UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad #6

TÍTULO: Sistema para el control autónomo de espacio en disco en servidores de media

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR: Alejandro Alvarez Figueras.

TUTOR: Ing. Reinier Pupo Ruiz.

Ciudad de la Habana, 20 de Junio del 2011

"Año 53 de la Revolución"

"Poseemos sin embargo invencibles armas. La principal es la educación....Todo lo transformará y seremos pronto el pueblo más educado y culto del mundo. Ya nadie lo duda dentro y fuera de Cuba".

Dedicatoria

Le dedico este trabajo a lo más grande que tengo en mi vida, a mi madre, a ti te debo todo lo que soy hoy. También a mi papá por siempre apoyarme y ser un ejemplo para mí y a una personita muy especial mi abuela Melba, gracias por ser como una madre para mí.

Agradecimientos

Agradecer a mis padres por siempre darme todo el apoyo y cariño del mundo, por estar presentes cuando lo he necesitados, por guiarme y alimentar mis sueños y esperanzas.

A mi abuelo Fausto ya fallecido que fue como un padre para mí y a mi abuela Melba, que ha sido una madre para mí, no tengo como agradecerte por todo el amor y cariño que me has dado, te quiero.

A mis hermanos por siempre estar a mi lado, en especial a mi hermano menor Leonardo, gracias por darme la dicha de contar con su afecto y cariño.

Quiero también agradecer a mis tíos Pucho y Eduardo mis tías abuelas Kira y Maguie, a mis primos hermanos Karina, Ivan, Reinier y Frank Ernesto que ha sido como hermano mayor para mi, gracias por ayudarme y guiarme durante mi vida.

También agradecer a Arturo, Ivonne y Arturito por ser como una segunda familia para mí, gracias por su apoyo incondicional.

Declaración de Autoría

Ciudad de La Habana, junio, 2011

“Año 53 de la Revolución”

Yo: Alejandro Alvarez Figueras declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del 2010.

Alejandro Alvarez Figueras. **Ing. Reinier Pupo Ruiz.**

_____ _____

FIRMA DEL AUTOR **FIRMA DEL TUTOR**

Datos de Contacto

Autor

Nombre: Alejandro.

Apellidos: Alvarez Figueras.

Correo electrónico: afigueras@estudiantes.uci.cu

Tutor

Nombre: Reinier

Apellidos: Pupo Ruiz.

Correo electrónico: rpupo@uci.cu

Resumen

En la actualidad el almacenamiento, administración y procesamiento de grandes lotes de información es de gran importancia para muchas instituciones. El aseguramiento de los recursos necesarios para el desarrollo de esta actividad es una de las premisas fundamentales para garantizar el almacenamiento de estos grandes cúmulos de información. La Universidad de las Ciencias Informáticas (UCI) cuenta con diversos centros relacionados con la elaboración de soluciones informáticas vinculadas con la captura de flujos de videos, su almacenamiento, procesamiento y posterior utilización. Las unidades destinadas para guardar toda esta información, debido a las grandes cantidades de información que se manejan se saturan con rapidez, impidiendo la realización de otras tares que dependen de la disponibilidad del espacio en los servidores, para el almacenamiento de nuevas medias. Por estos motivos es necesario realizar un constante proceso de monitorización del estado de estos dispositivos, por determinado personal. Además la realización de esta tarea de forma manual es poco eficaz, muy engorrosa para un humano y con un tiempo de respuesta lento ante cualquier suceso que llene el servidor destinado para el almacenamiento de las medias. Por lo que surge la necesidad de desarrollar un Sistema para el control autónomo de espacio en disco en servidores de media. Para su elaboración se realizó un estudio de diversos sistemas de administración de centros de datos a nivel mundial, así como las tecnologías y herramientas actuales más factibles para el desarrollo del sistema. En este trabajo se presentan los resultados que se obtuvo desde un ámbito económico-social en el desarrollo de la solución propuesta.

Abstract

Currently the storage, management and processing of large amount of information has a great importance for many institutions. The ensuring of necessary resources for this activity is a fundamental premise to ensure storage of these large clusters of information. The University of Informatics Sciences (UCI) has various centers related to the development of informatics solutions associated with the capture of video streams, storage, processing and subsequent use. The units used to store all this information, due to large amounts of information are handled, become quickly saturate, preventing the execution of other task that depends on the availability of server space for storage of new media. For these reasons it is necessary a constant process of monitoring the status of these devices, for certain users. Also performing this task manually is inefficient, very tricky for a human and a slow response time to any incident that fills the server for storage of Medias. For these reasons there is a need to develop a system for autonomous control of disk space on media servers. For its development, a study was done of various management systems data centers worldwide and current tools and technologies more feasible to develop the system. This research presents the results obtained in the development of the proposed solution.

Índice de contenido

Índice de Contenido

Índice de Contenido

Índice de Figuras y Tablas

Índice de figura

Índice de tablas

Índice de Figuras y Tablas

Introducción

Introducción

Desde tiempos remotos, el manejo y almacenamientos de los bienes representan tareas de suma importancia para la humanidad. Paralelo a esto, surgió la problemática del espacio disponible para el almacenamiento de los recursos. Diversas fueron las alternativas de solución para optimizar y aprovechar al máximo el espacio disponible y erradicar este problema.

En la actualidad el aprovechamiento al máximo del espacio es un parámetro presente en cada una de las soluciones que el hombre elabora a diario. Así, con el surgimiento de las computadoras y con ellas el manejo y almacenamiento de la información, el hombre se encontró ante la situación de cómo aprovechar y administrar el espacio disponible para almacenar dicha información de forma eficiente.

El desarrollo progresivo de nuevas generaciones de computadoras, más eficientes y con mayor capacidad de procesamiento no eliminó la situación de la administración de los recursos que se almacenan en estas; a pesar de las grandes cantidades de información que pueden almacenar los dispositivos creados para almacenamientos de datos, la gestión manual del espacio de memoria es una tarea difícil y engorrosa, proclive a fallas humanas ya que se necesitarían varias personas para poder accionar un poco más rápido frente a la ocurrencia de determinado evento que requiera la toma de acciones rápidas y precisas.

Todos estos avances en conjunto con otros logros tecnológicos han propiciado un desarrollo vertiginoso de las TICs (Tecnologías de la Información y la Comunicación). Cuba no ha estado exenta de esta revolución tecnológica, el estado cubano consiente de la influencias y facilidades brindadas por las TICs para el desarrollo cultural e intelectual de la humanidad ha realizado innumerables esfuerzos para la informatización del país y el desarrollo de soluciones informáticas para de esta manera insertarse en el mercado del software mundial.

La Universidad de Ciencias Informáticas (UCI) es un reflejo de los esfuerzos de nuestra nación por no estar ajena al desarrollo tecnológico, en esta institución se desarrollan disímiles soluciones integrales de software y se brindan servicios con el objetivo de automatizar muchos procesos y disminuir la complejidad y esfuerzos en la realización de estos. La Universidad cuenta con varios centros de desarrollo, entre los

que se encuentra GEYSED el cual está compuesto por los centros Geoinformática y Señales Digitales.

En dicho centro, diversos son los proyectos que se encuentran relacionados con los procesos de gestión y almacenamiento de contenido multimedia; ya sean archivos multimedia para su posterior utilización, publicación o flujos de videos provenientes de cámaras IP o señales de televisión. Los servidores para el almacenamiento de toda esta información, se saturan debido a los grandes cúmulos de información que manejan, imposibilitando la correcta realización de otras tareas dependientes del buen funcionamiento de los servidores de grabación.

Otro aspecto importante es que el personal vinculado al mantenimiento de los servidores se encuentra obligado a una constante inspección de la disponibilidad de almacenamiento de estos. Además el proceso para la gestión y administración de los espacios en disco en estos dispositivos de manera manual es una tarea muy lenta, difícil y poco segura, por lo que se determina para la investigación el siguiente problema científico:

Problema a resolver: Las personas no son capaces de tomar decisiones rápidamente en caso de que a un servidor de medias se le esté agotando el espacio en disco.

Objetivo General: Desarrollar un sistema que controle la capacidad en disco de un servidor de medias y que tome decisiones dependiendo de reglas.

Objeto de estudio: Proceso de chequeo autónomo de los servidores de medias.

Campo de acción: Espacio en disco de servidores de almacenamiento de medias.

DISEÑO METODOLÓGICO DE LA INVESTIGACIÓN:

Idea a defender: El desarrollo de una aplicación autónoma para el manejo de espacio de memoria en los servidores. Logrando un sistema que pueda adherirse a diversos servidores de medias para su control y monitoreo.

Tareas de la Investigación:

1. Realizar un análisis valorativo de los sistemas de control de espacio en disco en la actualidad.

2. Seleccionar y argumentar la Metodología de Desarrollo de Software a usar en el proceso.
3. Caracterizar las herramientas y tecnologías a utilizar en la realización del sistema.
4. Definir las reglas del sistema.
5. Definir acciones a ejecutarse al cumplirse una regla.
6. Obtener los modelos de dominio, sistema y diseño de la aplicación.
7. Implementar un sistema que controle el espacio en disco en servidores de media.
8. Generar la documentación del proceso de desarrollo de software.

Métodos Teóricos:

Analítico-Sintético: Con este método se analizan los procesos de funcionamiento autónomo en servidores de medias. También es empleado en el análisis y valoración de soluciones similares en búsqueda de características similares y conceptos teóricos que sean de relevancia para la investigación.

Histórico-Lógico: Permite valorar el desarrollo, evolución y tendencias actuales en el proceso de gestión y administración de manera autónoma de servidores de medias.

Modelación: Durante la investigación se utiliza para crear abstracciones que permitan comprender el funcionamiento del sistema mediante los modelos de implementación y despliegue.

Métodos Empíricos:

Observación: Este método es utilizado con la finalidad de caracterizar y evaluar cómo es realizado actualmente todo el proceso de gestión y administración de los servidores de medias y definir los mecanismos para el desarrollo de un sistema que brinde una solución a las necesidades actuales.

CapítuloIo I: Fundamentación Teórica

1.1. Introducción

En este capítulo se identifican los conceptos y elementos teóricos que permiten un mejor entendimiento y sirven de soporte para la realización de un sistema para el control de la capacidad en disco de un servidor de medias. Además se realiza un análisis de soluciones existentes con características similares al prototipo a desarrollar. También se plasma un estudio sobre las principales metodologías y herramientas utilizadas para el desarrollo de la aplicación.

1.2. Conceptos asociados al dominio del problema.

Centro de Procesamiento de Datos (DataCenter): Local donde se concentran todos los recursos necesarios para el almacenamiento y procesamiento de la información de una institución. Esta definición engloba los sistemas y dependencias a través de los cuales:

Los datos son almacenados, tratados y distribuidos al personal o procesos autorizados para consultarlos y/o modificarlos.

Los servidores en los que se albergan estos datos se mantienen en un entorno de funcionamiento óptimo (1).

Captura: En esta investigación el término de captura se utiliza para definir toda operación o conjunto de operaciones que se realizan con la finalidad de obtener datos para transferir hacia un soporte de almacenamiento; puede ser realizar grabaciones de ficheros o archivos de audio y video. Por lo general a este conjunto de acciones se le denomina proceso de captura.

Media: Los medios de comunicación son un instrumento o forma de contenido por el cual se realiza el proceso comunicacional o comunicación. Algunos de los principales medios de comunicación como la televisión y la radio transmiten datos (generalmente ficheros de audio y video) mediante los cuales se establece dicha comunicación con las personas. Por lo que se define como medias todo el contenido de tipo video y sonido usado por los medios de trasmisión para divulgar la información.

1.3. Evolución de los sistemas de administración de centros de datos.

La administración de los recursos en los sistemas de almacenamientos es una tarea de gran importancia y de carácter obligatorio para evitar el derroche y uso ineficientes de los recursos. Es necesario el conocimiento de cada información que se almacena

en la infraestructura, la manera en que se procesa y la infraestructura física con la que se cuenta, para de esta manera conocer en qué estado se encuentra el sistema y poder tomar decisiones de lo que se debe hacer.

El desarrollo de estos sistemas se ha visto frenado por algunos obstáculos como el costo, la complejidad y el dilema del almacenamiento. Steve Duplessie, analista de The Enterprise Strategy Group, plantea que dos obstáculos fundamentales son el costo y la complejidad de la instalación y el uso de dichas herramientas y expresa que:

“LA administración de los recursos es esencialmente gratuita”, declaró. “El costo será compensado con la posibilidad de recuperar los activos de la infraestructura, de tomar decisiones inteligentes sobre políticas futuras y capacitar al personal operativo de las TI”.
Diversos son los sistemas actuales en función de satisfacer estas necesidades entre los que se encuentra los desarrollados por **Avocent Corporation y Synmatec**. Estos garantizan los siguientes aspectos:

Reducir la Complejidad: Los sistemas garantizan de forma rápida y eficiente las tareas de administración. Estos de manera automática inspeccionan el estado del servidor y brinda un informe del estado de este. Tienen la capacidad de detectar errores que solo se descubrirían al encontrarse con estos y facilitan la disminución del personal vinculados con el proceso de administración de los centros de datos (datacenter).

Visibilidad y control de los recursos: Es necesario el conocimiento del estado de los recursos que se manejan, para la realización de futuras acciones, los sistema deben ser capaces de caracterizar el estado de los servidores, en cuanto a capacidad actual y disponible, rendimiento. Tener un control de los recursos almacenados.

Disminución de costos: Debido al aumento de los datos de las empresas promedios anualmente de un 55%, esto provoca un aumento neto en gasto en la compra de dispositivos de almacenamientos. Los sistemas de administración de servidores garantizan la recuperación de los activos subutilizados de la infraestructura logrando una mejor planificación de los recursos y disminuyen así los gastos, facilitan la toma de decisiones inteligentes sobre políticas futuras a tomar en la adquisición de activos, según el estado del servidor. También disminuyen el costo correspondiente al personal utilizado en la administración de estos sistemas.

1.4. Estado del arte de los sistemas de administración de centros de datos.

Capítulo I: Fundamentación Teórica

Durante la investigación se analizaron diversas aplicaciones desarrolladas a nivel mundial con cierto grado de similitud con el sistema a desarrollar, entre las que se encuentran:

Veritas CommandCentral Storage: Es la solución de administración de recursos de almacenamiento de Symantec, que ofrece un control altamente centralizado en entornos de almacenamiento heterogéneo. Un enfoque centralizado de la administración del almacenamiento aumenta la responsabilidad del departamento TI[1] y su capacidad para satisfacer las necesidades de almacenamiento de un conjunto de aplicaciones variado para los diversos sectores a los que asiste.

Veritas CommandCentral Storage permite:

- **La capacidad de almacenamiento, planificación y previsión:** Permite conocer cuánto espacio de almacenamiento se dispone y cuánto se necesitará en el futuro, facilita la recuperación del material almacenado perdido o mal utilizado e implementa prácticas de compra según los datos históricos y actuales.

- **El monitoreo centralizado:** Monitorear de manera continua el entorno de almacenamiento (implantaciones matriciales, switches, hosts, aplicaciones, etc.) relacionado con el rendimiento, la conectividad, las estadísticas del entorno y las condiciones del espacio. Informa de cualquier problema relacionado con la disponibilidad o el rendimiento, según las políticas predefinidas.

