Universidad de las Ciencias Informáticas

Facultad 5

Subsistema para la comunicación del SCADA Guardián del ALBA con terceros a través de la especificación OPC XML-DA.

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

Autores: Anisleidy Moya Torres
Daniel Cera Tamayo

Tutor: Ing. José Antonio Aragón Cáceres
Co-Tutor: Ing. Yanelys Del Rosario Lalcebo

La Habana, mayo 2013
"Año 55 de la Revolución"

"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado. Un esfuerzo total es una victoria completa."

Mahatma Gandhi (1869-1948

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y concedemos a la Universidad de la Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Firma del Autor

Anisleidy Moya Torres

Firma del Autor

Daniel Cera Tamayo

Firma del Tutor

Ing. José Antonio Aragón Cáceres

DATOS DE CONTACTO

Nombre y apellidos: José Antonio Aragón Cáceres.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: jaaragon@uci.cu

Ingeniero en Ciencias Informáticas y especialista graduado en la UCI en el año 2009.Tiene 6 años de experiencia en el desarrollo del software y actualmente se desempeña como arquitecto de software del SCADA Guardián del ALBA.

Nombre y apellidos: Yanelys Del Rosario Lalcebo.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: ydelrosario@uci.cu

Ingeniera en Ciencias Informáticas graduada en la UCI en el año 2012. Tiene 1 año de experiencia en el desarrollo del software y actualmente se desempeña como Jefe de Línea de Desarrollo de la línea Comunicaciones del CEDIN.

AGRADECIMIENTOS

A toda mi familia, en especial a mis padres por todo su amor, guía y apoyo a lo largo de mi vida. Por haberme permitido con su esfuerzo y sacrificio que hoy estuviera en el lugar donde estoy.

A mis abuelos, por quererme tanto y permitir que aun estén ahí para mí.

A mi novio Frank por todo su cariño y dedicación. Muchas gracias, no creo que hubiese alcanzado estos resultados sin tu ayuda.

A mis amigos de la UCI, a mi grupo y a las chicas del apartamento por hacerme pasar momentos inolvidables.

A mis tutores de la tesis, en especial a Yanelys, por todos sus consejos, ayuda y preocupación.

A mi compañero de tesis, por ser tan paciente y soportar todas mis exigencias.

A todos los que de cierta forma, contribuyeron a la realización de este trabajo.

Anita

A mi mamá, por ser mi inspiración, la persona en quien pienso antes de cumplir una tarea, por darme la seguridad de que siempre podré contar con ella.

A mi papá, por ser mi guía, mi ejemplo a seguir, por tener siempre una solución para cada problema, un consejo oportuno en momentos de duda.

A mi hermana, por indicarme el camino a seguir desde que somos pequeños.

A mi novia y su familia, que se ha convertido en la mía también.

A mis tíos José y Carlitos, que tanto me han apoyado.

A mis abuelos, por haberme cuidado desde siempre.

Y a toda mi familia y amigos, un eterno agradecimiento por haberme ayudado a llegar hasta hoy.

Daniel

DEDICATORIA

A mis padres, mi abuela Rupe, mi novio y a toda mi familia.
Anita

A mis padres, mi sobrinito, mi hermana, mi novia y a toda mi familia.
Daniel

RESUMEN

El presente trabajo muestra el desarrollo de un subsistema para la comunicación con terceros a través de la especificación OPC XML DA, que le va a permitir al SCADA Guardián del ALBA una efectiva vía de comunicación con sistemas externos que manejen sus datos a través de este formato.

Para ello se comenzó por la investigación acerca del estado del arte de las comunicaciones de sistemas SCADA con sistemas externos. Además se realizó un análisis sobre el estándar OPC, describiendo brevemente cada una de las especificaciones que brinda y enfatizando en la especificación asociada con el acceso a datos en formato XML.

Se definieron los requisitos funcionales y no funcionales para el desarrollo del subsistema, así como el modelado de la solución a partir de estos. Se implementó el diseño definido y se realizaron las pruebas a las funcionalidades más importantes, lo que permitió validar la tecnología y el modelado propuesto a través de los resultados satisfactorios de las mismas.

PALABRAS CLAVE

Comunicación con terceros, Estándar OPC XML-DA, SCADA.

ÍNDICE DE CONTENIDOS

INTRODUCCIÓN

Los sistemas SCADA, acrónimo de *Supervisory Control And Data Acquisition*, utilizan la computadora y las tecnologías de comunicación para automatizar el monitoreo y control de procesos industriales. Estos sistemas son partes integrales de la mayoría de los ambientes industriales complejos o geográficamente dispersos ya que pueden recolectar la información de una gran cantidad de fuentes rápidamente, y la presentan a un operador en una forma amigable (1).

A medida que se han desarrollado estos sistemas, se ha incrementado aún más la conectividad y la necesidad de la adquisición de datos más allá del hardware. De esta forma surge la necesidad de insertar a la recolección, fuentes de datos constituidas por software, como pueden ser otros SCADA, Sistemas de Control Distribuido, Sistemas de Bases de Datos y aplicaciones personalizadas de gestión de producción entre otros (2).

Dada la gran variedad de productores de hardware y de software a nivel mundial, la integración con sistemas de supervisión y control se ha hecho posible mediante el estándar *Open Platform Communications* (OPC), el cual proporciona un mecanismo para extraer datos de una fuente y comunicarlos con cualquier aplicación cliente.

El desarrollo de productos basados en este estándar, garantiza una alta compatibilidad entre plataformas tecnológicas en el área de la automatización industrial, permitiendo a su vez, una independencia y generalización que libera a las industrias de utilizar sólo la plataforma de programación propia del fabricante, y supone un abaratamiento en la inversión, debido a que ahora, el usuario tiene mayores alternativas para escoger los equipos con la tecnología que considere más conveniente para sus procesos.

OPC consta de varias especificaciones que determinan la forma de realizar tareas comunes en el campo del control de procesos. Una de estas especificaciones es XML-DA (*eXtensible Markup Language - Data Access*). Esta especificación se utiliza para el intercambio de datos entre clientes y servidores posibilitando aumentar las capacidades de interoperabilidad de los sistemas de automatización, supervisión y control que se ejecutan sobre sistemas distribuidos COM/DCOM (*Component Object Model/Distributed Component Object Model*), proporcionando a su vez compatibilidad multivendedor e interoperabilidad basada en Servicios Web.

El uso de esta especificación facilita el intercambio de datos de plantas a través de Internet y hacia arriba en el dominio empresarial. Es por esta razón que los sistemas de control hoy en día han añadido como una funcionalidad más, la publicación de datos a través de servidores OPC XML-DA. Esto permite que sistemas externos a la aplicación, puedan acceder a la información desde distintas plataformas.

En el Centro de Informática Industrial (CEDIN), perteneciente a la Facultad 5 de la Universidad de la Ciencias Informáticas (UCI) de conjunto con empresas venezolanas, fue desarrollado el SCADA Guardián del ALBA (GALBA) mediante un convenio entre Cuba y la República Bolivariana de Venezuela con el objetivo principal de desplegarlo en las empresas de Petróleos de Venezuela S.A (PDVSA). Este sistema está dividido de forma general en varios subsistemas que le posibilitan la escalabilidad de sus soluciones: Configuración, Aplicaciones, Adquisición, Base de Datos Histórico y Seguridad que interactúan a través de la biblioteca Comunicaciones que permite el intercambio de información entre todos los subsistemas y la interacción con sistemas externos.

Para la comunicación con sistemas externos, el GALBA utiliza los Servicios Web y un servidor que implementa la especificación OPC DA, estas dos vías se pueden agrupar bajo el nombre de “subsistema Integración con terceros”. Mediante los servicios Web los terceros pueden acceder a servicios brindados, tales como: el intercambio de variables, alarmas, eventos y autenticación para el control de acceso a la información. A través del servidor OPC DA los terceros pueden leer las variables que hayan sido configuradas y procesadas en el GALBA pero solo aquellos que se encuentren sobre la plataforma Windows ya que la capa de comunicación OPC es dependiente de la tecnología COM/DCOM de Microsoft.

A raíz de la liberación de la especificación OPC XML-DA por parte de la Fundación OPC, algunos sistemas de control añadieron como una funcionalidad más, la publicación de sus datos a través de este formato ya que brinda la posibilidad de combinar las potencialidades de utilizar Servicios Web y OPC DA en una misma solución, independiente de un sistema operativo, tipo de computador e incluso un lenguaje de programación específico.

Con el objetivo de lograr una mayor integración con terceros en el SCADA GALBA y poder llevar la información a niveles superiores de gestión empresarial, se comenzó en el año 2011, el desarrollo de un servidor que implementara la especificación OPC XML-DA. Luego de pasar por un período de pruebas, se determinó que la solución obtenida no cumplía con los requisitos necesarios para un funcionamiento exitoso por lo que no pudo ser integrada al SCADA GALBA.

Actualmente muchos de los sistemas y dispositivos de control con los que el SCADA GALBA

necesita comunicarse publican sus datos a través de OPC XML-DA. Esto constituye una limitante puesto que no se cuenta con un servidor que permita obtener información publicada por estos sistemas y exportar datos en este formato que sean accesibles desde cualquier lugar con acceso a Internet, ni con un cliente que permita acceder a dicha información.

Para dar respuesta a la **situación problemática** expuesta anteriormente se considera el siguiente **problema a resolver:** ¿Cómo permitir la publicación de datos a través de la especificación OPC XML-DA en el SCADA Guardián del ALBA y acceder a la información de otras aplicaciones externas que implementen dicha especificación?

La investigación se enmarca en el **objeto de estudio** asociado a la comunicación de sistemas industriales con sistemas externos, que tiene como **objetivo general:** Desarrollar un subsistema para el módulo de Integración con terceros del SCADA Guardián del ALBA que permita la comunicación a través de la especificación OPC XML-DA. Centrándose en el **campo de acción:** el subsistema de Integración con terceros del SCADA Guardián del ALBA.

Planteándose la siguiente **idea a defender**: con la incorporación al SCADA Guardián del ALBA de un subsistema que permita el intercambio de datos publicados mediante el especificación OPC XML-DA se garantizará la comunicación con gran variedad de dispositivos y sistemas que existen actualmente en la industria petrolera venezolana.

Las tareas de investigación que se definen son:

- Análisis de la especificación OPC XML-DA del estándar OPC para identificar las características y los requerimientos a tener en cuenta en el desarrollo del sistema.
- Análisis de la documentación referente a OPC XML-DA para el establecimiento de los conceptos básicos asociados a su implementación.
- Selección de las herramientas que no limiten la capacidad multiplataforma de la solución final.
- Implementación de funcionalidades del servidor OPC XML-DA.
- Diseño de una aplicación cliente OPC XML-DA según las características y requisitos identificados.
- Implementación de una aplicación cliente OPC XML-DA según las estructuras de programación seleccionadas.
- Pruebas al subsistema desarrollado para validar la solución.

Para realizar las tareas de la investigación se tienen en cuenta varios **métodos de investigación**:

Métodos teóricos:

- **Analítico-sintético:** Utilizado para analizar las teorías y documentos relacionados con el estándar OPC XML-DA permitiendo la extracción de los rasgos y elementos más importantes de los servidores y clientes OPC XML-DA dentro de los sistemas SCADA.
- **Análisis Histórico-lógico:** Utilizado para constatar los antecedentes y la evolución de los servidores y clientes OPC XML-DA en sistemas SCADA y conocer las tendencias actuales sobre el estándar OPC. Además para profundizar en los conceptos, términos y vocabulario propio del objeto de estudio y el campo de acción.

Método empírico:

- **Observación:** Utilizado para identificar y analizar las características específicas que identifica a los servidores y clientes OPC XML-DA.

Este documento está estructurado de la siguiente manera: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, referencias bibliográficas, bibliografía consultada, anexos y glosario de términos.

En el Capítulo 1: "Fundamentación Teórica" se brinda un estudio sobre la comunicación con terceros en sistemas SCADA, el estándar OPC y la especificación OPC XML-DA así como la determinación de los lenguajes, metodología y herramientas a utilizar en el desarrollo del subsistema de comunicación.

En el Capítulo 2: "Análisis y diseño del sistema" se definen los actores y requisitos con los que debe cumplir el sistema y se modelan los diagramas de clases de diseño, los diagramas de secuencia y el de casos de uso. Conjuntamente se exponen y especifican los patrones de diseño utilizado y la arquitectura del subsistema.

En el Capítulo 3: " Implementación y pruebas" se muestran las vistas de implementación y despliegue. Se diseñan los casos de prueba y se ejecutan las pruebas de unidad para detectar los errores y validar la solución propuesta.

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En el presente capítulo se muestran los conceptos fundamentales relacionados con los sistemas SCADA: su descripción y funcionalidades así como un breve estudio de cómo se establece la comunicación en ellos. Se aborda el tema de la comunicación de los SCADA con sistemas externos, teniendo en cuenta el estudio del estándar OPC para establecer dicha comunicación y dentro de él, la especificación OPC XML-DA para el desarrollo del subsistema en cuestión. Además se describen la metodología, lenguajes y herramientas necesarias para desarrollar el subsistema de comunicación a través de OPC XML-DA.

1.2 Sistemas SCADA.

Un sistema SCADA es una aplicación o conjunto de aplicaciones de software, con la finalidad de controlar y supervisar procesos a distancia. Está especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, entre otros) y controlando el proceso de forma automática desde una computadora. Además, envía la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa permitiendo de esta forma la participación de otras áreas como por ejemplo: control de calidad, supervisión, mantenimiento, entre otros (2).

Actualmente el campo de aplicación para un SCADA es muy amplio, considerando el incremento a nivel mundial de industrias con procesos automatizados. Entre algunas de sus aplicaciones se pueden mencionar: el telecontrol de estaciones remotas, el monitoreo y control de señales analógicas y digitales, el control de oleoductos, yacimientos de gas y petróleo, redes de distribución de gas natural, sistemas de transmisión de energía eléctrica, sistemas de distribución de agua, entre otros.

1.2.1 Funciones principales de los sistemas SCADA.

A continuación se muestran algunas de las funcionalidades con las que debe contar un sistema SCADA (2).

- **Supervisión remota de instalaciones y equipos:** Permite al operador conocer el estado de desempeño de las instalaciones y los equipos alojados en la planta, lo que

permite dirigir las tareas de mantenimiento y estadística de fallas.

- **Control remoto de instalaciones y equipos:** Mediante el sistema se puede activar o desactivar los equipos remotamente (por ejemplo abrir válvulas, activar interruptores, prender motores, entre otros), de manera automática y también manual. Además es posible ajustar parámetros, valores de referencia, algoritmos de control, entre otros.
- **Procesamiento de datos:** El conjunto de datos adquiridos conforman la información que alimenta el sistema, esta información es procesada, analizada, y comparada con datos anteriores, y con datos de otros puntos de referencia, dando como resultado una información confiable y veraz.
- **Visualización de gráfica dinámica:** El sistema es capaz de brindar imágenes en movimiento que representen el comportamiento del proceso, dándole al operador la impresión de estar presente dentro de una planta real. Estos gráficos también pueden corresponder a curvas de las señales analizadas en el tiempo.
- **Generación de reportes:** El sistema permite generar informes con datos estadísticos del proceso en un tiempo determinado por el operador.
- **Representación de señales de alarma:** A través de las señales de alarma se logra alertar al operador de que está frente a una falla o de la presencia de una condición perjudicial o fuera de lo aceptable. Estas señales pueden ser tanto visuales como sonoras.
- **Almacenamiento de información histórica:** Se cuenta con la opción de almacenar los datos adquiridos, esta información puede analizarse posteriormente, el tiempo de almacenamiento dependerá del operador o de los requisitos del sistema.
- **Programación de eventos:** Esta referido a la posibilidad de programar subprogramas que brinden automáticamente reportes, estadísticas, gráfica de curvas, activación de tareas automáticas, entre otros.

1.2.2 Comunicaciones.

Las comunicaciones desempeñan un papel significativo en los sistemas SCADA. Los mismos necesitan comunicarse de varias maneras, con los dispositivos de campo, comunicación entre sus subsistemas o módulos y con aplicaciones externas incluyendo otros SCADA.

Para realizar el intercambio de información entre los dispositivos de campo y las estaciones del SCADA es necesario un medio de comunicación. Cada fabricante de equipos para sistemas SCADA emplea diferentes protocolos de comunicación, que son un conjunto de reglas y procedimientos que permite a las unidades remotas y central, el intercambio de información.

Debido a esto no existe un estándar para la estructura de los mensajes, sin embargo existen estándares internacionales que regulan el diseño de las interfaces de comunicación entre los equipos del sistema SCADA y equipos de transmisión de datos.

Los sistemas SCADA hacen uso de los protocolos de las redes industriales. Normalmente se usan manejadores (*driver*) que hablan el idioma de los autómatas colocados en el campo para transmitir la información del campo hacia el SCADA o se utiliza el propio *driver* OPC (3).

Un sistema SCADA cubre por lo general, áreas geográficas grandes, y necesita diferentes medios de comunicación. La calidad de la tecnología de un sistema SCADA es un aspecto importante que define la capacidad de garantizar la confiabilidad en la transferencia de los datos al usar estos medios.

En el caso de las comunicaciones internas del SCADA por lo general se utiliza un *Middleware* o software de comunicación entre aplicaciones, definido como la capa de software que se encuentra por encima de los niveles físicos y de transporte y por debajo de las capas de gestión y aplicaciones de usuario y su función principal radica en establecer un método de comunicación entre las aplicaciones que se encuentran ejecutándose en cualquier punto del sistema (3). En la siguiente figura que representa la pirámide de automatización de un sistema SCADA se puede ver con claridad la ubicación de esta capa.

