Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Componente para la transmisión de información de control y video para un Sistema de Laboratorios Virtuales y a Distancia.

Autor: Yohan Díaz Acosta

Tutores: MSc. Omar Mar Cornelio

La Habana, junio de 2015

"Año 57 de la Revolución"

¨ Los que dicen que es imposible… No deberían molestar a los que lo están haciendo.¨

Albert Einstein

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yohan Díaz Acosta	MSc. Omar Mar Cornelio
_____	_____
Firma del Autor	Firma del Tutor

Autor:

- Yohan Díaz Acosta
- Universidad de las Ciencias Informáticas
- Correo: yacosta@estudiantes.uci.cu

Tutor:

- MSc. Omar Mar Cornelio
- Universidad de las Ciencias Informáticas
- Correo: omarmar@uci.cu

Agradecimientos

Esta es una de las partes más difíciles de la tesis, aquí es donde le agradezco a todas las personas, que de una forma u otra me han ayudado con la realización de este hermoso sueño.

Primero que todo a dos personas que han permitido hoy ser quien soy, mis padres, que han estado ahí para mí y me han apoyado en todo momento. Gracias por darme ánimo siempre, por aconsejarme, guiarme en la vida y por permitir que nunca me rindiera.

A mis hermanas Yausmara y Aismara.

A mi familia en general por su gran apoyo, muchas gracias por confiar en mí.

A todos mis amigos que me es imposible mencionarlos a todos, se convirtieron en este tiempo en mi familia, gracias por compartir tanto los momentos buenos como los malos durante estos 5 años y mi gente del Palenque Record tengan presente que nunca los olvidaré.

A mi tutor Omar Mar que supo guiarme hacia mi meta, gracias por haberme ayudado y por su grandísimo esfuerzo.

A los miembros del tribunal por su exigencia, dedicación y preocupación durante el periodo de desarrollo del trabajo de diploma. Muchas Gracias.

Dedicatoria

A toda mi familia, en especial a mis padres Hildeliza Acosta Romero y Plácido Díaz Díaz, a quienes les debo lo que he logrado en esta vida.

A mis hermanas Yausmara y a Aismara que siempre han sido un ejemplo para mí.

Resumen

Con el desarrollo de las Tecnologías de la Información y Comunicaciones (TICs) son cada vez más las áreas del conocimiento que se integran. El control automático ha aportado los Sistemas de Laboratorios Virtuales y a Distancia (SLVD) con el objetivo de intercambiar recursos tecnológicos a través de Internet. La Universidad de las Ciencias Informáticas (UCI) de conjunto con la Universidad Central de las Villas “Marta Abreu” (UCLV), tiene como objetivo la integración de los centros educacionales del Ministerio de Educación Superior, en el desarrollo de prácticas de laboratorio en la asignatura de control automático, para lo que se ha desarrollado un SLVD. Sin embargo, este sistema no tiene un procedimiento estandarizado para la transmisión de información de control entre las diferentes entidades del SLVD y presenta insuficiente optimización del flujo de video entre las estaciones de trabajo, haciendo uso innecesario del ancho de banda. La presente investigación describe una solución a la problemática planteada a partir de la implementación de un componente para la transmisión de información de control y flujo de video para un SLVD, para lo cual se guía el proceso de desarrollo mediante la metodología Open UP, utilizando como herramienta de modelado Visual Paradigm for UML, como lenguaje de programación C++ y como marco de trabajo QT sobre el entorno de desarrollo integrado QT Creator. El sistema propuesto brinda una estructura definida para la transmisión de información de control, así como la optimización del flujo de video generado en las estaciones de trabajo utilizando como protocolo de transporte UDP. Para garantizar el correcto funcionamiento del componente informático, se realizó la prueba de caja blanca, haciendo uso de la técnica de camino básico.

Palabras Claves: Sistema de Laboratorios Virtuales y a Distancia, Protocolo.

Abstract

With the development of Information Technology and Communications are increasingly knowledge areas that are integrated. Automatic control systems has provided the Virtual Labs and Distance with the aim of exchanging technological resources via the Internet. University of Information Sciences in conjunction with the Central University of Las Villas "Marta Abreu" aims at integrating the educational centers of the Ministry of Higher Education in the development labs in the course of automatic control which it has developed a system of virtual laboratories and distance. However, this does not have a standardized system for the transmission of control information between the different entities of Virtual Labs and Distance and has insufficient flow optimization video between workstations, making it unnecessary bandwidth use procedure. This paper describes a solution to the issues raised from the implementation of a component for transmitting control information and video stream for a Virtual Labs and Distance, for which the development process is guided by the methodology Open UP, using modeling tool Visual Paradigm for UML as programming language C ++ and QT as a framework for work on the integrated development environment QT Creator. The proposed system provides a defined structure for transmission of control information, as well as optimization of the video stream generated on workstations using UDP as the transport protocol. To ensure the proper functioning of the software component, the white-box testing was performed, using the technique of basic path.

Keywords: *Virtual Labs and Distance, Protocol.*

Índice general

Índice de figuras

Índice de tablas

Introducción

Históricamente el hombre ha sentido la necesidad de estudiar su medio y aprender de él, para lo que ha buscado las vías factibles y novedosas para ello. Con el paso del tiempo y el desarrollo tecnológico, aprender se torna necesario y en muchas ocasiones imprescindible para adaptarse al contexto sociopolítico actual. El avance de las Tecnologías de la Información y las Comunicaciones (TICs) ha favorecido esta tarea. Cada vez más se hace necesario acceder a información y el auto-aprendizaje se pone a la vanguardia.

La incorporación de tecnologías en el sector educacional ha permitido potenciar la educación a distancia con la creación de nuevas propuestas con fines de desarrollo profesional y de colaboración. El uso combinado de métodos pedagógicos y materiales de auto-aprendizaje con las diversas tecnologías, ha propiciado el acercamiento entre las personas y el acceso generalizado a un mismo material de enseñanza (Trapero, 2009).

Durante la última década, se han desarrollado plataformas conocidas como SLVD, demostrando tener potencial para la formación en el campo de la ingeniería robótica y de control. Un SLVD permite a los estudiantes por medio de experimentos en tiempo real interactuar con dispositivos físicos a distancia (Chacez, 2008).

El sistema educacional cubano también ha incursionado en el aprendizaje virtual y a distancia. El Departamento de Automática y Sistemas Computacionales de la UCLV, en conjunto con el Departamento de Automática, Ingeniería Electrónica e Informática Industrial, de la Universidad Politécnica de Madrid desarrolló un SLVD para la Asignatura de Regulación Automática I (Ching, 2010). Este sistema presenta una arquitectura cliente servidor. Los clientes se conectan a través de Internet al Servidor Web, que a su vez estará conectado a las Estaciones de Trabajo que contiene dispositivos a controlar, los cuales son monitorizados mediante una cámara que envía las imágenes en tiempo real a los usuarios que realizan las prácticas de laboratorios.

Para que los usuarios visualicen el contenido de las prácticas en SLVD se usa el protocolo de comunicación TCP (Protocolo de Control de Transmisión), el cual implementa mecanismos de retransmisión en caso de que el receptor no confirme la recepción de los datos. Además implementa mecanismos de control de congestión y de control de flujo, implicándose un mayor consumo del ancho de

banda disponible y retardo en el flujo de información transmitido, lo que hace ineficiente el proceso de intercambio de información.

Con el objetivo de incorporar más instituciones teniendo en cuenta las principales carencias que presentan los centros del Ministerio de Educación Superior, se ha decidido actualizar el sistema existente con vista a mejorar su funcionamiento. Para lo cual la Universidad de las Ciencias Informáticas (UCI) en conjunto con la UCLV, mediante proyecto de colaboración en el campo de la educación, está desarrollando una nueva plataforma al SLVD.

Sin embargo, el autor en su práctica cotidiana con la utilización del método empírico entrevista (Ver Anexo 1) pudo identificar las siguientes problemáticas:

- No existe un método estandarizado para la transmisión de información de control entre las diferentes entidades del SLVD, provocando diferentes codificaciones entre las estaciones de trabajo.
- Insuficiente optimización del flujo de vídeo generado entre las estaciones de trabajo y el servidor del SLVD, consumiendo mayor ancho de banda del necesario.
- Inexistencia en el mercado de SLVD libres, imposibilitando la personalización o modificación de la solución según las necesidades del Sistema Educacional Cubano.

Por lo antes planteado, se tiene como **problema de la investigación:** ¿Cómo transmitir información de control y video en un Sistema de Laboratorios Virtuales y a Distancia?

Se define como **objeto de estudio** de la investigación: Los procesos de transmisión de señales digitales y queda planteado el **campo de acción** como: Los procesos de transmisión de señales digitales en un Sistema de Laboratorios Virtuales y a Distancia.

Para dar respuesta al problema antes planteado, se tiene como **objetivo general:** desarrollar un componente para la transmisión de información de control y video para un Sistema de Laboratorios Virtuales y a Distancia.

Preguntas científicas:

1. ¿Cuáles son los fundamentos metodológicos para la transmisión de información de control y video en un SLVD?

2. ¿Cómo guiar el desarrollo del componente para la transmisión de información de control y video en un SLVD?

3. ¿Cómo comprobar las funcionalidades del componente para la transmisión de información de control y video?

Para dar cumplimiento al objetivo general y a las preguntas científicas, se plantean las siguientes **tareas de la investigación:**

- Determinar los conceptos asociados a los SLVD.
- Seleccionar la metodología y las herramientas a utilizar en el proceso de desarrollo del componente para la transmisión de información y video para un SLVD.
- Analizar el estado del arte de los SLVD para conocer las funcionalidades con que cuentan estos en el mundo.
- Realizar el diseño ingenieril de los procesos para representar las funcionalidades del componente para la transmisión de información de control y video para el SLVD.
- Implementar las funcionalidades de la solución propuesta.
- Aplicar pruebas de software al componente para la transmisión de información de control y video para un SLVD, con el objetivo de identificar posibles errores y corregirlos.

Métodos científicos de la investigación

Para determinar la problemática existente, así como para lograr el desarrollo y cumplimiento del objetivo planteado, se hace uso de un conjunto de métodos científicos, los cuales se presentan a continuación.

Métodos teóricos:

Analítico – Sintético: Se utilizó para realizar un análisis de la bibliografía existente para el procesamiento de señales digitales en los SLVD.

Histórico – Lógico: Se utilizó para realizar una investigación acerca de la evolución de los SLVD, identificando sus características y funcionalidades, lográndose un entendimiento de los mismos.

Métodos empíricos:

Entrevista: Aplicada a profesores y especialistas que imparten las asignaturas Robótica y Control en la UCLV, con el objetivo de diagnosticar las principales deficiencias del SLVD.

El documento se encuentra estructurado en Introducción, tres capítulos, Conclusiones Generales, Referencias Bibliográficas y Anexos. A continuación se realiza una descripción de los Capítulos.

Capítulo I: Fundamentos Teóricos

En este capítulo se exponen los fundamentos teóricos que dan sustento al trabajo de diploma, abordando los elementos conceptuales que fundamentan la investigación, incluyendo temas sobre la transmisión de información de control y video. Se describen las herramientas, lenguajes y metodologías que guiarán el desarrollo del componente propuesto.

Capítulo II: Análisis y Diseño del componente

En este capítulo se describe el modelo conceptual para representar los conceptos más importantes del negocio que se muestran en el dominio del problema. Se realizan el diagrama de casos de uso y el modelo de diseño del sistema, para permitir un mejor entendimiento de las funcionalidades del componente, además se explica la arquitectura a utilizar para la solución.

Capítulo III: Implementación y Prueba

En este capítulo se aborda la implementación del componente, donde se presentan el diagrama de despliegue y el diagrama de componentes. Además se realiza el proceso de validación del sistema a partir de las pruebas realizadas al componente, para verificar su correcto funcionamiento.

Capítulo 1: Fundamentos Teóricos.

