Universidad de las Ciencias Informáticas
Facultad 5

Título: Servicio de Integración con Terceros para el Acceso a Variables del sistema SCADA Guardián del ALBA.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: José Antonio Aragón Cáceres

Beatriz Llanes Jiménez

Tutor: Ing. Maikel Pérez Javier

Ciudad de la Habana, Mayo 2009

"Año del 50 Aniversario del Triunfo de la Revolución¨

Sin embargo, cuando ya llevaban corriendo una media hora o así, y estaban completamente secos otra vez, el Dodo dijo de repente en voz alta: ((¡La carrera ha terminado!)), y se agruparon todos a su alrededor, jadeando y preguntado: ((Pero, ¿quién ha ganado?)). El Dodo no podía contestar a esta pregunta sin meditarlo mucho antes, y permaneció largo rato con un dedo apretado en la frente (en la postura que normalmente veis a Shakespeare en los retratos), mientras el resto esperaba en silencio.

LEWIS CARROLL, Alicia en el País de las Maravillas.

DATOS DE CONTACTO

Nombre y apellidos: Maikel Pérez Javier
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: Ingeniero en Informática.
e-mail: mperezj@uci.cu

Profesor Instructor con tres años de experiencia docente en la Universidad de las Ciencias Informáticas (UCI) y tres años de experiencia en la producción de software, específicamente en el desarrollo del *middleware* del Sistema Supervisor de Procesos Automatizados (SCADA).

AGRADECIMIENTOS

Agradezco a la Revolución por brindarme la oportunidad de realizar un sueño, a mi familia por apoyarme siempre, a mis amigos por estar a mi lado, inclusive en los momentos más difíciles, a aquellos que están hoy y a los que por algún motivo no pueden estar presentes, a todos mil gracias, porque junto a ustedes he madurado, me he superado y he aprendido todo lo que hoy conozco. Un besi grande a todos, mis lágrimas, hoy, son para ustedes, sonrían.

Betty

A mi madre por quererme tanto. A mi hermana por confiar en mí. A Yeri por soportarme. A todos los profesores que me dieron clases. Al proyecto SCADA por todo el conocimiento adquirido. A Ariel por sus soluciones. A mi grupo por ser tan especial. A Fidel, a la Revolución y a la UCI.

Toni

DEDICATORIA

A mi alma gemela, mi Mi. A mi tío Benito, a mi familia y a José Antonio.

Betty.

A mi madre, mi hermana, a Yeri.

Toni.

RESUMEN

El presente trabajo muestra el desarrollo de un servicio para el acceso a variables del sistema SCADA, Guardián del ALBA, proporcionándole una vía efectiva de comunicación con sistemas de diversos tipos desarrollados por terceros que pudieran necesitar de la información manejada por este, relacionada con variables.

De manera general se comenzó por la investigación acerca de sistemas SCADA y su integración con sistemas de gestión MES y ERP, otros SCADA, entre otros. Se adquirió una noción general acerca del estado del arte de las comunicaciones entre sistemas SCADA con sistemas externos, para poder llevar a cabo una investigación acertada de diversas tecnologías de comunicación distribuida que fueran utilizadas a nivel mundial y permitieran el desarrollo exitoso del Servicio de Acceso a Variables.

Se seleccionaron las tecnologías de comunicación distribuida: ICE y Servicios Web, la primera aportó eficiencia en la transmisión de los datos y la segunda una total interoperabilidad al servicio. Se definieron los requerimientos funcionales y no funcionales que poseería el servicio, seguido por el modelado de la solución utilizando el patrón de arquitectura en tres capas. Finalmente se implementó el diseño definido y se realizaron pruebas a las funcionalidades críticas del servicio, arrojando resultados satisfactorios que validaron las tecnologías utilizadas y el modelado propuesto.

PALABRAS CLAVE

Acceso a variables en sistemas SCADA, Comunicación con terceros, SCADA.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En los inicios de la Automatización, los sistemas que se utilizaban para el control de procesos eran tecnológicamente simples. Con el tiempo ha ido aumentando la complejidad de estos sistemas de manera exponencial. Para la industria, la productividad es un factor crítico, minutos de paro, se pueden traducir en miles o hasta en millones de pesos de pérdidas o de ventas no realizadas. Es por ello que en la actualidad han tenido gran popularidad los sistemas SCADA, los cuales tienen como función principal, el monitoreo y control total de los procesos industriales. Los sistemas de este tipo pueden aplicarse en diversas ramas de la industria, desde el control de tráfico, hasta el monitoreo, control y supervisión de plataformas petroleras. El concepto de SCADA se aplica entonces en todo proceso donde se exige un control de calidad, producción y optimización del servicio.

Petróleos de Venezuela Sociedad Anónima (PDVSA), es la corporación estatal de la República Bolivariana de Venezuela a cargo de la exploración, producción, transporte y comercialización de los hidrocarburos. Esta corporación empleaba para el control de sus instalaciones diversos sistemas SCADA suministrados por compañías privadas encargadas de brindar la seguridad y eficiencia operacional de los sistemas automatizados. Dichas compañías tuvieron un papel protagónico en el sabotaje petrolero acaecido durante diciembre del 2002 y enero del 2003 en la que se intervinieron y descontrolaron sistemas automatizados que garantizaban la distribución del crudo y sus derivados, y el bloqueo de diversos servicios tecnológicos esenciales. (PDVSA, 2009)

En la lucha por lograr la independencia, soberanía y autonomía tecnológica, el presidente de la República Bolivariana de Venezuela, Hugo Chávez Frías, emitió el Decreto N° 3.390, publicado en la Gaceta Oficial Nº 38.095 de fecha 28/ 12/ 2004 sobre el uso obligatorio del software libre en el país para todas las dependencias públicas de carácter oficial. De esta forma, el Ejecutivo nacional estableció que: Es prioridad del Estado incentivar y fomentar la producción de bienes y servicios para satisfacer las necesidades de la población, mediante el uso de estas herramientas desarrolladas con estándares abiertos para robustecer la industria nacional, aumentando y aprovechando sus capacidades y fortaleciendo nuestra soberanía. (Chacón, 2006)

En el contexto actual de las relaciones entre Cuba y Venezuela y debido a la necesidad en este hermano país después del mencionado sabotaje de independizarse tecnológicamente se comenzó el desarrollo conjunto de un sistema SCADA utilizando software libre que permitiría sustituir los ya existentes por uno de elaboración nacional, que eliminaría la excesiva dependencia de sistemas SCADA propietarios, los cuáles en muchos casos no cubren las expectativas de diseño del usuario final. Algunos de estos,

actualmente en operación, están obsoletos y sobredimensionados, otros sin soporte del fabricante y los que cuentan con el mismo, tienen que pagar altos costes por este servicio y por la asistencia técnica, además de la poca disponibilidad de personal especializado en las regiones para su manipulación.

La Universidad de Ciencias Informáticas (UCI) asumió el reto que ello representaba, de esta forma en la Facultad 5, se comenzó el desarrollo del sistema SCADA "Guardián del ALBA" y con el posterior surgimiento del Polo de Hardware y Automática, pasa a formar parte del mismo. Con la culminación de los primeros trabajos pactados y los avances generales alcanzados, el desarrollo del sistema SCADA Guardián del ALBA ha llegado al punto en que resulta conveniente agregar funcionalidades que permitan que el producto se aproxime al estado del arte de los sistemas SCADA que lideran el desarrollo de aplicaciones destinadas al control de procesos.

Una de las funcionalidades necesarias identificadas fue la de proveer al Guardián del ALBA de un mecanismo que le permitiera comunicarse e intercambiar datos con sistemas externos, por lo que se comienza el desarrollo del Subsistema Comunicación con Terceros cuyo objetivo fundamental sería ofrecer una solución mediante la cual los sistemas externos pudieran acceder a los datos manejados por el sistema SCADA Guardián del ALBA.

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:

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

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

El dato más común manejado por un sistema SCADA y de mayor interés para los sistemas externos, es la variable (también llamada punto) que constituye el fenómeno físico que deseamos medir. Dependiendo del proceso, la naturaleza de este fenómeno es muy diversa, puede ir desde viscosidad de una sustancia hasta la presión en una caldera.

En la actualidad por la importancia que tiene para los niveles gerenciales de PDVSA la gestión de la información relacionada con las variables del Guardián del ALBA, existe la necesidad de brindar mecanismos para la integración con aplicaciones especializadas en la gestión, almacenamiento y procesamiento de estas variables.

Para dar solución a la situación problémica existente se tiene que responder la presente interrogante del **Problema Científico**:

¿Cómo permitir el intercambio de información relacionada con variables entre el sistema SCADA Guardián del ALBA y sistemas externos?

Del problema científico anterior, se define que el **Objeto de Estudio** de este trabajo es la Comunicación de Sistemas Externos con sistemas SCADA y el **campo de acción**, es la Integración con Terceros para el Acceso a Variables del sistema SCADA Guardián del ALBA. De acuerdo con el problema científico planteado la **Idea a Defender** es la siguiente:

Si se implementa el Servicio de Integración con Terceros para el Acceso a Variables del sistema SCADA Guardián del ALBA, entonces se posibilitarán a aplicaciones externas la conexión con el sistema y se proveerán interfaces que brinden funcionalidades para acceder a las variables de este, siempre dependientes de un mecanismo de suscripción y de la lógica de seguridad requerida.

El **Objetivo General** que se desea alcanzar para darle cumplimiento a este trabajo es implementar un Servicio de Integración con Terceros que permita el acceso a la información relacionada con variables del sistema SCADA Guardián del ALBA. Definiéndose los siguientes **Objetivos Específicos** para lograr un mayor nivel de precisión en la implementación:

- Estudiar y analizar temas relacionados con sistemas SCADA, comunicación con sistemas externos y servicios de integración con terceros para adquirir conocimientos respecto a estos elementos.
- Analizar las tendencias y tecnologías actuales que permitan desarrollar el Servicio de Integración con Terceros para el intercambio de información relacionada con variables del sistema SCADA Guardián del ALBA.
- Evaluar y seleccionar la tecnología y componentes libres adecuados para desarrollar el Servicio de Integración con Terceros para el intercambio de información relacionada con variables del sistema SCADA Guardián del ALBA.
- Modelar a través del diseño las funcionalidades relacionadas con el Servicio de Integración con Terceros para el intercambio de información relacionada con variables del sistema SCADA

Guardián del ALBA.

- Implementar las funcionalidades relacionadas con el Servicio de Integración con Terceros para el intercambio de información relacionada con variables del sistema SCADA Guardián del ALBA.
- Probar las funcionalidades implementadas para el Servicio de Integración con Terceros para el intercambio de información relacionada con variables del sistema SCADA Guardián del ALBA.

Para el desarrollo de las tareas científicas se combinan diferentes **Métodos y Técnicas** en la búsqueda y procesamiento de la información, los fundamentales son:

A nivel teórico

- **Análisis-síntesis:** Utilizado para analizar las teorías y documentos generados por desarrolladores que usen herramientas para el tratamiento de aplicaciones distribuidas permitiendo la extracción de los elementos que se relacionen con la implementación del Servicio de Integración con Terceros para acceder a variables del sistema SCADA Guardián del ALBA.
- **Análisis histórico-lógico:** Utilizado para conocer, con mayor profundidad los antecedentes y las tendencias actuales referidas a la comunicación con sistemas externos y servicios de integración con terceros de los sistemas SCADA.
- **Modelación:** Utilizado para definir y representar gráficamente las funcionalidades que tendrá el Servicio de Acceso a Variables del sistema SCADA Guardián del ALBA dentro del módulo de Comunicación con Terceros usando el Lenguaje Unificado de Modelado (*UML*).

A nivel empírico

Experimentos: Empleado en la elaboración de prototipos funcionales, con el objetivo de comprobar la efectividad de la implementación de las funcionalidades del módulo de Comunicación con Terceros referentes al acceso a variables del sistema SCADA Guardián del ALBA.

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 describe el estado del arte del servicio de Integración con Terceros para el acceso a variables y los estándares o soluciones existentes que implementan o permiten implementar este servicio.

En el Capítulo 2: "Selección de Tecnologías y modelado de la solución", se selecciona la tecnología de comunicación distribuida para implementar el Servicio de Acceso a Variables así como todas las herramientas que se usarán para su desarrollo. También se modela la solución propuesta.

En el Capítulo 3: "Implementación y Prueba", se muestra las vistas de implementación e implantación del servicio, así como los casos de pruebas realizados a este.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se muestran los conceptos fundamentales relacionados con los sistemas SCADA: su descripción, funcionalidades y componentes, resaltando como es que se comunican estos con sistemas externos. Se aborda el tema de los terceros más comunes que necesitan interactuar con sistemas de tipo SCADA. Se trata además de los servicios que debe brindar un Módulo de comunicación con terceros en un SCADA, haciendo énfasis en el Servicio de Integración con Terceros para el acceso a variables. También se realiza un estudio de las tendencias y tecnologías actuales que se utilizan para la implementación de este tipo de servicio, lo que permitirá tener un criterio más acertado en la toma de decisiones para la selección de la tecnología y posterior implementación del Servicio de Acceso a Variables.

1.1. Descripción general de un SCADA.

Un sistema SCADA, acrónimo de *Supervisory Control and Data Acquisition* (en español, Control Supervisor y Adquisición de Datos) es una aplicación o conjunto de aplicaciones software especialmente diseñada para el control de la producción, que se comunica con los dispositivos de campo y controla los procesos de producción de forma automática desde la pantalla del ordenador, que puede ser configurada y modificada por el usuario. Provee de toda la información que se genera en el proceso productivo a diversos usuarios: operadores, supervisores de control de calidad, mantenimiento, y otros. Los sistemas SCADA en la actualidad tienen un incontable número de aplicaciones dentro de las que se pueden mencionar: el control de oleoductos, sistemas de transmisión de energía eléctrica, yacimientos de gas y petróleo, redes de distribución de gas natural, subterráneos, generación energética**,** sistemas de distribución de agua, entre otros.

Los sistemas SCADA, como sistemas de control, proporcionan una nueva característica de automatización que realmente pocos sistemas ofrecen, y es lo que los identifica: la **supervisión**. Esta función es llevada a cabo por los supervisores u operadores del sistema.

La labor del supervisor representa una tarea delicada y esencial desde el punto de vista normativo y operativo; de ésta acción depende en gran medida garantizar la calidad y eficiencia del proceso que se desarrolla. En el supervisor descansa la responsabilidad de orientar o corregir las acciones que se desarrollan.

Figura 1: Operador de SCADA interactuando con la interfaz hombre-máquina (HMI). (kersel, 2008)

1.1.1. Funcionalidades principales.

A continuación se muestran algunas de estas funcionalidades:

- **Adquisición y almacenamiento de datos:** Para la recolección, procesamiento y almacenamiento de la información recibida, en forma continua y confiable.
- **Supervisión de procesos:** Para el monitoreo del funcionamiento del proceso, procesamiento estadístico de los datos y confección de reportes. Además se brindan notificaciones al operador del sistema sobre cambios detectados en la instalación. Estas notificaciones se clasifican principalmente en alarmas y eventos. Las **alarmas** se basan en la vigilancia de los parámetros de las variables del sistema. Son los sucesos no deseables, porque su aparición puede dar lugar a problemas de funcionamiento. Este tipo de sucesos requieren de la atención de un operario para su solución antes de que se llegue a una situación crítica que detenga el proceso. El resto de las situaciones normales, tales como puesta en marcha, paro, cambios de consignas de funcionamiento, consultas de datos, entre otras, serán los denominados **eventos** del sistema o sucesos. Los eventos no requieren de la atención del operador del sistema, registran de forma automática todo lo que ocurre en el sistema. También será posible guardar estos datos para su posterior consulta.

- **Control de procesos:** Para el control de los procesos de la fábrica, planta o industria, actuando sobre los reguladores autónomos básicos como eventos y alarmas, o directamente sobre el proceso mediante las salidas conectadas.
- **Transmisión de datos:** Para la transmisión de información entre dispositivos de campo y computadoras de control o entre dispositivos ubicados a un mismo nivel.
- **Presentación de la información:** Para la representación gráfica de los datos. Interfaz del Operador o HMI (Human Machine Interface).

1.1.2. Componentes de hardware.

Dentro de los componentes que integran un sistema SCADA, se destacan cuatro fundamentales: (Castro Lozano, 2007)

- **Instrumentación de campo:** Dentro de instrumentación de campo se incluye los sensores y actuadores que están directamente conectados a la planta o el equipo que está siendo controlado y supervisado por el sistema SCADA. Generalmente no suelen ser considerados parte del sistema SCADA, pero sí son parte integrante del esquema general de control.
- **Unidad Terminal Remota (RTU, Remote Terminal Units):** Estos son dispositivos basados en microprocesadores que están conectados físicamente al instrumento de campo, permitiendo obtener señales independientes de los procesos y enviar la información a un sitio remoto que generalmente es una sala de control donde se encuentra un sistema central SCADA, el que permite visualizar las variables enviadas por la RTU. Las RTU en los últimos tiempos han sido desplazadas por los Controladores Lógicos Programables (PLC) quienes han fortalecido sus facilidades de comunicación a través de protocolos para sistemas de control.
- **Ordenador central (MTU, Master Terminal Unit):** Esta es la estación maestra o equipo host que actúa como central controlador para el sistema SCADA. La MTU es la responsable de recoger los datos del campo de las RTU y procesar la información para generar las medidas de control necesarias, todos estos datos son visualizados a través del HMI.
- **Comunicaciones:** Esta es la espina dorsal de todo sistema SCADA. Se utiliza para la transferencia de datos entre el Ordenador Central y la Unidad Terminal Remota, o con otros equipos que se encuentran dentro de la red corporativa. Los medios de comunicación pueden ser cableados, inalámbricos o la combinación de estos.

1.1.3. Componentes de software.

Los módulos o bloques de software que permiten las actividades de adquisición, supervisión y control son los siguientes:

- **Configuración:** Permite al usuario definir el entorno de trabajo de su aplicación, según la disposición de pantallas requerida y los niveles de acceso para los distintos usuarios.
- **Interfaz gráfica del operador:** Proporciona al operador las funciones de control y supervisión de la planta.
- **Módulo de proceso:** Ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas. La programación se realiza por medio de bloques de programa en lenguaje de alto nivel.
- **Gestión y archivo de datos:** Se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a estos.
- **Comunicaciones:** Se encarga de la transferencia de información entre la planta y la arquitectura de hardware que soporta el sistema SCADA, y entre ésta y el resto de elementos informáticos de gestión.

1.1.4. Comunicaciones.

Las comunicaciones desempeñan un papel significativo en los sistemas SCADA. Contribuciones recientes se han centrado en dos grandes áreas: comunicación con los dispositivos a nivel de campo, y la comunicación con equipos de supervisión a alto nivel.

- En la actualidad las comunicaciones de tipo serie a **nivel de campo** han estado cambiando por los buses de campo y la red Ethernet que permiten un acceso rápido a las mediciones que se realizan. Además como los sistemas SCADA tienden a crecer y las operaciones están cubriendo cada vez áreas más extensas, diversos sistemas han necesitado integrarse en un único sistema. El uso de las tecnologías basadas en estándares interoperables se vuelve cada vez más necesarias.
- La integración de sistemas supervisores a **alto nivel** se encuentra usualmente basada en Ethernet y TCP/IP, que está ganando gran popularidad en la industria. Estos protocolos proveen los datos a un servidor que a su vez proporciona estos datos a las estaciones de trabajo para la operación, ingeniería y mantenimiento. En la actualidad esta información no se limita a brindarse en una Intranet corporativa sino que también para Internet, medio que en la actualidad está siendo

utilizado como capa de aplicación en diversos sistemas SCADA.

Un sistema SCADA cubre generalmente áreas geográficas grandes, y necesita diversos medios de comunicación. Un aspecto importante que define la calidad de la tecnología de un sistema SCADA es la capacidad de garantizar la confiabilidad en la transferencia de los datos al usar estos medios. Los sistemas SCADA utilizaron en sus inicios, 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.

La selección de redes de comunicación en sistemas SCADA siempre estará ligada a la topografía del lugar donde se despliega el sistema, al presupuesto con que cuenta la empresa para invertir en la solución, entre otros. Más complejo se puede tornar la definición de una tecnología para la gestión de las comunicaciones entre las aplicaciones de un sistema SCADA teniendo en cuenta los niveles de persistencia, seguridad, velocidad y precisión de los datos que necesitan estos para su operación. Por lo general se utiliza un "*middleware*" o software de comunicación entre aplicaciones. Al remitirse a la pirámide de automatización de un sistema SCADA, se puede decir que el *middleware*, es la capa de software que se encuentra por encima de los niveles físicos y de red y por debajo de las aplicaciones de usuario. En la figura 2 se muestra con una línea roja la ubicación de esta capa en la pirámide de automatización de un sistema SCADA.

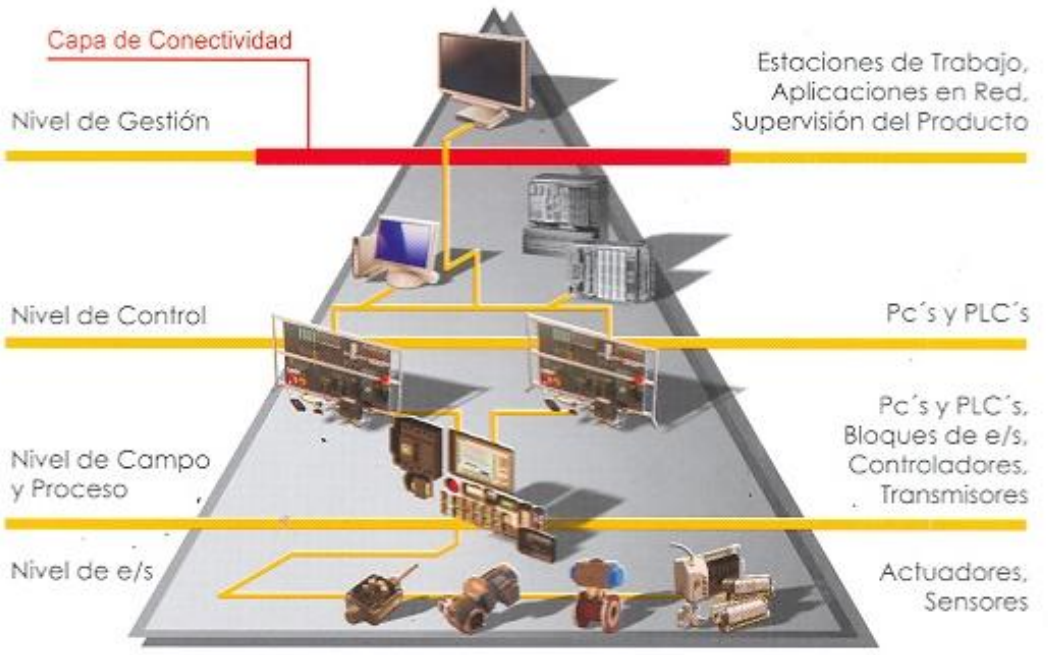

Figura 2: Pirámide de Automatización de un SCADA.

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.

1.2. Sistemas SCADA y Comunicación con Terceros.

La comunicación con terceros se refiere a la comunicación que se establece entre sistemas externos y un sistema SCADA con el objetivo de intercambiar información. Estos terceros pueden ser otros sistemas SCADA, Sistemas de Control Distribuido (DCS por sus siglas en inglés), aplicaciones de gestión, gerenciales y de negocio.

En la actualidad dentro de los terceros más comunes se encuentran los Sistemas de Ejecución de Manufactura (en inglés **MES**, Manufacturing Execution Systems), y los Sistemas de Planificación de Recursos de la Empresa (en inglés **ERP**, Enterprise Resource Planning).

Sistemas de Ejecución de Manufactura (MES): (Sepúlveda, 2006)

Los Sistemas de Ejecución de Manufacturas tienen como objetivo fundamental, maximizar el rendimiento operacional del proceso productivo. Se alimentan en tiempo real y en línea de datos provenientes de otros sistemas (historiadores de proceso, HMI/SCADA, servidores OPC, bases de datos relacionales, entre otros), y los convierte en información para la toma de decisiones. Algunos de estos sistemas MES se utilizan también en muchas instalaciones de producción para entregar sus resultados a otros software como los sistemas ERP. Varios de estos sistemas son modulares, lo que permite una implementación por etapas y también proveen conectividad con distintos sistemas, tanto a Nivel SCADA como a Nivel de Planificación.

Sistemas de Planificación de Recursos de la Empresa (ERP):

Los Sistemas de Planificación de Recursos de la Empresa, son sistemas de información para la gestión integral de las partes más importantes del negocio. Estos abarcan diferentes áreas como: Finanzas, Producción, Logística, Recursos Humanos y *Marketing*. Se nutren de la información brindada por sistemas MES para optimizar la gestión de la producción. Dentro de los sistemas ERP más conocidos se encuentran: *SAP, Baan, QAD, SSA, PeopleSoft, JD Edwards, Epicor Software y Visual Manufacturing.*

1.2.1. Servicios brindados a Terceros en Sistemas SCADA.

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.

Los servicios más comunes que deben proveerse a terceros por un sistema SCADA son los siguientes: (Herrera Vásquez, 2008)

- **Acceso a variables:** 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.
- **Acceso a 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 puede ser sincrónica o asincrónica en dependencia de la naturaleza de las aplicaciones. Usualmente se utiliza este servicio para la escritura de variables ya que esta interacción debe de 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.

1.2.1.1. Variables en un sistema SCADA.

El flujo principal de información en los sistemas SCADA lo constituyen las variables (puntos). Estas variables pueden representar innumerables indicadores como son: presión, temperatura, flujo, potencia, peso, intensidad de corriente, voltaje, potencial hidrógeno, densidad, carga, resistencia o capacitancia entre otros. Las variables son adquiridas mediante instrumentación o utilizando sensores conectados a autómatas o equipos de control. Después de convertidas a señales eléctricas, estas variables pasan a ser estructuras que contienen datos, los que pueden ser de tipos simples (entero, flotante, cadenas, y otros) o de tipos complejos. La información de estas variables permite conocer el estado del sistema y su historia.

Las variables contienen un conjunto de atributos que pueden variar de un sistema a otro, generalmente se destacan:

1. **Nombre y/o Identificador:** Es el atributo que identifica y hace única a la variable en el sistema.
2. **Valor:** Es el atributo que contiene el estado (valor) de la variable en un instante de tiempo.
3. **Marca de tiempo:** Representa el instante de tiempo en que se adquirió o calculó la variable.
4. **Calidad:** Representa la calidad del valor de la variable.

Por su naturaleza, el Servicio de Acceso a Variables es el más utilizado y el que mayores variantes para su resolución presenta en la actualidad. El servicio debe permitir la lectura y monitorización de estas variables; comúnmente se emplean servicios de seguridad y suscripción de los clientes a las variables para asegurar los datos que se manejan y optimizar la comunicación.

El servicio debe ser flexible y permitir el establecimiento de la comunicación entre estaciones con distintos sistemas operativos y superar los límites de una LAN (Local Area Network), por ejemplo, vía Internet. Esta comunicación debe ser eficiente, garantizando una velocidad en la transmisión de las variables, adecuadas para aplicaciones de visualización y cálculos de algoritmos. De todos los servicios que se deben proveer a terceros, este es el que más necesidad de transmisión de datos en tiempo real presenta, comúnmente muestreos inferiores a un segundo. Este requerimiento se impone en los sistemas

de tipo SCADA debido a que el operador asume que los datos que está recibiendo están siendo actualizados en tiempo real y a partir de estos, toma las acciones pertinentes de control.