- **El aprovisionamiento del almacenamiento heterogéneo:** Optimiza el proceso de aprovisionamiento con una sola herramienta de administración en una infraestructura de almacenamiento heterogéneo.

Ventajas claves:

- Visibilidad de la ruta de datos completa desde la aplicación hasta cada unidad de disco individual en entornos de almacenamiento y servidores físicos y virtuales, lo que ayuda a garantizar el rendimiento óptimo y la disponibilidad de las aplicaciones comerciales críticas.

[1] Tecnologías de la Información.

- Identificación del espacio desaprovechado en todo el almacenamiento de la empresa y utilización mejorada, lo que permite una mejor planificación de la capacidad y reduce los gastos.
- La aplicación de políticas de servicios de calidad a toda la ruta de datos ofrece a los administradores una visibilidad completa de la relación entre las aplicaciones y los recursos.
- Genera transparencia en los procesos operativos ayudando a identificar la propiedad de los datos (2).

Avocent Data Center Planner: Es una nueva fase de Merger Point Infrastructure Explorer, este permite a la organización obtener grandes ventajas y mejorar las condiciones para lograr la consolidación, virtualización y los objetivos de eficiencia energética. Este permite una gestión de centros de datos con información completa y precisa acerca de donde los dispositivos y equipos están situados, sus capacidades actuales y proyectadas de crecimiento.

Blake Carlson vicepresidente de administración de producto de Avocent y los Servicios de negocios de Emerson Network Power plantea que: Avocent Data Center Planner le da a los administradores de los centros de datos un sistema de almacenamiento físico para los equipos de los centro de datos (ubicación, espacio, dispositivo de energía, conectividad y redes). Esta capacidad central es la base de todos los otros procesos y funciones de la solución y hace posible el uso eficiente de recursos y la optimización de la infraestructura del centro de datos (3).

Otras herramientas desarrolladas son: Dsview3 Managent Software y Dsview3 power Manager Software.

1.5. Metodologías, herramientas y tecnologías a utilizar para el desarrollo de la solución.

1.5.1. Metodologías de desarrollo.

Durante el proceso de desarrollo del software se realizan determinadas actividades siguiendo un grupo de fases y etapas en las que se van obteniendo los componentes de software que dan lugar a un producto final. La realización de estas tareas se basa en un grupo de técnicas y herramientas que guiaran el proceso de desarrollo de software. Por lo que se puede decir que en un proyecto de desarrollo de software la metodología brinda una guía y organización para el desarrollo del producto.

Capítulo I: Fundamentación Teórica

Las metodologías de desarrollo de software se dividen en dos grandes grupos: Metodologías Ágiles de Desarrollo y Metodologías Tradicionales o Robustas, por lo que para el desarrollo del sistema se analizan los máximos exponentes de cada grupo: XP (Extreme Programming, Programación Extrema) y RUP (Rational Unified Process, Proceso Unificado de Desarrollo).

XP: Es considerada la metodología ágil que más popularidad a ha alcanzado entre los desarrolladores de software. La programación extrema se diferencia de las demás metodologías de desarrollo de software porque hace especial énfasis en la adaptabilidad y no en la previsibilidad durante el desarrollo del software. Esta engloba un conjunto de prácticas y técnicas encaminadas hacia dos principios fundamentales: obtener un producto con algunas funcionalidades de manera rápida y extrema y a la vez incrementalmente añadir nuevas funcionalidades que aparecerán con el análisis de nuevos requisitos durante el desarrollo del ciclo de vida del software; producto de la constante comunicación entre los clientes y el equipo de desarrollo, propiciando una favorable retroalimentación entre los mismo (16).

Los principios básicos de la programación extrema son: Simplicidad, Comunicación, Retroalimentación y Coraje.

Simplicidad: La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento. La refactorización del código logra la simplicidad de este debido a que está expuesto a constantes cambios aumentando gradualmente su complejidad. También la simplicidad es aplicada a la documentación, de ésta manera el código debe comentarse en su justa medida, intentando eso sí que el código esté autodocumentado. La simplicidad es aplicada también en la autoría colectiva y la programación en parejas lo que facilita un profundo conocimiento del proyecto en desarrollo durante su ciclo de vida.

Comunicación: La comunicación se realiza de diversas maneras. Para los programadores la simplicidad del código los hace más entendible y de fácil comprensión. El código autodocumentado es más seguro que los comentarios ya que estos pueden quedar desfasados ante cambios del código. Existe una comunicación constante entre los colectivos del grupo, manteniendo buenas relaciones interpersonales, producto de la programación en dúos. El cliente forma parte del equipo de desarrollo lo que brinda una retroalimentación cliente-desarrolladores.

Retroalimentación: La integración del cliente al equipo de trabajo es esencial, eso facilita una opinión en tiempo real del estado del proyecto. Se establecen reglas

centradas a las necesidades del cliente, los ciclos cortos tras los cuales se muestran resultados evitan que los programadores tengan que rehacer partes que no cumplan con los requisitos y se centren en lo verdaderamente importante. También el código representa una fuente de retroalimentación, las pruebas unitarias informan sobre el estado de salud del código. Ejecutar las pruebas unitarias frecuentemente permite descubrir fallos debidos a cambios recientes en el código.

Las características fundamentales del método son:

- Desarrollo iterativo e incremental: nueva mejoras de manera consecutiva.
- Pruebas unitarias[2] continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión[3].
- Programación en parejas.
- Integración del cliente o usuario al equipo de trabajo.
- Corrección de todos los errores antes de añadir nueva funcionalidad.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenimiento pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, éste método promueve que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados (9).

RUP: Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Se centra fundamentalmente en los procesos, tareas a realizar y herramientas a utilizar.

[2] Prueba Unitaria: Es una forma de probar el correcto funcionamiento de un módulo de código, mediante casos de prueba extremos.

[3] Prueba de Regresión: Tipo de pruebas de software que intentan descubrir las causas de nuevos errores, carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software.

Los cinco principios claves de RUP:

Adaptar el proceso: El proceso deberá adaptarse a las características propias del proyecto u organización para el cual se está desarrollando el software. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

Balancear prioridades: Los requerimientos de los diversos participantes en ocasiones pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga las necesidades de todos los inversores del proyecto de manera equilibrada, lo que permitirá la corrección de futuros desacuerdos.

Elevar el nivel de abstracción: Éste principio dominante motiva el uso de conceptos reutilizables tales como patrón del software o marcos de referencia (Frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requerimientos y sin comenzar desde un principio pensando en la reutilización del código. Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Estas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con el lenguaje UML.

Enfocarse en la calidad: El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

Colaboración entre equipos: Debe haber una comunicación fluida para coordinar requisitos, desarrollos, evaluaciones, planes y resultados.

Demostrar valor iterativamente: Los proyectos se entregan, aunque sea de una forma interna, en etapas iteradas. En cada iteración se evaluará la calidad y estabilidad del producto y se analizará la opinión y sugerencias de los inversores.

Principales características:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo del software de forma iterativa

- Administración de requisitos
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software (9)

Se seleccionó RUP como metodología de desarrollo a utilizar debido a las características actuales del proyecto. Además al no contar con una retroalimentación constante con el cliente del producto, no resulta factible aplicar como metodología XP, ya que no sería conveniente optar por la adaptabilidad en vez de la previsibilidad en el desarrollo de software que se alcanza con RUP. Además RUP soporta UML para el modelado y la herramienta CASE a utilizar, Enterprise Architect, lo soporta completamente.

1.5.2. Tecnologías a usar para el desarrollo del sistema.

Las tecnologías a usar para el progreso de una aplicación están estrechamente relacionadas con el lenguaje que se seleccione, para llevar a cabo el proceso de desarrollo. Diversos son los lenguajes existentes tales como JAVA, C# 4.5 y C++ entre otros. El desarrollo de la investigación persigue lograr un sistema que sea portable para varios sistemas operativos, lo que facilitó la selección de **C++** como lenguaje de desarrollo después de un análisis de las diversas características que brinda las cuales se exponen a continuación.

C++, es un lenguaje imperativo orientado a objetos, derivado del C y fuertemente tipado. Soporta multitarea mediante clases. Permite modularidad (vía ficheros: includes). Es un lenguaje muy eficaz en cuanto a rapidez y uso de memoria en las aplicaciones que se obtienen con este. Es muy flexible por lo que permite programar con múltiples estilos, como por ejemplo el estructurado "no llevado al extremo" (permitiendo ciertas licencias de ruptura), el orientado a objetos (POO). Además genera programas compactos y rápidos. El código es portable, es decir, un programa ANSI en C o C++ podrá ejecutarse en cualquier máquina y bajo cualquier sistema operativo (1).

1.5.3. Herramientas.

Para el desarrollo del sistema es necesario la utilización de varias herramientas que se agrupen en grandes grupos en relación con las funciones que estas realizan:

Ingeniería y documentación: Más conocidas por herramientas CASE[4]. Las más reconocidas en éste campo son Rational Rose, Enterprise Architect y Visual Paradigm.

Rational Rose: Es una herramienta CASE desarrollada por la International Business Machines (IBM)[5]. Permite el modelado y la visualización de elementos enmarcados en todo el ciclo de vida del software, lo que favorece la construcción del mismo. Utiliza modelado UML para trabajar en diseños de base de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos. Integración con otras herramientas de desarrollo de Rational. Es una herramienta propietaria que utiliza como plataforma a Windows, en la actualidad goza de gran popularidad (13).

Enterprise Architect 6.5: Es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambio. Entre las principales características que brinda se pueden mencionar:

- Crea elementos del modelo UML para un amplio alcance de objetivos.
- Ubica esos elementos en diagramas y paquetes.
- Documenta los elementos que ha creado.
- Genera código para el software que está construyendo.
- Realiza ingeniería directa e inversa de código en lenguajes como ActionScript, C++, C#, Java, PHP, entre otros.
- Soporta todos los diagramas y modelos del UML. Puede modelar procesos de negocio, sitios web, interfaces de usuario, redes, configuraciones de hardware, mensajes y más. Estima el tamaño del proyecto en esfuerzo de trabajo en horas. Captura y traza requisitos, recursos, planes de prueba, solicitudes de cambio y defectos (14).

[4] CASE: Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador.

[5] Empresa conocida como Gigante Azul, se dedica a la fabricación y comercialización de productos informáticos.

Visual Paradigm: Es una herramienta CASE que brinda sustento al desarrollo de software orientado a objetos mediante UML. Ofrece soporte para el modelado y visualización de los componentes necesarios, durante todo el ciclo de vida del software. Constituyendo una plataforma para el desarrollo de sistemas con bajos costos y con calidad. Contiene un diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad y posee la Capacidades de ingeniería directa (versión profesional) e inversa (10).

La herramienta CASE seleccionada por las políticas del proyecto es Enterprise Architect6.5. Esta es una herramienta de gran flexibilidad ya que soporta todos los diagramas y modelos de UML. También realiza ingeniera inversa y directa a varios de los lenguajes entre los que se encuentra C++ el cual se seleccionó para el desarrollo del sistema; lo que facilitará la generación de código y agilización en el proceso de desarrollo.

1.5.4. IDEs de desarrollo.

La selección de los IDEs de desarrollo, deben estar vinculados con el lenguaje de programación seleccionado para el desarrollo, como el lenguaje de desarrollo seleccionado es C++ se selecciona el siguiente IDE de desarrollo:

QTCreator: Es un IDE (entorno de desarrollo integrado) multiplataforma desarrollado por Nokia. Este posee un editor de código con soporte para C++, QML y ECMAscript. También brinda una ayuda sensitiva al contexto y un resaltado de sintaxis y el autocompletado de código. Brinda una GUI (Interfaz Gráfica de Usuario) integrada y un diseñador de formularios muy eficiente que son totalmente funcionales y se pueden visualizar inmediatamente (1).

1.6. Conclusiones

En este capítulo se han abordado diferentes conceptos relacionados con el dominio del problema que ayudarán a la comprensión de la investigación. Conjuntamente se realizó una descripción del objeto de estudio, tratando los aspectos más importantes. Se puede afirmar además que de las soluciones existentes estudiadas, pueden servir de ayuda a la hora de implementar el sistema, pero la adquisición de las mismas no resulta factible en cuanto a costo y tecnologías utilizadas para el almacenamiento de medias, quedando demostrado de esta manera la necesidad de crear un Sistema de control autónomo de espacio en disco para servidores de medias.

Además a partir del análisis y las comparaciones de varios lenguajes de programación, herramientas case, entornos de desarrollo integrado y metodologías de desarrollo de software, para el desarrollo del sistema de control autónomo se logró conocer las potencialidades de C++ para el desarrollo de aplicaciones de escritorio en función de alcanzar un alto rendimiento en el uso de memoria y rapidez. La caracterización de varias metodologías de desarrollo nos permitió definir la más adecuada para el desarrollo del sistema (RUP), lo que nos permitió apreciar la necesidad durante el desarrollo de software de un proceso que guie de forma ordenada las actividades del equipo, asigne tareas y responsabilidades en función de obtener un producto final con la calidad requerida.

Capítulo II: Características del sistema

2.1. Modelo del Dominio.

El modelo de dominio es una representación estática de los objetos reales representados en el entorno del proyecto. Este permite mostrar de manera visual los principales conceptos que se manejan en el dominio del sistema en desarrollo. El modelo conceptual “como también se le conoce” es una representación de las clases conceptuales del mundo real y no de componentes de software. Este puede considerarse como una recopilación de información representada visualmente sobre las abstracciones relevantes, información y conceptos del dominio, lo que facilitará una mejor compresión acerca de los conceptos utilizados por los usuarios, los conceptos con los que trabajan y con los que deberá trabajar la aplicación a desarrollar.

2.1.1. Conceptos fundamentales.

Para facilitar la compresión del modelo de dominio a continuación se proporciona un marco conceptual de las definiciones identificadas.

Sistema Autónomo: Sistema con independencia en la toma de decisiones y ejecución de acciones, ante diversos sucesos desencadenados por acciones críticas que influyen directamente sobre los recursos que maneja el sistema.

Unidad de Almacenamiento: Dispositivos de hardware empleado para el almacenamiento y procesamiento de información.

Registro de configuración: Conjuntos de reglas y acciones almacenadas que brindan la información necesaria para garantizar la autonomía del sistema.

Gestor: Sistema externo de control y procesamiento de información referente a todos los recursos que maneja el sistema.

Cliente: Personal vinculado con el control y administración del sistema.