1.1 Introducción

Los fundamentos teóricos constituyen la base sobre la cual se realiza la presente investigación, por ello en este capítulo se realizará una descripción detallada de cada elemento teórico que posibilitará comprender, definir y explicar la situación problemática que dio origen al problema a resolver y así se guiará el desarrollo investigativo hacia una solución. Además se definirá la metodología de desarrollo de software, así como las herramientas y tecnologías que permiten llevar a cabo la modelación, diseño y construcción de la futura solución.

1.2 Conceptos asociados a la investigación

La **educación a distancia** es la gestión o proceso de educar o ser educado cuando este proceso auto dirigido se realiza a distancia. Tal acto educativo es apoyado por el material didáctico elaborado por un equipo experimentado y multidisciplinario. Este tipo de educación se centra en hacerle llegar la información a los estudiantes, eliminando la necesidad de que se encuentren presentes en el momento de impartir la clase, apoyándose para ello en otros medios como video conferencias, sesiones de chat, documentos, multimedia, presentaciones u otra de las posibilidades que brinda Internet (Gómez, 2012).

El **aprendizaje electrónico** (conocido también por el anglicismo e-learning) se refiere a la educación a distancia completamente virtualizada a través de los nuevos canales electrónicos (las nuevas redes de comunicación, en especial Internet), utilizando para ello herramientas o aplicaciones de hipertexto (correo electrónico, páginas web, foros de discusión, mensajería instantánea, plataformas de formación, etc.) como soporte de los procesos de enseñanza-aprendizaje (Torres, 2011).

Con el objetivo de materializar estos renovadores conceptos de aprendizaje, se crearon los **Sistema de Laboratorios Virtuales y a Distancia**. Diferentes estudiosos han tratado de dar una definición lo más exacta posible de los Laboratorios a Distancia, por ejemplo:

“Son simulaciones de prácticas manipulativas que pueden ser hechas por la/el estudiante lejos de la universidad y el docente” (Monge-Magera, 1999).

En el Informe de la reunión de expertos (Vary, 1999) sobre laboratorios virtuales organizada por el Instituto Internacional de Física Teórica y Aplicada (IITAP) Ames, celebrada en Iowa del 10 al 12 de mayo de 1999, se determinó que un Laboratorio virtual es:

"Un espacio electrónico de trabajo concebido para la colaboración y la experimentación a distancia, con el objetivo de investigar o realizar otras actividades creativas, y elaborar y difundir resultados mediante tecnologías de información y comunicación".

También puede considerarse como "un centro sin paredes cuyos usuarios pueden investigar sin tener en cuenta su situación geográfica, interactuando con los colegas, teniendo acceso a los instrumentos, compartiendo los datos y los recursos informáticos, y recurriendo a la información de las bibliotecas electrónicas. Ese entorno se apoya en programas informáticos que permiten trabajar en colaboración y simultáneamente a diversas personas desde distintos sitios" (Wulf, 2000).

Resumiendo lo antes expuesto y tomando como frontera los dominios de la investigación, se puede decir que un SLVD es *"un espacio electrónico de trabajo que permite desarrollar prácticas de laboratorio sin que los participantes estén localizados en la misma ubicación geográfica".*

Los SLVD permiten controlar de forma remota **estaciones de trabajo** y los distintos dispositivos que la componen. Estas estaciones consisten en una computadora con acceso a la red. Cada estación de trabajo puede contar con uno o varios dispositivos conectados, dígase, cámaras, microscopios virtuales, robots automáticos entre otros que se gestionan a través de algún software especificado, permitiendo además el control y visualización de dichos dispositivos de forma remota (Ching, 2010).

1.3 Soluciones existentes

A continuación se presentan algunas de las soluciones existentes que facilitan la transmisión de información de control y video, tanto en el ámbito nacional como internacional.

1.3.1. Sistema de Laboratorios a Distancia para la Asignatura de Regulación Automática I

El Sistema de Laboratorios a Distancia para la Asignatura de Regulación Automática I se encuentra desplegado en la UCLV, enfocado en el estudio de la automática. Su principal objetivo es permitir a los usuarios aprender a ajustar controladores predefinidos y a diseñar sus propios controladores para posteriormente probarlos sobre un conjunto de dispositivos físicos a través de Internet (Castellanos, y

otros, 2004).

El sistema utiliza herramientas que permiten una rápida y fácil integración de nuevos procesos para experimentos de control y en la actualidad están disponibles: un motor de corriente directa y un robot manipulador. Esta herramienta se basa en el protocolo TCP, el cual no proporciona rapidez en el envío de la información en la red, lo cual constituye una desventaja.

1.3.2. Laboratorio virtual RoboLab

El Laboratorio virtual RoboLab fue desarrollado por el Departamento de Física, Ingeniería de Sistemas y Teoría de la Señal, de la Universidad de Alicante, para el aprendizaje de la asignatura Ingeniería. Permite la interoperabilidad de los estudiantes en la asignatura de Ingeniería, realizando distintos ejercicios prácticos en la simulación robótica on-line. Este laboratorio, al igual que el SLVD de la UCLV, se basa en el protocolo TCP (Cikajlo 2012).

1.3.3. Red Universitaria de Laboratorios Interactivos (UNILabs)

La Red Universitaria de Laboratorios Interactivos (UNILabs, por sus siglas en inglés) posibilita la realización de prácticas de Automática, Matemáticas, Control de Procesos Avanzados y Técnicas Experimentales de Física (Unilabs, 2014). Para poder acceder a cada uno de sus módulos primeramente se tiene que haber realizado una reserva para el mismo. Las simulaciones de los experimentos están desarrolladas en *Easy Java Simulation* (EJS), una herramienta de código abierto desarrollada en Java, diseñada para la generación de simulaciones interactivas dinámicas. Al igual que los anteriores, está basado en TCP.

1.3.4. Virtual Labs

Virtual Labs es un portal de laboratorios virtuales en 97 campos de 9 disciplinas distintas de Ciencia e Ingeniería, incluyendo Electrónica, Comunicaciones, Informática y Biotecnología (Lab, 2014). Se encuentra orientado a estudiantes universitarios y a usuarios interesados en la investigación científica. Permite realizar experimentos en línea, para lo que sólo es necesario un ordenador con conexión a internet.

1.3.5. Labster

Laboratorio a distancia que permite realizar experimentos a través de la Web en equipos de un laboratorio real, entre los que se encuentran la fermentación, genética médica, clonación molecular, cromatografía y microbiología (Labster, 2014).

1.3.6. Tabla comparativa con las soluciones existentes

Una vez analizadas cada una de las soluciones existentes, se procede a realizar una comparación entre ellas, con el objetivo de determinar si pueden servir para dar solución a la problemática planteada o pueden aportar algo que contribuya en la realización de un nuevo sistema. Dicha comparación está basada en los siguientes aspectos: disponibilidad, protocolo de transporte y métodos de estandarización de la transmisión de información de control.

Tabla 1: Tabla comparativa de las soluciones existentes.

Soluciones	Privativa	Protocolo de transporte	Métodos de estandarización
RoboLab	No	TCP	No
UNILabs	No	TCP	No
Labster	Si	TCP	No
Virtual Labs	No	TCP	No
SLD UCLV	No	TCP	No

Luego de ser realizada las comparaciones entre las soluciones existentes en la actualidad, tanto nacionales como internacionales, se determinó que ninguna constituye una solución para el problema planteado. Sin embargo, se comprendió a mayor escala la importancia de una solución.

1.4 Tecnologías, metodologías y herramientas para el desarrollo del software

Para el desarrollo del componente a implementar se adoptan las metodologías, herramientas y tecnologías definidas por el grupo de desarrollo de la plataforma SLVD. De esta manera se garantiza compatibilidad entre los diferentes productos en desarrollo.

1.4.1. Metodología de desarrollo de software

Para desarrollar el componente para la transmisión de información de control y transmisión de video para un Sistema de Laboratorios Virtuales y a Distancia se utiliza la metodología Open UP. Esta metodología de desarrollo de software, basada en RUP (Rational Unified Process), contiene el conjunto mínimo de prácticas que ayudan a un equipo de desarrollo de software a realizar un producto de alta calidad, de una forma eficiente. Open UP, es un proceso unificado, iterativo e incremental, que se centra en el desarrollo colaborativo de software para generar sistemas de calidad. Los elementos que forman Open UP son: tareas, disciplinas, artefactos y procesos.

El ciclo de vida de un proyecto, según la metodología Open UP, permite que los integrantes del equipo de desarrollo aporten con micro-incrementos, que pueden ser el resultado del trabajo de unas pocas horas o unos pocos días. El progreso se puede visualizar diariamente, ya que la aplicación va evolucionando en función de este micro-incremento. El objetivo de Open UP es ayudar al equipo de desarrollo, a lo largo de todo el ciclo de vida de las iteraciones, para que sea capaz de añadir valor de negocio a los clientes, de una forma predecible, con la entrega de un software operativo y funcional al final de cada iteración (Gimon, y otros, 2012).

Open UP consta de cuatro fases: inicio, elaboración, construcción y transición. Cada una de estas fases se divide a su vez en iteraciones (Gimon, y otros, 2012).

Se escoge como metodología de desarrollo de software a Open UP, ya que permite al equipo de desarrollo integrarse a requerimientos cambiantes que se pueden dar durante el desarrollo del proyecto. Es una metodología ágil, muy favorable para proyectos que presentan pocos integrantes y es aplicable a sistemas de bajo costo.

1.4.2. Lenguaje de modelado del sistema

Una exigencia de las instituciones es que los desarrollos de *software* se formalicen con un lenguaje estándar y unificado. Es decir, se requiere que cada una de las partes que comprende el desarrollo de todo software de diseño orientado a objetos, se visualice, especifique y documente con un lenguaje común (Cornejo 2008). Un lenguaje unificado que cumple con estos requisitos, es ciertamente el Lenguaje Unificado de Modelado (UML, del inglés, *Unified Modeling Language*).

Lo fundamental de UML es la capacidad de diagramación y los diferentes tipos de diagramas que soporta. UML 2.0 está basado en el Paradigma Orientado a Objetos. También puede conectarse con lenguajes de programación (Ingeniería directa e inversa) y permite documentar todos los artefactos de un proceso de desarrollo. Es un lenguaje muy expresivo, que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas (Cornejo 2008).

De acuerdo a esto se decide utilizar UML 2.0 como lenguaje de modelado, ya que permite modelar el análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos; además permite realizar la verificación y validación del modelo desarrollado y es flexible para admitir cambios no previstos durante el diseño o el rediseño.

1.4.3. Herramienta de modelado

Visual Paradigm 8.0 es una herramienta CASE que soporta el ciclo completo de desarrollo de un sistema a través de la representación de todo tipo de diagramas. Es una herramienta libre y multiplataforma que se basa en UML como lenguaje de modelado. Presenta licencia comercial y gratuita. Permite la realización tanto de ingeniería directa como inversa, además de un diseño enfocado al negocio que genera un software de mayor calidad (Menéndez, 2014).

Esta herramienta permite aumentar la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del software. Aumenta el conocimiento informático de una empresa ayudando así a la búsqueda de soluciones para los requisitos. También permite la reutilización del software, portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería del Software.

1.4.4. Lenguaje de programación

C++ es un lenguaje de programación de utilidad general diseñado para que el programador serio haga su trabajo de modo más agradable. Salvo algunos detalles menores, C++ es un súper conjunto del lenguaje de programación C. Además de los recursos que ofrece C, C++ proporciona mecanismos flexibles y eficientes para definir nuevos tipos de datos (Stroustrup, 1993).

Este lenguaje de programación está considerado por muchos como uno de los lenguajes más potentes, debido a que permite trabajar tanto a alto nivel como a bajo nivel. Se puede decir que C++ abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Dicho lenguaje gana mucho en potencia ya que tiene la posibilidad de sobrecargar operadores. Las principales características de C++ son las facilidades que proporciona para la programación orientada a objetos y para el uso de plantillas o programación genérica.