1.3. Tendencias y Tecnologías actuales.

En la actualidad existen diversas tecnologías de comunicación distribuida que permiten el desarrollo de servicios de acceso a variables, en esta sección se describirán las más utilizadas a nivel mundial.

1.3.1. OPC.

El OLE para el Control de Procesos (OPC), es un conjunto de especificaciones basadas en los estándares de *Microsoft* (COM, DCOM, OLE *Automation* y *ActiveX*) que cubren los requerimientos de comunicación industrial entre aplicaciones y dispositivos, especialmente en lo que se refiere a la atención al tiempo real. (Herrera Vázquez, 2008)

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.

En la actualidad OPC cuenta con una variedad de especificaciones como son Alarmas y Eventos, Seguridad, Comandos, Acceso a datos Históricos, pero solo se analizarán en profundidad los estándares de Acceso a Datos, Acceso a Datos XML y OPC Arquitectura Unificada que son los que generalmente se utilizan para la implementación de servicios de acceso a variables.

Ventajas de OPC

- OPC brinda apertura de comunicación a plataformas no industriales, posibilitando así realizar soluciones costo-efectivas a procesos específicos, se puede mencionar también la facilidad de comunicación de los sistemas SCADA con los sistemas de automatización, dando libertad casi total de elección.
- Existe una gran variedad de servidores OPC para todas las marcas y estándares, permitiendo elegir el más adecuado para las necesidades o conocimientos de cada uno, de igual manera los fabricantes de hardware sólo tienen que desarrollar un conjunto de componentes de programas para que los clientes los utilicen en sus aplicaciones, por lo que éstos no tienen que adaptar los drivers ante cambios del hardware.
- Con OPC, la integración de sistemas en un entorno heterogéneo se vuelve sencilla, puesto que un cliente OPC se puede conectar a servidores OPC proporcionados por más de un proveedor.

- OPC permite integrar paquetes nuevos de software SCADA con los sistemas ya existentes, incluso de varias décadas de instalación, aun cuando se ha perdido o no se han desarrollado interfaces compatibles.
- OPC brinda integración multiplataforma (*Windows*, *Linux*, *Unix*, *SuSe*) a través del uso de COM, DCOM, *ActiveX* y *EntireX*, posibilita también el desarrollo de las comunicaciones dentro de redes LAN (*Local Access Network*) y WAN (*World Access Network*), así como exportar datos a Internet de forma eficaz y flexible, desde el nivel de procesos hasta el nivel de gestión.

Desventajas de OPC

- En soluciones de software la utilización de OPC puede provocar que el desempeño en términos de tiempo de respuesta y fiabilidad no resulte óptimo.
- La utilización de un servidor OPC básico puede ser muy sencilla, pero generalmente son los que tienen menores prestaciones. Los servidores OPC de calidad industrial (que pueden dar respuestas en tiempo real) demandan procedimientos de configuración más engorrosos.
- En varias ocasiones utilizar el estándar OPC resulta más caro que adquirir un SCADA con sus drivers integrados.
- OPC muestra ser un estándar industrial ideal pero al ser tan transparente en el marco de las aplicaciones e interoperable en distintas plataformas presenta problemas de Seguridad en estos aspectos. Por ejemplo:
    - Los creadores de Buses de Campo, han desarrollado pasarelas para interconectar sus protocolos propietarios a redes Ethernet, lo que puede causar que personas inescrupulosas tengan la posibilidad de acceder desde el nivel de gestión hasta el nivel de proceso, dañando en gran medida todo el sistema de comunicaciones.
    - Debido a que en la actualidad existe la capacidad de desarrollar programas ejecutables fundamentados en las tecnologías bases de OPC, e ingresar en los sistemas de red de diferentes sistemas operativos a nivel WAN a través de Entire X y Active X, la red de comunicaciones OPC puede exponerse a escala mundial y ser accesible desde cualquier plataforma operativa.(Colectivo de Autores, 2000)

1.3.1.1. Acceso a Datos OPC.

Acceso a Datos OPC (OPC DA) se trata de una especificación de la Fundación OPC que define cómo se pueden transferir datos en tiempo real entre una fuente de datos y un destino determinado (entre un PLC y un HMI). Usando OPC DA, aplicaciones de software pueden obtener datos en tiempo real con el fin de poder controlar un determinado proceso. Es usado fundamentalmente para la adquisición de datos pero también puede ser utilizado para escribir valores permitiendo así el manejo de aplicaciones de control y supervisión, por lo que OPC DA es muy adecuado para utilizarlo en sistemas SCADA. OPC DA suele ser utilizado por casi todas las aplicaciones de control de procesos que se utilizan tales como un HMI, sistemas ERP, entre otros. Esta especificación no es para el manejo de datos históricos, si lo que se necesita es el acceso a datos históricos, la fundación OPC desarrolló la especificación OPC Acceso a Datos Históricos (OPC HDA).

No existe una única especificación de OPC DA, esta ha evolucionado en el tiempo como se muestra en la siguiente tabla: (MatrikonOPC, 2009)

Año	Versión	Comentario
1996	1.0	Especificaciones iniciales.
1997	DA 1.0a	Adopta el nombre de Acceso a datos (DA), para diferenciarla de otras especificaciones que se desarrollaron simultáneamente.
1998	DA 2.0 – DA 2.05a	Se le agregan numerosas especificaciones, aclaraciones y modificaciones.
2003	DA 3.0	Más adiciones y modificaciones.

Tabla 1: Versiones de la especificación OPC DA.

Características de OPC DA

- El estándar Acceso a Datos le permite a clientes OPC la lectura, modificación y monitorización de variables del proceso como:

  - Datos de sensores en Tiempo Real (temperatura, presión, flujo).

  - Parámetros de Control (abrir, cerrar, ejecutar, parar).

  - Información de Estado (estado de conexiones hardware y del software local y subsistemas).

  - Puede representar cualquier dato disponible.

- Las interfaces brindadas por OPC Acceso a Datos:

  - Permiten desarrollar aplicaciones para acceso sencillo a subsistemas de datos.

  - Soportan acceso a datos por llamada y excepción.

- Son independientes del fabricante.

- Son flexibles, eficientes y escalables.

- La especificación OPC Acceso a Datos es la primera del grupo de especificaciones OPC.
- Está basado en COM.
- Diseñado para comportarse eficientemente en red.
- Tiene su mayor aceptación y adopción en entornos industriales.

Uso del estándar OPC DA en la actualidad.

En la actualidad existen en el mercado, gran cantidad de sistemas distribuidos que utilizan el estándar OPC DA, en gran parte debido a que este fue el primero de toda la serie de estándares con que cuenta la Fundación OPC. El software ***Genesys32*** es un producto comercializado por la Empresa *ICONICS*, encargado de la construcción de aplicaciones de control y automatización que requieran visualización, control y supervisión, adquisición de datos y sistemas avanzados de alarmas; este software utiliza el estándar OPC DA para la captura de datos en tiempo real, su integración en bases de datos y el análisis de tendencias. ***SIMATIC WinCC*** es un sistema de supervisión distribuido por la empresa multinacional alemana *SIEMENS*, que funciona sobre plataforma *Microsoft Windows* 95 y *Windows NT*, que utiliza el estándar OPC DA 3.0. Este sistema cuenta con un servidor OPC DA integrado, que tiene acceso en línea a todos los valores que se manipulan en el sistema y por otro lado brinda estos datos a sistemas ERP y sistemas MES.

1.3.1.2. OPC Acceso a Datos XML.

Con el objetivo de brindar información acerca de las aplicaciones y sistemas a través de la red sin necesidad de estar atado a un sistema operativo, tipo de computador e incluso un lenguaje de programación específicos y basado en los servicios web, la fundación OPC desarrolló el estándar OPC XML-DA (OPC XML Data Access) donde se adoptan las tecnologías XML para facilitar el intercambio de información entre aplicaciones como en OPC-DA pero en vez de utilizar tecnología COM/DCOM utiliza mensajes SOAP (sobre HTTP) con documentos en XML. 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. (OPC Foundation, 2003)

OPC XML-DA se basa en la arquitectura cliente/servidor, brinda 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.

La especificación OPC-XML-DA proporciona:

- Soporte para datos OPC Data Access 2.0x/3.0.
- Soporte para HTTP, y SOAP.
- Soporte para servicios basados en suscripción.
- Soporte para un enfoque de seguridad.

XML-DA soporta los siguientes servicios de suscripción básica, los cuales se denominan transacciones:

- *Subscribe*: Utilizado para iniciar un contrato de suscripción con el servidor.
- *SubscriptionPolledRefresh*: Usado para adquirir los últimos cambios de valor en los datos cada cierto tiempo.
- *SubscripctionCancel*: Empleado para terminar la suscripción.

  Dentro de las funcionalidades que ofrecen el cliente y el servidor se encuentran: *Browse, GetProperties, GetStatus, Read y Write,* también ofrece seguridad independientemente del tipo de red en que se encuentre. (FERNANDO ALEJANDRO CAMPOS, 2004)

Uso del estándar OPC XML DA en la actualidad.

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* (ICONICS, 2009). Kassl GmbH es una empresa alemana que se especializa en el desarrollo de software y componentes para OPC (*Kassl GmbH*, 2009). *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. (*Northern Dynamic*, 2009)

1.3.1.3. OPC Arquitectura Unificada.

OPC Arquitectura Unificada (UA) es en la actualidad la generación más novedosa de OPC. La primera parte de la especificación fue liberada en agosto de 2006, después de tres años de haber sido anunciada.

La fundación OPC ha desarrollado la especificación OPC UA teniendo en cuenta los siguientes factores:

- Las bases de las primeras especificaciones de OPC, son ahora oficialmente tecnologías heredadas de Microsoft, como COM y DCOM.
- Los Servicios Web ofrecen ahora el principal mecanismo para el transporte de datos entre ordenadores (y también proporcionan una mejor opción para las comunicaciones con dispositivos de una planta).
- Especificaciones anteriores de OPC no lograron presentar un único modelo de datos coherente.

En la figura 3 se muestran las características más novedosas de esta arquitectura.

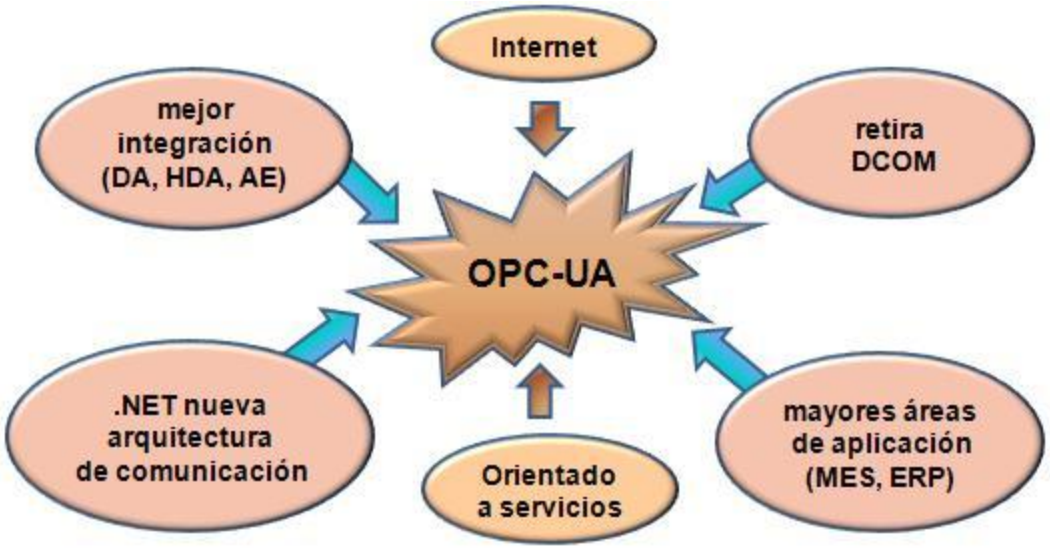

Figura 3: Características novedosas de OPC UA.

La especificación OPC UA es mucho más compleja que las anteriores especificaciones OPC; a continuación se describen una serie de características:

- La arquitectura de OPC UA ofrece un modelo único y coherente de datos, y utiliza los servicios web para el transporte principal. Tecnologías como .NET y WCF también se ofrecen para la conveniencia de desarrolladores y DCOM para el rendimiento.
- Combina las funcionalidades brindadas por las especificaciones OPC DA, A&E y HDA en un conjunto único de servicios.
- Posee un modelo de estructuras de datos complejas para la colaboración con otras organizaciones de normalización.
- Permite el manejo de datos sencillos y complejos de forma nativa, datos OPC, tanto los que son estándares como los personalizados.
- Puede ser utilizado en diferentes plataformas, desde los sistemas empotrados, hasta los sistemas empresariales.
- Es altamente escalable, pues posee espacios de nombres avanzados que permite tipos de nodos y relaciones ilimitados. Cada nodo puede participar en un número ilimitado de relaciones con otros nodos, esto permite que no exista en el futuro un sistema que sea demasiado complicado para ser modelado vía OPC-UA.
- Un servidor OPC UA agrega todas las funcionalidades que antes eran separadas en distintos servidores por el modelo OPC.

Uso del estándar OPC UA en la actualidad.

El estándar OPC UA es usado en la actualidad por diversas empresas, a continuación se muestra un listado publicado por Thomas J. Burke, Presidente de la Fundación OPC y su Director Ejecutivo:

ABB	Invensys/Foxboro	SISCO
Absynt Technologies Ltd	Invensys/Wonderware	**SMAR**
ascolab GmbH	**Kepware**	**Softing AG**
Beckhoff	**Matrikon**	**Software Toolbox**
CAS	Metso Automation	SRI International
Cognex	**Microsoft**	Tampere University of Technology
Cyberlogic	OPC-F	Technosoftware AG
Helsinki University of Technology	**OSIsoft, Inc.**	VTT
Honeywell	Prosys PMS Ltd	Wapice Ltd
Iconics	**Rockwell**	**Yokogawa Electric Asia**
InduSoft LLC	SAP	
Ing.-Buero Allmendinger	**Siemens**	

Figura 4: Empresas que utilizan el estándar OPC.

1.3.2. Servicios Web.

Según el World Wide Web Consortium (W3C), un servicio web es una aplicación software identificada por un URI, cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML. Un servicio Web soporta interacciones directas con otros agentes software utilizando mensajes XML intercambiados mediante protocolos basados en Internet.

Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios web. A diferencia del modelo tradicional de cliente/servidor, los servicios Web no proporcionan una interfaz gráfica de usuario. En cambio, los servicios Web comparten la lógica de negocio, datos y procesos a través de una interfaz de programación mediante una red (internet por ejemplo). Los desarrolladores pueden añadir el servicio Web a una GUI (por ejemplo, una página Web o un programa ejecutable) para ofrecer una funcionalidad específica a los usuarios.

Los servicios web se basan en diversos estándares abiertos (XML, SOAP, WSDL y UDDI) para la integración de aplicaciones proporcionando interoperabilidad, seguridad y capacidad de administración. XML se utiliza para etiquetar los datos, SOAP para transferirlos, WSDL para describir los servicios web disponibles y UDDI para listar los servicios que estén disponibles.

A continuación se exponen los conceptos fundamentales de cada uno de estos estándares:

- **XML (Extensible Markup Language)**: Es la base de la tecnología de los Servicios Web, es un formato de texto estructurado, extensible, portable, por lo que constituye una forma eficiente y efectiva de almacenar y compartir información, contribuyendo de forma fácil a que diferentes aplicaciones puedan acceder y controlar estos datos. Brinda la posibilidad a los diseñadores de crear sus propias etiquetas personalizadas, lo que permite la definición, transmisión, validación e interpretación de datos entre aplicaciones y entre organizaciones. XML se deriva del SGML, que sin embargo es un lenguaje muy complejo para ser usado en la mayoría de las aplicaciones.
- **SOAP (Simple Object Access Protocol):** Es un protocolo de alto nivel que estandariza el intercambio de mensajes entre aplicaciones. Por ello la función básica de SOAP es definir un formato de mensajes estándar (basado en XML) que encapsulará la comunicación entre aplicaciones. Este protocolo deriva de un protocolo creado por David Winer en 1998, llamado XML-RPC. SOAP fue creado por Microsoft, IBM y otros, se encuentra actualmente bajo el auspicio de la W3C. Los mensajes SOAP son independientes de cualquier sistema operativo o protocolo y puede ser transportados utilizando una variedad de protocolos de Internet como: **FTP** (File Transfer Protocol), **SMTP** (Simple Mail Transfer Protocol), **HTTP** (Hypertext Transfer Protocol) y **HTTPS** (HTTP Secure).
- **WSDL (Web Services Description Languages):** Es un estándar de descripción de servicios web, utiliza un documento con formato XML que detalla la forma en la cual los clientes externos pueden interactuar con el servicio web existente, los métodos que soportan y la sintaxis de los protocolos de comunicación (HTTP, SOAP) que utiliza. Un documento WSDL contiene información acerca de la interfaz, la semántica y los aspectos administrativos involucrados en una solicitud realizada a un servicio web. El WSDL es equivalente a los lenguajes de definición de interfaces (IDLs) utilizado por los *middleware*, pero en el caso de los Servicios Web son más complejos de describir.
- **UDDI (Universal Description Discovery and Integration):** Es un directorio distribuido y basado en Web que permite listar, buscar, describir y descubrir servicios web. Se asemeja en su funcionalidad a las páginas amarillas de un directorio telefónico. UDDI define estructuras de datos y APIs para buscar y publicar descripciones de servicios. Permite a los desarrolladores encontrar información para escribir los clientes de los servicios, y posibilita a estos últimos preguntar al registro y obtener las referencias a los servicios de interés.

Este directorio distribuido se clasifica en función del uso que se le dará a la información, como se muestra a continuación: (Álvarez, Bañares, 2008)

- **Páginas blancas:** Estas recogen la información de contacto. Listados de organizaciones y servicios. Los clientes UDDI pueden encontrar servicios Web dados por una empresa.
- **Páginas amarillas:** Estas recogen la información de la industria. Clasificación de compañías y servicios Web de acuerdo a una taxonomía.
- **Páginas Verdes:** Estas recogen la información técnica y especificaciones. Describe como puede ser invocado un servicio Web. Contienen punteros a descripciones de servicios (descripciones externas al registro UDDI).

Para mejorar la interoperabilidad entre distintas implementaciones de servicios web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares anteriormente mencionados.

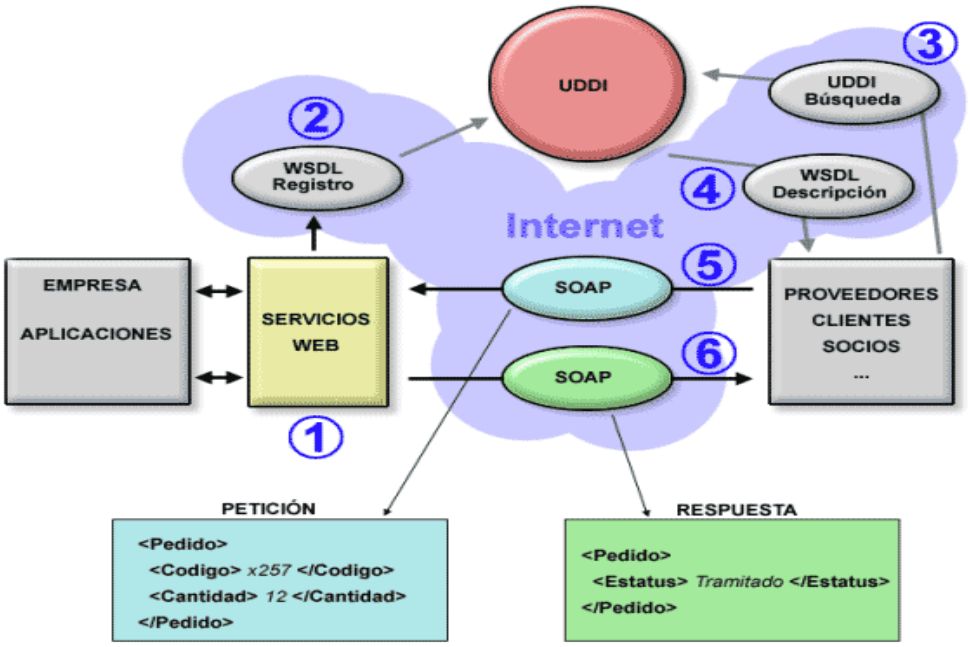

Figura 5: Funcionamiento de los Servicios Web.

A continuación, a través de una explicación de la figura 5 se expone como interactúan y la funcionalidad de los diversos estándares que conforman los servicios web: (MonteJava, 2007)

- El ciclo se origina cuando las empresas se deciden a desarrollar y exponer la funcionalidad de sus aplicaciones en forma de Servicio Web.
- Una vez que los Servicios Web se han desarrollado, deben ser registrados en un nodo UDDI para poder ser localizados por los potenciales usuarios. En dicho registro se aportarán datos sobre la empresa, los Servicios Web que se ofrecen y demás, también la descripción de las interfaces de uso de cada Servicio Web (WSDL). Cuando algún consumidor solicite dicho Servicio Web, el servidor UDDI le redireccionará la URI proporcionada por el fabricante.
- Los posibles consumidores (proveedores, clientes, socios, entre otros) se conectan al servidor UDDI para buscar los Servicios Web que les interesan.
- Una vez que encuentran el Servicio Web que desean, obtienen la descripción de sus interfaces de uso (WSDL).
- Gracias a la descripción de las interfaces de uso, los consumidores son capaces de elaborar paquetes SOAP para comunicarse con el proveedor del Servicio Web.
- El proveedor del Servicio Web elabora un paquete SOAP como respuesta a la petición del consumidor del Servicio Web.

Ventajas de los Servicios Web: (Herrera Vásquez, 2008)

- Promueven la interoperabilidad**:** La interacción entre un proveedor y un solicitante de servicio está diseñada para que sea completamente independiente de la plataforma y el lenguaje. Esta interacción requiere un documento WSDL para definir la interfaz y describir el servicio, junto con un protocolo de red, generalmente se utiliza HTTP.
- Permiten una integración fácil: El proceso de descubrimiento se ejecuta dinámicamente, a medida que los solicitantes de servicios utilizan a los directorios distribuidos para encontrar proveedores de servicios. Una vez que el solicitante y el proveedor de servicio se han ubicado, se utiliza el documento WSDL del proveedor para enlazar al solicitante con el servicio, esto significa que los solicitantes, los proveedores y los directorios distribuidos actúan en conjunto para crear sistemas que son auto-configurables, adaptables y robustos.
- Reducen la complejidad: Los solicitantes y los proveedores del servicio se preocupan por las interfaces necesarias para interactuar. Como resultado, un solicitante de servicio no sabe cómo fue implementado el servicio por parte del proveedor, y éste a su vez, no sabe cómo utiliza el cliente el

servicio. Estos detalles se encapsulan en los solicitantes y proveedores (encapsulamiento).

- Dan una "nueva vida" a las aplicaciones: Es relativamente correcto tomar una aplicación de escritorio, generar una envoltura SOAP, luego generar un documento WSDL para moldear la aplicación como un servicio web.
- Abren la puerta a nuevas oportunidades de negocio: Los servicios web facilitan la interacción con socios de negocios, al poder compartir servicios internos con un alto grado de integración.

Desventajas de los Servicios Web

- Para el intercambio de datos no se pueden comparar con los estándares abiertos de comunicación distribuida como CORBA, debido a que estos poseen un mayor grado de desarrollo.
- Posee un bajo rendimiento comparado con otras tecnologías de comunicación distribuida, debido a que se adopta un formato basado en texto, como lo es XML.
- Al apoyarse en protocolos basados en internet, puede evitar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

Uso de servicios web en la actualidad:

Con la generalización del uso de Internet se ha hecho necesario que los desarrolladores de SCADA incluyan facilidades de operatividad a través de cualquier navegador. Un ejemplo de ello es el software *SuiteVoyager* de *Invensys Wonderware*, que es un portal de Servicios Web que puede ofrecer información de diversos servidores de datos mediante el uso de una conexión sencilla a Internet y el empleo de un navegador convencional, tal como *Internet Explorer*. (*Invensys Systems*, *Inc*., 2009) También a través de esta solución, los ingenieros tendrán un acceso en línea a documentos técnicos y de mantenimiento. Por su parte la firma *ICONICS* ofrece numerosas herramientas de acceso remoto como son: *ICONICS WEB-HMI Technology*, *ICONICS GEN-Browser node*, *ICONICS GENESIS32 remote node* y *Microsoft* Terminal Server. Otro ejemplo de este tipo de aplicación es el *infoAgent* de GE *Fanuc- Intellution* que proporciona un servidor y admite múltiples clientes. (*GRAELLS*, 2003)

Utilizando este tipo de facilidad los operadores pueden recibir pantallas actualizadas en tiempo real directamente desde su sistema de automatización, así como puede ser trasmitida información de cambio de estado, requerimientos de mantenimiento o alarmas a cualquier computadora que esté conectada a Internet. A su vez pueden tomar decisiones de forma inmediata que controlen cualquier evento inesperado que se produzca en el sistema sin necesidad de trasladarse físicamente a la instalación.

1.3.3. ICE (*Internet Communication Engine*).

ICE es un *middleware* orientado a objetos que ofrece herramientas, API, bibliotecas para la creación de aplicaciones cliente/servidor. Las aplicaciones desarrolladas utilizando ICE, son adecuadas para el uso en entornos heterogéneos, ya que los clientes y servidores pueden ser implementados en diferentes lenguajes de programación. Es multiplataforma, funciona en máquinas de diferentes arquitecturas y sobre una variedad de redes. El código fuente de este *middleware* es portable, independiente del entorno donde se despliegue y está distribuido bajo la licencia GPL (*General Public License*). ICE es una alternativa más moderna a CORBA, COM/DCOM/COM+.

Dentro de las funcionalidades que cumple ICE se encuentran:

- Provee un *middleware* orientado a objetos para su uso en entornos heterogéneos.
- Provee características que permiten el desarrollo de aplicaciones distribuidas en tiempo real.
- Rehúye de la complejidad innecesaria.
- Provee una implementación eficiente en ancho de banda, uso de memoria y sobrecarga en el CPU.
- Provee medidas de seguridad que permiten utilizar las aplicaciones en redes inseguras.

Características generales de ICE (*ZeroC*, 2009)

ICE es un *middleware*, una plataforma para desarrollo de aplicaciones de comunicación para Internet de alto rendimiento que incluye varias capas de servicios y extensiones, y se compone de los siguientes paquetes:

- ***Slice*:** Es el Lenguaje de especificación de ICE que establece un contrato entre clientes y servidores, también se utiliza para describir los datos persistentes.
- **Compiladores *Slice*:** Las especificaciones SLICE pueden ser compiladas en diferentes lenguajes de programación. Soporta C++, Java, C#, *Visual Basic*, *Python*, PHP y *Ruby*. Los clientes y servidores de ICE trabajan juntos independientemente del lenguaje de programación.
- ***Ice*:** Es la biblioteca núcleo de ICE, que entre otras funciones gestiona todas las tareas de comunicación utilizando un protocolo de gran eficiencia (incluyendo el protocolo de compresión y apoyo para TCP y UDP), proporciona un contenedor de hilos (pull de hilos) flexibles para servidores multiproceso y una funcionalidad adicional que apoya la extrema escalabilidad con millones de potencialidades de objetos ICE.

