conservaré por siempre. A todos y cada uno de mis compañeros de grupo, de apartamento, de edificio, fue especial compartir con ustedes.

Agradecimientos

A mi tutora Eliani por su confianza y mi cotutor Adnan por ayudarme y aconsejarme tanto durante el desarrollo de mi tesis. A Asiel porque sin él hubiese sido mucho más difícil, gracias por tu tiempo. A mi oponente Janet que en los últimos momentos del desarrollo de mi tesis fue un gran apoyo, subió mi autoestima y me demostró que si se podía.

Agradezco en general a todas las personas que han dado su aporte y que han estado al tanto durante todo este tiempo, nunca olvidaré cada detalle, preocupación y ayuda, gracias.

Declaración de autoría

Declaro que soy la única autora de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Arianny Sánchez Medina

_____

Eliani Varen Caballero

_____

Datos de contacto

Tutor: Ing. Eliani Varen Caballero

Formación académica:

Ingeniera en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2010.

Centro Laboral:

Centro de Desarrollo de Geoinformática y Señales Digitales. Facultad 6.

Correo electrónico: evaren@uci.cu

Cotutor: Ing. Adnan Fuentes Díaz

Formación académica:

Ingeniero en Ciencias Informáticas, Universidad de Ciencias Informáticas, 2010.

Centro Laboral:

Centro de Desarrollo de Geoinformática y Señales Digitales. Facultad 6.

Correo electrónico: afuentesd@uci.cu

Resumen

Gracias al constante desarrollo de la sociedad, el intercambio audiovisual se ha convertido en una poderosa herramienta de comunicación utilizada con innumerables fines. Para permitir el soporte de múltiples formatos, las grandes empresas de procesamiento audiovisual se esfuerzan por crear y poseer los sistemas codificadores más avanzados en la industria de transcodificación. El presente trabajo tiene como objetivo la implementación de un plugin para la integración del Sistema Gestor de Procesos de Media con la herramienta transcodificadora de flujo de trabajo de vídeo digital *Vantage*, permitiendo controlar los procesos que se ejecutan en esta aplicación. Con el fin de obtener los mejores resultados el desarrollo se realiza bajo la guía de la metodología RUP, la aplicación es implementada haciéndose uso del lenguaje de programación C++ con el IDE de desarrollo Qt Creator y C# utilizando el entorno de desarrollo Visual Studio. Se presentan los requisitos identificados y el cumplimiento de los mismos en la aplicación final, que es validada además por las pruebas realizadas propuestas por la metodología de desarrollo utilizada. La investigación se centraliza en realizar una integración que soluciona los principales problemas que existen para codificar a formatos estandarizados que son utilizados en la postproducción. Además el sistema resultante logra la trasferencia de archivos en un ambiente multiplataforma.

Palabras claves

Codificación, integración, plugin, procesos, transferencia.

Índice de contenido

Índice de contenido

Índice de tablas

Índice de figuras

INTRODUCCIÓN

En la actualidad, la forma en que los consumidores acceden a los contenidos de video ha evolucionado vertiginosamente. En lugar de usar dispositivos independientes para datos, voz y video, utilizan dispositivos integrados más modernos que proporcionan estos servicios. De esta forma el consumidor puede fácilmente pasar los contenidos audiovisuales de un dispositivo a otro y de una ubicación a otra, de manera rentable y en tiempo real. Para ello, son necesarios dispositivos flexibles capaces de proporcionar funciones tales como la transcodificación, para resolver incompatibilidades en los formatos de compresión, resoluciones de visualización, capacidad de memoria y potencia de procesamiento de los diferentes dispositivos.

A fin de solucionar los problemas relacionados con la compatibilidad de formatos y con el objetivo de brindar servicios de transcodificación, existen empresas que proveen herramientas y sistemas que utilizan las últimas tecnologías para ofrecer una calidad impecable, excelentes velocidades de procesamiento. Además de brindar una industria de soporte de formato de archivo más extenso.

Telestream, el proveedor líder de transcodificación de vídeo y flujo de trabajo ofrece herramientas de flujo de trabajo de video digital y soluciones, ganándose así la confianza de un creciente número de usuarios de una amplia escala de entornos de negocios. Empresas como *SYNCRO Services y TURNER Studios* entre otros medios de comunicación, utilizan Telestream para agilizar sus operaciones, llegar a un público más amplio y generar más ingresos. Esto se logra gracias al completo trabajo que realiza este abastecedor abarcando todo el ciclo de vida completo de medios digitales, incluyendo la captura de vídeo. Ingieren además, en directo y bajo demanda de codificación y transcodificación, reproducción, distribución y transmisión en vivo, así como la gestión y automatización del flujo de trabajo (Telestream, Inc., 2012).

Uno de los productos más completos de este proveedor es la herramienta *Vantage*, la cual brinda una amplia gama de formatos de archivos de entrada y es una solución altamente concentrada que acelera la transcodificación y embalaje para la entrega multipantalla. Además produce la mejor calidad de imagen en el menor tiempo posible y empaqueta los medios de comunicación, junto con imágenes en miniatura, los subtítulos y

metadatos, para la entrega directamente a los servidores de origen o CDN1 (Telestream, Inc., 2012).

Numerosas empresas exitosas como *Brookings, Outdoor Cannel*, y muchas otras que realizan procesamiento de video, poseen servidores de transcodificación de medias que usan *Vantage*.

En Cuba, existen empresas, como el ICRT, que utilizan el software de descarga libre TMPG*Encoder* para convertir videos en sus canales, un ejemplo puede ser el canal Multivisión. Sin embargo, debido a toda la gestión audiovisual que requiere una televisora de este tipo precisan utilizar los sistemas codificadores de videos más avanzados en el mercado internacional.

La Universidad de las Ciencias Informáticas (UCI), como una de las principales líderes en desarrollo de *software* en Cuba, se compone de varios centros de producción. Específicamente el centro GEYSED2 está compuesto por dos departamentos. Uno de ellos es el departamento Señales Digitales, que se encarga de elaborar productos para la gestión de audiovisuales. El referido departamento cuenta con un Sistema Gestor de Procesos de Media (SGPM), el cual surge por la existencia de serios problemas de gestión y planificación de los procesos de medias llevados a cabo en el departamento. Procesos como codificación de audio y video a numerosos formatos, extracción de fotogramas claves así como extracción de logos de materiales audiovisuales, entre otros, que son llevados a cabo por soluciones de *software* que ejecutan bien los mismos.

Este producto está descompuesto en cuatro subsistemas, uno de ellos es la Plataforma de Codificación e Indexación. La misma codifica a formatos como son OGG, AVI, Web M, H.264, RPG2 sin embargo, no codifica a la mayoría de los formatos estandarizados para la postproducción que usan las compañías que se dedican al procesamiento de audiovisuales, formatos como *DV AVI, DVCPro, Avid / Final Cut Pro*, H.264, LXF, MXF, *Omneon*, P2 MXF *Stream*, que son utilizados por empresas como *Supersport, MAGENTA TV, UNC Charlotte,* empresas que además de utilizar todos los formatos anteriormente citados, usan servidores *Vantage*, por lo que se puede llegar a la conclusión de que el

1 CDN significa *Content Delivery Network* (red de entrega de contenidos). Estas redes generalmente consisten en servidores web distribuidos por todo el mundo. Cuando un usuario solicita información de un sitio web, la solicitud es cumplida por el servidor web en la CDN que pueda contestarla más rápidamente.

2 GEYSED: Centro de desarrollo de productos, servicios y soluciones informáticas para el procesamiento de Señales Digitales y la Geoinformática.

SGPM no satisface los requerimientos de muchas empresas, dificultando así la posibilidad de negociar con esta solución en el mercado.

Teniendo en cuenta lo anteriormente planteado se puede definir como **problema a resolver:** ¿Cómo controlar los procesos de codificación y transferencia de media de la herramienta *Vantage* desde el Sistema Gestor de Procesos de Media v2.0? Para dar solución a la problemática planteada el **objeto de estudio** de la presente investigación son los procesos que se realizan en la herramienta *Vantage*. El **objetivo general** lo constituye desarrollar un plugin para la integración del Sistema Gestor de Procesos de Media v2.0 con la aplicación *Vantage*.

Se define como **idea a defender:** El desarrollo de un plugin[3] para la integración del Sistema Gestor de Procesos de Media con la herramienta *Vantage* permitirá la manipulación de los procesos de codificación y transferencia de media de dicha herramienta en consecuencia con las necesidades de este sistema alcanzando así como **campo de acción** los procesos de codificación y transferencia de media que se realizan en la herramienta *Vantage*.

Como guía para la realización de esta investigación se han propuesto las siguientes tareas de la investigación:

- Caracterizar los procesos de codificación y transferencia de media de la herramienta *Vantage*.
- Realizar el diseño ingenieril del plugin de acuerdo a la metodología propuesta.
- Realizar pruebas al plugin para la integración del Sistema Gestor de Procesos de Media v2.0 con la herramienta *Vantage*.

Se espera como resultado de la investigación:

- Informe referente a la investigación realizada.
- Documentación de los artefactos resultantes según la Metodología de Desarrollo de *Software* utilizada.

3 La palabra “plugin” se refiere a partes que se le añaden a un sistema, mediante los cuales este puede funcionar de una forma más eficiente, puede tener nuevas funcionalidades o características, o simplemente mejorar las que ya tenía. (Varen, 2012)

- Plugin para la integración del Sistema Gestor de Procesos de Media v2.0 con la herramienta *Vantage*.

Para obtener los conocimientos necesarios que hagan posible el cumplimiento del objetivo trazado en la investigación, se pondrán en práctica diferentes métodos científicos. Los métodos usados fueron teóricos, estos son:

Analítico–sintético: Este método se utiliza con el objetivo de conocer, a través de documentos encontrados durante la investigación la esencia de todo lo relacionado con los procesos que se realizan en la herramienta *Vantage*.

Hipotético–deductivo: Este método se emplea corrientemente tanto en la vida ordinaria como en la investigación científica. Es el camino lógico para buscar la solución a los problemas planteados. Consiste específicamente en emitir hipótesis acerca de las posibles soluciones al problema planteado (Sánchez, 2011). Se utilizó este método para conocer el funcionamiento de la herramienta *Vantage* y las posibilidades existentes de modelar un plugin para la integración con el SGPM.

La presente investigación científica cuenta con una estructura de 3 capítulos que serán pormenorizados con las siguientes cualidades:

Capítulo #1: Se realiza la Fundamentación Teórica y una caracterización general del objeto de estudio de la investigación y se especifican las herramientas necesarias para el desarrollo de la solución, así como, argumentar la metodología de desarrollo que mejor se ajusta a la propuesta de solución a la cual se desea arribar.

Capítulo #2: Se realiza todo el diseño ingenieril del plugin y la interfaz de comunicación, de acuerdo a la metodología de desarrollo seleccionada.

Capítulo #3: En este capítulo se abordan temas fundamentales para la implementación del plugin, se presentan además las pruebas realizadas al sistema desarrollado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se abordan los principales conceptos asociados al dominio del problema, los cuales tienen como objetivo facilitar un mejor entendimiento de la situación problemática, creando así una base de conocimiento del tema, así como una caracterización del objeto de estudio. Se definen además la metodología, las herramientas y el lenguaje a utilizar en el desarrollo del sistema.

1.1 Conceptos asociados al dominio del problema

Video

La palabra video que proviene del latín *"videre"* y quiere decir "yo veo", hace referencia a la captación, procesamiento, transmisión y reconstrucción de una secuencia de imágenes y sonidos que representan escenas en movimiento. Una señal de video está formada por un grupo de líneas organizadas en cuadros, que son a la vez fraccionados en dos campos para guardar la información relacionada con el color y la luz de la imagen. Por tanto se puede definir el video como la combinación de imágenes y sonidos en un fichero que ofrece una mayor sensación a la vista humana (Wright, 2010).

En el mundo actual el uso del video es muy común en internet, como medio de difusión y promoción de mensajes al público es muy importante. Es hoy un logro significativo, pues con la imagen se captura todo lo que realmente se quiere demostrar. Tiene la ventaja de ser muy descriptivo a la vista del ojo humano, suele ser fundamental a la hora del domino de objetos y el sentir de las personas. Desde su creación por primera vez para los sistemas de televisión se ha hecho muy popular, sobre todo porque el ser humano experimenta mejores sensaciones cuando puede tener una visualización de la información (Wright, 2010).