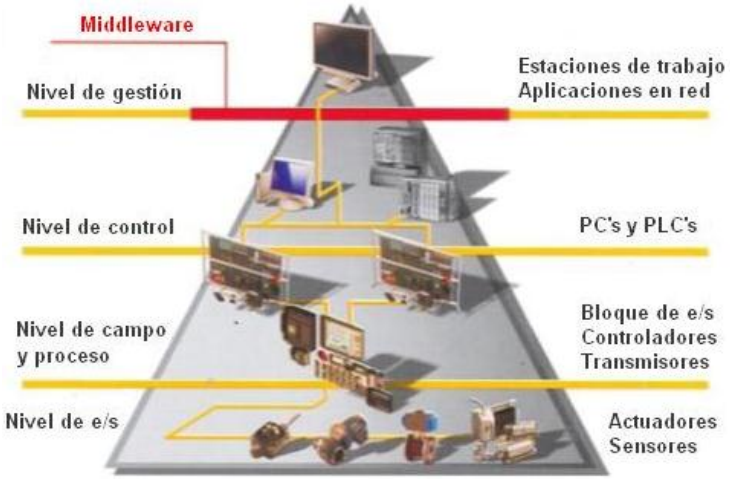

Figura 1. Ubicación del *Middleware* en un Sistema SCADA (4).

En sus inicios los sistemas SCADA utilizaron enlaces de comunicación lentos, los cuales han ido evolucionando de manera vertiginosa, pasando de las comunicaciones a través de la radio hasta las comunicaciones satelitales. Actualmente con la evolución de la programación desde el enfoque estructurado a enfoques orientados a objetos, arquitecturas orientadas a servicios, unido a las tecnologías desarrolladas como Microsoft DCOM, .NET, CORBA, entre otras, han

permitido que los sistemas SCADA se expandan más allá de las redes LAN y lleguen a ofrecer sus servicios por interfaces Web que pueden ser accedidas por los clientes en Internet.

1.3 Comunicación con terceros en sistemas SCADA.

La mayoría de los sistemas SCADA que son instalados hoy, ya no son vistos por la dirección de las empresas, simplemente como herramientas operacionales, sino como un recurso importante de información, debido a que proporcionan datos a sistemas y usuarios fuera del ambiente del centro de control, que dependen de la información oportuna en la cual basan sus decisiones económicas cotidianas. Estos sistemas que son externos al centro de control, se le llaman terceros, y pueden ser otros sistemas SCADA, sistemas de Planificación de Recursos de la Empresa, Sistemas de Ejecución de Manufactura o Sistemas de Control Distribuido, entre otros.
Dentro de las funciones más específicas que debe tener un sistema SCADA para su óptima explotación e integración en un entorno más amplio, se encuentra el uso de los datos adquiridos, para el proceso de gestión, tanto de la calidad como de la producción y administración y para el control estadístico y financiero. Esta información se puede brindar a un sistema externo o publicarse en una red corporativa a través de servicios (5).
Los sistemas externos usualmente utilizan los datos obtenidos de los sistemas SCADA para el registro y gestión de los datos. En el caso más general, las necesidades a cubrir se formulan como (4):

- Conocer la situación de la planta y de la producción en tiempo real.
- Adquisición de datos para análisis históricos, control de calidad, cálculo de costes.
- Generación de informes.

Esto puede llegar a incluir funciones de identificación de operarios, mensajería, gestión de almacén y gestión de mantenimiento entre otros.

1.3.1 Servicios brindados a terceros en sistemas SCADA.

Los servicios más comunes que se proveen a terceros por un SCADA son los siguientes (5):

- **Intercambio de información relacionada con variables del sistema (puntos):** Este servicio se encarga de garantizar a sistemas externos el acceso a las variables que se manipulan en un sistema SCADA. Estas variables pueden ser de tipos simples o de tipos complejos. La información de dichas variables permite conocer el estado del sistema y su historia, de ahí la importancia de que sistemas de gestión o sistemas gerenciales puedan acceder a dicha información.

- **Intercambio de información relacionada con alarmas y eventos:** Este servicio se encarga de garantizar el flujo de eventos y alarmas a través de todas las capas de la pirámide de control y supervisión, permitiendo tomar decisiones correctivas del estado del sistema. El caso de las alarmas requiere una atención diferente ya que son eventos especiales que requieren de una atención adecuada puesto que se relaciona con manifestaciones anómalas del proceso y de comportamiento del sistema como un todo.
- **Interacción a través de comandos:** Este servicio se encarga de definir la forma en que los clientes externos solicitan servicios de ejecución de comandos de manera segura a los servidores disponibles. Las respuestas de estos comandos pueden ser sincrónicas o asincrónicas en dependencia de la naturaleza de las aplicaciones. Usualmente se utiliza este servicio para la escritura de variables ya que esta interacción debe contar con una mayor seguridad.
- **Intercambio de Información asociada a la Visualización Hombre-Máquina:** Este servicio brinda a supervisores u otros entes del negocio, un sistema de visualización que permite presentar información gerencial y de control. En la actualidad los sistemas SCADA ofrecen la posibilidad de obtener esos despliegues gráficos desde nodos externos a la red local del sistema. La solicitud de reportes al sistema puede ser enviado para su posterior visualización en los clientes. Normalmente devueltos en formato XML o HTML.
- **Servicios de Seguridad:** Este servicio es el encargado de evitar que mediante ataques malignos los datos que viajan por la red sufran alguna alteración. La modificación de variables protegidas e invocación de comandos que modifiquen estados del sistema causando la inestabilidad, son algunos de los ataques que pueden resultar perjudiciales. Este servicio debe proveer la autenticación de cliente, el control de acceso de los recursos y la codificación de la información para su seguridad.

1.4 OPC.

OPC, antes conocido como *OLE* para el Control de Procesos, ahora sinónimo de conectividad abierta dado que en noviembre de 2011, la Fundación *OPC* rebautizó oficialmente el acrónimo con el significado" *Open Platform Communications*". OPC consiste en un conjunto de especificaciones basadas en los estándares de Microsoft (*COM, DCOM, OLE Automation* y *ActiveX*) con el cual es posible comunicar dispositivos industriales con sistemas de información o aplicativos de escritorio. OPC permite desarrollar de una manera muy práctica y eficiente

aplicaciones que pretendan comunicarse con equipos controlados por PLC (*Programmable Logic Controller)* (6).

En otras palabras, es un mecanismo estándar de comunicación, que interconecta en forma libre, numerosas fuentes de datos, y es utilizado generalmente en el campo del control y supervisión de procesos.

1.4.1 Surgimiento de OPC.

En los sistemas de automatización se encuentran múltiples elementos de control y monitorización, cada uno con su protocolo de comunicaciones específico (Modbus, BISAP-Serial, Ethernet, entre otros) y con características propias. Para el acceso a cada elemento de control se necesita utilizar un programa distinto por cada protocolo, acompañado de otro software para la visualización de los datos recolectados del dispositivo, provisto por el propio fabricante (7).

Esto trae consigo que los desarrolladores de sistemas de automatización, se vean obligados a utilizar el software y hardware de un determinado proveedor, en la mayoría de los casos. Las aplicaciones de software desarrolladas por un fabricante no son compatibles con el hardware de otro, aunque tengan el mismo protocolo, ya que no implementan interfaces comunes para poder lograr la compatibilidad entre ambos.

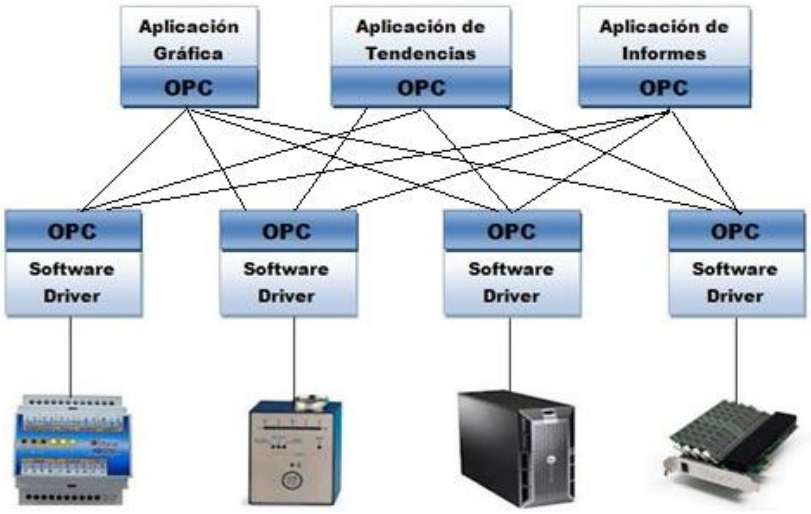

Figura 2. Red donde no se aplica el estándar OPC (7).

OPC surgió con la idea de suprimir el problema anteriormente descrito creando un estándar orientado al modo de intercambio de datos, independientemente de la tecnología utilizada para hacerlo. Cualquier que sea la fuente de datos (un PLC, un regulador de temperatura, entre otros) el formato de presentación y acceso a los datos será fijo. De esta manera, permitirá

intercambiar datos con cualquier equipo que cumpla el estándar OPC y además una reducción de costes considerable, pues cada manejador se deberá escribir una sola vez (7).

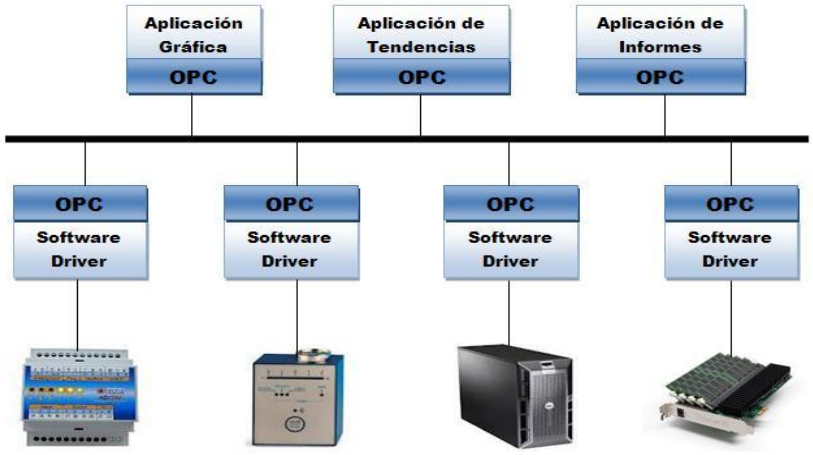

Figura 3. Solución OPC (7).

1.4.2 Arquitectura de OPC.

OPC emplea la arquitectura cliente/servidor, lo que hace posible la comunicación entre elementos que cumplan con el estándar. Permite además el acceso a los datos de forma local y remota en tiempo real.

Aunque OPC está diseñado principalmente para acceder a datos desde un servidor en red, las interfaces OPC pueden utilizarse en muchos lugares dentro de una aplicación. En el nivel más bajo se puede obtener los datos en bruto de los dispositivos físicos, hacia sistemas SCADA o Sistemas de Control Distribuido (DCS), o desde estos sistemas, hacia la aplicación (7).

Figura 4. Relación Cliente/Servidor de OPC (7).

Cliente OPC: Aplicación que solo utiliza los datos brindados por el servidor OPC. Cualquier cliente OPC puede conectarse con cualquier servidor OPC sin importar el tipo de dispositivo que recoge los datos.

Servidor OPC: Es una aplicación que recolecta los datos de los dispositivos que se encuentran configurados en campo, permite el acceso libre a los datos recolectados desde otras aplicaciones que los soliciten (Clientes OPC) (7).

1.4.3 Ventajas de OPC.

- Los fabricantes de hardware, tienen que hacer solamente un conjunto de componentes de software para que los clientes los utilicen en sus aplicaciones.
- Los desarrolladores de software no tienen que reescribir *drivers* debido a cambios en características o adiciones en un hardware.
- Los clientes tendrán más opciones con las cuales puedan desarrollar diversos sistemas de aplicación industrial.
- Permite una realización flexible y eficiente de la lectura/escritura de datos entre una estación central y un dispositivo de control de proceso (6).
- Provee mecanismos para que sus clientes sean notificados de la ocurrencia de acontecimientos y de condiciones de alarmas especificadas.
- Permite la lectura, procesamiento y corrección de datos históricos (6).
- Con OPC, la integración de un sistema industrial en un ambiente de computación heterogéneo resulta simple, puesto que un cliente OPC se puede conectar a servidores OPC proporcionados por más de un proveedor (8).

1.4.4 Especificaciones OPC.

- **Estándar OPC para Acceso a Datos (OPC DA):** Es la primera de un conjunto de especificaciones conocidas como las Especificaciones OPC. Se suele emplear para transferir datos en tiempo real desde PLC y otros dispositivos de control, hacia HMI (*Human Interface Machine*) y otros clientes (8).
- **Estándar OPC para Alarmas y Eventos (OPC AE):** Proporciona notificaciones de alarma o de evento bajo demanda. Se incluyen procesamiento de alarmas, acciones de operador, mensajes de información y mensajes de seguimiento y auditoría (8).
- **Estándar OPC para Acceso a Datos Históricos (OPC HDA):** Proporciona acceso a datos previamente almacenados. Desde un simple registro de datos en serie hasta complejos sistemas SCADA, los históricos pueden ser accedidos de manera uniforme (8).

- **Estándar OPC para Seguridad:** Especifica cómo se controla el acceso de los clientes a los servidores con el fin de proteger información sensible y evitar modificaciones no autorizadas de los parámetros de los procesos (8)**.**
- **Estándar OPC para Intercambio de Datos (OPC DX):** Esta especificación se aleja de un entorno cliente/servidor mediante comunicación por medio de redes de buses de campo sobre Ethernet. Con esto se soporta la interoperabilidad entre múltiples vendedores. También soporta servicios de configuración remota, diagnóstico y monitorización/administración (8)**.**
- **Estándar OPC, Arquitectura Unificada (OPC UA):** Integra la funcionalidad de las anteriores especificaciones (OPC DA, OPC-HDA, OPC A&E, OPC-DX). Abandona COM/DCOM en favor de dos transportes: SOAP/HTTP(S) (*Simple Object Application Protocol* / *Hypertext Transfer Protocol*) y un mensaje binario codificado en la parte superior de TCP (6).

1.5 OPC XML-DA.

OPC XML-Data Access (OPC XML-DA) es la adopción del conjunto de tecnologías XML por parte de la Fundación OPC para facilitar el intercambio de datos de planta a través de Internet, y hacia arriba en el dominio empresarial. Al igual que OPC DA, su objetivo es brindar información acerca de las aplicaciones y sistemas a través de la red, pero en vez de utilizar tecnología COM/DCOM utiliza mensajes SOAP sobre *(*HTTP), con documentos en XML. De esta forma este estándar simplifica el intercambio de datos existente entre los diferentes niveles de las aplicaciones (van desde los dispositivos de bajo nivel hasta los sistemas empresariales) y en una amplia gama de plataformas (9).

Utilizando OPC XML-DA se pueden integrar todos los servicios necesarios para el control y supervisión de dispositivos de campo en un proceso industrial, y usar los mismos desde cualquier sitio remoto con acceso a Internet, o dentro de una red de área local (LAN).

Debido a la interfaz SOAP que ofrece para los objetos OPC DA 2.0/3.0 se está convirtiendo en el método estándar para el intercambio de datos entre las aplicaciones de empresa, permitiendo a las aplicaciones clientes ser escritas en Java, Perl, Python y otros idiomas que soporta SOAP (7).

La especificación OPC-XML-DA proporciona soporte para datos OPC Data Access, soporte para HTTP y SOAP, soporte para servicios basados en subscripción y para un enfoque de seguridad.

1.5.1 Arquitectura de subscripción de OPC XML-DA.

OPC XML-DA se basa en la arquitectura cliente/servidor, brindando solución a las limitaciones del protocolo HTTP a través de la técnica “suscripción de encuestas”, donde el cliente inicia la suscripción y se realizan periódicamente solicitudes de actualización, la respuesta de los mensajes asociados contendrá entonces todos los datos que se han actualizado desde la última encuesta realizada. Para evitar la alta carga de trabajo del servidor y desechar los valores no actualizados, se introdujo un enfoque más sofisticado haciendo que el servidor retrase las respuestas de las solicitudes que se realizan a los datos, mediante el llamado “tiempo de espera”, de esta forma el servidor envía un mensaje de respuesta inmediato con los cambios que se realizaron a los datos, éste método ofrece enormes ventajas en cuanto a la comunicación y la latencia del tráfico en la red.

OPC XML-DA ofrece los siguientes servicios de suscripción los cuales también se denominan transacciones (10):

- *Subscribe:* es utilizado para iniciar un contrato de suscripción con el servidor.
- *SubscriptionPolledRefresh:* es llamado periódicamente para adquirir los últimos cambios de valor de la variable.
- *SubscriptionCancel*: se usa para terminar el contrato de suscripción con el servidor.