1.4.5. Marco de trabajo

QT4 es un framework de desarrollo integral con herramientas diseñadas para simplificar la creación de aplicaciones e interfaces de usuario para el escritorio, embebidos y plataformas móviles. Permite realizar aplicaciones avanzadas y desplegarlas en escritorio y sistemas operativos integrados sin tener que reescribir el código fuente, contribuyendo de esta forma a disminuir el tiempo y el costo de desarrollo del producto. Es fácil de usar, aprender, mantener y de código reutilizable. Además posee un alto rendimiento en tiempo de ejecución y ocupa poco espacio en disco (Garrido, 2009).

1.4.6. Entorno de desarrollo

QT Creator 4.8 es un entorno de desarrollo integrado (en inglés *Integrated Development Environment*) multiplataforma que proporciona herramientas para diseñar y desarrollar aplicaciones informáticas utilizando el framenwork QT (Rischpater, 2014).

Características que se destacan de QT Creador (Rischpater, 2014).

- Avanzado editor de código C++.
- Tiene integrado diseñadores de interfaces gráficas de usuario.
- Ayuda sensible al contexto.
- Resaltado y autocompletado de código.

1.6 Conclusiones parciales

Se realizó el análisis a Sistemas de Laboratorios Virtuales y a Distancia existentes, donde se pudo identificar que poseen deficiencias en el proceso de transmisión de información de control y video, lo cual los descarta como una solución al problema planteado.

Se definieron, en conjunto con el grupo de desarrollo del Sistema de Laboratorios Virtuales y a Distancia, las tecnologías, herramientas y metodología para el proceso de desarrollo del software, las cuales son mencionadas a continuación. Como metodología para guiar el proceso del desarrollo del software Open UP, como lenguaje de programación C++, haciendo uso del entorno de desarrollo integrado QT Creator, como marco de trabajo QT y como lenguaje de modelado UML, haciendo uso de la herramienta de modelado Visual Paradigm for UML.

Capítulo 2: Análisis y diseño de la solución

2.1 Introducción

En el presente capítulo se abordarán argumentos importantes para la elaboración del componente para la transmisión de información y video para un SLVD. Además, se efectuará la representación de los artefactos logrados a través del modelo de dominio y la especificación de los requisitos funcionales y no funcionales que debe presentar la aplicación una vez terminada. Posteriormente se realiza el diagrama de casos de usos del sistema y la descripción detallada de cada caso de uso. También se realiza el modelo de diseño, así como los diagramas de secuencia que permitirán un mejor entendimiento de las funcionalidades del componente.

2.2 Modelo conceptual

El Modelo conceptual permite representar los conceptos más importantes de los objetos que se muestran en el dominio del problema. Un modelo conceptual es una representación de las clases conceptuales del mundo real, no de componentes software. Los conceptos representan las "cosas" que existen o los eventos que suceden en el entorno en el que trabaja el sistema (Larman, 2003).

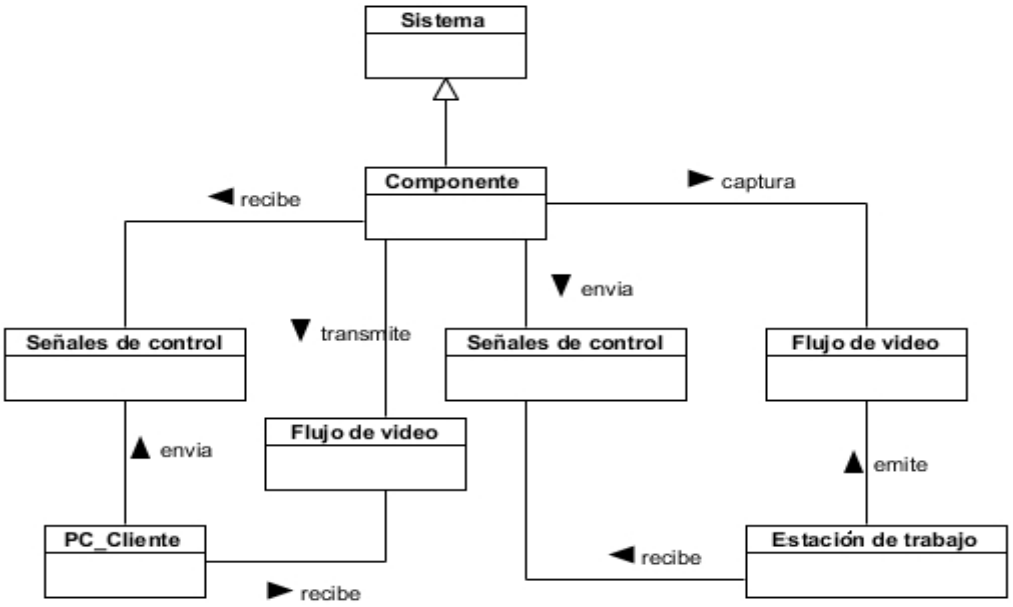

Figura 1: Modelo conceptual

En la Figura 1 se muestra el Modelo conceptual asociado al componente de transmisión de información y video para un SLVD. Este componente recibe señales de control enviadas desde una PC_Cliente, las cuales son enviadas posteriormente a una Estación de trabajo. Esta Estación de trabajo emite un flujo de video, el mismo es capturado por el componente desarrollado y transmitido a la PC_Cliente.

2.2.1 Glosario de términos del Modelo conceptual

Sistema: Sistema al cual se integrará el componente para la transmisión de información de control y video para un Sistema de Laboratorios Virtuales y a Distancia.

Estación de Trabajo: Computadora con acceso a la red a la que se encuentran conectados los dispositivos.

PC_Cliente: Es de donde se van a ejecutar las señales de control y se visualizará el video capturado, procesado y enviado por el componente.

Componente: Es el encargado de transmitir la información de control y el video entre las entidades PC_Cliente y Estaciones de trabajo.

2.3 Especificación de los requisitos de software

2.3.1 Requisitos funcionales

Los Requisitos funcionales son declaraciones de servicios que el sistema debe proporcionar, definen la manera en que éste debe reaccionar a determinadas entradas y cómo se debe comportar en situaciones particulares (Pressman, 2010). Estos requisitos son las características fundamentales del sistema y expresan la capacidad de acción del mismo.

Nota: Se usa el prefijo RF en su nomenclatura.

RF 1. Capturar flujo de video.

RF 2. Procesar flujo de video.

RF 3. Transmitir video.

RF 4. Recibir señales de control.

RF 5. Enviar señales de control.

2.3.2 Requisitos no Funcionales

Los Requisitos no Funcionales son las restricciones de los servicios o funciones ofrecidas por el sistema, restringen el espacio de posibles soluciones (Pressman, 2010). Estos requisitos son los que se centran fundamentalmente en las características generales del sistema y no en los rasgos particulares del mismo.

Nota: Se usa el prefijo RNF en su nomenclatura.

RNF 1: Requisitos de software:

El sistema operativo que soportará la solución será GNU/Linux Ubuntu 14.04.

La PC cliente debe tener instalado el reproductor media VLC v2.2.0

RNF 3: Restricciones en el diseño e implementación:

Lenguaje de programación **C++**.

Se utilizará **QT Creator 4.8.**

Se utilizarán las bibliotecas **Gstreamer 1.0 y Qxmlrpc.1.0**

2.4 Casos de Uso del Sistema

Un caso de uso (CU) expresa todas las formas de usar un sistema para alcanzar una meta particular para un usuario. En conjunto, los casos de uso le proporcionan todos los caminos útiles de usar el sistema e ilustran el valor que este provee (Ivar, 2013).

2.4.1 Descripción de los Casos de Uso del Sistema

Tabla 2: CU Transmitir flujo de video.

CU	Transmitir flujo de video
Actor	Sistema
Descripción	Permite transmitir el flujo de video haciendo uso del protocolo UDP,

	después de ser capturado de un dispositivo cámara IP.
Referencia	RF1, RF2, RF3

Tabla 3: CU Procesar señales de control.

CU	**Procesar señales de control**
Actor	Sistema
Descripción	Permite capturar y enviar señales de control encapsuladas mediante XML y enviadas por HTTP haciendo uso del protocolo de llamadas de retorno XML-RPC.
Referencia	RF4, RF5

2.4.2 Definición del actor del sistema

Según Roger Pressman “los casos de uso se definen desde el punto de vista de un actor. Un actor es un papel que desempeñan las personas (usuarios) o los dispositivos cuando interactúan con el software.” (Pressman, 2013).

Tabla 4: Definición del actor del sistema.

Actor	**Descripción**
Sistema	Es el sistema al cual el componente se va a integrar y que mediante peticiones automáticas, hará uso de las funcionalidades del componente para la transmisión de información de control y flujo de video.

2.4.3 Diagrama de Casos de Uso del Sistema

Un Diagrama de Casos de Uso presenta la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios u otros sistemas.

A partir de la identificación del actor que interactúa con el componente informático, así como la recopilación del conjunto de funcionalidades, que a su vez han sido agrupadas en casos de usos según sus particularidades, se conforma el Diagrama de Casos de Uso que se representa a continuación.

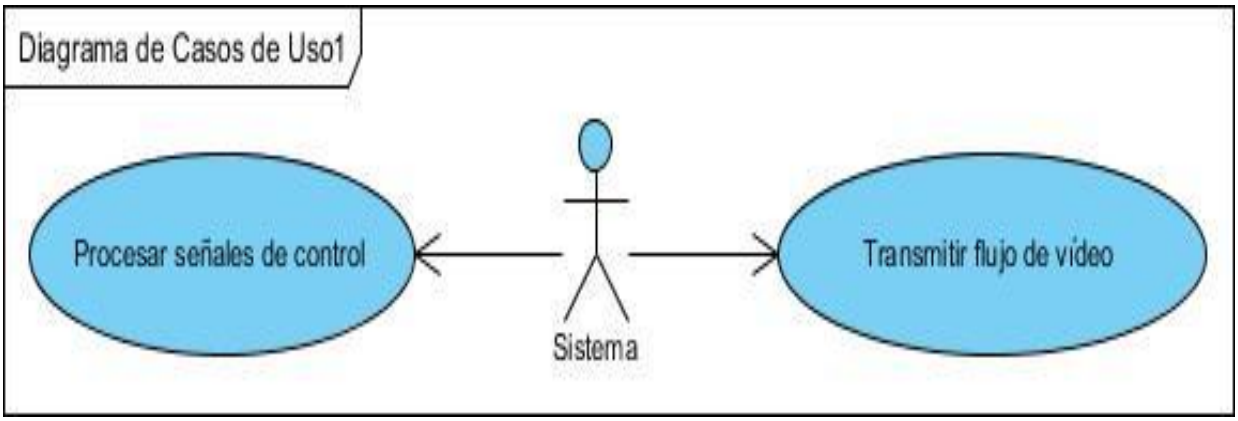

Figura 2: Diagrama de Casos de Uso del Sistema

En la especificación de los Casos de Uso se describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del actor. A continuación se describen los casos de uso asociados al componente.

Descripción del Caso de Uso arquitectónicamente significativo del sistema.

CU1: Transmitir flujo de video.

Tabla 5: Descripción del Caso de Uso Transmitir flujo de video.

Objetivo	Permitir la transmisión del flujo de video.
Actores	Sistema: (Inicia) Iniciar Proceso.
Resumen	El CU inicia cuando el actor, en este caso el Sistema, hace peticiones automáticas al componente para solicitar la transmisión del flujo de video emitido por una cámara IP. El CU termina cuando se inicia la transmisión.
Complejidad	Alta.
Prioridad	Crítica.
Precondiciones	No procede.

Poscondiciones	Transmisión de flujo de video iniciada por el componente a través del protocolo UDP hacia las PCs clientes.		
Flujo de eventos			
Flujo básico Transmitir flujo de video			
	Actor		**Componente**
1.	Indica al componente que comience a transmitir mediante peticiones.		
2.			Recibe la dirección del flujo de video a transmitir.
3.			Captura el flujo de video emitido.
4.			El flujo de video es encapsulado.
5.			Comienza la transmisión por el protocolo UDP del flujo de video. Termina el CU.
Relaciones	**CU incluidos**	No existen.	
	CU extendidos	No procede.	
Requisitos no funcionales	No procede.		
Asuntos pendientes	No procede.		