- ***IceUtil*:** Una colección de funciones de utilidad tales como la manipulación de un estándar de codificación e hilos de programación. (Solamente C++).
- ***IceBox*:** Un servidor de aplicaciones específicamente para aplicaciones ICE. Puede ejecutar y administrar los servicios ICE que están cargados dinámicamente como una DLL, biblioteca compartida o una clase Java.
- ***IceGrid*:** Un sofisticado servidor de activación y despliegue de herramientas avanzadas para computación *grid*. Además de ser un servicio de localización, *IceGrid* tiene muchas otras características dentro de las que se encuentran la replicación y balanceo.
- ***Freeze*:** *Freeze* representa el conjunto de servicios persistentes de ICE:
  - ✓ *Freeze evictor* es una implementación altamente escalable de un servicio de localización de ICE que proporciona persistencia automática y desalojo de objetos ICE con un código de aplicación mínimo.
  - ✓ *Freeze map* es un contenedor asociativo genérico. Las aplicaciones interactúan con *Freeze map* como cualquier otro contenedor asociativo, excepto que las claves y los valores de este son persistentes.

- **Ice SSL:** Un SSL dinámico que transporta extensiones para el núcleo de ICE. Ofrece autenticación, cifrado e integridad en los mensajes utilizando el protocolo estándar de la industria SSL.
- **Glacier:** Uno de los retos más difíciles para los *middleware* es la seguridad y los cortafuegos. Glacier es la solución de cortafuegos para ICE, que simplifica el despliegue de aplicaciones seguras. *Glacier* autentica y filtra las solicitudes del los clientes y permite llamadas de retorno al cliente en un modo seguro. En combinación con Ice *SSL*, *Glacier* ofrece una potente solución de seguridad que no permite la entrada de intrusos y es fácil de configurar.
- ***IceStorm*:** Es un servicio eficiente de publicación-suscripción para aplicaciones Ice. Desde un punto de vista funcional, *IceStorm* actúa como mediador entre el publicador y el subscriptor, proporcionando varias ventajas:
  - ✓ Sólo es necesaria una llamada al servicio *IceStorm* para distribuir la información a los subscriptores.
  - ✓ Independencia entre el emisor y los receptores de información, permitiendo que el primero se ocupe de las responsabilidades relativas a la aplicación y no de tareas administrativas.
  - ✓ Los cambios introducidos en el código son mínimos para incorporar la funcionalidad de

IceStorm.

- ***IcePatch*:** Es un servicio de parches para distribuciones de software. Mantener un software actualizado es a menudo una tarea tediosa. *Ice Patch* automatiza la actualización de archivos individuales así como jerarquías completas de directorios. Sólo los archivos que han cambiado son descargados a la máquina cliente, utilizando algoritmos de compresión eficiente.

Ventajas de ICE:

- Mantiene una semántica orientada a objetos.
- Proporciona un manejo síncrono y asíncrono de mensajes.
- Soporta múltiples interfaces.
- Es independiente de la máquina.
- Es independiente del lenguaje de programación.
- Es independiente de la implementación, es decir, el cliente no necesita conocer como el servidor implementa sus objetos.
- Es independiente del sistema operativo.
- Soporta el manejo de hilos.
- Es independiente del protocolo de transporte empleado.
- Mantiene transparencia en lo que a localización y servicios se refiere (*IceGrid*).
- Es seguro (SSL y *Glacier2*).
- Soporta comunicaciones cliente/servidor y publicación/suscripción.
- Permite hacer persistentes las aplicaciones (*Freeze*).
- Tiene disponible el código fuente.

Desventajas de ICE:

- No responde a ningún estándar, su especificación es de los creadores del proyecto *ZeroC*.

Uso de ICE en la actualidad:

Analytical Engineering, *Inc*, es un proveedor de servicios de pruebas e instrumentación para la industria del diesel. ICE se utiliza como infraestructura de comunicación en los sistemas de control de esta empresa.

Skype, es la oferta de comunicación a través de internet de más rápido crecimiento en el mundo, permitiendo a personas establecer, desde cualquier zona del planeta, comunicaciones de voz y vídeo,

utilizando ICE como parte de su infraestructura de comunicaciones. *Skype* está disponible en 27 idiomas y se utiliza en casi todos los países del mundo.

Baosteel, es una de las empresas de acero más rentables en el mundo con competencia a nivel internacional. Utiliza ICE como parte de su infraestructura de comunicaciones en sistemas de control de procesos en tiempo real.

Solution Space, usa ICE como base para recibir datos de aplicaciones de software industriales tales como, sistemas de visualización de la información y los sistemas de comunicación.

1.3.4. ArchestrA.

ArchestrA es la arquitectura tecnológica basada en .NET de *Microsoft*, desarrollada por *Wonderware* para facilitar e impulsar la Integración de Dispositivos y Sistemas a distintos niveles. Se trata de una arquitectura sobre la que se genera software que es fácilmente desplegable e integrable. *ArchestrA*, se compone de varios productos de software**:** (Logitek, 2006)

1. A nivel Servidor el producto base es *System Platform* 3.0, que incluye el motor de ejecución de aplicación (*Industrial Application Server*), servicios de almacenamiento y gestión de información de proceso e infraestructuras (*Historian*), servicios Web (*Wonderware Information Server*) y drivers de comunicación con dispositivos de campo para facilitar la integración horizontal.
2. A nivel Cliente: *InTouch*, *ActiveFactory*, Integración con MS Office.

Los productos construidos con la arquitectura *ArchestrA* reducen drásticamente los costes de desarrollo de nuevos proyectos de automatización y el tiempo requerido para su implementación por reutilización de aplicaciones existentes. *ArchestrA* permite la integración de las inversiones en sistemas de automatización existentes con la última tecnología en software, por lo que las operaciones de producción se transforman en las más eficientes y económicas posibles. El coste total de proyectos de automatización se reduce debido a que esta plataforma facilita la ampliación y expansión de sistemas existentes para mantener la competitividad y mejora e incrementa la productividad.

La arquitectura software de automatización e información *ArchestrA* proporciona estos beneficios de negocio a través del uso de un modelo común de aplicación modular. Los servicios más frecuentes requeridos por cada aplicación se facilitan bajo un conjunto común de servicios de infraestructura, por lo que los usuarios no tienen que invertir tiempo y recursos en el desarrollo de todas estas funciones en la creación de aplicaciones. En vez de esto, los usuarios pueden centrarse en invertir sus esfuerzos de ingeniería en el caso concreto de su proyecto, lo que proporciona un gran valor. La arquitectura *ArchestrA*

facilita además el trabajo de ingeniería cooperativo para que múltiples usuarios puedan desarrollar sus trabajos en paralelo y pone las herramientas para que los resultados de este esfuerzo sean fácil y automáticamente integrados.

Las soluciones de software de *Wonderware* demuestran su utilidad en más de 100.000 plantas de todo el mundo y cubren un amplio espectro de aplicaciones críticas para los negocios en empresas industriales y de manufactura. Estas soluciones de software integradas se enmarcan en las siguientes ofertas dirigidas a los clientes: HMI de supervisión, SCADA y Gestión de Producción y Rendimiento.

CAPÍTULO 2: SELECCIÓN DE TECNOLOGÍAS Y MODELADO DE LA SOLUCIÓN

En el presente capítulo se exponen los requisitos funcionales y no funcionales del Servicio de Acceso a Variables; también se analizan las tecnologías de comunicación distribuida abordadas previamente y se realiza la selección de las mismas para la implementación de este servicio. Se selecciona además la metodología a utilizar, los lenguajes, tanto de programación como de modelado, y las herramientas que se emplearán para la implementación del Servicio de Acceso a Variables. Se muestra además la integración del Subsistema de Comunicación con Terceros dentro del SCADA Guardián del ALBA, los componentes del Servicio de Acceso a Variables, la arquitectura a utilizar, así como las descripciones de los paquetes del diseño arquitectónicamente significativos.

2.1. Requerimientos del Servicio de Acceso a Variables.

A continuación se enuncian los requisitos con que contará el Servicio de Acceso a Variables del Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA. Para su obtención se tuvieron en cuenta diversos criterios de personal especializado en el trabajo con sistemas SCADA desplegados en la red de PDVSA pertenecientes al DST-AIT Mérida y de profesionales cubanos con experiencia acerca del tema en el país. Además se estudiaron las aplicaciones de software privativo utilizadas para la gestión de los datos provenientes de sistemas SCADA en esta red.

2.1.1. Requisitos Funcionales.

A continuación se exponen las funcionalidades (RF) identificadas en el proceso de conformación del Subsistema de Comunicación con Terceros para el Servicio de Acceso a Variables del guardián del ALBA:

RF 1. El Servicio de Acceso a Variables debe permitir a sistemas externos, el acceso a la información relacionada con variables del sistema SCADA Guardián del ALBA.

- **RF 1.1.** Se debe brindar un mecanismo que posibilite a sistemas externos, obtener la descripción y/o interfaces para la comunicación con el Servicio de Acceso a Variables.
- **RF 1.2.** El Servicio de Acceso a Variables debe proporcionar a sistemas externos un mecanismo de solicitud de suscripción a las variables manipuladas por el Guardián del ALBA.
- **RF 1.3.** El Servicio de Acceso a Variables debe entregar los valores de las variables a los terceros.

RF 1.3.1 El Servicio de Acceso a Variables debe adquirir los datos mediante la conexión con el *middleware* del Guardián del ALBA.

2.1.2. Requisitos No Funcionales.

A continuación se exponen las características (RNF) con que debe contar el Servicio de Acceso a Variables para que pueda ser integrado al sistema SCADA Guardián del ALBA y utilizado por sistemas externos.

RNF 1. Software

- **RNF 1.1** El Servicio de Acceso a Variables se debe desarrollar sobre el Sistema operativo GNU/Linux, distribución Debian, Kernel 2.6

RNF 2. Diseño e implementación

- **RNF 2.1** Para el desarrollo del Servicio de Acceso a Variables se debe utilizar el lenguaje de programación C++.
- **RNF 2.2** Para el desarrollo del Servicio de Acceso a Variables se debe utilizar como paradigma de programación, la Programación Orientada a Objetos.

RNF 3. Portabilidad

- **RNF 3.1** El Servicio de Acceso a Variables debe garantizar que los terceros accedan a él sin importar la plataforma que estos utilicen.

RNF 4. Fiabilidad

- **RNF 4.1** El Servicio de Acceso a Variables debe integrarse al Servicio de Seguridad del Guardián del ALBA para garantizar el acceso seguro y de personal autorizado a los datos.
- **RNF 4.2** El Servicio de Acceso a Variables debe garantizar una comunicación eficiente y fiable para evitar las pérdidas de información (puntos).

RNF 5. Rendimiento

- **RNF 5.1** El Servicio de Acceso a Variables debe garantizar una velocidad de transmisión de muestras en el orden de los 500 milisegundos para su utilización en aplicaciones de visualización y cálculos de algoritmos.

RNF 6. Soporte

- **RNF 6.1** El Servicio de Acceso a Variables 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.

2.2. Selección de Tecnologías para la implementación del Servicio de Acceso a Variables.

En la actualidad existen numerosos estándares y tecnologías para el desarrollo de aplicaciones que permiten el acceso a variables en sistemas SCADA, generalmente las implementaciones de los más utilizados se encuentran sobre tecnologías privativas como Microsoft .NET, como es el caso de OPC UA Y *Archestra*, lo que limita su aplicación en el sistema SCADA Guardián del ALBA, aunque no es prohibitivo debido a que es una manera de lograr comunicarse con sistemas propietarios que utilicen estas tecnologías. Por otra parte la solución ideal sería, crear implementaciones libres de estas arquitecturas lo que dotaría al sistema SCADA Guardián del ALBA de una interfaz para la comunicación al nivel del estado del arte.

A continuación se hace un análisis de las tecnologías mencionadas en el capítulo anterior para posteriormente hacer una selección de las tecnologías que permitirán la implementación del Servicio de Acceso a Variables del sistema SCADA Guardián del ALBA.

2.2.1. Análisis de los estándares OPC.

La inmensa mayoría de los sistemas SCADA proveen la comunicación con terceros a través de estándares mundiales, típicamente OPC. Lo ideal para un sistema SCADA como el Guardián del ALBA sería proveer esas interfaces para los servicios de acceso a variables. Las especificaciones de OPC están muy ligadas a DCOM de Microsoft lo que limita su utilización en otras plataformas, aunque pueden utilizarse pasarelas que a la vez debilitan la eficiencia y fiabilidad de la solución. El costo de implementar cada uno de los estándares OPC relacionados con el acceso a variables, como son OPC DA, OPC XML DA y OPC HDA es muy alto en tiempo de desarrollo, más cuando se necesitan otras herramientas de desarrollo que no se dominan. Adoptar estos estándares, significaría ignorar el hecho de que en la actualidad la mayoría de las aplicaciones de software que los utilizan, están migrando a gran velocidad a OPC UA y ya existen pasarelas entre OPC UA y OPC DA y está disponible para miembros de la Fundación OPC.

2.2.2. Análisis de OPC UA.

Además de todos los beneficios que introduce esta especificación ya explicados anteriormente, OPC UA, es la especificación del presente y futuro cercano. OPC UA está especificada de manera independiente a la plataforma, incluso puede implementarse como Servicios Web ya que la Fundación OPC ofrece el WSDL para esta y ya existen las pasarelas entre OPC UA y OPC DA lo que eliminaría problemas de compatibilidad con las versiones anteriores. Esta especificación describe todas las

funcionalidades que se deben exponer a los terceros y dentro de ellas el acceso a variables, permitiendo además, exponer las funcionalidades como servicios web, sin embargo, tiene como principal y gran desventaja, que es propietaria, se necesita pagar para obtener la especificación y no se puede adquirir para, de manera libre, implementar los servicios y/o funcionalidades que especifica.

2.2.3. Análisis de Servicios Web.

Los servicios web resuelven los problemas de interoperabilidad encontrados en tecnologías como CORBA, DDS, y demás, son fáciles de implementar, usar y manejar. El módulo de comunicación con terceros debe brindar, a los sistemas externos, mecanismos de acceso al sistema SCADA Guardián del ALBA, lo más universales posible, que puedan ser usados tanto por aplicaciones implementadas en Windows como en Linux. Aunque una desventaja de los servicios web es la velocidad de transmisión de la información, existen herramientas que logran mejorar considerablemente este inconveniente. Los Servicios Web permiten además, desarrollar soluciones bastante genéricas y robustas.

2.2.4. Análisis de ICE.

Es una tecnología que garantiza un rendimiento considerable en las comunicaciones inter SCADA durante el intercambio de un elevado número de variables y altos niveles seguridad en las comunicaciones. Además de todas las características expuestas, ICE permite la comunicación con distintos ORB (característica que no cumple ninguna otra tecnología de este tipo), brinda *wrapper* para varios lenguajes de programación. Es ligero, al punto de usarse en sistemas empotrados hacia los que se deberá migrar en algún momento.

2.3. Tecnologías propuestas para el desarrollo del Servicio de Acceso a Variables.

Como resultado del análisis de las tecnologías antes mencionadas, y teniendo en cuenta la opinión de varios especialistas en el tema de sistemas SCADA y tecnologías de comunicación distribuida, se seleccionaron las tecnologías ICE y Servicios Web, las que en conjunto cuentan con parámetros claves como son: facilidad de implementación, documentación disponible, software libre, posibilidad de crear soluciones universales, desarrollo de aplicaciones distribuidas, soporte y rendimiento

Se utilizará para la implementación del Servicio de Acceso a Variables, la tecnología de **Servicios Web**. Esta tecnología permite una gran interoperabilidad, ya que es independiente de la plataforma y está siendo utilizada en la actualidad como principal mecanismo para el transporte de datos entre ordenadores.

Además se implementará usando **ICE,** un pequeño subsistema dentro del servicio, el cual será utilizado por clientes interesados específicamente, por el acceso en tiempo real a las variables del sistema SCADA Guardián del ALBA. Esta tecnología posee mejores prestaciones en cuanto a transmisión de datos en tiempo real de forma natural.

De la solución planteada anteriormente se puede apreciar la similitud que presenta con respecto a OPC UA, pero esta tecnología, que es la que poseen la mayoría de los sistemas SCADA en la actualidad no puede ser empleada debido a que el sistema SCADA Guardián del ALBA no es miembro de la Fundación OPC.

Todas las afirmaciones y valoraciones acerca de las tecnologías son resultado de la elaboración y el posterior análisis de la matriz de selección de Tecnologías del Anexo 1.

2.4. Metodología, lenguajes y herramientas para el desarrollo del Servicio de Acceso a Variables.

La calidad de un producto de software es determinada en gran medida por la calidad del proceso utilizado para desarrollarlo y mantenerlo, por lo que se dedica esta sección a la selección de la metodología de desarrollo, lenguaje de programación, herramienta y lenguaje de modelado, así como la herramienta de desarrollo a utilizar en el Servicio de Acceso a Variables del sistema SCADA Guardián del ALBA.

2.4.1. RUP como metodología de desarrollo.

Con una buena metodología se pretende reducir costos y retrasos de proyectos, así como mejorar la calidad del software. La metodología RUP, fue desarrollada en 1998 por Grady Booch, Ivar Jacobson y James Rumbaugh. Se caracteriza por ser dirigida por Casos de Uso, centrada en la arquitectura, iterativa e incremental, además proporciona una visión completa de la construcción del producto. (BOOCH, 2007)

RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminarse cada una de las iteraciones. Es más apropiada para proyectos de gran envergadura, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es probable que no sea posible cubrir los costos de dedicación del equipo de profesionales necesarios. En este caso se trata de un software de grandes dimensiones como es el sistema SCADA Guardián del ALBA.

2.4.2. UML como Lenguaje de Modelado.

UML (*Unified Modeling Language*) es desde finales de 1997 un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software. (BOOCH, 2007)

Es válido destacar que UML es un lenguaje de modelado, no un método o un proceso; actualmente se publicó la versión 2.0, que proporciona a los analistas, arquitectos y desarrolladores; herramientas cada vez más potentes que les posibilita aprovechar mejor los modelos y generar así una mayor cantidad de código reduciendo en gran medida el ciclo de desarrollo de sus aplicaciones.

UML constituye un lenguaje más expresivo, claro y uniforme que los anteriores definidos para el diseño Orientado a Objetos, 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. Además UML posee una corrección fiable de errores en todas las etapas de la construcción del software y es aplicable para tratar asuntos en sistemas complejos de misión crítica, tiempo real y cliente/servidor.

2.4.3. RSA como herramienta CASE.

IBM Rational Software Architect es una herramienta de desarrollo y diseños integrados que potencia el desarrollo orientado al modelado con UML para la creación de aplicaciones y servicios con una buena arquitectura. *Rational Software Architect* es un componente del paquete *IBM Rational Profesional,* este aprovecha el Lenguaje Unificado de Modelado (UML) para el diseño de la arquitectura para C++ y Java 2 *Enterprise Edition (*J2EE) y aplicaciones de servicios web. (*Sowre Consulting* , 2009)

Con *Rational Software Architect* (RSA), se pueden convertir los diseños y diagramas UML creados en código java y C++. También soporta la iniciativa del *Object Management Group* (OMG) llamada *Model Driven Architecture* (MDA), que permite al usuario definir múltiples niveles de modelos acoplados con transformaciones de usuario entre modelos y código, resultando en una clara separación de asuntos a través del ciclo de vida del proyecto, de igual forma la utilización de patrones de diseño que están incluidos en el RSA pueden usarse para construir el contenido del proyecto de una manera mucho más rápida.

2.4.4. C++ como lenguaje de programación.

El lenguaje de programación C++ fue creado en la década de 1980 por Bjarne Stroustrup como extensión del lenguaje C. Este lenguaje es utilizado ampliamente en la industria del software. En este caso se cuenta con varias razones para seleccionarlo como lenguaje base de la solución propuesta. El conocimiento que la gran mayoría del equipo de desarrollo tiene de este lenguaje y el hecho de que la concepción del Guardián del ALBA desde su definición ha sido el desarrollo en C++ influyeron en gran medida para su selección, además los recursos con que cuenta son muy útiles en la rama de la automatización, que es la base de este producto.

Características generales

- ✓ C++ es un lenguaje híbrido.
- ✓ Es un lenguaje multiplataforma, orientado a objetos e imperativo.
- ✓ Usa tipos de datos fuertes y estáticos.
- ✓ La eficiencia en tiempo de ejecución de C/C++ está por encima de todos los lenguajes de programación de alto nivel.
- ✓ Varias implementaciones: GNU Compiler Collection, Microsoft Visual C++, Borland C++ Builder, Dev-C++ y C-Free.
- ✓ Manejo de memoria por parte del programador, lo que permite un mejor control de esta y una buena administración de recursos de computadora.
- ✓ Posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.
- ✓ C++ permite trabajar tanto a alto como a bajo nivel.

2.4.5. Eclipse como entorno de desarrollo.

Uno de los entornos de desarrollo integrado (IDE) es Eclipse, un proyecto de software libre. Eclipse es multiplataforma, fue desarrollado por IBM y en la actualidad lo mantiene la Fundación Eclipse. En sus inicios este IDE se desarrolló para los programadores que utilizaban el lenguaje Java. Actualmente emplea extensiones (plug-ins) para agregar funcionalidades en dependencia de las necesidades del

desarrollador, gracias a estas extensiones se ha extendido el soporte de Eclipse hasta lenguajes como C/C++, *Phyton*, PHP y otros.

Además permite utilizar lenguajes de procesado de texto, aplicaciones de red, Sistemas de Gestión de Bases de Datos. Brinda soporte para Sistemas de Control de Versiones e incluye extensiones para realizar pruebas de unidad. Actualmente el IDE Eclipse se encuentra en la versión 3.4 y una de sus principales ventajas es que proporciona extensiones para casi cualquier cosa, entre ellos se encuentran los de herramientas de revisión de código.

2.4.6. gSOAP como herramienta de desarrollo de Servicios Web.

Para la implementación del Servicio de Acceso a Variables se hace imprescindible una herramienta libre que permita desarrollar servicios web en C++ y que integre los estándares empleados para la implementación de estos como son: SOAP, UDDI, WSDL y componentes de seguridad.

En la investigación realizada para la selección de la herramienta de desarrollo necesaria para la implementación del Servicio Web de Acceso a Variables, se analizaron las herramientas Apache Axis, *GlassFish, NetBeans IDE, gSOAP WS Toolkit, Oracle JDeveloper y Sun J2EE 5*. En el análisis se tuvo en cuenta cada uno de los aspectos anteriormente mencionados y considerados indispensables para una correcta selección.

En la tabla que aparece a continuación se relacionan las herramientas analizadas con los requisitos con que deben contar cada una de ellas para la implementación, reflejando la ausencia del requisito con (–) y la tenencia del requisito con (X):

Herramientas	SW Libre	SOAP	WSDL	UDDI	WS-Security	C++
Apache Axis	X	X	X	--	X	X
GlassFish	X	X	X	X	X	–
NetBeans IDE	X	X	X	--	–	–
gSOAP WS Toolkit	X	X	X	X	X	X
Oracle JDeveloper	--	X	X	X	X	–
Sun J2EE 5	--	X	X	X	X	–

Figura 6: Comparación entre diferentes herramientas que permiten el desarrollo de Servicios Web.

De los datos reflejados en la tabla, se obtiene como conclusión que la herramienta “gSOAP WS Toolkit”, es la única que cuenta con todas las características necesarias que se requieren en la

implementación del Servicio de Acceso a Variables. Aunque en la tabla anterior no se reflejan todas las herramientas existentes en la actualidad, en la bibliografía consultada, aparece esta herramienta como la elección clara para la implementación de servicios web en C/C++ y sobre software libre.

Entrando en detalles se puede decir que la herramienta gSOAP (versión 2.7.9) permite desarrollar servicios web con C y C++. Se utiliza principalmente para desarrollar servicios web a partir de su descripción en WSDL. Entre las herramientas disponibles se incluye un generador de código fuente que realiza parte del trabajo de codificación e incorpora un analizador de WSDL y de esquemas XML capaz de asociar automáticamente los tipos que aparecen en los esquemas con tipos de datos de C y C++. (El reto de los servicios Web para el software libre, 2008)

Características de gSOAP

- Permite trabajar con SOAP, WSDL, UDDI y con otras tecnologías como WS-Addressing y WS-Security.
- Se distribuye con tres tipos de licencia: GNU GPL, código abierto público gSOAP y «comercial».
- Es multiplataforma.

2.4.6.1. gSOAP y MTOM.

Una de las desventajas que tienen los Servicios Web es la velocidad de transmisión de los datos, gSOAP utiliza para la solución de este problema **MTOM** *(Message Transmission Optimization Mechanism)* por sus siglas en inglés. Este mecanismo de optimización para la transmisión de mensajes especifica y optimiza el formato de la transmisión de datos binarios a través de los mensajes SOAP. Es muy utilizado para lograr mayor eficiencia en la codificación de la información binaria que puede venir insertada en un XML, como por ejemplo arreglos, imágenes, documentos.

2.4.6.2. ¿Quiénes usan gSOAP?

Dentro de las empresas que utilizan esta herramienta se encuentran: *Adobe Systems*, AOL, BEA, *Boeing, Cisco Systems,* CNR, eBay, Ericsson, Exxon/Mobile, HP, IBM, *Intel, Microsoft, Nokia, Pfizer, Siemens, WindRiver, Xerox*, y muchos otros.

2.5. Integración del Subsistema de Comunicación con Terceros con el Guardián del ALBA.

El sistema SCADA Guardián del ALBA como caso particular de sistemas SCADA, está dividido de forma general, en varios subsistemas o módulos que posibilitan la escalabilidad de la solución, como se

muestra en la figura 7. El módulo de Comunicación con Terceros se comunica directamente con el subsistema *middleware*, que se encarga de la gestión de las comunicaciones entre los diferentes subsistemas del sistema SCADA Guardián del ALBA.

Para la versión 2.0 del Guardián del ALBA, el módulo de Comunicación con Terceros debe brindar a sistemas externos los servicios de intercambio de información relacionada con variables, intercambio de información relacionada con alarmas y eventos, interacción a través de comandos y servicio de seguridad.

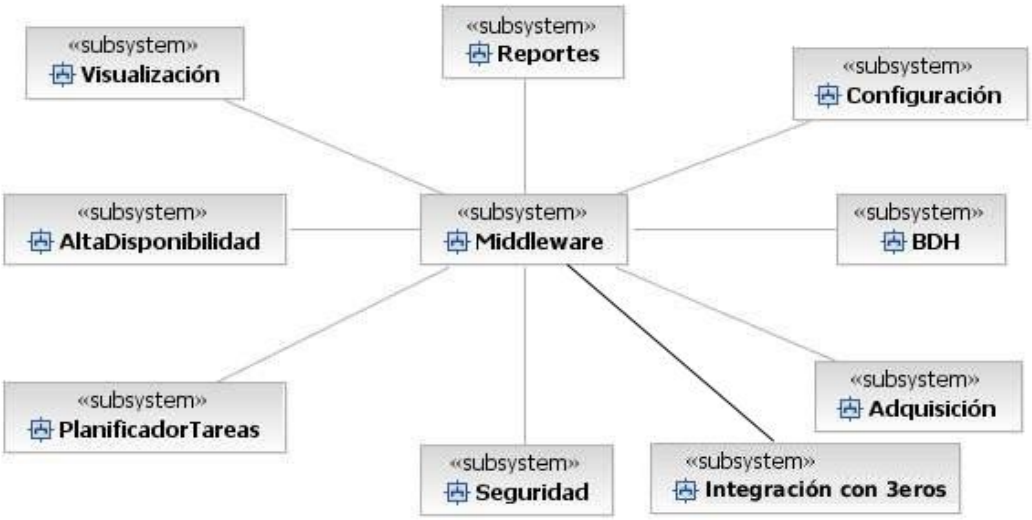

Figura 7: Arquitectura del sistema SCADA Guardián del ALBA.