La forma de interactuar es la siguiente: El cliente inicia la suscripción y el servidor devuelve un “*handle*” o código de manejo como respuesta a la petición. El servidor también retorna cualquier valor inicial [valor (*value*), calidad (*quality*), y fecha (*timestamp*) que esté listo y disponible si la opción *ReturnValuesOnReply* está en estado verdadero. Posteriormente el cliente entra en un ciclo de registro y continúa escribiendo periódicamente por medio de la publicación de una petición de actualización, pasando cada vez el código de manejo de suscripción. El servidor responde inmediatamente devolviendo todos los valores y/o los cambios desde el registro anterior. Este proceso continúa hasta que el cliente no desea mantener más la suscripción, punto en el cual publica una petición de cancelación de suscripción al servidor. El servidor libera los recursos y lleva a cabo todas las acciones necesarias para terminar el contrato de suscripción que llevaba con el cliente (10).

Los contratos simples que ofrecen el cliente y el servidor son: *Browse, GetProperties, GetStatus, Read y Write. Browse* busca jerárquicamente los nombres de las variables que se encuentran en el servidor; *GetProperties* retorna información asociada con una o más variables; *GetStatus* retorna información acerca del servidor, tal como: versión, modo actual, estado general, entre otros. *Read* devuelve el valor, calidad y tiempo de consulta de algunas de las

variables seleccionadas con el método *Browse* o dando un nombre específico y *Write* permite escribir en el servidor uno o más valores de las variables (10).

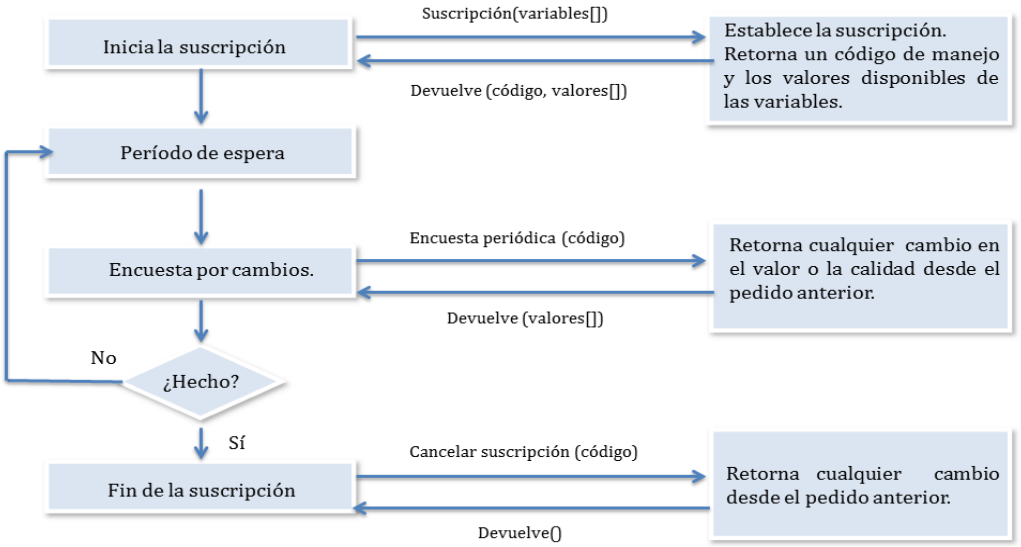

Figura 5. Arquitectura de suscripción de OPC XML-DA (9).

1.5.2 Ventajas de OPC XML-DA.

Dentro de las ventajas que brinda OPC XML-DA podemos encontrar (9):

- Capacidad de publicar datos de la planta de la fábrica mediante Internet, a través de cualquier *firewall*.
- La capacidad de utilizar XML, HTTP, SOAP y la tecnología de Internet para desarrollar clientes y servidores de plataformas ajenas a Microsoft (como Linux, Mac OS, Palm OS, y otros).
- La capacidad de publicar datos de la planta en formatos que son fácilmente utilizados por las aplicaciones empresariales, tales como, planificación, planeación de manufactura, calidad y la gestión de activos.

1.5.3 Usos de OPC XML-DA.

Dentro de los productores de software que aplican este estándar se encuentran: *Advosol*, proveedor líder de componentes y servicios para .NET basados en soluciones OPC. *ICONICS* situada en la sede de *Foxborough*, *MA*. Fundada en 1986, es un proveedor líder para la Web, basado en OPC, en sistemas SCADA y en software de aplicaciones inteligentes, para el sistema operativo *Windows* (11). *Kassl GmbH* es una empresa alemana que se especializa en el desarrollo de software y componentes para OPC. *Northern Dynamic*, es una marca de software de *Software ToolsBox Inc*., líder mundial en la fabricación de componentes de

productos de software para la automatización. Sus soluciones se basan en aprovechar la amplia experiencia y conocimientos de la industria para la aplicación de la tecnología OPC en el proceso industrial de control (12). *Kepware* líder mundial en software se centra en el desarrollo de *drivers* de comunicación para controladores de automatización, dispositivos de E / S y de campo basados en OPC y sistemas embebidos.

1.6 Escenario actual del módulo de Integración con terceros del GALBA.

El SCADA Guardián del ALBA es una solución de supervisión y control orientado inicialmente hacia la industria petrolera, pero con la visión de solventar las necesidades de supervisión y control de procesos de otras industrias para los países del ALBA y contribuir a la soberanía tecnológica de los mismos. Este posee una arquitectura distribuida, permitiendo así que los módulos que lo componen puedan trabajar de forma independiente en el control de sus procesos.

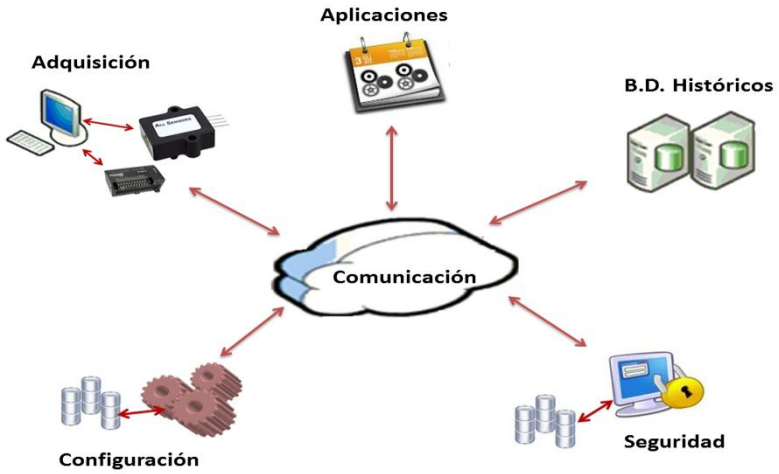

Figura 6. Arquitectura del sistema SCADA Guardián del ALBA.

En su enfoque distribuido, el GALBA tiene la necesidad de proporcionar interfaces de comunicación, que permitan una integración global entre todos los objetos del negocio, tanto de supervisión, control, como aplicaciones gerenciales.

Por tal motivo se necesitó una solución de comunicación distribuida que permitiera, a agentes externos al Guardián del ALBA, acceder a los servicios de intercambio de información relacionada con variables del sistema (puntos), alarmas, eventos y la interacción a través de comandos.

En el Guardián del ALBA, se previó la obtención de datos de terceros, sólo por la vía el estándar OPC DA 2.05. Para ello se diseñó una pasarela (*Gateway* OPC DA) que cumple con

requisitos de escalabilidad y funcionalidad, teniendo restricción de portabilidad debido que la capa de comunicación OPC es dependiente de la tecnología COM/DCOM de Microsoft. El estándar utilizado para ello fue OPC con la especificación *Data Access* (DA) v2.05a. La aplicación *Gateway* OPC DA posee dos subsistemas: el Explorador *Gateway* OPC DA (E-GOPC) y el Panel de Control *Gateway* OPC DA (PC-GOPC).Dentro de las funcionalidades que ofrece el servidor de OPC DA se encuentran: (13)

- Permitir agregar variables al espacio de trabajo del servidor OPC, con la nueva propiedad del identificador (Id), usadas como identificador de cada variable en el SCADA.
- Permitir conexión de uno o más clientes de OPC al servidor.
- Permitir explorar el espacio de trabajo del servidor desde un cliente OPC.
- Lectura de una variable desde un cliente OPC.
- Escritura de una variable desde un cliente OPC.
- Creación de grupos de muestreo por frecuencia.

En el año 2011 se comenzó el desarrollo de un servidor OPC XML-DA. Para ello se adoptaron las tecnologías XML con el objetivo de facilitar el intercambio de información entre aplicaciones pero en vez de utilizar tecnología COM/DCOM como en OPC-DA utilizar mensajes SOAP (sobre HTTP) con documentos en XML. El servidor debía cumplir con todas las funcionalidades de la especificación OPC XML-DA para su funcionamiento exitoso pero después de haberle realizado un análisis y un proceso de pruebas resultaron varios errores en la comunicación con clientes y dentro de sus funcionalidades específicas. Los errores encontrados fueron los siguientes:

- No se encuentra implementada la funcionalidad: "*Browse*", de vital importancia para el proceso de lectura escritura entre cliente y servidor.
- Existen errores de funcionamiento en las funcionalidades: *SubscriptionPolledRefresh*, *GetProperties* y *Read*. Además ninguna de las funcionalidades permite definir las *RequestOptions* como parámetro sin generar un error.
- El servidor no establece comunicación con ninguna de las siguientes aplicaciones probadas que implementan la especificación OPC XML-DA:
  - ✓ dOPC DA CLient Version 5.0 Kassl GmbH
  - ✓ Kassl OPC Explorer Version 5.6.2.20
  - ✓ Kassl-OPCXMLDAClient

1.7 Metodología, lenguajes y herramientas de desarrollo.

A continuación se describe la metodología, lenguajes y herramientas utilizadas para el desarrollo del subsistema de comunicación a través de la especificación OCP XML-DA. Los mismos son exponentes del movimiento de software libre y se ajustan a las necesidades del proyecto SCADA GALBA.

1.7.1 Metodología de desarrollo: RUP.

RUP (*Rational Unified Process*), el proceso unificado racional es un proceso de desarrollo de software que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto (14). Se caracteriza por ser dirigida por caso de uso, centrada en la arquitectura, iterativo e incremental, además proporciona una visión completa de la construcción del producto. Dentro de sus principales ventajas están la mitigación temprana de posibles riesgos altos, la proporción de un lenguaje común y además que permite a los usuarios estar involucrados continuamente en el desarrollo del software.

1.7.2 Lenguaje de programación: C++.

C++ es un lenguaje muy potente para el desarrollo de aplicaciones críticas o complejas, por la facilidad que brinda la manipulación de la memoria del ordenador a través de los punteros. Como lenguaje orientado a objetos brinda ventajas propias de esta filosofía de programación, como son: la abstracción, el encapsulamiento, la herencia y el polimorfismo (15)**.** Una de las características más interesantes del lenguaje es la sobrecarga de operadores. Esto significa que a los operadores internos se les puede redefinir la semántica: se pueden escribir funciones que en vez de tener un nombre, se asocian a un operador, que debe tener por lo menos un parámetro de tipo clase.

1.7.3 Biblioteca de C++: Boost.

Boost es un conjunto de bibliotecas de software libre y revisión por pares preparadas para extender las capacidades del lenguaje de programación C++. Su diseño e implementación permiten que sea utilizada en un amplio espectro de aplicaciones y plataformas. Abarca desde bibliotecas de propósito general hasta abstracciones del sistema operativo. Actualmente Boost está formada por más de 80 bibliotecas individuales que en su mayoría están basadas en cabeceras, funciones en línea y plantillas, por lo que no tienen que ser construidas antes de su uso (16)**.**

1.7.4 Lenguaje de modelado: UML.

El Lenguaje Unificado de Modelado ayuda a especificar, visualizar y documentar modelos de sistemas de software, incluyendo su estructura y diseño, de forma tal que cumpla con todos los requisitos descritos anteriormente. Permite modelar casi cualquier tipo de aplicación, que se ejecute en cualquier tipo de combinación de *hardware*, sistema operativo, lenguaje de programación o infraestructura de red. Está diseñado para ser usado sobre los fundamentos de la orientación a objetos, convirtiéndolo en un marco ideal para los lenguajes orientados a objetos como C++, Java y C#, teniendo uso limitado para otros paradigmas de programación. UML no brinda el total éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos al posibilitar una nueva y fuerte integración entre las herramientas, los procesos y los dominios (17).

1.7.5 Entorno de desarrollo: Eclipse.

Eclipse es un entorno integrado de código abierto independiente de una plataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores, es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web y programas en C++. Esta plataforma, típicamente ha sido usada para desarrollar Entornos Integrados de Desarrollo (IDE), como el IDE de Java (JDT) *Java Development Toolkit* y el compilador (ECJ) que se entrega como parte de Eclipse y que son usados también para desarrollar el mismo Eclipse (18).

1.7.6 Herramienta CASE: Visual Paradigm.

El *Visual Paradigm* es una potente herramienta CASE (*Computer-Aided Software Engineering*, Ingeniería de Software Asistida por Computadora) que proporciona un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación, ha sido seleccionada para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas (17).

1.7.7 *Framework* de desarrollo de servicios web: WSO2 WSF/C++.

WSO2 WSF/C++ (*WSO2 Web Services Framework for C++)* es una extensión de WSO2 WSF/C (*WSO2 Web Services Framework for C*), que proporciona un conjunto de normas y estándares de código abierto, siendo a su vez una solución completa para crear y desplegar

servicios web. La herramienta es liberada con licencia Apache v2.0, y se basa en la familia de proyectos de Apache de código abierto (19). El *framework* proporciona **Interfaz de Programación de Aplicaciones** (API) para la implementación de servicios web y clientes de servicios web. WSO2 WSF/C++ incluye (20):

- **Soporte de estándares para servicios web**: soporta estándares como SOAP 1.1, SOAP 1.2 y soporte de especificaciones como *WS-Addressing*, *WS-Policy*, *WS-Security y WS- Security Policy*.
- **Portabilidad y soporte de plataformas:** disponible para plataformas como Windows XP - Microsoft Visual C++ 8.0, Linux – gcc/g++ 4.1.1, *Sun Solaris* 2.10 x86 - gcc 3 y MacOS 10.4.10 - gcc/g++ 4.0.1.
- **Generación de código WSDL2CPP:**WSO2 WSF/C++ viene con la herramienta de generación de código que genera recibos de los clientes (*client stub)* y los esqueletos de servicio para un determinado WSDL (*Web Services Definition Language*).
- **Extensiones de Servidores Web**: puede implementarse como una extensión del lado del servidor en un servidor Web, como *Apache Web Server* o Microsoft *IIS*. También incluye un servidor HTTP y puede desplegarse como un servidor independiente.

1.7.8 *Framework* para interfaces gráficas: Qt.

Qt es una biblioteca multipropósito que permite el desarrollo de interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola. Se puede integrar con distintos IDE existentes, como es el caso de Eclipse, Microsoft *Visual Studio*, *Code Blocks*, entre otros. Utiliza el lenguaje de programación C++ de forma nativa, además existen interfaces para Python (PyQt), Java (QtJambi), Perl (PerlQt), Gambas (Gb.Qt), Ruby (QtRuby), PHP (PHP-Qt), Mono (Qtoto), entre otros (21).

Funciona en todas las principales plataformas, y tiene un amplio apoyo. El API de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos, además de estructuras de datos tradicionales *(21)*.

1.7.9 Servidor Web: Apache.

Apache es el servidor web encargado de interpretar el protocolo HTTP, cuyo nombre proviene de la frase inglesa “*a patchy server*” y es completamente libre, ya que es un software de código abierto y bajo la Licencia Pública General (GPL). Una de las ventajas más grandes de Apache, es que es un servidor web multiplataforma ya que puede trabajar con diferentes sistemas

operativos (Linux, Windows, Mac) y mantener su excelente rendimiento. Es utilizado principalmente, para realizar servicio a páginas web estáticas y dinámicas y se puede integrar a la perfección con otras aplicaciones (22).

1.8 Conclusiones.

En este capítulo, se realizó un análisis de la documentación referente a los sistemas SCADA y la comunicación que establecen con otros sistemas, así como el estado del subsistema de integración con terceros en el SCADA GALBA. Además se estudió el estándar OPC y su especificación XML-DA, y se describieron las herramientas, tecnologías y metodologías utilizadas para el diseño e implementación de la solución.

CAPÍTULO 2
ANÁLISIS Y DISEÑO DEL SISTEMA

2.1 Introducción.

En este capítulo se expone el flujo de trabajo Análisis y Diseño realizado para modelar y dar respuesta al problema planteado. Se desarrolla el modelo de dominio donde se expone un marco conceptual y se establecen las relaciones entre los conceptos que lo conforman. Se especifican los requisitos de software tanto funcionales como no funcionales y se representan los actores y casos de uso a través de un diagrama de casos de uso. Se realizan los diagramas de clase del diseño así como los diagramas de secuencia. Además se describen los patrones de diseño y el modelo arquitectónico que sigue el subsistema para la comunicación con terceros a través del estándar OPC XML-DA.

2.2 Modelo de dominio.

El Modelo de Dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, conceptos del mundo real en lugar de componentes de software. A continuación se representa y se describen los conceptos fundamentales del dominio del sistema.

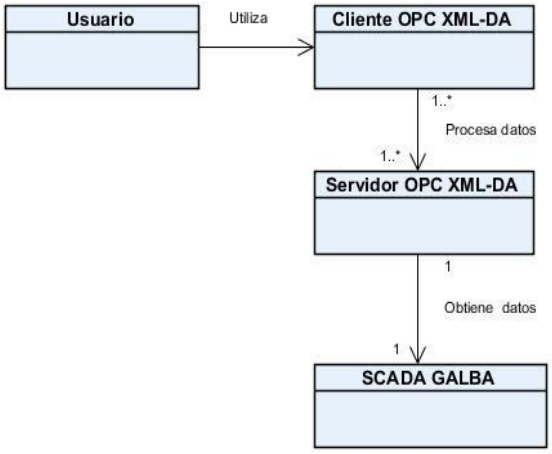

Figura 7. Modelo del dominio.