CU: Procesar señales de control.

Tabla 6: Descripción del Caso de Uso Procesar señales de control.

Objetivo	Recibir y enviar señales de control.
Actores	Sistema: (Inicia) Iniciar Proceso.
Resumen	El CU inicia cuando el actor, en este caso el Sistema, hace peticiones automáticas al componente para solicitar la captura y envío de señales de control, enviadas por una PC cliente y recibidas por una estación de trabajo. El

	CU termina cuando inicia la captura y envío de las señales de control.		
Complejidad	Alta.		
Prioridad	Crítica.		
Precondiciones	No procede.		
Poscondiciones	Recibidas y enviadas exitosamente las señales de control.		
Flujo de eventos			
Flujo básico Procesar señales de control			
	Actor		**Componente**
1.	Indica al componente que comience a procesar las señales de control mediante peticiones.		
2.			Recibe el puerto por donde va recibir las señales de control enviadas desde las PC cliente.
3.			Recibe las señales de control.
4.			Envía las señales de control por un puerto hacia la estación de trabajo.
5.			Comienza la transmisión por el protocolo UDP del flujo de video. Termina el CU.
Relaciones	**CU incluidos**	No existen.	
	CU extendidos	No procede.	
Requisitos no funcionales	No procede.		
Asuntos pendientes	No procede.		

2.5 Modelo del diseño

El modelo de diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizado como una entrada fundamental de las actividades de implementación (Pressman, 2013).

2.5.1 Diagrama de clases del diseño

Los diagramas de clases del diseño son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro. Describen gráficamente las especificaciones de las clases del software y de las interfaces en una aplicación. En su diseño contienen como información clases, métodos, información sobre los tipos de los atributos y dependencias (Larman, 2013).

En la Figura # 3 se muestra la representación del diagrama de clases del diseño.

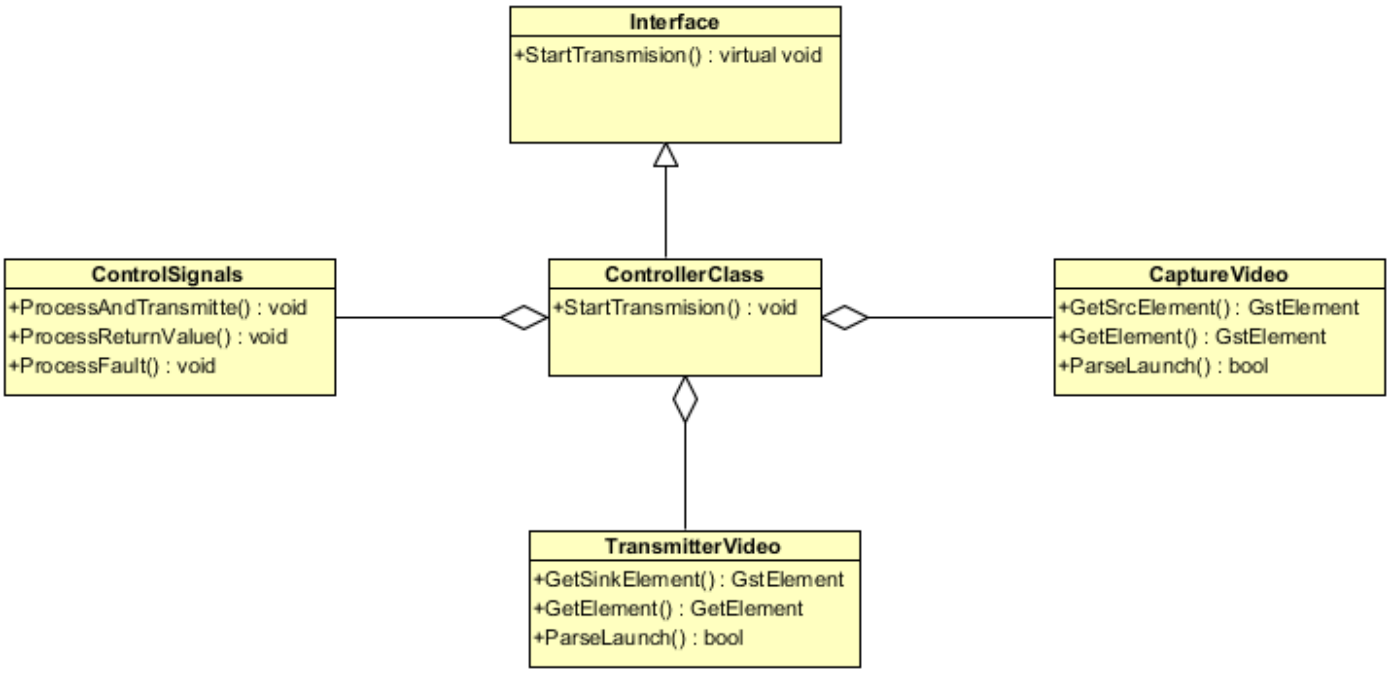

Figura 3: Diagrama de clases del diseño

A continuación se realiza una breve descripción de las clases del diagrama de clases del diseño, representado en la figura anterior.

Nombre: ControllerClass
Descripción: es la clase encargada de manejar los eventos del sistema.

Nombre: TransmiterVideo
Descripción: permite transmitir flujos de video.

Nombre: CaptureVideo
Descripción: es la clase encargada de capturar y procesar el video.

Nombre: ControlSignals
Descripción: permite recibir y enviar las señales de control.

Nombre: Interface
Descripción: es la clase intermediaria entre el actor y las funcionalidades del componente.

2.5.2 Estilo Arquitectónico utilizado

“La arquitectura del software proporciona una visión global del sistema a construir. Describe la estructura y la organización de los componentes del software, sus propiedades y las conexiones entre ellos. Los componentes del software incluyen módulos de programas y varias representaciones de datos que son manipulados por los mismos” (Pressman, 2005). El logro efectivo del diseño de un software que persigue el objetivo de cumplir con los requisitos propuestos, está guiado por una arquitectura que defina los diferentes patrones que brindan un esquema de referencia útil para guiar el desarrollo de software. Posibilitando que todos los integrantes del equipo de proyecto trabajen sobre la misma línea y exista una compatibilidad entre ellos para alcanzar los objetivos trazados.

“Los estilos arquitectónicos son también patrones de construcción y se les conoce como la descripción de una categoría del sistema que contiene: un conjunto de componentes que realizan una función requerida por el sistema; un conjunto de conectores que posibilitan la comunicación, la coordinación y la

cooperación entre los componentes; restricciones que definen cómo se pueden integrar los componentes que forman el sistema; y modelos semánticos que permiten al diseñador entender las propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes" (Pressman, 2005).

En la realización del componente para la transmisión de información de control y video para un Sistema de Laboratorios Virtuales y a Distancia, se empleó una arquitectura basada en componentes. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos (Buschman, 2009).

2.5.3 Estilo arquitectónico basado en componentes

Características del estilo arquitectónico basado en componentes (Buschman, 2009):

- Es un estilo de diseño para aplicaciones compuestas de componentes individuales.
- Pone énfasis en la descomposición del sistema en componentes lógicos o funciones que tienen interfaces bien definidas.
- Define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades.

Principios Fundamentales:

Un componente es un objeto de software específicamente diseñado para cumplir con cierto propósito. Los principios fundamentales, cuando se diseña un componente, es que estos deben ser reusables, sin contexto específico, extensibles, encapsulados e independientes (Buschman, 2009).

Beneficios:

La arquitectura basada en componentes incluye beneficios en la creación y estructura de un software, facilitando la instalación y desarrollo. Tiene las siguientes características:

- Facilidad de desarrollo: Los componentes implementan una interfaz bien definida para proveer la funcionalidad definida, permitiendo el desarrollo sin impactar otras partes del sistema.

- Reusable: El uso de componentes reutilizables significa que ellos pueden ser usados para distribuir el desarrollo y el mantenimiento entre múltiples aplicaciones y sistemas.
- Mitigación de complejidad técnica: Los componentes mitigan la complejidad por medio del uso de contenedores de componentes y sus servicios.
- Reducción de Costos: El uso de componentes de terceros permite distribuir el costo del desarrollo y del mantenimiento.
- Facilidad de despliegue: Cuando una versión de un componente esté disponible, se puede remplazar la versión existente sin impacto en otros componentes o en el sistema.

2.6 Patrones definidos

Los patrones son formas de describir las mejores prácticas, buenos diseños, y encapsulan la experiencia de forma tal que es posible para otros reutilizar dicha experiencia. Constituyen mecanismos cuyo objetivo es la solución de problemas que ocurren repetidamente dentro de un contexto muy bien definido.

2.6.1 Patrones de diseño

A la hora de desarrollar un software es importante el uso de Patrones de diseño, pues 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, constituyen una guía para resolver problemas comunes en la programación (Microsoft, 2003).

Entre los patrones de diseño se utilizaron los GRASP (*General Responsibility Assignment Software Patterns*) y GOF (*Gang of Four*). Los patrones GRASP son patrones generales de software para asignación de responsabilidades y los patrones GOF se clasifican en 3 categorías: creacionales, estructurales y de comportamiento. Para el desarrollo del componente fueron utilizados los siguientes patrones:

Patrones GRASP

Creador: Soluciona el problema de ¿Quién debería ser el responsable de la creación de una instancia de una clase? Guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea

común en sistemas orientados a objetos. El intento básico de este patrón es encontrar un creador que necesite estar conectado al objeto creador en un evento en particular. Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones (Montero, y otros, 2013).

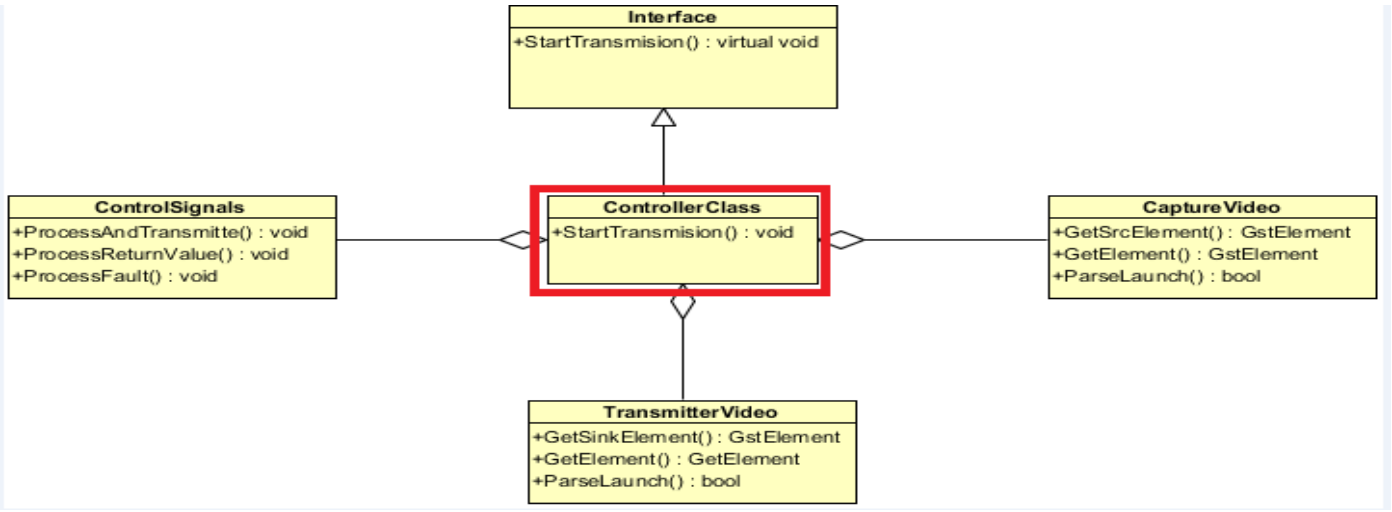

Figura 4: Aplicación del patrón Creador.