Codificación

La codificación o transcodificación, es la conversión de un sistema de datos a otro distinto. La codificación de video es la compresión de un video en un formato distinto del original o la conversión con uno u otro códec 4si se tratase del mismo formato (Martínez, 2011).

4 "Códec" significa pares de COdificador (CO)/DEsCodificador (DEC). Un códec no es ni más ni menos que una serie de funciones algorítmicas necesarias para comprimir un archivo, a este proceso de compresión se le denomina "codificación" y descomprimir o decodificar los datos de audio y vídeo.

Transferencia de media

Del latín "*transferens*" RAE define transferencia como pasar o llevar algo desde un lugar a otro (ESPAÑOLA, 2013).

El término transferencia se aplica en el área de la tecnología especialmente a la noción de transferencia de datos. La transferencia de datos es hoy en día uno de los procesos más simples y útiles de realizar con los aparatos electrónicos y muchos de las posibilidades están especialmente adaptados para facilitar la tarea al usuario promedio **(ABC definición, 2007-2013)**.

Cuando se habla de transferencia estamos haciendo referencia a una de las tareas más fáciles que nos permite realizar una computadora con otro aparato electrónico. Esta transferencia es siempre de datos y estos pueden estar representados en diferentes estilos ya sea en material multimedia, textos, o *software* entre otros. De este modo uno puede acceder a diversos tipos de archivos y material en diferentes lugares si se cuenta con los métodos y dispositivos apropiados **(ABC definición, 2007-2013)**.

Se define transferencia al proceso de trasladar una media desde un dispositivo de almacenamiento hasta otro. Las transferencias se originan producto de la necesidad de almacenar la media resultante de un procesamiento.

1.2 Procesos que se ejecutan en la herramienta *Vantage*

Un contenido de video permite brindar gran cantidad de información, por lo que son utilizados en múltiples campos de la sociedad como uno de los principales medios de comunicación. Sin embargo, esta ventaja en ocasiones no se puede aprovechar como realmente se necesita, pues estos contenidos presentan algunos inconvenientes que impiden brindar la información de una manera óptima, presentando problemas de compatibilidad, calidad, etc., por lo que es necesario el procesamiento de los mismos para impedir este tipo de problemas. *Vantage* brinda una gama de procesos que facilitan el trabajo con los archivos de media haciendo el trabajo más agradable.

Para poder trabajar con la herramienta Vantaje es necesario crear un flujo de trabajo, el cual será activado, lo que permitirá enviar tareas y poder supervisarlas desde el plugin que se implementará.

Se debe abrir el grupo de monitor en la barra de acciones, haga clic en la flecha en la parte superior esquina izquierda del grupo para mostrar las acciones asociados.

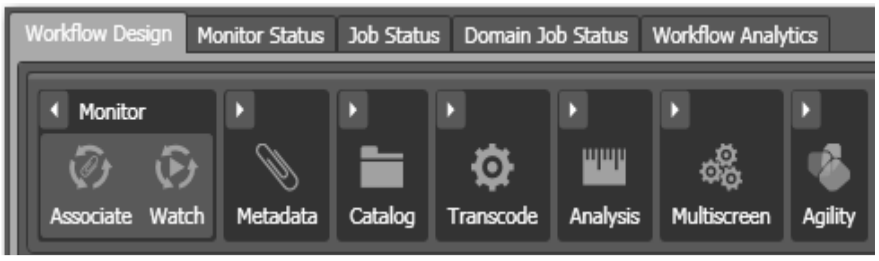


Figura 1. Seleccionar acciones (Telestream, 2012).

Haga clic y arrastre una acción del Monitor hacia el espacio de trabajo para agregar su primera acción al flujo de trabajo. (Tenga en cuenta que se centra automáticamente)

Figura 2. Arrastrar acción *Watch* (Telestream, 2012)**.**

Abra el grupo de acción transcodificación u otro grupo y arrastre la acción hacia el espacio de trabajo. Para crear una cadena de acciones, que los conecta entre sí hay dos maneras:

- Haga clic y arrastre una línea de conexión de una clavija de conexión a otro.

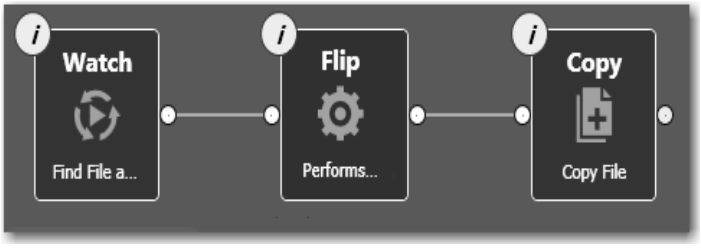


Figura 3. Conectar acciones enlazando clavijas (Telestream, 2012)**.**

- Haga clic y arrastre una acción y chocar con otra acción.

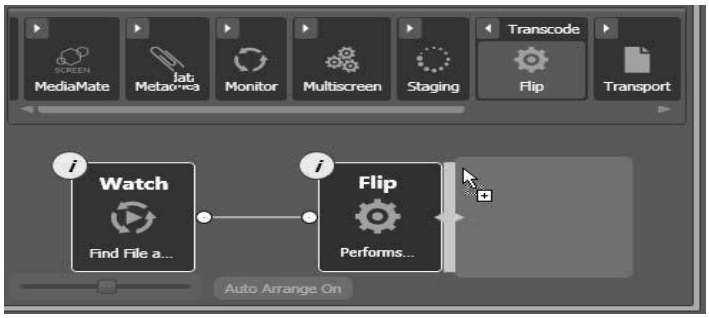


Figura 4. Conectar acciones Chocando acciones (Telestream, 2012)**.**

Cada acción en un flujo de trabajo realiza una función específica y debe estar configurado para realizar esta tarea de la manera deseada para un flujo de trabajo determinado. Cada acción muestra un icono de i (para configurar) de color amarillo en la esquina superior izquierda, el color amarillo indica que aún no se ha configurado (se muestra en azul cuando está configurado).

Una vez que se configuren las acciones queda creado básicamente un flujo de trabajo. Se debe activar en la barra inferior en el botón *Activate* y queda listo para mandar a ejecutar tareas.

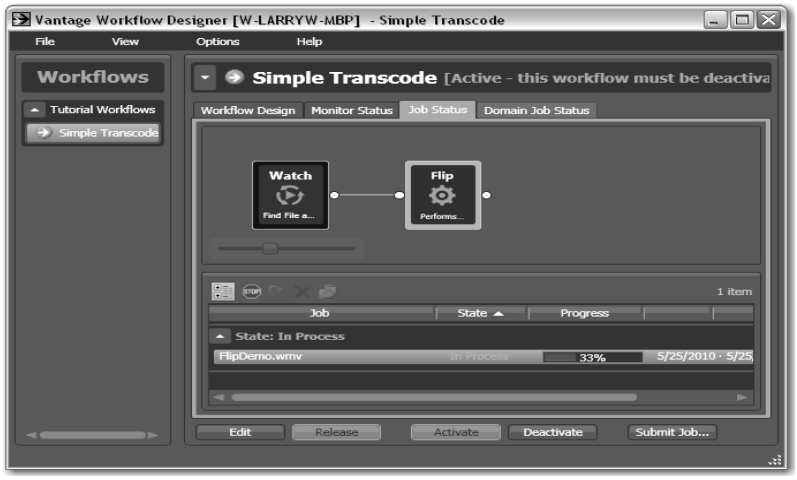


Figura 5. Ejecución de una tarea (Telestream, 2012).

A continuación se muestra un diagrama especificando las acciones que realiza la herramienta *Vantage*.

Figura 6. Procesos que se realizan en la herramienta *Vantage*.

Monitor Actions

Las acciones de este grupo se utilizan para identificar nuevos archivos a procesar, y en el caso de la acción de reloj (una acción origen), iniciar trabajos (Vantage, 2012).

- **Acción Asociado**

Una acción asociado utiliza el servicio de monitor continuamente (y en períodos regulares), para encuestar una ubicación de destino (un directorio, por ejemplo) en un dispositivo o sistema de archivos (FTP, carpeta de red, etc.) para descubrir nuevos archivos.

- **Acción Vea**

Una acción de reloj es una acción de origen especial, que utiliza el servicio de monitorización para el sondeo de una ubicación de destino (un directorio, por ejemplo) en un dispositivo o sistema de archivos (FTP, carpeta de red, etc.) para descubrir nuevos archivos (Vantage, 2012).

Cuando un archivo se descubre, la acción de reloj envía un trabajo al flujo de trabajo, para la tramitación del expediente, por lo general un archivo multimedia (Vantage, 2012).

- **Metadata**

Las acciones de metadatos son una característica opcional, bajo licencia disponible en las ediciones *Vantage*. Se utilizan para convertir entre archivos de datos adjuntos, etiquetas de metadatos y variables (Vantage, 2012).

- **Rellenar Acción**

La acción rellenar se utiliza para transformar datos entre las variables, archivos adjuntos, y las etiquetas de metadatos. Rellenar también se puede utilizar para obtener cierta información del sistema relacionado con el trabajo actual y asigna dicha información a variables (Vantage, 2012).

- **Transformar Acción**

Una acción de transformación transforma los datos entre los archivos XML (archivos adjuntos) y etiquetas. Se utilizan hojas de estilo XSL para realizar estas transformaciones (Vantage, 2012).

Catalog Actions

- **Acción Existe**

La acción existe es usada para determinar si existe un aglutinante del mismo nombre ya existente, lo que indica que un trabajo ya se ha ejecutado en el mismo archivo de medios (Vantage, 2012).

- **Acción Registrar**

La acción de registro registra la carpeta de flujo de trabajo en un catálogo de *Vantage*. Esto permite preservar la carpeta y todos los archivos referenciados por ella, incluso después de que expire un trabajo (Vantage, 2012).

Transcode Actions

En el diseño de flujo de trabajo está el grupo de *transcode*, el cual permite codificar con la acción *Flip* (Vantage, 2012).

Acelera la codificación x264 *Streaming* adaptativo a *Apple, Adobe Dynamic Streaming, Microsoft Smooth Streaming*, y MPEG-DASH y proporciona soporte de decodificación a 38 formatos para archivos de video a través de la acción *Flip* (Vantage, 2012)

Entre los más importantes se encuentran DV AVI, DVC*Pro*, *Avid / Final Cut Pro*, H.264, LXF, MXF, *Omneon*, P2 MXF *Stream* (Vantage, 2012).

Transport Actions

En el diseño de flujo de trabajo está el grupo de transporte, el cual permite transferir archivos con las acciones mover, copiar y eliminar (Vantage, 2012).

- **Copiar**

La acción copiar permite realizar operaciones de copia de archivos en los archivos especificados, identificados por apodos. Copian el duplicado del archivo de origen especificado a otro lugar, dejando el archivo de origen en las acciones place. Esta acción se limita a sistemas que apoyan las operaciones estándar de copia de archivos por ejemplo, Windows, FTP, etc (Vantage, 2012).

- **Mover**

La acción de movimiento permite realizar operaciones de copia de archivos en los archivos especificados, identificados por apodos. La acción de movimiento reproduce el archivo especificado a otro lugar, a continuación, elimina el archivo (Vantage, 2012).

- **Eliminar**

La acción eliminar permite borrar los archivos especificados, identificados por apodos (Vantage, 2012).

- **Implementar**

Permite implementar copias de uno o más archivos a un destino en un solo paso, y puede realizar pasos adicionales personalizados dependiendo del tipo de implementación (Vantage, 2012).

Analysis Actions

Estas acciones son usadas en los flujos de trabajo cuando se quiere realizar mediciones a los archivos de los medios de comunicación, comparar archivos multimedia, o identificar las características de los archivos multimedia y publicar los resultados como variables o etiquetas de metadatos, así como establecer el estado del flujo de trabajo basado en los resultados (Vantage, 2012).