- **Usuario:** es la persona capacitada para el manejo y funcionamiento del subsistema para la comunicación con terceros.

- **Cliente OPC XML-DA:** es el encargado de manipular toda la información correspondiente a las variables y establecer la comunicación con el servidor.
- **Servidor OPC XML-DA:** es quien recibe los datos del cliente y procesa la información que obtiene del SCADA GALBA.
- **SCADA GALBA:** representa la fuente de datos del servidor OPC XML-DA.

2.3 Especificación de los requisitos de software.

La especificación de los requerimientos de software constituye un elemento de vital importancia para la elaboración de un software de calidad superior. Es necesario hacer énfasis en la precisión con que se debe realizar esta tarea pues tiene un papel primordial en la implementación de la interfaz, y se enfoca en un área fundamental: la definición de lo que se desea producir. A continuación se enuncian los requisitos funcionales (RF) y no funcionales (RNF) determinados para desarrollar el subsistema.

2.3.1 Requisitos funcionales.

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Estos no alteran la funcionalidad del producto, lo que quiere decir que se mantienen invariables sin importarle con que propiedades o cualidades se relacionen (23). A continuación se exponen los requisitos funcionales identificados para el desarrollo del subsistema de comunicación a través de OPC XML-DA.

- **RF 1:** El sistema debe permitir crear grupos.
- **RF 2:** El sistema debe permitir eliminar grupos.
- **RF 3:** El sistema debe permitir activar grupos.
- **RF 4:** El sistema debe permitir desactivar grupos.
- **RF 5:** El sistema debe permitir adicionar servidores OPC XML-DA.
- **RF 6:** El sistema debe permitir eliminar los servidores OPC XML-DA.
- **RF 7:** El sistema debe permitir conectar servidores OPC XML-DA.
- **RF 8:** El sistema debe permitir desconectar servidores OPC XML-DA.
- **RF 9:** El sistema debe permitir consultar el estado de los servidores OPC XML-DA.
- **RF 10:** El sistema debe permitir adicionar variables.
- **RF 11:** El sistema debe permitir leer variables.
- **RF 12:** El sistema debe permitir modificar los valores de las variables.
- **RF 13:** El sistema debe permitir consultar las propiedades de las variables.
- **RF 14:** El sistema debe permitir eliminar las variables.

- **RF 15:** El sistema debe permitir explorar los servidores OPC XML-DA.
- **RF 16:** El sistema debe permitir abrir una configuración.
- **RF 17:** El sistema debe permitir guardar una configuración.
- **RF 18:** El sistema debe permitir la escritura de variables en el servidor.

A continuación, los requisitos funcionales serán agrupados en casos de uso del sistema (CUS). En este sentido los casos de uso identificados son los siguientes:

CUS 1: Gestionar Grupo.

- RF 1: El sistema debe permitir crear grupos.
- RF 2: El sistema debe permitir eliminar grupos.
- RF 3: El sistema debe permitir activar grupos.
- RF 4: El sistema debe permitir desactivar grupos.

CUS 2: Gestionar servidor.

- RF 5: El sistema debe permitir adicionar servidores OPC XML-DA.
- RF 6: El sistema debe permitir eliminar los servidores OPC XML-DA.
- RF 7: El sistema debe permitir conectar servidores OPC XML-DA.
- RF 8: El sistema debe permitir desconectar servidores OPC XML-DA.

CUS 3: Consultar estado de servidor.

- RF 9: El sistema debe permitir consultar el estado de los servidores OPC XML-DA.

CUS 4: Operar variable.

- RF 10: El sistema debe permitir adicionar variables.
- RF 11: El sistema debe permitir leer variables.

CUS 5: Administrar variable.

- RF 12: El sistema debe permitir modificar los valores de las variables.
- RF 13: El sistema debe permitir consultar las propiedades de las variables.
- RF 14: El sistema debe permitir eliminar las variables.

CUS 6: Explorar servidor.

- RF 15: El sistema debe permitir explorar los servidores OPC XML-DA.

CUS 7: Administrar Configuración.

- RF 16: El sistema debe permitir abrir una configuración.
- RF 17: El sistema debe permitir guardar una configuración.

CUS 8: Escribir variables en el servidor.

- RF 18: El sistema debe permitir la escritura de variables en el servidor.

2.3.2 Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto. A continuación se exponen los requisitos no funcionales con los que debe cumplir el subsistema.

RNF 1. Software.

- RNF 1.1 El subsistema para la comunicación a través de OPC XML-DA se debe desarrollar sobre el Sistema operativo GNU/Linux, distribución Debian.

RNF 2. Apariencia o Interfaz Externa.

- RNF 2.1 La interfaz debe ser sencilla y amigable. Diseño sencillo e intuitivo, ofreciendo una interfaz de fácil manejo por parte de los usuarios, permitiendo lograr uniformidad y dinamismo en la solución.

RNF 3. Diseño e implementación.

- RNF 3.1. En el desarrollo del subsistema para la comunicación a través de OPC XML-DA se debe utilizar el lenguaje de programación C++.
- RNF 3.2. En el desarrollo del subsistema para la comunicación a través de OPC XML-DA se debe utilizar como paradigma de programación, la Programación Orientada a Objetos.
- RNF 3.3. El código debe cumplir con los estándares de codificación establecidos por la especificación OPC XML-DA.
- RNF 3.4. En el desarrollo del subsistema para la comunicación a través de OPC XML-DA se debe utilizar el *framework* de desarrollo WSO2 WSF/C++.

RNF 4. Portabilidad.

- RNF 4.1 El subsistema para la comunicación a través de OPC XML-DA debe garantizar que los terceros accedan a él sin importar la plataforma que estos utilicen.

RNF 5. Fiabilidad.

- RNF 5.1. El subsistema para la comunicación a través de OPC XML-DA debe garantizar una comunicación donde no ocurran pérdidas de información (variables).

RNF 6. Soporte.

- RNF 6.1. El subsistema para la comunicación a través de OPC XML-DA debe permitir una amplia interoperabilidad para que aplicaciones elaboradas por terceros e implementadas en diferentes lenguajes y sistemas operativos puedan comunicarse con este de forma fácil y eficiente.

2.4 Modelo de casos de uso del sistema.

El modelo de casos de uso del sistema describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del cliente. Está formado por actores, casos de uso y las relaciones que se establecen entre ellos, representando gráficamente a los procesos y su interacción con los actores y constituye una entrada de gran valor para las siguientes fases de construcción de un software.

2.4.1 Actores del sistema.

Un actor del sistema es una persona o la abstracción de un software; son terceros fuera del sistema: puede intercambiar información con él, representar el rol que juegan una o varias personas, puede ser un recipiente pasivo de información, un equipo o un sistema automatizado. En el subsistema en desarrollo se definen los siguientes actores del sistema:

- **Operador del sistema:** Representa al usuario que va a hacer uso de la aplicación, teniendo la posibilidad de interactuar con las funcionalidades que le brinda la misma.
- **Sistema externo:** Es el actor encargado de escribir los datos en el servidor.

2.4.2 Diagrama de casos de uso del sistema.

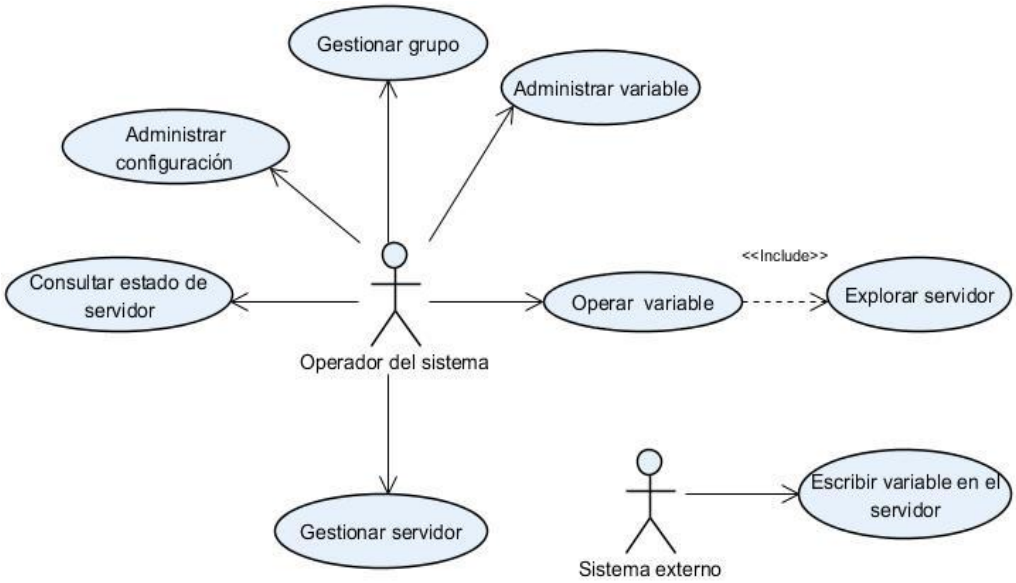

Figura 8. Diagrama de casos de uso del sistema.

2.4.3 Descripción de los casos de uso del sistema.

2.4.3.1 CUS Gestionar servidor.

Caso de uso	Gestionar servidor
Actor	Operador del sistema
Resumen	El caso de uso inicia cuando el Operador del sistema desea adicionar, eliminar, conectar o desconectar un servidor.
Referencias	RF 5,RF 6, RF 7,RF8
Prioridad	Secundario
Precondiciones	Debe existir un servidor OPC XML-DA.
Poscondiciones	Se adiciona, elimina, conecta o desconecta un servidor.

Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El actor accede al sistema para adicionar, eliminar, conectar, o desconectar un servidor.	2. El sistema muestra la interfaz principal con el panel de servidores y los grupos con los que cuenta. Si el actor desea: • Adicionar servidor, ver Sección 1. • Eliminar servidor, ver Sección 2. • Conectar servidor, ver Sección 3. • Desconectar servidor, ver Sección 4.
Sección 1: "Adicionar Servidor"	
1. Clic derecho en el panel de servidores.	2. Muestra una ventana con el campo requerido para adicionar un servidor. • URL
3. Introduce el dato requerido. 4. Presiona el botón Aceptar.(Alterno 1)	5. Adiciona el servidor. 6. Termina el caso de uso.
Flujo alterno de eventos	
(Alterno 1) <u>Presiona el botón Cancelar</u>	
Acción del actor	**Respuesta del sistema**
1. En el paso 4 del flujo normal, presiona el botón Cancelar.	2. Cierra la ventana Adicionar servidor. 3. Cancela el proceso de Adicionar servidor.

	4. Termina el caso de uso.
Sección 2: "Eliminar Servidor"	
1. Clic derecho sobre el servidor que se desea eliminar.	2. El sistema muestra las opciones: • Adicionar grupo • Conectar Servidor • Desconectar servidor • Eliminar servidor • Estado del servidor
3. Selecciona la opción Eliminar servidor.	4. Elimina los grupos asociados al servidor. 5. Elimina el servidor y muestra mensaje de confirmación. 6. Termina el caso de uso.
Sección 3: "Conectar servidor"	
1. Clic derecho sobre el servidor.	2. El sistema muestra las opciones: • Adicionar grupo • Conectar Servidor • Desconectar servidor • Eliminar servidor • Estado del servidor
3. Selecciona la opción Conectar servidor.	4. Verifica que el servidor que se desea conectar está corriendo.(Alterno 1) 5. Conecta el servidor. 6. Termina el caso de uso.
Flujo alterno de eventos	
(Alterno 1) <u>Servidor corriendo</u>	
Acción del actor	**Respuesta del sistema**
	1. Si el servidor no está corriendo se muestra el mensaje de error: "No se puede conectar el servidor" 2. Termina el caso de uso.
Sección 4: "Desconectar servidor"	
1. Clic derecho sobre el servidor.	2. El sistema muestra las opciones: • Adicionar grupo • Conectar Servidor

	• Desconectar servidor • Eliminar servidor • Estado del servidor
3. Selecciona la opción Desconectar servidor.	4. Desactiva todos los grupos asociados al servidor. 5. Desconecta el servidor. 6. Termina el caso de uso.

Tabla 1. Descripción del CUS Gestionar servidor.

2.4.3.2 CUS Administrar variable.

Caso de uso	Administrar variable
Actor	Operador del sistema
Resumen	El caso de uso inicia cuando el Operador del sistema desea eliminar, modificar o consultar las propiedades de las variables.
Referencias	RF 12, RF 13,RF 14
Prioridad	Crítico
Precondiciones	Debe existir un grupo con un conjunto de variables asociadas.
Poscondiciones	Se elimina, modifica o muestra las propiedades de una variable.
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El actor da clic derecho sobre una variable.	2. El sistema muestra las opciones: • Eliminar variable, ver Sección 1. • Modificar valor, ver Sección 2. • Propiedades, ver Sección 3.
Sección 1: “Eliminar variable”	
1. Selecciona la opción Eliminar variable.	2. Elimina la variable. 3. Termina el caso de uso.
Sección 2: “Modificar variable”	
1. Selecciona la opción Modificar valor.	2. Verifica que la variable sea de tipo escritura. (Alterno 1) 3. Muestra una ventana con los campos requeridos para escribir el ítem.

	• Valor actual • Nuevo valor
4. Llena el campo requerido. 5. Presiona el botón Aceptar. (Alterno 2)	6. Escribe la variable. 7. Termina el caso de uso.
Flujo alterno de eventos	
(Alterno 1) <u>Tipo de acceso</u>	
Acción del actor	**Respuesta del sistema**
	1. Si el tipo de acceso de la variable no es de escritura muestra el mensaje: "La variable es de solo lectura". 2. Termina el caso de uso.
(Alterno 2) <u>Presiona el botón Cancelar</u>	
Acción del actor	**Respuesta del sistema**
1. En el paso 5 del flujo normal, presiona el botón Cancelar.	2. Cierra la ventana Modificar variable. 3. Cancela el proceso de Modificar variable. 4. Termina el caso de uso.
Sección 3: "Propiedades"	
1. Selecciona la opción Propiedades.	2. Muestra las propiedades de la variable. • Nombre. • Valor. • Camino. • Calidad. • Estampa de tiempo. • Información • Tipo de acceso 3. Termina el caso de uso.

Tabla 2. Descripción del CUS Administrar variable.

2.5 Diagramas de interacción.

Un diagrama de interacción consiste en un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Existen dos tipos de diagrama de interacción: diagramas de secuencia y diagramas de colaboración. Para el desarrollo del diseño del sistema se seleccionó el diagrama de secuencia como la forma de representar el comportamiento dinámico del sistema. A continuación se muestran los diagramas de secuencia realizados para cada caso de uso.

2.5.1. CU Gestionar servidor.

Sección "Adicionar servidor"

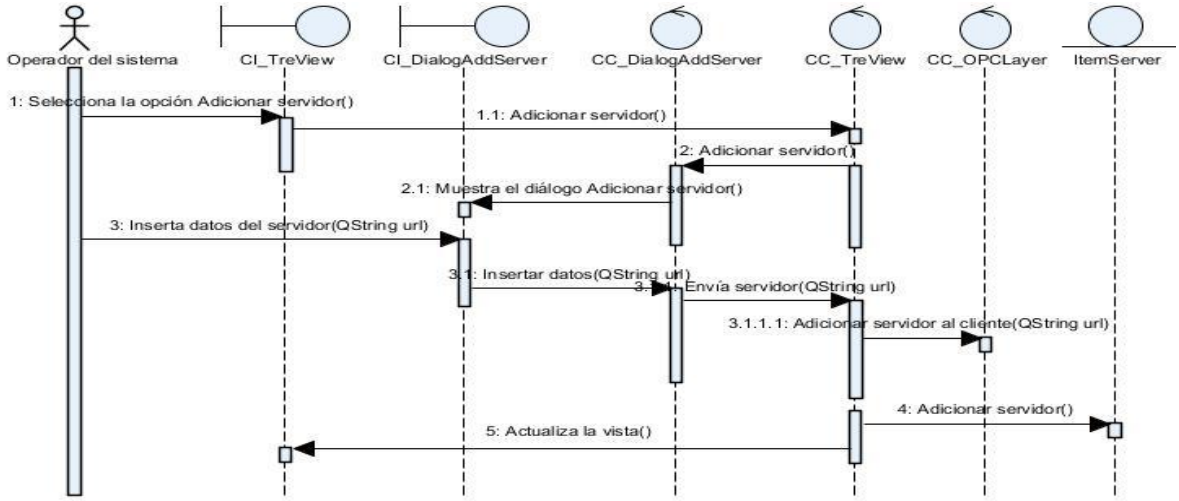

Figura 9. Diagrama de secuencia Adicionar servidor.