Por la complejidad que supone el desarrollo de cada uno de estos servicios, y teniendo en cuenta el campo de acción de este trabajo, se hará referencia solamente al Servicio de Acceso a Variables.

En un primer intento de tratar que el sistema SCADA Guardián del ALBA pudiera intercambiar variables con agentes externos, se diseñó una pasarela (Gateway OPC DA) en la cual el estándar utilizado fue OPC con la especificación Data Access (DA) v2.05a., teniendo esta solución numerosas restricciones de portabilidad debido a que la capa de comunicación OPC es dependiente de la tecnología COM/DCOM de Microsoft.

Estos problemas de portabilidad serán solucionados implementando el Servicio de Acceso a Variables utilizando la combinación de las tecnología ICE y Servicios Web, justificadas anteriormente.

2.6. Subsistema de Comunicación con Terceros.

El gráfico que se muestra a continuación esquematiza el subsistema de Comunicación con Terceros, compuesto por un grupo de paquetes de los cuales se dará una descripción en detalle en esta sección.

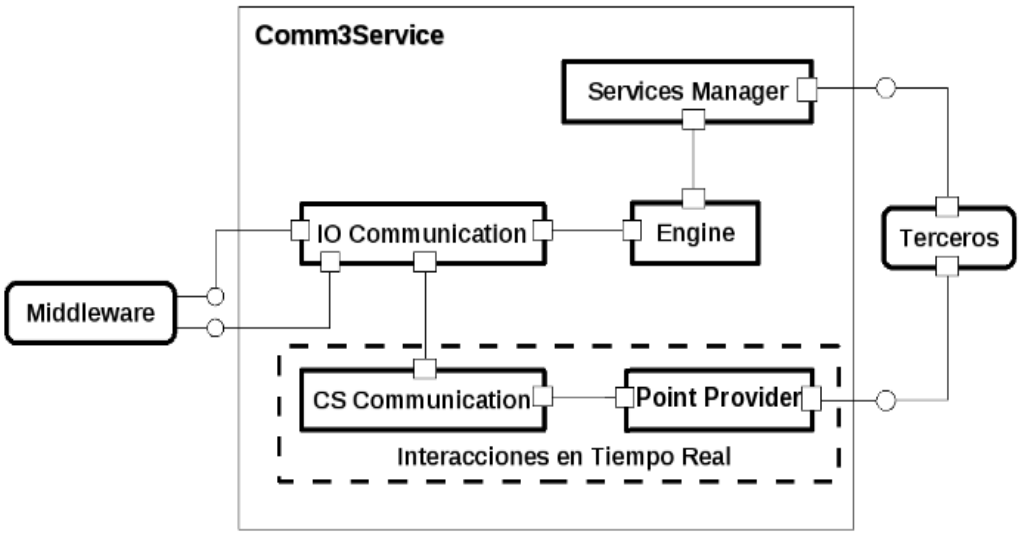

Figura 8: Subsistema de Comunicación con Terceros.

Terceros: Hace referencia a toda aquella aplicación externa incluyendo otros sistemas SCADA que requieran intercambiar información con el sistema SCADA Guardián del ALBA. Estas aplicaciones pueden hacer uso de cualquiera de las interfaces de comunicación expuestas, a través del *Services Manager* o del *PointProvider*.

***Services Manager*:** Encargado de exponer las funcionalidades para el acceso a puntos, alarmas, eventos e interacción a través de comandos con los terceros. Además de la invocación de los servicios, permite a los *Engine* registrar los servicios y a los terceros conocer los servicios y sus características haciendo uso de las descripciones de los mismos.

***Engine*:** Encargado de manejar las suscripciones y accesos a las variables, alarmas y eventos así como las invocaciones de comandos realizadas desde las aplicaciones externas a través del *Services Manager* siempre dependiente de los mecanismos de seguridad que el propio módulo implemente.

***IO Communication*:** Recibe las peticiones del *Engine y CSCommunication*, las que satisface haciendo pedidos de información al SCADA. Para lograr lo antes expuesto utiliza las interfaces del *middleware* a fin de recibir puntos, alarmas, eventos y para enviar comandos.

***PointProvider*:** Provee una interfaz de comunicación de alto nivel que permite a los terceros solicitar y recibir información de variables en tiempo real y de forma segura.

CSCommunication: Subsistema que encapsula la lógica de negocio del Servicio de Acceso a Variables para interacciones en tiempo real. Usa el subsistema *IO Communication* para pedir los datos del sistema SCADA Guardián del ALBA utilizando las interfaces que provee el *middleware* para interacciones de este tipo.

Middleware: Representa al *middleware* del sistema SCADA Guardián del ALBA, se refiere a las conexiones con los servicios del *middleware* y las interfaces que este provee para el acceso a los datos.

2.7. Modelo de diseño del Subsistema de Comunicación con Terceros.

El modelo de diseño está dividido en tres capas donde cada capa tiene su responsabilidad bien definida. A continuación se explica la disposición por capas y sus responsabilidades:

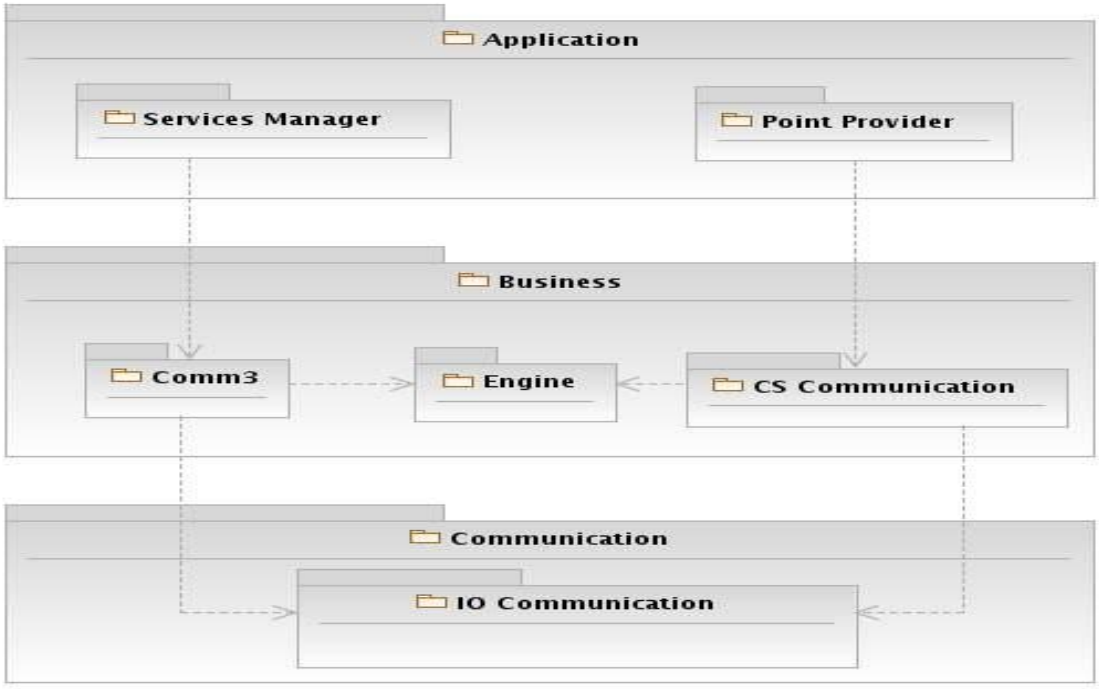

Figura 9: Vista lógica de la arquitectura del Subsistema de Comunicación con Terceros.

A continuación se explica la disposición por capas y sus responsabilidades:

***Application* (Capa de presentación o aplicación):** Esta capa tiene la responsabilidad de exponer las interfaces de comunicación que van a permitir a las aplicaciones externas solicitar información de puntos, alarmas y eventos y la interacción a través de comandos. Las responsabilidades de esta capa están divididas en paquetes. El paquete *Services Manager* expone los servicios web y *PointProvider* que expone las interfaces de comunicación de alto nivel para interacciones en tiempo real usando ICE como protocolo de comunicación. *Application* depende de la capa de negocio para dar respuesta las peticiones de los clientes.

***Business* (Capa de Negocio):** Esta capa tiene la responsabilidad de resolver las peticiones de los terceros invocadas desde la capa superior, define la lógica de negocio relacionada con las suscripciones, acceso a la información del sistema SCADA Guardián del ALBA, invocación de comandos y los mecanismos de seguridad. También abstrae el paradigma de comunicación de las tecnologías utilizadas. Las responsabilidades están divididas en las clases contenidas en varios paquetes, entre ellos: el paquete *Comm3* que encapsula las clases que definen la lógica para inicializar el servidor de *Comm3* que incluye inicializar la comunicación vía SOAP, los servicios y la comunicación con el *middleware*; el paquete *Engine*, que encapsula las funcionalidades relacionadas con la suscripción y acceso a la información del sistema SCADA Guardián del ALBA y el paquete *CSCommunication* que abstrae el paradigma cliente/servidor de ICE.

***Communication* (Capa de Comunicación):** Esta capa tiene como principal responsabilidad realizar las peticiones a los módulos del sistema SCADA Guardián del ALBA a través de las interfaces de comunicación del *middleware*. Las responsabilidades están divididas en los paquetes *IO Communication*, encargado de gestionar las comunicaciones a través del *middleware*. Las responsabilidades están bien definidas en las clases del paquete *IO Communication*.

2.8. Arquitectura en tres capas.

La arquitectura tres capas posibilita que el mantenimiento y mejoras a la solución que se proponga sea más fácil de manejar, debido al bajo acoplamiento y alta cohesión entre las capas. Al implementar esta arquitectura se obtiene escalabilidad, tolerancia a fallos y un mejor rendimiento de la solución

propuesta. El uso de capas ayuda a controlar y encapsular la complejidad del Servicio de Acceso a Variables.

La arquitectura adoptada permite obtener una solución de fácil reutilización por otros SCADAS o DCS que tengan la necesidad de contar con una solución de comunicación con aplicaciones externas. Además, posibilita el trabajo colaborativo, es decir, diferentes personas pueden trabajar en el desarrollo u optimización de capas diferentes con una necesidad mínima de comunicación entre ellos debido a que las interacciones de la capa *Application* con la capa *Business*, y la de la capa *Business* con la capa *Communication* es mínima y se realiza a través de interfaces bien definidas.

2.9. Paquetes de diseño arquitectónicamente significativos.

A continuación se muestran los diagramas de clases del diseño de los paquetes relacionados específicamente con el Servicio de Acceso a Variables del Subsistema de Comunicación con Terceros.

2.9.1. Paquete *Services Manager*.

Este paquete contiene las clases que definen la lógica para registrar y exponer las funcionalidades del sistema SCADA Guardián del ALBA como servicios web, permitiendo a los terceros conocer los servicios expuestos y sus descripciones, y al servidor de Comunicación con Terceros registrar los servicios que implementa.

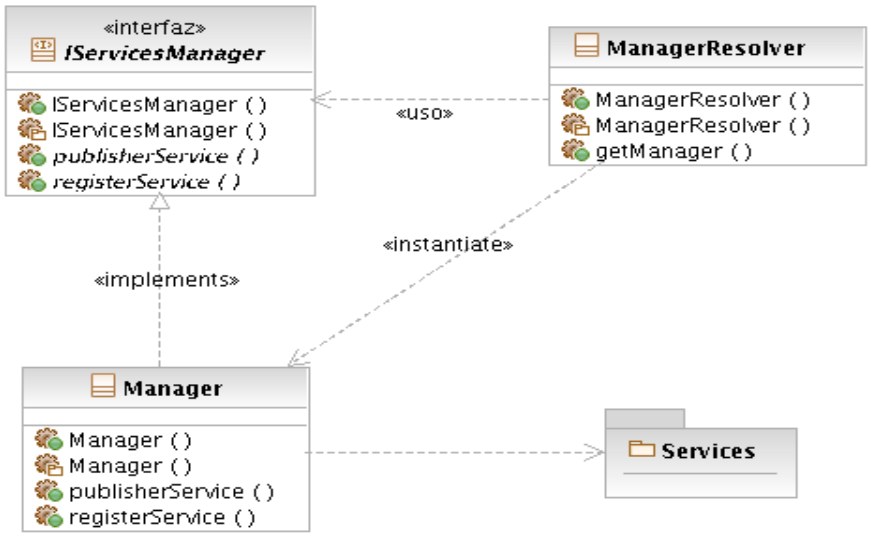

Figura 10: Diagrama de clases del paquete Service Manager.

2.9.1.1. Descripción de clases y métodos del paquete *Services Manager*.

Descripción de la clase Manager Resolver	
Elemento	**Descripción**
ManagerResolver	Esta clase juega el rol de una fábrica de manejadores de servicios (*ServicesManager*). Permite solicitar un objeto de la interfaz *IServicesManager* para hacer solicitudes de publicación y registro de servicios. Como *IServicesManager* es una interfaz pura con métodos virtuales puros sin implementación, la fábrica devuelve un objeto del tipo de la interfaz pero lo que lleva dentro es un hijo de esta, un Manager. Por eso cuando se invoca un método, por polimorfismo se logra ejecutar los métodos de la implementación, en este caso, los métodos definidos en Manager. *Se basa en el patrón "factory method".*
Métodos	
getManager():IServicesManager	Este método devuelve un apuntador a la interfaz *IServicesManager* y en su implementación crea un objeto del tipo Manager y retorna el padre, es decir, un *IServicesManager*.

Tabla 2: Descripción de la clase *ManagerResolver*

Descripción de la Interfaz *IServicesManager*	
Elemento	**Descripción**
IServicesManager	Esta interfaz es una fachada del subsistema *ServiceManager* que expone las funcionalidades del mismo permitiendo invocar métodos para publicar y registrar los servicios que implementa el servidor de Comm3.
Métodos	

publicateService():bool	Este método debe ser invocado cuando se desea publicar un servicio. Es virtual puro, por polimorfismo se ejecuta el método *publicateService()* de la clase Manager.
registerService():bool	Este método debe ser invocado cuando se desea registrar un nuevo servicio implementado por el servidor de Comm3. Es virtual puro sin implementación, por polimorfismo se ejecuta el método *registerService()* de la clase *Manager*.

Tabla 3: Descripción de la interfaz *IServicesManager*

Descripción de la clase Manager	
Elemento	**Descripción**
Manager	Esta clase implementa la interfaz *IServicesManager*, define la lógica para publicar y registrar servicios.
	Métodos
publicateService():bool	Este método define la lógica para publicar un servicio y para ello utiliza los servicios definidos en el paquete *Services*. El mismo se ejecuta cuando es invocado el método *publicateService()* desde la interfaz *IServicesManager.*
registerService():bool	Este método define la lógica para registrar un servicio y para ello utiliza los servicios definidos en el paquete *Services*. El mismo se ejecuta cuando es invocado el método *registerService()* desde la interfaz *IServicesManager*.

Tabla 4: Descripción de la clase Manager

En la figura 10 se puede observar que el paquete *ServiceManager* contiene a su vez un paquete denominado *Service*. Este paquete contiene las clases que definen la lógica de descripción de los servicios web en C++, y sus operaciones. A continuación se presenta a manera de diagrama de clases la descripción del Servicio de Acceso a Variables que será utilizada por los terceros para la implementación de sus clientes.

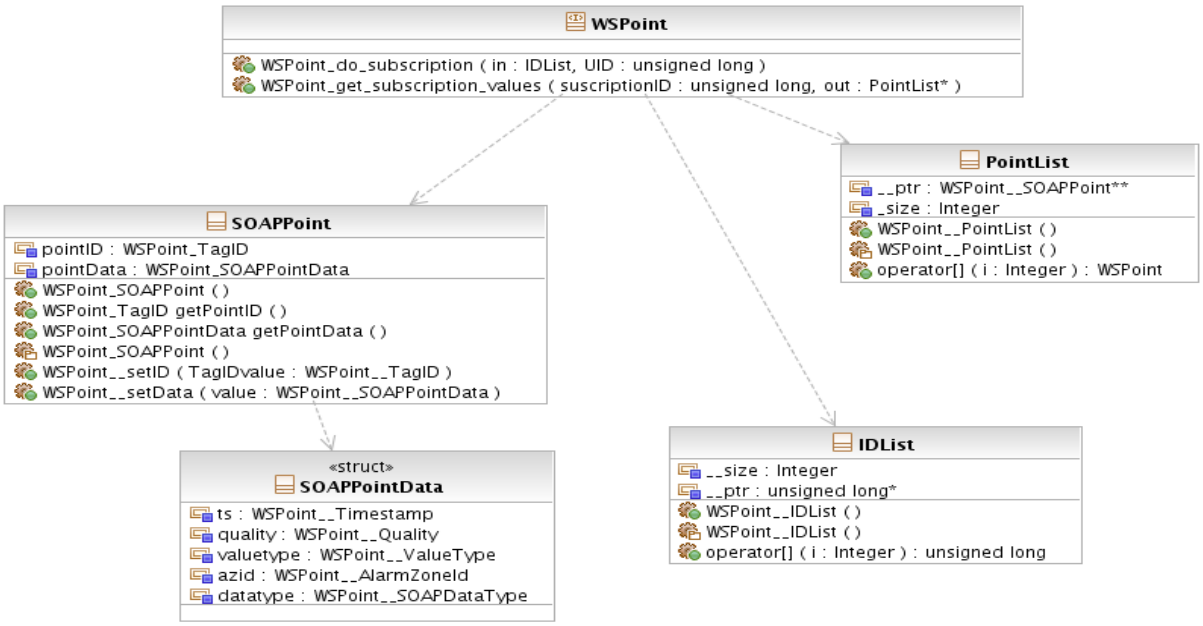

Figura 11: Diagrama de clases de la descripción del Servicio de Acceso a Variables.

Descripción de la clase *WSPoint*	
Elemento	**Descripción**
WSPoint	Esta clase describe el servicio web para acceso a variables del Guardián del ALBA, y las operaciones soportadas por él.
Métodos	
WSPoint_do_subscription()	En la descripción del servicio web este es el método definido para que los terceros puedan invocar la suscripción de una lista de puntos en el Servicio de Acceso a Variables.
WSPoint_get_subscription_values()	En la descripción del servicio este es el método definido para que los terceros puedan solicitar los valores de los puntos publicados por el Guardián del ALBA.

Tabla 5: Descripción de la clase *WSPoints*

Descripción de la clase *SOAPPoint*

Elemento	Descripción
SOAPPoint	Esta clase contiene la definición del punto que utilizará el sistema que se comunicará con el Servicio de Acceso a Variables del Guardián del ALBA.
	Atributos
pointID	Identificador del punto.
pointData	Estructura que contiene los demás atributos del punto, como son: *TimeStam, Quality, AlarmZoneID, ValueType y DataType.*
	Métodos
SOAPPoint()	Constructor de la clase.
~ SOAPPoint()	Destructor de la clase.
setID()	Método para cambiar el identificador del punto.
getId():unsigned long	Método para obtener el identificador del punto.
setData()	Método que permite cambiar los atributos del punto.
getData():SOAPPointData	Método para obtener los atributos del punto.

Tabla 6: Descripción de la clase SOAPPoint.

Los elementos que aparecen en el diagrama de clases y no son descritos son estructuras de datos utilizados en la definición de la interfaz para el Acceso a Variables. *SOAPPointData* contiene una estructura que almacena todos los atributos del punto, excepto su identificador. PointList e IDList son clases creadas con el objetivo de contener las listas de puntos y listas de identificadores respectivamente, necesarias para los métodos de la clase *WSPoint*.

2.9.2. Paquete *Comm3.*

Este paquete contiene las clases que definen la lógica para iniciar el Subsistema de Comunicación con Terceros, estas clases van a implementar la lógica que permite conectarse al *middleware* del sistema SCADA Guardián del ALBA, iniciar la comunicación mediante servicios web para escuchar las peticiones de los terceros. También contiene las clases definen la lógica de implementación de los servicios que puede invocar el cliente. Además usa el paquete *Engine* para resolver las peticiones de suscripción y

acceso a la información. Este paquete es el punto de entrada al Subsistema de Comunicación con Terceros siendo el *main()* de la aplicación.

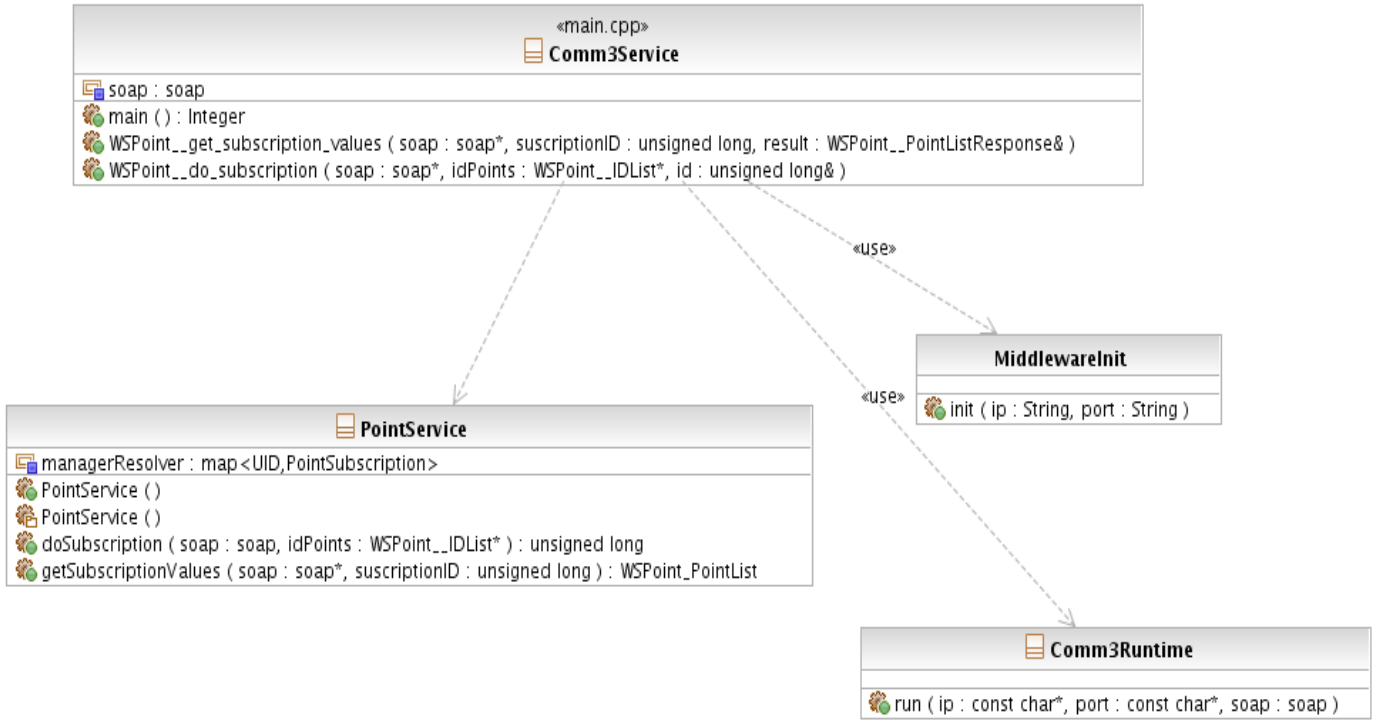

Figura 12: Diagrama de clase del paquete Comm3.

2.9.2.1. Descripción de clases y métodos del paquete *Comm3.*

Descripción de la clase *Comm3Service*	
Elemento	**Descripción**
Comm3Service	Este elemento es el *main()* de la aplicación, se modela como una clase con el estereotipo de <<*main.cpp*>>. Tiene la responsabilidad de iniciar la comunicación con el *middleware* e iniciar la comunicación mediante servicios web para garantizar las comunicaciones entre los terceros y el servidor de Comm3 a través de los servicios web.
Métodos	
main()	1. Define el ip y el puerto de conexión al *middleware* e invoca el método *init()* de la clase *MiddlewareInit*.

	2. Inicia la comunicación mediante servicios web invocando el método *run()* de la clase *Comm3Runtime*.
WSPoint__do_Subscription()	Este método hace una llamada a la función ***doSubscription*** de la clase *PointService* que implementa el método de suscripción a las variables del Guardián del ALBA.
WSPoint__getSubscription Values()	Este método hace una llamada a la función ***getSubscriptionValues*** de la clase *PointService* que implementa el método de obtención de los valores de la suscripción.

Tabla 7: Descripción de la clase Comm3Service

Descripción de la clase *MiddlewareInit*	
Elemento	**Descripción**
MiddlewareInit	Esta clase tiene la responsabilidad de iniciar la comunicación del Subsistema de Comunicación con Terceros con el *middleware* del Guardián del ALBA usando el paquete *IO Communication.*
Métodos	
init(ip: String, port: String)	Este método es invocado desde el *main()* de la aplicación identificando el ip y puerto por donde se está ejecutando el *middleware*. Define la lógica para que el Subsistema de Comunicación con Terceros se conecte a los servicios del *middleware*.

Tabla 8: Descripción de la clase *Comm3Service*

Descripción de la clase *Comm3Runtime*	
Elemento	**Descripción**
Comm3Runtime	Esta clase tiene la responsabilidad de iniciar las comunicaciones mediante servicios web y permitir de esta manera que el servidor de Comm3 reciba las

	peticiones realizadas, garantizando una comunicación cliente-servidor a través del protocolo SOAP.
	Métodos
run()	Este método es invocado desde el *main()* de la aplicación en el mismo instante en que se levanta el subsistema e inmediatamente después de establecer los parámetros de conexión con el *middleware*. En la lógica se define la inicialización de todos los parámetros y variables necesarios para establecer la comunicación con los clientes mediante los servicios web.

Tabla 9: Descripción de la clase Comm3Runtime

Descripción de la clase *PointService*	
Elemento	**Descripción**
PointService	Esta clase tiene la responsabilidad de definir la lógica de implementación del Servicio de Acceso a Variables descrito en la clase WSPoint.
	Métodos
doSubscription()	Este método define la lógica de implementación del método de suscripción a las variables (puntos) del servicio, y utiliza las clases definidas en el paquete *PointEngine* para registrar la suscripción y entregar al cliente el identificador de esta.
getSubscriptionValues()	Este método define la lógica de implementación del método de lectura de valores de las variables del servicio y utiliza el paquete *PointEngine* para entregar al cliente los valores de los puntos solicitados en la suscripción.

Tabla 10: Descripción de la clase *PointService*

2.9.2.2. Diagramas de Secuencia del paquete Comm3.

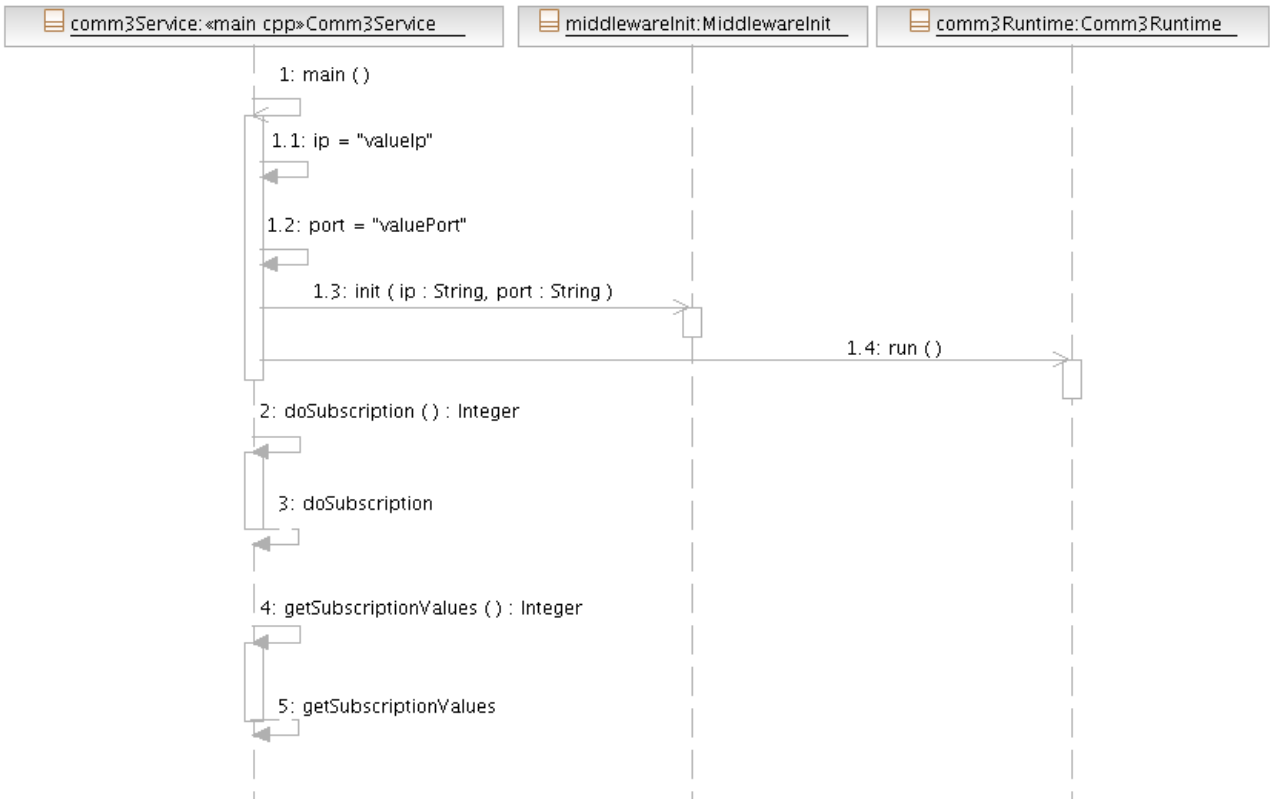

Figura 13: Diagrama de secuencia Iniciar Servidor de Comm3.

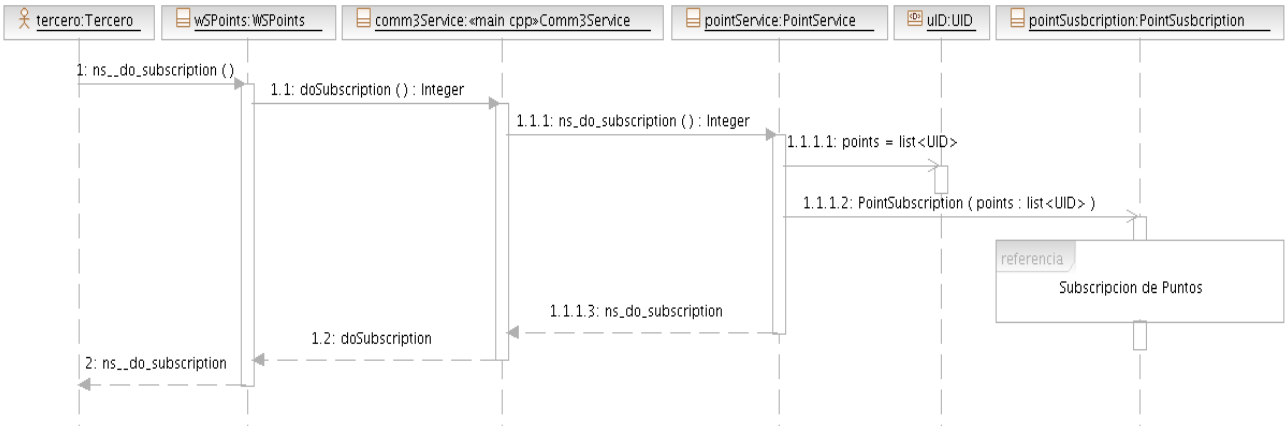

Figura 14: Diagrama de secuencia Solicitar suscripción a puntos.

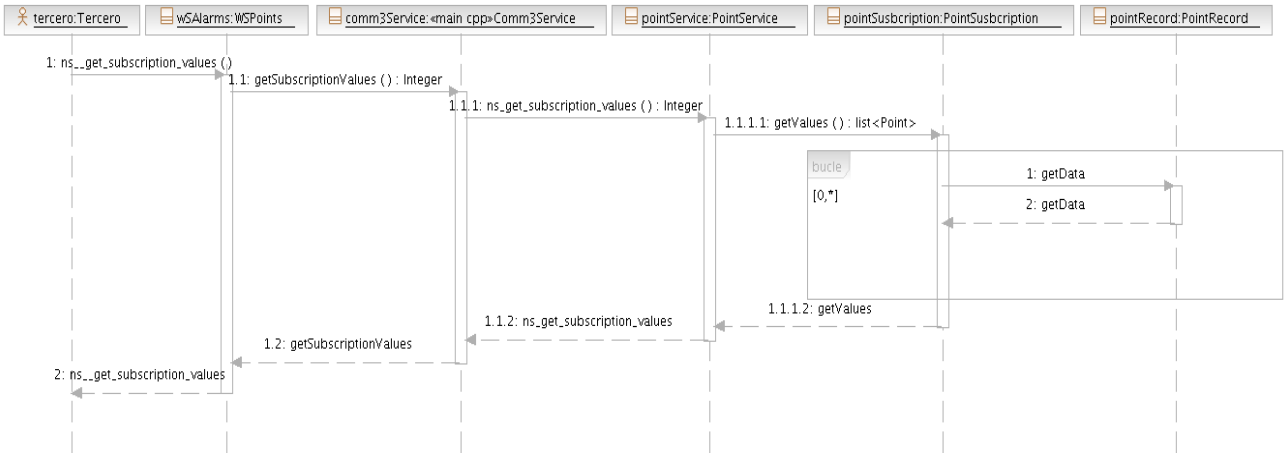

Figura 15: Diagrama de secuencia Solicitar valores de puntos.

2.9.3 Paquete *Engine.*

En este paquete se encuentran las clases que definen la lógica para la suscripción y acceso a los datos. Estas clases resuelven los pedidos de suscripciones y acceso a la información del sistema SCADA Guardián del ALBA que se realicen a través de los servicios brindados por el Subsistema de Comunicación con Terceros. Dicho paquete está estructurado por un grupo de paquetes con vistas a obtener una mejor organización del sistema y de los elementos que lo componen.

Solamente se describirá en profundidad el paquete *PointEngine*, pues en él es donde se implementa toda la lógica del Servicio de Acceso a Variables, que constituye el campo de acción del actual trabajo.

Si bien publicar una serie de servicios accesibles a todo el mundo supone una clara ventaja de integración entre sistemas, también puede convertirse en un problema de seguridad. El objetivo principal del servicio de seguridad, contenido en el paquete *SecurityEngine* es proteger adecuadamente los servicios expuestos por el Guardián del ALBA y en específico el Servicio de Acceso a Variables lo que permite:

- Asegurar la autenticación mutua entre el consumidor que accede al servicio web de acceso a variables y el proveedor de dicho servicio.
- Ofrecer la posibilidad de que un consumidor se identifique una sola vez y pueda acceder a varias funcionalidades del servicio, sin tener que identificarse nuevamente en cada uno de ellos.
- Garantizar la confidencialidad e integridad de los datos, protegiéndolos frente a alteraciones fortuitas o deliberadas.

Todo esto se logra debido a que este servicio utiliza *WS-Security* (Seguridad en Servicios Web) que es un protocolo de comunicaciones que suministra un medio para aplicar seguridad a los Servicios Web. Como se muestra en la figura 17 todos los paquetes se relacionan con *SecurityEngine*, obteniendo de este toda la lógica de seguridad.

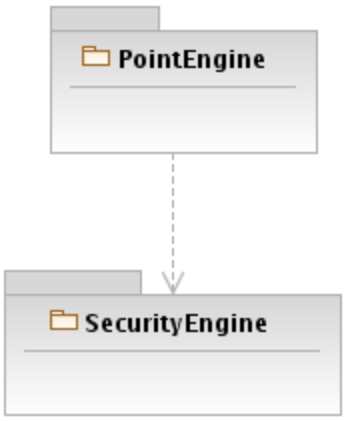

Figura 16: Relación de paquetes del *Engine* con aportes significativos para el Servicio de Acceso a Variables.

2.9.3.1 Paquete *PointEngine*.

En este paquete se encuentran las clases que definen la lógica de suscripción y acceso a las variables del sistema. Estas son las clases que se implementarán para resolver las suscripciones y el acceso a la información del sistema SCADA Guardián del ALBA relacionada con las variables después de establecer una comunicación mediante servicios web entre el cliente (tercero) del servicio y el servidor de Comm3.

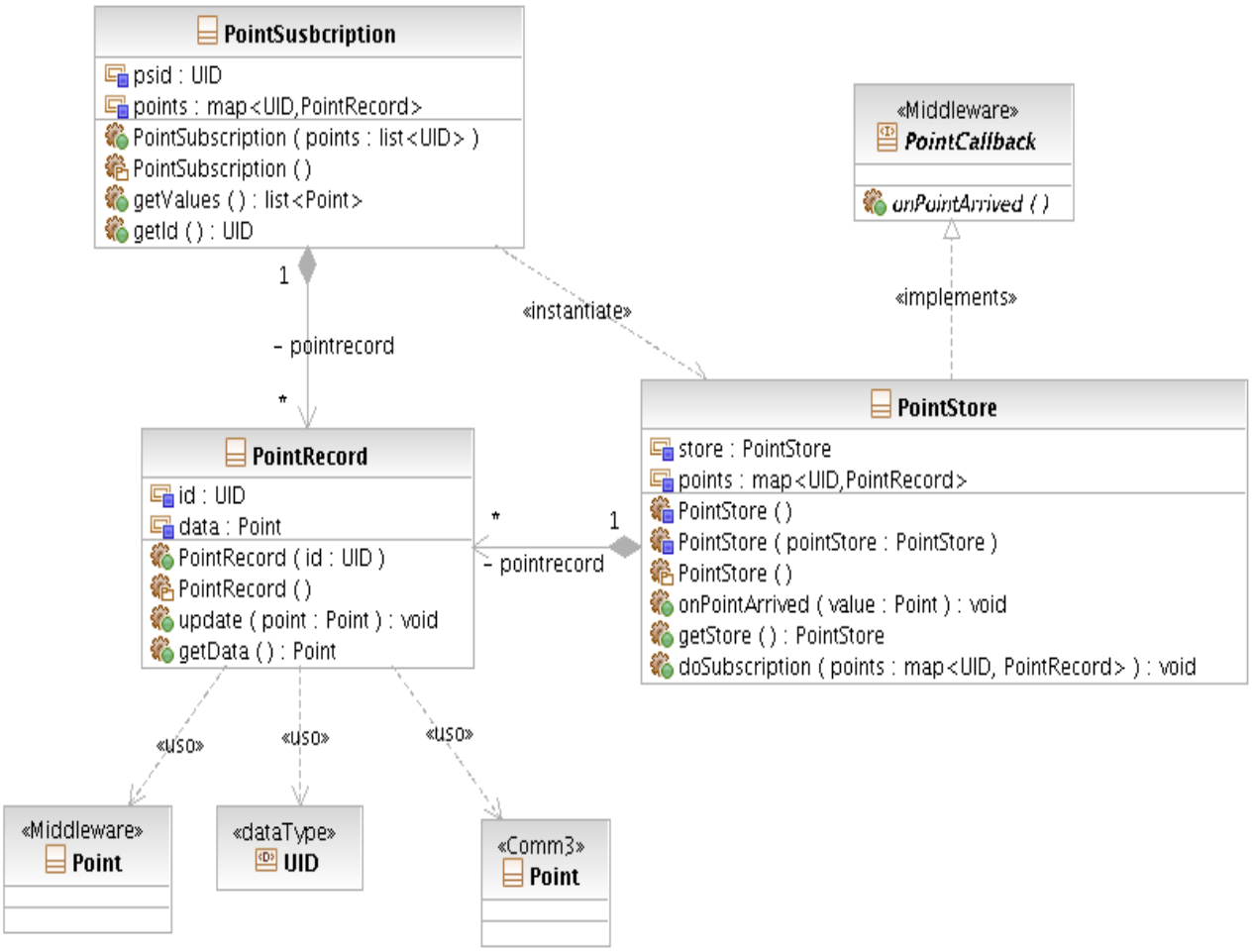

Figura 17: Diagrama de Clases del Paquete *PointEngine* que encapsula el Servicio de Acceso a Variables del Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA.

2.9.3.1.1 Descripción de clases y métodos del paquete *Point Engine.*

Descripción de la clase *PointSubscription*	
Elemento	**Descripción**
PointSubscription	Esta clase recibe peticiones de suscripción desde la clase *PointService* que implementa el método de suscripción del servicio web para puntos. *PointSubscription* tiene la responsabilidad de registrar la suscripción usando la clase *PointStore*, entregando posteriormente a la clase *PointService*

	el identificador de la suscripción que asignó el sistema al cliente y se encarga además de entregar los valores actualizados de las variables cada una cierta frecuencia usando la clase *PointRecord*.
Atributos	
subscriptionID	Identificador de la Suscripción.
mypoints	Arreglo asociativo que almacena los valores de las suscripciones.
Métodos	
PointSubscription(std::list<UID> points)	Este es el constructor de la clase y define la lógica para registrar la suscripción a puntos según la lista de Id pasada por parámetros. Para ello pide al *Store* la lista de *Record* existentes, crea un nuevo record con la lista de Id y se los pasa al *Store* a través del método *doSubscription()* para que este se encargue de culminar la suscripción. *PointSubscription()* asigna un Id de suscripción al cliente.
~PointSubscription()	Destructor de la clase.
getValues(): std::list<UID> points	Este método permite a la clase *PointService* pedir la lista de valores actualizados de las variables. En la lógica del método se piden los record de puntos actualizados en *PointRecord* usando el método *getData()*.
getId():UID	Este método permite a la clase *PointService* obtener el identificador de suscripción que fue asignado al cliente.

Tabla 11: Descripción de la clase *PointSubscription*.

Descripción de la clase *PointRecord*	
Elemento	**Descripción**

PointRecord	Esta clase tiene la responsabilidad de almacenar y mantener actualizada la lista de puntos por suscripción.
Atributos	
id	Identificador del punto almacenado.
_data	Punto obtenido del *middleware*.
Métodos	
PointRecord(UID id)	Constructor de la clase.
~PointRecord()	Destructor de la clase.
update(point:Point)	Este método verifica que no exista el punto que se pasa por parámetro asignándole el valor al atributo data de la clase y creando un nuevo punto. De de lo contrario se elimina el punto que se pasa por parámetros.
getData():Point	Este método devuelve un punto determinado. Es usado por la clase *PointSubscription* para conformar la lista de puntos para una suscripción y así entregar los datos a los clientes.

Tabla 12: Descripción de la clase *PointRecord*

Descripción de la clase PointStore	
Elemento	**Descripción**
PointStore	Esta clase tiene la responsabilidad de verificar en *PointRecord* la existencia de los puntos solicitados a través de una suscripción, de no existir, actualiza la lista de los record para así empezar a recibir los valores de las variables existentes y las nuevas solicitadas. También tiene la responsabilidad de actualizar los valores de las variables guardadas en los record

	de suscripciones a partir de recibir las variables desde el *middleware*. Hereda de la interfaz *PointCallback*, re-implementando el método *onPointArrived()* y de esta manera puede decidir que hacer cada vez que arribe un punto desde el *middleware*. Esta clase se define usando el patrón *Singleton*.
Atributos	
store	Objeto estático de la clase PointStore.
points	Arreglo asociativo que almacena los valores de los puntos que arriban del *middleware*.
Métodos	
PointStore()	Constructor de la clase
~PointStore()	Destructor de la clase.
onPointArrived(point: Point):void	Este método es la implementación de *onPointArrived()* de la interfaz *PointCallback* que brinda el *middleware* para escuchar los puntos que son publicados por el Guardián del ALBA, permitiendo ejecutar un *callback* cada vez que arriba un punto. La lógica del método está orientada a verificar que el Id del punto que llegó existe en la lista de *PointRecord* y actualizar el valor existente con el nuevo valor.
getStore():PointStore	Este método retorna una instancia de la propia clase que será usada en el momento de decirle al *middleware* que desea recibir puntos especificando el *callback* (la función *getStore()*) que se va a ejecutar cuando llegue un punto. Esta llamada se usa en el método *receivePoint()* de la clase *MiddlewareIO*.
doSubscription(points: map<UID, PointRecord>)	Este método define la lógica para verificar en el registro de suscripciones de puntos la existencia de los puntos pedidos en una suscripción nueva y así saber que puntos pedir al *middleware*. De no existir los puntos se hace un nuevo registro y empieza a escuchar nuevos valores de puntos además de

los ya registrados.

Tabla 13: Descripción de la clase *PointStore*

Descripción de la clase *Point*	
Elemento	**Descripción**
Point	Punto manipulado en el Subsistema de Comunicación con Terceros. Esta clase representa el punto que maneja el sistema de comunicación con terceros, de interfaz lo más sencilla posible.
Atributos	
pointID	Identificador del punto.
timestamp	Estampa de tiempo del punto.
quality	Calidad del punto.
alarmZoneId	Zona de alarma.
value	Valor del punto.
valuetype	Tipo de valor.
Métodos	
Point(UID pid)	Constructor de la clase, tiene la responsabilidad de crear puntos, pasándole el identificador del punto que se quiere crear por parámetro.
~Point()	Destructor de la clase.

Tabla 14: Descripción de la clase Point

2.9.3.1.2 Diagramas de Secuencia del paquete Point *Engine.*

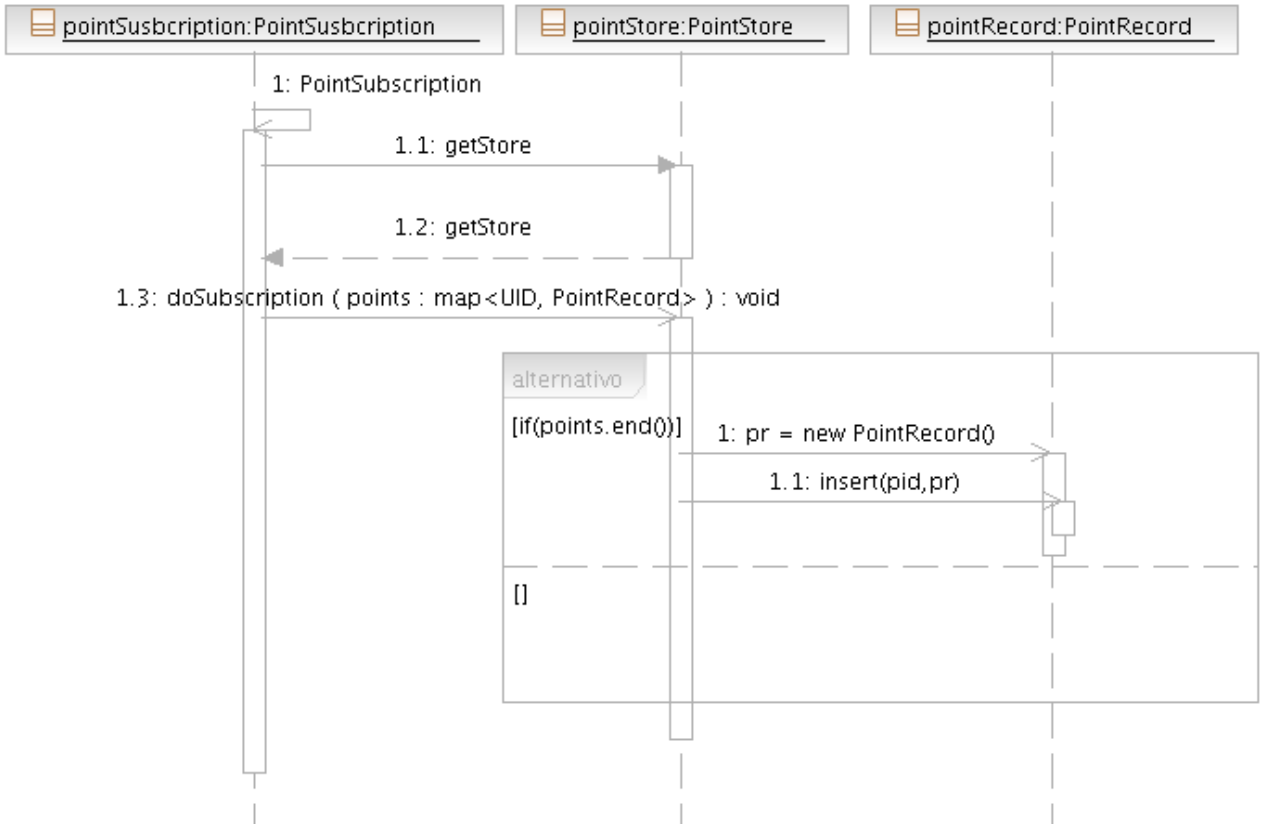

Figura 18: Diagrama de secuencia Suscripción de Puntos.

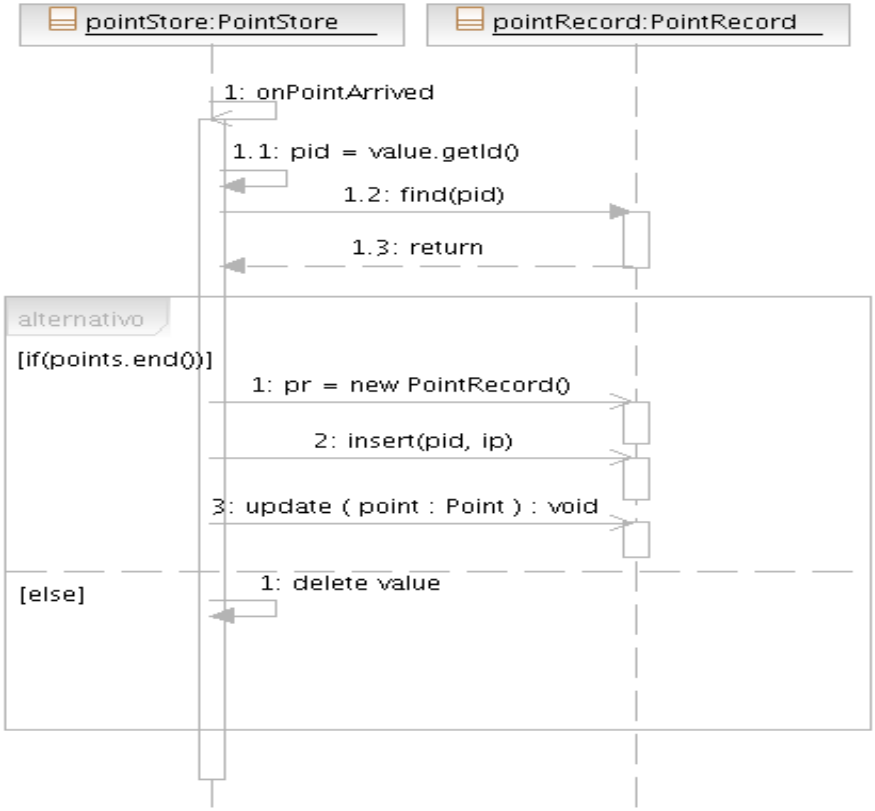

Figura 19: Diagrama de secuencia Capturar y Registrar puntos.

2.9.4 Paquete *IO Communication.*

Este paquete contiene las clases que definen la lógica para solicitar a través del *middleware* los datos relacionados con puntos. También define la lógica para realizar invocaciones de comandos y peticiones al módulo de seguridad. Todo se logra a partir de usar las interfaces del *middleware* y especificar el tipo de dato que se desea escuchar, así como el canal de comunicación.

La interfaz *InputPointManager* que se muestran en la figura 20 es la interfaz del subsistema *middleware* que se utiliza para la recepción de puntos entre otros datos. *La interfaz InputPointManager, InputAlarmManager e InputEventManager* no son clases definidas en el Subsistema de Comunicación con Terceros ni se incluyen en su diseño.

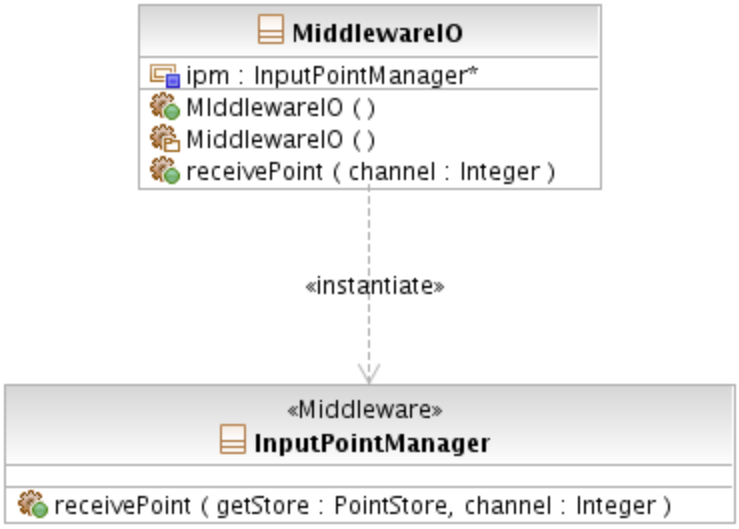

Figura 20: Diagrama de Clase del Paquete *IO Communication.*

2.9.4.1 Descripción de clases y métodos del paquete *IO Communication.*

Descripción de la clase *MiddlewareIO*	
Elemento	**Descripción**
MiddlewareIO	Esta clase tiene como responsabilidad de establecer la comunicación con el *middleware* a partir de una solicitud realizada desde la clase *MiddlewareInit* del paquete *Comm3* en el momento de iniciar (ejecutar) el servidor de comunicación con terceros. También le especifica al *middleware* la información que desea obtener del Guardián del ALBA. Se puede observar que en sus atributos contiene una instancia de cada una de las interfaces del *middleware* que va a utilizar.
Métodos	
receivePoint(callback: PointCallback)	Este método da a conocer al *middleware* que se quiere recibir puntos y lo hace a través de la interfaz *callback*. El *callback* es lo que se ejecuta cuando llega un punto, en este caso el *callback* que se pasaría sería un *PointStore* que implementa la interfaz *PointCallback* e implementa el método *onPointArrived()* que define las operaciones que va a realizar el *PointStore* cuando llegue un punto.

Tabla 15: Descripción de la clase *MiddlewareIO*

2.9.4.2 Diagrama de Secuencia del paquete *IO Communication.*

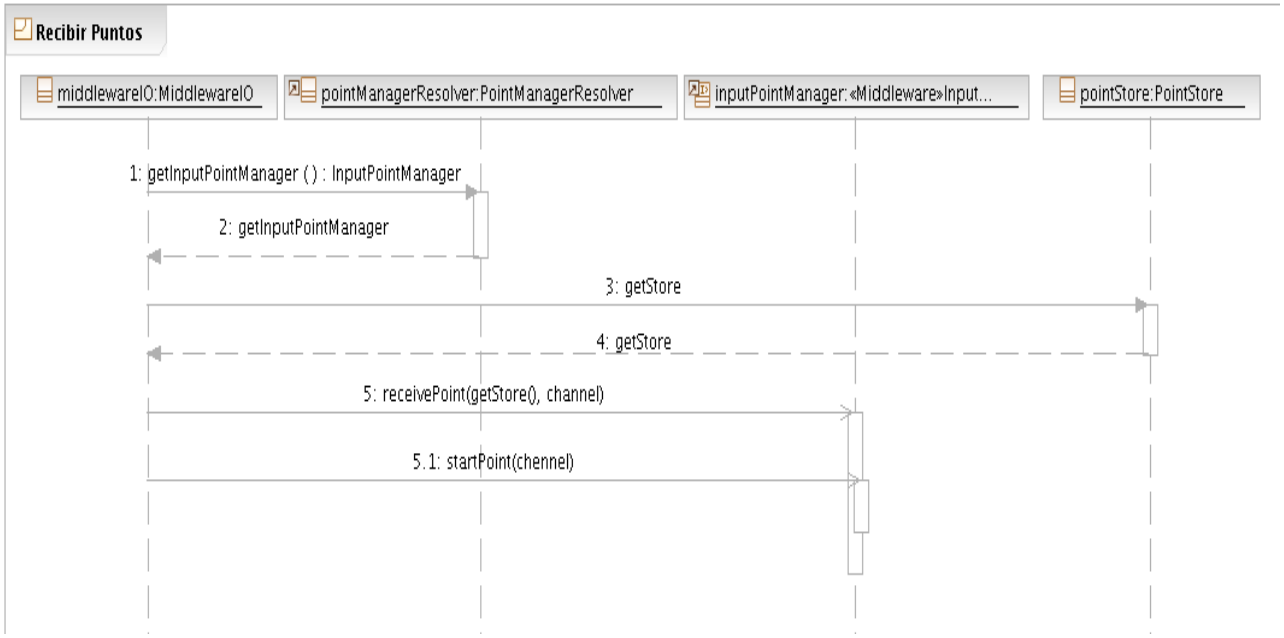

Figura 21: Diagrama de secuencia Recibir puntos.

2.9.5 Paquete *PointProvider*.

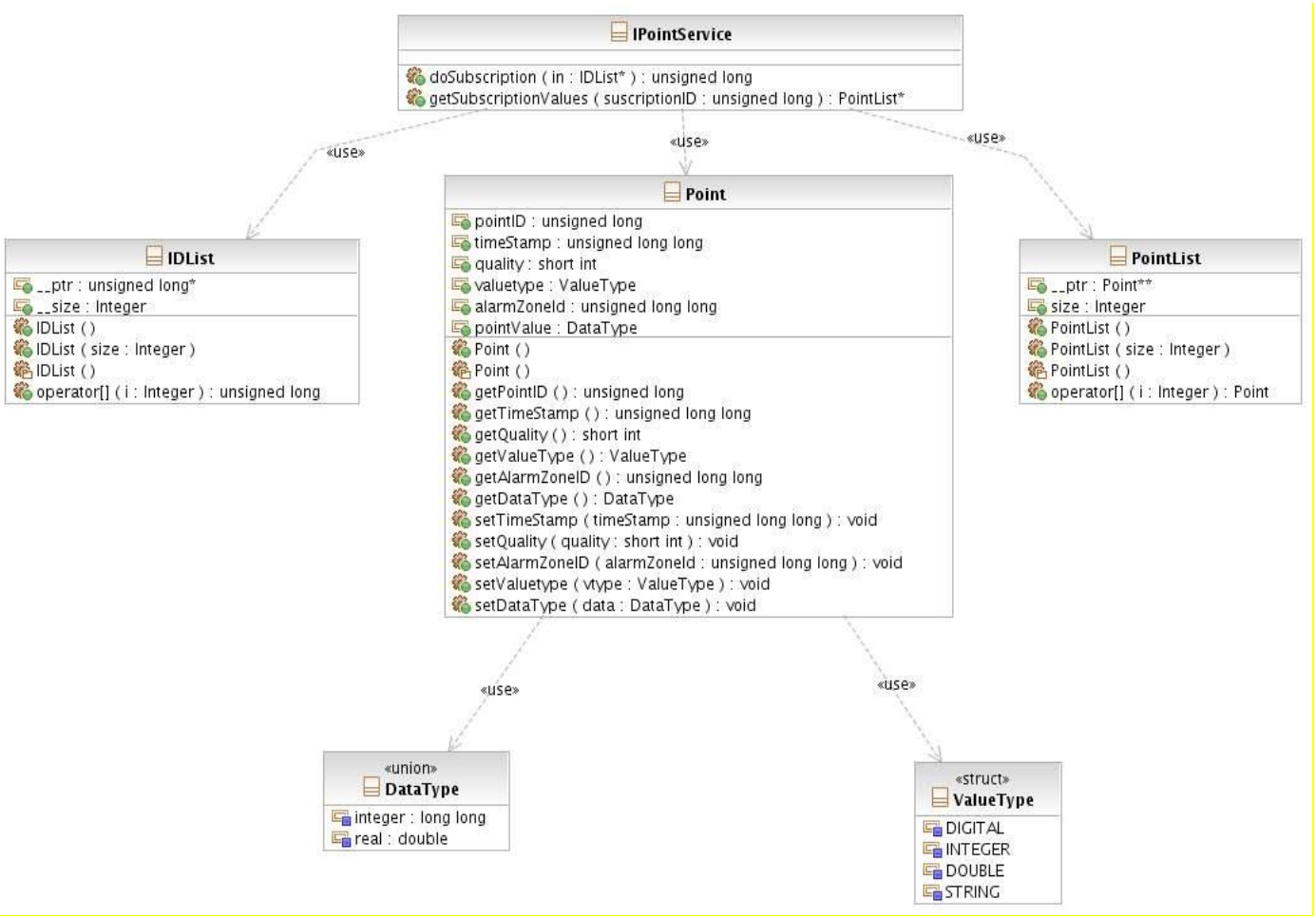

Figura 22: Diagrama de clases del paquete *PointProvider*.

2.9.5.1. Descripción de clases y métodos del paquete *Point Provider.*

Descripción de la interfaz *IPointService*	
Elemento	**Descripción**
IPointService	Utilizando el lenguaje Slice, se describen las interfaces, las operaciones, y los tipos de datos intercambiados entre los terceros y el Servicio de Acceso a Variables.
Métodos	
doSubscription()	Operación que se brinda a los terceros para la realización de suscripciones a variables.

getSubscriptionValues()	Operación que se brinda a los terceros para la obtención de valores de los puntos, previamente solicitados a través de una suscripción.

Tabla 16: Descripción de la interfaz *PointProvider*.

2.9.6 Paquete *CSCommunication*.

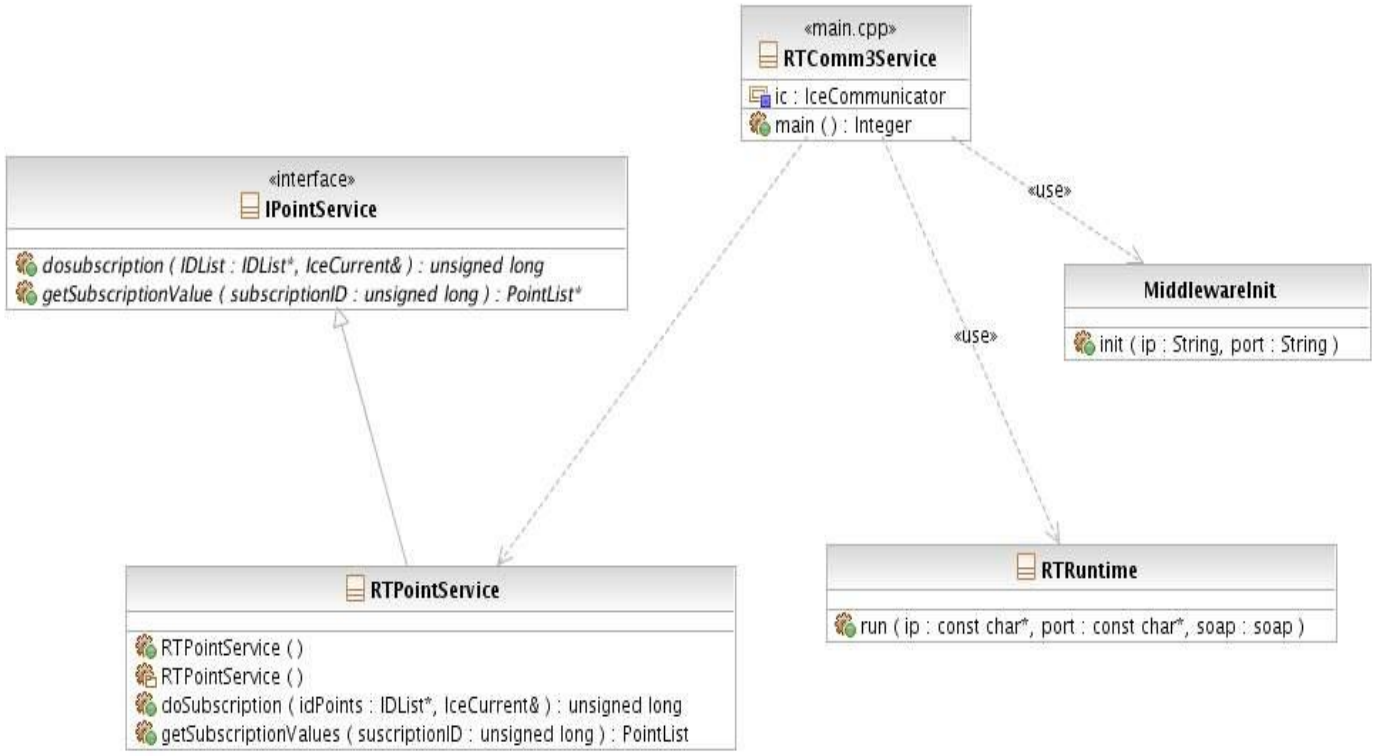

Figura 23 : Diagrama de clases del paquete *CSCommunication*.

2.9.6.1. Descripción de clases y métodos del paquete *CSCommunication.*

Descripción de la clase *RTComm3Service*

Elemento	Descripción
RTComm3Service	Esta clase tiene la responsabilidad de iniciar la comunicación con el *middleware* e iniciar el servidor implementado en ICE para garantizar las comunicaciones entre los terceros y el servidor de Comm3 en tiempo real.
	Atributos
ic:IceCommunication	Atributo para inicializar el entorno de ejecución de ICE.
	Métodos
main()	Inicializa la comunicación con el middleware y la ejecución del servidor para la respuesta a solicitudes de puntos mediante el protocolo ICE.

Tabla 17: Descripción de la clase RTComm3Service.

Descripción de la clase ***RTPointService***	
Elemento	**Descripción**
RTPointService	Clase que implementa la interfaz generada por el compilador Slice de ICE. Esta clase es la que invoca las funcionalidades del Servicio de Acceso a Variables que se encuentra en el paquete *PointEngine*.
	Métodos
RTPointService()	Constructor de la clase.
~RTPointService()	Destructor de la clase.
doSubscription()	Método encargado de realizar la invocación a la funcionalidad de realizar suscripción al paquete *PointEngine.*

getSubscriptionValues()	Método encargado de realizar la invocación a la funcionalidad de obtener valores de suscripción al paquete *PointEngine*. Se encarga de la conversión de la lista de puntos manipulada por el subsistema de Comunicación con Terceros a lista de puntos manipulada por el tercero.

Tabla 18: Descripción de la clase *RTPointService*.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

En el presente capítulo se abordan los temas de implementación del servicio. Se muestran las vistas de despliegue e implementación con sus respectivos componentes, el estilo de código a utilizar, algunas vistas significativas del código y las pruebas realizadas al servicio que validan su funcionamiento.

3.1. Vista de implementación.

El Servicio de Acceso a Variables por una parte implementa un acceso a las variables mediante servicios web, como el resto de los servicios que posee el subsistema de Comunicación con Terceros. Este servicio expone además una interfaz de comunicación de alto nivel para que sistemas externos accedan a las variables del Guardián del ALBA, que se basa en el paradigma cliente/servidor de ICE siempre abstrayendo a los clientes de los mecanismos de comunicación utilizados.

En la figura 24 se puede observar el modelo de implementación del Subsistema de Comunicación con Terceros que describe los componentes y sus relaciones que se construyen a partir de la implementación del sistema. Aunque se realizó un diseño arquitectónico de tres capas no significa que exista un ejecutable o bibliotecas por cada capa, los componentes por capas están empaquetados en estos componentes, en bibliotecas y en algunos casos ejecutables.

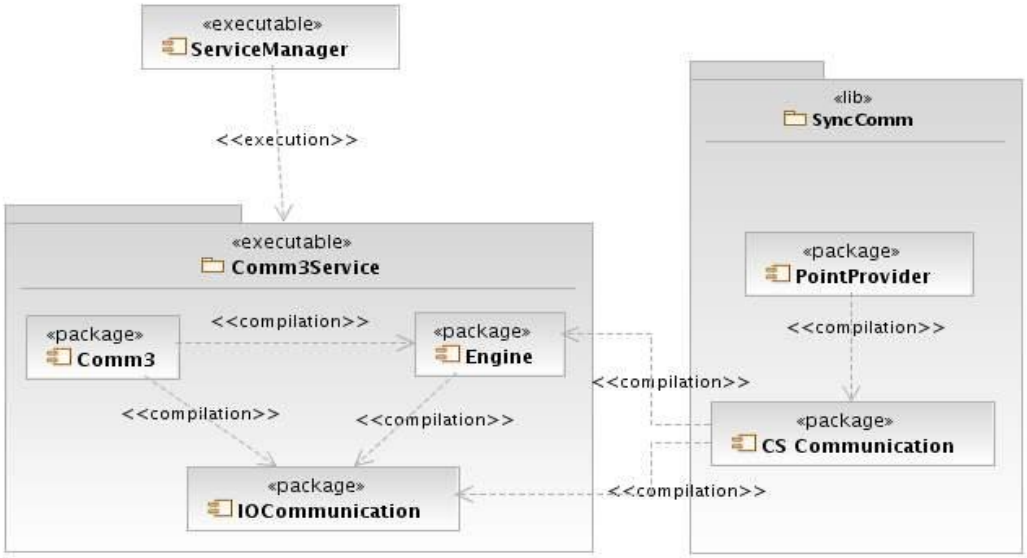

Figura 24: Modelo de Implementación de Comm3.

Descripción de los componentes

***ServiceManager*:** Es el componente encargado de publicar la descripción del Servicio de Acceso a Variables en una dirección HTTP que al ser invocado por los terceros, transmite una solicitud al servidor de Comunicación con Terceros, Comm3Service, a través del protocolo SOAP. Este componente representa la capa de presentación en la arquitectura definida.

Comm3Service: Representa el servidor de Comunicación con Terceros que define la lógica de las suscripciones y acceso a variables del sistema SCADA Guardián del ALBA como sus principales funcionalidades. Este componente encapsula los componentes *Comm3* y *Engine* representantes de la Capa de Negocio de la arquitectura y *IO Communication* con las funcionalidades de la Capa de Comunicación con el *middleware*.

LibSynComm: Esta biblioteca implementa los mecanismos de comunicación sincrónica, encapsula los componentes necesarios para resolver las peticiones cliente-servidor entre el SCADA y los terceros usando *ICE*. La misma empaqueta entre sus elementos principales el subsistema *PointProvider* de la Capa de Presentación que contiene la interfaz que usarán los sistemas SCADA para la solicitud de variables del Guardián del ALBA y *CSCommunication* como el componente de la Capa de Negocio que abstrae el paradigma cliente-servidor ICE.

3.1.1. Modelo de Componente de la capa de Lógica del Negocio.

El modelo de implementación tiene trazas con las clases del diseño que implementa, por lo que el mismo está representado en capas, compuestas por subsistemas y componentes agrupados en paquetes para una mejor organización. A continuación se muestran solamente los componentes pertenecientes a la capa de Lógica del Negocio por contener la mayor cantidad de componentes dentro del Subsistema de Comunicación con Terceros pertenecientes al Servicio de Acceso a Variables.

Esta capa contiene los componentes que implementan los mecanismos de suscripción y acceso a las variables del Guardián del ALBA usando los servicios web, así como los mecanismos de comunicación usando la tecnología ICE. Está representada por los componentes agrupados en *Comm3*, *PointEngine* y *CSCommunication*.

Componentes agrupados en *Comm3*

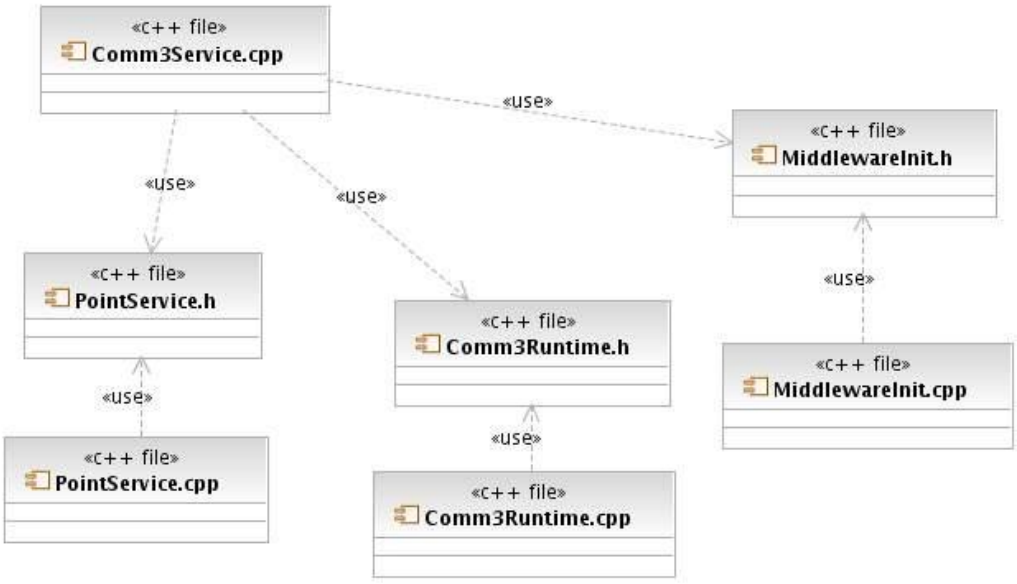

Figura 25: Diagrama de componentes agrupados en Comm3.

Componentes agrupados en *PointEngine*

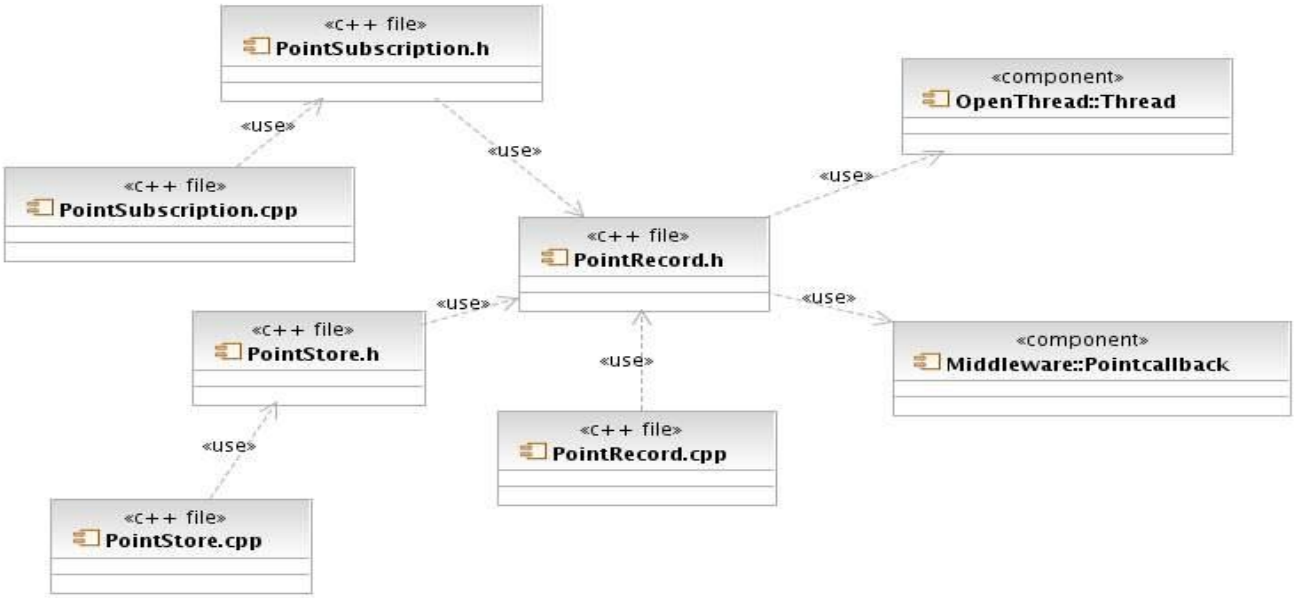

Figura 26: Relación entre componentes de PointEngine.

Componentes agrupados en *CSCommunication*

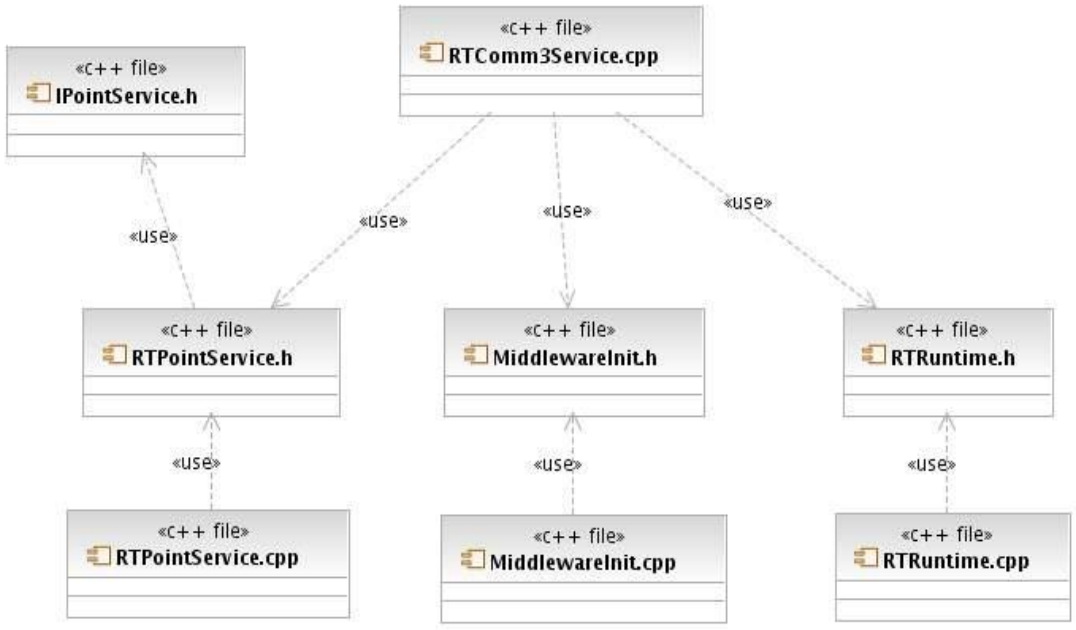

Figura 27: Relación entre los componentes de CSCommunication.

3.2. Vista de implantación.

La implantación o despliegue del sistema, satisfacen la solución planteada y muestra la comunicación entre el servidor de *Comm3* y los terceros a través de los servicios web y a través de la interfaz de comunicación de alto nivel basada en *ICE*. A continuación se muestra la vista de despliegue.

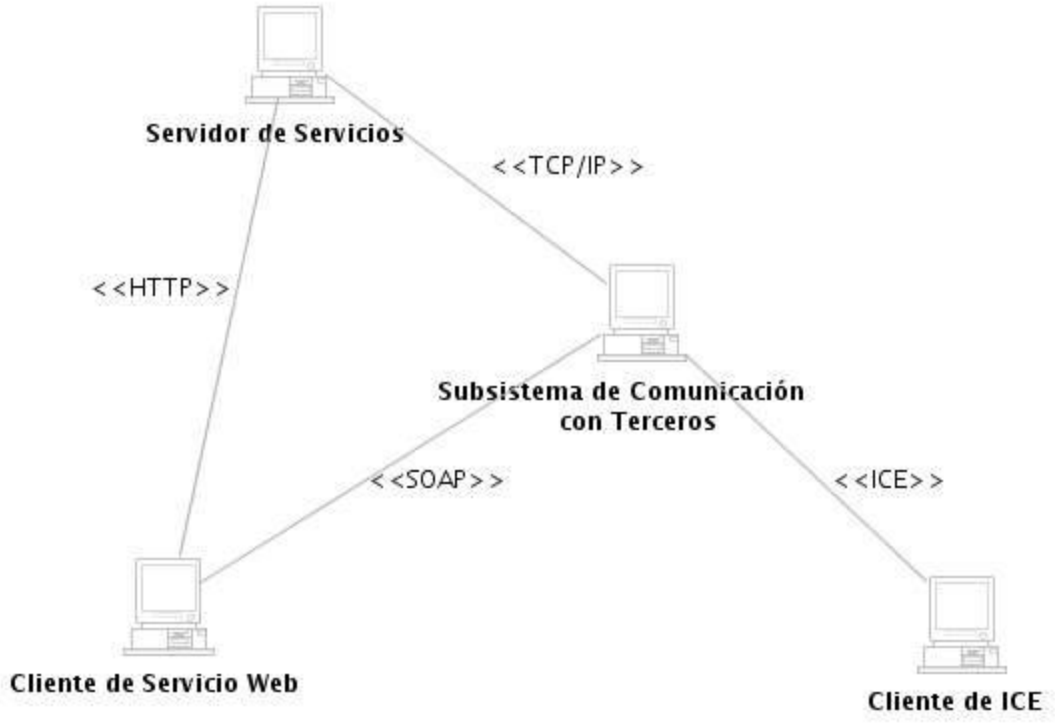

Figura 28: Diagrama de Despliegue.

Descripción de la vista de Despliegue

La figura 28 muestra el despliegue en nodos físicos identificando los diferentes nodos y la forma de conexión entre ellos. A continuación se expone una breve descripción de los nodos y sus conexiones.

Servidor de Servicios: En este nodo radica la instalación del Servidor de Servicios encargado de administrar y exponer los servicios web en una dirección HTTP a completa disposición del cliente. Este servidor puede ser parte del Subsistema de Comunicación con Terceros o un módulo de Apache con estas funcionalidades.

Cliente de Servicio Web: En este nodo radica la instalación de una aplicación personalizada que requiere acceder a la información del sistema SCADA Guardián del ALBA a través de los servicios web y

para ello se comunica por HTTP con el Servidor de Servicios para acceder a los servicios que este expone. Cuando el cliente realiza una petición acerca de cualquier servicio y obtiene su descripción, este es capaz de comunicarse con el Subsistema de Comunicación con Terceros a través del protocolo SOAP.

Cliente de ICE: Este nodo representa un tercero que tiene un enlace con el sistema SCADA Guardián del ALBA a través de una biblioteca basada en *ICE* que le permite invocar métodos remotos implementados por el Subsistema de Comunicación con Terceros. La comunicación será a través de TCP/IP o UDP que son transportes que proporciona la tecnología *middleware ICE*, por eso la conexión se especifica como *ICE*.

Subsistema de Comunicación con Terceros: En este nodo está instalado el Subsistema de Comunicación con Terceros del Guardián del ALBA, el mismo se comunica por TCP/IP con el Servidor de Servicios para registrar los mismos. La forma de comunicación entre ellos depende mucho de la solución final, es decir, si el Servidor de Servicios llega a ser parte del Subsistema de Comunicación con Terceros o se implementa como un módulo a parte. Este nodo recibe peticiones SOAP desde el Cliente de Servicio Web y envía respuestas vía SOAP. También recibe peticiones de otros terceros a través de una interfaz *ICE*, la conexión será a través del protocolo de comunicación que se decida, TCP o UDP que son transportes que proporciona la tecnología *middleware ICE.*

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. Seguir estas normas también evitará caer en errores y malas prácticas que dificultan la comprensión de las líneas de código.

A continuación se muestra el estilo de código utilizado en la implementación del Servicio de Acceso a Variables del Subsistema de Comunicación con Terceros. Este estilo se definió para ser utilizado por todos los subsistemas del sistema SCADA Guardián del ALBA (Estilo de Código SCADA, 2007)

3.3.1. Nombres.

- Los nombres de las clases son sustantivos singulares.
- Los nombres deben reflejar el ¿qué? y no el ¿cómo?
- 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.

• Concatenar calificadores de cómputo a las variables que almacenen el producto de tal cómputo (avg, sum, min, max, index).

• 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.

• Variables booleanas deben contener “Is” en su nombre.

• Los nombres de constantes deben contener solo letras mayúsculas.

• Minimizar el uso de abreviaturas. En caso de ser requeridas, se debe ser consistente en su uso y cada abreviación debe significar solo una cosa. En general agregar a la documentación las abreviaturas.

• 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.

• Se establece un tamaño de indentación estándar de tres (3) espacios, sin tabulaciones. Alinear secciones del código.

• 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.

• Emplear constantes en sustitución de números o cadenas de caracteres literales.

• Minimizar el alcance de las variables para evitar confusión y facilitar el mantenimiento.

• Emplear cada variable y rutina solo para un propósito.

• Evitar el uso de variables públicas, sustituirlas por variables privadas y métodos que proporcionen el valor de tal variable, para mantener el encapsulamiento.

• Minimizar el uso de conversiones de tipo forzadas (castings), cuando se requiera su uso, debe ser comentada la justificación.

• Emplear select-case o switch en sustitución de if anidados sobre la misma variables.

• 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.

• Mantener la modularidad del código bajo el criterio de la lógica que encierra, no exagerar la modularidad.

• 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.

• Inicializar todas las variables.

• Emplear líneas en blanco para separar los pasos lógicos (declaraciones, lazos, etc.).

• Siempre asignar NULL a los apuntadores luego de ser destruidos (solo aplica para C).

• Evitar prácticas que incrementan explosivamente la complejidad, como lo son: objetos y variables globales y saltos tipo go-to.

3.4. Vistas más significativas del código.

A continuación se muestran algunos fragmentos de código fundamentales para el funcionamiento del Servicio de Acceso a Variables.

En la figura 29 se muestra un fragmento del método ***onPointArrived*** utilizado para cuando arriba un punto, buscar en los puntos que están almacenados en el arreglo asociativo de puntos, si ya existe se actualiza su valor, si no se encuentra se adiciona. La clase a la cual pertenece este método fue implementada utilizando el patrón ***Singleton,*** lo que garantiza que esta clase sólo tenga una instancia y proporciona un punto de acceso global a esta. Esto es de vital importancia en nuestro servicio pues en esta clase se almacena toda la información acerca de los puntos que llegan del *middleware* y así se garantiza un solo punto de acceso a esta información.

*PointStore.cpp

```
19
20      void PointStore::onPointArrived(SCADA::Middleware::Point* value)
21      {
22          //actualizar los records de puntos.
23          //buscar el id en la lista.
24          UID pid = value->getID();
25
26          pthread_mutex_lock( &pointMutex );
27          std::map<UID, SCADA::Comm3::PointRecord*>::iterator it;
28          it = this->points.find(pid);
29
30          //actualizar el valor del punto.
31          if(it == this->points.end()) // no existe.
32          {
33      #ifdef NOCOM3CONFIG
34              //adicionar el punto al registro actual.
35              PointRecord* pr = new PointRecord(pid);
36              points.insert(std::make_pair<UID,PointRecord*>(pid,pr));
37
38              pr->update(value);
39      #else//desechar el punto
40              //notificar llegada de punto no configurado.
41              delete value;
42      #endif
43              pthread_mutex_unlock( &pointMutex );
44          }
45          else
46          {
```

Tasks | Console | Properties | Progress

Figura 29: Pantalla que muestra el método que maneja la llegada de cada nuevo punto obtenido del *middleware*.

En la figura 30 se muestra el método ***doSubscription***, encargado de realizar una suscripción, el cual contiene toda la lógica para la realización de dicha operación además del control de concurrencia para el control del acceso a la memoria compartida.

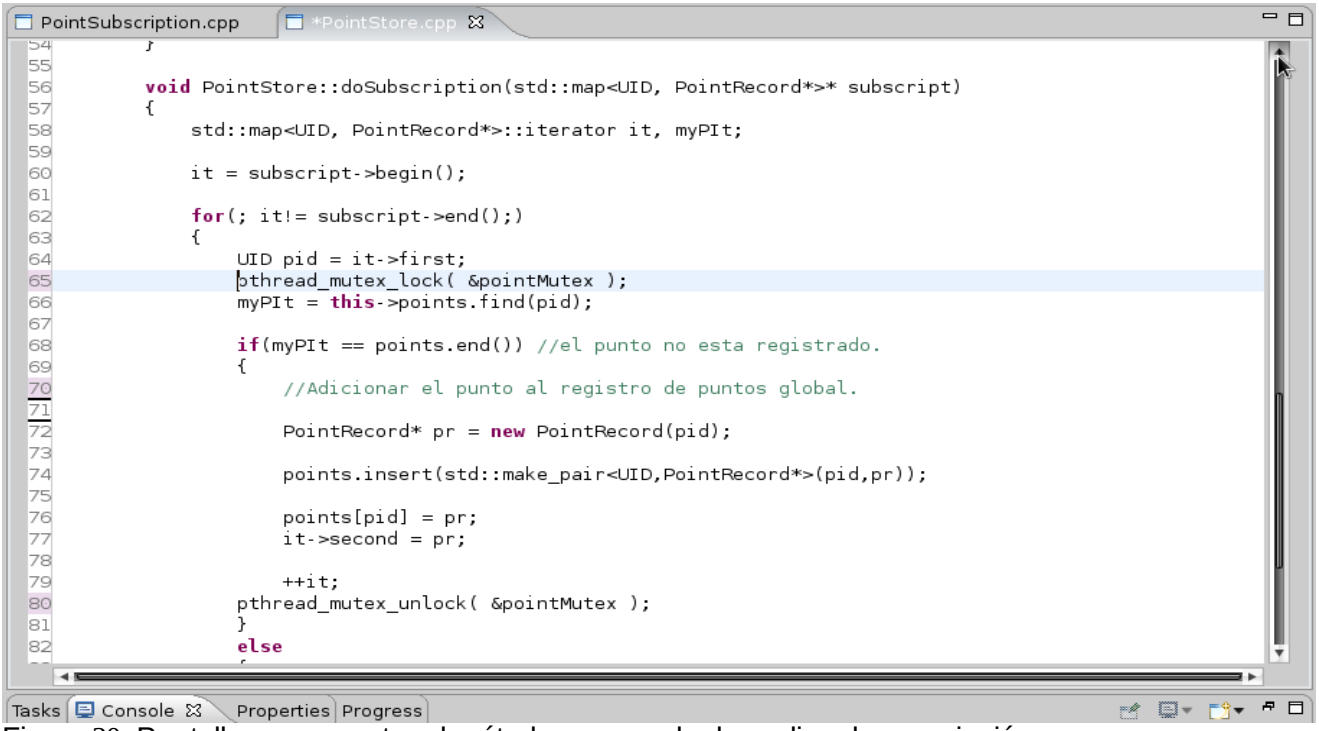

Figura 30: Pantalla que muestra el método encargado de realizar la suscripción.

3.5. Prueba.

La prueba del software constituye un elemento crítico para la garantía de la calidad del software. Los dos métodos de prueba más utilizados son el Método de Prueba de Caja Blanca y el Método de Prueba de Caja Negra. La diferencia entre ambos consiste en que con el método de Caja Blanca es necesario conocer el código y estar bastante relacionado con él para saber exactamente cuál es la lógica interna de lo que se va a probar, sin embargo en el método de Caja Negra solo basta con conocer las posibles entradas y salidas del programa.

A continuación se presentan una serie de pruebas de Caja Negra utilizando la técnica de particiones o clases de equivalencia que permiten validar la implementación del Servicio de Acceso a Variables del sistema SCADA Guardián del ALBA. De forma general se comprueban las funcionalidades principales del servicio para satisfacer las necesidades del cliente (requisitos).

3.5.1. 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 Servicio de Acceso a Variables.

Recursos Físicos

- Las computadoras utilizadas para el desarrollo de las pruebas contaron con 1.0 giga byte de memoria RAM, microprocesador Intel Core2Duo E4500 con velocidad de 2.20 Ghz, motherboard Intel y una capacidad en disco duro de 160 gigas. Red alámbrica.

Recursos Lógicos

- El sistema operativo en el cual se desarrollaron las pruebas fue Debian 5.0 (lenny), Kernel Linux 2.6.26-1-686, GNOME 2.22.3.
- La versión del *middleware* que se utilizó fue c.2.3.
- La versión del servicio de puntos utilizada fue la 0.2.
- La velocidad de la red con que se contó fue de 100 megabits por segundo.

3.5.2. Diseño de Casos de Prueba.

En esta sección se muestran los casos de pruebas para probar cada una de las funcionalidades definidas como críticas dentro del Servicio de Acceso a Variables y los resultados obtenidos en las mismas. Solo se reflejan las pruebas de funcionalidad y algunos detalles de rendimiento.

En la siguiente figura se muestra la forma en que se implementó el mecanismo para medir los tiempos de ejecución de cada uno de los casos de prueba que se expondrán a continuación. Se utiliza la función ***gettimeofday*** para medir de forma precisa el tiempo que tarda en ejecutarse la operación, esta función se encuentra en la librería *"sys/time.h".* Esta función ofrece teóricamente una precisión de microsegundos (0,001 milisegundos).

Su sintaxis es la siguiente:

int ***gettimeofday*** *(struct timeval *tp, NULL);*

Siendo *timeval* un registro con 2 campos: int *tv_sec*, int *tv_usec*, que indican los segundos y microsegundos, respectivamente, transcurridos desde el 1 de enero de 1970. (Ginés García Mateos, 2007)

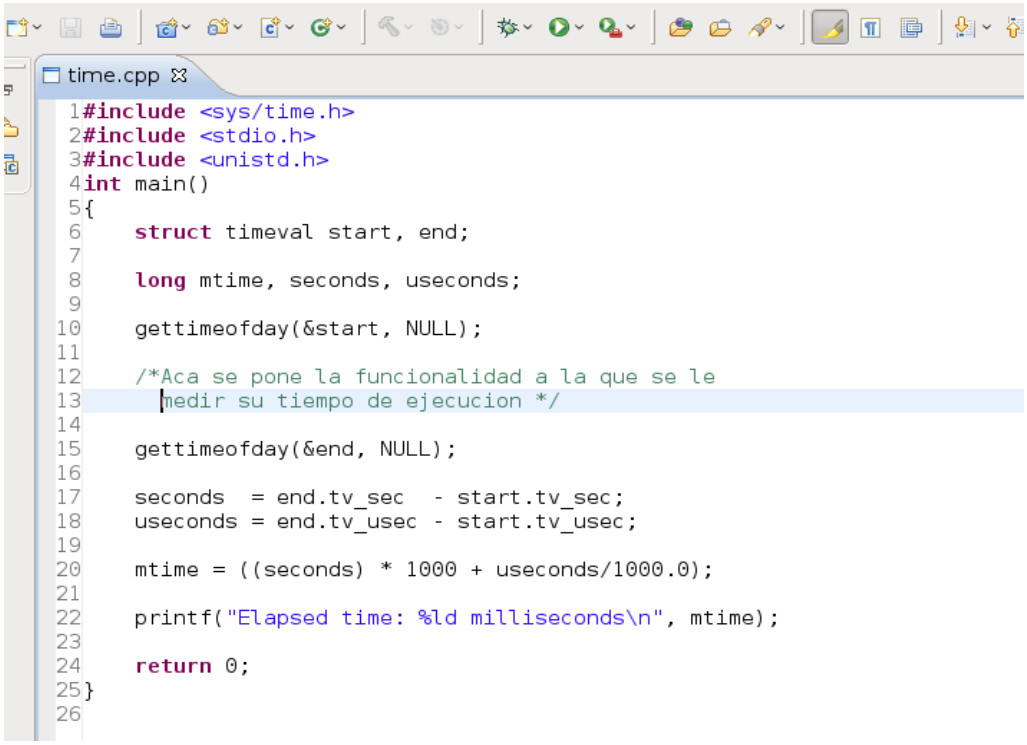

Figura 31: Funcionalidad utilizada para medir el tiempo de realización de las pruebas.

3.5.2.1. CPR 1: Realizar suscripción de puntos usando Servicios Web.

Descripción

El caso de prueba permite comprobar la realización exitosa de una suscripción en el Servidor de Comunicación con Terceros. Un cliente envía cierta cantidad de identificadores de puntos, los que son almacenados en el Servidor, y este le proporciona al cliente un identificador de la suscripción realizada.

Flujo Central

- Se obtiene la descripción (wsdl) del Servicio de Acceso a Variables.
- Se elabora un cliente de este servicio.
- Se crea la información de los identificadores de variables de las que se desean conocer los parámetros.
- Se selecciona la opción Suscribir Puntos.
- El cliente recibe el identificador de la suscripción realizada.

Condiciones de Ejecución

- Tiene que estar en ejecución el Servicio de Comunicación con Terceros.
- La prueba debe realizarse de forma remota, de manera que el servidor de Comunicación con Terceros se esté ejecutando en una estación de trabajo diferente a la del cliente del Servicio de Acceso a Variables.

Resultados

Caso #	Clases Válidas	Clases Inválidas	Resultados Esperados	Resultados Obtenidos	Observaciones
1	Se solicita la realización de una suscripción enviando **1000** identificadores de puntos.		-Se espera que se realice la suscripción de forma satisfactoria. -Se espera que la suscripción de puntos se efectúe en un tiempo inferior a los **500** ms, donde el consumo del procesador y memoria de RAM sean relativamente bajos.	Tiempo de respuesta: **14.3** ms. % de consumo de procesador: **1%.** Memoria RAM consumida: **5.6** Mb.	
2	Se solicita la realización de una suscripción enviando **5000** identificadores de puntos.		-Se espera que se realice la suscripción de forma satisfactoria. -Se espera que la suscripción de puntos se efectúe en un tiempo inferior a los **500** ms, donde el consumo del procesador y memoria de RAM sean relativamente bajos.	Tiempo de respuesta: **95.8** ms. % de consumo de procesador: **1%.** Memoria RAM consumida: **5.6** Mb.	
3	Se solicita la realización de una suscripción enviando **10000** identificadores de puntos.		-Se espera que se realice la suscripción de forma satisfactoria. -Se espera que la suscripción de puntos se efectúe en un tiempo inferior a los **500** ms, donde el consumo del procesador y memoria de RAM sean relativamente bajos.	Tiempo de respuesta: **252.4** ms. % de consumo de procesador: **1%.** Memoria RAM consumida: **5.6** Mb.	

Tabla 19: Resultados de las pruebas a la funcionalidad Realizar Suscripción de Puntos.

3.5.2.2. CPR 2: Obtener valores de Suscripción usando Servicios Web.

Descripción

El caso de prueba permite comprobar la obtención de valores de una suscripción, solicitados al servidor de Comunicación con Terceros. Un cliente envía un identificador de suscripción al servidor y este le retorna los valores de los puntos de dicha suscripción.

Flujo Central

- Se obtiene la descripción (wsdl) del Servicio de Acceso a Variables.
- Se elabora un cliente de este servicio.
- Se crea la información de los identificadores de variables de las que se desean conocer los parámetros.
- Se selecciona la opción Suscribir Puntos.
- El cliente recibe el identificador de la Suscripción realizada.
- Se selecciona la opción Iniciar Lectura, donde de forma transparente al usuario se le envía al servidor el identificador de la suscripción realizada anteriormente.

Condiciones de Ejecución

- Tiene que estar en ejecución el *middleware* del Guardián del ALBA.
- Tiene que estar en ejecución la aplicación de Envío de Puntos.
- Tiene que estar en ejecución el Servicio de Comunicación con Terceros.
- La prueba debe realizarse de forma remota, de manera que el servidor de Comunicación con Terceros se esté ejecutado en una estación de trabajo diferente a la del cliente del Servicio de Acceso a Variables.

Resultados

Caso #	Clases Válidas	Clases Inválidas	Resultados Esperados	Resultados Obtenidos	Observaciones
1	Se solicita la obtención de valores de **1000** puntos.		-Se espera que se obtengan los valores de la suscripción exitosamente. -Se espera que la obtención de los valores de los puntos se	Tiempo de respuesta: **28.1** ms	
				% de consumo de procesador: **2%**	

			efectúe en un tiempo inferior a los **500** ms.	Memoria RAM consumida: **9.4** Mb	
2	Se solicita la obtención de valores de **5000** puntos.		-Se espera que se obtengan los valores de la suscripción exitosamente. -Se espera que la obtención de los valores de los puntos se efectúe en un tiempo inferior a los **500** ms.	Tiempo de respuesta: **298.2** ms	
				% de consumo de procesador: **2%**	
				Memoria RAM consumida: **9.4** Mb	
3	Se solicita la obtención de valores de **10000** puntos.		-Se espera que se obtengan los valores de la suscripción exitosamente. -Se espera que la obtención de los valores de los puntos se efectúe en un tiempo inferior a los **500** ms.	Tiempo de respuesta: **425** ms	
				% de consumo de procesador: **3%**	
				Memoria RAM consumida: **9.5** Mb	

Tabla 20: Resultados de las pruebas a la funcionalidad Obtener valores de la Suscripción.

3.5.2.3 CPR3: Transmisión de una cantidad N de puntos de manera distribuida usando ICE.

Descripción

El caso de prueba permite comprobar la velocidad de transmisión de variables de forma remota usando la tecnología ICE. Una aplicación emisora envía una cantidad específica de puntos. Una aplicación receptora, recibe todos los puntos enviados por el emisor y muestra por pantalla el tiempo de recepción para la cantidad de puntos que fueron transmitidos registrando el mismo en una tabla. Son diferentes pruebas que van desde mil hasta 10 mil puntos y se realizan cinco iteraciones por cada cantidad diferente obteniendo una mayor muestra que permite graficar e interpretar mejor los resultados. El punto que se transmite es la misma estructura de puntos definida en el Guardián del ALBA. Este caso de prueba se realiza utilizando el mecanismo de cliente/servidor de ICE, que es el utilizado en el Servicio de Acceso a Variables.

Flujo Central

- Se ejecuta una aplicación que actuará como servidora, especificando la cantidad de puntos que debe crear, para su posterior envío a los clientes.
- Se ejecuta una aplicación que actuará como cliente solicitando los valores de las variables

almacenadas en el servidor.

- Se registra el tiempo de recepción.
- La prueba se repite 5 veces, cada vez que el cliente termine de recibir la cantidad especificada se ejecuta nuevamente el servidor con otra cantidad de variables y así hasta concluir las cinco iteraciones.

Condiciones de ejecución

- Al término de las cinco iteraciones de cada prueba, se debe finalizar la ejecución de todos los servicios de ICE.

Resultados

Por cada prueba y/o iteración se capturó y registró el tiempo de recepción de las variables. Se probó la transmisión de 3 cantidades de puntos desde mil hasta 10 mil con diferencia de 5 mil entre pruebas y cinco iteraciones de cada prueba para un total de 15 pruebas realizadas. En la tabla que aparece a continuación solo se muestra el promedio de las cinco iteraciones para cada una de las cantidades de puntos enviadas.

Caso #	Clases Válidas	Clases Inválidas	Resultados Esperados	Resultados Obtenidos
1	Se envían **1000** puntos.		Se espera que el envío de puntos se efectúe en un tiempo inferior a los **500** ms.	Tiempo de respuesta promedio: **98** ms.
2	Se envían **5000** puntos.		Se espera que el envío de puntos se efectúe en un tiempo inferior a los **500** ms.	Tiempo de respuesta promedio: **145** ms.
3	Se envían **10000** puntos.		Se espera que el envío de puntos se efectúe en un tiempo inferior a los **500** ms.	Tiempo de respuesta promedio: **364** ms.
4	Se envían **15000** puntos.		Se espera que el envío de puntos se efectúe en un tiempo inferior a un segundo.	Tiempo de respuesta promedio: **654** ms

5	Se envían **20000** puntos.		Se espera que el envío de puntos se efectúe en un tiempo inferior a un segundo.	Tiempo de respuesta promedio: **827** ms

Tabla 17: Resultados de las pruebas a la funcionalidad Transmisión de una cantidad N de puntos usando ICE.

3.5.3. Análisis de los resultados de las pruebas.

Los resultados arrojados por la ejecución de los casos de prueba ***CPR1 y CPR2*** fueron el resultado de la ejecución del Subsistema de Comunicación con Terceros, el cliente del Servicio de Acceso a Variables, y la aplicación a la cual se conecta el subsistema para obtener los datos del subsistema *middleware*, en máquinas diferentes, por ello los resultados de las pruebas pertenecen a un escenario totalmente distribuido. Si se analizan las tablas se puede apreciar que a medida que aumenta la cantidad de información a recibir aumentan los tiempos de recepción de la información. Para una cantidad de puntos superior a los 10000 se superará los 500 milisegundos (para 15 mil puntos se demora 738 milisegundos).

Como la transmisión de datos a través de Servicios Web es mediante el protocolo HTTP y las pruebas realizadas fueron dentro de una misma subred a una hora en la que el número de usuarios conectados en la universidad era mínimo, se puede asegurar que los resultados pueden variar con facilidad y superar el límite de 500 milisegundos e incluso duplicarlo si se aplica en otras condiciones.

El caso de prueba ***CPR3*** muestra los resultados del intercambio de puntos entre un cliente y un servidor utilizando ICE sobre TCP. Esta tecnología es más eficiente con mayores volúmenes de datos a transmitir por lo que su utilización en el Servicio de Acceso a Variables garantiza que se puedan enviar grandes cantidades de datos de manera rápida a los terceros.

CONCLUSIONES

Al término de la presente investigación para el desarrollo del Servicio de Integración con Terceros para el Acceso a Variables del sistema SCADA Guardián del ALBA, se concluye que:

- ✓ El estándar OPC Arquitectura Unificada es el que generalmente se utiliza en la actualidad para el desarrollo de aplicaciones que se comunican con sistemas SCADA y no se puede utilizar en el desarrollo del Servicio de Acceso a Variables debido a que la especificación de este estándar se distribuye bajo licencias comerciales.
- ✓ El mecanismo de comunicación desarrollado para el acceso a variables se integra de manera satisfactoria al Subsistema de Comunicación con Terceros del sistema SCADA Guardián del Alba.
- ✓ El Servicio de Acceso a Variables desarrollado permite a los sistemas externos establecer una comunicación sencilla y efectiva, con el sistema SCADA Guardián del ALBA, posibilitando el acceso a sus puntos.
- ✓ El Servicio de Acceso a Variables desarrollado posibilita el envío de información relacionada con variables en tiempo real (envío de 1000 a 10000 puntos a los terceros en un tiempo inferior o igual a los 500 milisegundos) usando la tecnología de comunicación distribuida ICE.
- ✓ El Servicio de Acceso a Variables 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 cliente o tercero.

RECOMENDACIONES

- ✓ Utilizar el Servicio de Acceso a Variables en otros sistemas que necesiten de las variables del Guardián del ALBA, para poder identificar nuevas funcionalidades.
- ✓ Implementar el manejo de suscripciones por descripción de variables.
- ✓ Definir el mecanismo de manejo de excepciones en el Subsistema de Comunicación con Terceros para la implementación en el Servicio de Acceso a Variables.
- ✓ Realizar un estudio de factibilidad sobre la aplicación de una Arquitectura Orientada a Servicios (SOA) en la comunicación del Guardián del ALBA con sistemas externos.

REFERENCIAS BIBLIOGRÁFICAS

1. **BOOCH, GRADY, JACOBSON, IVAR, RUMBAUGH, JAMES. 2007.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : ADDISON-WESLEY, 2007. 978-84-7829-087-1.
2. **Carlos de Castro Lozano, Cristóbal Romero Morales. 2007.** Introducción a SCADA. *Universidad de Córdoba.* [En línea] 2007. http://www.uco.es/grupos/eatco/automatica/ihm/descargar/scada.pdf.
3. **Colectivo de Autores. 2000.** OPC: UN ESTANDAR EN LAS REDES INDUSTRIALES Y BUSES DE CAMPO. *geosites.* [En línea] 2000. http://es.geocities.com/pvidalp/OPC.pdf..
4. *El reto de los servicios Web para el software libre.* **J. J. Domínguez Jiménez, A. Estero Botaro, I. Medina Bulo, M. Palomo Duarte y F. Palomo Lozano. 2008.**
5. http://www.willydev.net/InsiteCreation/v1.0/WillyCrawler/2008.06.07.Articulo.El%20reto%20de%20los%20servicios%20Web%20para%20el%20software%20libre.pdf.
6. *Especificación de Comunicación con Terceros.* **M.Sc Moisés Herrera Vázquez, Ing. Maikel Pérez Javier. 2008.** Mérida, Venezuela: Proyecto SCADA Guardián del ALBA, 2008, Vol. I.
7. **FERNANDO ALEJANDRO CAMPOS, GERMÁN MAURICIO CORAL, OSCAR AMAURY ROJAS. 2004.** Sistema de Supervisión y Control Industrial Multiplataforma. *Enlace Informático.* [En línea] Diciembre de 2004. http://enlaceinformatico.unicauca.edu.co/docs/v3e1a2.pdf. http://enlaceinformatico.unicauca.edu.co/docs/v3e1a2.pdf.
8. **ICONICS. 2009.** [En línea] ICONICS Corporation, 2009. http://www.iconics.com/company/corporate_info.asp.
9. **Invensys Systems, Inc. 2009.** Wonderware Invensys. [En línea] 2009. http://global.wonderware.com/EN/Pages/LearningTools.aspx#suitevoyagertutorial.
10. **Kassl GmbH. 2009.** German Software Development Company Kassl GmbH. [En línea] 2009. http://www.kassl.de/.
11. **MatrikonOPC. 2009.** OPC Data Access (OPC DA) Versions & Compatibility. *matrikonopc.* [En línea] 2009. http://www.matrikonopc.com/opc-server/opc-data-access-versions.aspx.
12. **Ministerio del Poder Popular para la Energía y Petróleo. 2009.** *Sitio oficial de Petróleos de Venezuela.* [En línea] 2 de 04 de 2009. http://www.pdvsa.com/.
13. **MonteJava. 2007.** Soluciones empresariales. *J2EE vs .NET (II): Servicios Web. Un territorio común.* [En línea] 2007. http://www.montejava.es/articulo14.asp.
14. **Northern Dynamic. 2009.** [En línea] Northern Dynamic, 2009. http://www.nordyn.com/.
15. **OPC Foundation. 2003.** OPC XML DA Specification. *opcfoundation.org.* [En línea] 2003. http://www.opcfoundation.org/Login.aspx?PageState=100.

16. **Pedro Álvarez, José Ángel Bañares. 2008.** Servicios Web. [En línea] 2008. http://iaaa.cps.unizar.es/docencia/SW/6.TecnologiaBasicaSW@.pdf.
17. **Ramadan Fan, Saudi Aramco, Dhahran, Dr. Lahouari Cheded, Dr. Onur Toker.** An XML Web Services Approach to SCADA Systems Design. *King Fahd University of Petroleum and Minerals.* [En línea] [Citado el: 23 de Enero de 2009.] http://faculty.kfupm.edu.sa/EE/ajmal/conf/IEEEGCC2004/83-t2p391-revised.pdf.
18. **Sepúlveda, José. 2006.** MES: Sistemas de Ejecución de Manufactura. *ElectroIndustria.* [En línea] Octubre de 2006. http://www.emb.cl/electroindustria/articulo.mv?xid=166&tip=7.
19. *Software de Supervisión y Control.* **GRAELLS, JORDI AYZA. 2003.** s.l. : Revista Automática e Instrumentación, 2003, Vol. 344.
20. **ZeroC. 2009.** *ZeroC™, the Home of Ice.* [En línea] 2009. http://www.zeroc.com/ice.html.
21. **Engelen, Robert A. van. 2006.** Florida State University, Computer Science Department. *The gSOAP Toolkit for SOAP Web Services and XML-Based Applications.* [En línea] 2006. [Citado el: 20 de diciembre de 2008.] http://www.cs.fsu.edu/~engelen/soapmain.html.
22. **kersel. 2008.** [En línea] 2008. http://kersel.com.mx/servicios.htm.
23. **Chacón, Carolina Gómez. 2006.** Software Libre en Venezuela: Independencia o Soberanía Tecnológica. *Software Libre Chile.* [En línea] 11 de Noviembre de 2006. [Citado el: 20 de Marzo de 2009.] http://www.softwarelibre.cl/drupal//?q=node/84/print.
24. **Ginés García Mateos. 2007.** MEDIDA DEL TIEMPO Y LA MEMORIA DE UN PROGRAMA . *Departamento de Informática y Sistemas.* [En línea] 2007. [Citado el: 24 de abril de 2009.] http://dis.um.es/~ginesgm/medidas.html.

BIBLIOGRAFÍA CONSULTADA

1. **Ramadan Fan, Saudi Aramco, Dhahran, Dr. Lahouari Cheded, Dr. Onur Toker.** An XML Web Services Approach to SCADA Systems Design. *King Fahd University of Petroleum and Minerals.* [Citado el: 23 de Enero de 2009.]
2. *El reto de los servicios Web para el software libre.* **J. J. Domínguez Jiménez, A. Estero Botaro, I. Medina Bulo, M. Palomo Duarte y F. Palomo Lozano. 2008.** 2008.
3. **Pérez, Maikel.** *Informe de Pruebas del Middleware v1.1.* 2008.
4. Multiprograma & Multitarea. *zator.com.* [En línea] Zator Systems. [Citado el: 10 de Abril de 2009.]
    a. http://www.zator.com/Cpp/E1_7b.htm.
5. **ZeroC. 2009.** *ZeroC™, the Home of Ice.*2009. http://www.zeroc.com/ice.html.
6. **Ramadan Fan, Saudi Aramco, Dhahran, Dr. Lahouari Cheded, Dr. Onur Toker.** An XML Web Services Approach to SCADA Systems Design. *King Fahd University of Petroleum and Minerals.* [Citado el: 23 de Enero de 2009.]
7. **BOOCH, GRADY, JACOBSON, IVAR, RUMBAUGH, JAMES. 2007.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : ADDISON-WESLEY, 2007. 978-84-7829-087-1.
8. *Especificación de Comunicación con Terceros.* **M.Sc Moisés Herrera Vázquez, Ing. Maikel Pérez Javier. 2008.** Mérida, Venezuela: Proyecto SCADA Guardián del ALBA, 2008, Vol. I.
9. *Selección de Tecnologías para el desarrollo del Subsistema de Comunicación con Terceros.* **Ing. Maikel Pérez Javier. 2009.** Habana, Cuba: Proyecto SCADA Guardián del ALBA, 2009.
10. *Documento de entrega formal del prototipo de Subsistema de Comunicación con Terceros.* **Ing. Maikel Pérez Javier, José Antonio Aragón Cáceres. 2009.** Habana, Cuba: Proyecto SCADA Guardián del ALBA, 2009.
11. **Robert A. van Engelen. 2008.** *A Framework for Service-Oriented Computing with C and C++ Web Service Components*, ACM Transactions on Internet Technologies, Volume 8, Issue 3, Article 12, May 2008.
12. **Robert A. van Engelen and Wei Zhang.2008.** *Identifying Opportunities for Web Services Security Performance Optimizations*, to appear in the proceedings of the IEEE Services Computing Conference (SCC), 2008.

ANEXOS

Anexo I: Matriz de Decisión elaborada para la selección de Tecnologías para la implementación del Módulo de Comunicación con Terceros del sistema SCADA Guardián del ALBA.

La matriz de decisiones permite la toma de decisiones durante la fase de arquitectura de la aplicación:

- Especificando las necesidades y dando prioridad a una lista de criterios a evaluar.
- Evaluación, valoración, y comparación de las diferentes soluciones.
- Seleccionar la mejor solución correspondiente.

En resumen, una matriz de decisión es una herramienta de decisiones utilizada por los arquitectos de software como parte de sus herramientas a la hora de seleccionar una tecnología o aplicación a utilizar.

Elementos de la Matriz de Decisión:

A continuación se detalla toda la información que debe venir en la matriz con el fin de realizar una decisión imparcial:

Criterios:	Opciones:	Importancias:	Puntajes:
Desarrollar una jerarquía de criterios de decisión, también conocido como modelo de decisión.	Determinar las opciones, también llamado soluciones o alternativas.	Asignar un valor a cada criterio sobre la base de su importancia en la decisión final, generalmente es un valor comprendido entre 0 y 100.	Tasa de cada opción en una relación de escala mediante la calificación a cada criterio.

Tabla 21: Elementos de la matriz de decisión.

Como calificar una opción?

Calificación	Descripción
0	No Aceptable
1	Poco Aceptable
2	Aceptable
3	Sobresaliente
4	Exelente

Tabla 22: Valores cuantitativos de las calificaciones a las tecnologías.

Modelo de Decision		ALTERNATIVAS							
		OPC(***)		OPC UA(***)		WS		ICE	
Criterios	Importancia	Calificacion	Puntaje	Calificacion	Puntaje	Calificacion	Puntaje	Calificacion	Puntaje
Facilidad de implementación.	10	1	10	0	0	3	30	2	20
Documentación disponible y a la alcance del personal	10	3	30	0	0	3	30	2	20
Posibilidad de crear soluciones universales	20	2	40	4	80	4	80	3	60
Desarrollo de aplicaciones distribuidas	15	3	45	3	45	3	45	4	60
Consumo de recursos	15	2	30	0	0	2	30	2	30
Soporte de los desarrolladores de las tecnologías	15	3	45	3	45	3	45	2	30
Transmisión de un elevado número de variables(1000, 5000 y 10000)	15	3	45	0	0	2	30	4	60
Total	100	17	245	10	170	20	290	19	280

Puntaje = Valor * Importancia
Importancia = Rango entre 1-100
(***) = tecnología propietaria

Tabla 23: Modelo de decisión de la matriz de selección de tecnología para la implementación del Servicio de Acceso a Variables del Subsistema de Comunicación con Terceros.

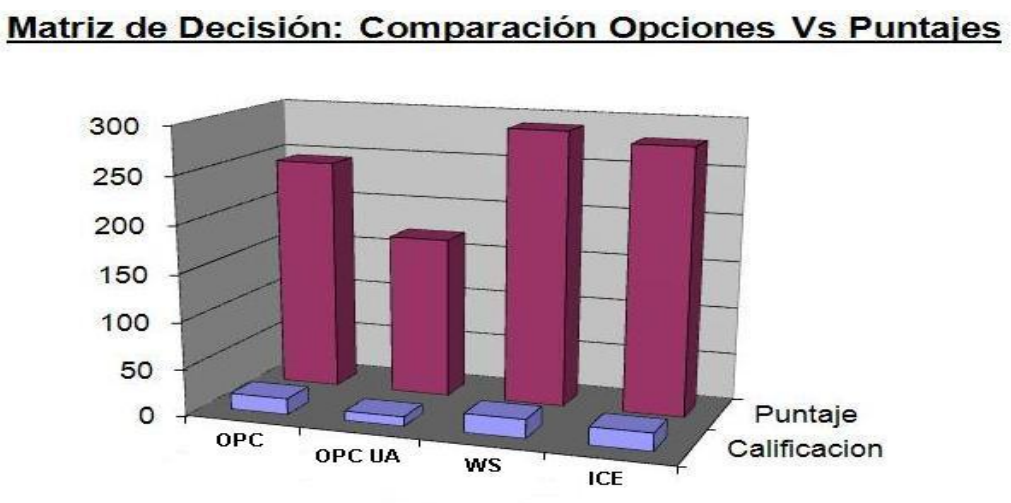

Figura 32: Matriz de decisión, para la selección de la tecnología a utilizar.

Anexo II: Utilización de las herramientas proporcionadas por gSOAP.

soapcpp2 es un compilador que a partir de un fichero de cabecera de C/C++, genera código de los stubs del cliente, los esqueletos de los servicios web implementados en el servidor y los ficheros para la codificación/decodificación de datos.

wsdl2h es un parseador de código WSDL y esquemas de XML que genera un fichero de cabecera (.h) con los servicios web y los tipos de datos en C/C++ usados por esos servicios.

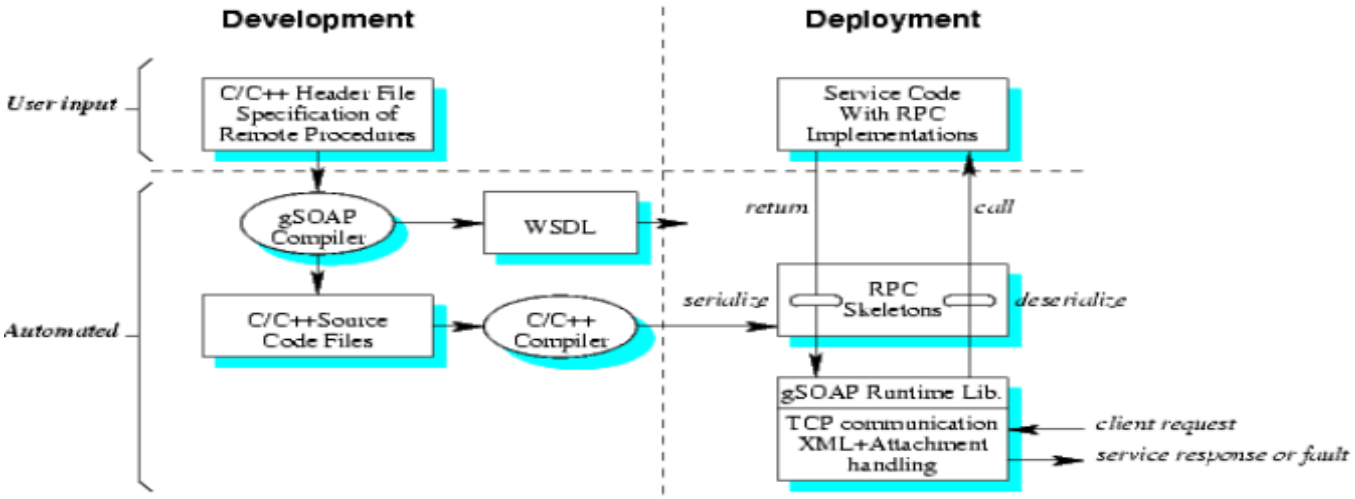

Figura 33: Proceso de creación de un servicio web a partir de la elaboración de un fichero cabecera (.h). **(Engelen, 2006)**

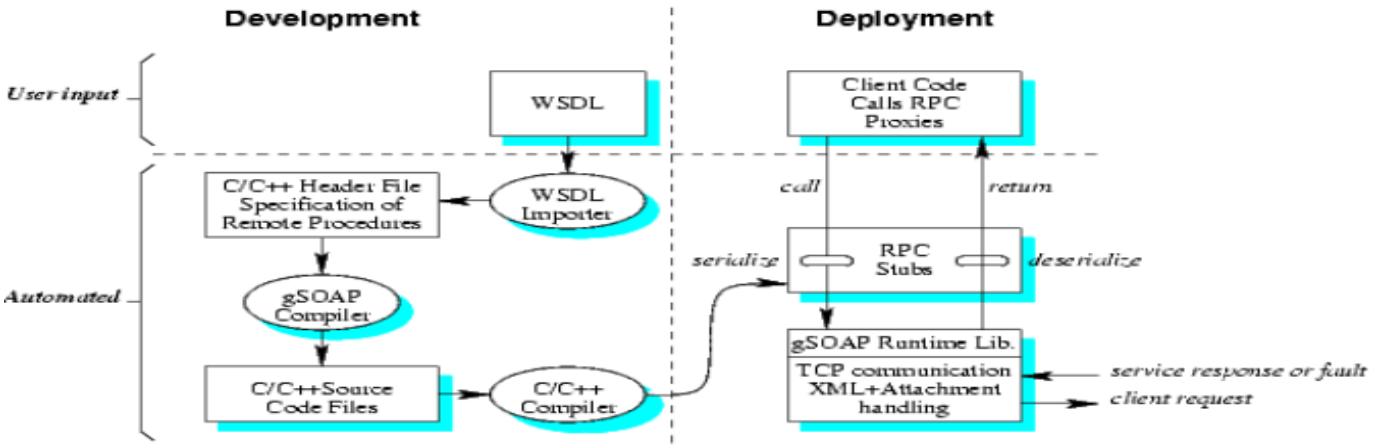

Figura 34: Proceso de creación de un cliente de un servicio web a partir de un fichero wsdl. **(Engelen, 2006)**

Opciones utilizadas del compilador soapcpp2:

-C genera el código de la parte cliente solamente.

-S genera el código de la parte servidora solamente.

-L No genera soapClientLib/*soapServerLib*, los cuales están específicamente destinados a construir bibliotecas clientes/servidoras de manera estática o dinámica.

Opciones utilizadas del parseador wsdl2h:

-s para deshabilitar la opción de importar por defecto el uso de vectores de la Standard Library.

Para la construcción del esqueleto del Servicio de Acceso a Variables:

soapcpp2 -S -L WSPoint.h

El comando anterior genera un fichero WSDL que será expuesto a los clientes para que puedan consumir los servicios.

Para la construcción del stub de un cliente del Servicio de Acceso a Variables:

wsdl2h -s WSPoint.wsdl

El fichero WSPoint.wsdl es un fichero WSDL que es expuesto al cliente con la descripción del Servicio de Acceso a Variables del Guardián del ALBA.

soapcpp2 -C -L WSPoint.h

Todo listo para implementar el cliente, e fichero cabecera ***WSPoint.h*** es generado por el comando anterior.

Si se desea elaborar un cliente en otro sistema operativo o en otro lenguaje de programación utilizo otras herramientas como Apache Axis o NetBeans IDE, aunque la herramienta gSOAP es multiplataforma, pero solo para la elaboración de clientes de servicios web en C++.

Anexo III: Instrucciones de ejecución del Servicio de Acceso a Variables.

1. Ejecutar el *middleware* del Guardián del ALBA (versión c2.3):

2. Ejecutar la aplicación para el Envío de Puntos:

- Situarse en el proyecto *Sender*
- Ejecutar ./*Debug*./*Sender*
- Posteriormente se introduce ip, puerto por donde se está ejecutando el *middleware* y canal por donde se desea enviar los datos.
- Se adicionan los puntos que va a enviar el *middleware*.
- Se establece la comunicación con el *middleware* para comenzar a enviar estos puntos. (Botón Establecer)
- Si se quiere que un punto que estoy enviando tenga valor constante, hago doble clic encima del valor correspondiente al punto en cuestión.
- Si se quiere volver a enviar valores aleatorios, se escribe (*****) en el campo Valor del punto, e inmediatamente comenzara a enviar valores aleatorios nuevamente.

3. Ejecutar el Servicio de Acceso a Variables:

- Situarse en el proyecto ServicioAccesoVariables
- Ejecutar ./*Debug*/SAV IpComm3 PuertoComm3
- Posteriormente se solicita ip y puerto por donde se está ejecutando el middleware y canal donde se están publicando los datos.

4. Para ejecutar el cliente del Servicio de Acceso a Variables con interfaz visual:

- Situarse en el proyecto Receiver
- Ejecutar ./*Debug*/CSAV
- Posteriormente se introduce ip y puerto por donde se está ejecutando el Servidor de Comm3.
- Se adicionan los identificadores de los puntos que se quieren recibir.

- Se introduce la frecuencia de refrescamiento en milisegundos.
- Solicitar suscripción de puntos (Botón Suscribir Puntos).
- Iniciar la lectura de los valores puntos (Botón Iniciar Lectura).

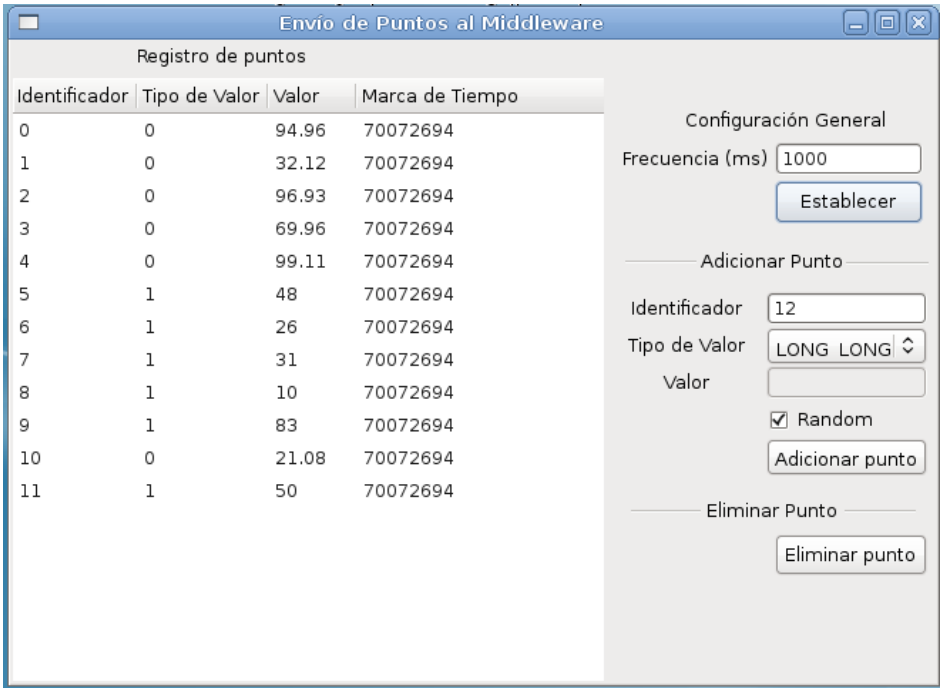

Figura 35: Interfaz gráfica de la aplicación de envío de puntos al middleware.

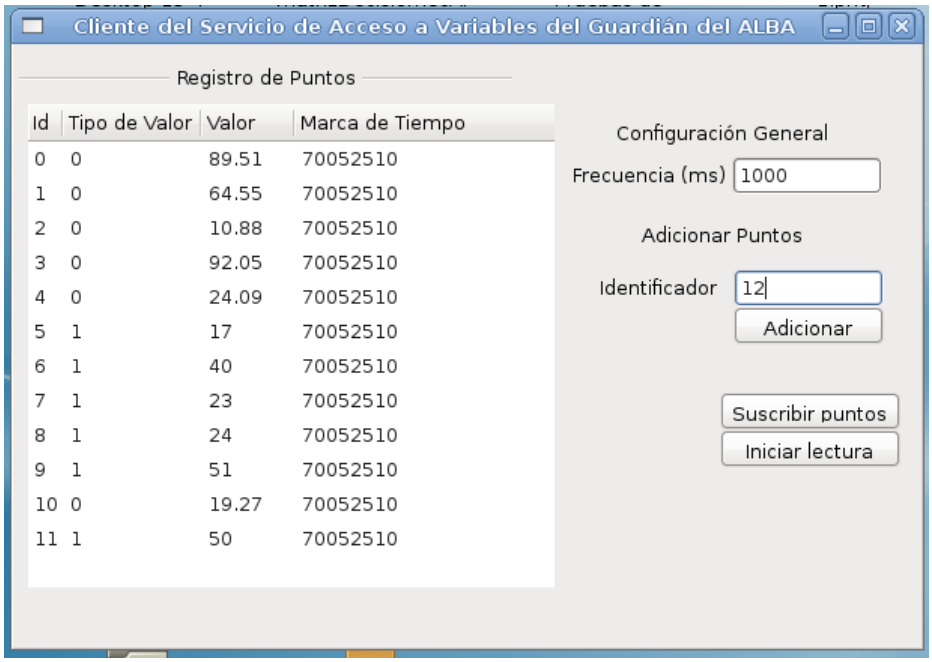

Figura 36: Interfaz gráfica del cliente del Servicio de Acceso a Variables.

GLOSARIO

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: Alternativa Bolivariana para las Américas.

API: *Aplication Program Interface*, en español, Interfaz de Programación de Aplicaciones, es un conjunto de especificaciones de comunicación entre componentes software.

Cliente/Servidor: Esta arquitectura consiste básicamente 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.

Computación *grid*: La computación *grid* o en malla es un nuevo paradigma de computación distribuida en el cual todos los recursos de un número indeterminado de computadoras son englobados para ser tratados como un único superordenador de manera transparente

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.

DDS: Es la especificación para un *middleware* de tipo publicador/suscriptor en sistemas distribuidos. DDS ha sido creado en respuesta a las necesidades de la industria de estandarizar sistemas centrados en datos. La especificación DDS describe dos niveles de interfaces:

- Una DCPS (*Data-Centric Publish-Subscribe*) a nivel inferior que tiene por objeto hacer un reparto de la información de forma eficiente a los recipientes apropiados.
- Una capa superior opcional DLRL (*Data Local Reconstruction Layer*) que permite una integración simple de DDS en la capa de aplicaciones.

En el año 2004 los dos principales proveedores de DDS la americana *Real Time Innovations* (RTI) y la francesa Thales Group se unieron para crear las especificaciones DDS que han sido aprobadas por el OMG.

Entire X: Constituye una implementación comercial de la tecnología DCOM de *Microsoft*.

Eternhet: Protocolo de comunicación entre computadoras de una red local que especifica la forma en que se colocan los datos y se recuperan de un medio de transmisión común.

FTP: *File Transfer Protocol* o Protocolo de Transferencia de Archivos, es un protocolo de red para la transferencia de archivos entre sistemas conectados a una red TCP, basado en la arquitectura cliente-servidor.

GPL: *General Public License*, es una licencia que regula los derechos de autor de los programas de software libre (free software) promovido por la Fundación de Software Libre en el marco de la iniciativa GNU.

GUI: *Graphic User Interface* o Interfaz Gráfica de Usuario, es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

HMI: *Human Machine Interface* o Interfaz Hombre-Máquina, es el aparato de cómputo que presenta los datos recolectados del campo a un operador.

HTML: *Hypertext Markup Language* o Lenguaje de Marcado de Hipertexto, es un lenguaje para crear documentos de hipertexto para uso en el www o intranets, 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. HTTP ha sido usado por los servidores *World Wide Web* desde su inicio en 1993.

HTTPS: *Hypertext Transfer Protocol Secure* o Protocolo Seguro de Transferencia de Hipertexto, es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.

IDE: *Integrated Development Environment* o Entorno de Desarrollo Integrado, es un software que provee facilidades a los desarrolladores en lenguaje de programación y compatibilidad con el sistema operativo o plataforma en que se trabaje.

IDL: *Interface Definition Language* o Lenguaje de especificación de interfaces, se usa como parte de la tecnología CORBA. Ofrece la sintaxis necesaria para definir los métodos que se quieren invocar remotamente.

LAN: *Local Area Network* o Red de área local, es una red que conecta los ordenadores en un área relativamente pequeña y predeterminada (como una habitación, un edificio, o un conjunto de edificios).

MDA: *Model Driven Architecture* o Arquitectura Dirigida por Modelos, es un acercamiento al diseño de software, propuesto y patrocinado por el OMG. Esta fue concebida para dar soporte a la ingeniería dirigida a modelos de los sistemas software. MDA es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos.

.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.

OLE: *Object Linking* and *Embedding*, es un sistema de objeto distribuido y un protocolo desarrollado por Microsoft.

OMG: *Object Management Group* o Grupo de Gestión de Objetos, es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas.

ORB: *Object Request Broker*, constituye el núcleo de la tecnología CORBA.

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.

Publicación/Suscripción: Es un modelo de cooperación multipunto en el que cada mensaje puede tener múltiples consumidores. Las aplicaciones encargadas de el envío de mensajes, los publican generalmente en un *middleware*, que se encarga de redirigirlos, y las aplicaciones que están interesadas en recibir mensajes de un determinado tipo se subscriben, registrando su interés.

SGML: *Standard Generalized Markup* Language o Lenguaje de Marcado Generalizado, es un sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares (ISO) normalizó este lenguaje en 1986. Este lenguaje sirve para especificar las reglas de etiquetado de documentos y no impone en sí ningún conjunto de etiquetas en especial.

SMTP: *Simple Mail Transfer Protocol* o Protocolo Simple de Transferencia de Correo, es un protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos, como teléfonos móviles, entre otros.

Software Libre: Las cuatro reglas esenciales que deben cumplir las aplicaciones para ser consideradas como software libre son:

- Libertad 0: Libertad de ejecutar el programa como quieras.
- Libertad 1: Libertad de estudiar le código fuente y cambiarlo para realizar lo que desees.
- Libertad 2: Libertad de realizar copias y distribuirlas cuando quieras.
- Libertad 3: Libertad de distribuir o publicar versiones modificadas cuando desees.

TCP/IP: Es un conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de ordenadores. En ocasiones se le denomina conjunto de protocolos TCP/IP, en referencia a los dos protocolos que la componen: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP), que fueron los dos primeros en definirse, y que son los más utilizados de la familia.

UDP: *User Datagram Protocol*, es un protocolo del nivel de transporte basado en el intercambio de datagramas, permitiendo el envío de estos a través de la red sin que se haya establecido previamente una conexión.

URI: *Uniform Resource Identifier* o Identificador Uniforme de Recurso, es una cadena corta de caracteres que identifica inequívocamente un recurso.

W3C: Es una iniciativa creada en 1994, en la que participan 400 organizaciones, y que pretende que el *World Wide Web* alcance su máximo potencial desarrollando protocolos comunes que permitan su evolución y aseguren su interoperabilidad.

WAN: Son las siglas de *Wide Area Network* o Red de Área Amplia, una red de ordenadores que abarca un área geográfica relativamente grande. Normalmente consiste en dos o más redes de área local (LANs).

WCF: *Windows Communication Foundation*, es la nueva plataforma de mensajería que forma parte de la API de la Plataforma .NET 3.0

XML: Son las siglas de *Extensible Markup Language*, Lenguaje de marcas extensible, 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