- **Examinar**

Permite seleccionar un analizador específico para obtener ciertos indicadores sobre el contenido de la misma secuencia de medios: Detección Negro, Detección de pizarra. Dependiendo de la métrica especificada, se puede configurar el analizador para satisfacer sus necesidades, y luego publicar las métricas como los metadatos, variables o estados, todos los cuales pueden ser transmitidos a otras acciones en el flujo de trabajo para la utilización apropiada (Vantage, 2012).

El filtro **Detección Negro** analiza el archivo especificado para determinar donde el negro y el silencio existen al inicio, fin y medio del archivo, así como determinar la duración del contenido (Vantage, 2012).

El filtro **Detección de pizarra** (similar a la detección Negro filtro) analiza el archivo especificado para detectar la pizarra principal y barras de color en el archivo especificado, y proporciona un punto de inicio para el contenido que excluye a las barras de pizarra / color (Vantage, 2012).

- **Identificar**

Permite extraer ciertas propiedades de archivos, valores de metadatos y propiedades de archivos multimedia, o crear un hash MD5 del archivo especificado, identificado por apodo. Los valores que se extraen (o crean, en el caso de la hash) se pueden asignar a un variable. También puede establecer el estado del flujo de trabajo, en función de si o no los valores podrían ser extraídos, y en el caso de propiedades de media, audio o vídeo, se detecta (Vantage, 2012).

- **Compare**

La acción Compare es utilizada para examinar dos archivos de Medias y determinar sus diferencias (Vantage, 2012).

Agility Actions

Las acciones del grupo Agilidad se utilizan donde se tiene un sistema de agilidad y desea enviar los trabajos a la misma para el procesamiento. Se pueden utilizar las acciones de agilidad para crear flujos de trabajo, someter y controlar los trabajos de proceso, archivos MMF y trabajos de control durante la ejecución de Agility (Vantage, 2012).

- **Perfil**

Esta acción es utilizada para obtener y mostrar una lista de los perfiles de una agilidad ECS (que debe ser configurado en la consola de administración de *Vantage* y disponible en tiempo de diseño), desde el que se pueden seleccionar, configurar y presentar trabajos (Vantage, 2012).

- **XML**

Esta acción es utilizada para enviar el trabajo descrito en el documento XML especificado para el sistema Agility y ser procesado. Esta es una alternativa basada en *Vantage* y

utiliza la acción Perfil Laboral, y está destinado a ser utilizado principalmente por usuarios de habilidad con implementaciones SDK para generar documentos XML basados en perfiles de trabajo para la ejecución de Agility (Vantage, 2012).

- **Proceso de MMF**

Esta acción es utilizada para procesar un archivo adjunto MMF en la agilidad y generar variables de la misma (Vantage, 2012).

- **Recibir**

Una acción de recepción es una acción de origen especial que está diseñada para aceptar la salida de otro flujo de trabajo. La acción de recepción es una característica opcional, con licencia para su uso con SDK y aplicaciones basadas en flujo de trabajo de aplicaciones de portal (Vantage, 2012).

- **Acción Sincronizar**

La acción Sincronizar se utiliza en flujos de trabajo en múltiples acciones para conectar múltiples acciones posteriores (Vantage, 2012).

Staging Actions

Las acciones en el servicio de clasificación por etapas son responsables de las operaciones de archivos como preparación para mover archivos a otros sistemas con los requisitos de embalaje especiales (Vantage, 2012).

- **Acción Archivo**

La acción Archivo es usada para convertir el archivo de entrada seleccionado a otro formato, por lo general, para copiar, mover, o implementar un archivo en un sistema determinado que tiene requisitos especiales de formato de archivo (Vantage, 2012).

- **Acción Reúna**

Mediante esta acción se reúnen para recoger uno o más archivos desde un servidor de directorio especificado y llevarlos al flujo de trabajo como archivos adjuntos. El uso de un patrón de coincidencia de archivo le permite seleccionar ciertos tipos de archivos (Vantage, 2012).

Communicate Actions

Comunicar acciones se utilizan para comunicarse con sistemas externos (Vantage, 2012).

- **Acción Mensaje**

Una acción de mensajes es una acción que le permite generar y transmitir un mensaje de correo electrónico, un sistema electrónico, por ejemplo. Debe configurar *Vantage* para usar un servidor SMTP de correo electrónico antes de que pueda ser utilizado, lo que se puede hacer en la consola de administración de *Vantage* (Vantage, 2012).

- **Acción Notificar**

Una acción Notificar es una acción que guarda la información de trabajo a un archivo XML, invoca a un grupo de comandos y pasa los datos, o interfaces con un sistema externo a través de *Web Services* (Vantage, 2012).

1.3 Situación Problemática

En la UCI, específicamente en el centro GEYSED, se encuentra el SGPM, en el cual se ejecutan varios flujos de procesamiento de audio y video. Sin embargo debe permitir flexibilidad en la incorporación de nuevas funcionalidades con el fin de mantenerse constantemente en la competencia ya que uno de sus objetivos específicos es lograr integración con disímiles sistemas y *hardware* especializados para la producción audiovisual. A pesar de que hoy en día el SGPM se está utilizando como una solución de *software* orientada a empresas de televisión, no está aun al nivel de lo que buscan las televisoras de Cuba, debido al interés que tienen este tipo de instituciones en obtener *software* avanzados y con una tecnología capaz de agilizar sus procesos por lo que una solución primaria consiste en realizar una integración del SGPM con la herramienta *Vantage.*

Esta solución, siendo uno de los productos más desarrollados del Proveedor Telestream y a diferencia del SGPM, ofrece una forma potente y sencilla de construir y automatizar los flujos de trabajo de vídeo. Ofrece además transcodificación, la captura de los medios de comunicación, procesamiento de metadatos y análisis en un sistema unificado. Es un sistema flexible y escalable al tiempo que ofrece resultados visibles, predecibles y confiables.

Brinda soluciones factibles, con las características que cualquier competencia busca, siendo, la integración del SGPM con esta herramienta una solución factible para el departamento Señales Digitales.

1.4 Tecnologías y herramientas de desarrollo

Para la implementación del plugin y la interfaz de comunicación, es necesario realizar un estudio que abarque las tecnologías y herramientas a utilizar para cumplir el objetivo planteado en la investigación.

1.4.1 Metodología de desarrollo

Las Metodologías de Desarrollo de *Software* surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto *software*. Dichas metodologías pretenden guiar a los desarrolladores al crear un nuevo *software*, pero los requisitos de un *software* a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación del software (Carrillo, 2008).

Proceso Unificado Racional (RUP por sus siglas en inglés)

Lo que sustenta el proceso de desarrollo de *software* en esta metodología son: el proyecto, las personas, el producto y el proceso, existe una estrecha relación entre ellas. Es conocido como las cuatro P en el desarrollo del *software*. RUP define para cada etapa: el flujo de trabajo, los trabajadores que intervienen, las actividades que realizan y los artefactos que se necesitan o producen. Su meta es asegurar la producción de *software* con la más alta calidad, que cumpla con las necesidades de los usuarios dentro del cronograma planeado y la inversión prevista. El proceso unificado está basado en componentes, lo cual quiere decir que el sistema *software* en construcción está formado por componentes *software* interconectados a través de interfaces bien definidas. Además, el proceso unificado utiliza el Lenguaje Unificado de Modelado (UML por sus siglas en inglés) para expresar gráficamente todos los esquemas de un sistema *software* (Guerrero, 2009).

La metodología de desarrollo RUP se divide en cuatro fases, desglosadas de la siguiente manera:

- Inicio: El objetivo de esta etapa es determinar la visión del proyecto.
- Elaboración: El objetivo es determinar la estructura óptima.
- Construcción: El objetivo es obtener la capacidad operacional inicial.
- Transición: El objetivo es obtener la primera versión del proyecto (*release*).

RUP unifica los mejores elementos de otras metodologías, lo que garantiza que sea una metodología de primer nivel; una de las razones fundamentales es que está preparada para desarrollar proyectos grandes y complejos, además de estar definido por tres aspectos esenciales:

- Dirigido por casos de uso: Basándose en los casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que los llevan a cabo. Además, estos modelos se validan para que sean conformes a los casos de uso. Finalmente, los casos de uso también sirven para realizar las pruebas sobre los componentes desarrollados (Guerrero, 2009).
- Centrado en la arquitectura: En la arquitectura de la construcción, antes de construir un edificio éste se contempla desde varios puntos de vista: estructura, conducciones eléctricas, fontanería, etc. Cada uno de estos aspectos está
- Representado por un gráfico con su notación correspondiente. Siguiendo este ejemplo, el concepto de arquitectura *software* incluye los aspectos estáticos y dinámicos más significativos del sistema (Guerrero, 2009).
- Iterativo e incremental: Todo sistema informático complejo supone un gran esfuerzo que puede durar desde varios meses hasta años. Por lo tanto, lo más práctico es dividir un proyecto en varias fases. Actualmente se suele hablar de ciclos de vida en los que se realizan varios recorridos por todas las fases. Cada recorrido por las fases se denomina iteración en el proyecto en la que se realizan varios tipos de trabajo (denominados flujos). Además, cada iteración parte de la anterior incrementado o revisando la funcionalidad implementada (Guerrero, 2009).

Después de realizar este análisis se llega a la conclusión de que RUP es muy utilizada para la realizacion y documentación de *software* orientado a objetos. El equipo de desarrollo cuenta con experiencia en el uso de esta metodología, ademas propone varias iteraciones lo que viabiliza la corrección de errores y mejoras del *software* permitiendo a los desarrolladores poder realizar un producto con mayor calidad.

Con esta metodología el sistema se va desarrollando y documentando al mismo tiempo, esto es importante ya que, si algún miembro del equipo no puede seguir con el proceso, quien ocupe su lugar podrá guiarse y conocer qué fue lo que se realizó hasta el momento. RUP permite construir un *software* de alta calidad y permite que se trabaje de forma

organizada donde se controle y documente todo lo relacionado con el proyecto, descartando los posibles riesgos que se presentan en el mismo, lo cual no podría lograrse sin el empleo de una metodología que se adapte a las características propias del *software* que se esté desarrollando.

1.4.2 Herramienta de modelado de *software*

Se puede definir a las herramientas CASE (*Computer-Aided Software Engineering*: Ingeniería de *Software* Asistida por Ordenador) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de *software* y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software* ().

Tienen como objetivo:

- Permitir la aplicación práctica de metodologías estructuradas.
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
- Simplificar el mantenimiento de los programas.
- Mejorar y estandarizar la documentación.
- Aumentar la portabilidad de las aplicaciones.
- Facilitar la reutilización de componentes del *software*.
- Permitir un desarrollo y un refinamiento visual de las aplicaciones.

Visual Paradigm 8.0

Visual Paradigm para UML es una potente plataforma de desarrollo visual compatible con el ciclo completo de una aplicación y combina el modelado con excelentes herramientas para generación de códigos, ingeniería inversa y la interoperabilidad con otras aplicaciones.

Es compatible con las notaciones de UML más recientes para el modelado y proporciona una interfaz intuitiva centrada que se integra perfectamente con aplicaciones como *Eclipse/WebSphere, JBuilder, NetBeans/Sun ONE, IntelliJ IDEA, JDeveloper y WebLogic Workshop* ofreciendo una sincronización sofisticada y en tiempo real de los códigos y modelos. Además, el código de ingeniería inversa es compatible con Java, C++, Net DLL o EXE, *Corba* IDL, XML *Schema, Hibernate* y JDBC. Asimismo, es compatible con el ciclo de vida completo de la aplicación, directamente orientado al desarrollo y la construcción de objetos (Softpedia, 2001).

Se ha seleccionado Visual Paradigm como herramienta CASE para el desarrollo de la solución propuesta por las razones siguientes:

Es multiplataforma y posibilita la generación de código y documentación. Permite dibujar 13 tipos de diagramas diferentes a través de un intuitivo modelado visual y la aplicación de ingeniería inversa. Admite la importación y exportación de XML e imágenes, la administración de requerimientos, la creación de esquemas de clases a partir de una base de datos y viceversa, además cuenta con un soporte que facilita el trabajo simultáneo sobre un mismo diagrama entre dos desarrolladores.