Sección "Conectar servidor"

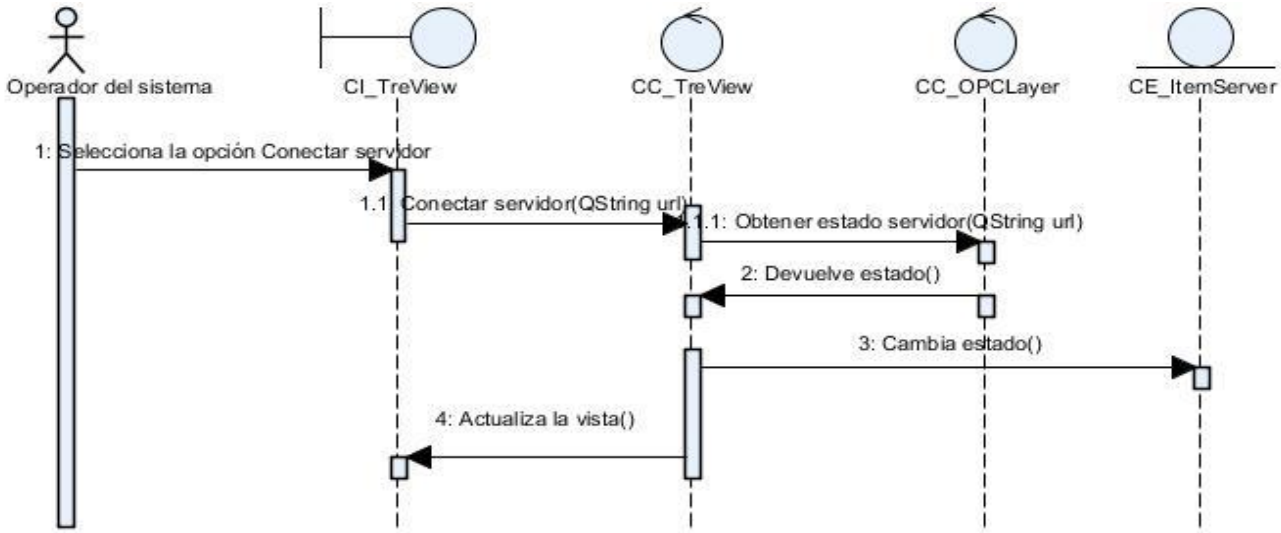

Figura 10. Diagrama de secuencia Conectar servidor.

2.6 Modelo del diseño.

En esta fase se modela el sistema de forma tal que soporte los requisitos funcionales y no funcionales mencionados anteriormente, sentando así las bases para poder realizar las actividades correspondientes a la fase de implementación. El siguiente esquema muestra los distintos paquetes que componen el subsistema de comunicación con terceros a través de OPC XML-DA, cuyos propósitos se explican a continuación:

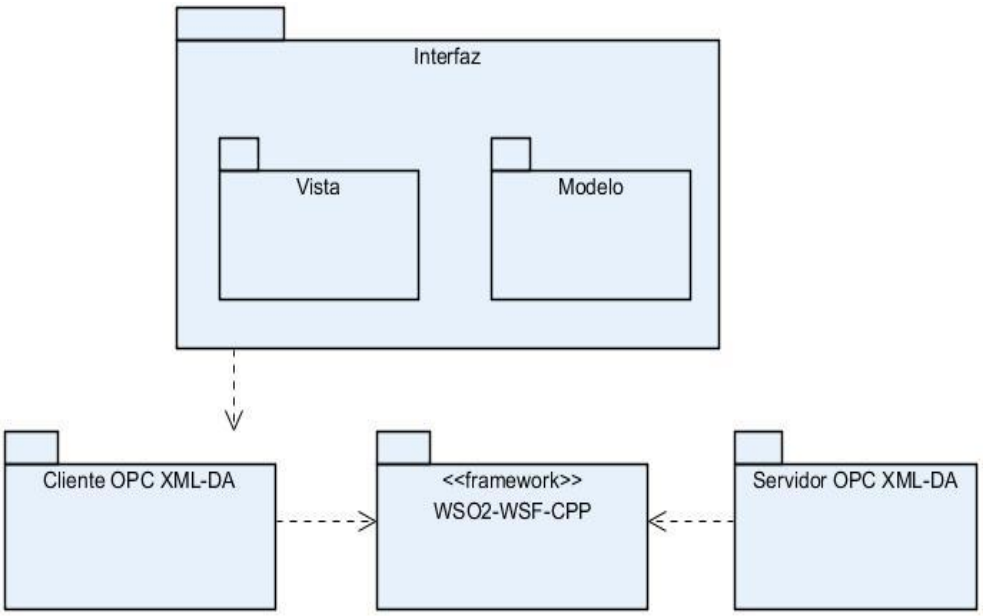

Figura 11. Diagrama de paquetes.

2.6.1 Interfaz.

En este paquete es donde se encuentra el diseño de la interfaz gráfica de la aplicación. Este paquete se divide en dos partes:

- **Vista:** agrupa las clases que implementan la lógica de dicho paquete encargadas de mostrar la interfaz gráfica con la que va a interactuar el usuario.
- **Modelo:** contiene las clases que implementan la lógica que rigen a la biblioteca OPC_XMLDA_Client.

A continuación se muestra el diagrama de clases del diseño correspondiente a este paquete así como una breve descripción de cada una de las clases que lo componen:

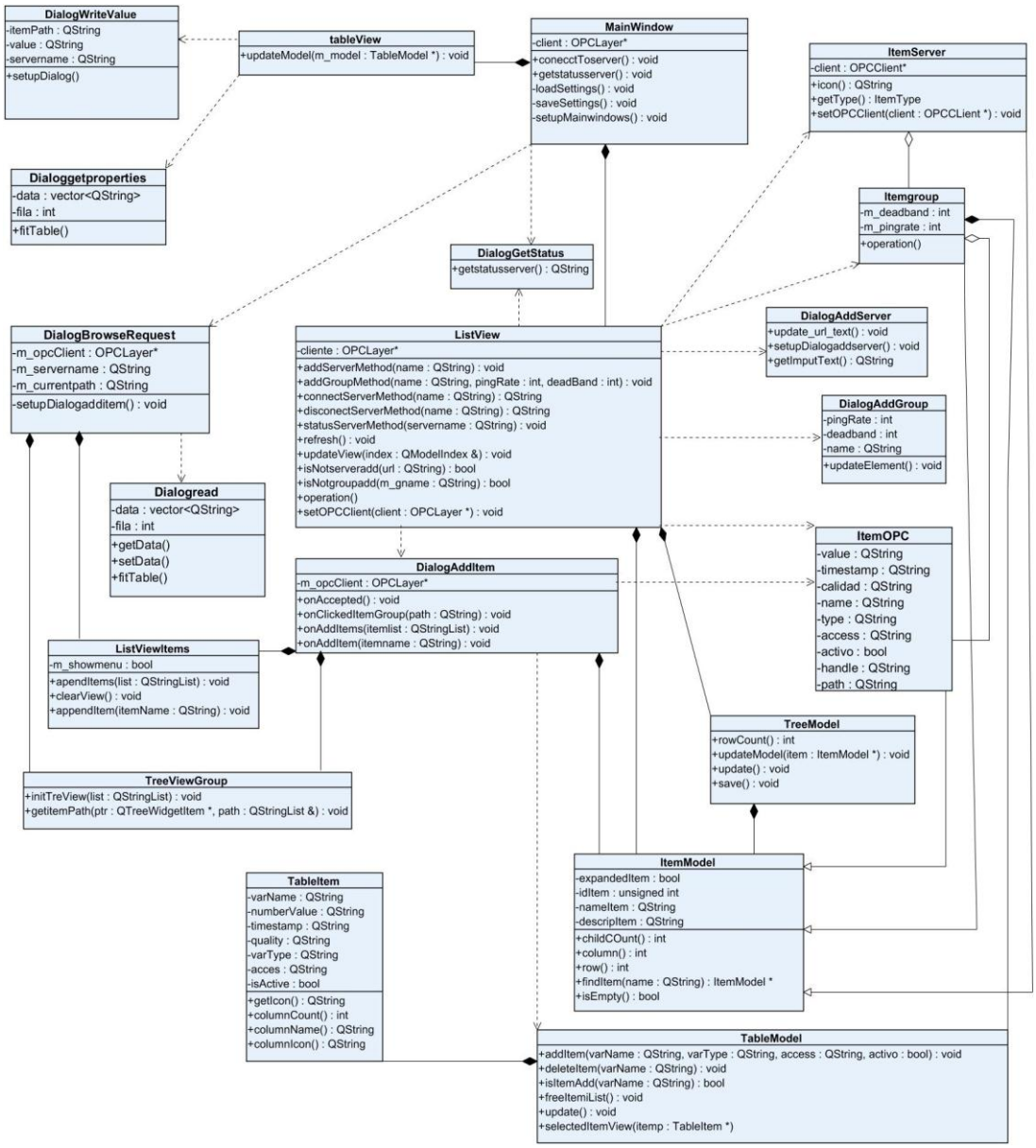

Figura 12. Diagrama de clases del paquete Interfaz.

Breve descripción de las clases del paquete Interfaz.

- ***MainWindows*:** Es la clase encargada de gestionar los componentes visuales y de crear el objeto de la clase OPCClient que permite el acceso a la biblioteca OPC_XMLDA_Client.
- ***ListView*:** Se encarga de la invocación de funcionalidades de la biblioteca OPC_XMLDA_Client a través del objeto que recibe de la clase *MainWindows*.
- ***TableView*:** Permite visualizar los elementos contenidos dentro de los grupos.

- ***TreeViewGroup*:** Se encarga de visualizar las variables agrupadas por el path en forma de árbol después de haber realizado la llamada a la funcionalidad Explorar_Servidor.
- ***ListViewItems*:** Se encarga de visualizar las variables contenidas en cada grupo después de haber realizado la llamada a la funcionalidad Explorar_Servidor.
- ***DialogGetStatus*:** Representa la ventana de diálogo donde se muestra el estado del servidor.
- ***DialogAddItem*:** Es la clase contenedora de los elementos que visualizan el proceso de añadir y gestionar variables.
- ***DialogAddServer*:** Representa la ventana de diálogo para insertar la dirección URL de un servidor OPC XML-DA y permite adicionar el mismo a la lista de servidores.
- ***DialogAddGroup*:** Representa la ventana de diálogo que permite adicionar un nuevo grupo.
- ***DialogGetProperties*:** Clase encargada de mostrar las propiedades de las variables.
- ***DialogWriteValue*:** Clase que permite modificar el valor de una variable en dependencia de su tipo de acceso.
- ***DialogBrowseRequest*:** Clase encargada de mostrar las variables que contiene el servidor OPC-XML-DA a través del método Explorar_Servidor.
- ***DialogRead*:** Clase encargada de mostrar las propiedades de un conjunto de variables cuando se ejecuta la funcionalidad de leer variables.
- ***ListModel*:** Gestiona las operaciones sobre la vista del árbol de servidores.
- ***ItemModel***: Es la clase sobre la cual se crean todos los ítems que contiene la aplicación: servidores, grupos y variables y contiene las operaciones para trabajar sobre los mismos.
- ***TreeModel*:** Es la clase que funciona como modelo de la vista de árboles de servidores y grupos.
- ***TableModel*:** Se encarga de la representación visual de los campos que contienen los valores de las variables. Contiene una lista de *TableItem*.
- ***TableItem*:** Es la clase encargada de almacenar los atributos de cada variable en memoria.
- ***ItemServer*:** Clase específica para los ítems de tipo servidor, hereda de *ItemModel*.
- ***ItemGroup*:** Clase específica para los ítems de tipo grupo, hereda de *ItemModel*.
- ***ItemOPC*:** Clase específica para los ítems de tipo opc, hereda de *ItemModel*, se utiliza como contenedor de los datos de los ítems recibidos.

2.6.2 Cliente OPC XML-DA.

Este paquete contiene la biblioteca que brinda las funcionalidades de acceso al servidor. A continuación se muestra el diagrama de clases del diseño correspondiente a este paquete así como una breve descripción de cada una de las clases que lo componen:

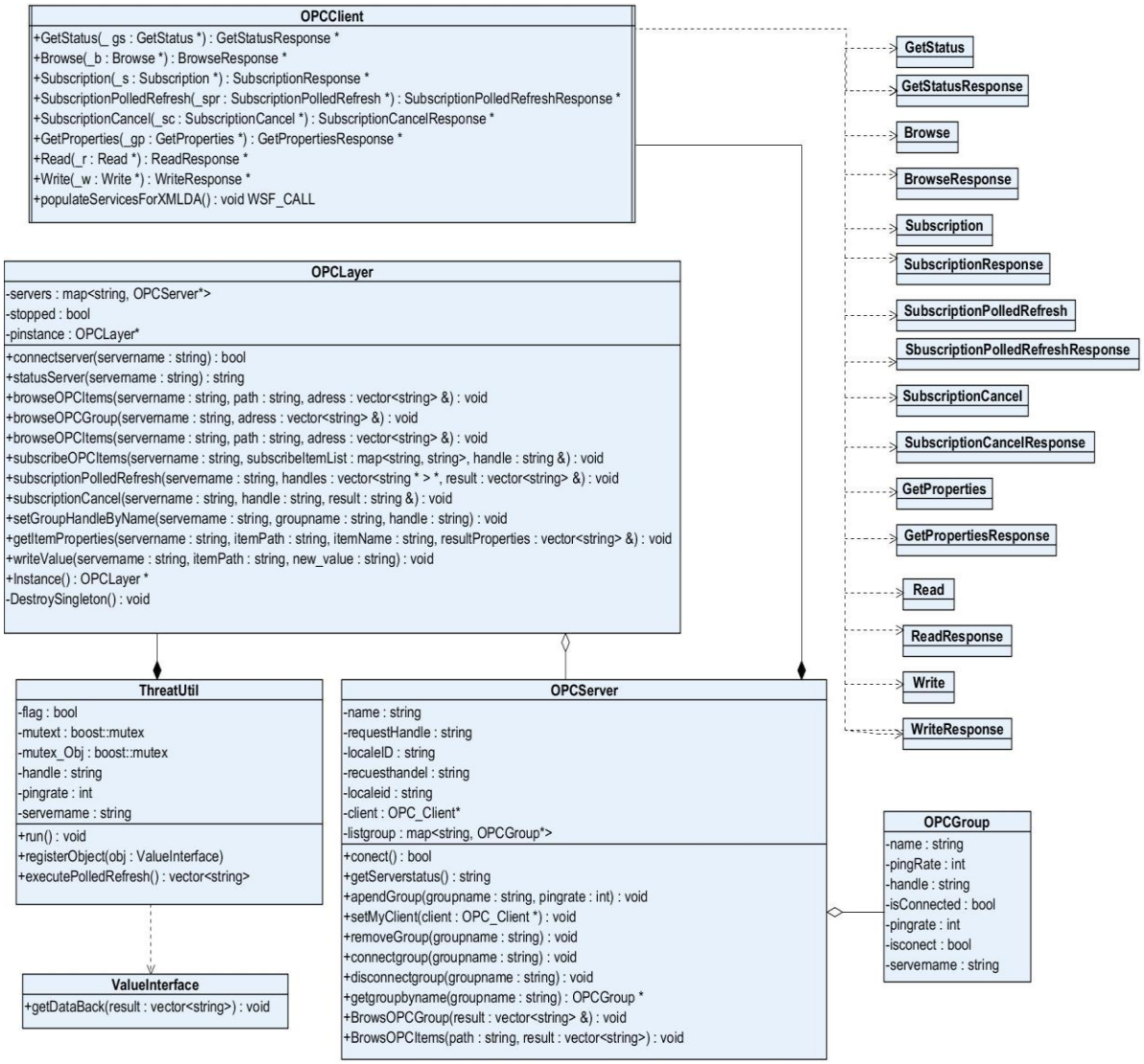

Figura 13. Diagrama de clases del paquete Cliente OPC XML-DA.

Breve descripción de las clases del paquete Cliente OPC XML-DA.

- ***OPCClient*:** Es la clase encargada de implementar las funcionalidades que permiten la comunicación con el servidor OPCXML-DA, a través de las interfaces que provee la clase Stub del *framework* WSO2/WSFCPP.

- ***OPCServer*:** Posee una instancia de OPCClient y una lista de grupos de variables, se encarga de invocar las funcionalidades de OPCClient.
- ***OPCLayer*:** Posee una lista de OPCServer, se encarga de gestionar las operaciones de los grupos y los servidores.
- ***OPCGroup*:** Se encarga de almacenar los atributos de cada grupo en memoria.
- ***ThreatUtil*:** Contiene un hilo que se encarga de la llamada de forma periódica de la funcionalidad SubscriptionPolledRefresh, para obtener los últimos cambios de valor de la variable, y devuelve el valor a través de una instancia de la clase *ValueInterface*.
- ***ValueInterface*:** Contiene el método para obtener los datos de la encuesta periódica.

Las clases *GetStatus*, *GetStatusResponse*, *Browse*, *BrowseResponse*, *Subscription*, *SubscriptionPolledRefresh*, *SubscriptionPolledRefreshResponse*, *SubscriptionResponse*, *SubscriptionCancel*, *SubscriptionCancelResponse*, *GetProperties*, *GetPropertiesResponse*, *Read*, *ReadResponse*, *Write* y *WriteResponse* son clases definidas tanto para el cliente como para el servidor pues son autogeneradas por el *framework* WSO2/WSFCPP y de manera general permiten la comunicación a través de los servicios web.