2.1.2. Diagrama del Modelo de Dominio.

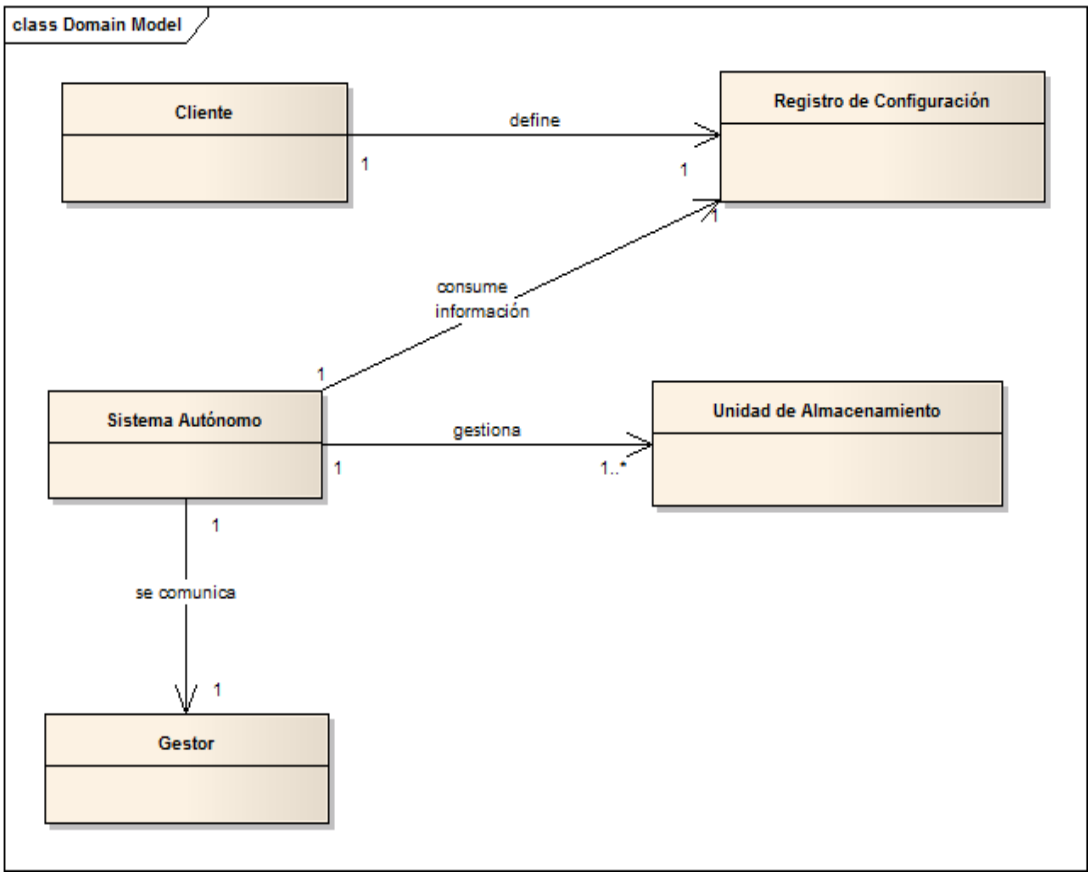

Figura 1: Modelo de dominio.

2.2. Breve descripción del sistema.

El sistema a desarrollar tiene como objetivo fundamental la gestión de las unidades de almacenamiento para medias. Durante el proceso de grabación de los flujos de videos provenientes de las cámaras IP es necesario garantizar la disponibilidad de los dispositivos de almacenamiento, de manera que la grabación de los flujos capturados pueda realizarse sin ningún tipo de afectación. El desarrollo de este sistema de monitoreo y administración de manera autónoma facilitará la ejecución de estas tareas; el sistema administrará los recursos almacenados y tomara decisiones de manera independiente ante diversas situaciones críticas que interrumpan el proceso de grabación, garantizando de esta manera la disponibilidad de los recursos para almacenamiento durante todo el proceso de grabación.

Estas acciones realizadas por el sistema serán definidas por el usuario, el mismo podrá definir estas reglas estableciendo las acciones que desea realizar ante determinado evento y cuando se ejecutará dicha acción. También el sistema brindará

una información completa y detallada sobre los recursos que maneja y el estado de respuesta actual del sistema ante un determinado suceso.

2.3. Requerimientos del Sistema.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

2.3.1. Requerimientos funcionales del Sistema Autónomo.

Requerimientos funcionales del Sistema Autónomo.

RF1 Comunicarse con sistemas externos.
RF1.1 Intercambiar información con el Gestor.
RF1.2 Notificar al Gestor sobre cambios sobre los recursos que maneja el sistema.
RF2 Monitorizar de acciones periódicas.
RF3 Detectar los procesos de grabaciones.
RF4 Monitorizar las unidades de almacenamiento.
RF5 Gestionar medias almacenadas.
RF5.1 Eliminar medias almacenadas.
RF5.2 Mover medias almacenadas.
RF6 Filtrar reglas de configuración.

Requerimientos funcionales del cliente Administrador.

RF7 Gestionar reglas de configuración.

RF7.1 Adicionar reglas de configuración.
RF7.2 Eliminar reglas de configuración.
RF7.3 Editar reglas de configuración.

RF8 Definir parámetros de configuración de monitorización.
RF9 Definir parámetros de configuración para la conexión con sistemas externos (Gestor).
RF10 Mostrar estado de las unidades de almacenamiento.
RF11 Gestionar Medias almacenadas manualmente.
RF11.1 Eliminar medias almacenadas.
RF11.2 Mover medias almacenadas.

2.3.2. Requerimientos no funcionales del sistema.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Son características que hacen al producto atractivo, usable, rápido o confiable. Estos requerimientos se agrupan en varias categorías:

Requerimientos no funcionales del cliente Administrador.

RNF 1 El Cliente debe tener una interfaz gráfica, visualmente atractiva para el usuario.

RNF 2 El Sistema debe facilitar al usuario mensajes que lo ayuden a llevar a cabo la tarea que realiza.

RNF 3 Se debe hacer uso de botones, con imágenes que indiquen de modo intuitivo la función que realizan.

RNF 4 Los controles visuales deben mostrar mensajes que indiquen su función.

RNF 5 La configuración predeterminada debe brindar buena comodidad.

RNF 6 Los parámetros de funcionamiento del módulo deben ser configurables.

2.3.3. Requerimientos de Fiabilidad.

RNF 7 El sistema debe estar en funcionamiento de forma permanente.

RNF 8 Debe funcionar de manera independiente, dándole cierto grado de autonomía al sistema.

2.3.4. Requerimientos de Diseño e Implementación.

RNF 9 Debe ser implementado en C++ ajustándose a las funcionalidades de QTCreator.

2.3.5. Requerimientos de Interfaz de usuario.

RNF 10 Se requiere que el Cliente tenga una interfaz gráfica que permita la interacción con el sistema brindado toda la información referente a los recursos que maneja el sistema.

RNF 11 Se requiere una interfaz ligera, que permita de manera sencilla configurar los parámetros de conexión y monitoreo del sistema.

RNF 12 Se requiere una interfaz gráfica que permita gestionar las reglas del sistema antes situaciones críticas.

2.3.6. Requerimientos de Funcionamiento.

RNF13 Usar como sistema operativo Windows XP SP2 o superior y Linux.

2.4. Definición de Casos de Usos.

2.4.1. Definición de los Actores.

Actores	Descripción
Cliente	Rol encargado de velar por el buen funcionamiento de sistema, configurarlo y ejecutar todas las tareas que este brinda.
Monitor de grabación	Rol ficticio para representar el inicio del proceso de grabación como iniciador de algún caso de uso.
Temporizador	Rol ficticio que representa un intervalo de tiempo tras el cual se realiza una tarea de manera repetitiva.

Tabla 1 Actores del Sistema

2.4.2. Listado de los casos de uso

CU-1	Mostrar estado de Almacenamiento.
Actor	Cliente.
Descripción	Visualizar para el cliente el estado actual de las unidades de almacenamiento.
Referencia	RF10

Tabla 2 CU Mostar Estado

CU-2	Gestionar estado manualmente.
Actor	Cliente.
Descripción	El cliente manualmente selecciona las medias a eliminar o mover hacia otra unidad de almacenamiento.
Referencia	RF11.1, RF11.2

Tabla 3 CU Gestionar Estado

CU-3	Gestionar reglas.
Actor	Cliente.
Descripción	El cliente añade, elimina o edita reglas de configuración.
Referencia	RF7.1, RF7.2, RF7.3

Tabla 4 CU Gestionar Reglas

CU-4	Definir parámetros de monitorización.
Actor	Cliente.
Descripción	El cliente define los parámetros de monitorización para el proceso de detección de flujos de grabación.
Referencia	RF8

Tabla 5 CU Definir parámetros de monitorización

CU-5	Definir parámetros de conexión.
Actor	Cliente.
Descripción	El cliente define los parámetros para la comunicación con sistemas externos (Gestor).
Referencia	RF9

Tabla 6 CU Definir parámetros de conexión

CU-6	Comunicación con sistemas externos.
Actor	
Descripción	El sistema se comunica con el gestor para brindarle información y obtener información de este.
Referencia	RF1.1, RF1.2

Tabla 7 CU Comunicación con Sistemas Externos

CU-7	Monitorizar acciones periódicas.

Actor	Temporizador.
Descripción	Permite al sistema la ejecución de acciones programadas periódicamente de manera autónoma.
Referencia	RF2

Tabla 8 CU Monitorizar acciones periódicas

CU-8	Monitorizar Grabación.
Actor	Monitor de grabación.
Descripción	Permite al sistema conocer el inicio y culminación de los procesos de grabaciones.
Referencia	RF3

Tabla 9 CU Monitorizar grabación

CU-9	Monitorizar Unidad de Almacenamiento.
Actor	Temporizador.
Descripción	El sistema monitoriza periódicamente las unidades de almacenamiento en búsqueda de situaciones críticas.
Referencia	RF4, RF6

Tabla 10 CU Monitorizar unidad de almacenamiento

CU-10	Ejecutar Acciones.
Actor	Monitor de grabación.
Descripción	El sistema elimina o mueve las medias almacenadas en las unidades de almacenamiento, en dependencia de la solicitud que este atendiendo.
Referencia	RF5, RF6

Tabla 11 CU Ejecutar acciones

CU-11	Gestionar Log

Actor	
Descripción	El sistema gestiona los log almacenados correspondientes a los últimos cambios en el sistema y notifica al gestor sobre los cambios efectuados.
Referencia	RF1.2, RF2

Tabla 12 CU Gestionar Logo

CU-12	Notificar Sucesos
Actor	
Descripción	El sistema notifica a sistemas externos sobre cambios realizados en este.
Referencia	RF1.2

Tabla 13 CU Notificar Sucesos

2.5. Diagramas de casos de usos.

Los diagramas de Casos de Uso del Sistema: Están compuestos por varios casos de usos que se relacionan entre si y colaboran para garantizar el funcionamiento del sistema de manera autónoma y satisfacer las necesidades de cada uno de los actores implicados. Este cuenta con dos actores ficticios, un temporizador que inicializara diversas acciones y un flujo de grabación el cual activará un conjunto de acciones encargadas de la monitorización del proceso de grabación en ejecución.

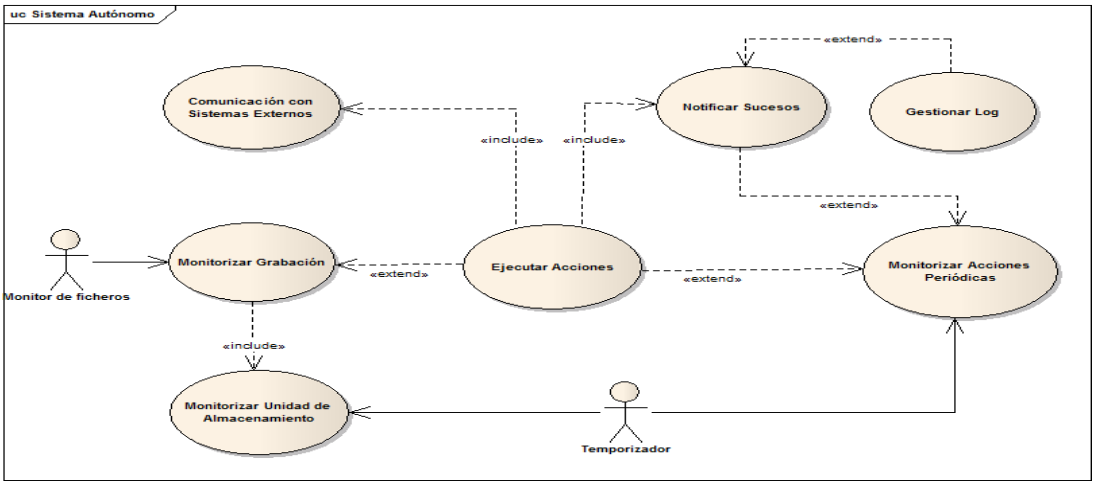

Figura 2: Diagrama de Casos de Usos Del Sistema.

Diagrama de casos de usos del cliente administrador: está compuesto por un único actor(Cliente) y varios casos enfocados de una manera más intuitivas con los requisitos funcionales especificados para dicho usuario y brindar un resultado de valor para el mismo. Lo cual le facilitara al cliente manejar diversos recursos controlados por el sistema y definir diversos parámetros para el funcionamiento del mismo.

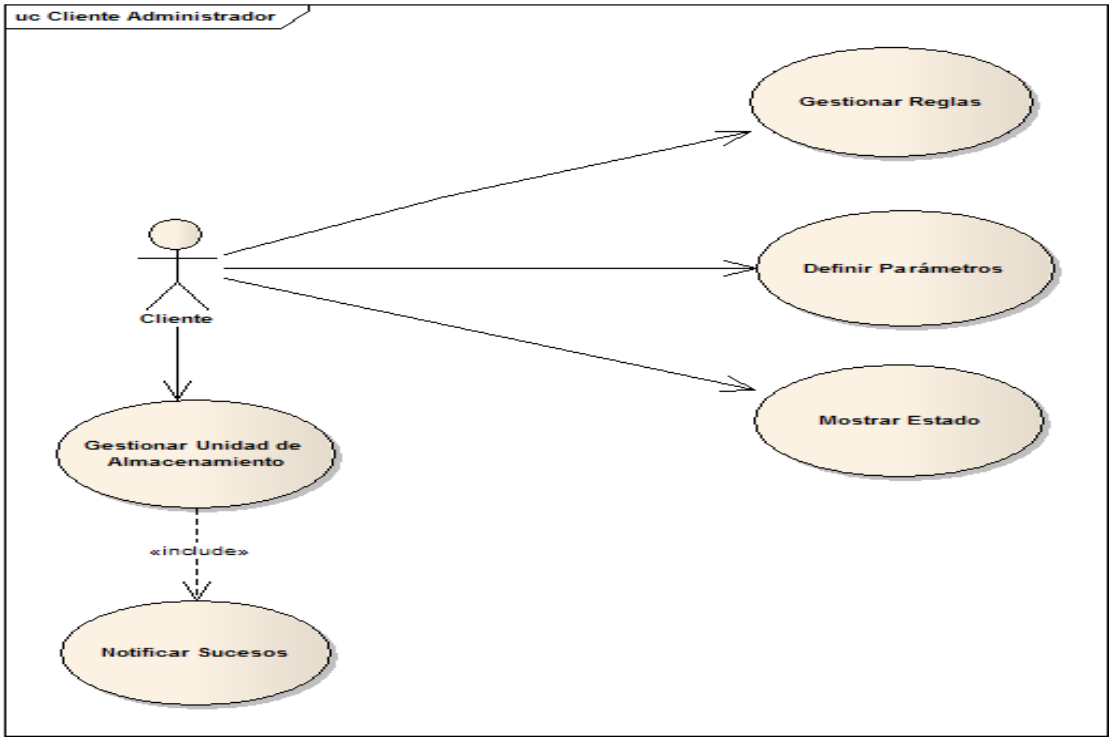

Figura 3: Diagrama de casos de usos del cliente administrador.