Experto: es el encargado de asignar una responsabilidad al experto en información, constituyendo este la clase que cuenta con la información necesaria para cubrir la responsabilidad (Montero, y otros, 2013). En este trabajo, este tipo de patrón es utilizado a la hora de determinar las responsabilidades con las cuales debe de cumplir cada clase y las funcionalidades que realizan cada una de ellas.

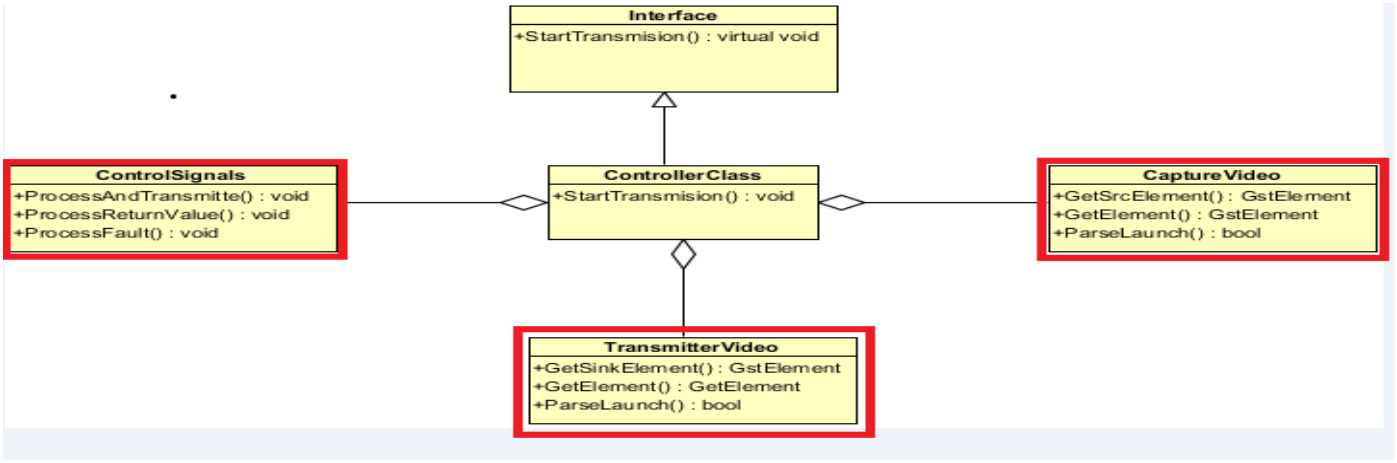

Figura 5: Aplicación del patrón Experto.

Controlador: es el responsable de asignar a clases especificas la responsabilidad de controlar la salida de sucesos en el sistema, permitiendo a través de estas clases interactuar con las demás clases del componente (Montero, y otros, 2013).

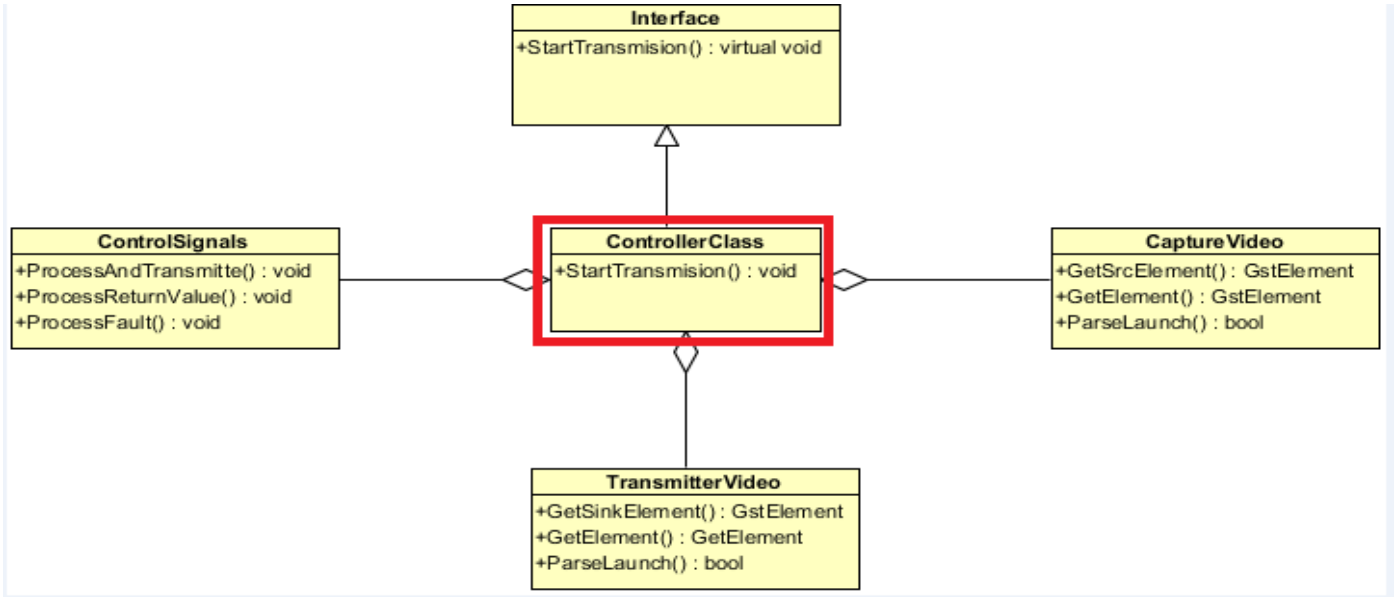

Figura 6: Aplicación del patrón Controlador.

Alta Cohesión: la alta cohesión es la medida en la que un componente se dedica a realizar solo la tarea para la cual fue creado, delegando las tareas complementarias a otros componentes. Una clase debe de

hacer lo que respecta a su entidad, y no hacer acciones que involucren a otra clase o entidad (Montero, y otros, 2013).

Bajo Acoplamiento: soluciona el problema de ¿Cómo dar soporte a las bajas dependencias y al incremento de la reutilización? Plantea tener las clases lo menos ligadas entre sí, de tal forma que en caso de producirse una modificación en alguna de ellas, provoque poca afectación en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las mismas (Montero, y otros, 2013).

Patrones GOF

Constructor o Builder: Es usado para permitir la creación de una variedad de objetos. Con la utilización de este patrón, se reduce el acoplamiento y permite variar la representación interna de estructuras complejas, respetando la interfaz común de la clase Builder (Prieto, 2008).

Fachada o Facade: Simplifica el acceso a un conjunto de clases proporcionando una única clase que todos utilizan para comunicarse con dicho conjunto de clases. El cual se pone de manifiesto en la clase Interface.

Ventajas

- Los clientes no necesitan conocer las clases que hay tras la clase fachada.
- Se pueden cambiar las clases ocultas sin necesidad de cambiar los clientes.
- Sólo hay que realizar los cambios necesarios en la fachada.

2.7 Diagrama de secuencia

Un diagrama de secuencia indica la forma en la que los eventos provocan transiciones de un objeto a otro; representa el comportamiento, describiendo la forma en la que las clases pasan de un estado a otro (Pressman, 2013).

2.7.1 Diagrama de secuencia del CU Transmitir flujo de video

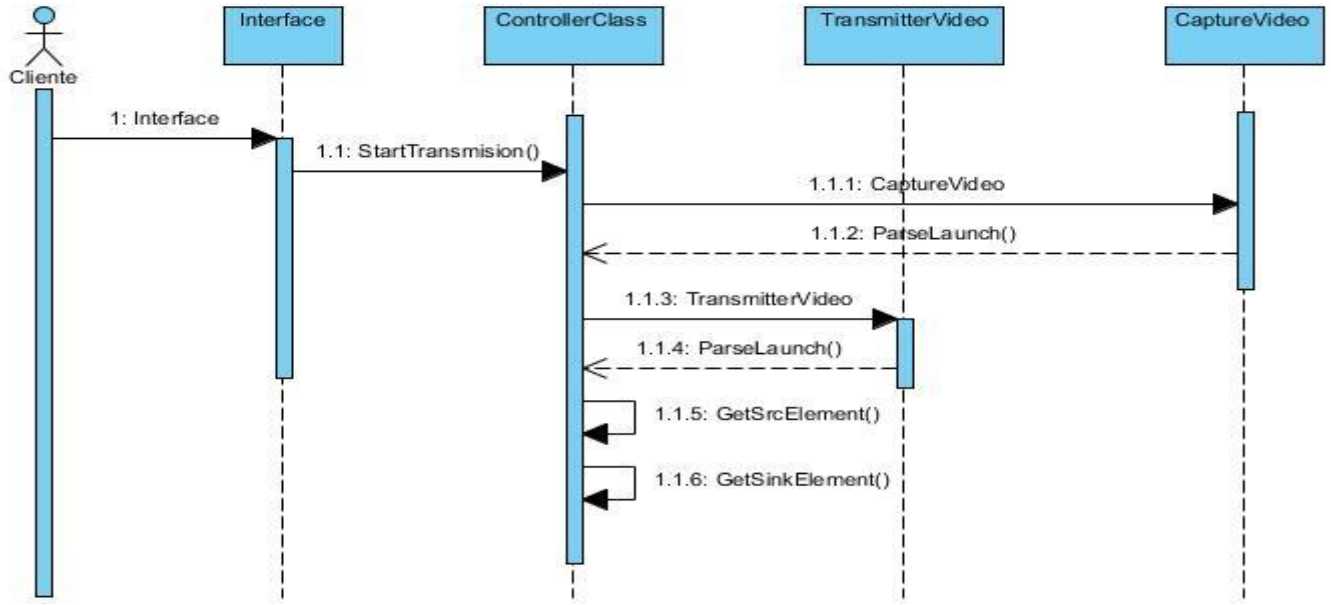

Figura 7: Diagrama de secuencia del CU Transmitir flujo de video.

2.8 Conclusiones del capítulo

Para lograr un mejor entendimiento del funcionamiento del sistema y al no existir un proceso de negocio visible y bien definido, se utilizó como apoyo el modelo conceptual, pues permitió identificar y detallar los principales eventos, términos y conceptos presentes en el entorno donde éste trabajará.

Fueron plasmadas las prestaciones y características del componente para la transmisión de información de control y video, mediante la captura de requisitos. Se identificaron los casos de usos del sistema como elementos claves en el desarrollo del componente para la transmisión de información de control y video para un SLVD, según la metodología Open UP.

En el diagrama de clases del diseño se representaron las clases y las relaciones existentes entre ellas de manera detallada, teniendo en cuenta la arquitectura basada en componentes. Como resultado principal de este capítulo, quedó plasmada la propuesta de solución para el problema actual del sistema, detallada a través del actor, casos de uso y sus diagramas, así como las especificaciones de los casos de uso.

Capítulo 3: Implementación y Prueba

3.1 Introducción

En este capítulo se describe la implementación y validación del sistema. Se realizará el diagrama de componentes y el diagrama de despliegue del componente para la transmisión de información de control y video para un SLVD. Además se desarrollan las pruebas al componente con el objetivo de validar que el sistema cumpla con la calidad requerida.

3.2 Modelo de implementación

El modelo de implementación describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados. Al mismo tiempo describe la dependencia entre los componentes (Ornelas, 2003).

3.2.1 Diagrama de componentes

Un diagrama de componentes muestra la organización y las dependencias entre un conjunto de componentes. Puede ser un tipo especial de diagrama de clases que se centra en los componentes físicos del sistema. Se utiliza para modelar la gestión de la configuración de los archivos de código fuente, tomando como productos de trabajo precisamente estos ficheros (Ferré, y otros, 2011). A continuación se muestra el diagrama de componentes perteneciente a la aplicación a desarrollar.