2.6.3 Servidor OPC XML-DA.

Este paquete contiene las clases que implementan la especificación OPC XML-DA a través de la biblioteca OPC_XMLDA_Server. A continuación se muestra el diagrama de clases del diseño correspondiente a este paquete así como una breve descripción de cada una de las clases que lo componen:

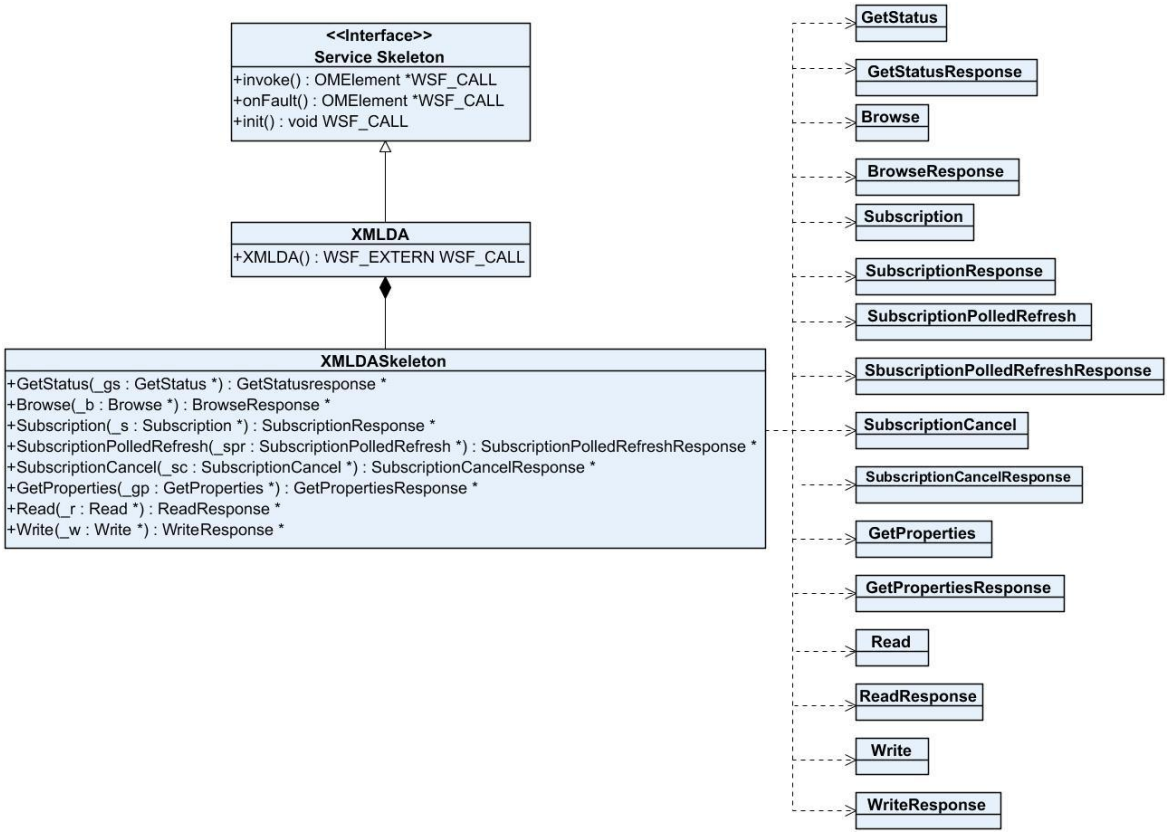

Figura 14. Diagrama de clase del paquete Servidor OPC XML-DA.

Breve descripción de las clases del paquete Servidor OPC XML-DA.

- ***ServiceSkeleton*:** Representa la interfaz que debe ser implementada por cualquier servicio en C++ con WSF/CPP. Define tres métodos virtuales que deben ser implementados en la clase que herede de ella.
- ***XMLDA*:** Clase que hereda de *ServiceSkeleton*, implementa los métodos de la lógica de negocio de los servicios web.
- ***XMLDASkeleton*:** Se encarga de implementar las funcionalidades de la especificación OPC XML- DA, que serán invocadas por el cliente.

2.6.4 WSO2-WSF-CPP.

Este paquete representa el framework que permite el consumo e implementación de servicios Web en C++, pertenece a la suite WSO2, y es utilizado tanto por el cliente como por el servidor. Esta herramienta es la única dependencia entre ambos, y al tener soporte para plataformas Windows y Linux, permite que la solución brindada sea multiplataforma.

2.7 Arquitectura en capas.

La arquitectura en capas es un estilo de programación donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio y mecanismos de almacenamiento (24). Para el desarrollo del subsistema de comunicación a través de OPC XML-DA se tuvo en cuenta una arquitectura en tres capas. A continuación se explica la disposición por capas y sus responsabilidades:

En el cliente OPC XML-DA:

- **Capa de presentación:** en esta capa se encuentra la interfaz gráfica del cliente que permite mostrar la información obtenida del servidor a través de la biblioteca OPC_XMLDA_Client.
- **Capa de lógica de negocio:** en esta capa se encuentra la biblioteca OPC_XMLDA_Client, que implementa las funcionalidades de acceso al servidor siguiendo la especificación OPC XML-DA.
- **Capa de transporte:** esta capa se encarga de la invocación a métodos del *framework* WSF2CPP que permiten la comunicación mediante SOAP sobre HTTP.

En el servidor OPC XML-DA:

- **Capa de presentación:** esta capa representa la aplicación Axis2 que permite publicar el servidor desarrollado como un servicio web.
- **Capa de lógica de negocio:** en esta capa se encuentra la biblioteca OPC_XMLDA_Server, que implementa la especificación OPC XML-DA.
- **Capa de transporte:** esta capa se encarga de la invocación a métodos del *framework* WSF2CPP que permiten la comunicación mediante SOAP sobre HTTP.

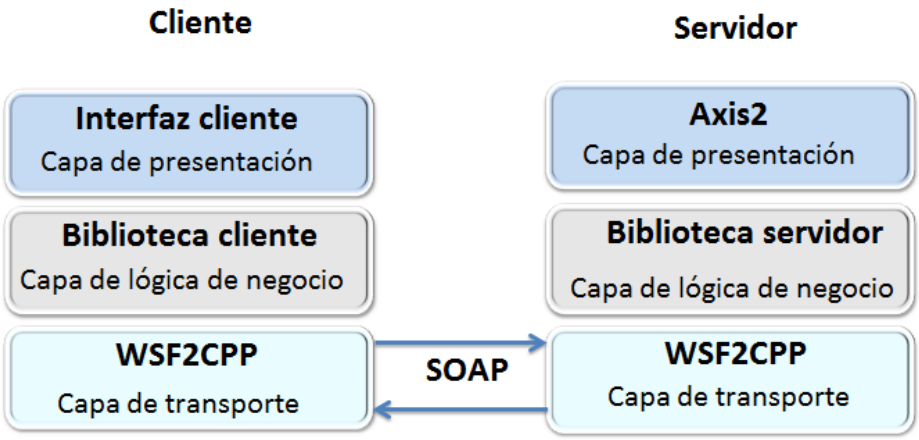

Figura 15. Arquitectura del subsistema de comunicación a través de OPC XML-DA.

2.8 Patrones de diseño.

Un patrón de diseño nombra, abstrae e identifica los aspectos clave de un diseño estructurado común, que lo hace útil para la creación de diseños orientados a objetos reutilizables. Definen una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones (25).

Los patrones de diseño tienen como propósito:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.

Para el diseño del subsistema de comunicación con terceros a través de OPC XML-DA se emplearon los patrones *Singleton*, *Facade y Modelo-Vista*. A continuación se explica cómo fueron utilizados cada uno ellos.

2.7.1 *Singleton* (Instancia única).

Se encarga de restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. En la aplicación este patrón se evidencia al garantizar que solo se pueda crear una instancia única de la clase OPCLayer, así como crear un punto de acceso global a esta para las demás clases del sistema.

2.7.2 *Facade* (Fachada).

Consiste en proporcionar una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar. La fachada del subsistema de comunicación está vista por la clase OPCLayer que permite a las clases pertenecientes a la interfaz de usuario acceder a las funcionalidades del sistema.

2.7.3 Modelo-Vista.

El patrón de diseño Modelo-Vista es una variante del patrón Modelo-Vista-Controlador que se utiliza para el diseño de aplicaciones que presentan interfaces sofisticadas.

El modelo es el núcleo de la aplicación. Su única responsabilidad respecto a la vista es avisar de cambios importantes mediante eventos. Además debe llevar un registro de las vistas del sistema, para que sea capaz de notificar a estos sobre los cambios que pueda producir un agente externo a los datos.

La vista es la encargada de mostrar al usuario el progreso de funcionamiento del modelo. Cuando el modelo cambia la vista solicita el nuevo estado y se actualiza. Presenta el modelo en un formato que pueda interactuar con el usuario, por lo general es la interfaz del usuario. Este patrón se utiliza en la interfaz gráfica del cliente OPC XML-DA.

2.9 Conclusiones.

El desarrollo de este capítulo permitió diseñar el subsistema que se desea construir. Se especificaron los requisitos funcionales y no funcionales así como los actores y casos de uso. Fue definida la arquitectura y los patrones de diseño a utilizar y se modelaron los diagramas de clases y los diagramas de secuencia para una mejor comprensión del funcionamiento del subsistema.

CAPÍTULO 3
IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción.

En el presente capítulo se muestran las vistas de implementación y despliegue con sus respectivos diagramas de componentes. Además se expone el estilo de código utilizado y las pruebas que validan el funcionamiento del subsistema de comunicación.

3.2 Implementación.

La implementación comienza con los resultados del diseño e implementa el sistema en términos de componentes como ficheros de código fuente, *scripts*, ficheros de código binario, ejecutables y similares. Teniendo como propósito principal el desarrollo de la arquitectura como un todo.

3.2.1 Diagrama de componentes.

A continuación se representan, de acuerdo a los paquetes identificados en el diseño, las interacciones existentes entre los principales componentes que integran la solución, así como sus dependencias más significativas.

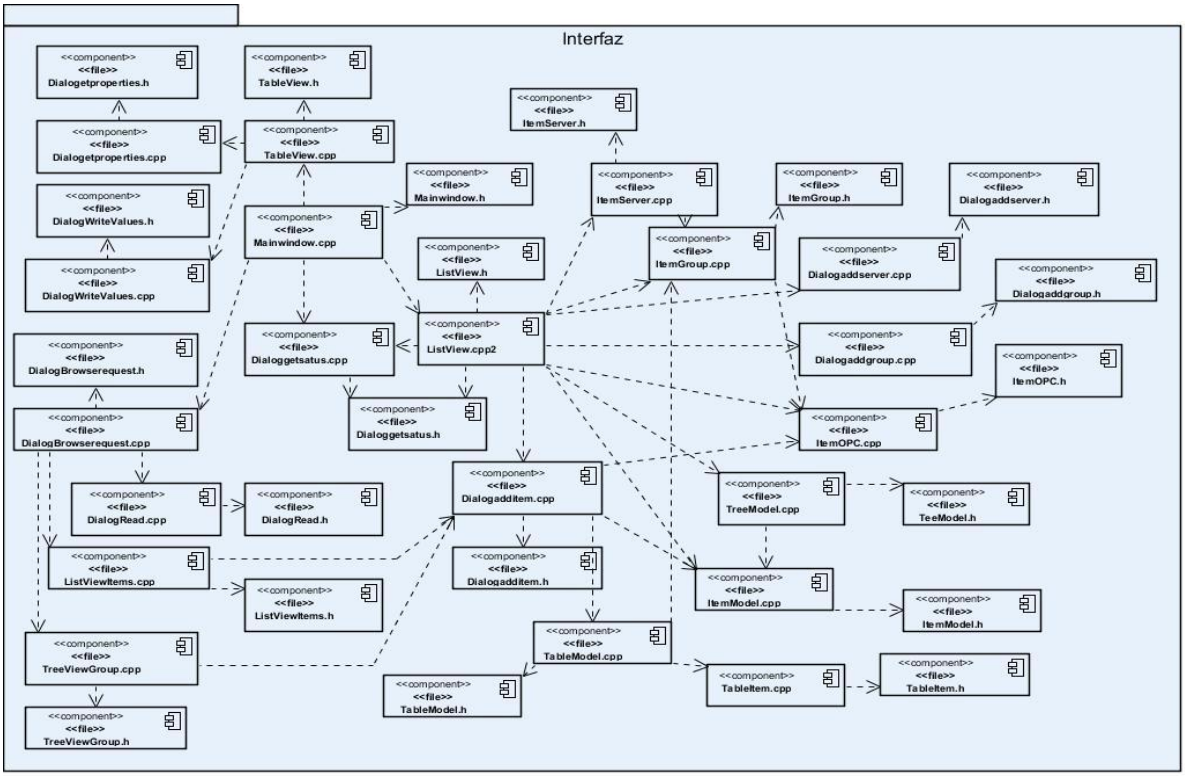

Figura 16. Diagrama de componentes del paquete Interfaz.

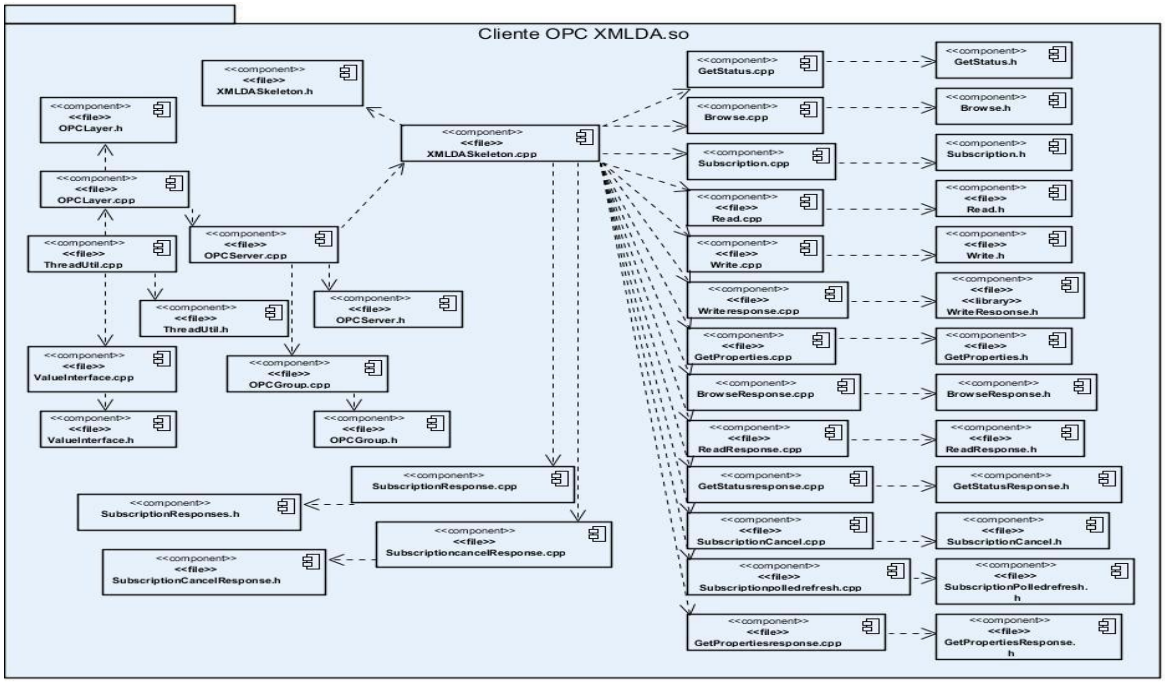

Figura 17. Diagrama de componentes del paquete Cliente OPC XMLDA.

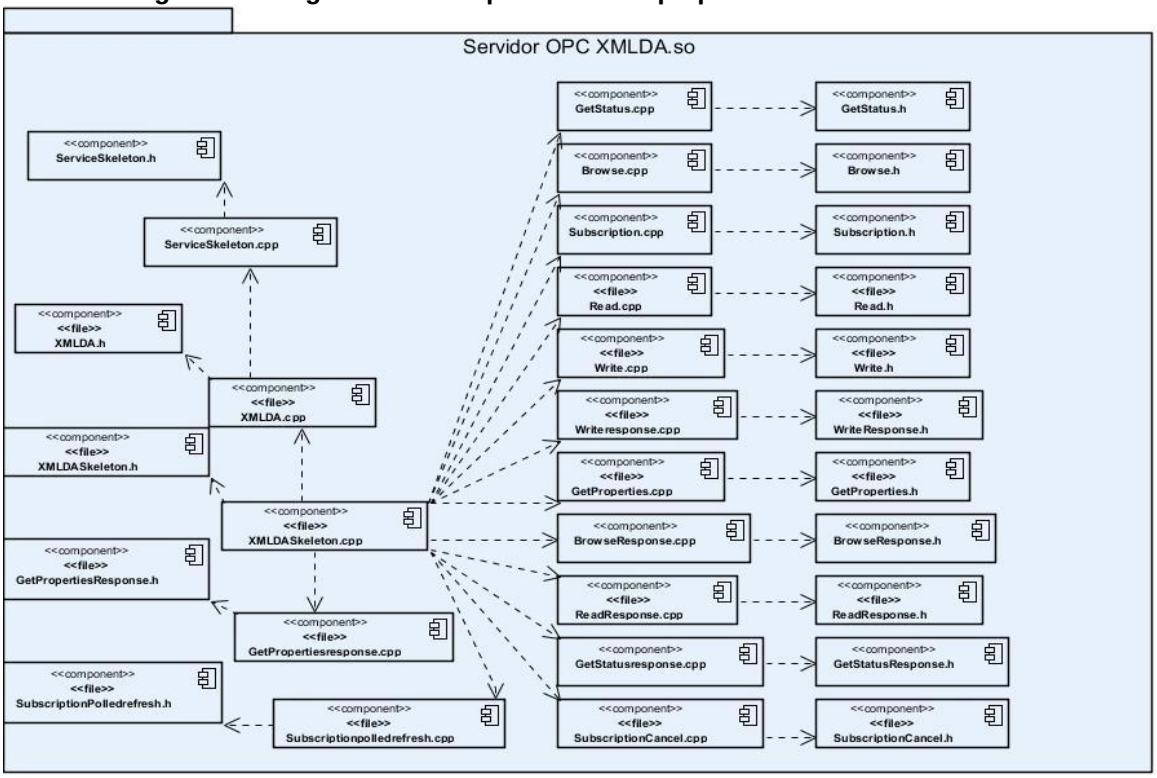

Figura 18. Diagrama de componentes del paquete Servidor OPC XMLDA.