2.6. Casos de Usos por ciclo.

2.6.1. Primer Ciclo de Desarrollo.

Código	Nombre del caso de uso.	Justificación de la selección.
CU-3	Gestionar reglas.	La toma de decisiones de manera autónoma del sistema ante situaciones críticas depende de las reglas definidas.
CU-4	Definir parámetros de monitorización.	Para el correcto funcionamiento del sistema es necesario la definición de los parámetros de monitoreo.
CU-5	Definir parámetros de conexión.	Parámetros necesarios para establecer vínculos con el Gestor para el

		intercambio de información.
CU-6	Comunicación con sistemas externos.	Un gran cumulo de información para el correcto funcionamiento del sistema se obtiene del gestor.
CU-7	Monitorizar acciones periódicas.	Garantiza la ejecución de acciones programadas con determinado intervalo de tiempo, para la gestión del espacio en disco.
CU-8	Monitorizar Grabación.	Es lo que garantiza que el sistema entre en un estado de alerta durante el proceso de grabación de los flujos de video, para evitar la ocurrencia de sucesos que interrumpan dicho proceso.
CU-9	Monitorizar Unidad de Almacenamiento.	Detecta e informa la ocurrencia de situaciones críticas.
CU-10	Ejecutar Acciones	Garantiza el espacio en disco para el almacenamiento de las medias.
CU-11	Gestionar Log.	Notifica periódicamente sucesos al gestor que en determinado momento quedaron pendientes.
CU-12	Notificar Sucesos.	Notifica sucesos al gestor que ocurren ante la solución de situaciones críticas.

Tabla 14 CU Críticos

2.6.2. Segundo ciclo de Desarrollo.

Código	Nombre del caso de uso.	Justificación de la selección.
CU-1	Mostrar estado de Almacenamiento.	Permite una visión del estado de los recursos que maneja el sistema.
CU-2	Gestionar estado manualmente.	Permite la gestión del espacio en disco, de manera manual.

Tabla 15 CU Segundo Ciclo

2.7. Casos de Usos Expandidos.

Caso de uso

CU-2	Gestionar Estado Manualmente.
Propósito	Garantizar la liberación de espacio en las unidades de almacenamiento de manera manual.
Actores: Cliente.	
Resumen: El caso de uso se inicia cuando el cliente selecciona la opción de eliminar o mover una media.	
Referencias	RF11.1, RF11.2

Acción del actor	Respuesta del sistema
Escenario Eliminar	
1.- El cliente selecciona la opción de eliminar una media.	2.- El sistema le muestra un formulario con varios parámetros de cada una de las medias para la evaluación del administrador.
3.- El cliente selecciona las medias y selecciona el botón aceptar.	4.- El sistema muestra una advertencia acerca de si desea continuar la operación.
5.-El cliente elige continuar o cancelar la operación.	6.- El sistema elimina la media o cancela la operación.
Escenario Mover	
1.- El cliente selecciona la opción de mover una media.	2.- El Sistema le muestra un formulario con varios parámetros de cada una de las medias para la evaluación del administrador.
3.- El cliente selecciona las medias y selecciona el botón aceptar.	4.- El sistema le muestra una forma con los nuevos posibles directorios de almacenamiento para las medias.
5.-El cliente selecciona el nuevo directorio y oprime el botón aceptar.	6.- El Sistema mueva las medias hacia el nuevo directorio.
Casos de Uso Asociados	CU-6 Comunicarse con sistemas externos. (caso de uso incluido)
Puntos de extensión	Se usa el caso de uso Comunicarse con Sistemas Externos en:

	Escenario Eliminar: Acción 6 Escenario Mover: Acción 6

Tabla 16 CU Expandidos

Caso de uso		
CU-3	Gestionar reglas.	
Propósito	Garantizar la definición de reglas para la toma de dediciones del sistema ante la ocurrencia de situaciones críticas.	
Actores: Cliente.		
Resumen: El caso de uso se inicia cuando el Administrador selecciona la opción de adicionar, eliminar o editar una regla.		
Referencias	RF7.1, RF7.2, RF7.3	
Acción del actor		Respuesta del sistema
Escenario Adicionar		
1.- El cliente selecciona la opción de adicionar una regla.		2.- El Sistema le muestra un formulario para introducir los datos necesarios para la definición de las reglas del sistema.
3.- El cliente introduce los datos y oprime el botón aceptar.		4.- El Sistema comprueba los datos. 5.- El Sistema almacena los nuevos datos si son correctos.
Escenario Eliminar		
1.- El cliente selecciona la regla sobre la cual realiza la operación seleccionada posteriormente.		
2.- El cliente selecciona la opción de eliminar una regla.		3.- El Sistema muestra una advertencia acerca de si desea continuar la operación.
4.-El cliente elige continuar o cancelar la operación.		5.- El sistema elimina la regla o cancela la operación.
Escenario Editar		
1.- El cliente selecciona la regla sobre la cual realiza la operación		

seleccionada posteriormente.	
2.- El cliente selecciona la opción de editar una regla.	3.- El Sistema muestra la información referente a la regla seleccionada.
4.- El cliente modifica los datos y acepta.	5.-El Sistema almacena los datos modificados.
Casos de Uso Asociados	
Puntos de extensión	

Tabla 17 CU Gestionar Reglas

Caso de uso		
CU-4	Definir parámetros de monitorización.	
Propósito	Garantizar la definición de los parámetros para el desarrollo de varias acciones autónomas de monitorización realizadas por el sistema.	
Actores: Cliente.		
Resumen: El caso de uso se inicia cuando el cliente selecciona la opción de definir parámetros de monitorización.		
Referencias	RF8	
Acción del actor		Respuesta del sistema
Escenario Definir parámetros		
1.- El cliente selecciona la opción de definir parámetros de monitoreo.		2.- El Sistema le muestra un formulario para introducir los parámetros de monitorización.
3.- El cliente entra los datos necesarios y acepta la entrada.		4.- El Sistema almacena los nuevos datos.
Escenario Editar parámetros		
1.- El cliente selecciona la opción de editar parámetros de monitoreo.		2.- El Sistema muestra la información referente a los parámetros de monitoreo.
3.- El cliente modifica los datos y acepta.		4.-El Sistema almacena los datos modificados.
Casos de Uso Asociados		

Puntos de extensión	

Tabla 18 Cu Definir Parámetros de Monitorización

Caso de uso	
CU-5	Definir parámetros de conexión.
Propósito	Garantizar la definición de los parámetros para la comunicación con los sistemas externos (Gestor)
Actores: Cliente.	
Resumen: El caso de uso se inicia cuando el cliente selecciona la opción de definir parámetros de conexión.	
Referencias	RF9
Acción del actor	Respuesta del sistema
Escenario Definir parámetros	
1.- El cliente selecciona la opción de definir parámetros de conexión.	2.- El Sistema le muestra un formulario para introducir los parámetros de conexión.
3.- El cliente entra los datos necesarios y acepta la entrada.	4.- El Sistema almacena los nuevos datos.
Escenario Editar parámetros	
1.- El cliente selecciona la opción de editar parámetros de conexión.	2.- El Sistema muestra los datos de los parámetros de conexión.
3.- El cliente modifica los datos y acepta.	4.-El Sistema almacena los datos modificados.
Casos de Uso Asociados	
Puntos de extensión	

Tabla 19 CU Definir Parámetros de Conexión

Caso de uso	
CU-6	Comunicación con sistemas externos.
Propósito	Garantizar la comunicación con los sistemas externos (Gestor)

	para el intercambio de información.
Actores:	
Resumen: El caso de uso se inicia cuando el CU-11(Gestionar Unidad de Almacenamiento) solicita determinada información contenida en el Gestor para su funcionamiento.	
Referencias	RF1.1, RF1.2
Acción del actor	Respuesta del sistema
Escenario Solicitar Información	
	1.- El sistema solicita información para la ejecución de una determinada acción.
2.- El Gestor le brinda la información.	3.- El sistema consume la información.
Casos de Uso Asociados	
Puntos de extensión	

Tabla 20 Comunicación con Sistemas Externos

Caso de uso	
CU-7	Monitorizar acciones periódicas.
Propósito	Ejecutar diversas acciones programadas con determinado intervalo de periodicidad.
Actores: Temporizador.	
Resumen: El caso de uso se inicia cuando el temporizador determinar que una acción periódicas de las definidas debe iniciarse.	
Referencias	RF2
Acción del actor	Respuesta del sistema
Escenario Gestionar Unidad	
1.- El temporizador detecta el inicio de una acción periódica.	2.- El sistema determina el tipo de acción, en el caso de ser de gestión de las unidades de almacenamiento ejecuta la acción en conjunto con la facilidades

	brindadas por el CU-11(Gestionar Unidad de Almacenamiento).
	3.- Configurar el temporizador para la detección de la próxima acción.
Escenario Gestionar Log	
1.- El temporizador detecta el inicio de una acción periódica.	2.- El sistema determina el tipo de acción a ejecutar, si es de gestión de log este reutiliza las acciones definidas en el CU-13(Gestionar Log) y ejecuta la acción.
	3.- Configurar el temporizador para la detección de la próxima acción.
Casos de Uso Asociados	CU-11 Gestionar Unidad de Almacenamiento (extendido). CU-13 Gestionar Log (extendido).
Puntos de extensión	Se usa el caso de uso Gestionar Unidad de Almacenamiento en: Escenario Gestionar Unidad: Acción 2 Se usa el caso de uso Gestionar Log en: Escenario Gestionar Log: Acción 2

Tabla 21 CU Monitorizar Acciones Periódicas

Caso de uso		
CU-8	Monitorizar Grabación.	
Propósito	Detectar y monitorizar los procesos de grabaciones de flujos de datos.	
Actores: Monitor de ficheros.		
Resumen: El caso de uso se inicia cuando un proceso de grabación comienza.		
Referencias	RF3	
Acción del actor		Respuesta del sistema

Escenario Detectar Proceso	
1.- El flujo de grabación se inicia.	2.- El sistema detecta el flujo de grabación. 3.- Ejecución del CU-9 Monitorear Unidad de Almacenamiento. 4- Monitorizar periódicamente el estado del flujo de grabación.
Escenario Situación Crítica	
	1.- El sistema detecta una situación crítica. 2.- Ejecución del CU- 11 Gestionar Unidad de Almacenamiento.
Casos de Uso Asociados	CU- 11 Gestionar Unidad de Almacenamiento (extendido). CU-9 Monitorizar Unidad de Almacenamiento (incluido).
Puntos de extensión	Se usa el caso de uso Monitorizar Unidad de Almacenamiento en: Escenario Detectar proceso: Acción 3. Escenario Situación crítica: Acción 1. Se usa el caso de uso Gestionar Unidad de Almacenamiento en: Escenario Situación crítica: Acción 2.

Tabla 22 CU Monitorizar Grabación

Caso de uso	
CU-9	Monitorizar Unidad de Almacenamiento.
Propósito	Monitorizar y notificar sobre cambios en las unidades de almacenamiento durante la grabación de los flujos de videos.
Actores: Temporizador.	
Resumen: El caso de uso se inicia cuando se pone en funcionamiento el caso de uso Monitorizar Grabación.	

Referencias	RF4, RF6
Acción del actor	Respuesta del sistema
Escenario Monitorizar Espacio	
1.- El temporizador inicia periódicamente una monitorización a las unidades de almacenamiento.	2.- El sistema define las programación de ocurrencia de las situaciones criticas 3 El sistema comprueba el estado de las unidades de almacenamiento en busca de situaciones críticas. 3.- El sistema notifica la ocurrencia de una situación crítica.
Casos de Uso Asociados	CU- 10 Procesar Reglas (incluido).
Puntos de extensión	Se usa el caso de uso Ejecutar Acciones en: Escenario Monitorizar Espacio: Acción 2.

Tabla 23 Monitorizar Unidad de Almacenamiento

Caso de uso	
CU-10	Ejecutar Acciones.
Propósito	Desarrollar diversas acciones para garantizar la disponibilidad de espacio de almacenamiento durante el proceso de grabación.
Actores:	
Resumen: El caso de uso se inicia cuando se dispara una situación crítica o se ejecuta una acción periódica.	
Referencias	RF5, RF6
Acción del actor	Respuesta del sistema
Escenario Liberar Espacio	
	1.- Filtrar reglas de configuración en función de varios parámetros determinados. 2.- Eliminar las medias permitidas después del filtrado.

	3.- Enviar una notificación al gestor sobre los cambios realizados.
Escenario Mover Media	
	1.- Filtrar reglas de configuración en función de varios parámetros determinados. 2.- Mover las medias permitidas hacia la nueva partición especificada en la regla ejecutada. 3.- Enviar una notificación al gestor sobre los cambios realizados.
Casos de Uso Asociados	CU- 10 Procesar Reglas (incluido). CU-14 Notificar Sucesos (incluido).
Puntos de extensión	Se usa el caso de uso Procesar Reglas en: Escenario Monitorear Espacio: Acción 2.

Tabla 24 CU Ejecutar Acciones

Caso de uso	
CU-11	Gestionar Log.
Propósito	Notificar al gestor sobre cambios ejecutados en el sistema, los cuales desconoce.
Actores:	
Resumen: El caso de uso se inicia cuando el caso de uso Monitorear Acciones Periódicas lo inicia.	
Referencias	RF1.2, RF2
Acción del actor	Respuesta del sistema
Escenario Enviar Log	
	1.- El Sistema establece conexión con el gestor. 2.- Si existe conexión el sistema notifica al gestor sobre la información contenida en los logs.

	3.- El sistema elimina los logs enviados al gestor.
Casos de Uso Asociados	CU-13 Notificar Sucesos.
Puntos de extensión	Notificar Sucesos en: Escenario Enviar Log: Acción 2.

Tabla 25 CU Gestionar Log

Caso de uso		
CU-12	Notificar Sucesos	
Propósito	Notificar al gestor y recibe eventos acerca de cambios realizados en los recursos que maneja el sistema	
Actores: Gestor.		
Resumen: El caso de uso se inicia cuando el caso de uso Gestionar Log lo inicia.		
Referencias	RF1.2	
Acción del actor		Respuesta del sistema
Escenario Detectar eventos		
1.-El Gestor envía diferentes eventos ante situación determinadas a sus clientes.		2.- El Sistema detecta eventos lanzados a los cuales este suscripto y ejecuta determinadas acciones.
Escenario Notificar Eventos		
		1.- El Sistema ejecuta diversos eventos ante situaciones determinadas.
2.-El gestor recibe estas notificaciones y se actualiza.		
Casos de Uso Asociados		
Puntos de extensión		

Tabla 26 CU Notificar Sucesos

Capítulo II: Características de Sistema

Conclusiones

En el capítulo se han representado los principales conceptos obtenidos con la realización del modelo del dominio, lo que permitió conocer las peculiaridades y parámetros a tener en cuenta durante el proceso de gestión y administración de las unidades de almacenamiento para el correcto almacenamiento de los flujos capturados durante el proceso de grabación. Esto facilitó una mejor compresión de todo el proceso a automatizar. Se obtuvieron además los requisitos funcionales y no funcionales del sistema que se precisa, lográndose una propuesta certera del mismo.

Capítulo III: Análisis y diseño del sistema

3.1. Arquitectura.

Arquitectura del sistema de video vigilancia Suria.

Antes de exponer la propuesta del sistema que se presenta como solución en la presente investigación, es necesario analizar las particularidades de la arquitectura del sistema Suria, en el cual se inscribe dicha solución y dicta las especificaciones de la arquitectura a seguir.

El sistema Suria está basado en una arquitectura base en forma de Pizarra, esta se basa en contar con determinados agentes y un instrumento de control denominado pizarra en su variante de tablero de control. Esta arquitectura separa el sistema en varios agentes autónomos que suelen ser programas especializados en una tarea concreta o elemental. Todos ellos cooperan para alcanzar una meta común, estos depende de información generadas por otros agentes y a la vez producen un resultado necesarios para los otros agentes.

Cada agente se rige por interfaces estándares que permiten cambios en el ambiente externo al agente sin que éste sufra cambios en su funcionamiento interno. El elemento central de todo este funcionamiento denominado Repositorio Activo es el encargado de todo este flujo de información hacia los agentes, coordinando las actividades de estos y entregando y recibiendo información de los mismos. En el caso del sistema de video vigilancia Suria, el Gestor cumple con la función de Repositorio activo al que se pueden conectar diferentes agentes autónomos.

El sistema autónomo encargado de la administración de almacenamiento también forma parte de los agentes autónomos que colaboran en conjunto a través de la arquitectura de pizarra y es una pieza fundamental para el buen funcionamiento del sistema de grabación. Este sistema es una aplicación que monitoriza permanentemente el estado en que se encuentra el disco donde se almacenan los videos, para prevenir que el sistema falle por falta de espacio disponible para el almacenamiento de los mismos, para esto implementa reglas que definen que hacer al ocurrir un evento determinado.

El Cliente del Administrador de Almacenamiento es una interfaz gráfica que le permite a un usuario humano interactuar con el sistema, para definir las reglas así como administrar el espacio destinado para el almacenamiento de los videos.

Debido a las características del sistema y que sus funcionalidades son independientes de la manera en que se le presenta la información al usuario, se ha separado la lógica del funcionamiento del sistema de la representación visual, de este manera el sistema autónomo desempeña el rol de Modelo y las interfaces para la interacción con el sistema se definen como las Vistas. Por lo que se selecciona como patrón arquitectónico: Modelo vista Controlador (MVC).

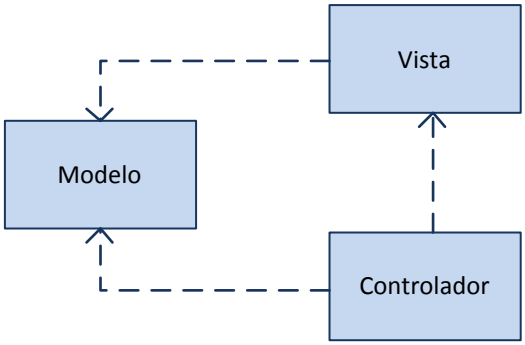

Figura 4 Patrón arquitectónico Modelo Vista Controlador.

3.2. Modelo de Análisis.

El análisis consiste en obtener una visión del sistema que se preocupa de lo que hace el mismo. El modelo de análisis es la entrada fundamental para el comienzo del diseño.

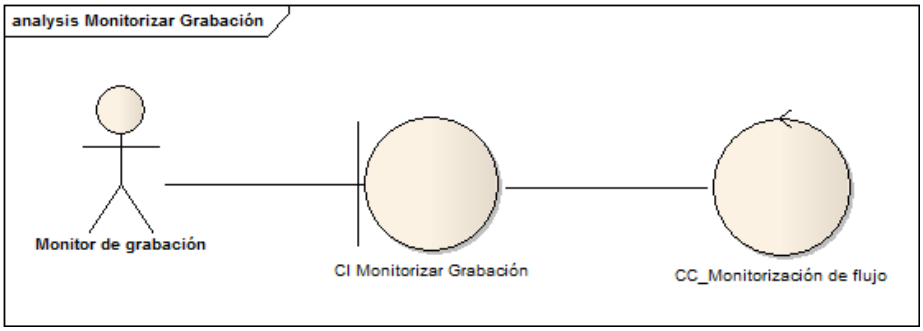

Figura 5: CU Monitorizar Grabación.

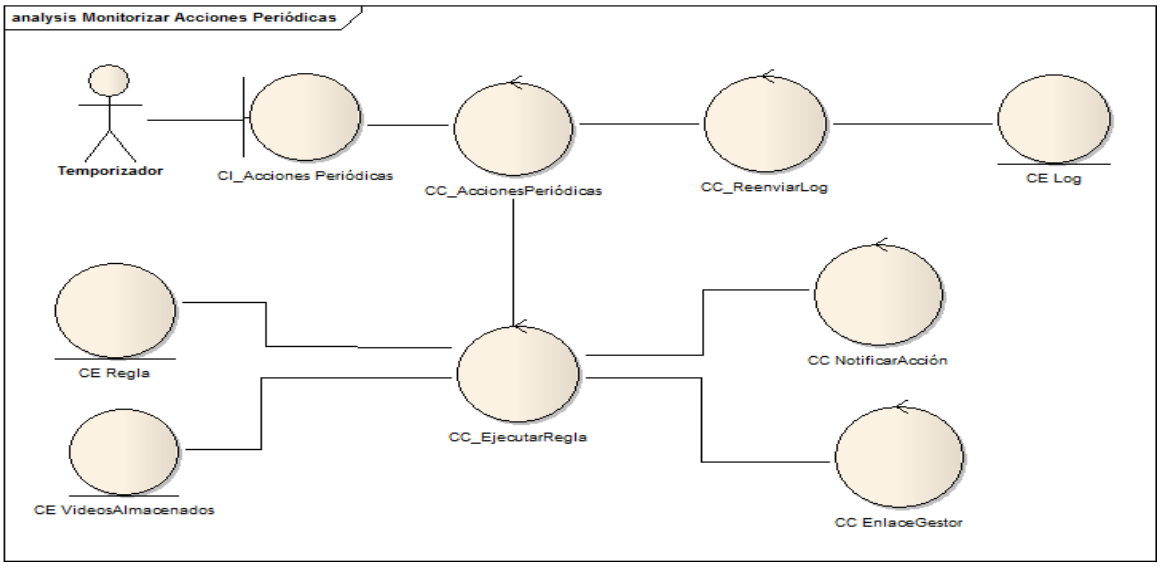

Figura 6: CU Monitorizar Acciones Periódicas.

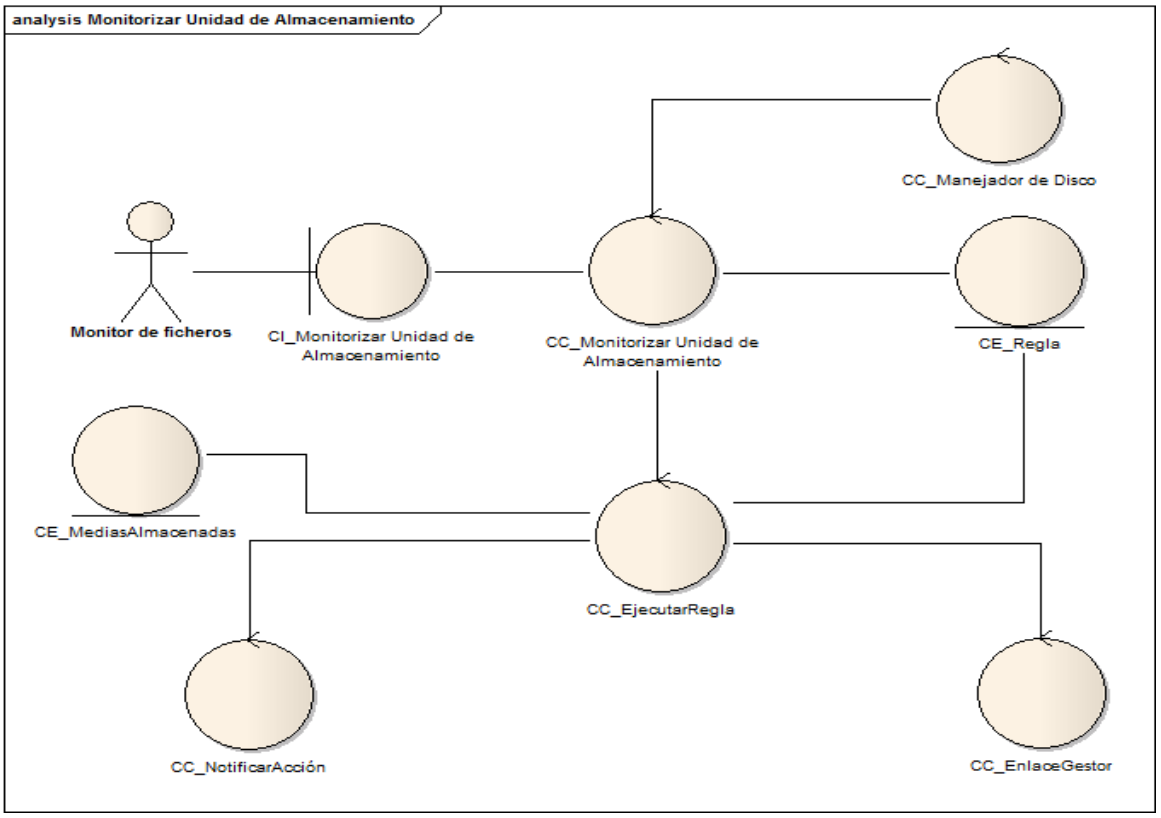

Figura 7: CU Monitorizar Unidad de Almacenamiento.

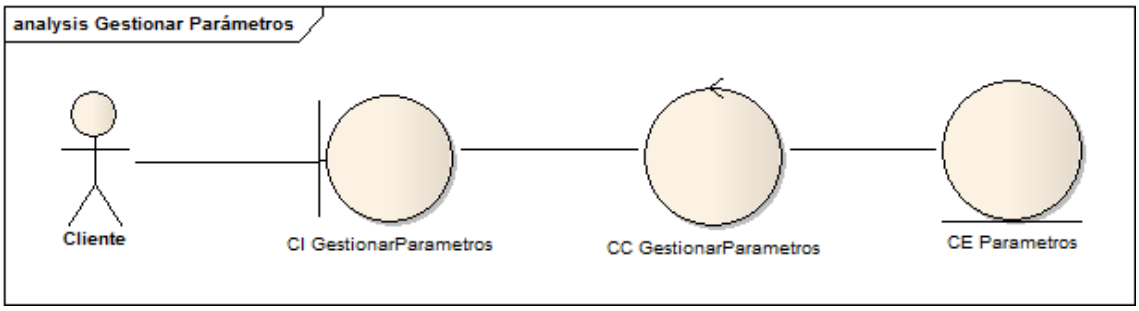

Figura 8: CU Gestionar Parámetros.

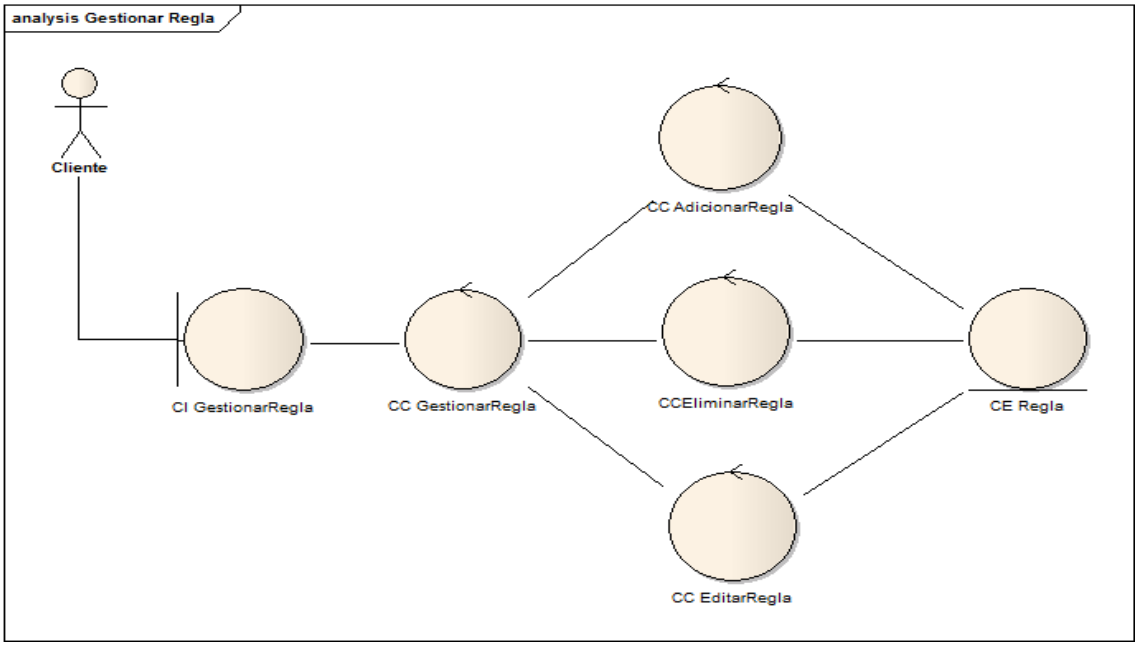

Figura 7: CU Gestionar reglas.

3.3. Modelo de Diseño.

Un modelo de diseño es un modelo de objetos físicos que describen la realización física de los casos usos centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema.

El diseño representa el centro de atención al final de la fase de elaboración y es el comienzo de las iteraciones de construcción, este representa un plano del modelo de implementación garantizando de esta manera una arquitectura sólida y estable. El modelo de diseño perdurara durante todo el ciclo de vida del software y constituye la entrada fundamental utilizada para el correcto desarrollo de la implementación.

A continuación se presentan los diagramas de clases de los casos de usos fundamentales, un diagrama de secuencia para cada uno de ellos y una descripción detallada de las principales clases involucradas en estos.

3.3.1. Monitorización de Grabación.

Diagrama de clases del CU Monitorizar Unidad de Almacenamiento, está compuesto por varias clases y en algunos casos clases pertenecientes al funcionamiento de casos de usos extendidos e incluidos necesarios para la ejecución del CU Monitorizar Unidad de Almacenamiento. Las clases fundamentales son: **DirectorySytemWatcher**, esta clase le permitirá al sistema detectar el comienzo de un proceso de grabación en determinado sector de la unidad de almacenamiento y delegar la tarea de monitorización de dicho flujo a la clase encargada de esta tarea.

La clase encargada de este proceso es: **WatcherThread**, la cual establecerá una constante monitorización del sector del disco en el cual se está almacenado el flujo de video proveniente de determinada CamaraIp, para determinar la ocurrencia de alguna situación crítica que obstaculicé el proceso de grabación, para el desarrollo de dicha tarea trabajara en conjunto con otras dos clases, la clases **ManagerRule** la cual le brindará un patrón sobre los desencadenadores(sucesos que afectan el proceso de grabación) para chequear la ocurrencia de los mismos y **ManagerDrive** clase que le brindará toda la información relacionada con el estado actual del servidor de almacenamiento.

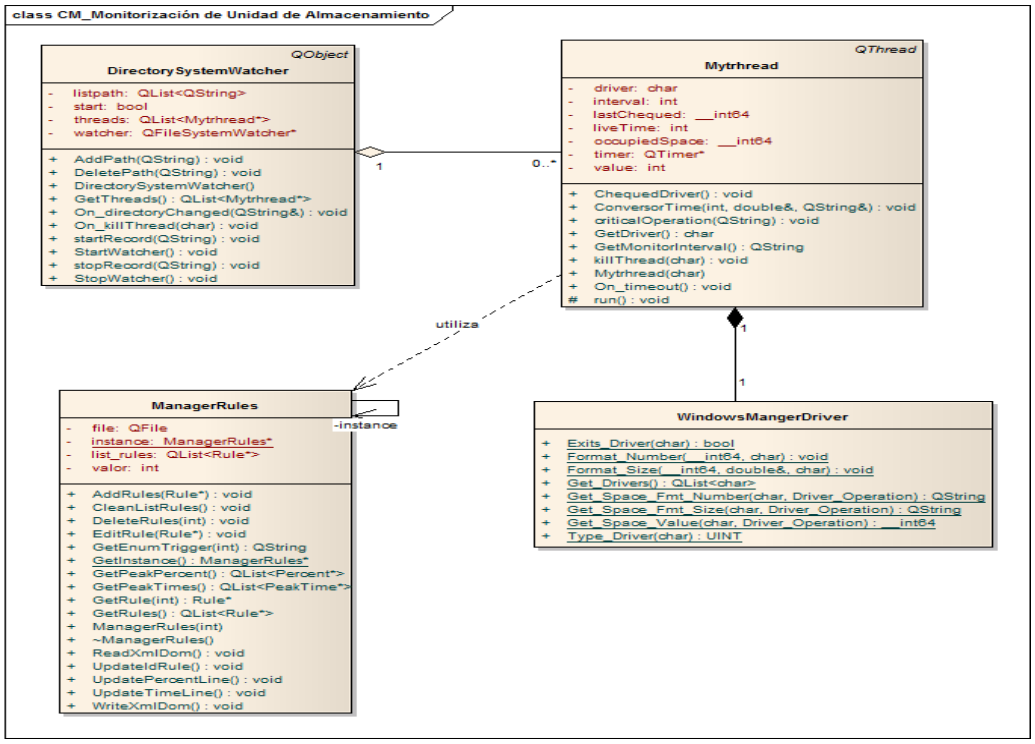

Figura 8: Diagrama de clases del CU Monitorizar Grabación.

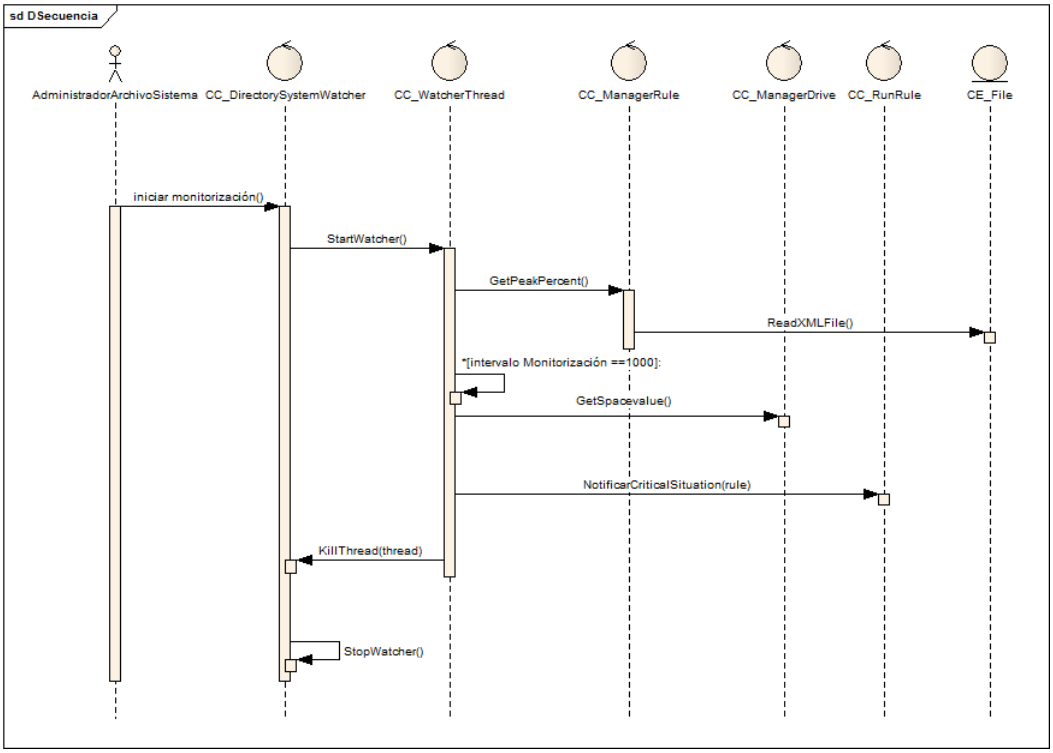

Figura 9: Diagrama de secuencia del CU Monitorizar Grabación.

3.3.2. Monitorización de Acciones Periódicas.

Diagrama de clases del CU Monitorizar Acciones Periódicas está compuesto por diversas clases estas son: **Time Line** clase encargada de definir cuando ejecutar una acción periódica y el tipo de la misma, toda esta información será facilitada por la clases **Manager Rule** la cual brindará la Programación de cada una de las reglas definidas periódicamente, con lo que se conformara una línea de tiempo que permitirá al sistema determinar cuándo ha de ejecutarse cada una de estas reglas.

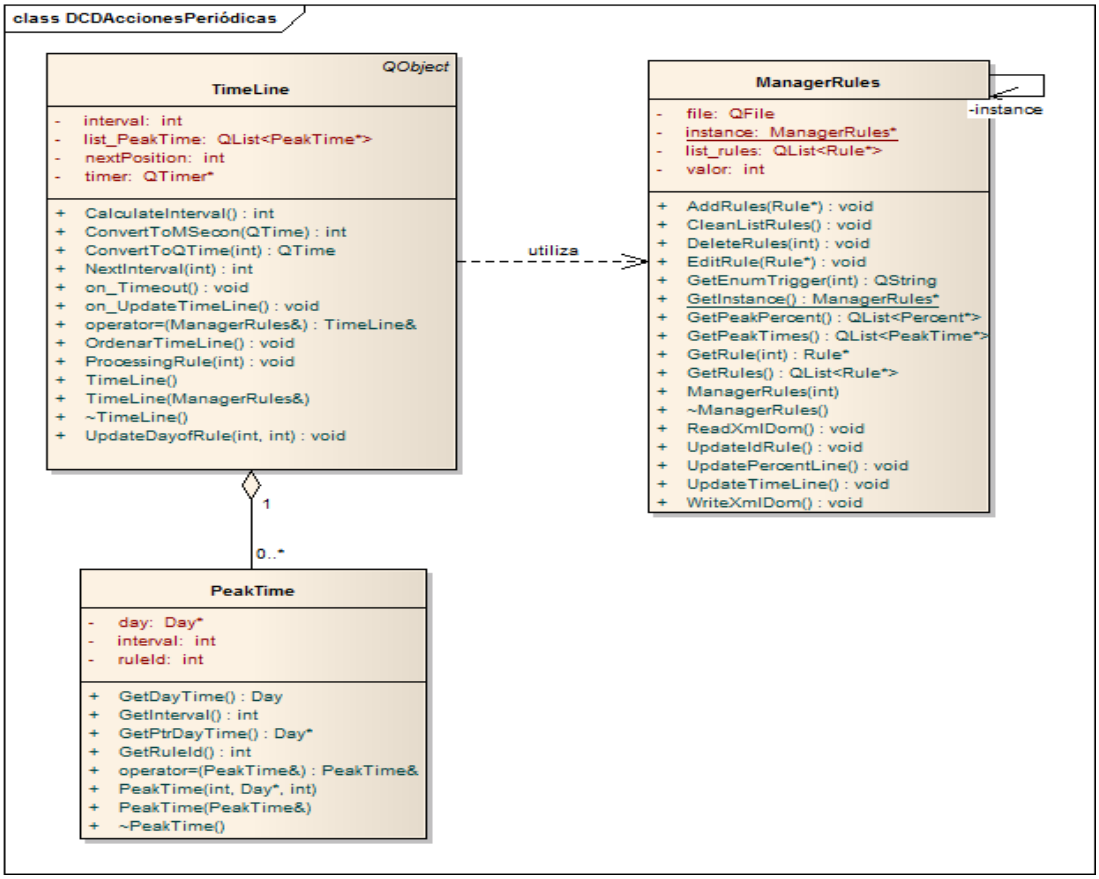

Figura 10: Diagrama de clases del CU Monitorizar Acciones Periódicas.

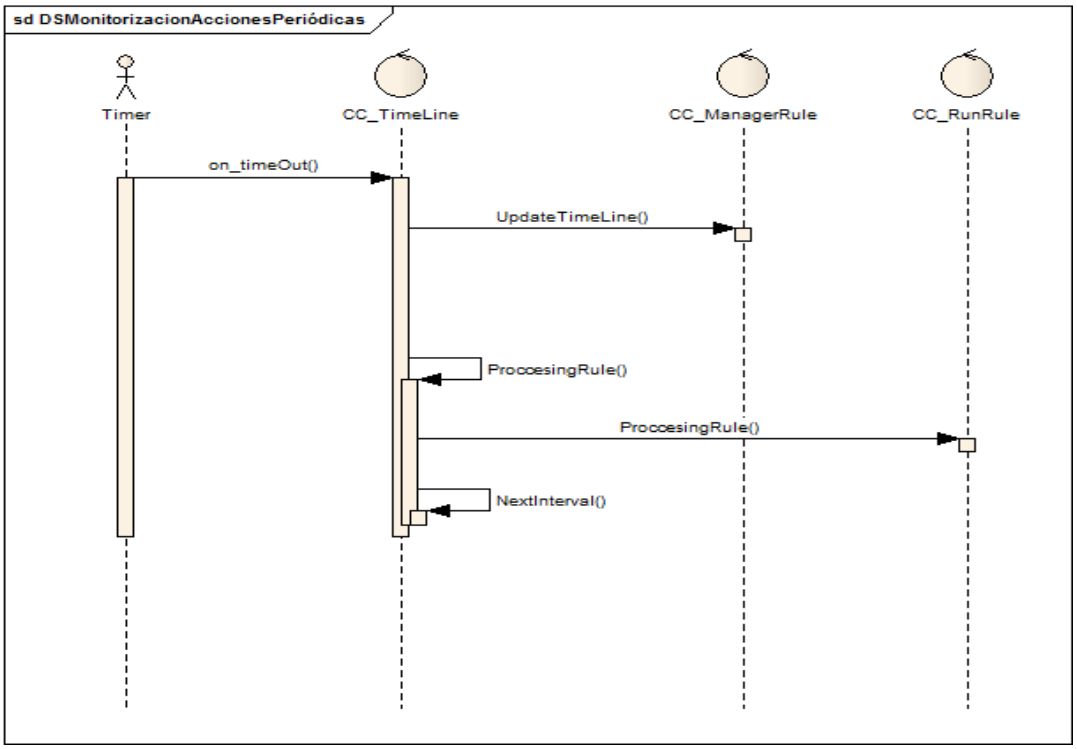

Figura 11: Diagrama de secuencia del CU Monitorizar Acciones Periódicas.

3.3.3. Ejecutar Acciones.

Diagrama de clases del CU Ejecutar Acciones: compuesto por varias clases entre las que se encuentra la clase **RunRule,** de gran peso en la realización de este flujo de trabajo. **RunRule** es la clase encargada de todo el procesamiento y ejecución de las reglas que se deben ejecutar, estas reglas son determinadas por las clases **TimeLine,** clase perteneciente al CU Monitorizar Acciones Periódicas y la clase **DirectorySystemwatcher** clase perteneciente al CU Monitorizar Grabación**.**

Para el desarrollo de este flujo de actividades la clase **RunRule** establece una dependencia con otras dos clases, **ManagerRules** y **GestorLink,** la primera le brindará toda la información necesaria sobre la regla a procesar y posteriormente ejecutar para garantizar la disponibilidad de espacio en el disco duro, la segunda clases **GestorLink** le facilitará toda la información necesaria sobre los recursos que manejas (medias) y sobre los cuales se aplicará la regla en ejecución.

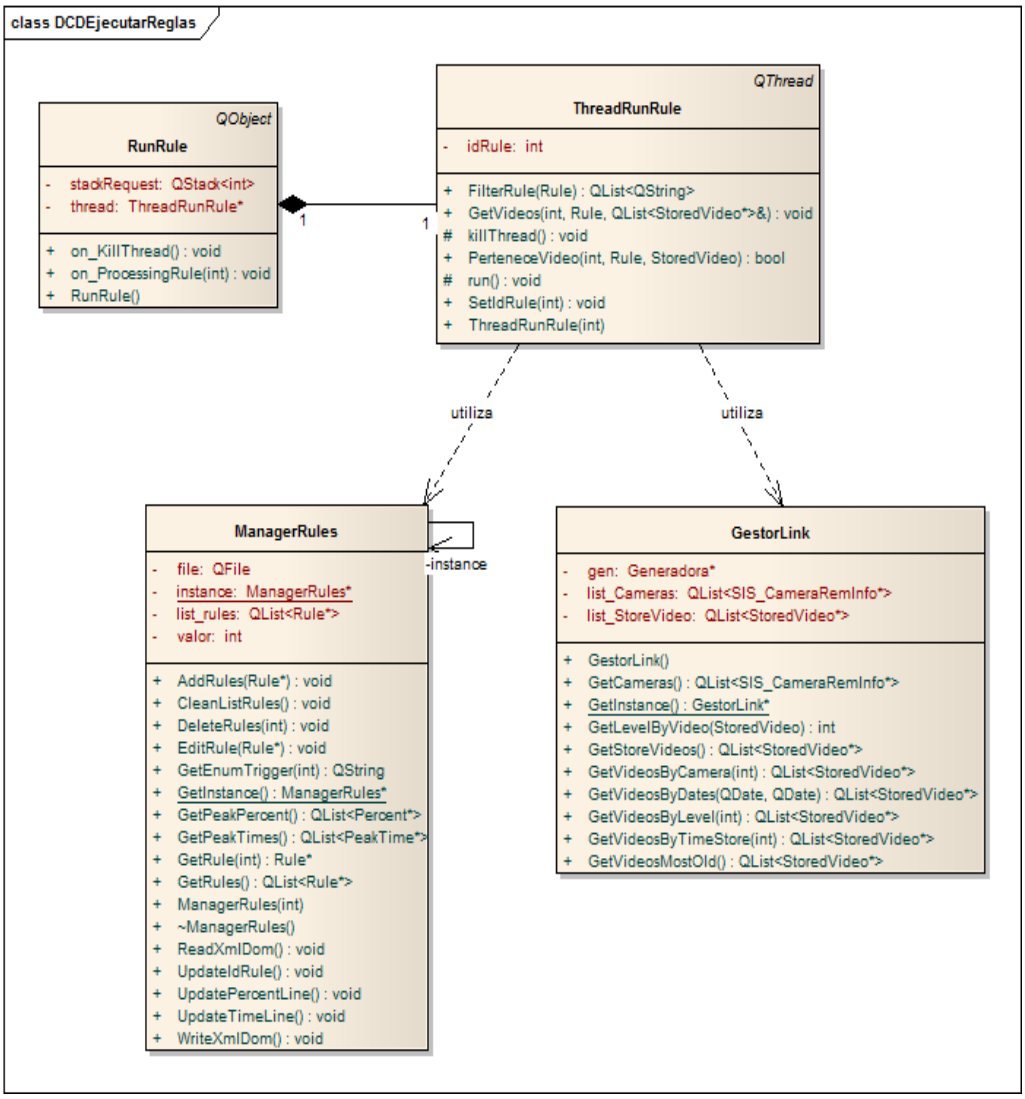

Figura 12: Diagrama de clases del CU Ejecutar Acciones.

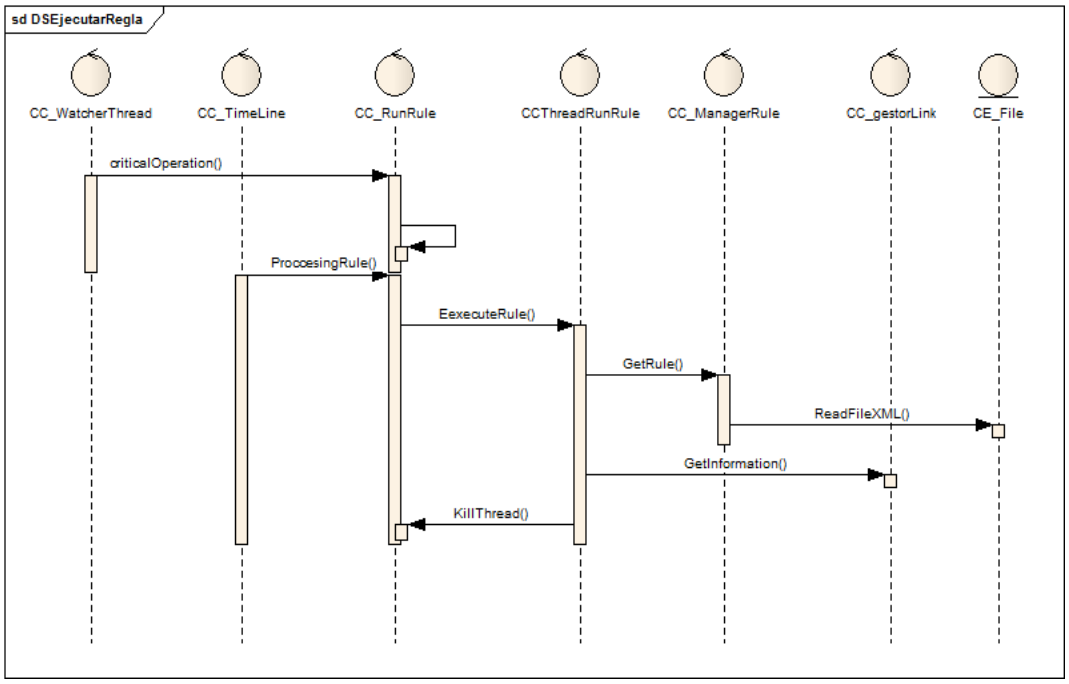

Figura 13: Diagrama de secuencia del escenario Ejecutar Reglas.

3.3.4. Descripción de las clases significativas.

Nombre: DirectorySystemWatcher	
Tipo de clase : Controladora	
Atributo	**Tipo**
start	bool
threads	QList<WatcherThread>
watcher	QFileSystemWatcher
List_paths	QList<QString>
Para cada responsabilidad:	
Nombre:	DirectorySystemWatcher()
Descripción:	Constructor que inicializa el monitorear disco con los datos requeridos.
Nombre:	Addpath()
Descripción:	Adiciona un nuevo directorio para su monitorización.
Nombre:	DeletePath ()

Descripción:	Adiciona un nuevo directorio para su monitorización.
Nombre:	StartWatcher()
Descripción:	Inicia el proceso de monitoreo. Crea una instancia de la clase WatcherThread y le asigna la tarea de monitorización.
Nombre:	StopWatcher()
Descripción:	Detiene el proceso de monitorización.
Nombre:	On_DirectoryChanged()
Descripción:	Recibe la señal de comienzo de grabación de un flujo de video e invoca la funcionalidad StartWatcher().
Nombre:	On_KillThread()
Descripción:	Recibe la notificación de la culminación de la tarea otorgada a un hilo y se encarga de liberarlo.

Tabla 27 Clase Directory SystemWatcher

Nombre: TimeLine	
Tipo de clase : Controladora	
Atributo	**Tipo**
interval	int
timer	Qtimer
nextPosition	int
List_peakTime	QList<PeakTime>
Para cada responsabilidad:	
Nombre:	TimeLine()
Descripción:	Constructor que inicializa la clase de control de acciones periódicas con los datos requeridos.
Nombre:	OrdenarTimelIne()
Descripción:	Ordena la línea de tiempo de las acciones periódicas a ejecutar para su posterior análisis.
Nombre:	CalculateInterval(StoredVideo[] video, int[] acción)

Descripción:	Calcular al iniciar el sistema cual es la acción periódica más cercana a la hora actual del sistema para ser atendida posteriormente.
Nombre:	NextInterval(int)
Descripción:	Determina el intervalo de tiempo de espera desde la ejecución de una determinada acción periódica hasta la próxima.
Nombre:	ProccesingRule(int)
Descripción:	Notificar a la clase RunRule la ejecución de una regla determinada correspondiente a la acción periódica activa.
Nombre:	UpdateDayofRule ()
Descripción:	Modifica los valores de determina acción periódica.

Tabla 28 Clase TimeLine

Nombre: ManagerRule	
Tipo de clase : Controladora	
Atributo	**Tipo**
value	int
File	QFile
List_Rule	QList<Rule>
instance	ManagerRule
Para cada responsabilidad:	
Nombre:	Set _Desencadenador()
Descripción:	Modifica el desencadenador de la regla.
Nombre:	AddRule()
Descripción:	Adicionar reglas al sistema y luego las almacena en un fichero.
Nombre:	DeleteRule()
Descripción:	Eliminar reglas del sistema y del fichero de almacenamiento de las mismas.
Nombre:	EditRule()

Descripción:	Editar reglas del sistemas y almacenar los cambios en el fichero de almacenamiento de las mismas;
Nombre:	GetInstance()
Descripción:	Brindar una instancia estática de la misma clase para la implementación del patrón singlenton.
Nombre:	GetPeakPercent()
Descripción:	Brindar una línea de porciento sobre todas las reglas definidas que tenga como desencadenador el parámetro porciento ocupado.
Nombre:	GetTimeLine()
Descripción:	Brindar una línea de tiempo sobre todas las reglas definidas que estén definidas para ejecutarse en determinadas fechas.
Nombre:	GetRule(int)
Descripción:	Obtener una determinada regla especificando su id.
Nombre:	WriteXMLDom()
Descripción:	Escribe en un fichero todos los datos obtenidos con la interacción de una regla (Adicionar, Eliminar, Editar).
Nombre:	ReadXMLDom()
Descripción:	Leer de un fichero toda la información que posee, para el funcionamiento del sistema con dichos datos.

Tabla 29 Clase ManagerRule

Nombre: RunRule	
Tipo de clase : Controladora	
Atributo	**Tipo**
stackRequest	QStack<int>
Para cada responsabilidad:	
Nombre:	RunRule()
Descripción:	Constructor que inicializa la clase encargada de la atención de peticiones para la ejecución de determinadas reglas, con los datos requeridos.

Nombre:	FilterRule(Rule)
Descripción:	Filtrar los parámetros de la regla a analizar para obtener los metadatos de las medias que se deben procesar.
Nombre:	GetVideos(int, Rule, QList<StoreVideo>)
Descripción:	Brindar los videos que cumple con determinado parámetro de la regla en ejecución a través de información brindad por la clase GestorLink.
Nombre:	PerteneceVideo(int, Rule, StoreVideo)
Descripción:	Filtrar la lista de videos obtenidas con la funcionalidad GetVideos (), en función de los demás parámetros definidos en la regla.
Nombre:	ExecuteAction()
Descripción:	Ejecutar la acción definida en la regla, sobre las direcciones de las medias obtenidas durante los procesos anteriores.

Tabla 30 Clase RunRule

Conclusiones

Durante este capítulo, se mostró la arquitectura escogida para la realización de la aplicación, se desarrolló el modelo de análisis y diseño del sistema. Elaborándose los diagramas de clases del análisis en donde se exponen los conceptos básicos del sistema, sus partes y relaciones. También los diagramas de clases del diseño y los diagramas de interacción por cada uno de los casos de usos y por último las descripciones de las clases del diseño. Todo esto contribuye a una mayor comprensión de los flujos casos de usos y los requisitos, esto permite apreciar los casos de usos como un modelo de objetos físicos que brinda una descripción detallada de la realización de estos, ajustándose a los requisitos determinados. Además a través de este capítulo se obtiene una visión más clara en términos de desarrollo del sistema a desarrollar que servirá como base o plano para el modelo de implementación garantizando de esta manera una arquitectura sólida y estable.

Capítulo IV: Implementación y Validación de la Solución Propuesta

Capítulo IV: Implementación del sistema y Validación de la Solución Propuesta.

Durante la implementación, se parte de los resultados obtenidos en el diseño y se desarrolla la aplicación en términos de componentes tales como: ficheros de código fuente, ficheros de código binario, ejecutables, entre otros. Primeramente, se detalla el modelo de datos del sistema, que es el modelo donde se ve la estructura en la cual se almacenan toda la información requerida en el sistema. Luego se muestra el modelo de implementación que está compuesto por el diagrama de despliegue y el diagrama de componentes. Estos diagramas, describen los componentes a construir, su organización y dependencias entre los nodos físicos en la que funcionará la aplicación.

4.1. Diagrama de despliegue.

El diagrama de despliegue que se muestra se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. Se utiliza además para visualizar la distribución de los componentes de software en los nodos físicos.

La aplicación desarrollada tiene tres elementos fundamentales: Servidor de Grabación, la PC Cliente y la Cámara IP, además son imprescindibles en su despliegue el Gestor y el Servidor de Base de Datos.

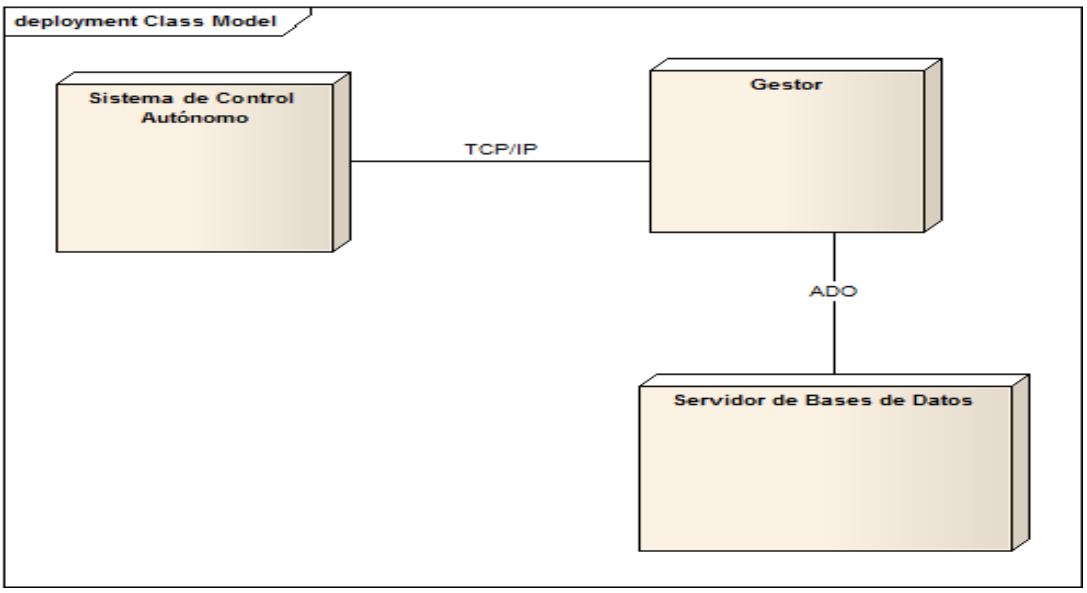

Figura 14: Diagrama de despliegue

Capítulo IV: Implementación y Validación de la Solución Propuesta

4.2. Diagrama de Componentes.

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes, se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión de software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo.

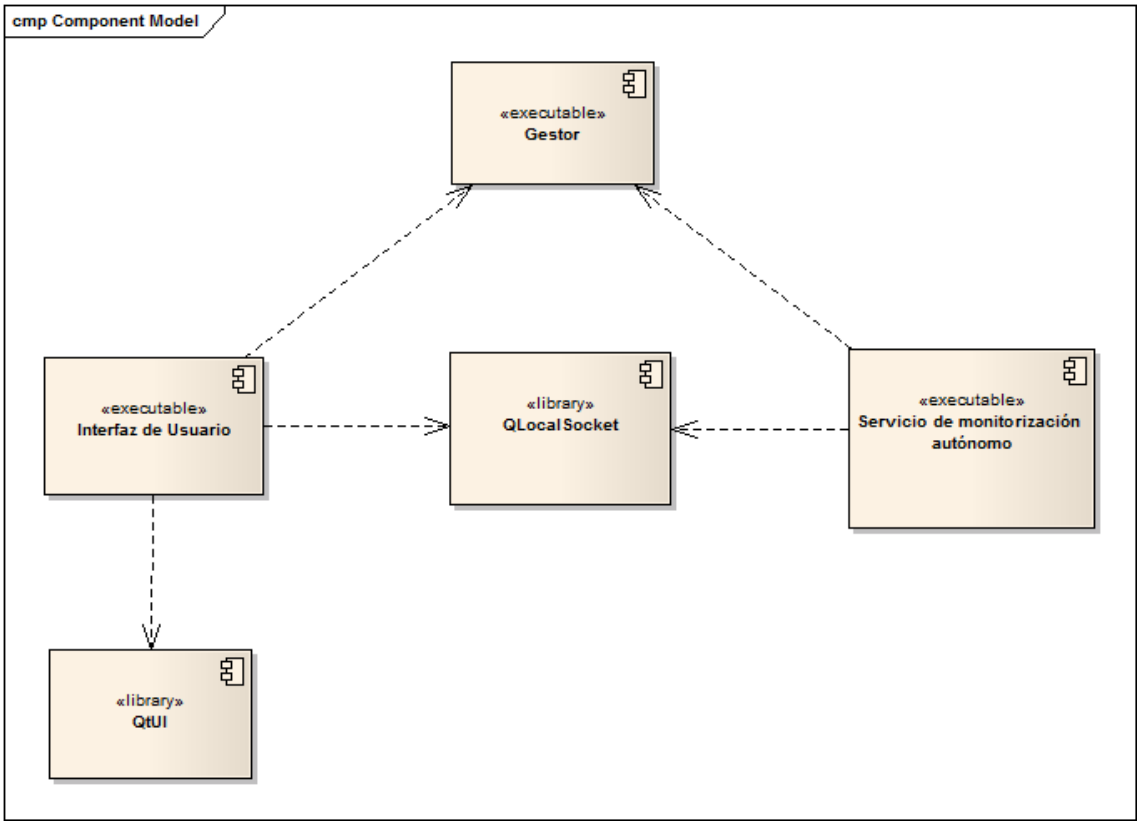

Figura 15: Diagrama de Componentes

4.3. Validación de la solución propuesta

La prueba de una aplicación informática es de gran importancia para la validación y caracterización de la eficacia del software en desarrollo, este proceso representa una revisión detallada de las especificaciones, del diseño y de la codificación de los elementos que componen la aplicación. Existen una gran variedad de pruebas para las

validaciones de los sistemas, enfocadas cada una en un aspecto específico a valorar. En el presente capítulo se describirán las pruebas funcionales que se le realizaran al Sistema para el control autónomo de espacio en disco en servidores de media.

4.4. Elementos del proceso de pruebas

La realización de las pruebas es un elemento crítico para la garantía de la calidad del software, por lo que para la creación de pruebas minuciosas y bien desarrolladas se hace necesario analizar los elementos a tener en cuenta en el desarrollo de las mismas. A continuación se mostrarán dichos elementos**.**

4.5. Niveles de pruebas.

Prueba de desarrollador: Están enfocadas al diseño e implementación de las pruebas más adecuadas a ejecutar por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para la prueba de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad.

Prueba independiente: Es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores. Estas pruebas persiguen como objetivo proporcionar una perspectiva diferente y en un ambiente más rico, que incluye validación y verificación independientes. Una vista de la prueba independiente es la prueba independiente de los stakeholder, que son pruebas basadas en las necesidades y preocupaciones de los stakeholders.

Prueba de unidad: Esta prueba se enfoca en los elementos del software más pequeños y testeables. Están pruebas son aplicadas a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. Los detalles de la prueba de unidad se describen en la disciplina de implementación.

Prueba de integración: Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso. Se prueba un paquete o un conjunto de paquetes del modelo de implementación. A través de estas pruebas se detectan errores o falta de

elementos en las definiciones de las interfaces de los paquetes. Esta prueba debe ser responsabilidad de desarrolladores y de independientes, sin solaparse las pruebas.

Prueba de Sistema: Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados.

En un ciclo iterativo estas pruebas ocurren más temprano, tan pronto como subconjuntos bien formados de comportamiento de caso de uso son implementados.

Prueba de aceptación: Prueba de aceptación del usuario es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

4.6. Tipos de pruebas.

Funcionalidad:

- Función: Prueba enfocada en la validación de funciones del sistema, servicios y casos de usos del mismo.

- Seguridad: Comprobar que el sistema y los datos solo son accedidos por los actores definidos.

- Volumen: Enfocada en verificar las capacidades del sistema para procesar grandes volúmenes de información ya sean de entrada o salida al sistema o residentes en Bases de datos.

Fiabilidad

- Integridad: Está enfocada en la validación de la robustez del sistema (resistencia a fallo).

- Stress: Está dirigida a evaluar como el sistema responde bajo condiciones anormales (extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible).

Rendimiento:

- Benchmark: Es un tipo de prueba que compara un elemento nuevo o desconocido a uno de carga de trabajo de referencia conocido.

- Carga: Esta dirigida a validar y valorar la aceptabilidad de los limites operacionales de un sistema bajo carga de trabajo variable, mientras el sistema bajo prueba permanece constante. La variación en carga es simular la carga de trabajo promedio y con picos que ocurre dentro de tolerancias operacionales normales.

- Contención: Esta dirigida a valorar la capacidad del sistema para manejar de manera correcta la demanda de múltiples actores sobre el mismo recurso (registro de recursos, memoria, etc.).

Capacidad:

- Configuración: Enfocada en comprobar el funcionamiento del sistema en diferentes configuraciones de hardware y software.

- Instalación: Enfocada en asegurar la instalación del sistema en diferentes configuraciones de hardware y software.

4.7. Pruebas Funcionales

Se denominan pruebas funcionales a las pruebas de software que tienen por objetivo validar que los sistemas desarrollados, de solución a las necesidades planteadas en los requerimientos del mismo, el desarrollo de este tipo de prueba es llevado a cabo en la mayoría de los casos por analistas de pruebas con apoyo de algunos usuarios finales, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta el paso siguiente es el pase a producción (4).

Estos tipos de pruebas también son conocidos con el nombre de pruebas de caja negra o comportamiento debido a que estas no se enfocan en cómo se desarrolla las respuestas del sistema sino que se enfoca en el análisis de los datos de entradas y

salidas, estas se enfocan en los requisitos del sistema para garantizar una entrada de datos al sistema en función de estos y analizar la veracidad de los resultados arrojado por el sistema. El objetivo perseguido con las pruebas de este tipo es encontrar la siguiente categoría de errores (12):

- Función incorrecta o ausente
- Errores de interfaz
- Errores en estructura de datos
- Errores de rendimiento
- Errores de inicialización y terminación.

4.8. Diseño de pruebas funcionales

Con el objetivo de validar el correcto funcionamiento del Sistema para el control autónomo de espacio en disco en servidores de media se realizaron pruebas de caja negra a los casos de uso Gestionar Reglas y Gestionar Unidad de Almacenamiento.

A continuación se relacionan los casos de pruebas para los casos de uso seleccionados:

4.8.1 Secciones a probar en el caso de uso Gestionar Regla

Nombre de la Sección.	Escenario de la Sección.	Descripción de la funcionalidad.
SC1: Adicionar Regla.	EC 1.1: Crear una nueva regla satisfactoriamente.	El sistema verifica la validez de los datos y que no existan campos vacíos de los requeridos por el formulario, he inserta los datos en un fichero local.
	EC1.2: Crear una nueva regla falla.	El sistema verifica la no validez de los datos requeridos en el formulario y no se adiciona la regla Muestra un mensaje informando el error que se ha producido.
SC2: Eliminar	EC 2.1: Elimina una	El Sistema corrobora la selección de

regla.	nueva regla satisfactoriamente.	la regla a eliminar y su existencia, posteriormente ejecuta dicha acción y actualiza los componentes.
	EC 2.2: Eliminar una nueva falla.	El sistema verifica la no valides de los datos requeridos par a la eliminación de una determinada regla y muestra un mensaje
SC3: Editar regla.	EC 3.1: Editar una nueva regla satisfactoriamente.	El sistema carga los datos de la regla a editar y corrobora que los nuevos datos sean correctos.
	EC 3.2: Eliminar una nueva falla.	El sistema verifica la no valides de los datos requeridos par a la edición de una determinada regla y muestra un mensaje.

Tabla 31 Pruebas CU Gestionar Reglas

4.8.2 Secciones a probar en el caso de uso Gestionar Unidad de Almacenamiento.

Nombre de la Sección.	Escenario de la Sección.	Descripción de la funcionalidad.
SC1: Eliminar medias manualmente.	EC 1.1: Eliminar medias satisfactoriamente.	El sistema valida las medias seleccionadas y ejecuta la operación determinada.
	EC 1.2: Eliminar medias falla.	El sistema valida la no valides de la selección de las medias y muestra un mensaje de información.
SC2: Mover medias manualmente.	EC 2.1: Mover medias satisfactoriamente.	El sistema valida las medias seleccionadas y ejecuta la operación determinada.
	EC 2.2: Mover medias falla.	El sistema valida la no valides de la selección de las medias y muestra un mensaje de información.

Tabla 32 Pruebas CU Gestionar Unidad de Almacenamiento

4.9. Resultado de las pruebas funcionales.

Capítulo IV: Implementación y Validación de la Solución Propuesta

Durante la fase de pruebas, específicamente las funcionales, se pudieron encontrar elementos que afectaban el buen desenvolvimiento de la aplicación. A continuación se detallan los elementos encontrados por los desarrolladores:

- El sistema permitía introducir datos incorrectos durante el proceso de adicionar y editar regla referente al campo tipo de acción de la regla.
- El Sistema no muestra ningún mensaje de información cuando el usuario trata de eliminar o editar alguna regla sin antes haberse seleccionado.
- No se muestra un mensaje de confirmación durante el proceso de eliminación de una regla.
- El sistema permite editar o eliminar una regla en ejecución
- El Sistema no muestra ningún mensaje de aviso cuando el usuario intenta mover o eliminar alguna media sin antes haberla seleccionado.
- El Sistema no muestra en caso de que diversas condiciones del sistema impidan la ejecución de la tarea ningún aviso sobre este proceso.

Conclusiones

En este capítulo se desarrolló el Diagrama de Componentes, en el que se aprecia la distribución de los elementos del software que conforman al sistema y el diagrama de Despliegue, que describe la estructura de distribución de la aplicación. El estudio y análisis de los diversos tipos de pruebas permitió seleccionar las pruebas a desarrollar para validar la solución propuesta en función de las especificaciones del producto y sus requisitos funcionales establecidos en la fase inicial del proceso de desarrollo del mismo. Los resultados arrojados por las pruebas realizadas permitieron conocer las no conformidades con que contaba el sistema, resultado esencial para la corrección de errores del sistema en función de satisfacer los requisitos del cliente para la culminación del sistema para el control autónomo de espacio en disco en servidores.

CONCLUSIONES GENERALES

Durante el desarrollo de la presente investigación, mediante el cumplimiento de las tareas y objetivos propuestos se llegó a las siguientes conclusiones:

- El análisis y caracterización del proceso de administración de los Servidores de medias para el almacenamiento de los flujos de videos, permitió obtener una mayor claridad de la situación que se desea automatizar y el desarrollo de las funcionalidades de manera eficiente, enmarcadas en satisfacer las necesidades del cliente.
- También se realizó un análisis sobre las herramientas y tecnologías a utilizar para lograr un conocimiento de las facilidades que las mismas brindan para un aprovechamiento recomendable de estas.
- El estudio y búsquedas de aplicaciones similares permitió conocer los aspectos fundamentales en los que se enfocan estas herramientas de administración de Servidores para su óptimo funcionamiento, también permitió conocer que poseen limitantes para su adquisición debido a su costo en el mercado.
- Todo este proceso facilitó el desarrollo de un producto capaz de la monitorización de Servidores de Medias de manera autónoma, logrando un control estricto sobre las medias que se manejan, de una manera agradable para el usuario y garantizar una anticipación del sistema a eventos que afecten el funcionamiento del Módulo de grabación del Sistema Suria.

Recomendaciones

RECOMENDACIONES

El sistema desarrollado no se considera un producto acabado, si no que se le pueden agregar nuevas funcionalidades que lo complementen; para ello se recomienda:

- Implementar un mecanismo para la restricción de usuarios al sistema.
- Adicionar nuevas acciones a las reglas del sistema como comprimir las medias y codificarlas.
- Crear nuevos plugins de parámetros de filtrado para lograr la adaptabilidad del sistema a nuevos entornos de administración de servidores en función de los metadatos que se manejen.
- Lograr la integración con el módulo de Envió de Notificación por diferentes medios de comunicación.

Bibliografía

Referencias Bibliográficas

1. ACENS. [Online] [Cited: 12 4, 2010.] http://www.acens.com/blog/que-es-un-data-center.html.

2. Synmatec. [Online] [Cited: 11 23, 2010.] http://www.symantec.com/es/mx/business/commandcentral-storage.

3. DiarioTI. [Online] [Cited: 11 28, 2010.] http://www.diarioti.com/gate/n.php?id=27790 and http://www.avocent.es/Products/Category/Data_Center_Management_Software.aspx.

4. **Vieto, Abel diaz Berenguer y Alberto Ramon Roman.** *Sistema de administracion y Configuracion de la solucion de captura y catalogacion de medias.* Habana : s.n., 2009.

6. **González, Juan Carlos Ferrer Rivas y Magdiel Rivero.** *Suria Recorder: Grabador de flujos de video.* Habana : s.n., 2010.

8. EVA. [Online] [Cited: 11 20, 2010.] http://eva.uci.cu/mod/resource/view.php?id=33748.

9. UsabilidadWeb. [Online] [Cited: 11 25, 2010.] http://www.usabilidadweb.com.ar/cpp.php.

10. [Online] [Cited: 11 25, 2010.] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

11. QT.nonkia. [Online] [Cited: 11 25, 2010.] http://developer.qt.nokia.com/wiki/Category:Tools::QtCreator_Spanish.

12. **García Pañeda, Xabiel, Marín Prendes, Ignacio y Melendi Palacios, David.** Universidad de Oviedo. *Área de Ingenieria Telemática, Departamento de Informática.* [En línea] [Citado el: 29 de noviembre de 2010.] http://www.it.uniovi.es/old/material/cursos/streaming_EU_122004/Intro.pdf.

13 . IBM. [En línea] [Citado el: 16 de 1 de 2011.] http://www.rational.com.

14. Sparx System. [En línea] [Citado el: 16 de 1 de 2011.] http://www.sparxsystems.com.ar/products/ea/index.html.

Bibliografía

15. Visual paradimg. [En línea] [Citado el: 16 de 1 de 2011.] http://www.visual-paradigm.com/product/vpuml.

16. Extreme Programming. [En línea] [Citado el: 20 de 1 de 2011.] http://www.extremeprogramming.org/.

17. Ayuda de RUP. s.l. : IBM Corp., 2006.

Bibliografía

1. ACENS. [En línea] [Citado el: 4 de 12 de 2010.] http://www.acens.com/blog/que-es-un-data-center.html.

2. Synmatec. [En línea] [Citado el: 23 de 11 de 2010.] http://www.symantec.com/es/mx/business/commandcentral-storage.

3. DiarioTI. [En línea] [Citado el: 28 de 11 de 2010.] http://www.diarioti.com/gate/n.php?id=27790 and http://www.avocent.es/Products/Category/Data_Center_Management_Software.aspx.

4. **Vieto, Abel diaz Berenguer y Alberto Ramon Roman.** *Sistema de administracion y Configuracion de la solucion de captura y catalogacion de medias.* Habana : s.n., 2009.

6. **González, Juan Carlos Ferrer Rivas y Magdiel Rivero.** *Suria Recorder: Grabador de flujos de video.* Habana : s.n., 2010.

8. EVA. [En línea] [Citado el: 20 de 11 de 2010.] http://eva.uci.cu/mod/resource/view.php?id=33748.

9. UsabilidadWeb. [En línea] [Citado el: 25 de 11 de 2010.] http://www.usabilidadweb.com.ar/cpp.php.

10. [En línea] [Citado el: 25 de 11 de 2010.] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

11. QT.nonkia. [En línea] [Citado el: 25 de 11 de 2010.] http://developer.qt.nokia.com/wiki/Category:Tools::QtCreator_Spanish.

12. DICCIONARIO DE LA LENGUA ESPAÑOLA - Vigésima segunda edición. [En línea] REAL ACADEMIA ESPAÑOLA, 2001. [Citado el: 18 de noviembre de 2010.] http://buscon.rae.es/draeI/.

13. **Ivan Jacobson, grady Boosch, james Rumbaun.** El proceso unificado de desarrollo de software.

14. [En línea] [Citado el: 25 de 1 de 2011.] http://www.scribd.com/doc/297224/RUP.

Bibliografía

15. **Sommerville., Ian.** Ingenieria de software septima edicion . *Ingenieria de software septima edicion .*

16. **Pressman, • Roger.** Ingeniería del Software: Un Enfoque Práctico (Sexta Edición).

17. **Larman, Craig.** Appliying uml and patterns.

18. **Frank Bushman, Regine Meunier, Hans ronerht, peter sommerland, Michael Stal.** Pattern oriented software arquitecture, a system of pattners.

19. **Erich Gamma, Richard Helm, Ralph Johnson, Jhon Visslides.** Desing pattners.

20. *Ayuda de RUP. s.l. : IBM Corp., 2006.*

21.Oré B., Ing. Alexander. CalidadySoftware.com. FUNCTIONAL TESTING - PRUEBAS FUNCIONALES. [En línea] 2009. [Citado el: 4 de Mayo de 2010.] http://www.calidadysoftware.com/testing/pruebas_funcionales.php.