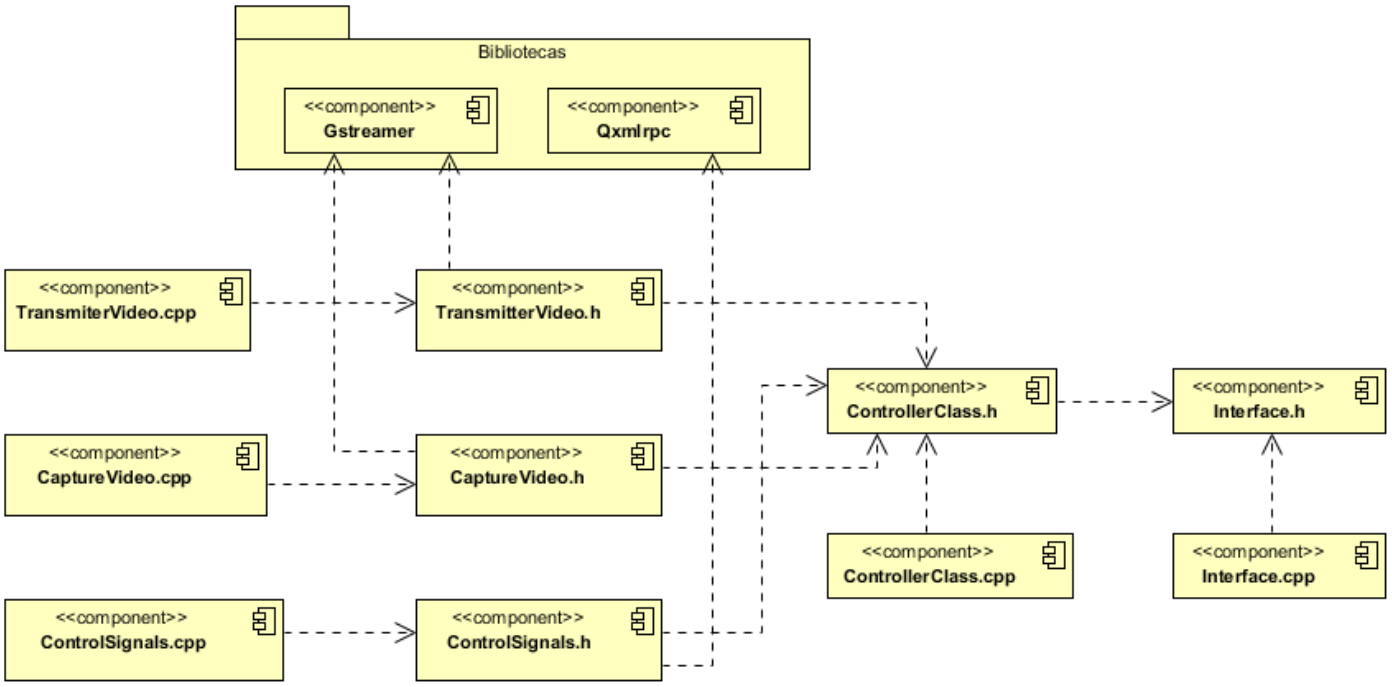

Figura 8: Diagrama de componentes.

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. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar. Los nodos presentan relaciones que representan medios de comunicación entre ellos. El modelo de despliegue además puede describir diferentes configuraciones de red (García, y otros, 2008. La Figura 9 muestra el diagrama de despliegue del componente.

3.3.1 Diagrama de despliegue

Los Diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos (Marca, y otros, 2003).

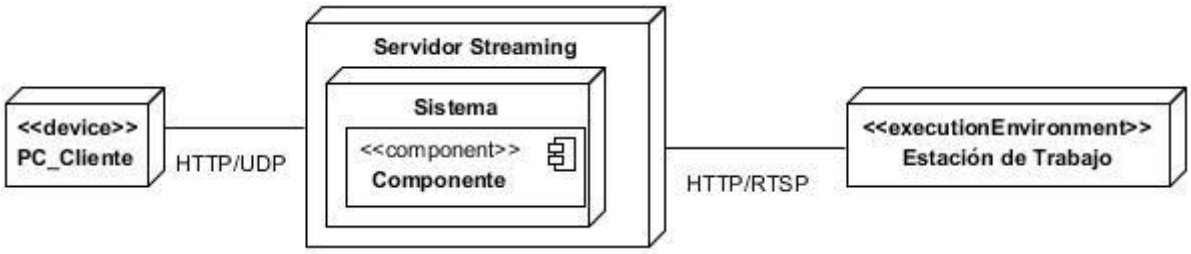

Figura 9: Diagrama de despliegue

La PC_Cliente estará conectada al Servidor Streaming por el protocolo HTTP, que será usado para la transmisión de las señales de control, y por el protocolo UDP que es utilizado para la transmisión de video. A su vez el Servidor Streaming estará conectado con la Estación de Trabajo por medio del protocolo HTTP y del protocolo RTSP utilizado para la captura del flujo de video emitido por una cámara IP.

3.4 Estándar de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Los programadores deben implementar un estándar de forma prudente. Este debe permitir lograr un código fuente con un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez (Microsoft, 2003).

3.4.1 Estilos de codificación utilizados

La utilización de un estándar de codificación constituye una buena práctica de programación que reduce considerablemente la posibilidad de cometer errores. Finalmente se obtiene una aplicación con código legible y comprensible por otras personas del equipo de desarrollo. Para la elaboración del estándar de codificación utilizado, se tuvo en cuenta las características del framework de desarrollo Qt, el entorno integrado de desarrollo Qt Creator y el lenguaje de programación C++. A continuación se muestran algunas de las características con las cuales cumple la implementación del componente:

- En la definición del nombre de las clases se utiliza Upper Camel Case.
- En la nomenclatura de los métodos se utiliza Upper Camel Case.
- Dentro de la estructura del código las líneas pueden tener hasta 150 caracteres.
- Para separar palabras se usa la diferenciación entre mayúscula y minúscula.

Ejemplos de código fuente

```
void ControllerClass::StartTransmision()
{
    gst_bin_add(GST_BIN(capturar->GetElement()),gstreamer->
GetElement());
    gst_element_link(capturar->GetSrcElement(),gstreamer->
GetSinkElement());
    gst_element_set_state(capturar->GetElement
(),GST_STATE_PLAYING);

    //qDebug()<<gst_element_link(capturar->getSrcElement
(),gstreamer->getSinkElement());

    bus = gst_pipeline_get_bus(GST_PIPELINE(capturar->GetElement
()));
    gst_bus_add_watch(bus,process_message,capturar->GetElement
());
    g_object_unref(bus);
}
```

Figura 10: Ejemplo de código fuente

```
#ifndef CONTROLLERCLASS_H
#define CONTROLLERCLASS_H
#include <CaptureVideo.h>
#include <TransmitterVideo.h>
#include <ControlSignals.h>
#include <gst/gst.h>
#include "Interface.h"

class ControllerClass : public Interface
{
public:
    ControllerClass(gchar *url, quint16 listenPort);
    void StartTransmision();

private:
    TransmitterVideo *gstreamer;
    CaptureVideo *capturar;
    ControlSignals *ctrlSignals;
```

Figura 11: Nomenclatura de las clases

3.5 Pruebas de software

Las pruebas realizadas a una aplicación son el proceso que permite verificar y revelar la calidad de un software. Básicamente son una fase en el desarrollo de software donde se prueban las aplicaciones. Pueden integrarse dentro de las diferentes fases del ciclo del software y determinan el nivel de calidad de un producto informático mediante medidas o pruebas que permiten comprobar el grado de cumplimiento respecto a las especificaciones iniciales del sistema (Tenorio, 2010).

Niveles de Prueba

La prueba es aplicada para diferentes tipos de objetivos, en determinados momentos del ciclo de vida del software. Existen diferentes niveles de prueba como: pruebas de desarrollo, independientes, de integración, de sistema y de aceptación. En el desarrollo de la fase de pruebas del componente de transmisión de información de control y video, se aplicará la prueba a nivel de desarrollador para verificar los requerimientos del sistema.

Algunos objetivos de las pruebas

- Ejecución del programa con la intención de descubrir errores.
- Confección de casos de pruebas con altas probabilidades de mostrar un error no descubierto hasta entonces.
- Definición de hasta qué punto las funcionalidades del software funcionan de acuerdo con las especificaciones y cuán cerca se está de alcanzar los requisitos de rendimiento.
- Definición de la fiabilidad del software a partir de los datos obtenidos en las pruebas (Tenorio, 2010).

3.5.1 Prueba de caja blanca

La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control de diseño procedimental para obtener los casos de pruebas (CP). Este método permite obtener los CP que garanticen que (Tenorio, 2010):

- Se ejercitan al menos una vez todos los caminos independientes de cada módulo.
- Se ejercitan todas las decisiones lógicas en sus vertientes verdaderas y falsas.
- Se ejecutan todos los bucles en sus límites y con sus límites operacionales.
- Se ejercitan las estructuras internas de datos para asegurar su validez.

Una de las **técnicas** utilizadas para realizar este tipo de prueba es la del camino básico. Esta técnica permite al diseñador de casos de pruebas obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los CP obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (Tenorio, 2010).

Los pasos a seguir para la realización de las pruebas de caja blanca son los siguientes:

1. Generar el grafo de flujo de datos.
2. Calcular la complejidad ciclomática, V (G).

V (G) = NA (Numero de aristas) – NN (Numero de nodos) + 2.

V (G) = P (Nodos predicados) + 1.

V (G) = Números de regiones.

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, define el número de caminos independientes del conjunto básico de un programa y especifica un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Tenorio, 2010).

3. Determinar los caminos independientes o básicos.
4. Generar un CP para cada camino de ejecución.

Diseño de prueba de caja blanca

Para realizar la prueba de caja blanca con la técnica de camino básico, según los pasos analizados anteriormente, se necesita seleccionar el código fuente al cual se le desea aplicar el procedimiento.

Método StartTransmision (Ver Anexos)

Paso 1: Generar el grafo.

Figura 12: Grafo del Método StartTransmision

Paso2: Calcular la complejidad ciclomática.

V (G) = NA (Número de aristas) – NN (Número de nodos) + 2.

V (G) = 2 – 3 + 2 =1

Paso 3: Determinar los caminos básicos.

CB 1: 1-2-3

Paso 4: Generar caso de prueba para el camino básico.

Caso de prueba para el camino básico 1.

Entrada: gst_bin_add (GST_BIN (capturar ->GetElement()), gstreamer->GetElement());

Resultado esperado: Transmisión iniciada.

Resultado de la Prueba: Satisfactorio.

Método ProcessAndTransmitte (Ver Anexos)

Paso 1: Generar el grafo.

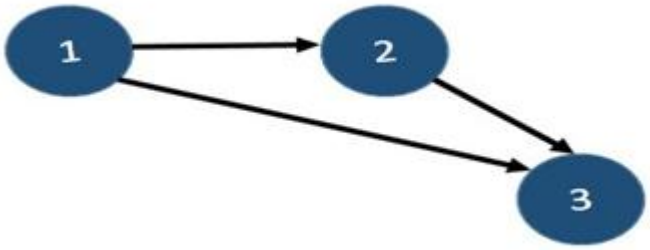

Figura 13: Grafo del Método ProcessAndTransmitte

Paso 2: Calcular la complejidad ciclomática.

V (G) = NA (Número de aristas) – NN (Número de nodos) + 2.

V (G) = 3 – 3 + 2 = 2

Paso 3: Determinar los cambios básicos.

CB 1: 1-2-3

CB 2: 1-3

Paso 4: Generar caso de prueba para el camino básico.

Caso de prueba para el camino básico 1.
Entrada: if (methodName == ¨Transmitir¨).
Resultado esperado: Enviar lista de parámetros.
Resultado de la prueba satisfactorio.

Caso de prueba para el camino básico 2.
Entrada: No se cumple la condición.
Resultado esperado: No envía lista de parámetros.

Resultado de la prueba: Satisfactorio.

3.5.2 Pruebas de aceptación

Las pruebas de aceptación representan aquella fase del ciclo de vida de desarrollo de software en el que el equipo de desarrollo y los usuarios de un sistema de información comprueban que el sistema desarrollado se corresponde con los requerimientos definidos (González, y otros, 2014).

Dentro de las pruebas de aceptación se encuentran las de tipo Alfa y las de tipo Beta, como se muestra a continuación:

Pruebas de aceptación Alfa: es aquella en la que se le entrega a un usuario final todo el producto terminado, junto a su documentación correspondiente para que éste, en presencia del desarrollador y en entornos previamente preparados para el proceso de dichas pruebas, vaya informando de las inconsistencias y errores que detecte.

Pruebas de aceptación Beta: es la que se le hace a los usuarios finales, situados en los puestos de trabajo donde finalmente será el software implantado, para que sea otra vez el usuario y sin contar con la presencia del equipo de desarrollo, el que de nuevo vuelva a emitir informes de resultados e impresiones de la aplicación o software desarrollado (González, y otros, 2014).

En la presente investigación se utilizaron específicamente las pruebas de aceptación de tipo Alfa, en un entorno dispuesto por el cliente, que sirvió para corroborar que el sistema satisface los requisitos definidos (Ver Anexo 4 “Acta de aceptación de la propuesta de solución”).

3.6 Conclusiones parciales

Una vez concluido el proceso de implementación del componente, se realizó la distribución física para su despliegue. Se aplicaron pruebas de caja blanca, obteniendo casos de pruebas que garantizaron que se ejercitara por lo menos una vez todos los caminos independientes de las funcionalidades más importantes de las clases ControllerClass y ControlSignals. Los resultados arrojados por dichas pruebas fueron satisfactorios, lo cual evidencia que el componente está apto para ser utilizado.

Conclusiones generales

Tras haber culminado los tres capítulos con los que cuenta el presente trabajo de diploma, se logró dar cumplimiento a las tareas de la investigación, permitiendo arribar a las siguientes conclusiones.

- El análisis y caracterización del estado del arte actual de los SLVD, permitió afirmar que las soluciones existentes no resuelven la problemática planteada, trayendo consigo la necesidad del desarrollo de la propuesta.
- El uso de la metodología y las herramientas seleccionadas, posibilitó realizar, guiar y documentar el proceso de diseño e implementación de la herramienta para la transmisión de información de control y flujo de video para un SLVD, dándole cumplimiento a los 5 requisitos funcionales definidos y especificados en el presente trabajo.
- El diseño y ejecución de los casos de pruebas permitieron validar el correcto funcionamiento del componente para la transmisión de información de control y flujo de video para un SLVD.

Glosario de términos

HTTP (Hyper Text Transfer Protocolo): protocolo de transferencia de hipertexto.

XML (Extensible Markup Language): es un Lenguaje de Etiquetado Extensible, el cual tiene un papel fundamental en el intercambio de una gran variedad de datos.

SMTP (Simple Mail Transfer Protocol): protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico.

Protocolo: *¨Es un conjunto de normas y procedimientos útiles para la transmisión de datos, conocidos por el emisor y el receptor¨* (Corona 2004).

UDP (Protocolo de datagramas de usuario): UDP es un protocolo del nivel de transporte basado en el intercambio de datagramas que proporciona una sencilla interfaz entre la capa de red y la capa de aplicación. Sirve para el envío de estos a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. No tiene confirmación, ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o de recepción. Este protocolo se utiliza principalmente cuando el orden en que se reciben los mismos no es un factor fundamental, o también cuando se quiere enviar información de poco tamaño que cabe en un único datagrama (Coto 2009).

TCP (Protocolo de Control de Transmisión): Protocolo orientado a conexión, localizado sobre la capa IP (Internet Protocol): Es un protocolo confiable, los datos siempre llegarán de forma ordenada, ya que implementa mecanismos de retransmisión en caso de que el receptor no confirme la recepción de los datos. También implementa mecanismos de control de congestión y control de flujo. Lo que implica, que TCP es más lento que UDP. Normalmente TCP se utiliza para mensajes de control, y no se recomienda para la transmisión de vídeo aunque a pesar de ello se utiliza en algunas aplicaciones de vídeo (Coto 2009).

SOAP (*Acrónimo de Simple Object Access Protocol*), es un protocolo ligero diseñado para el intercambio de información, estructurado en un entorno descentralizado y distribuido. Utiliza XML para la codificación de los mensajes que pueden ser intercambiados a través de varios protocolos de transporte como HTTP y SMTP, entre otros. Es independiente del lenguaje de programación y consta de tres partes: una envoltura

que define un marco para describir lo que está en un mensaje y cómo procesarlo, un conjunto de reglas de codificación para expresar instancias a solicitudes de tipos de datos definidos y una convención para representar llamadas a procedimientos remotos y respuestas.

De manera general, es un protocolo que proporciona un mecanismo estándar para estructurar mensajes utilizando XML, con el fin del intercambiar información con entornos heterogéneos utilizando el protocolo de transporte HTTP y otros.

XML-RPC: es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

Es un protocolo muy simple ya que solo define unos cuantos tipos de datos y comandos útiles, además de una descripción completa de corta extensión. La simplicidad de XML-RPC contrasta con la mayoría de los protocolos RPC, que tienen documentación extensa y requieren considerable soporte de software para su uso (DaveWinder, 2001).

MPEG-4 parte 10/H.264: MPEG-4 parte 10/H.264 es una norma que define un códec de video de alta comprensión. Diseñado para comprimir y descomprimir video digital con el fin de reducir la cantidad de ancho de banda necesario para transmitir y almacenar el video. Cuenta con los mismos elementos o bloques funcionales que sus antecesores, ya que también adopta un algoritmo híbrido de predicción y transformaciones para la reducción de la correlación espacial y de la señal residual.

Ventajas del estándar MPEG-4 parte 10/H.264 frente a sus antecesores

El MPEG-4 parte 10/H.264 es el estándar oficial de comprensión de video más reciente. Ofrece mejoras en la calidad de video, una mayor eficiencia en el proceso de codificación, así como un alto rendimiento en la comprensión y descompresión de video digital, con el fin de reducir la cantidad de ancho de banda necesario para transmitir y almacenar el video. La ventaja más significativa para los sistemas de video IP que utilizan este estándar, es que pueden seguir ofreciendo un video digital de alta calidad y baja latencia, pero con un ahorro de los requisitos de ancho de banda y almacenamiento de entre 25 y 50%. Dicho de otro modo, se puede ofrecer un video de una calidad mucho mayor con el mismo ancho de banda.

Referencias

Abraham, F. 2012. *Desarrollo y aplicación de un sistema de soporte a la decisión diagnóstico cognitivo en web para la gestión de contenidos en un sistema de educación virtual.* 2012.

Buschman, Regine Meunier. 2009. *La Guía de Arquitectura. Versión 2.0.* 2009.

Buschmann, F., et al. 1996. *Pattern Oriented Software Architecture: A System of Patterns.Inglaterra:John Wiley & Sons, 1996. 0471958697.* 1996.

Carrillo Pérez, Isaías, Pérez González, Rodrigo y Rodríguez Martín, Aurelino David. 2008. *Metodología de desarrollo del software.* 2008.

Castellanos, A, y otros. 2004. s.l. : Sistemas de Laboratorios a Distancia (SLD): laboratorio para la enseñanza del control automático a distancia, 2004, Vol. 7. ISSN 1516-084X.

Chacez, R. 2008. *Laboratorios virtuales y a distancia como apoyo a la educación de sistemas robóticos.* 2008. ISSN 1002-2565.

Ching, Ivan Santana. 2010. s.l. : Enero, 2010, Vol. Revista Iberoamericana de Automática e Informática Industrial. ISSN: 1697-7912.

Coromoto, Janeth. 2007. *APLICACIÓN DE LA INGENIERÍA INVERSA DE SOFTWARE SOBRE EL SISTEMA DE APROVISIONAMIENTO DE LA RED BANDA ANCHA DE CANTV.* 2007.

Corona, Adrian Estrada. 2004. 8, s.l. : Revista Digital Universitaria, 2004, Vol. 5. ISSN: 1067-6079.

Cortés, Ing. Anibal Coto. 2009. s.l. : Capa de transporte, 2009.

Cornejo, J. ¿Qué es UML? El Lenguaje de Modelado Unificado. In, 2008

Coto, A. 2009. *Capa de transporte.* 2009. ISSN 8995-5654.

Cikajlo, I. 2012. *Telerehabilitation using virtual reality task can improve balance in patients with stroke. Disability & Rehabilitation, Vol 34, No1. pp 13–18. ISSN 0963-8288. ISSN 1464-5165*

Delgado, Jordan. 2011. Teoría de Programación. [En línea] 2011. http://teoria-de-programacion.globered.com.

DigiartySoftware. 2010. [En línea] 2010. http://www.winxdvd.com/..

DivX. 2012. [En línea] 2012. http://www.divx.com/es..

Dominguez, H, Garcia, J y Ruiz, J. 2007. *Descripción del nuevo estándar de video H.264. . S.l.: s.n.* 2007.

Espinosa, J Koldobika, y otros. 2006. *El Uso de la Tecnología Streaming Multimedia en la Educación Superior On-Line. Bilbao : s.n.,.* 2006.

Ferré, X y Sánchez, I. 2011. s.l. : Desarrollo Orientado a Objetos con UML, 2011.

García, J, Conde, A y Bravo, S. 2008. s.l. : Diseño orientado a objetos, 2008.

Garrido, S. 2009. *Introducción a QT. Programación gráfica en C++ con Qt4. S.l.: s.n.* 2009.

Gimon, Loraine, Gil, Daniel Gustavo y Rossi, Gustavo. 2012. *Especialista en Ingeniería de Software.* 2012. ISSN 9587-2378.

Gómez, P. 2012. *El papel de la autoría en la mediación de los procesos de enseñanza aprendizaje en las unidades didácticas de la educación a distancia.* s.l. : http://revista.inie.ucr.ac.cr/uploads/tx_magazine/papel-autoria-mediación-procesos-enseñanza-aprendizaje-unidades-didácticas-educación-distancia-gomez.pdf, 2012.

González, J y Dominguez, J. 2014. *Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental. España, Universidad de Sevilla.* 2014.

Grau, Xavier Ferré y Segura, María Isabel Sánchez. 2011. *Desarrollo Orientado a Objetos con UML.* 2011.

Informática.com, Grupo de trabajo de La Revista. 2006. La Revista Informática.com. [En línea] 2006. http://www.larevistainformatica.com..

Internet, Software.com.ar. Available from. <http://www.software.com.ar/visual-paradigm-para-uml.html>. [En línea] [Citado el: 24 de 1 de 2013.]

Ivar, Jacobson. 2013. *www.ivarjacobson.com/download.ashx?id=2018.* 2013.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El Proceso Unificado de Desarrollo de Software.*

Jacques Garcías, Fausto Abraham. 2012. *Desarrollo y aplicación de un sistema de soporte a la decisión diagnóstico cognitivo en web para la gestión de contenidos en un sistema de educación virtual.* 2012.

Lab, V. 2014. *Virtual Lab. National Mission on Education through ICT.* s.l. : http://www.vlab.co.in: Ministry of Human Resource Development (MHRD), 2014.

Labster, L. 2014. *Virtual laboratories for High School and College.* s.l. : http://www.labster.com, 2014.

Larman, C. 2003. *UML y Patrones. 2a. S.l.: s.n.* 2003.

Larman, Craig. 1998. *Applying UMLand Patterns. Prentice Hall.* 1998.

—. 2013. UML y Patrones. *UML y Patrones.* [En línea] http://www.ceneinnova.com/eddyesanchez/archivos/ads/UML%20y%20Patrones%20%202da%20Edicion.pdf., 2013.

Leal, Myriam Yadira, Leal, Yuli Carolina y Medina, Leydi Carolina. 2011. *Arquitectura Cliente-Servidor. Cundinamarc : UNIVERSIDAD DE CUNDINAMARC.* 2011.

Loja. 2008. *http://www.utpl.edu.ec/eva/descargas/material/175/G18401.8.pdf.* 2008.

López Carrillo, Angel Fabricio. 2008. *Características Generales de IPTV.* 2008.

Marca, C, y otros. 2003. s.l. : Diagrama de Despliegue, 2003.

Martínez, Alejandro y Martínez, Raúl. 2002. *Proceso Unificado de Rational.* s.l. : Escuela Politécnica Superior de Albacete –Universidad de Castilla la Mancha, 2002.

Martínez, C. 2008. s.l. : La educación a distancia: sus características y necesidad en la educación actual, 2008, Vol. 17. ISSN 1049-9403.

Menéndez, Evelyn. 2014. *Monografías.com.* s.l. : http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml, 2014.

Microsoft. 2003. *Microsoft Developer Network.* [En línea] 2003. https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

Min Qin, Roger Zimmermann. 2008. Streaming en vivo de alta definión-la adquisición y la comprensión de video, transmisión de alta definición. Los Ángeles : s.n., 2008.

Miyara, Federico. 2004. Conversores D/A y A/D. *Electrónica III.* 2004.

MONGE-NÁGERA, J. 1999. *La evolución de los laboratorios virtuales durante una experiencia de seis años con estudiantes a distancia,.* 1999.

Montero, V, y otros. 2013. s.l. : SISTEMA DE SUPERVISIÓN PARA CALL CENTERS.ARQUITECTURA, HERRAMIENTAS Y TECNOLOGÍAS, 2013.

Morimoto, y otros. 2008. *Windows Server 2008 Unleashed. s.l. : Sams Publishing, 2008. 0-672-32930-1.* 2008.

Ornelas, O. 2003. s.l. : Simulador gráfico de algoritmos de programación para computadoras, 2003.

Ornelas, Raquel Ochoa. 2003. *Simulador gráfico de algoritmos de programación para computadoras.* 2003.

Paradigm, V. 2015. *La revista informática.* s.l. : http://www.larevistainformatica.com, 2015.

Penadés, Mª Carmen, Canó, José y Letelier, Patricio. 2003. *Metodologías Ágiles en el Desarrollo de Software.* Universidad Politécnica de Valencia : s.n., 2003.

Poplavskiy,D. 2010. [En línea] 2010. https://packages.debian.org/es/wheezy/libs/libqxmlrpc1.

Potencier, Fabien. 2009. *Symfony la Guía Definitiva. [En linea] www.jesusda.com/docs/ebooks/symfony_guia_definitiva.pdf.* 2009.

Pressman, R. 2005. *Ingeniería de Software. Un enfoque práctico.* s.l. : Mc Graw Hill, 2005. ISBN 9701054733.

—. 2010. *Ingeniería del Software 6ta.* s.l. : http://ebookbrowse.com/roger-s-pressman-ingenieria-del-software-v-ed-pdf-d47183684, 2010.

Pressman, Roger. 2013. *Ingeniería de Software un enfoque práctico***.** *http://es.slideshare.net/jes4791/ingenieria-del-software-un-enfoque-practico.* 2013.

Pressman, Roger S. 2005. *Ingeniería de Software. Un enfoque práctico. .* s.l. : Editorial McGraw-Hill, 2005. ISBN: 9701054733.

Pressman, Rogger. 2005. *Ingeniería de software. Un enfoque práctico.* 2005.

Prieto, F. 2008. *Programación lll.ITI. De Sistemas. Patrones de Diseños. S.l.: s.n.* 2008.

Prieto, Fernando Posada. 2008. Qué es el Streaming. *Diseño de Materiales Multimedia_Web 2.0.* [En línea] 2008. http://www.ite.educacion.es.

Quintero, y otros. 2006. *Evaluación de servidores de streaming de video orientado a dispositivos móviles. Medellín : s.n.,.* 2006.

Reynoso, Carlos y Kicillof, Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Buenos Aires : Universidad de Buenos Aires.* 2004.

Rijo, Daniel. 2004. *Fundamentos de Video Streaming. Montevideo : Universidad de la República Montevideo,.* 2004.

Rischpater, F. 2014. *Application Development with Qt Creator. . S.l.: s.n.* 2014.

Rodríguez, Felix Ivan Romero. 2011. *Plataforma de televisión Informativa Primicia:Implementación del módulo de Redacción informativa PRIMICIA: Implementación del módulo de Redacción .* Ciudad de la Habana : Universidad de las Ciencias Informáticas. : s.n., 2011. s.n Tesis.

S, Christian Van Der Henst. 2001. ¿Qué es el PHP? [En línea] 23 de Mayo de 2001.

Salaverría, Ramón. 2001. *Aproximación al concepto de multimedia desde los planos comunicativo e instrumental.* Navarra : Estudios sobre el mensaje periodístico, 2001.

Somerville, IAN. 2010. *Ingeniería del Software. http://www.filecrop.com/ian-sommerville-ingenieria-de-software-septima-edicion.pdf.html.* 2010.

Stroustrup, Bjarne. 1993. *El lenguaje de programación C++, Segunda Edición.* s.l. : Wilmington, Delaware, E.U.A : Addison-Wesley Iberoamericana, 1993. ISBN 201-60104-4.

Tenorio, L. 2010. *Las Pruebas de Software y su Importancia en las Organizaciones. S.l.* 2010.

Tomasi, Wayne. 2003. *Sistema de Comunicaciones Electrónicas.* 2003.

Torres, L. 2011. *Unidad Docente de Medicina de Familia de Córdoba.* 2011. ISSN 2173-8262.

Trapero, A. 2009. *Importancia de las TIC para la Educación.* 2009. ISSN 1988-6047.

Unilabs, U. 2014. *Universidad Nacional de Educación a Distancia (UNED).* s.l. : http://unilabs.dia.uned.es, 2014.

Vary. 1999. *Informe de la reunión de expertos sobre laboratorios virtuales, Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura. S.l.: s.n.* 1999.

Veranes., Alejandro Miguel Rodríguez. 2012. Componente para localizar regiones de subtitulos en videos. La Habana, Cuba, UCI : s.n., 2012.

WordPress.com. [En línea] http://haltadefinicion.wordpress.com/2008/10/03/%C2%BFque-es-el-formato-mkv/..

Wulf, W. 2000. *El colaboratorio: Un nuevo enfoque hacia la investigación científica. Seminario de la Fundación nacional para la Ciencia. S.l.: s.n.* 2000.

Xiph.org, The xiph open sourse community. 1994 – 2012. [En línea] 1994 – 2012. http://www.xiph.org/ogg/..

Zavala, Esequiel Huavel, y otros. 2011. *Streaming de Video en Vivo por Internet.* Lima : s.n., 2011.

Anexos

Anexo 1. Entrevista a profesores y especialistas.

Soy estudiante de la carrera de Ingeniería en Ciencias Informáticas de la UCI y realizo un proyecto de investigación del que usted forma parte del objeto de estudio. Considero que usted tiene informaciones importantes para el desarrollo de este trabajo, por lo que por favor solicito su colaboración. Le informamos el carácter confidencial de sus respuestas.

Gracias de ante mano.

Objetivo: Diagnosticar el estado actual sobre la utilización del Sistema de Laboratorios Virtuales y a Distancia.

1. ¿Los Sistemas de Laboratorios Virtuales y a Distancia presentan tecnologías?

   _x__ Libres.

   _x__ Privativas.

   Especificar para otros: _____.

2. ¿Mediante qué protocolos de comunicación se lleva a cabo el proceso de transmisión de información de control y flujo de video?

   _x__ TCP.

   ___UDP.

   ___Otros.

Especificar para otros: _____.

3. ¿Qué formas de intercambiar las señales de control son usadas en los SLVD?

   ___ Estandarizada para todas las estaciones.

_x__ Personalizada para cada estación.

Anexo 2. Diagrama de Secuencia Procesar señales de control.

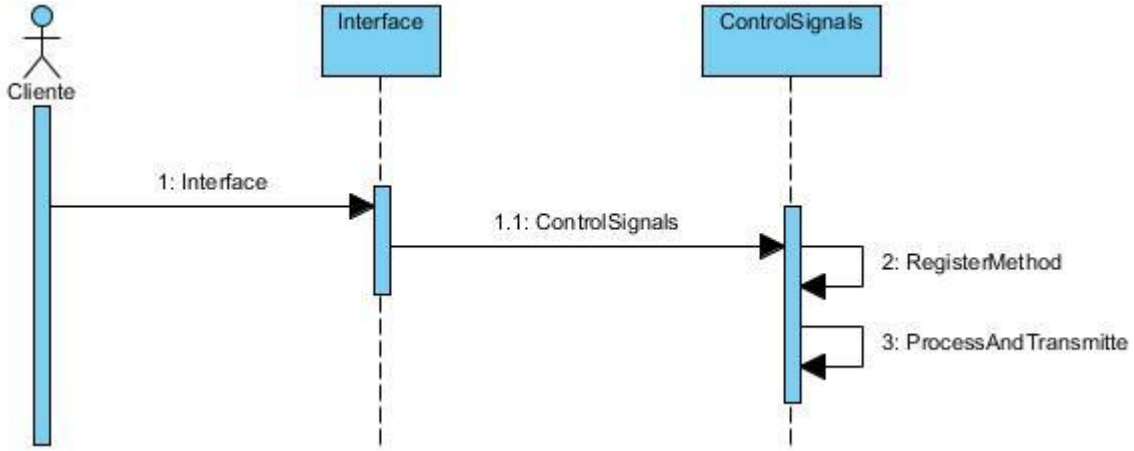

Figura 14: Diagrama de secuencia del CU Procesar señales de control

Anexo 3. Código de métodos utilizados en las pruebas de caja blanca.

Método StartTransmision.

```
voidControllerClass::StartTransmision()
{
 gst_bin_add(GST_BIN(capturar->GetElement()),gstreamer->GetElement());
 gst_element_link(capturar->GetSrcElement(),gstreamer->GetSinkElement());
gst_element_set_state(capturar->GetElement(), GST_STATE_PLAYING);

 //qDebug () <<gst_element_link(capturar->getSrcElement(),gstreamer->getSinkElement());

bus = gst_pipeline_get_bus(GST_PIPELINE(capturar->GetElement()));
gst_bus_add_watch(bus,process_message,capturar->GetElement());
g_object_unref (bus);
}
```

Método ProcessAndTransmitte.

```
VoidControlSignals::ProcessAndTransmitte (intrequestId, QStringmethodName,
QList<xmlrpc::Variant> parameters)
{
If (methodName == "Transmitir") {
qDebug ()<<"LLegooooo";
QList<QVariant>lista_parametros = QList<QVariant> ();
lista_parametros = parameters [0].toList ();
ptrClient->setHost("10.54.14.253"/*"localhost"*/, 8888);
ptrClient->request("Transmitir",parameters);
}
}
```

Anexo 4. Acta de aceptación de la propuesta.

UCI
Universidad de las Ciencias Informáticas

Proyecto Sistema de Laboratorios Virtuales y a Distancia

La Habana, 30 de Junio del 2015
"Año 57 de la Revolución".

ACTA DE ACEPTACIÓN

De una parte, el Jefe del Proyecto Sistema de Laboratorios Virtuales y a Distancia de la facultad 6, de la Universidad de las Ciencias Informáticas, representado en este acto por: Msc. Omar Mar Cornelio y de **otra parte** el estudiante: Yohan Diaz Acosta.

Primero: Que en cumplimiento de los requisitos funcionales han sido efectuadas las implementaciones correspondientes.

CONSIDERANDO: Que los hitos realizados han sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas por **Las Partes**.

CONSIDERANDO: Que los hitos que se han ejecutado cumplen con los requerimientos establecidos.

POR TANTO: Las Partes acuerdan formalizar mediante la presente Acta, la aceptación del producto:

Componente para la transmisión de información de control y video para un Sistema de Laboratorios Virtuales y a Distancia.

Y para que así conste, se extiende la presenta **Acta** en dos (2) ejemplares, rubricados por **Las Partes**.

Entrega

Recibe