3.2.2 Diagrama de despliegue.

El diagrama de despliegue muestra la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Los nodos no son más que elementos físicos que existen en tiempo de ejecución y representan un recurso computacional que generalmente tienen algo de memoria y capacidad de procesamiento. El subsistema de comunicación desarrollado tiene la siguiente disposición física:

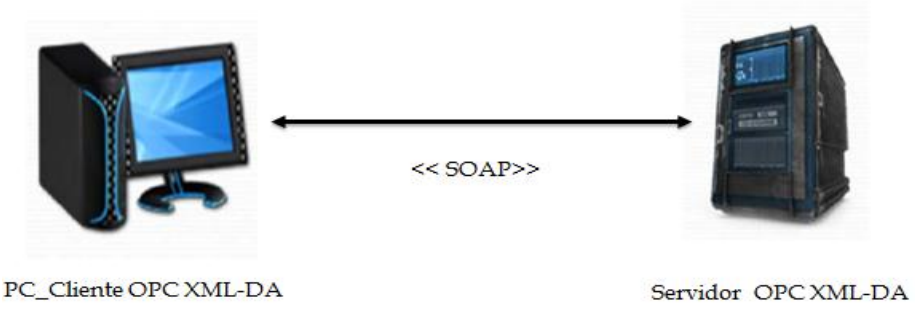

Figura 19. Despliegue del subsistema de comunicación a través de OPC XML-DA.

- **PC_Cliente OPC XML-DA:** representa el Cliente OPC XML-DA desarrollado e instalado en una computadora que se utilizará para interactuar con el Servidor OPC XML-DA.
- **Servidor OPC XML-DA:** representa la estación donde estará instalado el Servidor OPC XML-DA, éste se encarga de almacenar los datos que recibe de otros dispositivos y sistemas como el SCADA GALBA.
- **<< SOAP>>:** representa el protocolo que se utiliza para la comunicación entre el cliente y el servidor.

3.3 Estilo de código.

A la hora de programar es necesario seguir un estilo. Este permitirá revisar, mantener y actualizar el código de una manera más sencilla y ordenada. A continuación se describe el estilo de codificación y documentación que fueron utilizados en el desarrollo del Subsistema para la comunicación del SCADA GALBA con terceros a través del estándar OPC XML-DA. Este estilo de código se definió para ser utilizado en todos los subsistemas del SCADA Guardián del ALBA (Estilo de Código SCADA, 2007).

3.3.1 Nombres.

- Los nombres de las clases son sustantivos singulares.
- Los nombres no deben revelar detalles de implementación.
- Escoger nombres lo suficientemente largos para ser expresivos, pero evitando manejar

nombres que dificulten la labor de implantación.

- Evitar nombres que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Evitar redundancia no repitiendo nombres de clases en sus elementos.
- Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear mayúscula para el inicio de cada palabra y minúscula para el resto de las letras para el caso de los nombres de métodos y funciones. Para el caso de los nombres de variables y atributos debe aplicarse la misma convención, con excepción de la primera letra del nombre, la cual debe ser en minúscula.
- Los nombres de constantes deben contener solo letras mayúsculas.
- Los nombres de los métodos son frases que incluyen verbos.
- Los nombres de los atributos y parámetros son frases con sustantivos.
- Evitar el uso de nombres idénticos para distinto propósito.

3.3.2. Manejo de Errores.

- Se pueden manejar los errores mediante mecanismos de excepciones o mediante valores de retorno, aunque esto debe ser uniforme dentro de un mismo objeto.

- Es buena práctica emplear herramientas para identificar errores en la codificación en caliente.

3.3.3. Documentación y comentarios.

- En el código debe documentarse en forma explicativa los pasos que se van ejecutando.
- Emplear oraciones completas al documentar el código.
- Documentar mientras se programa.
- Documentar cualquiera cosa que no sea obvia en el código.
- Documentar eliminación de errores y cambios sobre el código.
- Al modificar el código se deben actualizar todos los comentarios y documentación asociada.
- Documentar cada rutina agregando: nombre del desarrollador, fecha, parámetros de entrada, valores de retorno, precondiciones, pos-condiciones, dependencia con otros métodos o funciones y descripción general del algoritmo. Además, de realizarse cambios al código, debe indicarse el nombre de la persona que realizó el cambio, la fecha y la descripción del cambio, comenzando desde el o los cambios más recientes.

- Evitar agregar comentarios al final de líneas de código, salvo en el caso de declaraciones. En este caso tales comentarios deben estar alineados.
- Antes de la entrega de la aplicación, eliminar todos los comentarios superfluos y/o temporales con la finalidad de evitar confusiones en su mantenimiento**.**

3.3.4. Codificación.

- Alinear verticalmente llaves de apertura y cierre.
- Usar espacios antes y después de los operadores que el lenguaje de programación permita.
- Emplear líneas en blanco para organizar el código, permitiendo crear párrafos de código para una mejor lectura.
- Evitar colocar más de una sentencia por línea.
- Liberar apuntadores de manera explícita.
- Emplear i, j, k, l, p, q, r para contadores en ciclos.
- Comentar siempre las llaves que cierran.
- Emplear al máximo operadores del tipo: +=, *=, /=, -=, ++, --, entre otros.
- Emplear correctamente los tipos de ciclos: si es al menos una vez usar do-while, si es ninguna o más veces usar while-do, y si se conoce el número exacto de ciclos usar for.

3.4 Pruebas.

Un factor de vital importancia para la producción de un software es lograr la calidad del producto, y precisamente las pruebas que se le realicen constituyen un papel protagónico pues con la realización de un conjunto finito de casos de pruebas es posible verificar el comportamiento del producto durante su ciclo de vida. Este proceso además de garantizar la calidad del software, tiene como objetivo comprobar la satisfacción de los requisitos y localizar para subsanar, el mayor número de deficiencias antes de liberar el producto final. El tamaño y complejidad del proceso de pruebas depende del tamaño y complejidad del producto que se está desarrollando y para ejecutarlo es necesario definir elementos tales como el nivel y método de pruebas a utilizar. Para el subsistema desarrollado se decidió realizar las siguientes pruebas definidas a continuación:

3.4.1 Nivel de Prueba: Sistema.

A este nivel es verificado que cada elemento interactúe de forma adecuada, que sea alcanzado la funcionalidad y el rendimiento del sistema total, sean validados los requisitos establecidos comparándolos con el sistema construido.

3.4.2 Tipo de Prueba: Funcionalidad.

Este tipo de prueba asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados. Verificar la apropiada aceptación de datos y se encuentra enfocada principalmente a los requisitos funcionales o casos de uso (26).

Método de Prueba: Caja Negra. Es denominada prueba de comportamiento, se centra en los requisitos funcionales del software, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa y está referida a las pruebas que se llevan a cabo sobre la interfaz del software examinando algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software (26).

3.4.3 Ambiente de prueba.

Para la correcta realización de las pruebas, se contó con una serie de recursos que facilitaron el trabajo de ejecutar cada caso de prueba sobre el subsistema de comunicación.

- **Recursos físicos:** Se utilizaron 2 máquinas con 1 *gigabyte* de memoria RAM, 160 *gigabyte* de capacidad de disco duro, microprocesador Intel Core 2 Duo con una velocidad de 2.2Ghz, *motherboard* Intel y red cableada.
- **Recursos lógicos:** El sistema operativo sobre el cual se desarrollaron las pruebas es Debian squeeze 6.0.5. La velocidad de la red es de 100 *megabits* por segundo. Debe estar instalado en cada computadora el *framework* WSO2 WSF2CPP con sus librerías. En la computadora donde estará el servidor se necesita además un servidor web para publicar el servicio, en este caso es el Apache, y en el caso de la computadora donde estará el cliente se necesita la biblioteca boost y las bibliotecas del *framework* Qt.

3.4.4 Diseño de casos de prueba.

A continuación se detallan los casos de prueba (CP) elaborados para los requisitos funcionales más importantes con los que debe cumplir el subsistema. Esto se realiza para comprobar el correcto funcionamiento del sistema bajo las diferentes condiciones a las que puede someterse.

3.4.4.1 CU Operar variable.

Las pruebas realizadas a este caso de uso son las siguientes:

- Adicionar una o varias variables.
- Leer una o varias variables.

Caso de Prueba (CP1) Adicionar variable.

1. Breve Descripción

Este Caso de Prueba permite comprobar la funcionalidad de adicionar una o varias variables a un grupo.

2. Flujo Central

2.1 Clic derecho sobre el grupo al que desea adicionar la variable y selecciona la opción “Adicionar variable”.

2.2 El sistema muestra una ventana con todas las variables existentes.

2.3 Clic derecho sobre la o las variables que desea adicionar y selecciona la opción “Adicionar variable”.

2.4 El sistema muestra una vista con la variable seleccionada.

2.5 El usuario da clic en el botón Aceptar.

2.6 El sistema adiciona la o las variables al grupo.

3. Condiciones de Ejecución

3.1 Debe existir un grupo.

3.2 Tienen que existir variables en el servidor.

4. Iteraciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado obtenido
Seleccionar la variable que se desea adicionar y presionar el botón “Aceptar”.		El sistema debe adicionar la variable al grupo y actualiza la lista de variables asociadas al mismo.	El sistema adiciona la variable al grupo y actualiza la lista de variables asociadas al mismo.
Seleccionar las variables que se desean adicionar y presionar el botón “Aceptar”.		El sistema debe adicionar las variables al grupo y actualiza la lista de variables asociadas al mismo.	El sistema adiciona las variables al grupo y actualiza la lista de variables asociadas al mismo.

	Pulsar “Aceptar” sin haber seleccionado una variable.	El sistema debe mostrar un mensaje indicando que no ha sido seleccionada ninguna variable.	El sistema muestra un mensaje indicando que no ha sido seleccionada ninguna variable.

Tabla 3. Caso de Prueba (CP1) Adicionar variable.

Caso de Prueba (CP2) Leer variable.

1. Breve Descripción

Este Caso de Prueba permite comprobar la funcionalidad de leer una o varias variables del sistema.

2. Flujo Central

2.1 Clic derecho sobre la o las variables que desea leer y selecciona la opción “Leer variable”.

2.2 El sistema muestra una ventana con las propiedades de la o las variables seleccionadas.

3. Condiciones de ejecución.

3.1 Tienen que existir variables en el servidor.

4. Iteraciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado obtenido
Selecciona la opción “Leer variable”.		El sistema debe mostrar las propiedades de la variable: • Nombre • Valor • Estampa de tiempo • Calidad • Camino	El sistema muestra las propiedades de la variable: • Nombre • Valor • Estampa de tiempo • Calidad • Camino

Selecciona la opción “Leer todas”.		El sistema debe mostrar las propiedades de las variables seleccionadas. • Nombre • Valor • Estampa de tiempo • Calidad • Camino	El sistema muestra las propiedades de las variables seleccionadas. • Nombre • Valor • Estampa de tiempo • Calidad • Camino

Tabla 4. Caso de Prueba (CP2) Leer variable.

3.4.4.2 CU Administrar variable.

La prueba realizada a este caso de uso es la siguiente:

- Modificar el valor de una variable.
- Consultar propiedades de una variable.

Caso de Prueba (CP3) Modificar variable.

1. Breve Descripción

Este Caso de Prueba permite comprobar la funcionalidad de modificar el valor de una variable del sistema.

2. Flujo Central

2.1 Clic derecho sobre la variable que se desea modificar y seleccionar la opción “Modificar variable”.

2.2 El sistema muestra una ventana con el valor actual de la variable y el campo requerido para insertar el nuevo valor.

2.3 Insertar el nuevo valor de la variable y presionar en el botón “Aceptar”.

3. Condiciones de ejecución.

3.1 El tipo de acceso de la variable seleccionada tiene que ser de lectura/escritura.

4. Iteraciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado obtenido
Selecciona la opción “Modificar variable”.		El sistema debe mostrar una interfaz con el valor	El sistema muestra la interfaz con el valor

		actual de la variable y un campo para insertar el nuevo valor.	actual de la variable y un campo para insertar el nuevo valor.
Se inserta el nuevo valor de la variable y se presiona el botón “Aceptar”.		El sistema debe actualizar el valor de la variable.	El sistema actualiza el valor de la variable.
	Pulsar “Aceptar” sin haber introducido un valor.	El sistema debe mostrar un mensaje indicando que debe introducir un valor.	El sistema muestra un mensaje indicando que debe introducir un valor.
	El valor introducido no se corresponde con el tipo de datos que almacena la variable.	El sistema debe mostrar un mensaje indicando que no se puede escribir.	El sistema muestra un mensaje indicando que no se puede escribir.

Tabla 5. Caso de Prueba (CP3) Modificar variable.

Caso de Prueba (CP4) Consultar propiedades de una variable.

1. Breve Descripción

Este Caso de Prueba permite comprobar la funcionalidad de consultar las propiedades de una variable.

2. Flujo Central

2.1 Clic derecho sobre la variable que se desea conocer las propiedades y seleccionar la opción “Propiedades”.

2.2 El sistema muestra una ventana con todas las propiedades de la variable seleccionada.

3. Condiciones de ejecución.

Debe existir al menos una variable en la tabla de variables.

4. Iteraciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado obtenido
Seleccionar la opción “Propiedades”.		El sistema debe mostrar una ventana con todas las propiedades de la variable seleccionada: • Nombre • Valor • Estampa de tiempo • Calidad • Camino	El sistema muestra una ventana con todas las propiedades de la variable seleccionada: • Nombre • Valor • Estampa de tiempo • Calidad • Camino

Tabla 6. Caso de Prueba (CP4) Consultar propiedades de una variable.

3.4.4.3 CU Consultar estado de servidor.

La prueba realizada a este caso de uso es la siguiente:

- Consultar el estado del servidor.

Caso de Prueba (CP5) Consultar estado de servidor.

1. Breve Descripción.

Este Caso de Prueba permite comprobar la funcionalidad de consultar el estado de un servidor.

2. Flujo Central.

2.1 Clic derecho sobre el servidor del que se desea conocer su estado.

2.2 El sistema muestra una ventana con las propiedades del servidor seleccionado.

3. Condiciones de ejecución.

3.1 Debe haberse establecido conexión con ese servidor.

4. Iteraciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado obtenido
Clic derecho sobre el servidor y seleccionar la opción		El sistema debe mostrar una interfaz con las propiedades del	El sistema muestra la interfaz con las propiedades del servidor

"Estado del servidor".		servidor seleccionado. • Estado • Tiempo • Versión • Información del proveedor • Versión interfaz	seleccionado. • Estado • Tiempo • Versión • Información del proveedor • Versión interfaz
	Seleccionar un servidor con el que no exista conexión.	El sistema debe mostrar un mensaje indicando que el servidor no está conectado.	El sistema muestra un mensaje indicando que el servidor no está conectado.

Tabla 7. Caso de Prueba (CP5) Consultar estado de servidor.

3.4.4.4 CU Administrar grupo.

Las pruebas realizadas a este caso de uso son las siguientes:

- Activar grupo.
- Eliminar grupo.

Caso de Prueba (CP6) Activar grupo.

1. Breve Descripción.

Este Caso de Prueba permite comprobar la funcionalidad de activar un grupo.

2. Flujo Central.

2.1 Clic derecho sobre el grupo que se desea activar.

2.2 El sistema muestra una tabla con las propiedades de las variables asociadas a ese grupo y los cambios que se van realizando.

3. Condiciones de ejecución.

3.1 Debe de existir un grupo con conjunto de variables asociadas.

4. Iteraciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado obtenido
Seleccionar el grupo y presionar "Activar grupo".		El sistema debe mostrar una tabla con las propiedades de las	El sistema muestra una tabla las propiedades de las variables y actualiza

		variables e ir actualizando periódicamente los valores: • Estampa de tiempo • Calidad • Valor	periódicamente los valores : • Estampa de tiempo • Calidad • Valor
	Seleccionar un grupo que no tenga variables asociadas.	El sistema debe mostrar un mensaje indicando que el grupo no contiene variables.	El sistema muestra un mensaje indicando que el grupo no contiene variables.

Tabla 8. Caso de Prueba (CP6) Activar grupo.

3.4.4.5 CU Explorar servidor.

La prueba realizada a este caso de uso es la siguiente:

- Explorar servidor.

Caso de Prueba (CP8) Explorar servidor.

1. Breve Descripción

Este Caso de Prueba permite comprobar la funcionalidad de explorar servidor de acuerdo a la opción seleccionada que puede ser “Explorar todo” o “Explorar por grupo”.