1.4.3 Lenguaje de modelado de *software* (UML por sus siglas en inglés)

Tal como su nombre lo indica, UML es un lenguaje de modelado y no un método o un proceso. El UML está compuesto por una notación muy específica y por las reglas semánticas relacionadas para la construcción de sistemas de *software*. El UML en sí mismo no prescribe ni aconseja cómo usar esta notación en el proceso de desarrollo o como parte de una metodología de diseño orientada a objetos (Sparks, 2012).

El UML soporta un conjunto rico en elementos de notación gráficos. Describe la notación para clases, componentes, nodos, actividades, flujos de trabajo, casos de uso, objetos, estados y cómo modelar la relación entre esos elementos. El UML también soporta la idea de extensiones personalizadas a través elementos estereotipados (Sparks, 2012).

El UML provee beneficios significativos para los ingenieros de *software* y las organizaciones al ayudarles a construir modelos rigurosos, trazables y mantenibles, que soporten el ciclo de vida de desarrollo de *software* completo (Sparks, 2012).

Se ha seleccionado UML como soporte de la modelación de la solución propuesta ya que las ventajas que posibilita su utilización son varias, por un lado el uso de lenguajes visuales facilita su asimilación y entendimiento por parte del equipo de trabajo; el tiempo invertido en el desarrollo de la arquitectura se minimiza; la detección y resolución de errores se agiliza siempre y cuando se haga uso de herramientas adecuadas de diagnóstico y depuración; y la trazabilidad y documentación del proyecto se confecciona de una forma ordenada y guiada por los casos de uso.

1.4.4 Lenguaje de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras.

Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

Lenguaje de programamación C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su relación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multi paradigma (Groningen, 2012).

Se escoge el lenguaje de programación C++ para el desarrollo del componente porque es un lenguaje que va a permitir la comunicación con sistemas externos, la cual es una funcionalidad vital que debe cumplir el componente. Es muy potente lo que facilitará el trabajo con las imágenes y garantizará la rapidez de respuesta del *software*. Puede además ser ejecutado en múltiples plataformas, es multiparadigma y brinda una efectiva gestión de excepciones.

Lenguaje de programación C#

La sintaxis de C# simplifica muchas de las complejidades de C++ y, a la vez, ofrece funciones eficaces tales como tipos de valores que aceptan valores NULL, enumeraciones, delegados, métodos anónimos y acceso directo a memoria, que no se encuentran en Java. C# también admite métodos y tipos genéricos, que proporcionan mayor rendimiento y seguridad de tipos, e iteradores, que permiten a los implementadores de clases de colección definir comportamientos de iteración personalizados que el código de cliente puede utilizar fácilmente. (Microsoft, 2012)

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. En C#, una estructura es como una clase sencilla; es un tipo asignado en la pila que puede implementar interfaces pero que no admite la herencia.

Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de *software* a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen (Microsoft, 2012):

- Firmas de métodos encapsulados denominadas delegados, que permiten notificaciones de eventos con seguridad de tipos.
- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- Comentarios en línea de documentación XML.

Se selecciona C# para el desarrollo del puente de aplicación porque es el lenguaje en el que está desarrollada la API de la herramienta *Vantage*. Además C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores crear una amplia gama de aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Puede utilizar este lenguaje para crear aplicaciones cliente para Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y muchas tareas más.

1.4.5 Entorno de desarrollo

Un entorno de desarrollo integrado (IDE por sus siglas en inglés) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.

Hoy en día los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C++, C#,

Java, Python y Visual Basic entre otros). Además es posible que un mismo entorno de desarrollo tenga la posibilidad de utilizar varios lenguajes de programación, como es el caso de Eclipse (E.U de Ingeniería Técnica Informática de Oviedo, 2012).

Qt Creator 4.8

Qt Creator ha sido desarrollado para ser un entorno de desarrollo integrado (IDE por sus siglas en inglés) multiplataforma adaptado a las necesidades de los desarrolladores de Qt. Se ejecuta en los sistemas operativos de escritorio *Windows, Linux/X*11 y *Mac OS X* y permite a los desarrolladores crear aplicaciones para múltiples escritorios y plataformas de dispositivos móviles.

Proporciona (Softpedia, 2001)**:**

- Un editor de código de C++ y JavaScript.
- Un diseñador de interfaz de usuario integrado.
- Herramientas de gestión de versiones y proyectos.
- Depuradores gdb y CDB.
- Control de versiones.
- Un simulador para las interfaces de usuario móviles.
- Compatibilidad con objetivos de escritorio y portátiles.

Se escoge este IDE para el desarrollo del componente porque es multiplataforma, es muy eficiente para desarrollar aplicaciones en C++ de manera sencilla y rápida, posee un avanzado editor de código C++ y también una interfaz gráfica de usuario integrada y diseñador de formularios, un depurador visual y auto-completado de código. Permite además que la combinación del lenguaje de programación y la librería para el tratamiento de las imágenes escogidos se pueda efectuar de una manera eficiente, logrando con esto que el componente tenga la calidad requerida y que solucione los problemas existentes en el departamento de Señales Digitales.

Visual Studio 2012

Visual Studio es un conjunto de herramientas de desarrollo basadas en componentes y otras tecnologías para compilar aplicaciones eficaces de alto rendimiento. Además, Visual Studio está optimizado para el diseño, el desarrollo y la implementación en equipo de soluciones empresariales.

Se selecciona Visual Studio como entorno de desarrollo porque es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML (Microsoft, 2013).

1.5 Conclusiones parciales

En el presente capítulo se detallaron los principales conceptos manejados en la investigación, realizándose además un estudio de los procesos fundamentales que realiza la herramienta *Vantage*, así como un análisis de las tecnologías que se utilizarán para el desarrollo del componente, por lo cual se puede concluir:

- Se definieron los principales conceptos asociados al procesamiento de audio y video, y a los principales procesos que se estudiarán en la herramienta *Vantage*, permitiendo así un mayor entendimiento de los términos empleados a lo largo de la investigación.
- Se realizó un estudio generalizado de los procesos que realiza la herramienta *Vantage* lo que permitió un mayor dominio de dicha herramienta facilitando a su vez el desarrollo de la investigación.
- Se seleccionaron las tecnologías a utilizar para la implementación del componente, demostrando que son tecnologías de fácil manejo y aptas para obtener el desarrollo deseado.

CAPÍTULO 2: PRESENTACIÓN DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se recogen los principales aspectos de la modelación del negocio enunciándose las reglas que lo rigen así como los trabajadores y actores de los mismos. También se definen los requerimientos funcionales y no funcionales del sistema y se muestra el diagrama de casos de uso del sistema con la descripción de los casos de uso del sistema que se consideran críticos.

2.1 Modelo del dominio

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las “cosas” que existen o los eventos que suceden en el entorno en el que trabaja el sistema (Jacobson, 2000).

El modelo del dominio se describe mediante diagramas de UML (especialmente mediante diagramas de clases). Estos diagramas muestran a los usuarios, clientes, revisores y a otros desarrolladores las clases del dominio y cómo se relacionan unas con otras mediante asociaciones (Jacobson, 2000).

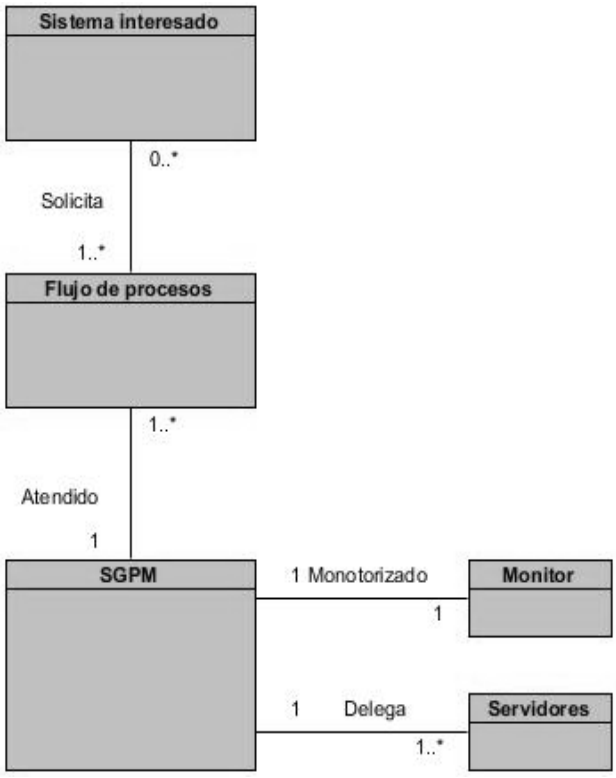


Figura 7. Diagrama de clases del dominio.

Descripción de las clases del dominio

Sistema interesado: Es un sistema, el cual solicita el un flujo de procesos.

Flujo de procesos: Procesos que se mandan a ejecutar. Estos pueden ser procesos de codificación o transferencia de archivos.

SGPM: Es el Sistema Gestor de Procesos de Media donde se gestionan todos los procesos que son enviados a ejecutar por los sistemas externos interesados.

Monitor: El monitor es donde se monotorizan los procesos, además se pueden tomar acciones de gestión sobre los mismos.

2.2 Identificación de los requisitos

Los requisitos son capacidades y condiciones con las cuales debe ser conforme el sistema y más ampliamente, el proyecto. El primer reto del trabajo de los requisitos es encontrar, comunicar y recordar (que normalmente significa registrar) lo que se necesita realmente, de manera que tenga un significado claro para el cliente y los miembros del equipo de desarrollo.

2.2.1 Requisitos funcionales

Los requisitos funcionales son el contrato que se debe cumplir, de modo que los usuarios finales tienen que comprender y aceptar los requisitos que son especificados y aprobados en un principio. Además la definición de los requisitos funcionales permite proveer a los desarrolladores un mejor entendimiento de lo que debe cumplir el sistema.

RF 1. Codificar video.

Permite al sistema realizar la codificación de un video a uno o varios formatos de los que se presentan a continuación:

- Codificar a formato DV AVI: Permite al sistema codificar un video al formato DV AVI.
- Codificar a formato DVCPro: Permite al sistema codificar un video al formato DVCPro.
- Codificar a formato Avid / Final Cut Pro: Permite al sistema codificar un video al formato Avid / Final Cut Pro.
- Codificar a formato H.264: Permite al sistema codificar un video al formato H.264.

- Codificar a formato LXF: Permite al sistema codificar un video al formato LXF.
- Codificar a formato MXF: Permite al sistema codificar un video al formato MXF.
- Codificar a formato Omneon: Permite al sistema codificar un video al formato Omneon.
- Codificar a formato P2 MXF Stream: Permite al sistema codificar un video al formato P2 MXF Stream.

RF 2. Mover video: Permite al sistema mover un video hacia una ruta especificada.

RF 3. Copiar video: Permite al sistema copiar un video hacia una ruta especificada.

RF 4. Eliminar video: Permite al sistema eliminar un video que se encuentra en una ruta especificada.

RF 5. Cancelar trabajo: Permite al sistema detener un proceso una vez que este se encuentre en ejecución.

2.2.2 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Usabilidad

- El plugin debe incluir el puente para la comunicación entre el mismo y el SGPM.
- El plugin debe ser manejado por técnicos medio en informática o graduados universitarios con especialidades afines a esta.

Finalidad

- El plugin tiene como objetivo permitir al SGPM controlar los procesos de codificación y transferencia de medias en la herramienta *Vantage*.

Requisitos de *Software*

Plugin:

- Protocolo de llamada a procedimiento remoto XML RPC.

Vantage:

- NET Framework 3.5 SP1.

- QuickTime 7.6.9.
- En Windows XP o Server 2003, Windows Installer 4.5 debe estar instalado.
- En Windows 2008 R2 o R1, Experiencia de escritorio debe estar habilitado.

Requisitos de *Hardware*

- Procesador: 2 QuadCore.
- Memoria RAM: 4 GB como mínimo.

Requisito de Fiabilidad

- Al reanudarse las operaciones luego de interrumpir la energía o las comunicaciones de red en el Sistema Gestor de Procesos de Media, se deben actualizar los datos de los flujos de trabajo que se ejecutan.
- El plugin no debe estar inactivo en ningún momento, pues los flujos de trabajo a ejecutar pueden llegar en cualquier momento.

Eficiencia

- El tiempo de respuesta por transacción no debe exceder los 5 segundos.

Restricciones de diseño

- El plugin será implementado siguiendo el paradigma de la Programación Orientada a Objetos.
- Para la modelación del sistema se utilizará como lenguaje de modelado UML.
- Se requiere el uso del estilo arquitectónico Cliente-Servidor.

2.3 Diagrama de Casos de Uso del Sistema

Un diagrama de casos de uso es una excelente representación del contexto del sistema; conforma un buen diagrama de contexto, muestra los límites de un sistema, lo que permanece fuera de él, y cómo se utiliza. Sirve como herramienta de comunicación que resume el comportamiento de un sistema y sus actores (Larman, 2003).

En la siguiente figura se muestra el diagrama de casos de uso.

CAPÍTULO 2: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

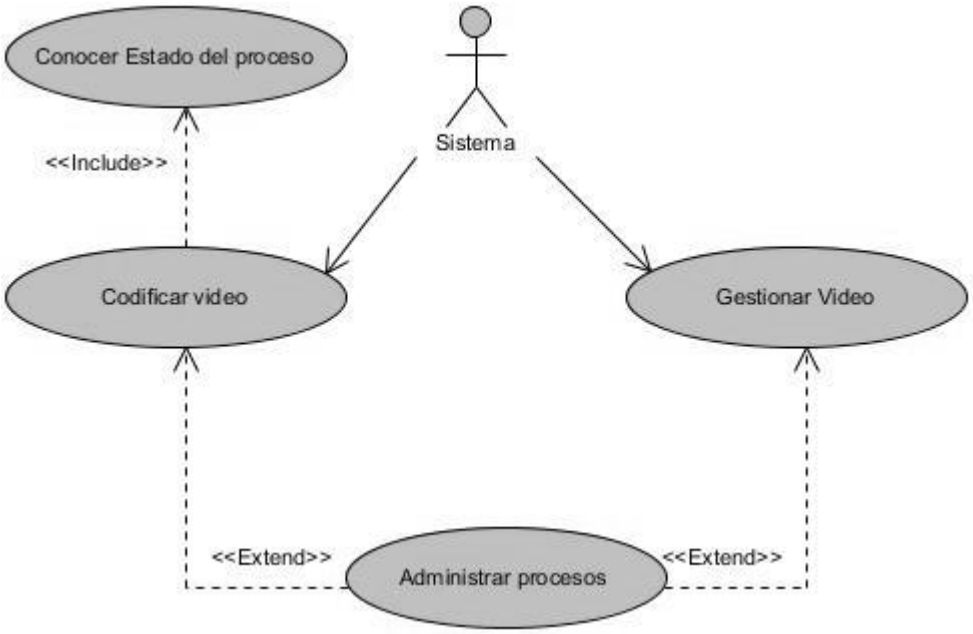


Figura 8. Diagrama de Casos de Uso del Sistema.

Descripción de los actores del sistema

Constituyen actores del sistema los terceros que interactúan con el mismo y no forman parte de él. Durante el desarrollo de la aplicación se definieron una serie de actores, los mismos se describen a continuación en la Tabla 1:

Tabla 1. Descripción de los actores del sistema.

Actor	Objetivo
Gestor_Procesos_Media	Es el encargado de delegar un flujo de procesos al servidor de tipo

A continuación se brinda un resumen de los casos de usos identificados para el sistema.

Tabla 2. Descripción del CU Gestionar procesos.

Caso de Uso:	Administrar procesos.
Actores:	Gestor_Procesos_Media
Resumen:	El caso de uso inicia cuando el Gestor desea administrar un proceso y puede conocer el estado de ejecución de determinado proceso o

	cancelar un proceso que se esté ejecutando. El caso de uso finaliza cuando se administra correctamente uno o varios procesos.
Complejidad	Media
Precondiciones:	Debe existir un proceso en ejecución.
Postcondiciones	Se administró correctamente uno o varios procesos.
Referencias	RF1, RF5
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Gestor manda a cancelar un proceso al servidor de *Vantage* disponible.	2. Cancela el proceso que está en ejecución.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
3. Manda a cancelar un proceso con id erróneo o que no existe.	4. Notifica error ocurrido. Volver al flujo 1 de la sección “Administrar procesos”.
Poscondiciones	

Tabla 3. Descripción del CU Codificar video.

Caso de Uso:	Codificar video
Actores:	Gestor_Procesos_Media
Resumen:	El caso de uso inicia cuando el Gestor delega un flujo de procesos a un servidor de tipo *Vantage* disponible con un flujo de trabajo creado, disponible con el formato al que se va a codificar. El caso de uso finaliza cuando la herramienta *Vantage* comienza a codificar.

Complejidad	Media
Precondiciones:	Debe existir un servidor disponible.
Postcondiciones	Se codificó exitosamente uno o varios videos.
Referencias	RF1
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Gestor delega un flujo de procesos al servidor disponible.	2. Manda a codificar a la herramienta *Vantage* de acuerdo a los parámetros de entrada recibidos.
	3. Codifica el video exitosamente. Finaliza el CU.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
4. Delega un proceso con parámetros erróneos.	5. Notifica error ocurrido en el proceso. Volver al flujo 1 de la sección “Codificar video”.
Poscondiciones	

2.4 Patrones y Estilos de Diseño de *Software*

Un patrón es una descripción de un problema y la solución, a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias, y considera los puntos fuertes y compromisos (Larman, 2003).

“Un patrón es una solución de diseño de *software* a un problema, aceptada como correcta, a la que se ha dado un nombre y que puede ser aplicada en otros contextos”. Un

patrón captura la experiencia y conocimiento de expertos, quienes han producido soluciones exitosas a problemas, a fin que esas soluciones queden a disposición de personas con menos experiencia; sin embargo, los patrones no proveen siempre las soluciones definitivas, algunas veces, los usuarios de patrones deben tener creatividad para utilizar, instanciar o implementar un patrón (Sosa, 2012).

2.4.1 Patrones de Arquitectura

Es necesario saber cuándo emplear el término Patrón Arquitectónico o Estilos Arquitectónicos. Los estilos sólo se manifiestan en arquitectura teórica descriptiva de alto nivel de abstracción; los patrones, por todas partes. Los partidarios de los estilos se definen desde el inicio como arquitectos; los que se agrupan en torno de los patrones se confunden a veces con ingenieros y diseñadores, cuando no con programadores con conciencia sistemática o lo que alguna vez se llamó analistas de *software* (Reynoso, 2005).

Arquitectura en componentes

“Un componente de *software*, es una unidad de composición con interfaces especificadas contractualmente y dependencias del contexto explícitas.” Que sea una unidad de composición y no de construcción quiere decir que no es preciso confeccionarla: se puede comprar hecha, o se puede producir en casa para que otras aplicaciones de la empresa la utilicen en sus propias composiciones (Bonillo, 2006).

En esencia, un componente es una pieza de código pre elaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Los componentes son los "ingredientes de las aplicaciones", que se juntan y combinan para llevar a cabo una tarea. Las arquitecturas basadas en componentes consisten en una rama de la Ingeniería de *Software* en la cual se trata con énfasis la descomposición del *software* en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más grandes de *software*. La estructura de la arquitectura basada en componentes contempla tres partes:

El nombre de los componentes: el nombre de un componente debe ser la identificación de la funcionalidad y uso que tiene como *software*. Generalmente, los desarrolladores usan algún tipo de convención que facilite la identificación de componentes, especialmente, cuando se trabaja en proyectos de gran envergadura (Sosa, 2012).

La interface de los componentes: es el área de intercambio (*input-output*) entre el interior y el exterior de un componente de *software*. La interface es quien permite acceder a los datos y funcionalidades que estén especificadas en el interior del componente (acceder funcionalmente, no a su especificación). Adicional a la interface se encuentra la documentación que muestra la información sobre cómo utilizar un componente (Sosa, 2012).

Cuerpo y código de implementación: es la parte del componente que provee la forma (implementación) sobre la cual un fragmento del componente realiza sus servicios y funcionalidades. Este es el área que debe cumplir con el principio de encapsulación.

La encapsulación se refiere a la especificación interna oculta o no investigable a través de la interface. Así se protege que el resto de los componentes y piezas de *software* dentro de un sistema, no se vean afectados por cambios en el diseño de uno de los componentes. Las arquitecturas basadas en componentes están encaminadas a la reutilización de código mediante un diseño centrado en interfaces y componentes; en el desarrollo de los sistemas de *software* que siguen este estilo, una vez definidas las interfaces de comunicación de los componentes, se pueden realizar las actualizaciones e incorporación de funcionalidades de manera fácil, pues los componentes funcionan como cajas negras de las que solo se necesita saber la funcionalidad que realizan. Para efectuar cualquier cambio en el modo en su funcionamiento solo es necesario reescribir el código interno de la implementación (Sosa, 2012).

Arquitectura Cliente Servidor

Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio) En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras (Puebla, 2013).

Las características básicas de una arquitectura Cliente/Servidor son:

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la

interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de *software* que maneja recursos compartidos tales como bases de datos, impresoras y módems (Puebla, 2013).

- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco y input-output devices (Puebla, 2013).
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red (Puebla, 2013).
- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores (Puebla, 2013).
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos (Puebla, 2013).
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes. No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio (Puebla, 2013).
- El ambiente es heterogéneo. La plataforma de *hardware* y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas (Puebla, 2013).
- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente/Servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores (Puebla, 2013).

2.4.2 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de *software*. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de *software* que están sujetos a contextos similares (Sosa, 2012).

Patrones Generales de *Software* para Asignar Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a clases, expresados en forma de patrones. A continuación se describen los utilizados en la investigación:

Bajo acoplamiento: Plantea asignar las responsabilidades de modo que se mantenga bajo acoplamiento. Soluciona el problema siguiente: ¿Cómo dar soporte a poca dependencia y a una mayor reutilización?

El acoplamiento se evidencia en la clase Servicios, clase que manda a ejecutar los procesos de la herramienta *Vantage* ya que este patrón no es más que la medida de la fuerza con que una clase está conectada a otras clases, además esta clase no depende de muchas otras.

Alta Cohesión: Plantea asignar las responsabilidades de modo que se mantenga una alta cohesión. Soluciona el problema siguiente: ¿Cómo mantener controlable la complejidad? La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza una clase con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.

El patrón alta cohesión se puede encontrar en la clase Servicios, clase que manda a ejecutar los procesos de la herramienta *Vantage*. Este patrón define que una clase tiene responsabilidades moderadas en un área funcional moderada con las otras para llevar a cabo las tareas. Al utilizar el patrón Alta Cohesión se logra un diseño con mayor claridad, mejoras en las funcionalidades y una mayor reutilización.

Controlador: Se utiliza este patrón en la clase ControladorLogicasReceptorPeticiones porque está diseñado para asignar las responsabilidades de controlar un flujo de eventos del sistema.

CAPÍTULO 2: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

El patrón Controlador sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

Creador: Este patrón se evidencia en la clase Nodo_Plataforma_Vantage del plugin Plugin_Controlador, pues guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. La intención principal de este patrón es encontrar un creador que se conecte con el objeto producido en cualquier evento. Ofrece soporte de bajo acoplamiento, lo que provee mayor reutilización.

Experto: Es el principio de que una clase debe poseer la información necesaria para cumplir con sus responsabilidades. Se puede observar en la clase PeticionSoap del plugin Plugin_Controlador. Consiste en asignarle las responsabilidades a una clase que dispone de la información necesaria para llevar a cabo una tarea. Soporta un bajo acoplamiento, que contribuye a tener un sistema robusto y de fácil mantenimiento.

Patrones GOF (*Gang of Four*)

Los patrones GOF, son una serie de posibles soluciones a problemas que suelen ser comunes cuando se desarrolla un *software*. Existen tres tipos fundamentales de patrones:

- Patrones Creacionales: Inicialización y configuración de objetos.
- Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
- Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos (Tedeschi, 2012).

Observador: Incluido en el grupo de los Patrones de Comportamiento, se utiliza en la clase Controlador. Su intención es proporcionar a los componentes una forma flexible de enviar mensajes de difusión a los receptores interesados.

Fachada: Se utiliza en la clase Service1 Porque proporciona una interfaz unificada que representa un conjunto de clases en un subsistema. Es decir, consiste en crear una única clase de manejo más fácil que permita acceder a un conjunto más o menos numeroso y complicado de clases que necesitan una inicialización compleja.

CAPÍTULO 2: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

2.5 Modelo de análisis

El diseño es una representación significativa de ingeniería de algo que se va a construir. Se puede hacer el seguimiento basándose en los requisitos del cliente, y al mismo tiempo la calidad se puede evaluar y cotejar con el conjunto de criterios predefinidos para obtener un diseño bueno (Pressman, 2002). En otras palabras se basa en un mejor entendimiento de los requisitos funcionales identificados, refinándolos y estructurándolos, lo que facilita la visión de las funciones reales del sistema basándose en las actividades que se realizan en las fases de diseño e implementación. Teniendo en cuenta que la metodología es altamente configurable, hay artefactos que pueden obviarse durante el desarrollo, lo que significa que este paso no es obligatorio.

Teniendo en cuenta lo anteriormente planteado, se decide pasar del flujo de requisitos directamente al diseño sin la implementacion del análisis, decisión basada en el estudio realizado y aclarando que para el desarrollo del plugin no se contempló el análisis como parte de la implementación de las soluciones. Ademas las tecnologías sobre las que se desarrolló el sistema son conocidas, permitiendo prescindir del refinamiento de los requisitos e ir directo al diseño.

2.6 Modelo de Diseño

El diseño de un *software* es realmente un proceso de muchos pasos pero que se clasifican dentro de uno mismo. En general, la actividad del diseño se refiere al establecimiento de las estructuras de datos, la arquitectura general del *software*, representaciones de interfaz y algoritmo.

A continuación se muestra el diagrama de clases del diseño.

CAPÍTULO 2: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

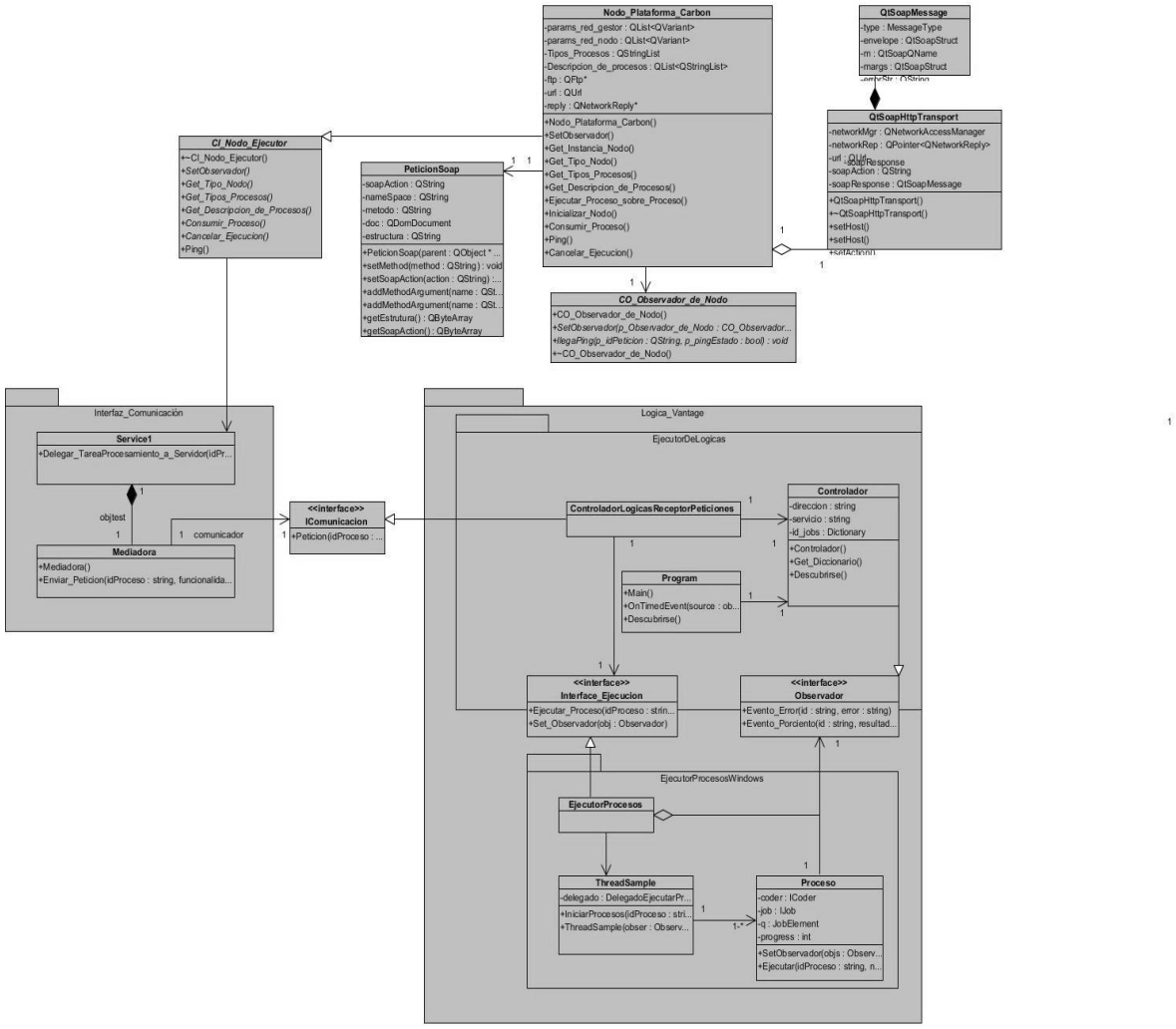


Figura 9. Diagrama de clases del diseño.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

2.7 Conclusiones parciales

Una vez realizada la valoración del diseño propuesto, se puede arribar a la siguiente conclusión:

1. Se realizó el Modelo de Dominio, lo que permite obtener un mayor conocimiento del entorno donde será implantada la solución propuesta.
2. Los requisitos funcionales y no funcionales identificados describen las características que debe cumplir el plugin, lo que permitió una mayor comprensión para desarrollar el sistema.
3. Se realizó la descripción de los CU lo que permite una mejor comprensión del flujo de sucesos.
4. La confección del modelo de diseño permite conocer la estructura del plugin a implementar.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En el siguiente capítulo se describen los elementos relacionados con la implementación del sistema propuesto. Dentro de los artefactos desarrollados en el capítulo se encuentran: el diagrama de componentes y el modelo de despliegue del plugin. Además se validará la solución propuesta con las pruebas de integración, teniendo en cuenta que dichas pruebas serán las que se llevarán a cabo, velando siempre porque se cumplan las necesidades que dieron inicio a la construcción del *software*.

3.1 Modelo de implementación

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas y mostrar las relaciones entre sus elementos. Es un diagrama que muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones.

El modelo de implementación del plugin está conformado por varias clases y la biblioteca XML_RPC, además se muestra la relación entre ellas. En este se encuentran las clases consumidoras de los servicios y las publicadoras de los mismos. A continuación se muestra el diagrama de implementación.

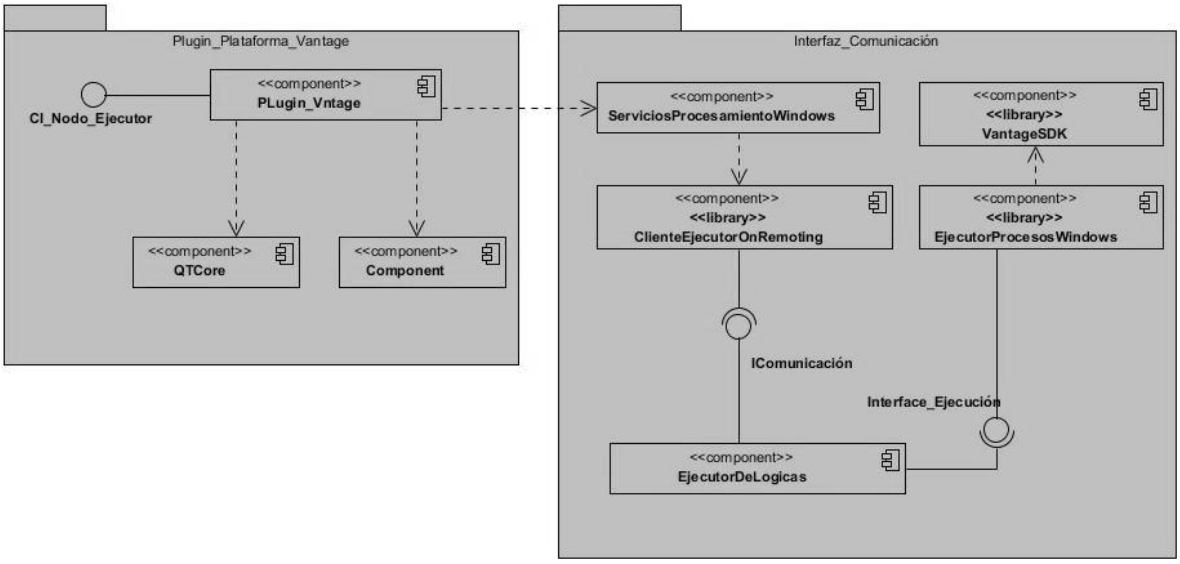


Figura 10. Diagrama de componentes.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.2 Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. También se puede entender como un conjunto de reglas de notación y nomenclatura, específicas de cada lenguaje de programación, que se usan y se siguen durante la fase de implementación (codificación) de una aplicación y reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores que no son detectados por lo compiladores, y reducen el tiempo y coste de las actividades de depuración y pruebas necesarias para la detección y corrección de los mismos (Blog, 2013).

La elaboración y el empleo de estándares de codificación ofrecen numerosas ventajas a la hora de elaborar cualquier tipo de producto software y entre sus principales ventajas se pueden mencionar:

- Se reduce la posibilidad de cometer errores.
- Se obtiene un código legible y comprensible.
- Mejora la comunicación entre los miembros del grupo de programadores del equipo de desarrollo.
- Se asegura la legibilidad del código entre distintos programadores, y facilita la compilación del mismo.

En la presente investigación se propone el uso del siguiente estándar de codificación para seguir con las políticas establecidas por el departamento Señales Digitales:

Tabla 4. Estándares de codificación.

Comentarios, separadores, líneas y espacios en blancos		

Ubicación de comentarios.	Antes de una línea o bloque de código.	Describir brevemente el objetivo (qué función realiza) de un método o línea de código específica. Ejemplo, comentario antes de una línea de código:
		// Aquí se cargó dinámicamente la dll de lógica del negocio
		Assembly m_assembly =
		Assembly.LoadFrom("Ejecutores/EjecutorProcesosWindows.dll");
		Type m_type =
		m_assembly.GetType("EjecutorProcesosWindows.EjecutorProcesos"); Interface_Ejecucion m_clase
		= (Interface_Ejecucion)Activator.CreateInstance(m_type);
		m_clase.Set_Observador(control);

Líneas en blanco.	Antes y después de cada función.	Dejar una línea en blanco entre las declaraciones de método. Ejemplo: void funcionX() {…} //espacio en blanco string funcionY() {…}
Espacios en blanco.	Entre operadores aritméticos y lógicos.	Usar un espacio en blanco entre los miembros de operaciones aritméticas y lógicas. Assembly m_assembly = Assembly.LoadFrom("Ejecutores/EjecutorProcesosWindows.dll"); Type m_type = m_assembly.GetType("EjecutorProcesosWindows.EjecutorProcesos");
Clases, objetos, funciones, atributos		

Nombre de clases y objetos.	Relacionados con el propósito de su función.	Nombrar las clases de manera tal que con solo leerla se note su objetivo. Ejemplo: public class ControladorLogicasReceptorPeticiones : MarshalByRefObject, IComunicacion {…}
Apariencia de atributos.	Letras minúsculas.	Los atributos deben ser escritos en minúsculas y su nombre debe guardar relación con el valor que almacena. Ejemplo: XMLRPC consumidor = new XMLRPC(direccion, servicio);

3.3 Modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño (Jacobson, 2000).

- Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo tipo harwdare similar.
- Los nodos poseen relaciones que representan medios de comunicación entre ellos, tales como internet, intranet y similares.
- El modelo de despliegue puede describir diferentes configuraciones de red, inlcuidas las configuraciones para pruebas y simulasiones.

Figura 11. Diagrama de despliegue.

Descripción de los nodos

A continuación se describen los nodos físicos representados en el diagrama de despliegue.

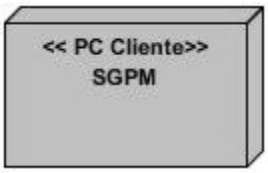


Figura 12. Descripción del nodo SGPM.

Este nodo representa una PC donde se encuentra el SGPM.

Figura 13. Descripción del nodo *Vantage*.

Este nodo representa una PC donde se encuentra el servidor de *Vantage*, en el cual se ejecutan los procesos de codificación y transferencia de media.

3.4 Pruebas

Las pruebas de *software* son los procesos que permiten verificar y revelar la calidad de un producto *software*. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un *software*. Básicamente es una fase en el desarrollo de *software* consistente en probar las aplicaciones construidas.

3.4.1 Pruebas Unitarias

La prueba de unidad es la primera fase de las pruebas del *software* y se realizan sobre cada módulo del *software* de manera independiente. El objetivo es comprobar que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado. En estas pruebas cada parte será probada por separada y lo hará, generalmente, la persona que lo creo. En general, un módulo se entiende como un componente *software* que cumple las siguientes características:

- Debe ser un bloque básico de construcción de programas.
- Debe implementar una función independiente simple.
- Podrá ser probado al cien por cien por separado.
- No deberá tener más de 500 líneas de código.

Se realizan en primer lugar las pruebas unitarias a la Interfaz de Comunicación y al Plugin Controlador por separadas. Una vez realizadas las pruebas unitarias se procede a las pruebas de integración, para esto se harán pruebas a las funciones públicas y se comprobara que los resultados han sido correctos.

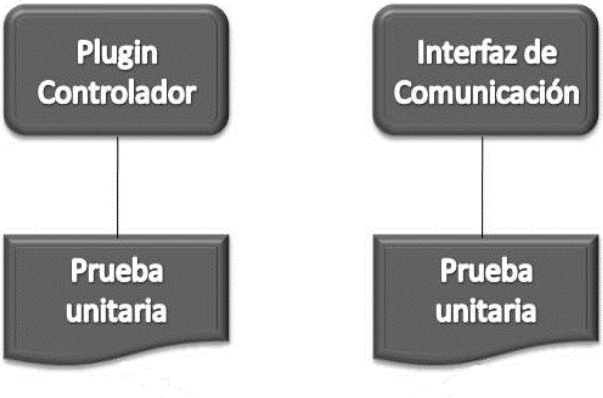


Figura 14. Pruebas unitarias.

3.4.2 Pruebas de Caja Blanca

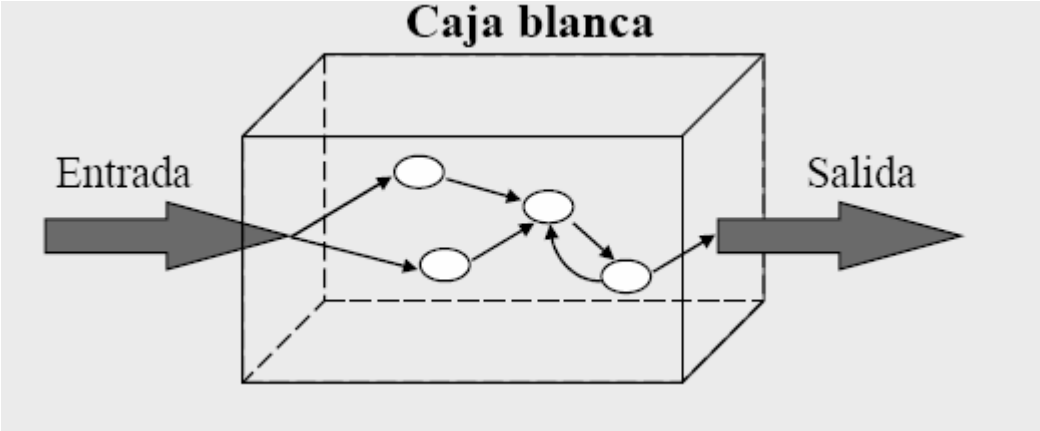


Figura 15. Prueba de Caja Blanca.

A este tipo de pruebas se le conoce también como Pruebas de Caja Transparente o de Cristal. Este método se centra en cómo diseñar los casos de prueba atendiendo al *comportamiento interno y la estructura del programa*. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento. El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa. Como se

ha indicado ya, puede ser impracticable realizar una prueba exhaustiva de todos los caminos de un programa.

Pruebas de Caja Blanca aplicadas al sistema

Clase Nodo_Plataforma_Vantage. Función Consumir_Proceso(Ver Anexo 1, Figura 17)

Paso #1: Grafo de flujos de datos

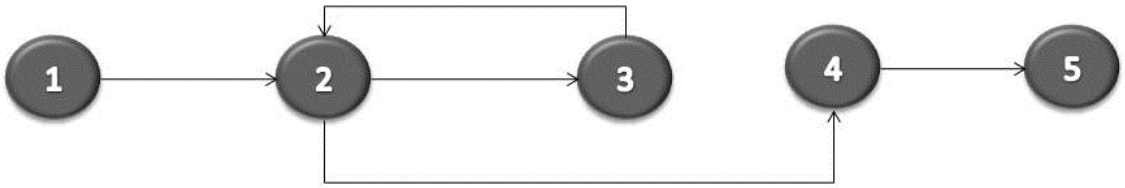


Figura 16. Grafo de flujo de datos. Clase Nodo_Plataforma_Vantage. Función Consumir_Proceso.

Paso #2: Complejidad ciclomática

V (G) =Cantidad Aristas - Cantidad Nodos + 2

V (G) = 5-5 + 2

V (G) = 2

Paso #3: Caminos básicos

CB1: 1, 2, 3, 2, 4, 5

CB2: 1, 2, 4, 5

Paso #4: Casos de prueba

Tabla 5. Método Consumir_Proceso. Caso de Prueba #1.

Caso de Prueba #1	
Camino :	1, 2, 3, 2, 4, 5
Caso de Prueba	Probando Función Consumir_Proceso

Entrada:	Para un valor de: p_id_consumo= ProcesoVantage1, p_nombre_operacion = Codificar_VideoV, params_red_nod = mp4, p_parametros = D:\Para relajar\Videos\Nuevo\NO JUEGES CON MI SOLEDAD.avi
Resultado:	Envía a codificar un video al Servidor *Vantage*.

Clase Proceso. Función Ejecutar (Ver Anexo 2, Figura 18)

Paso #1: Grafo de flujos de datos

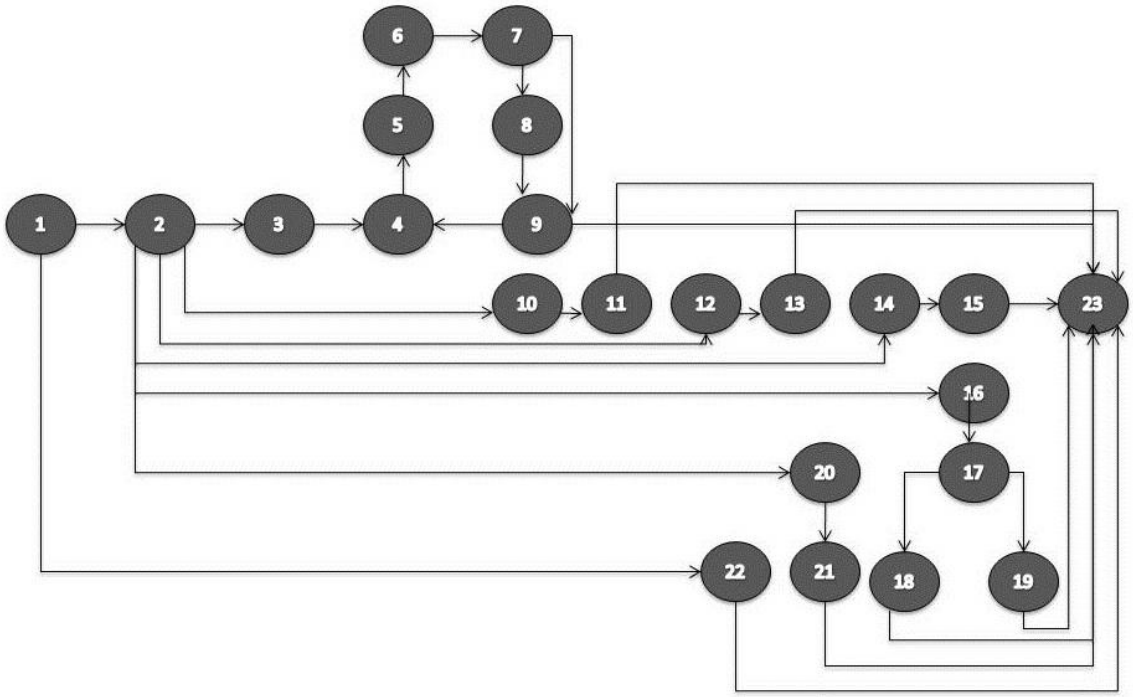


Figura 17. Grafo de flujo de datos. Clase Proceso. Función Ejecutar.

Paso #2: Complejidad ciclomática

V (G) =Cantidad Aristas - Cantidad Nodos + 2

V (G) = 31-23 + 2

V (G) = 10

Paso #3: Caminos básicos

CB1: 1, 22, 23

CB 2: 1, 2, 20, 21, 23

CB 3: 1, 2, 14, 15, 23

CB 4: 1, 2, 12, 13, 23

CB 5: 1, 2, 10, 11, 23

CB 6: 1, 2, 16, 17, 19, 23

CB 7: 1, 2, 16, 17, 18, 23

CB 8: 1, 2, 3, 4, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 23

CB 9: 1, 2, 3, 4, 5, 6, 7, 9, 23

CB 10: 1, 2, 3, 4, 5, 6, 7, 8, 9, 23

Paso #4: Casos de prueba

Caso de Prueba #1	
Camino :	1, 2, 3, 2, 4, 5
Caso de Prueba	Probando Función Consumir_Proceso
Entrada:	Para un valor de: id_Proceso= ProcesoVantage1, funcionalidad = Codificar_VideoV, parametro1 = mp4, parametro2 = D:\Para relajar\Videos\Nuevo\NO JUEGES CON MI SOLEDAD.avi
Resultado:	Codifica y notifica en caso de error.

3.4.3 Pruebas de integración

La necesidad de realizar las pruebas de integración viene dada por el hecho de que los partes que forman un programa suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual. Por ello se realiza este tipo de pruebas, lo que asegura que el plugin que se encuentra en el SGPM funciona de manera correcta a la hora de mandar una tarea al servidor de *Vantage*.

Integración Descendente

Consiste en empezar la integración y la prueba por los módulos que están en los niveles superiores de abstracción, e integrar incrementalmente los niveles inferiores.

Esta prueba se realiza con el objetivo de comprobar que los módulos integrados funcionen de manera correcta a través de pruebas de integración descendente. En la integración descendente se integran los módulos moviéndose hacia abajo por la jerarquía de control, comenzando con el Sistema Gestor de Procesos de Media hasta el Plugin Controlador que en este caso sería el modulo inferior.

Caso de Uso: Codificar video.

Descripción General: El caso de uso inicia cuando el Sistema Gestor de Procesos de Media manda a codificar un video al servidor *Vantage* y este comienza a codificar el video.

Condiciones de Ejecución: El Sistema Gestor de Procesos de Media debe tener cargado el Plugin Controlador.

Casos de prueba

Tabla 6. Caso de Uso Codificar Video.

Escenario	Descripción	Variable 1	Variable 2	Variable n	Respuesta del sistema	Flujo central
EC 1.1 Codificar video correctamente.	Se codifican uno o varios videos en un servidor de *Vantage* según una solicitud recibida.	NA	NA	NA	Codifica uno o varios videos.	No procede
EC 1.2 Fallo en la codificación del video.	Existen errores en los datos a codificar.	NA	NA	NA	Muestra un mensaje de error indicando que los datos para codificar el video son incorrectos o no existen.	No procede

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.5 Conclusiones parciales

Una vez realizada la implementación y pruebas, se puede arribar a la siguiente conclusión:

1. Se modeló el diagrama de despliegue lo cual garantiza que se pueda tener una idea de cómo estará distribuido físicamente el plugin para su correcto funcionamiento.
2. Se realizó el diagrama de componentes lo que permitió modelar la vista estática del Plugin Controlador y la Interfaz de Comunicación.
3. A través de las pruebas seleccionadas se pudo validar el cumplimiento de los requisitos funcionales establecidos para el proceso de desarrollo del Plugin Controlador, comprobándose así el correcto funcionamiento de los mismos.

CONCLUSIONES

Luego de realizar el presente trabajo se ha dado cumplimiento al objetivo general propuesto para esta investigación, concluyendo de la siguiente manera:

1. Con la fundamentación teórica se establecieron las bases para el desarrollo del plugin necesario para el SGPM.
2. La descripción general del objeto de estudio permitió llegar a conocer a fondo los principales grupos de acciones de la herramienta *Vantage* para un mayor conocimiento y dominio a la hora de realizar el plugin para consumir sus procesos.
3. Se realizó el diseño e implementación de los requisitos funcionales definidos para el *software*, logrando crear el plugin propuesto.
4. La selección de la arquitectura permitió obtener y llevar a cabo un esquema organizativo fundamental para sistemas, en este caso, el plugin a implementar.
5. Se realizaron pruebas unitarias, de integración y de caja blanca, para comprobar la calidad y funcionamiento del sistema, demostrando el cumplimiento de los requisitos establecidos.

RECOMENDACIONES

Al concluir el presente trabajo se recomienda:

Para potenciar el uso de la interfaz de comunicación y el plugin controlador se sugiere controlar más procesos de la herramienta *Vantage* como la acción Examinar, Comparar e Identificar.

BIBLIOGRAFÍA

1. **ABC definición. 2007-2013.** Definición ABC. [En línea] 2007-2013. http://www.definicionabc.com/tecnologia/transferencia.php.
2. **Blog, Miguel Matas. 2013.** Cómo Definir un Estándar de Codificación y/o Trabajo. [En línea] 2013. http://www.miguelmatas.es/blog/blog/2008/05/06/como-definir-un-estandar-de-codificacion-yo-trabajo/.
3. **Bonillo, Pedro. 2006.** *METODOLOGÍA PARA LA GERENCIA DE LOS PROCESOS DEL NEGOCIO SUSTENDA EN EL USO DE PATRONES.* 2006.
4. **Camacho, José Alberto Zilberstein. 2009.** *Ingeniería de Requisitos del Modelo para la Gestión de Riesgos en proyectos de software, desarrollado en la Universidad de las Ciencias Informáticas.* 2009.
5. **Cano, Erasmo Martínez. 2013.** Paperblog. [En línea] 2013.
6. **Carmona, Marcel Curbelo. 2009.** *Desarrollo de un sistema para la gestión del proceso de transferencia de archivos del sistema SAREN.* 2009.
7. **Definición.de. 2008.** Definición.de. [En línea] 2008. http://definicion.de/sonido/.
8. **Díaz, Alvaro A. Penroz. 2005.** *Graphical User Interface (GUI) para el programa servidor de mapas MapServer 4.6.1.* 2005.
9. **ESPAÑOLA, REAL ACADEMIA. 2013.** DICCIONARIO DE LA LENGUA ESPAÑOLA, Vigésima segunda edición. . [En línea] 2013. http://buscon.rae.es/draeI/SrvltConsulta?LEMA=transferir..
10. **Guerrero, Zorilin Alonso. 2009.** *Sistema de Captura y Transcripción de Audio.* 2009.
11. **Jacobson, Ivar. 2000.** *El Proceso Unificado de Desarrollo de Software.* 2000.
12. **José H. Canós, Patricio Letelier yMª Carmen Penadés. 2003.** *Métodologías Ágiles en el Desarrollo de Software.* 2003.
13. **Larman, Craig. 2003.** *UML y Patrones.* 2003.
14. **Martínez, Beny Cabrera. 2011.** *Plataforma de codificación e indexación de video para el Sistema de Captura y Catalogación de Medias.* 2011.

15. **Microsoft. 2012.** Introducción al lenguaje C# y .NET Framework. [En línea] 2012. http://msdn.microsoft.com/es-es/library/z1zx9t92%28v=vs.80%29.aspx.

16. **Mike. 2012.** Académica, Comunidad Digital de Conocimiento. [En línea] 2012. http://www.academica.mx/blogs/las-pruebas-integraci%C3%B3n-software.

17. **Moreno, Manuel. 2012.** ITespresso ES. [En línea] 2012. http://www.itespresso.es/discos-duros-de-estado-solido-con-480-gb-disponibles-en-2013-56313.html.

18. **Patricia Hoyos Carvajal y otros. 2002.** Perfeccionamiento contínuo para garantizar procesos exitosos. [En línea] 2002. http://www.greensqa.com/portal/index.php/soluciones/calidad-de-productos/53-pruebas-de-integracion.

19. **Pelaez, Juan Carlos. 2009.** Arquitectura basada en Componentes. [En línea] 2009. http://geeks.ms/blogs/jkpelaez/archive/2009/04/18/arquitectura-basada-en-componentes.aspx.

20. **Pérez, Fabián Bermeo. 2010.** Metodología RUP - desarrollo de software de calidad. [En línea] 2010. http://fabianbermeop.blogspot.com/2010/12/metodologia-rup-desarrollo-de-software.html.

21. **Pressman, Rogger. 2005.** *Ingeniería del Software, un enfoque práctico.* 2005.

22. **Pressman, Rogger S. 2002.** *Ingeniería del Software, un enfoque práctico.* 2002.

23. **Puebla, Universidad de las Américas. 2013.** *Colección de Tesis Digitales.* 2013.

24. **Quevedo, José Enrique Tejera. 2010.** *Subsistema de Audio y Subsistema de Video del Producto Captura y Catalogación de Medias (CCM).* 2010.

25. **Reynoso, Billy. 2005.** *Architect Academy: Seminario de Arquitectura de Software.* UNIVERSIDAD DE BUENOS AIRES : s.n., 2005.

26. **Rosquete, Daniel. 2012.** Evolución informática. [En línea] 2012. http://www.monografias.com/trabajos19/evolucion-informatica/evolucion-informatica.shtml#tecnolog.

27. **Sánchez, José Cegarra. 2011.** *Metodología de la investigación científica y técnológica.* 2011.

28. **Simerly, Tim. 2010.** Electrónica de potencia y sistemas de alimentación y regulación. [En línea] 2010. [Citado el: 23 de Septiembre de 2012.] http://www.convertronic.net/Other-Semiconduct/transcodificacion-de-audiovideo-para-electronica-de-consumo.html.

29. **Sosa, Raydel Raúl Viñolo. 2012.** *Sistema Gestor de Procesos de Media v2.* 2012.

30. **Tedeschi, Nicolás. 2012.** *MSDN.¿Qué es un Patrón de Diseño?* s.l. : Microsoft, 2012.

31. **Telestream. 2012** [En línea] [Citado el: 29 de Septiembre de 2012.] http://www.telestream.net/vantage/vantage-transcode.htm.

32. **Telestream, Inc. 2012.** Telestream. *Telestream.* [En línea] 16 de abril de 2012. [Citado el:] http://www.telestream.net/company/press/2012-04-16-Vantage-Transcode-HE-Server.htm.

33. **Vantage, Ayuda de. 2012.** Ayuda de Vantage. 2012.

34. **Varen, Eliana. 2012.** *Componente para el procesamiento de imagen en sistemas de información audiovisual.* 2012.

35. **Viklund, Andreas. 2010.** Information & Technology . [En línea] 2010.

36. **Whiskito. 2013.** Xataka. [En línea] 2013.

37. **Wright, Wisleidys Campos. 2010.** *PROPUESTA DE LA GENERACIÓN DE NOTICIAS EN FORMATO DE VIDEO PARA LA PLATAFORMA DE TELEVISIÓN INFORATIVA PRIMICIA.* 2010.

38. **Yanvary, Escobar. 2009.** Desarrollo del software. [En línea] 2009. http://www.monografias.com/trabajos39/desarrollo-del-software/desarrollo-del-software.shtml.

ANEXOS

Anexo 3

```
void Nodo_Plataforma_Vantage::Consumir_Proceso(QString p_id_consumo, QString p_nombre_operacion, QList<QVariant> params_red_nodo,QList<QVariant> p_parametros)
{
    QString ip= params_red_nodo.at(0).toString();                                                                    1
    int puerto= params_red_nodo.at(1).toInt();
    qDebug() << ip << QString::number(puerto)<< p_nombre_operacion << " ****Aqui esta el host y operacion***";
    p_parametros.removeLast();
    PeticionSoap *soap = new PeticionSoap();
    soap->setMethod("Delegar_TareaProcesamiento_a_Servidor");
    soap->addMethodArgument("idProceso",QVariant(p_id_consumo));
    soap->addMethodArgument("funcionalidad",QVariant(p_nombre_operacion));
    QStringList valores = QStringList();

    for(int i = 1; i < p_parametros.count(); i++)                                                                    2
    {
        valores.append(p_parametros.at(i).toString());                                                               3
    }

    soap->addMethodArgument("parametros",valores);                                                                   4
    url.setUrl("http://"+ip+"/Service1.asmx");
    QNetworkRequest request(url);
    QNetworkAccessManager *manager = new QNetworkAccessManager(this);
    request.setHeader(QNetworkRequest::ContentTypeHeader, QLatin1String("text/xml;charset=utf-8"));
    request.setRawHeader("SOAPAction",soap->getSoapAction());
    reply = manager->post(request,soap->getEstrutura());
    connect(reply,SIGNAL(readyRead()),this,SLOT(RespuestaReply()));
}                                                                                                                    5
```

Figura 18. Función Consumir_Proceso de la clase nodo_plataforma_vantage.

Anexo 4

```
public void Ejecutar(string idProceso, string funcionalidad, string [] parametros)
{
    try                                                                                 1

    {
        switch (funcionalidad)                                                          2
        {
            case "Codificar_VideoV":                                                    3
                {

                    string aux = Servicios.Codificar(parametros[0], parametros[1]).ToString();  4
                    observers.Get_Diccionario().Add(idProceso, aux);
                    int progreso = 0;
                    do                                                                  5
                    {
                       int prog= Servicios.ProgresoTrabajo(aux);                        6

                       if (prog != progreso) {                                          7
                           progreso = prog;                                             8
                           observers.Evento_Porciento(idProceso, progreso.ToString());
                       }
                       Thread.Sleep(1000);
                    } while (progreso<100);                                             9
                    observers.Get_Diccionario().Remove(idProceso);
                } break;
            case "Mover_VideoV":                                                        10
                {
                    string aux = Servicios.MoverArchivo(parametros[0]).ToString();      11
                } break;
            case "Copiar_VideoV":                                                       12
                {
                    string aux = Servicios.CopiarArchivo(parametros[0]).ToString();     13
                } break;
            case "Eliminar_VideoV":                                                     14
                {
                    string aux = Servicios.EliminarArchivo(parametros[0]).ToString();   15
                } break;
            case "Cancelar":                                                            16
                {
                    if (observers.Get_Diccionario().ContainsKey(idProceso))             17
                    {
                        Servicios.DetenerTrabajo(observers.Get_Diccionario()[idProceso]);   18
                        Servicios.EliminarTrabajo(observers.Get_Diccionario()[idProceso]);
                        observers.Get_Diccionario().Remove(idProceso);

                    }
                    else {
                        observers.Evento_Error(idProceso, "No existe el id proceso");   19
                    }

                } break;
            default:                                                                    20
                {
                    observers.Evento_Error(idProceso, "No existe esta funcionalidad");  21
                } break;

        }
    }
    catch (Exception e)
    {
        observers.Evento_Error(idProceso, "Error: "+e.Message);                         22
    }


}                                                                                       23
```

Figura 19. Función Ejecutar de la clase Proceso.