2. Flujo Central

2.1 El actor selecciona el servidor que desea explorar y escoge en el menú “Servidor” la opción “Explorar servidor”.

2.2 El sistema muestra una ventana para seleccionar la forma de explorar el servidor.

2.3 Selecciona una de las formas y presiona el botón “Aceptar”.

3. Condiciones de ejecución.

3.1 Debe existir al menos un servidor.

4. Iteraciones

Clases válidas	Clases inválidas	Resultado esperado	Resultado obtenido
Se selecciona la forma de explorar servidor “Explorar todo” y se presiona el botón “Aceptar”.		El sistema debe mostrar todas las variables del servidor.	El sistema muestra todas las variables que contiene el servidor.
Se selecciona la forma de explorar servidor “Explorar por grupo” y se presiona el botón “Aceptar”.		El sistema debe mostrar todos los grupos de variables del servidor.	El sistema muestra todos los grupos de variables del servidor.
	Pulsa “Explorar” sin haber seleccionado ninguna de las opciones.	El sistema debe mostrar un mensaje indicando que debe seleccionar una de las opciones.	El sistema muestra un mensaje indicando que debe seleccionar una de las opciones.

Tabla 9. Caso de Prueba (CP8) Explorar servidor.

3.4.5 Análisis de los resultados de las pruebas.

Para el correcto desarrollo del proceso de pruebas se definieron un total de 8 Casos de Pruebas, realizándose un total de 3 iteraciones, detectándose un total de 15 no conformidades. Estas no conformidades detectadas tenían una repercusión negativa sobre el funcionamiento del sistema y con la erradicación de los mismos se logró mejorar y optimizar gradualmente el funcionamiento de la aplicación.

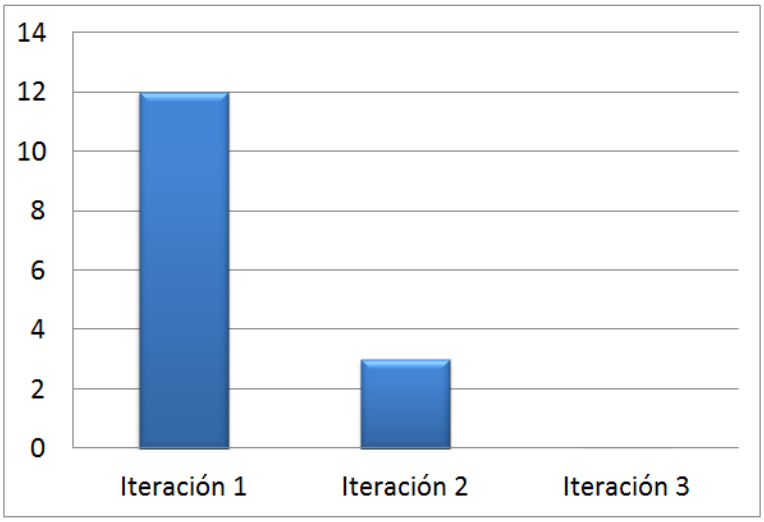

Figura 20. Cantidad de no conformidades por iteración.

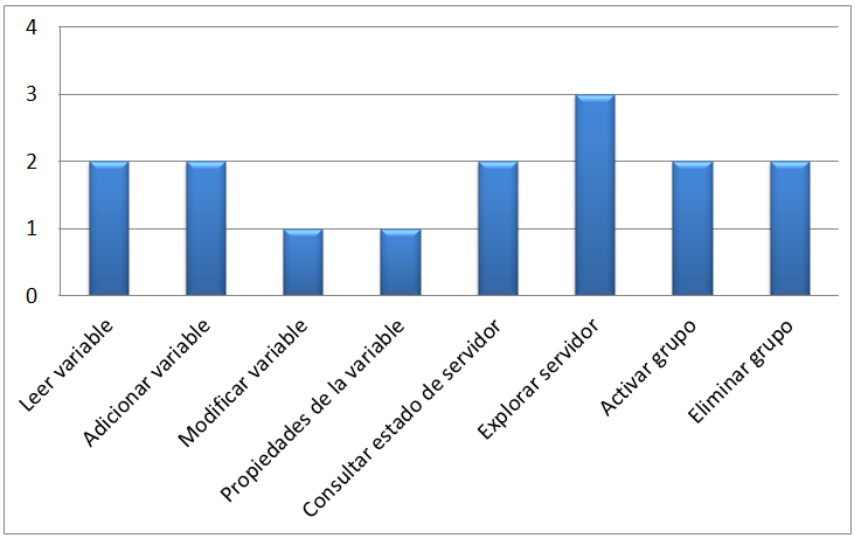

Figura 21. Cantidad de no conformidades por Caso de Prueba.

3.5 Conclusiones.

En este capítulo se desarrollaron los diagramas de componentes y de despliegue propios de la solución. Conjuntamente se realizaron las pruebas unitarias las cuales arrojaron resultados de importancia considerable para el funcionamiento del subsistema, ayudando a la corrección de los errores identificados y a implementar un producto con mayor calidad.

CONCLUSIONES GENERALES

Al término de la investigación para el desarrollo del Subsistema para la comunicación del SCADA GALBA con terceros a través de la especificación OPC XML-DA se concluye que:

- Se desarrolló un subsistema para el módulo de Integración con terceros del SCADA Guardián del ALBA que permite la publicación y el acceso a datos en formato OPC XML-DA.
- Se realizó el diseño e implementación de un cliente OPC XML-DA.
- Se realizó la implementación de todas las funcionalidades del servidor OPC XML-DA.
- El subsistema desarrollado permite una total interoperabilidad debido a que utiliza los Servicios Web, tecnología basada en la transmisión de texto, independiente del lenguaje de programación y del sistema operativo en el que se desarrolle el tercero.

RECOMENDACIONES

Se recomienda:

- Integrar el Subsistema para la comunicación con terceros a través de la especificación OPC XML-DA al SCADA Guardián del ALBA.
- Configurar y autenticar el servidor OPC XLM-DA con el SCADA Guardián del ALBA.
- Ampliar las funcionalidades del subsistema de manera que permita la comunicación de forma asíncrona.

REFERENCIAS BIBLIOGRÁFICAS

1. **Antunez Ojeda, Yaima y Ravelo Hernández, Luis Ángel.** MÓDULO DE COMUNICACIÓN PARA SISTEMAS SCADA USANDO TENOLOGÍAS LIBRES. [En línea] 2010. [Citado el: 5 de Noviembre de 2012.] http://tallertematico2010.fordes.co.cu/public/site/143.pdf.

2. **Aragón Cáceres, José Antonio.** SOLUCION INTEGRAL DE COMUNICACIÓN DE SISTEMAS DE SUPERVISIÓN Y CONTROL CON SISTEMAS EXTERNOS. [En línea] 2011. [Citado el: 15 de Noviembre de 2012.] http://www.informaticahabana.cu/node/886.

3. **Fernández Melgarejo, Yailyn.** Servicios de Integración con Terceros para el intercambio de Alarmas y Eventos que ocurran en el proceso y el sistema SCADA Guardián del ALBA. La Habana : s.n., 2009.

4. **Aragón Cáceres, José Antonio.** Servidor de Comunicación con sistemas externos del SCADA-UX. [En línea] 10 de Marzo de 2011. [Citado el: 8 de Diciembre de 2012.] publicaciones.uci.cu/index.php/SC/article/view/584.

5. **Aragón Cáceres, José Antonio y Llanes Jiménez, Beatriz.** Acceso a Variables del sistema SCADA Guardián del ALBA. Ciudad de La Habana : s.n., 2009.

6. OLE-DDC-OPC. OPC. [En línea] [Citado el: 15 de Marzo de 2013.] http://www.iuma.ulpgc.es/~avega/int_equipos/trab9899/ole_dde_opc/index.html#_Toc40907314 2.

7. The OPC Fundation. [En línea] [Citado el: 27 de Septiembre de 2012.] http://opcfoundation.org/Default.aspx/01_about/01_history.asp?MID=AboutOPC...

8. **Fundación, CTIC.** Arquitectura OPCWASUP. [En línea] 12 de Diciembre de 2007. [Citado el: 10 de Octubre de 2012.] http://forge.morfeo-project.org/wiki/images/2/2e/ArquitecturaOPCWASUP.pdf.

9. **OPC, Fundación.** OPC XML DA Specification. [En línea] 2003. [Citado el: 8 de Octubre de 2012.] http://www.opcfoundation.org/Login.aspx?PageState=100.

10. **Campos , Fernando Alejandro, Coral, Germán Mauricio y Rojas, Amaury Oscar.** SISTEMA DE CONTROL Y SUPERVISIÓN INDUSTRIAL MULTIPLATAFORMA.

11. **ICOINS, Corporations.** ICOINS. [En línea] [Citado el: 9 de Diciembre de 2012.] http://www.iconics.com/company/corporate_info.asp..

12. Northern Dynamic. [En línea] [Citado el: 24 de Noviembre de 2012.] http://www.nordyn.com.

13. Conferencia Introducción a la Arquitectura del Guardián del ALBA. s.l. : UCI, 2010.

14. **Bermeo Perez, Fabián .** Metodología RUP - desarrollo de software de calidad. [En línea] 8 de Diciembre de 2010. [Citado el: 7 de Diciembre de 2012.] http://fabianbermeop.blogspot.mx/2010/12/metodologia-rup-desarrollo-de-software.html.

15. **Stroustrup, Bjarne.** The desing and evolution pf c++. s.l. : Addison Wesley, 1994.

16. IBM. Más utilidades de Boost . [En línea] [Citado el: 2012 de Diciembre de 10.] http://www.ibm.com/developerworks/ssa/aix/library/au-boostutilities/.

17. Visual Paradigm for UML 10.1. [En línea] [Citado el: 10 de Enero de 2013.] http://www.visual-paradigm.com/product/vpuml/.

18. Clickmatica. [En línea] [Citado el: 12 de Enero de 2013.] http://www.clickmatica.com/tecnologias/.

19. WSO2. WSO2 Web Services Framework for C++. [En línea] [Citado el: 12 de Enero de 2013.] http://wso2.com/products/web-services-framework/cpp.

20. WSO2. WSO2 WSF/C++ Features. [En línea] [Citado el: 13 de Enero de 2013.] http://wso2.org/project/wsf/cpp/2.0.0/docs.

21. QtProjec. Qt. [En línea] [Citado el: 15 de Enero de 2013.]

22. **Noguera, Bulmaro .** ¿Que es Apache? [En línea] 9 de Febrero de 2011. [Citado el: 4 de Diciembre de 2012.] http://culturacion.com/2011/02/%C2%BFque-es-apache/.

23. **Ramírez Despaine, Maikel, Milán Nuñez , Eduardo y Moreno Vega, Valery.** Herramienta para programar un controlador lógico programable basado en hardware reconfigurable. [En línea] Junio de 2011. [Citado el: 2012 de Enero de 26.] http://rielac.cujae.edu.cu/index.php/rieac/article/download/83/pdf_77..

24. **Tentor, Julio.** Arquitectura de N-Capas y N-Niveles. [En línea] [Citado el: 15 de Abril de 2013.] http://www.jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx.

25. **Larman , Craig.** UML y Patrones. Introducción al análisis y diseño orientado a objeto. s.l. : Preason.

26. **Juristo , Natalia, Moreno, Ana Maria y Vegas, Sira.** Técnicas de Evaluación de Software. 2006.

BIBLIOGRAFÍA CONSULTADA

1. **Perez Javier, Maikel y Aragon Caceres, Jose Antonio.** Documento de entrega formal del prototipo de Subsistema de Comunicación con Terceros. Habana : s.n., 2009. Proyecto SCADA Guardian del ALBA,2009.

2. **Herrera Vázquez, M.Sc Moisés y Pérez Javier, Ing. Maikel.** Especificación de Comunicación con Terceros. Mérida,Venezuela : s.n., 2008. Vol. I, Proyecto SCADA Guardián del ALBA, 2008.

3. **Rumbaugh, James, Jacobson, Ivar y Booch, Grady.** El Proceso Unificado de Desarrollo de Software. La Habana : Editorial Felix Varela, 2004.

4. **Mendiburo Diaz, Henry.** Sistemas SCADA. [En línea] [Citado el: 17 de Enero de 2013.] http://hamd.galeon.com.

5. **Diaz Toledo, Moises Daniel.** Web Services.Introducción y Escenarios para su Uso. [En línea] [Citado el: 5 de Febrero de 2013.] http://www.moisesdaniel.com/articles.html.

6. **Martínez, Marlon.** Scribd. OLE for Process Control. [En línea] [Citado el: 8 de Diciembre de 2012.] http://es.scribd.com/doc/36562704/QUE-ES-OPC.

7. Advosol. XML-DA .NET Server Toolkit. [En línea] [Citado el: 18 de Enero de 2013.] http://www.advosol.com/pc-12-9-xml-da-rapid-server-toolkit.aspx.

8. OPC: ¿ De que se trata y como funciona ?.Guía para entender la tecnologia OPC. [En línea] 2009. [Citado el: 23 de Febrero de 2013.] http:\\www.infoPLC.net.

9. OPC programmer´s conection. [En línea] [Citado el: 15 de Febrero de 2013.] http://www.opcconnect.com/xml.php#xmlspec.

10.**MatrikonOPC**. OPC Data Access (OPC DA) Versions & Compatibility. matrikonopc. [En línea] 2009. http://www.matrikonopc.com/opc-server/opc-data-access-versions.aspx.

11. Source making. Patrón de Diseño. [En línea] [Citado el: 14 de Marzo de 2013.] http://sourcemaking.com/patr%C3%B3n-de-dise%C3%B1o.

12. **Iglesias Machado, Annaliet .** Desarrollo de un Explorador Cliente Object Linking and Embedding for Control Process of Data Access (OPC DA). Ciudad de la Habana : UCI, 2011.

13. National Instruments. HMI/SCADA, LabView. [En línea] [Citado el: 15 de Octubre de 2012.] http://www.ni.com/labview/.

ANEXOS

Anexo 1: Certificado de validación de la solución.

GUARDIÁN del ALBA

Mérida, a los 24 días del mes mayo del 2013

A quien pueda interesar:

Por este medio hacemos de su conocimiento que las herramientas generadas como producto del desarrollo del trabajo de diploma titulado: "Subsistema para la comunicación del SCADA Guardián del ALBA con terceros a través de la especificación OPC XML-DA", implementa correctamente las siguientes funcionalidades:

- Gestionar grupos (crear, eliminar, activar y desactivar grupos).
- Gestionar servidores OPC XML-DA.(Eliminar,conectar y desconectar servidores OPC XML-DA).
- Consulta del estado de los servidores OPC XML-DA.
- Adicionar, leer, modificar, eliminar y consultar las propiedades variables.
- Explorar los servidores OPC XML-DA.
- Escritura de variables en el servidor.

Garantizando con este desarrollo la comunicación con gran variedad de dispositivos y sistemas que existen actualmente en la industria petrolera venezolana.

Ing. José Antonio Aragón Cáceres
Arquitecto del sistema

Ing. Evián Suarez Rodriguez
Líder de la línea de comunicaciones

Figura 22. Certificado de validación de la solución.

GLOSARIO DE TÉRMINOS

- **ActiveX:** Entorno de aplicaciones desarrollado por Microsoft. Incluye un gran número de componentes que inter-operan y enlazan diferentes aplicaciones en una computadora o red de computadoras.
- **ALBA**: Alianza Bolivariana para las Américas.

- **Cliente/Servidor:** Esta arquitectura en un cliente que realiza peticiones a otro programa -el servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora, es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.
- **COM**: *Component Object Model*, es una plataforma de Microsoft para componentes de software desarrollada en 1993, es utilizada para permitir la comunicación entre procesos y la creación dinámica de objetos, en cualquier lenguaje de programación que soporte dicha tecnología.
- **CORBA:** *Common Object Request Broker Architecture*, es un estándar que establece una plataforma de desarrollo de sistemas distribuidos, facilitando la invocación de métodos remotos bajo el paradigma orientado a objetos.
- **DCOM**: *Distributed Common Objects Model* o Modelo de Objetos de Componentes Distribuidos, es una tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí. Extiende el modelo COM de Microsoft y proporciona el sustrato de comunicación entre la infraestructura del servidor de aplicaciones COM+ de Microsoft. Ha sido abandonada en favor del *framework* .NET.
- **Firewall:** es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas.
- **Gateway OPC DA:** Es una pasarela que tiene como función interactuar con los servidores OPC para dar respuesta a las peticiones provenientes de los clientes del SCADA GALBA.
- **HTML**: *Hypertext Markup Language* o Lenguaje de Marcado de Hipertexto, es un lenguaje para crear documentos de hipertexto para uso en el www o intranet, por ejemplo. Los archivos de HTML son usualmente visualizados por navegadores como *Internet Explorer*, *Firefox*, entre otros. Es independiente del sistema operativo del ordenador.

- **HTTP**: *Hypertext Transfer Protocol* o protocolo de transferencia de hipertexto, es un protocolo con la ligereza y velocidad necesaria para distribuir y manejar sistemas de información hipermedia.
- **.NET:** Es un proyecto de Microsoft para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.
- **Modbus:** es un protocolo de comunicaciones situado en el nivel 2 del Modelo OSI, basado en la arquitectura maestro/esclavo o cliente/servidor.
- **PLC**: *Programmable Logic Controller* o Controladores Lógicos Programables, es un hardware industrial, que se utiliza para la obtención de datos. Una vez obtenidos, los pasa a través de bus a un servidor.
- **SGML**: *Standard Generalized Markup* Language o Lenguaje de Marcado Generalizado, es un sistema para la organización y etiquetado de documentos. Este lenguaje sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial.
- **SOAP:** *Simple Object Access Protocol* es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.
- **OLE**: *Object Linking and Embedding,* es un sistema de objeto distribuido y un protocolo desarrollado por Microsoft.
- **XML**: *Extensible Markup Language*, es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium*. Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos.