# Universidad de las Ciencias Informáticas

## Facultad #1


**Título: Servicios de Integración con Terceros para el intercambio de Alarmas y Eventos que ocurran en el proceso y el sistema SCADA Guardián del ALBA.**



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

**Autor:** Yailyn Fernández Melgarejo

**Tutor:** Ing. Maikel Pérez Javier

**Ciudad de la Habana**

**Junio 2009**

**"Año del 50 Aniversario del Triunfo de la Revolución"**

*Son vanas y están plagadas de errores las ciencias que no han nacido del experimento,*

*madre de toda certidumbre.*

*Leonardo Da Vinci.*

## DATOS DE CONTACTO

**Tutor:**

Ing. Maikel Pérez Javier

Graduado de Ingeniería 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 el desarrollo de software, específicamente en el desarrollo del Middleware del Sistema de Supervisión, Control y Adquisición de Procesos Industriales (SCADA).

**AGRADECIMIENTOS**

*A mi tutor Maikel Pérez Javier por su ayuda certera y valiosa.*

*A mis profesores que hicieron posible que llegara este día.*

*A mis amigos por su apoyo incondicional que hicieron más placenteros estos años de mi vida.*

**DEDICATORIA**

*A la Revolución Cubana por haberme dado la oportunidad de estudiar y formarme en una Universidad de este tipo.*

*A mis padres, abuelos y a mis hermanos por el apoyo constante, el estímulo permanente y el amor infinito que me sirvieron siempre de acicate para seguir adelante y llegar a este triunfo final.*

*A mi novio Maikel Pérez Javier que con su tiempo, paciencia y comprensión hizo posible que llegara hasta aquí.*

**RESUMEN**

El presente documento presenta el proceso de diseño e implementación de los Servicios para el acceso a la información relacionada con las Alarmas y Eventos que ocurran en el sistema SCADA Guardián del ALBA, a partir de la necesidad de contar con interfaces de comunicación que permitan el intercambio de estos tipos de datos entre las Aplicaciones Externas o Terceros y el Guardián del ALBA.

De esta manera se realiza una investigación acerca de los sistemas SCADA y su integración con diferentes tipos de aplicaciones. Se describe de forma general el estado del arte de los tipos de comunicaciones usadas por los SCADA haciendo énfasis en la comunicación con Sistemas Externos. También se realiza un estudio y análisis de los estándares más utilizados en la actualidad para implementar soluciones de comunicación de este tipo y se evalúan otras tecnologías y tendencias actuales que pueden ser usadas en la solución.

Se selecciona la tecnología Servicios Web para implementar las soluciones de Integración con Terceros para el intercambio de Alarmas y Eventos logrando total interoperabilidad entre las aplicaciones y de esta manera una solución universal o multiplataforma. Se usa la herramienta gSOAP para implementar los servicios web desde C++. Se definen los requisitos funcionales y no funcionales del sistema, se muestra una vista de alto nivel independiente de la tecnología y el modelado de la solución sobre la base del patrón de arquitectura de 3 capas. Finalmente se obtiene la implementación del sistema a partir del diseño propuesto y se realiza un grupo de pruebas de funcionalidad con el fin de validar la solución. Como resultado se obtiene un producto que cumple con los objetivos del Trabajo de Diploma, las expectativas del cliente y que se integra a la versión actual del SCADA Guardián del ALABA.

**PALABRAS CLAVE**

Acceso a Alarmas y Eventos en sistemas SCADA, Comunicación con Terceros, SCADA, Servicios.

# ÍNDICE DE CONTENIDO

## ÍNDICE DE TABLAS

## ÍNDICE DE FIGURAS

## INTRODUCCIÓN

El desarrollo de la sociedad y la satisfacción de las necesidades crecientes de la población requieren del aumento de los niveles de eficiencia, la disminución de los costos y la optimización de los procesos de producción de las industrias. Actualmente, unos de los mecanismos más importantes para la consecución de estos objetivos es la automatización de los procesos que componen la cadena de producción. Automatizar las operaciones resulta una tarea prioritaria para nuestras sociedades que no sólo buscan su desarrollo sino que persiguen la justicia social y el cumplimiento de las leyes fundamentales de la sociedad socialista. Es por ello que en la actualidad han tenido gran popularidad los sistemas SCADA, acrónico de Supervisory Control and Data Acquisition (en español, Supervisión, Control y Adquisición de datos), los cuales tienen como función principal, el monitoreo y control total de los procesos industriales.

Los sistemas SCADA se encuentran en la mitad de la pirámide de control, entre el campo con sus sensores y sus autómatas y las aplicaciones de optimización, simulación y control avanzado sirviendo de puente entre los procesos físicos y las aplicaciones con modelos que pueden aportar la configuración más eficiente de su operación. Por este motivo los sistemas SCADA son una pieza estratégica para cualquier nación que pretenda industrializarse y más aún como lo ha demostrado la historia de nuestros países en su lucha contra el imperio, un elemento sin el que la seguridad de los países del ALBA no podrá concretarse.

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 la soberanía e independencia tecnológica, en el marco de las relaciones entre Cuba y Venezuela y debido a la necesidad de independencia tecnológica existente en este hermano país después del mencionado sabotaje, comenzó en Agosto del año 2006 el desarrollo conjunto de un software SCADA sobre software libre para PDVSA con la participación de Cooperativas y Empresas Venezolanas

y la Universidad de las Ciencias Informáticas (UCI), 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.

La Facultad 5 asumió la tarea y comenzó el desarrollo del sistema SCADA actualmente conocido como ***"Guardián del ALBA"***, Sistema de Supervisión y Control de Procesos Industriales para los Pueblos del ALBA que también fue la génesis de la creación del Polo de Hardware y Automática. Con la implementación de los principales módulos o partes del sistema definidas en sus inicios y los avances generales alcanzados, el desarrollo del 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.

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, Dot Net, CORBA, etc. han permitido que los 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. La comunicación con aplicaciones externas es una de las funcionalidades que mayor impacto posee en los sistemas actuales, debido a que posibilita el intercambio de información entre las aplicaciones de gestión y las de control del proceso. (Herrera Vázquez, y otros, 2008)

Identificada la necesidad de proveer al Guardián del ALBA de un mecanismo que le permitiera intercambiar información con aplicaciones externas, se comenzó el desarrollo del subsistema de Comunicación con Terceros con el objetivo de exponer las funcionalidades del SCADA en forma de servicios permitiendo así el acceso de agentes externos a la información manejada por el sistema.

El nivel de importancia de la Integración o Comunicación con Terceros radica en la utilización que le pueden dar los sistemas externos a la información recolectada, normalmente para operaciones de gestión. De forma general la información puede ser útil para:

- Obtener información de la producción a partir el acceso a la información de variables del sistema y del proceso.
- Obtener información del funcionamiento y estado de la planta a partir de la adquisición de información relacionada con las alarmas y eventos del proceso y el sistema.

- Realizar cálculos estadísticos a partir de la información del SCADA y la generación de informes.

Uno de los tipos de datos más importantes en un sistema SCADA y de interés para las aplicaciones externas, son las alarmas y los eventos, normalmente manejado todo como eventos cuando de Comunicación con Terceros se trata, siendo la comunicación de alarmas y eventos lo que se desea medir. A partir de esta información es posible tomar decisiones correctivas del estado del sistema utilizando técnicas de minería de los datos y sus correlaciones.

Por la importancia que tiene proveer interfaces de comunicación que permitan una integración global entre todos los objetos de negocio, tanto de supervisión, control, como aplicaciones gerenciales y especialmente la gestión de la información relacionada con las Alarmas y Eventos del Guardián del ALBA, existe la necesidad de brindar una solución de comunicación distribuida que permita, a agentes externos a nuestro sistema, el acceso a la información relacionada con estos tipos de datos.

Atendiendo a la situación problémica existente, constituye un **Problema Científico**, ¿Cómo permitir a los sistemas externos, el acceso a la información relacionada con las alarmas y eventos que ocurran en el proceso y el sistema SCADA Guardián del ALBA?

Para dar respuesta al problema existente es necesario plantearse el siguiente **Objetivo General:** Implementar Servicios de Integración con Terceros que permitan el intercambio de información relacionada con Alarmas y Eventos que ocurran en el proceso y el sistema SCADA Guardián del ALBA.

Del problema científico anterior y el objetivo general planteados, se decide central la investigación en la comunicación entre los sistemas SCADA y los sistemas externos como **Objeto de Estudio** y en la integración con terceros para el intercambio de información relacionada de alarmas y eventos como **Campo de Acción.** Como **idea a defender** se tiene que:

Si se implementan los Servicios de Integración con Terceros para el intercambio de información relacionada con Alarmas y Eventos del sistema SCADA Guardián del ALBA, entonces se proveerá a las aplicaciones externas de interfaces de comunicación con funcionalidades para acceder a la información relacionada con estos datos, siempre dependientes de un mecanismo de suscripción.

Se definen las siguientes tareas de investigación para lograr mejor nivel de precisión en la investigación y desarrollo del tema:

- Estudiar y analizar temas relacionados con sistemas SCADA, comunicación con sistemas externos y servicios de integración con terceros.
- Analizar las tendencias y tecnologías actuales que permitan desarrollar los Servicios de Integración con Terceros para el intercambio de información relacionada con Alarmas y Eventos del sistema SCADA Guardián del ALBA.
- Modelar a través del diseño de clases las funcionalidades relacionadas con los Servicios de Integración con Terceros para el intercambio de información relacionada con Alarmas y Eventos del sistema SCADA Guardián del ALBA.
- Implementar las funcionalidades relacionadas con los Servicios de Integración con Terceros para el intercambio de información relacionada con Alarmas y Eventos del sistema SCADA Guardián del ALBA.
- Probar las funcionalidades implementadas por los Servicios de Integración con Terceros para el intercambio de información relacionada con Alarmas y Eventos del sistema SCADA Guardián del ALBA.

Para el cumplimiento de las tareas científicas se emplean diferentes Métodos y Técnicas que nos facilitarán la búsqueda y el procesamiento de la información, de la manera que se exponen a continuación.

**A nivel Teórico:**

- *Histórico-Lógico:* Empleado durante la investigación de los antecedentes y las tendencias actuales referidas a la comunicación con sistemas externos y en la profundización de los conceptos, términos y vocabulario propio del objeto de estudio y el campo de acción.
- *Analítico-Sintético:* Utilizado durante el análisis y selección de las tecnologías para implementar los Servicios de Integración con Terceros.
- *Inductivo-Deductivo:* Utilizado para la revisión, estudio y justificación de la tecnología seleccionada para desarrollar los Servicios de Integración con Terceros. Por medio de análisis individuales de las tecnologías se llega a una propuesta general de características que se utilizan en la inferencia de la tecnología más adecuada.

- *Modelación:* Utilizado durante la elaboración de los modelos encargados de mostrar la vista lógica que se utilizará para el desarrollo de los servicios.

**A nivel Empírico:**

- *Entrevistas:* Se aplica a asesores y compañeros del grupo técnico del Guardián del ALBA sobre diferentes temas relacionados con la investigación.

El presente trabajo consta de Introducción, tres capítulos, Conclusiones, Recomendaciones, Referencias Bibliográficas, Bibliografía, Anexos y Glosario de Términos. Los capítulos se estructuran de la siguiente manera:

**Capítulo 1. Fundamentación Teórica:** describe los principales conceptos relacionados con sistemas SCADA, comunicación de sistemas externos, servicios de terceros, haciendo énfasis en los Servicios de Integración con Terceros para el intercambio de Alarmas y Eventos. Además se realiza un análisis y evaluación de las tendencias y tecnologías actuales haciendo énfasis en los estándares más utilizados para implementar soluciones de este tipo.

**Capítulo 2. Modelado de la Solución:** expone las funcionalidades o requisitos del sistema que son objeto de solución de acuerdo a los servicios que se desean implementar. Muestra una vista de alto nivel del sistema, se selecciona la(s) tecnología(s) adecuadas para implementar los servicios y finalmente se modela la solución a través del diseño de clases.

**Capítulo 3. Construcción de la Solución Propuesta:** describe el modelo de implementación, el estilo de código utilizado, las diferentes interfaces y la implementación de los componentes del sistema, así como los resultados de pruebas de funcionalidad para validar la implementación de los servicios.

*Por último se arriba a una serie de conclusiones y recomendaciones válidas para desarrollos futuros relacionados con el tema del trabajo. De modo general, el trabajo girará en torno a lo expresado hasta el momento.*

# CAPITULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se muestran los conceptos fundamentales relacionados con los sistemas SCADA: su descripción, funcionalidades y componentes, resaltando como se comunican con los sistemas externos existentes. Se describen los servicios de integración con terceros más utilizados por los sistemas SCADA en la actualidad enfatizando en el Servicio de Integración con Terceros para el intercambio de información relacionada con Alarmas y Eventos. Por último se exponen las características principales de los estándares y tecnologías más utilizados en la actualidad para implementar los Servicios abordados en el trabajo.

## 1.1. Sistemas SCADA

Los sistemas SCADA, comprenden todas aquellas soluciones de aplicación que requieren de la captura de información de un proceso o planta industrial, la cual es utilizada para realizar una serie de análisis o estudios con los que se pueden obtener valiosos indicadores que permitan una realimentación sobre un operador o sobre el propio proceso. **(Herrera Vázquez, 2008)**

Desde el punto de vista de Software, un SCADA, se puede definir como una aplicación diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autónomas programables, etc.) y controlando el proceso de forma automática desde la pantalla del ordenador. Además provee de toda la información que se genera en el proceso productivo a diversos usuarios tanto del mismo nivel como de otros supervisores dentro de la empresa: control de calidad, supervisión, mantenimiento, etc. **(Ferrari, 2005)**

Se dice que cualquier software dedicado a la supervisión y/o control puede considerarse un SCADA.

## 1.2. Descripción general de un SCADA

En la actualidad los sistemas SCADA son usados por un sin número de aplicaciones de las que se pueden mencionar: el control de oleoductos, sistemas de transmisión de energía eléctrica, yacimientos de gas y petróleo y control de todos los procesos de esta rama, redes de distribución de gas natural, subterráneos, generación energética**,** sistemas de distribución de agua, entre otros.

Cada uno de los ítems del SCADA (Supervisión, Control y Adquisición de datos) involucran muchos subsistemas, por ejemplo, la adquisición de los datos puede estar a cargo de un PLC (Controlador Lógico Programable) el cual toma las señales y las envía a las estaciones remotas usando un protocolo determinado, otra forma podría ser que una computadora realice la adquisición vía un hardware especializado y luego esa información la transmita hacia un equipo de radio vía su puerto serial, y así existen muchas otras alternativas.

Las tareas de Supervisión y Control generalmente están más relacionadas con el software SCADA, en él, el operador puede visualizar en la pantalla del computador de cada una de las estaciones remotas que conforman el sistema, los estados de ésta, las situaciones de alarma y tomar acciones físicas sobre algún equipo lejano, la comunicación se realiza mediante buses especiales o redes LAN. Todo esto se ejecuta normalmente en tiempo real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos. La figura # 1 muestra la interacción de un operador con las consolas o pantallas de un sistema SCADA. **(SCADA, 2008)**

*Figura 1: Interacción del Operador con el Sistema SCADA*

### 1.2.1. Funcionalidades principales de un SCADA

- *Supervisión remota de instalaciones y equipos:* Permite al operador conocer el estado de desempeño de las instalaciones y los equipos alojados en la planta, lo que permite dirigir las tareas de mantenimiento y estadística de fallas.
- *Control remoto de instalaciones y equipos:* Mediante el sistema se puede activar o desactivar los equipos remotamente (por ejemplo abrir válvulas, activar interruptores, prender motores, etc.), de manera automática y también manual. Además es posible ajustar parámetros, valores de referencia, algoritmos de control, etc.
- *Procesamiento de datos:* El conjunto de datos adquiridos conforman la información que alimenta el sistema, esta información es procesada, analizada, y comparada con datos

anteriores, y con datos de otros puntos de referencia, dando como resultado una información confiable y veraz.

- *Visualización de gráfica dinámica:* El sistema es capaz de brindar imágenes en movimiento que representen el comportamiento del proceso, dándole al operador la impresión de estar presente dentro de una planta real. Estos gráficos también pueden corresponder a curvas de las señales analizadas en el tiempo.

- *Generación de reportes:* El sistema permite generar informes con datos estadísticos del proceso en un tiempo determinado por el operador.

- *Representación se señales de alarma:* A través de las señales de alarma se logra alertar al operador de que está frente a una falla o de la presencia de una condición perjudicial o fuera de lo aceptable. Estas señales pueden ser tanto visuales como sonoras.

- *Almacenamiento de información histórica:* Se cuenta con la opción de almacenar los datos adquiridos, esta información puede analizarse posteriormente, el tiempo de almacenamiento dependerá del operador o de los requisitos del sistema.

- *Programación de eventos:* Esta referido a la posibilidad de programar subprogramas que brinden automáticamente reportes, estadísticas, gráfica de curvas, activación de tareas automáticas, etc.

### 1.2.2. Componentes principales de un Sistema SCADA

- *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.

- *Múltiples Unidades de Terminal Remota (RTU):* Se conectan al equipo físico. Leen los datos de un interruptor o válvula, o mediciones como temperatura, flujo, o presión. Pueden realizar control automatizado.
- *Estación Maestra / Computadoras HMI:* Presenta la información al operador. Incluye monitoreo, control de lazo abierto, generación de alarmas, registro de datos, históricos y seguridad.
- *Comunicaciones:* Se encarga de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre ésta y el resto de los medios informáticos de gestión. En este punto el SCADA puede tener mecanismos de comunicación entre sus módulos, comunicación con los dispositivos de campo y mecanismos de **comunicación con aplicaciones externas** (Integración con Terceros).

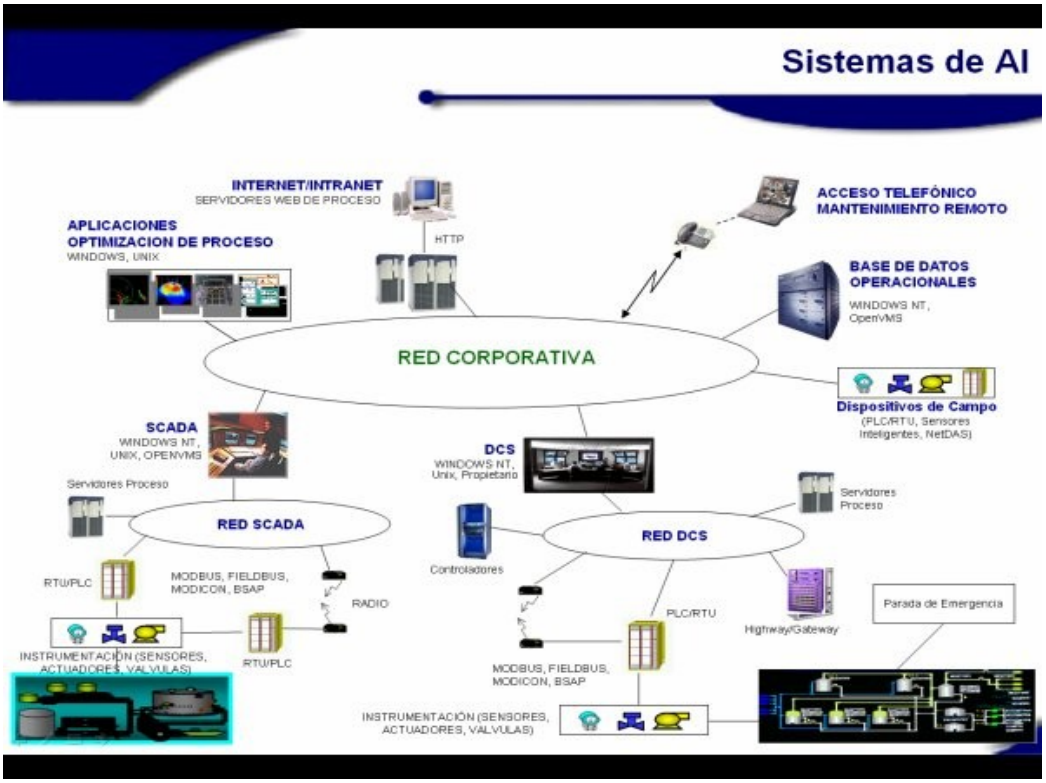


*Figura 2: Interrelación del SCADA y otros Sistemas*

### 1.2.3. Comunicaciones en los SCADA

Los sistemas SCADA necesitan comunicarse de varias maneras, con los dispositivos de campo, comunicación entre sus subsistemas o módulos y con aplicaciones externas incluyendo otros SCADA.

Para realizar el intercambio de información entre los dispositivos de campo y las estaciones del SCADA es necesario un medio de comunicación. Cada fabricante de equipos para sistemas SCADA emplean diferentes protocolos de comunicación y no existe un estándar para la estructura de los mensajes, sin embargo existen estándares internacionales que regulan el diseño de las interfaces de comunicación entre los equipos del sistema SCADA y equipos de transmisión de datos.

Un protocolo de comunicación es un conjunto de reglas y procedimientos que permite a las unidades remotas y central, el intercambio de información. Los sistemas SCADA hacen uso de los protocolos de las redes industriales.

Normalmente se usan manejadores (driver) que hablan el idioma de los autómatas colocados en el campo para transmitir la información del campo hacia el SCADA o se utiliza el propio driver OPC (*OLE for Process Control*).

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.

En el caso de las comunicaciones internas del SCADA por lo general se utiliza un Middleware o software de comunicación entre aplicaciones, definido como la capa de software que se encuentra por encima de los niveles físicos y de transporte y por debajo de las capas de gestión y aplicaciones de usuario y su función principal radica en establecer un método de comunicación entre las aplicaciones que se encuentran ejecutándose en cualquier punto del sistema, siendo su localización transparente a las demás. (Herrera Vázquez, y otros, 2008)

Al remitirse a la pirámide de automatización de un sistema SCADA, se puede aclarar la ubicación del Middleware en el mismo, a partir de la figura # 2.

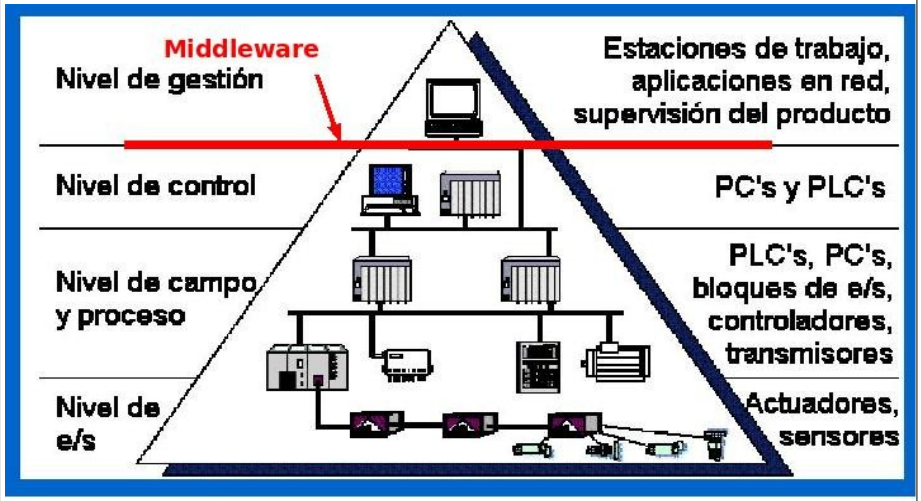


*Figura 3: Ubicación del Middleware en un Sistema SCADA*

En la comunicación entre el sistema SCADA y las aplicaciones externas también se intercala un Middleware que permite elevar la transparencia de los accesos a los servicios. Dentro de los retos más sobresalientes que debe suplir ese Middleware se encuentran: **(Herrera Vázquez, y otros, 2008)**

- La comunicación, frecuentemente necesitará enviar parámetros complejos, tal como registros, arreglos o cadenas de caracteres.
- La codificación de diferentes tipos de datos, puede ser utilizada si los clientes y los objetos no son desplegados en la misma plataforma de hardware y si ellos no son escritos en el mismo lenguaje de programación.
- Los parámetros y los valores de retorno pueden referirse a otros objetos distribuidos.
- Los desarrolladores y administradores pueden tener que implementar activaciones de objetos, esto es, la habilidad de un componente servidor responder a un requerimiento de un cliente aun cuando este no se encuentre actualmente corriendo en un proceso del sistema.
- La seguridad de tipos denota la garantía que una operación requerida por un cliente es actualmente implementada por un componente servidor y que los parámetros actuales suministrados por el cliente deben coincidir con la definición del servicio. Alcanzar la seguridad de tipos manualmente es una tarea absolutamente tediosa y propensa a errores, particularmente si los componentes del cliente y el servidor son desarrollados por diferentes partes.

- En ocasiones existen cualidades de servicio que no pueden ser garantizadas por el nivel de red.
- Intercambio de información en redes y plataformas son heterogéneas.

### 1.3. SCADA y Comunicación con aplicaciones externas

En términos de este documento, se define la **Comunicación con Terceros** como un modelo de comunicación orientado a servicios, los cuales pueden ser utilizados por un conjunto de aplicaciones externas al sistema SCADA.

Aunque las definiciones entre los términos de sistemas SCADA y Sistemas de Control Distribuidos (DCS por sus siglas en inglés), son cada vez más similares aún persiste un consenso de autores que indican que a diferencia de los DCS, el lazo de control, en los SCADA, es GENERALMENTE cerrado por el operador. Los DCS se caracterizan por realizar las acciones de control en forma automática. Hoy en día es fácil hallar un sistema SCADA realizando labores de control automático en cualquiera de sus niveles, aunque su labor principal sea de supervisión y control por parte del operador.

Dentro de las principales diferencias entre ambos enfoques se resalta que, por sus restricciones de tiempo real, los DCS sean limitados geográficamente a las áreas de proceso y sus componentes interconectados vía redes locales o conexión directa. Sin embargo los sistemas SCADA presentan una distribución más global expandiéndose por redes que pueden incluir trasmisiones en Internet. **(Herrera Vázquez, y otros, 2008)**

Estas diferencias cada día se estrechan más con el uso de las nuevas tecnologías, sin embargo han primado hasta lo días de hoy y han sido impactado en las técnicas y tecnologías utilizadas para la comunicación con aplicaciones externas a ambos sistemas; priorizándose las comunicaciones binarias de los DCS basadas en DCOM, en Windows, y CORBA en ambientes Unix y una mezcla de técnicas de intercambio vía texto como formatos XML y/o binarios en los SCADA.

Las aplicaciones de terceros pueden ser los DCS, aplicaciones personalizadas y los propios sistemas SCADA, pero 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:** (Sepúlveda, 2006)

Un producto MES se alimenta en tiempo real y en línea de datos provenientes de otros sistemas (historiadores de proceso, HMI/SCADA, servidores OPC, bases de datos relacionales como Oracle, SQL Server, etc.), y los convierte en información para la toma de decisiones. Algunos de estos productos MES entregan sus resultados también a otros software como los sistemas ERP. La siguiente figura nos permite visualizar el flujo de la información, aunque el límite entre cada nivel es algo difuso (algunas tareas pueden ser compartidas).

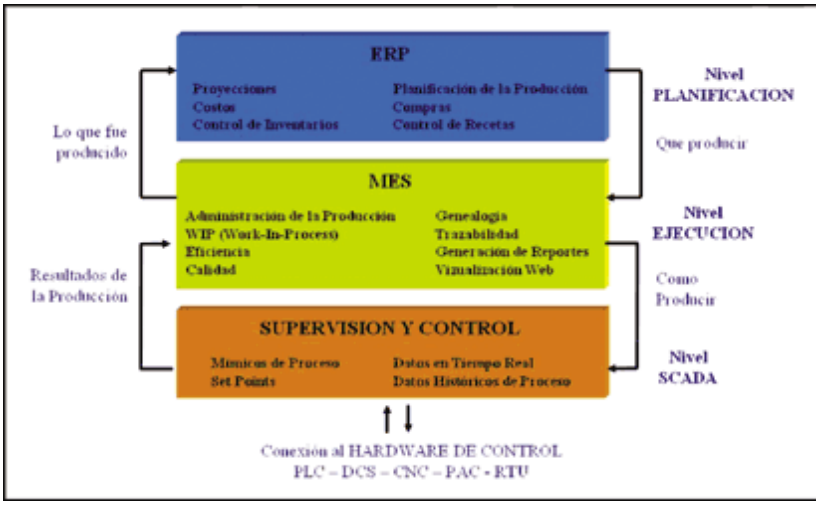


*Figura 4: Niveles Superiores del SCADA utilizando los conceptos de ERP y MES*

**Sistemas de Planificación de Recursos de la Empresa (ERP):** (Apter, 2009)

Los **sistemas ERP** son sistemas de gestión de información que tienen como función integrar y automatizar las actividades de negocio asociadas con los aspectos operativos o productivos de una empresa. Se nutren de la información brindada por sistemas MES para optimizar la gestión de la producción.

### 1.3.1. Servicios de integración con terceros

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 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 pueden proveerse a terceros por un sistema SCADA son los siguientes:

- Intercambio de información relacionada con variables del proceso y del sistema (puntos).
- Intercambio de información relacionada con alarmas y eventos.
- Interacción a través de comandos.
- Intercambio de información asociada a la Visualización Hombre Máquina. HMI. Visualización de los despliegues de proceso.
- Servicios de Seguridad.

A continuación se describe el servicio de integración con terceros para el intercambio de información relacionada con alarmas y eventos.

### 1.3.2. Alarmas y Eventos en un sistema SCADA

En un SCADA las alarmas y los eventos se pueden asociar con los siguientes conceptos:

**Alarmas:**

El sumario de alarmas es una interfaz gráfica que concentra las condiciones de procesos críticas, medias y baja presentes en el sistema y cuyo objetivo primordial es guiar al operador a la detección del origen de la falla y supervisar la ejecución de las medidas de corrección automatizada o manual. **(Charrouf, 2009)**

Una condición de proceso anormal, declaración automática cuando una condición de proceso anormal ocurre, dispositivo que llama la atención a la existencia de una condición de proceso anormal.

Evento generado por un dispositivo o una función que señala la existencia de una condición anormal a través de un cambio discreto audible o visible, o ambas, que requiere su atención inmediata. **(Villarreal, 2009)**

**Eventos:**

En términos de informática, particularmente en el lenguaje de los sistemas, "evento" se denomina a los sucesos generados por el sistema. Según su tipo pueden ser procesados en su totalidad por el mismo sistema o a través de mecanismos que corren bajo su propio control en forma de mensajes. **(Charrouf, 2009)**

Por lo general, los eventos se dividen en dos grandes categorías:

- Eventos de usuario
- Eventos de Sistema

Conocemos como eventos de usuarios, aquellas acciones que tienen su origen en el usuario del programa. Como ejemplos de este tipo serían acciones de pulsar un botón del ratón, tecla, etc.

En el caso de eventos de sistema, son aquellos que se inician internamente o son invocados por mecanismos internos. Ejemplos de este tipo pueden ser la generación de eventos a intervalos de tiempo predeterminados, redibujado de objetos gráficos en ventanas, etc.

### 1.3.3. Servicios de Integración con Terceros para intercambio de Alarmas y Eventos

El flujo de alarmas y eventos a través de todas las capas de la pirámide de control y supervisión es uno de los temas más estudiados en la actualidad ya que a través de él es posible tomar decisiones correctivas del estado del sistema utilizando técnicas de minería de los datos y sus correlaciones.

Aunque una alarma o un evento para una aplicación externa es lo mismo que para un SCADA, en ocasiones, ambos tipos de datos son tratados como eventos.

En el caso de las alarmas existe un consenso en catalogarlas como eventos especiales que requieren de una atención adecuada y está relacionada con manifestaciones anómalas del proceso y de comportamiento del sistema como un todo.

La mayoría de las especificaciones integran ambos conceptos en el mismo dándole un toque integral al manejo de alarmas y eventos utilizando atributos de severidad y prioridades al manejo de las mismas.

De forma general el servicio de alarmas y eventos provee: **(Departamento de Ingeniería de Sistemas)**

- Mecanismos para informar condiciones de alarma y eventos.
- Interfaces para que los clientes conozcan las alarmas que soporta el servidor y el estado actual.

El servicio es necesario para dar información de:

- Alarmas sobre datos de sensores: presión, temperatura, etc.
- Eventos de sistema, seguridad, comunicaciones, etc.
- Alarmas sobre parámetros de control: *start, stop, open*, etc.
- Actualizaciones sobre estado de información (HW,SW)
- Completar secuencias "*batch*".
- Otro tipo de eventos que no están en el servidor.

Descripción de los eventos:

1. Acontecimiento reseñable. Asociado o no a una condición.
2. Se distinguen tres tipos de eventos:

- *Condition Related*: asociados a una condición.
- *Tracking Related*: no asociados con alarmas, pero implican interacción cliente/servidor.
- Simple: Usados para codificar errores materiales.

Descripción de las alarmas:

1. Condición anormal a un proceso. Asociada o no a una condición.

Adicional a los eventos generados por usuarios y sistema, se tienen los generados por el proceso que está bajo supervisión y control, ejemplo de este tipo sería la activación de alarmas, el retorno a normal de las variables, entre otros.

La información contenida en las alarmas y eventos permite conocer el estado del sistema y posibilita la toma de acciones por parte de aplicaciones de gestión de negocio. Las alarmas y eventos contienen un conjunto de atributos, dentro de los que se destacan:

Definición de alarmas:

- **Identificador:** identificador de la alarma que la hace única.
- **Recurso Asociado:** identificador del recurso asociado a la alarma.
- **Estampa de tiempo:** instante de tiempo en que sucedió la alarma.
- **Tipo de Alarma:** identifica si es una alarma de sistema, proceso, usuario, etc.
- **Severidad y/o Prioridad:** prioridad de dicha alarma.
- **Fuente de la Alarma:** Quién o Qué genera la alarma.

Definición de eventos:

- **Identificador:** identificador del evento que lo hace único.
- **Descripción:** explica porque sucedió el evento.
- **Identificador del Recurso Asociado:** identificador del recurso asociado al evento.
- **Tipo de Recurso:** permite saber a qué tipo de recurso está asociado el evento.
- **Estampa de tiempo:** instante de tiempo en que sucedió el evento.
- **Tipo de Evento:** identifica el tipo de evento.
- **Fuente del Evento:** Quién o Qué genera el evento.
- **Valor Previo:** identifica el valor del recurso antes de suceder el evento.
- **Valor Actual:** identifica el valor del recurso después de suceder el evento.

El servicio puede proveer de un mecanismo de aviso de las alarmas y eventos, comúnmente se utilizan servicios de seguridad y suscripción de los clientes a las alarmas variables para optimizar la comunicación y asegurar la data.

Las alarmas y eventos están muy relacionados con los recursos del sistema SCADA es por eso que pueden ser un servicio perfectamente integrado a los demás.

### 1.4. Estándares más utilizados por los SCADA para la comunicación con sistemas externos

En la actualidad los Servicios de Integración con Terceros para el intercambio de Alarmas y Eventos puede ser desarrollado por medio de estándares o tecnologías definidas por fundaciones internacionales como son el caso de *OPC Foundation*, que dirige el consorcio de un grupo de especificaciones para el Control de Procesos, OPC y *Object Management Group* (OMG), que define estándares de comunicación para la automatización como *Data Adquisition from Industrial Systems (*DAIS), y su variante para históricos HDAIS. También por esquemas como los descritos por las arquitecturas OPC UA (*OPC Unified Architect*) de *OPC Foundation* y *Archestra de Invesys.* Los estándares más utilizados se explican a continuación:

- *OPC Alarms & Events,* OPC AE, donde se integran alarmas y eventos en un enfoque único y se portan en una plataforma basada en comunicación Microsoft DCOM.
- *OPC Unified Architect*, OPC UA, enfoque a Servicios de todas las especificaciones de OPC, independiente de la plataforma y la implementación existente es basada en Dot.Net.

#### 1.4.1. Descripción de OPC

OPC es un estándar de comunicación en el campo del control y supervisión de procesos. Este estándar permite que diferentes fuentes envíen datos a un mismo servidor OPC, al que a su vez podrán conectarse diferentes programas compatibles con dicho estándar. De este modo se elimina la necesidad de que todos los programas cuenten con drivers para dialogar con múltiples fuentes de datos, basta que tengan un driver OPC.

El OPC corresponde a 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.

Hasta la fecha *OPC Foundation* ha definido un grupo de estándares: **(OPC Foundation, 2009)**

- Acceso a Datos, *OPC Data Access* (OPC DA).
- Alarmas y Eventos, OPC AE.
- Seguridad. *OPC Security*.

- Comandos, *OPC Commands.*
- Históricos, *OPC HDA.*
- Acceso a Datos XML. OPC XML DA.
- Más recientemente, OPC UA.

**Ventajas de OPC**

- OPC es el conjunto de estándares más difundido en el mundo para el intercambio de información entre aplicaciones destinadas al control y supervisión de procesos.
- Define el conjunto de estándares necesarios para implementar cada uno de los servicios especificados (Acceso a datos, alarmas, eventos y las especificaciones para Comandos y Seguridad).
- Los fabricantes de hardware sólo tienen que hacer un conjunto de componentes de programa para que los clientes los utilicen en sus aplicaciones.
- Los fabricantes de software no tienen que adaptar los drivers ante cambios de hardware.

**Desventajas de OPC**

- Tiene un grupo de especificaciones, muy acopladas a la tecnología DCOM de Microsoft, lo que limita su utilización en otras plataformas, aunque pueden utilizarse pasarelas que sirvan como puente a un cambio de tecnología que a pesar de lograr la integración, debilita la eficiencia y fiabilidad de la solución debido a la adición de más componentes, en el sistema, propensos a fallos.
- A pesar de que existe una especificación de OPC para seguridad la mayoría de los clientes y servidores existentes no la implementan lo que limita la cantidad de aplicaciones que podrían intercambiar datos con el sistema, a un número muy reducido.
- No es una solución del todo universal porque obliga a que la comunicación de los terceros con el servidores se vía OPC, es decir que los terceros deben ser instancias de clientes OPC para tener acceso a los recursos del servidor.

*La evolución de OPC ha estado muy ligada al desarrollo de las tecnologías de programación. Desde la aparición de DCOM,nhasta las más recientes orientadas a servicios WEB*.

### 1.4.2. Descripción de OPC AE

*OPC Alarm and Event* (OPC AE) se trata de una especificación de la Fundación OPC que define cómo se pueden transferir alarmas y eventos en tiempo real entre una fuente de datos y un destino determinado.

**¿Cuál es la diferencia entre *OPC Data Access* y *OPC Alarm and Event*?**

*OPC Data Access* (OPC DA) aborda la necesidad del intercambio en tiempo real de los datos en el control de procesos utilizando una interfaz de comunicación estándar. En el control de procesos los datos en tiempo real incluyen la información analógica y discreta que proviene de los equipos de campo, sistemas, y los operadores. La especificación de alarmas y eventos de OPC se refiere a la necesidad de intercambiar en tiempo real alarmas y eventos utilizando un estándar de comunicación. Las alarmas y los eventos en tiempo real incluyen las condiciones que surgen de los equipos de campo y los sistemas y las acciones realizadas por los sistemas y los operadores durante el control de procesos de la planta. **(OPC Foundation, 2002)**

#### 1.4.2.1. Alarmas y eventos OPC

En el control de procesos de la industria, las condiciones de alarmas y eventos se utilizan para describir los sucesos en una planta que tienen un cierto significado. A continuación los conceptos principales: **(Pérez, 2007)**

*Alarmas:* una alarma es una condición anormal sobre el proceso y corresponde a una condición específica. Una alarma puede estar asociada o no a alguna condición y en caso de no estarlo simplemente representar un estado interno, que expresa algo significativo en el contexto del servidor.

*Condiciones:* Una condición es un estado nombrado en el servidor o en uno de sus objetos de interés para un cliente. Por ejemplo el *tag FIC 101* puede tener las condiciones "*LevelAlarm*" asociada a

él. Una condición puede tener asociada a ella sub condiciones. Las condiciones pueden ser de estado único o multiestado (*HighAlarm, HighHighAlarm, LowAlarm, LowLowAlarm*). Además pueden tener asociados tres estados variables:

- *Enabled*
- *Active* y
- *Acknowledge*

*Eventos:* Un evento es un acontecimiento detectable el cual es significativo para el servidor de eventos OPC, el dispositivo al que represente y sus clientes OPC. El evento puede estar asociado o no a una condición. Por ejemplo las transiciones dentro de la condición "*LevelAlarm*" y la vuelta a la normalidad de la condición son eventos asociados a condiciones, sin embargo los cambios en la configuración del sistema y los errores del sistema son ejemplos de eventos que no están relacionados con condiciones específicas.

#### 1.4.2.2. Clientes y servidores de Eventos OPC

La especificación de OPC AE describe los objetos y las interfaces que son implementadas por los servidores de eventos OPC, y que proporcionan los mecanismos para que los clientes OPC notifiquen la ocurrencia de condiciones de alarmas y eventos. Estas interfaces también brindan servicios que permiten a los clientes OPC determinar que eventos y condiciones soporta el servidor, y obtener su estado actual**. (OPC Foundation, 2002)**

La misma está destinada para ser usada con sistemas de automatización de procesos que generan alarmas y eventos. Un servidor alarmas y eventos OPC está escrito para exponer las alarmas y eventos presentes en el sistema. Este no crea las alarmas y eventos, sólo informa las alarmas y eventos previamente definidos en el sistema usando una interfaz común de comunicación. Una vez definidas, las alarmas y eventos en el sistema se generan de forma automática, sobre la base de las condiciones de funcionamiento y las acciones de rendimiento en el proceso de la planta. El servidor de alarmas y eventos OPC captura automáticamente las alarmas y eventos, y los pone a disposición de cualquier aplicación cliente que esté interesado en esta información.

Existen varios tipos de servidores OPC para alarmas y eventos. Algunos de los principales tipos soportados por OPC AE son los siguientes:

- Componentes que pueden detectar alarmas y/o acontecimientos (eventos) e informar a uno o más clientes.
- Componentes que pueden recolectar información de eventos y alarmas de múltiples fuentes (ya sea por la suscripción de otros servidores de alarma y de eventos OPC o por detección de alarmas y eventos propiamente) y facilitar dicha información a uno o más clientes.

Los clientes de los servidores de alarmas y eventos OPC son generalmente los componentes que se suscriben, muestran, procesan, y recolectan información de alarmas y eventos. Los clientes pueden incluir (pero no se limitan).

- Operador de estaciones.
- Componentes de *logging* de alarmas y eventos.
- Subsistemas de administración de alarmas y eventos.

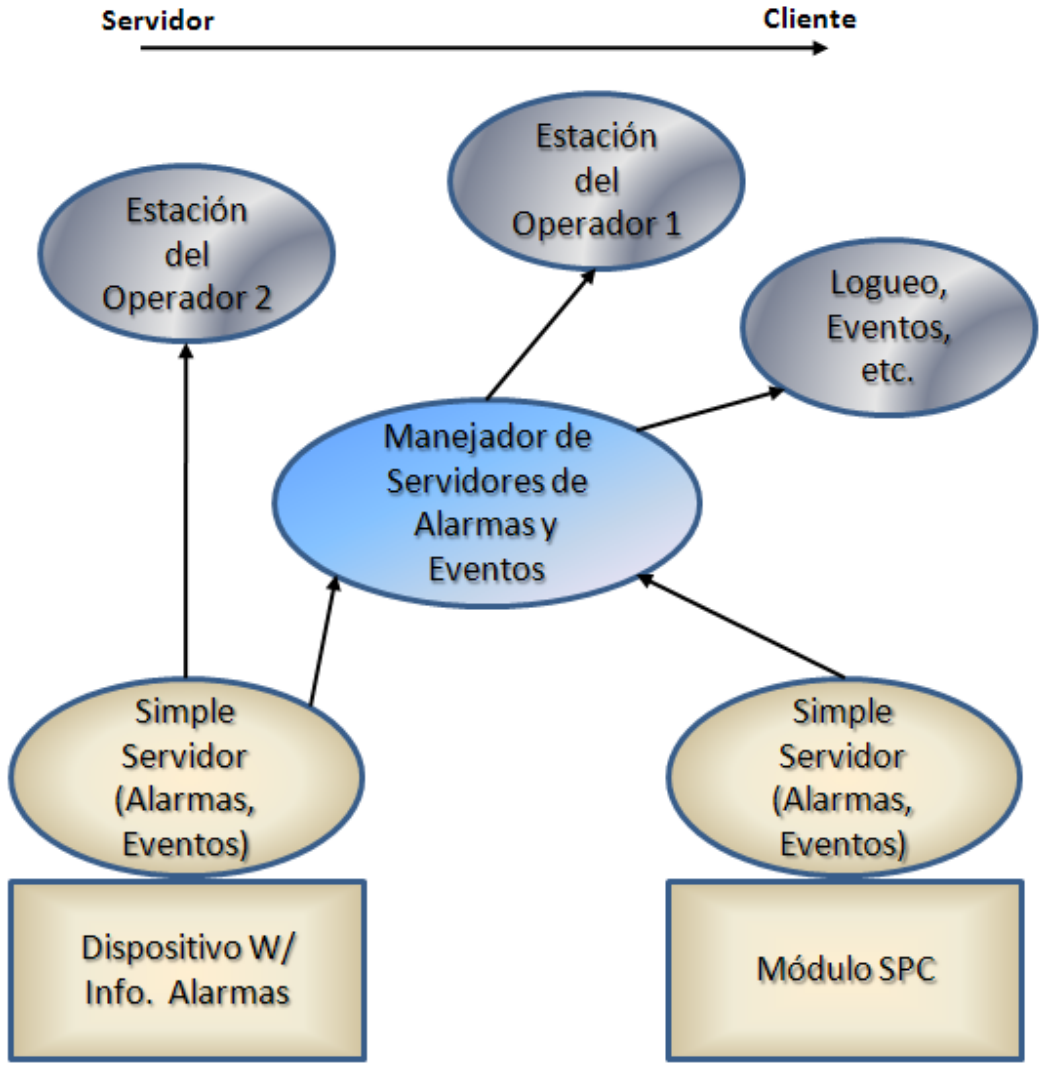


*Figura 5: Interacción entre Clientes y Servidores OPC AE*

#### 1.4.2.3. Ventajas de OPC AE

- Como una de las especificaciones de OPC este estándar es el más utilizado en la actualidad para el intercambio de información relacionada con alarmas y eventos entre SCADA y aplicaciones de terceros.
- Permite la obtención y envío de alarmas y eventos en tiempo real.

- Un cliente de alarmas y eventos OPC es un software que tiene implementada la especificación estándar de alarmas y eventos y que puede comunicarse con cualquier servidor OPC de alarmas y eventos. Al ser OPC un protocolo abierto, cualquier Cliente OPC puede conectarse con cualquier Servidor OPC sin importar desarrolladores ni fabricantes.

#### 1.4.2.4. Uso del Estándar OPC AE

En la actualidad existen en el mercado, gran cantidad de sistemas distribuidos que utilizan el estándar OPC AE, en gran parte debido a que este permite la obtención de información relacionada con condiciones anómalas surgidas en los procesos y el sistema. El proyecto *Archertra Invensys* define una Arquitectura para desarrollar SCADA e implementa el intercambio de alarmas y eventos entre aplicaciones usando este estándar. De igual manera *MatrikonOPC* ofrece una amplia variedad de productos relacionados con OPC AE entre lo que se pueden mencionar: **(MatrikonOPC, 2009)**

- ***MatrikonOPC Messenger***

  El producto *"MatrikonOPC Messenger"* es un Servidor OPC de A&E que además de permitir configurar lógica para la creación de alarmas, también permite el envió de notificaciones por email a partir de una lógica programable.

- ***MatrikonOPC A&E Historian***

  El producto *"MatrikonOPC A&E Historian"* está pensado para el almacenamiento de alarmas y eventos basados en texto. Se conecta a cualquier Sistema de Control para recolectar las alarmas y eventos generados y almacenarlos en cualquier Base de Datos ODBC como SQL Server, MS Access, Oracle, etc.

### 1.4.3. Descripción de OPC UA

Con la llegada de los Servicios WEB el consorcio OPC Foundation liberó la especificación OPC XML DA, que permite una interacción multiplataforma de clientes y servidores, pero traía como limitante un bajo desempeño y la inseguridad de utilizar un formato texto para la transmisión de datos.

Finalmente la Fundación OPC integra en una sola especificación una arquitectura independiente de la plataforma en que se implemente, que permite una integración más sólida de los servicios de OPC, esta especificación es nombrada *OPC Unified Architecture* (OPC UA). **(ABB, 2009)**

La especificación OPC UA aborda las deficiencias de XML DA:

- Integración de todas las especificaciones OPC permitiendo el acceso a las alarmas y datos históricos en la misma manera que los valores actuales.
- Una comunicación adecuada con dispositivos y a nivel empresarial.

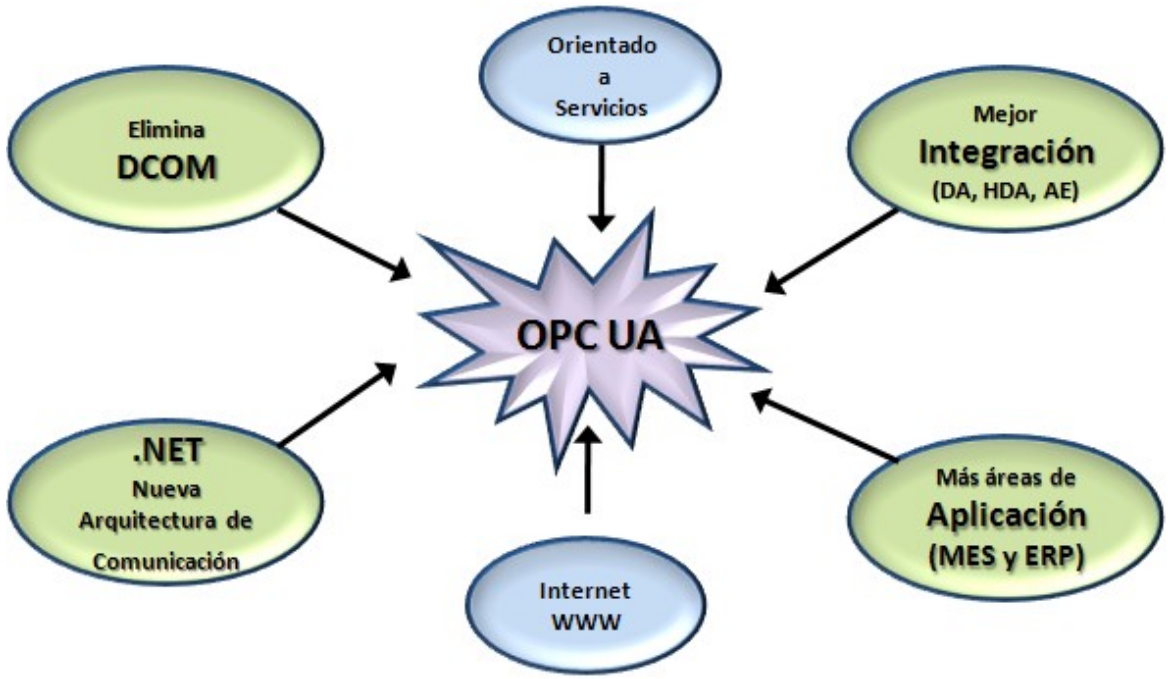


*Figura 6: Esquema Estructural de OPC UA*

#### 1.4.3.1. Características de OPC UA

OPC UA provee numerosos avances sobre las características de OPC que le precedieron. Dentro de ellas podemos mencionar: **(Massaro, 2008)**

- Integración de DA, A&E, HDA, Comandos, Datos Complejos y Tipos de datos.
- Manejo de datos complejos. OPC UA permite manejo de datos sencillos y complejos de forma nativa, datos OPC estándares y personalizados.
- Espacios de nombres avanzados**.** 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 se demasiado complicado para ser modelado vía OPC UA.
- Conjuntos de servicios básicos avanzado. La especificación base UA contiene los servicios genéricos necesarios para realizar navegación, búsqueda en los espacios de nombres, escritura y lectura de datos y publicación/suscripción de eventos y cambios de datos.
- Especificaciones UA derivadas. La especificación base de UA, es genérica y no implica una semántica avanzada. Esto permite que especificaciones funcionales avanzadas sean derivadas de la especificación base.
- Escalabilidad. Comparada con otras especificaciones OPC, UA podrá ser escalable desde dispositivos empotrados hasta grandes modelos de Negocios.
- Fiabilidad y Redundancia. OPC UA permite el diagnóstico y características informativas que permitan realizar aplicaciones fiables. Esto incluye la redundancia de los sistemas, re-sincronía, *keep-alives*, *resyncing*. Se permiten tanto clientes como servidores redundantes.
- Seguridad. OPC UA incluye la especificación de seguridad integrada, superando a su predecesor OPC XML DA.
- Comandos. OPC UA incluye la especificación de manejo de comandos.

#### 1.4.3.2. Ventajas de OPC UA

- La nueva "Arquitectura Unificada" (UA) crea la base para una nueva tecnología de información y comunicación de plataforma neutral.
- OPC UA utiliza en su arquitectura tecnologías que posibilitan la interoperabilidad entre distintos estándares, como XML y Web Services, así como protocolos de comunicación propietarios.
- Un servidor OPC UA agrega todas las funcionalidades que antes eran separadas en distintos servidores por el modelo OPC.
- Logra una integración de los estándares OPC DA, OPC AE y OPC HDA.
- Refleja el objetivo de Microsoft de retirar DCOM en favor de .NET y arquitecturas orientadas a servicio.
- OPC UA abandona COM/DCOM en favor de dos transportes: SOAP/HTTP(S) y un mensaje binario codificado en la parte superior de TCP.
- Presenta una nueva arquitectura de comunicación basada en .NET.
- Integración con más áreas de aplicación(MES y ERP).

#### 1.4.3.3. Uso del estándar OPC UA

El estándar OPC UA es usado en la actualidad por un sin número de empresas, a continuación se muestra un listado de alguna de estas, 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** | |

*Tabla 1: Empresas que usan el Estándar OPC UA*

### 1.4.4. Descripción de los Servicios Web

Un servicio web *(en inglés Web Service,* WS*)* es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los Servicios Web para intercambiar datos en redes de computadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS (*Organization for the Advancement of Structured Information Standards*) y W3C son los comités responsables de la arquitectura y reglamentación de los Servicios Web. 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. **(Wikipedia, 2009)**

#### 1.4.4.1. Características de los WS

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.
- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto no hay secretismos por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones. Es el futuro de las aplicaciones actuales.

#### 1.4.4.2. Estándares empleados

- *Web Services Protocol Stack:* Así se denomina al conjunto de servicios y protocolos de los Servicios Web.
- XML (*Extensible Markup Language):* Es el formato estándar para los datos que se vayan a intercambiar.
- SOAP (*Simple Object Access Protocol) o XML- RPC (XML Remote Producer Call):* Protocolos sobre los que, comúnmente, se establece el intercambio. Los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP *(Hypertext Transfer Protocol)*, FTP (*File Transfer Protocol*), o SMTP (*Simple Mail Transfer Protocol*).
- *WSDL (Web Services Description Languages):* Es el lenguaje de la interfaz pública para los Servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los WS.
- *UDDI* (*Universal Description, Discovery and Integration):* Protocolo para publicar la información de los Servicios Web. Permite comprobar qué WS están disponibles.

- *WS-Security (Web Service Security):* Protocolo de seguridad aceptado como estándar por OASIS. Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.

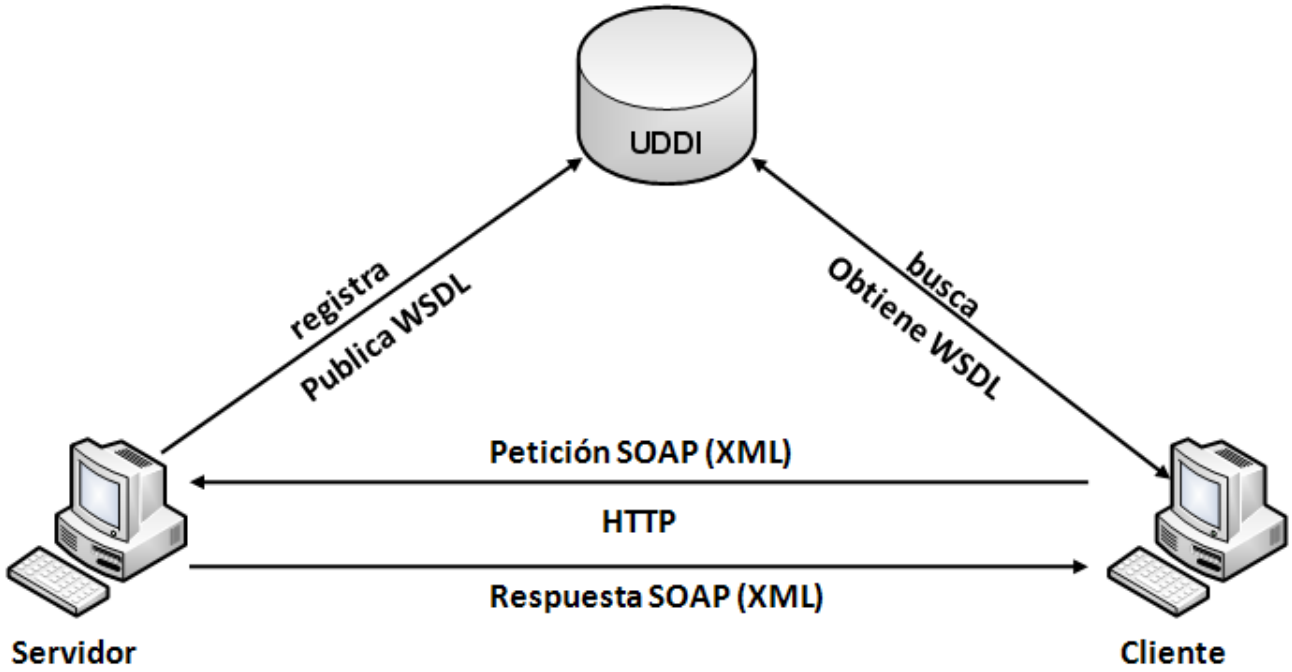


*Figura 7: Interacción Cliente-Servidor a través de los Servicios Web*

#### 1.4.4.3. Ventajas de los Servicios Web

- *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 HTTP).
- *Permiten la integración "justo-a-tiempo":* El proceso de descubrimiento se ejecuta dinámicamente, a medida que los solicitantes de servicios utilizan a los agentes para encontrar proveedores de servicio. Una vez 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 agentes actúan en conjunto para crear sistemas que son auto-configurables, adaptativos y robustos.

- *Reducen la complejidad por medio del encapsulamiento:* 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. El encapsulamiento es crucial para reducir la complejidad.
- *Dan una "nueva vida" a las aplicaciones:* Es relativamente correcto tomar una aplicación, generar un wrapper 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.
- *Disminuyen el tiempo de desarrollo de las aplicaciones:* Pues gracias a la filosofía de orientación a objetos utilizada, el desarrollo se convierte más bien en una labor de composición.

#### 1.4.4.4. Uso de Servicios Web

Con la generalización del uso de Internet se ha hecho una necesidad que los desarrolladores de SCADA incluyan facilidades de operatividad a través de cualquier navegador. A continuación se describen algunas aplicaciones SCADA que usan los Servicios Web para la comunicación con otras aplicaciones.

- ***Movicon 11*** es el primer HMI/SCADA basado completamente en tecnología XML, SOAP y tecnologías de Servicios Web. Su arquitectura innovadora se integra bien con XML, SVG y tecnologías de Servicios Web que permiten el acceso a los servidores a través de navegadores de Internet en cualquier plataforma. **(Progea, 2008)**
- ***NETSCADA*** es un SCADA basado en la web y en Internet. El uso de los servicios web le permite a los usuarios de este SCADA realizar las tareas de supervisión y monitoreo de forma remota usando un navegador web. **(Bentek, 2009)**
- Desde la versión 8, ***PcVue*** innova siendo el primer SCADA proporcionando la posibilidad de concebir una HMI 3D de forma simple. PcVue V8 propone también un conjunto de Servicios Web facilitando la creación del portal y la integración del sistema de monitorización y control

con las otras aplicaciones de la empresa como los MES, CMMS (*Computerised Maintenance Management System*), SCM (*Suplí Chain Management*) y las aplicaciones ERP.

### 1.4.5. Descripción de ICE Internet Communication Engine (ICE)

ICE (*Internet Communication Engine*) es un middleware es decir, un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas con soporte para los lenguajes de C++, C#, Java, Python, Ruby, PHP, y Visual Basic. ICE proporciona herramientas, APIs, y soporte de bibliotecas para construir aplicaciones cliente/servidor orientadas a objetos y entre muchos, un servicio de publicación/suscripción, IceStorm, que actúa como un distribuidor de eventos entre clientes y servidores. **(Henning, 2009)**

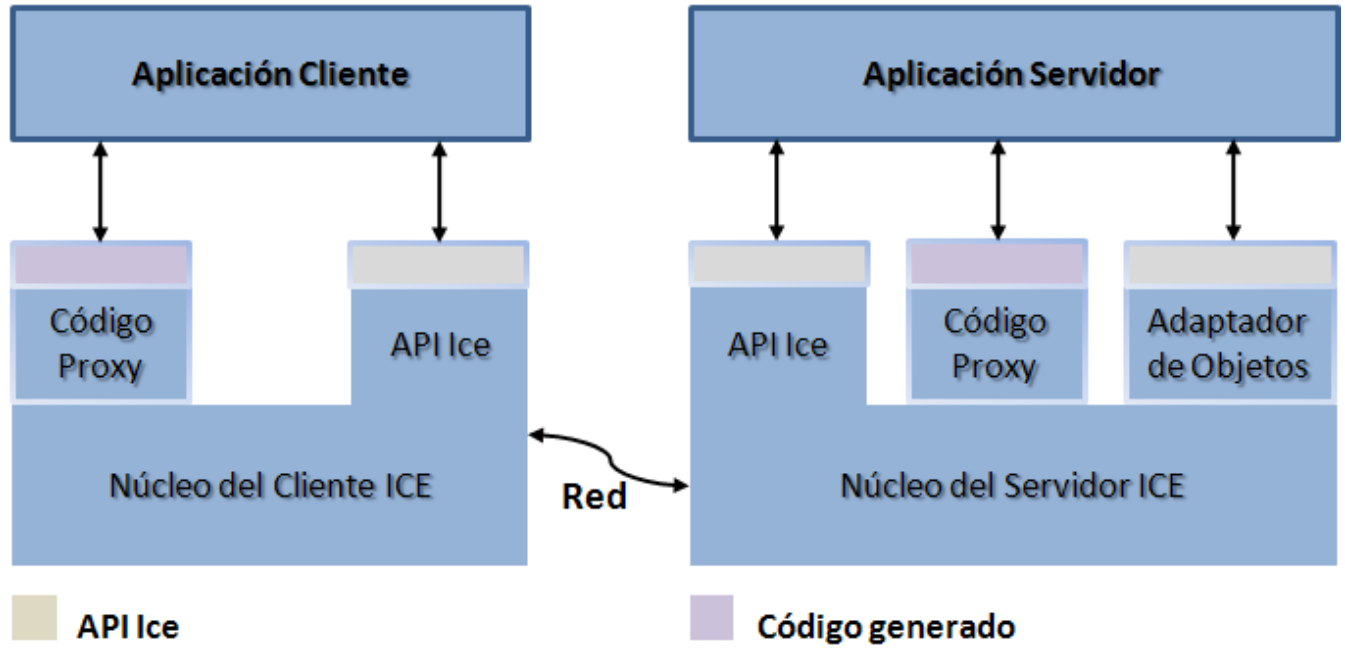


*Figura 8: Estructura Cliente/Servidor de ICE*

#### 1.4.5.1. Características de ICE

ICE no es más que un Middleware para el programador práctico, una plataforma para desarrollo de aplicaciones de comunicación para Internet de alto rendimiento que incluye varias capas de servicios y plugins, y se compone de los siguientes paquetes**: (ZeroC, 2009)**

- Plataforma de objetos distribuidos basados en enfoque cliente/servidor con persistencia.
- *Slice:* Es el Lenguaje de especificación de ICE que establece un contrato entre clientes y servidores y también se utiliza para describir los datos persistentes.
- *Slice Compilers:* Las especificaciones SLICE pueden ser compiladas en diferentes lenguajes de programación. Soporta C++, Java, C#, Visual Basic, Python y Ruby. Los clientes y servidores de ICE trabajan juntos independientemente del lenguaje de programación.
- *Ice:* Es la librería 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 pool 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, librería 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 (redundancia de servicios y servidores) y balanceo de carga.
- *Freeze*: Proporciona persistencia automática para servidores ICE. Con solo unas pocas líneas de código una aplicación puede incorporar un evictor (patrón) altamente escalable que gestiona de manera eficaz la persistencia de objetos.

- *FreezeScript*: Es común para cambiar tipos de datos persistentes, sobre todo en grandes proyectos de software. Con el fin de minimizar los impactos de estos cambios proporciona inspecciones y herramientas de migración Freeze para las bases de datos. Las herramientas soportan un XML basado en su capacidad de *scripting* que es a la vez poderoso y fácil de usar.
- *Ice SSL:* Un SSL dinámico que transporta plugin 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 sistemas 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 de mensajería que soporta federación. En contraste con la mayoría de los tipos de mensajerías o servicios de eventos, IceStorm soporta tipos de eventos, en el sentido de que el *broadcasting* de un mensaje a través de una federación es tan fácil como invocar un método en una interfaz. Se basa en el paradigma de comunicación publicación/suscripción.
- *Ice Patch:* 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.

#### 1.4.5.2. 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.
- Manejo de un elevado número de variables.
- Ya es usado en sistemas empotrados.

#### 1.4.5.3. Aplicaciones de ICE en la actualidad

Aunque es una tecnología novedosa ya existen varias implementaciones basadas en este Middleware:

- *Ice Touch:* con este Middleware podrás crear aplicaciones distribuidas para el IPhone de forma fácil y segura (Ofrece soporte a SSL), un ejemplo son los juegos en red. Está totalmente integrado con **Objetive-C y Cocoa**. **(ZeroC, 2009)**

- *ICE for Android* ofrece una robusta y totalmente equipada solución de Middleware que ayuda a crear aplicaciones de red fiables para nuevas plataformas móviles de Google. **(ZeroC, 2009)**

- *Ice-E ( "Embedded Ice")* es un compacto motor de comunicaciones diseñado específicamente para ser usado en entornos donde los recursos son escasos, tales como Internet habilitado en teléfonos inteligentes, asistentes digitales personales (PDA), y controladores embebidos. **(ZeroC, 2009)**

- El grupo de investigación de Arquitectura y Redes de Computadores (ARCO) de la Universidad de Castilla La Mancha se encuentra desarrollando una implementación de DDS (*Data Distribution Service*) de la OMG basada en el Middleware Ice. **(Villa, 2008)**

# CAPITULO 2: MODELADO DE LA SOLUCIÓN

El presente capítulo muestra la descripción de las funcionalidades del sistema, la selección de las tecnologías para implementar los Servicios de Integración con Terceros para el intercambio de Alarmas y Eventos y el diseño de cada una de las funcionalidades implicadas. En ocasiones se muestran modelos o escenarios relacionados con el Subsistema de Comunicación con Terceros como un todo para garantizar el mejor entendimiento para el lector pero solo se explica a fondo los elementos relacionados con los servicios desarrollados.

## 2.1. Descripción de las Funcionalidades del Sistema

A continuación se muestran los requisitos funcionales y no funcionales del Módulo de Comunicación con Terceros (Comm3), relacionados con los Servicios de Terceros para intercambio de Alarmas y Eventos.

### 2.1.1. Requisitos Funcionales

Los requisitos funcionales (RF) se pueden definir como capacidades o condiciones que el sistema debe cumplir, y que se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. A continuación la lista de requisitos del sistema relacionados con el servicio anterior:

RF1. El sistema debe permitir el intercambio de información relacionada con alarmas del sistema.
- RF1.1 Brindar un mecanismo para acceder al servicio de intercambio de alarmas.
- RF1.2 Proporcionar un mecanismo de solicitud de alarmas.
- RF1.3 Debe conectarse al Middleware del SCADA y obtener las alarmas solicitadas por los terceros.
- RF1.4 Entregar las alarmas a los terceros.

RF2. El sistema debe permitir el intercambio de información relacionada con eventos del sistema.
- RF2.1 Brindar un mecanismo para acceder al servicio de intercambio de eventos.
- RF2.2 Proporcionar un mecanismo de solicitud de eventos.

RF2.3 Debe conectarse al Middleware del SCADA y obtener los eventos solicitados por los terceros.

RF2.4 Entregar los eventos a los terceros.

### 2.1.2. Requisitos No Funcionales

Las características o requisitos no funcionales (RNF) del sistema, son propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido y confiable. Estos requisitos son muy importantes para que los clientes y usuarios puedan valorar características no funcionales del software como: seguridad, confiabilidad y usabilidad.

**RNF1. Software**

RNF1.1. Sistema operativo GNU/Linux, distribución Debian Lenny (5), Kernel 2.6.22-2

**RNF2. Usabilidad**

RNF2.1. El sistema debe permitir a los terceros el acceso al servicio de manera rápida y segura.

RNF2.2. El servicio solo será usado por clientes autorizados a obtener información del SCADA.

**RNF3. Fiabilidad**

RNF3.1. Las solicitudes de alarmas, eventos, van a ser siempre dependientes de los mecanismos de suscripción y de la lógica de seguridad del sistema.

RNF3.2. Debe existir una comunicación eficiente y fiable, garantizando no existan pérdidas de información relacionada con alarmas y eventos.

RNF3.3. Debe proveerse, a los servicios, de mecanismos de tolerancia ante fallos que permitan recuperarse antes errores.

**RNF4. Rendimiento**

RNF4.1. La velocidad de intercambio de alarmas y eventos puede ser inferior a la adquisición de variables.

**RNF5. Soporte**

RNF5.1. El desarrollo del sistema orientado a la exposición de interfaces y servicios debe brindar alta interoperabilidad para que aplicaciones externas implementadas en diferentes lenguajes y sistemas operativos, puedan comunicarse de manera fácil y eficiente con el servicio.

RNF5.2. Se debe implementar el servicio sobre la base de una arquitectura que introduzca en el sistema atributos de flexibilidad, escalabilidad y adaptación y que permita agregar y/o modificar funcionalidades con bastante facilidad y de forma eficiente sin impactar en los clientes y sin introducir grandes cambios.

**RNF6. Portabilidad**

RNF6.1. La solución adoptada debe poder ser implantada en cualquier aplicación que requiera servicios de este tipo y sobre cualquier plataforma.

## 2.2. Solución independiente de la tecnología

Independientemente de la tecnología seleccionada para desarrollar el sistema es necesario adoptar una solución de alto nivel que facilita dos vías de comunicación de los sistemas externos con el Guardián del ALBA. La primera debe garantizar la exposición de las funcionalidades del SCADA para acceso a las variables, alarmas, eventos e interacción a través de comandos con aplicaciones externas personalizadas sin restricciones de tiempo real y la segunda parte de la solución debe exponer una interfaz de comunicación de alto nivel que permita la intercambio de información relacionada con variables (solamente) en tiempo real y de forma segura entre el Guardián del ALBA y otros SCADA.

A continuación se expone una vista de alto nivel del subsistema de Comm3:

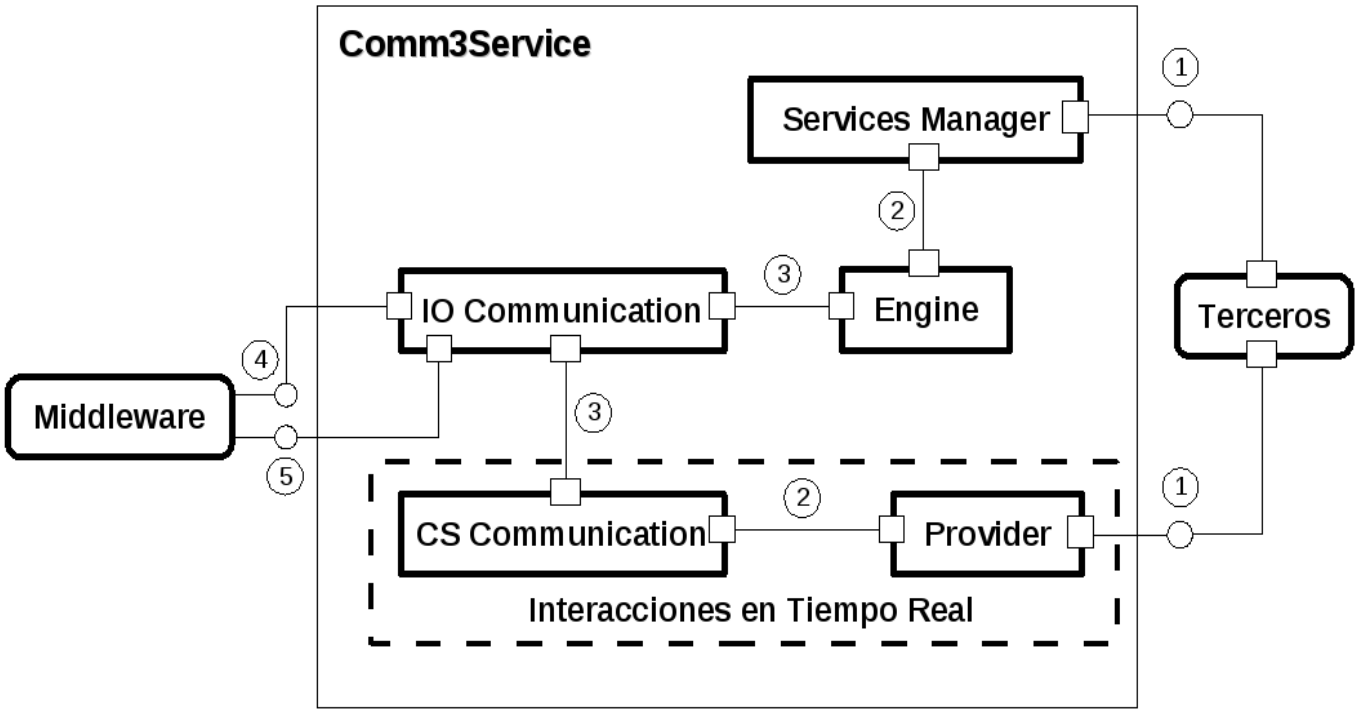


*Figura 9: Subsistemas del Módulo de Comm3*

### 2.2.1. Descripción de módulos o subsistemas

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

***Services Manager:*** este módulo o subsistema es el encargado de exponer las funcionalidades para 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:*** módulo o subsistema encargado de manejar las subscripciones y accesos a las variables, alarmas y eventos así como la invocaciones de comandos realizadas desde las aplicaciones externas a través del Services Manager, siempre dependiente de los mecanismos de seguridad implementados.

***IO Communication:*** después de recibir las peticiones del Engine, es el encargado de recibir la información que viene del Guardián del ALBA y pasarla hacia el nivel superior, el Engine. Para lograr sus objetivos utiliza la interfaces del Middleware para recibir puntos, alarmas, eventos y para enviar comandos.

***Provider:*** módulo que 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.

***CS Communication:*** subsistema que encapsula la lógica del negocio del servicio de acceso a variables para interacciones en tiempo real. Usa el subsistema *IO Communication* para pedir la data del SCADA usando la interfaces que provee el Middleware para interacciones de este tipo.

**Middleware:** representa el Middleware del Guardián del ALBA, se refiere a las conexiones con los servicios del Middleware y las interfaces que este provee para el acceso a puntos, alarmas, eventos e interacciones a través de comandos.

### 2.3. Selección de las tecnologías

Si tomamos como premisa los estándares y tecnologías actuales utilizados para implementar soluciones de este tipo y que fueron abordados en el Capítulo 1 y como elemento principal, los requisitos especificados al inicio de este capítulo, básicamente relacionados con brindar interfaces y servicios para el intercambio de alarmas y eventos entre el Guardián del ALBA y aplicaciones externas, entonces se pueden hacer los siguientes análisis.

### 2.3.1. Selección de los criterios de evaluación

A continuación se exponen una serie de criterios que deben ser evaluados por cada tecnología a fin de realizar los análisis apropiados y seleccionar la mejor opción:

1. *Facilidad de implementación:* este criterio pretende evaluar cuanto de difícil sería implementar la solución con las tecnologías analizadas. Evaluar este criterio es de suma importancia porque permite realizar una mejor estimación del tiempo de desarrollo de la solución y aporta seguridad al cliente teniendo en cuenta que si la implementación se hace fácil se puede cumplir con las entregas en el tiempo estimado.

2. *Documentación disponible y alcance del personal:* define la posibilidad existente de avanzar de forma rápida y segura en el desarrollo de la solución usando las tecnologías con documentación disponible. Contar con la documentación disponible aporta seguridad al desarrollador y al cliente. La combinación con el criterio anterior puede arrojar muy buenos resultados.

3. *Posibilidad de crear soluciones universales:* este atributo permite evaluar las funcionalidades que aportan las tecnologías analizadas para desarrollar soluciones universales. Para satisfacer la arquitectura del subsistema de Comunicación con Terceros es de suma importancia contar con una tecnología que permita desarrollar una solución universal siempre que se entienda por "solución universal" aquella solución de comunicación que permita a las aplicaciones externas interactuar con nuestro sistema independientemente del sistema de operativo donde se encuentre y el lenguaje de programación en que fue implementada.

4. *Desarrollo de aplicaciones distribuidas:* este criterio pretende evaluar la capacidad de desarrollar aplicaciones distribuidas que aportan las tecnologías analizadas siendo este criterio un punto a tener en cuenta y que no debe faltar en aplicaciones de este tipo.

5. *Consumo de Recursos:* este criterio tiene un valor importante y determinante en la selección permitiendo determinar aproximadamente cómo se comporta el consumo de CPU y Memoria en cada caso.

6. *Soporte por parte de los desarrolladores:* analizar este criterio permite tener claridad y seguridad en cuanto al soporte que tiene cada tecnología y saber si es seguro usar una u otra dependiendo de quién de el soporte y de qué forma se realice el mismo.

7. *Transmisión de un elevado número de variables:* normalmente las aplicaciones de terceros del tipo personalizadas no requieren información de un elevado número de variables en un tiempo determinado pero existe una parte de la arquitectura que modela la interacción con otros SCADA como sistemas externos que si requieren el acceso a un elevado número de variables es por ello la importancia de definir este criterio y analizar la capacidad de transferencia que tiene cada tecnología.

#### 2.3.1.1. Importancias

Se debe asignar un valor de importancia a cada criterio y el mismo debe estar en el rango de 1 a 100. La tabla #2 muestra los valores de importancia asignado a cada criterio definido en el apéndice 2.3.1.

| Criterios | Importancia |
|---|---|
| Facilidad de implementación. | 10 |
| Documentación disponible y al alcance del personal. | 10 |
| Posibilidad de crear soluciones universales. | 20 |
| Desarrollo de aplicaciones distribuidas. | 15 |
| Consumo de Recursos. | 15 |
| Soporte por parte de los desarrolladores. | 15 |
| Transmisión de un elevado número de variables (25000 variables). | 15 |

*Tabla 2: Importancias por Criterios de Evaluación*

#### 2.3.1.2. Calificación

Se refiere al nivel de aceptación que tiene un criterio en una tecnología determinada. El **Anexo I,** entre otros elementos, muestra los diferentes tipos de calificaciones que se puede otorgar a cada criterio de evaluación.

#### 2.3.1.3. Opciones Tecnológicas

Las opciones tecnológicas evaluadas también se pueden analizar en el **Anexo I** donde se muestra la matriz decisión con el puntaje alcanzado por cada tecnología en cada criterio y de forma general a partir de las calificaciones otorgadas a cada criterio evaluado. Esta matriz de decisión permite determinar la tecnología con mayor nivel de aceptación de acuerdo al puntaje alcanzado.

### 2.3.2. Justificación de la Evaluación de Criterio – Alternativas

Para observar los detalles de la evaluación de cada uno de los criterios y determinar el porqué de los valores otorgados en cada caso, se debe remitir a la **tabla # X** que se encuentra en el **Anexo I**. A continuación se realiza un análisis crítico por cada alternativa evaluada en la matriz de decisión.

#### 2.3.2.1. Análisis de OPC AE

La inmensa mayoría de los sistemas SCADA proveen la comunicación con terceros con estándares mundiales, típicamente OPC o DAIS. Lo ideal para un SCADA como el nuestro sería proveer esas interfaces para los Servicios de intercambio a las Alarmas y los Eventos. 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. Además de existir una especificación de OPC para seguridad la mayoría de los clientes y servidores existentes no la implementan lo que limita la cantidad de aplicaciones que podrían intercambiar datos con el sistema, a un número muy reducido. El costo de implementar el estándar OPC AE es alto, más cuando se necesitan otras herramientas (*toolkits*) que aún no conocemos. También adoptar esta solución es muestra de no estar mirando hacia delante y enfocados en los avances actuales relacionados con OPC porque los software migran a gran velocidad a OPC UA y ya existen pasarelas entre ambas versiones de la

especificación. OPC UA ya cuenta con implementación de pasarelas entre ambas OPC UA y OPC DA y está disponible para miembros *OPC Foundation* y también tiene la especificación de alarmas y eventos. Por último uno de los requisitos de módulo de Comm3 radica en brindar una solución de comunicación con las aplicaciones externas lo más universal posible, independiente de plataforma y sistemas operativo, algo que OPC AE no logra totalmente en el sentido que obliga a los terceros a tener un cliente de OPC AE para comunicarse con un servidor del mismo tipo.

#### 2.3.2.2. 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 el acceso a las alarmas y eventos a través de Servicios Web ya que *OPC Foundation* ofrece el WSDL para esta especificación. También existen las pasarelas entre OPC UA y OPC DA lo que eliminaría problemas de compatibilidad con las versiones anteriores. OPC UA podría ser la selección ideal, es una especificación que describe todas las funcionalidades que se deben exponer a los terceros, entre ellas manejo de alarmas y eventos, sin embargo tiene como principal desventaja que esta especificación se distribuye bajo licencia comercial limitando el acceso a la misma para de manera libre implementar los servicios y/o funcionalidades que define.

Si analizamos bien, la mayoría de los SCADA soportan el *scripting* y objetos Activex (COM en general), muchos de ellos utilizan *Visual Basic for Application*, lo normal es que los SCADA provean objetos que puedan ser utilizados desde estos scripts. Estos objetos no son más que *wrappers* que hacen de interfaces con los lenguajes de programación soportados por los SCADA. Así, si se quiere comunicar el Guardián del ALBA u otra aplicación con otro SCADA sólo tenemos que entregarles a los clientes ese objeto que pueda ser utilizado desde los *scripts*. Para adoptar esta solución o una solución similar se decide analizar el middleware Ice y los Servicios Web como tecnologías candidatas para desarrollar la solución.

#### 2.3.2.3. Análisis de Servicios Web

Los Servicios Web resuelven los problemas de interoperabilidad encontrados en tecnologías como CORBA, DDS, etc., son fáciles de implementar, usar y manejar. El módulo de Comunicación con Terceros debe brindar a los sistemas externos mecanismos de acceso al SCADA lo más universales posibles, que puedan ser usados tanto por aplicaciones implementadas en Windows como en Linux. Aunque no es menos cierto que una desventaja de los Servicios Web es la velocidad de transmisión de la información, esto no es un inconveniente para nuestra solución si analizamos que la comunicación con terceros no tiene restricciones de tiempo real y existen herramientas que logran mejorar considerablemente el inconveniente anterior. Los Servicios Web permiten además, desarrollar soluciones bastante genéricas y robustas.

#### 2.3.2.4. Análisis de ICE

Como se pudo observar en el Capítulo 1 ICE es una tecnología que puede imprimir un rendimiento considerable en las comunicaciones inter SCADAS 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 debemos migrar en algún momento. Brinda diferentes mecanismos de comunicación los cuales pueden ser usados en conjunto con los servicios que brinda para implementar los Servicios de intercambio de información relacionada con las Alarmas y Eventos del Guardián del ALBA.

## 2.4. Tecnología(s) Seleccionada(s)

Después de definir las principales tecnologías que permiten desarrollar la solución, describir sus características, justificar porque se tuvieron en cuenta y valorar los criterios de evaluación por cada tecnología ha llegado el momento de seleccionar la(s) tecnología(s) adecuada(s) para implementar los servicios analizados.

Teniendo en cuenta los puntos desarrollados y los resultados que arrojó la matriz de decisión se puede asegurar que ICE y los Servicios Web son las tecnologías con mayor aceptación, sin embargo en consenso con el equipo de desarrollo del SCADA y la contraparte se decide seleccionar los Servicios Web para implementar los servicios teniendo en cuenta que no existen restricciones de tiempo real y que no se necesitan comunicaciones tan rápidas como garantiza Ice pero si soluciones multiplataforma sin dependencia de la tecnología utilizada.

### 2.5. Herramientas y metodologías a utilizar

Las metodologías y herramientas que se exponen a continuación no son objeto de selección en este trabajo, las mismas fueron analizadas y evaluadas desde los inicios del proyecto SCADA es por ello que solo se describen sus principales características.

#### 2.5.1. RUP como metodología de desarrollo

El Proceso Unificado de Rational (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis y diseño, implementación y documentación de sistemas orientados a objetos, principalmente de grandes proyectos como el caso del SCADA Guardián del ALBA.

La metodología RUP, fue desarrollada en 1998 por Grady Booch, Ivar Jacobson y James Rumbaugh, reconocidos metodólogos en la industria de la tecnología y sistemas de información. RUP se caracteriza por ser: dirigido por Casos de Uso, centrado en la arquitectura, iterativo e incremental. Divide el proceso de desarrollo en ciclos o fases, teniendo un producto final al culminarse cada una de las interacciones por fases. **(Booch, y otros, 1999)**

Con el uso de una metodología se pretende reducir costos y retrasos de proyectos, así como mejorar la calidad del software.

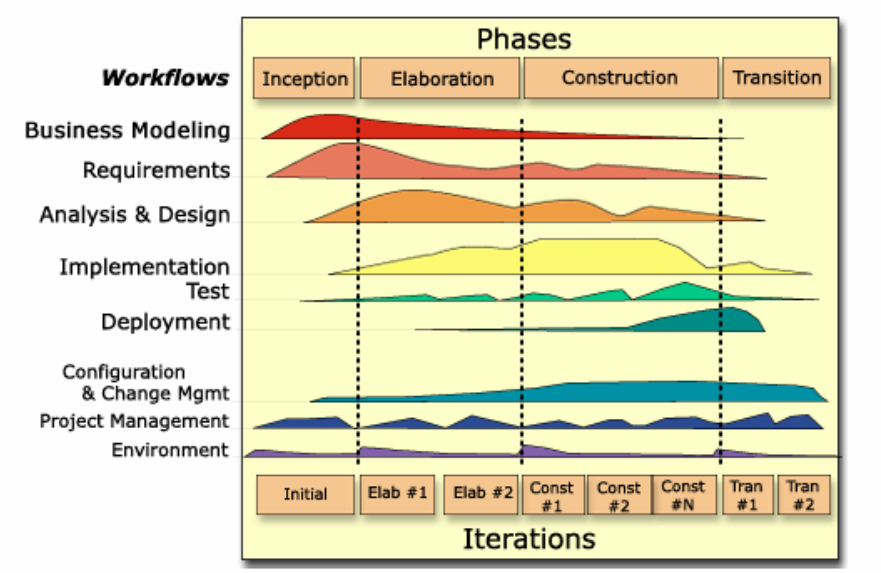


*Figura 10: Ciclo de Vida de RUP*

### 2.5.2. UML como lenguaje de modelado

UML (*Unified Modeling Language*) es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. **(Booch, y otros, 2007)**

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. 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.

**Características principales**

1. Divide cada proyecto en un número de diagramas que representan las distintas vistas del proyecto y juntos representan la arquitectura del mismo.
2. Permite describir un sistema en diferentes niveles de abstracción, simplificando la complejidad sin perder información, para que los usuarios y desarrolladores comprendan las características de la aplicación.
3. Se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del desarrollo de una aplicación, sin embargo no pretende definir un modelo de desarrollo sino únicamente un lenguaje de modelado.

### 2.5.3. RSA como herramienta CASE

RSA (*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 arquitectura sólida. Rational Software Architect es un componente del paquete *IBM Rational Profesional*, este aprovecha a UML para el diseño de la arquitectura para C++ y *Java 2 Enterprise Edition* (J2EE) y aplicaciones de servicios web.

Con *IBM Rational Software Architect* se puede: **(IBM, 2009**)

- Unificar todos los aspectos del diseño y desarrollo de software.
- Desarrollar aplicaciones más productivas.
- Sacar provecho de la tecnología más avanzada en lenguaje de modelado.
- Revisar y controlar la estructura de las aplicaciones Java.
- Realizar ingeniería inversa en los dos sentidos, desde código obtener el diseño y desde el diseño generar el código.
- Integrar otras facetas del ciclo de vida.

### 2.5.4. C++ como lenguaje de programación

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

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

C++ tiene varias características que otros lenguajes de programación no tienen. Las más destacadas son:

- Es un lenguaje multiplataforma, orientado a objetos e imperativo.
- En cuanto a la ejecución, es uno de los lenguajes más rápidos y eficiente.
- *Existen muchas herramientas para C++,* gran cantidad de compiladores, depuradores y librerías para C++.
- 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.
- C++ permite desarrollar aplicaciones de alto y bajo nivel.
- Existe gran cantidad de usuario en la red que usan este lenguaje aportando más conocimiento cada vez que es necesario.

Además de todas las ventajas que posee este lenguaje de programación las dos características que más peso tienen en la decisión son:

1. Desde la definición del SCADA se decidió usar C++ y ahora debe ser usado en todos los desarrollos del SCADA o relacionados con el mismo.
2. Es el lenguaje que más conoce y domina el equipo de desarrollo.

### 2.5.5. Eclipse como entorno de desarrollo

La plataforma Eclipse consiste en un Entorno de Desarrollo Integrado (*IDE, Integrated Development Environment*) abierto y extensible. Se utiliza como IDE Java y cuenta con numerosas herramientas de desarrollo de software. También da soporte a otros lenguajes de programación, como son C/C++, Cobol, Fortran, PHP y Python. A la plataforma base de Eclipse se le pueden añadir extensiones (plugins) para extender sus funcionalidades.

El término Eclipse además identifica a la comunidad de software libre para el desarrollo de la plataforma Eclipse. Este trabajo se divide en proyectos que tienen el objetivo de proporcionar una plataforma robusta, escalable y de calidad para el desarrollo de software con el IDE Eclipse que actualmente se encuentra en su versión 3.4.

#### 2.5.5.1. Características generales

- Muy completo
- Configurable mediante plugins
- Pensado para Java pero adaptable a otros lenguajes
- Versiones preparadas para C/C++
- Plugins para Subversion y Doxygen

### 2.5.6. gSOAP como herramienta de desarrollo de Servicios Web en C++

Para usar los Servicios Web como la tecnología que soporte la solución del servicio de alarmas y eventos es necesario una herramienta que permita desarrollar servicios Web en C++ y de ser posible integre los componentes o estándares de los servicios Web, como son: SOAP, UDDI, WSDL y componentes de seguridad.

De las herramientas existentes en la Web, que cumplan con los requisitos anteriores y que es muy usada podemos decir que gSOAP es la que más se acerca a nuestra necesidades por eso se decide usarla para implementar el Servicio de Integración con Terceros para el Acceso a la Información relacionada con Alarmas y Eventos.

En la tabla # 3 se puede observar la comparación entre las tecnologías analizadas. Las marcadas con “x” significan que el estándar es soportado por la tecnología en cuestión.

| 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 |

*Tabla 3: Herramientas para el Desarrollo de Servicios Web*

Aunque en la tabla anterior no se refleja, en la bibliografía consultada, aparece esta herramienta como la elección clara para la implementación de servicios web en C/C++ y sobre software libre. También de las herramientas de este tipo, es la más se usa en el mundo y la que más documentación tiene disponible.

El conjunto de herramientas gSOAP Web Services, 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++. **(gSOAP, 2009)**

#### 2.5.6.1. Características generales

- Es multiplataforma, funciona con cualquier compilador de C o C++.
- Si la necesidad de usar Linux, Windows o Mac como cliente del servicio es clave, gSOAP es la elección clara.
- En la actualidad permite trabajar con SOAP, WSDL y UDDI, así como con otras tecnologías como (*WS-Addressing *, WS-Security * y WS-Discovery *).*
- Las herramientas gSOAP se distribuyen con tres tipos de licencia: GNU GPL, código abierto público gSOAP y comercial.

- Es un *framework* que permite olvidarnos de todo lo que tiene que ver con la parte de comunicaciones y dedicarnos a programar la funcionalidad que nos interesa conseguir.

#### 2.5.6.2. Transmisión de Mensajes con g SOAP

Como se enunció anteriormente los Servicios Web se hacen lentos durante la transmisión de la data, gSOAP utiliza como solución a este problema *Message Transmission Optimization Mechanism* (MTOM). 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 y documentos.

## 2.6. Diseño de la Solución Propuesta

Aunque el trabajo está orientado a la implementación de los Servicios para intercambio de Alarmas y Eventos se expone la arquitectura del módulo como un todo para acercar más al lector a la solución.

### 2.6.1. Diseño de la arquitectura

El modelo de diseño está dividido en tres capas donde cada capa tiene su responsabilidad bien definida. Esta distribución en capas imprime atributos de flexibilidad y escalabilidad a la solución permitiendo modificar una capa sin afectar al resto y/o la interacción con el cliente y hacer cambios en el diseño y escalar la solución sin necesidad de hacer grandes modificaciones. 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 e interacción de comandos. Las responsabilidades de esta capa están divididas en dos paquetes, *Services Manager* que expone los servicios web y *Client Manager* que expone las

interfaces de comunicación de alto nivel para interacciones en tiempo real. Esta capa depende de la capa de negocio para resolver 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 subscripciones, acceso a la información del SCADA, 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 *Engines* que encapsula las funcionalidades relacionada con la suscripción y acceso a la información del SCADA y el paquete *CS Communication* que abstrae el paradigma cliente/servidor de ICE.

***Communication (Capa de Comunicación):*** esta capa tiene como principal responsabilidad hacer las peticiones a los módulos del SCADA a través de las interfaces de comunicación del Middleware. Las clases del paquete *IO Communication* se encargan de gestionar las comunicaciones a través del Middleware.

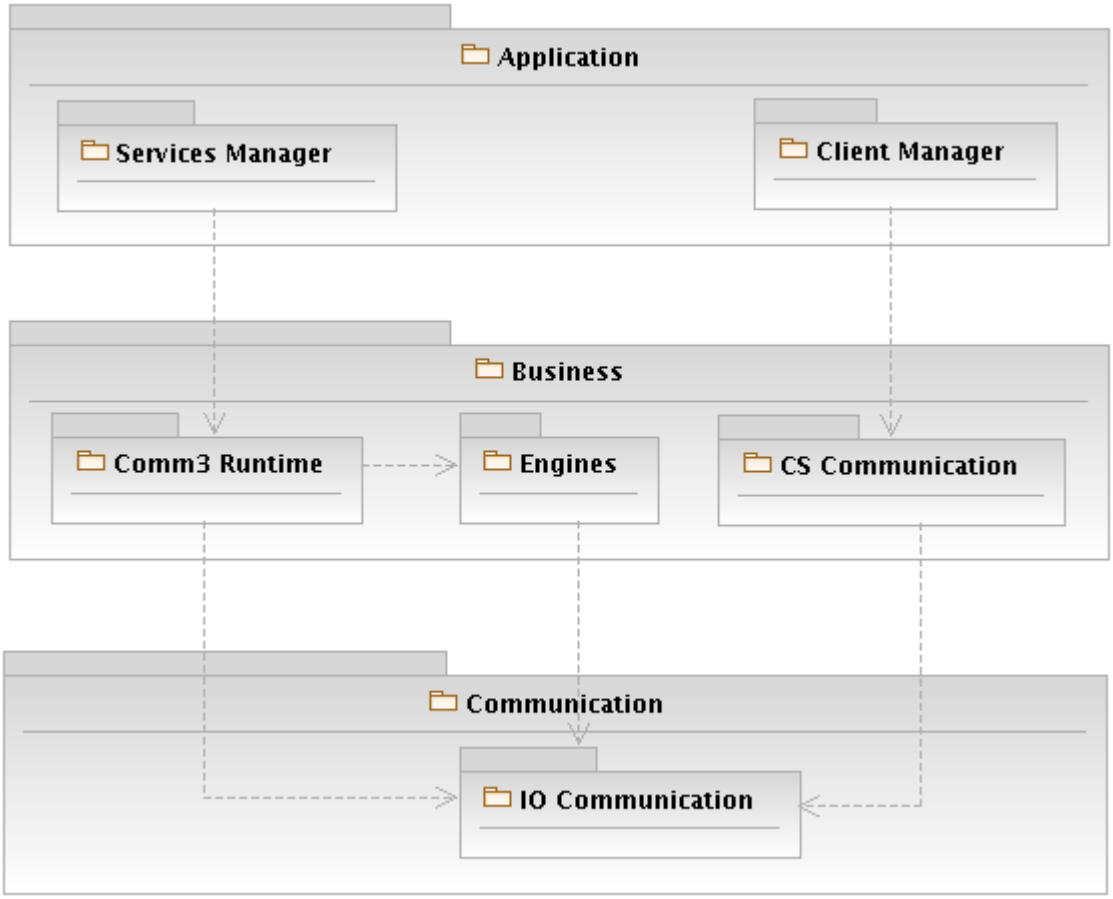


*Figura 11: Arquitectura del Subsistema de Comm3*

### 2.6.2. Diseño detallado de la solución

El presente modelo de diseño intenta modelar la parte del módulo de Comm3 relacionada con el Servicio de Integración con Terceros para el intercambio de información relacionada con alarmas y eventos mostrando un vista lógica que define la solución que permitirá la comunicación y el intercambio de información relacionada con alarmas y eventos entre las aplicaciones externas y el SCADA Guardián del ALBA.

El mismo describe las clases y sus relaciones distribuidas en paquetes de diseño bien definidos y con una responsabilidad determinada.

A continuación se expone los paquetes de diseño del subsistema de Comm3 y sus relaciones, involucrados en la solución del los servicios a implementar.

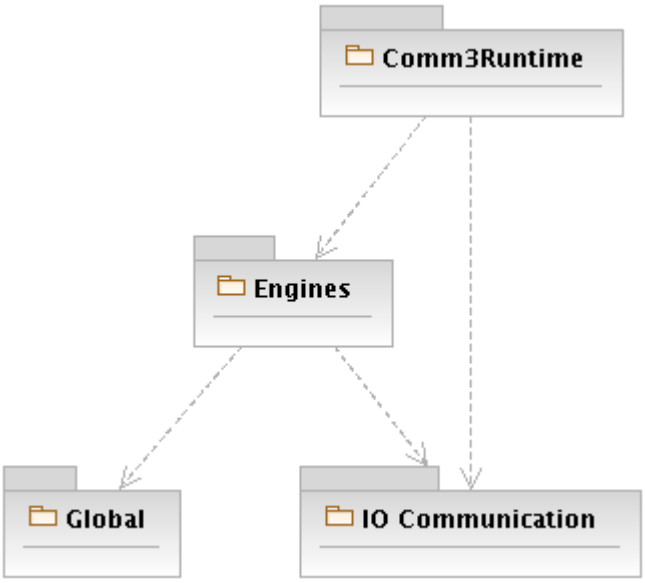


*Figura 12: Relación entre los Paquetes de Diseño*

El presente epígrafe siguiente muestra los diagramas de clases por paquetes de diseño, y la descripción textual de cada una de las clases y sus métodos.

#### 2.6.2.1. Paquete Comm3Runtime

El paquete de Comm3 para Ejecución contiene las clases que definen la lógica para iniciar los Servicios de Alarmas y Eventos de Comm3, estás clases van a implementar la lógica que permite conectarse al *Middleware* del SCADA, iniciar la comunicación vía SOAP para escuchar y resolver las peticiones de los clientes y la implementación de las funcionalidades que soporta cada servicio. Además usa el paquete *Engines* para resolver las peticiones de suscripción y acceso a la información. Este paquete es el punto de entrada de la aplicación.

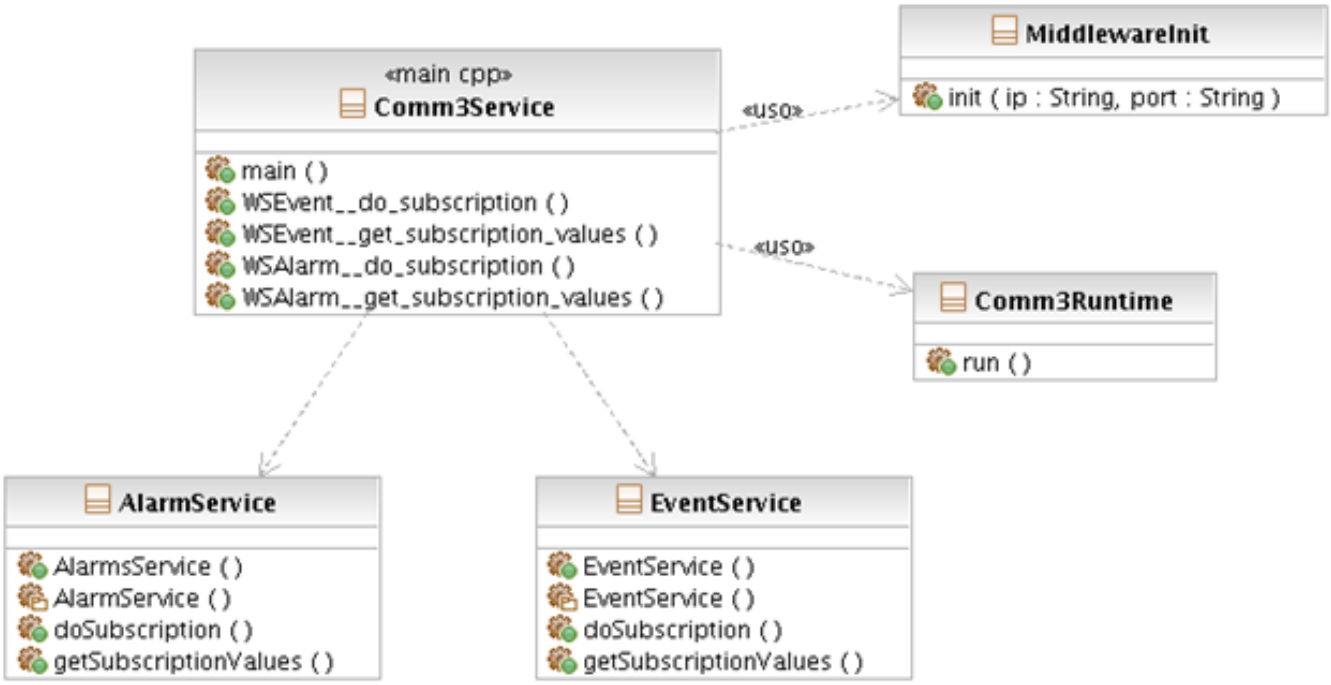


*Figura 13: Diagrama de Clases del Paquete Comm3Runtime*

**Descripción de Clases y Métodos**

| ***Main, Comm3Service*** | |
|---|---|
| Este elemento no es en realidad una clase, 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 vía SOAP para garantizar las comunicaciones entre los terceros y los servicios de integración con terceros para intercambio de alarmas y eventos. | |
| Nombre del método | Descripción |
| main() | Define la lógica para comunicarse con el resto del SCADA a través del Middleware, usando el método init () de la clase MiddlewareInit.<br>Define la lógica para iniciar la comunicación vía SOAP invocando el método run() de la clase Comm3Runtime. |
| WSAlarm__do_subscription(): Integer | Este método hace una llamada al método de suscripción a las alarmas definido por la clase AlarmService. |

| | |
|---|---|
| WSAlarm__get_subscription_values(): Integer | Este método hace una llamada al método de lectura de las alarmas a partir de una suscripción definido por la clase AlarmService. |
| WSEvent__do_subscription(): Integer | Este método hace una llamada al método de suscripción a los eventos definido por la clase EventService. |
| WsEvent__get_subscription_values(): Integer | Este método hace una llamada al método de lectura de los eventos a partir de una suscripción definido por la clase EventService. |

*Tabla 4: Descripción del main() de la Aplicación, Comm3Service*

| ***Clase, MiddlewareInit*** | |
|---|---|
| Esta clase tiene la responsabilidad de iniciar la comunicación de los servicios de Comm3 con el Middleware del Guardián del ALBA usando el paquete IO Communication y la información del Ip y Puerto por donde están ejecutándose los servicios de Middleware. | |
| Nombre del método | Descripción |
| init(ip: String, port: String) | Este método es invocado desde el main de la aplicación indicando el ip y el puerto por donde está corriendo el Middleware. Este método define la lógica para conectarse a los servicios de Middleware como un cliente más de este. |

*Tabla 5: Descripción de la Clase MiddlewareInit*

| ***Clase, Comm3Runtime*** | |
|---|---|
| Como bien dice su nombre esta clase tiene la responsabilidad de iniciar las comunicaciones vía SOAP y permitir de esta manera que los servicios de alarmas ye eventos reciban las peticiones realizadas vía SOAP, garantizando una comunicación cliente-servidor a través del protocolo SOAP. | |
| Nombre del método | Descripción |
| run() | Este método es invocado desde main() de la aplicación desde el mismo instante en que se levantan los servicios 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 SOAP necesarios para establecer la comunicación con los clientes. |

*Tabla 6: Descripción de la Clase Comm3Runtime*

| ***Clase, AlarmService*** |
|---|
| Está clase tiene como responsabilidad definir la lógica de implementación de los métodos de servicio de acceso a las alarmas descrito en la definición del servicio en C++, WSAlarm__Alarm. |

| Nombre del método | Descripción |
|---|---|
| doSubscription ( ) : unsigned long | Este método define la lógica de implementación del método de subscripción a las alarmas definido en la descripción del servicio y es invocado desde los clientes a partir del servicio web, WSAlarm.wsdl. Utiliza la lógica definida en el paquete AlarmEngine para registrar la suscripción y entregar al cliente el identificador de la misma. |
| getSubscriptionValues ( ) : WSAlarm__AlarmList | Este método define la lógica de implementación del método de lectura de valores de alarmas definido en la descripción del servicio y es invocado desde los clientes a partir del servicio web, WSAlarm.wsdl. Utiliza la lógica definida en el paquete AlarmEngine para a partir del identificador de la suscripción entregar al cliente la lista de alarmas solicitadas. |

*Tabla 7: Descripción de la Clase AlarmService*

| ***Clase, EventService*** | |
|---|---|
| Está clase tiene como responsabilidad definir la lógica de implementación de los métodos del servicio de acceso a los eventos descrito en la definición del servicio en C++, WSEvent__Event. | |
| Nombre del método | Descripción |
| doSubscription ( ) : unsigned long | Este método define la lógica de implementación del método de subscripción a los eventos definido en la descripción del servicio y es invocado desde los clientes a partir del servicio web, WSEvent.wsdl. Utiliza la lógica definida en el paquete EventEngine para registrar la suscripción y entregar al cliente el identificador de la misma. |
| getSubscriptionValues ( ) : WSEvent__EventList | Este método define la lógica de implementación del método de lectura de valores de eventos definido en la descripción del servicio y es invocado desde los clientes a partir del servicio web, WSEvent.wsdl. Utiliza la lógica definida en el paquete EventEngine para a partir del identificador de la suscripción entregar al cliente la lista de eventos solicitados. |

*Tabla 8: Descripción de la Clase EventService*

### Diagramas de Secuencia

### Iniciar Servicio de Eventos


*Figura 14: Diagrama de Secuencia Iniciar Servicio de Eventos*

**Iniciar Servicio de Alarmas**

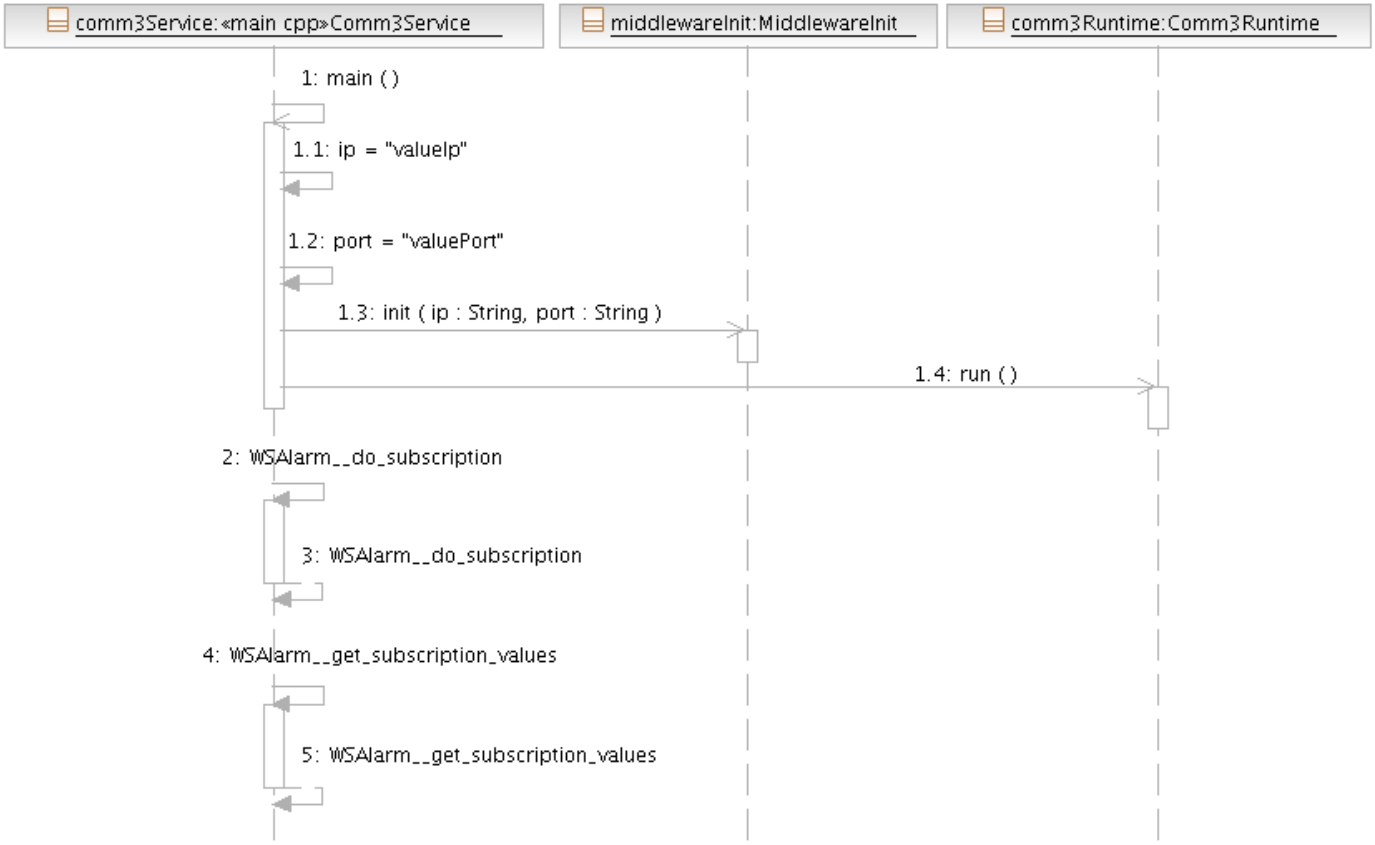


*Figura 15: Diagrama de Secuencia Iniciar Servicio de Alarmas*

**Solicitar suscripción a un grupo de alarmas**

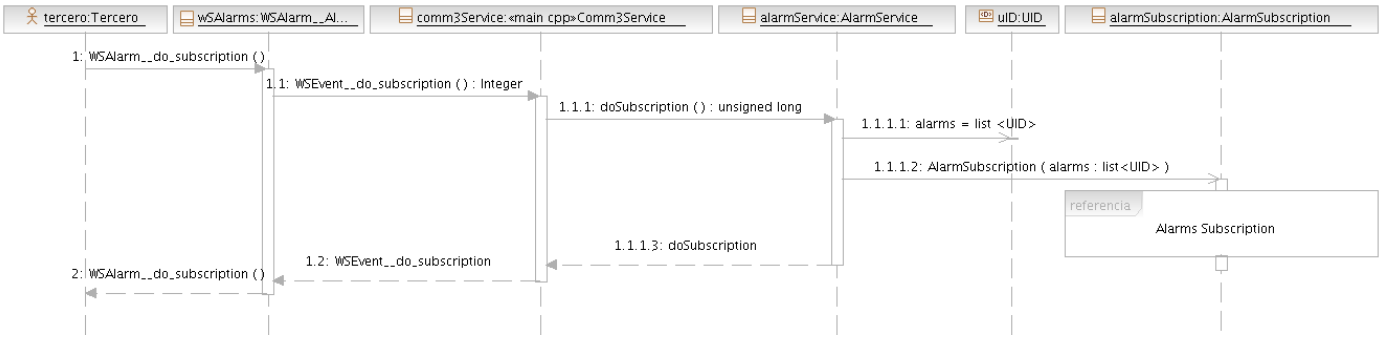


*Figura 16: Diagrama de Secuencia Solicitar un Grupo de Alarmas*

**Solicitar subscripción a un grupo de eventos**

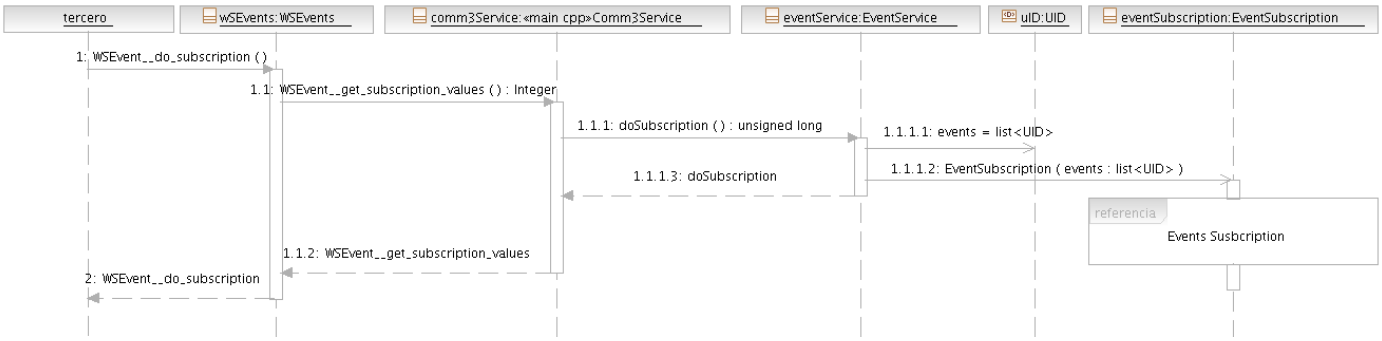


*Figura 17: Diagrama de Secuencia Suscripción a un Grupo de Eventos*

### 2.6.2.2. Paquete Engines

En este paquete se encuentran las clases que definen la lógica para la suscripción y acceso a los datos, entiéndase por datos, puntos, alarmas y eventos. También se define la lógica para las invocaciones de comandos y los mecanismos de seguridad. Estas son las clases que se implementarán para resolver los pedidos de suscripciones y acceso a la información del SCADA realizadas por las aplicaciones externas. Este paquete está estructurado por un grupo de paquetes con vistas a obtener una mejor organización del sistema y los elementos que lo componen. Solo se muestra la parte relacionada con los servicios de intercambio de información relacionada con alarmas y eventos.

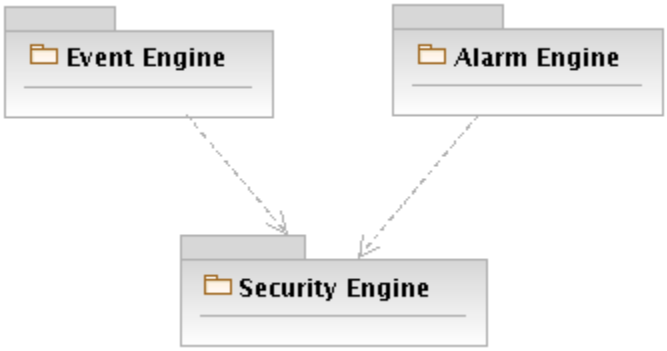


*Figura 18: Relación entre los Paquetes de Diseño del Engines*

#### 2.6.2.3. Paquete Alarm Engine

En este paquete se encuentran las clases que definen la lógica para la suscripción y acceso a las alarmas del sistema. Estas son las clases que se implementarán para resolver las suscripciones y acceso a la información del SCADA relacionada con las alarmas ocurridas, usando los mecanismos de comunicación propuestos por el Middleware. Las funcionalidades se inician en el momento que se establece la comunicación SOAP entre el cliente (tercero) del servicio y el servicio de alarmas. Este paquete de diseño se encarga de registrar las suscripciones, las alarmas que llegan a través de Middleware según la suscripción cuando estas son solicitadas y cada vez que cambian, así como de entregar al servicio la información solicitada por los terceros.

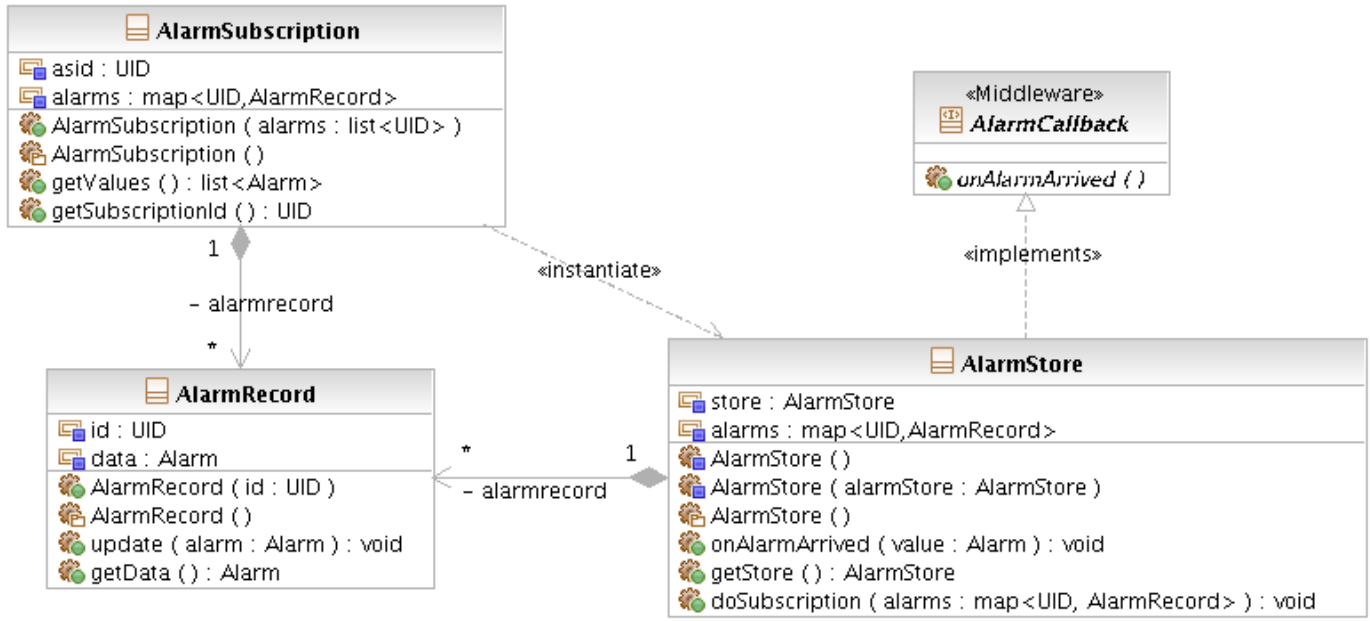


*Figura 19: Diagrama de Clases del Paquete Alarm Engine*

**Descripción de Clases y Métodos**

| ***Clase, AlarmSubscription*** | |
|---|---|
| Esta clase define la lógica para manejar las suscripciones realizadas desde la clase AlarmService que implementa el método de suscripción del servicio web para alarmas. Esta clase tiene la responsabilidad de registrar las suscripciones usando la clase AlarmStore, entregando posteriormente a la clase AlarmService el identificador de la suscripción que asignó el sistema al cliente y se encarga además de entregar los valores actualizados de los alarmas para cada suscripción, y para ello depende las funcionalidades de la clase AlarmRecord. | |
| Nombre del método | Descripción |
| AlarmSubscription(alarms: std::list<UID>) | Este es el constructor de la clase y define la lógica para registrar la suscripción a las alarmas según la lista de Id pasada por parámetros. Para ello pide al Store la lista de Record(registros) 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. AlarmSubscription() asigna un Id de suscripción a cada solicitud de cada cliente. |
| getSubscriptionId(): UID | Este método permite a la clase AlarmService obtener el Id de suscripción que fue a asignado al cliente y entregárselo a |

| | |
|---|---|
| | este. |
| getValues():list<SCADA::Comm3::Alarm> | Este método permite a la clase AlarmService obtener la lista actualizada de las alarmas sucedidas. En la lógica del método se pide los registros de alarmas actualizados en AlarmRecord usando el método *getData()*. |

*Tabla 9: Descripción de la clase AlarmSubscription*

| ***Clase, AlarmStore*** | |
|---|---|
| Esta clase tiene la responsabilidad de verificar en AlarmRecord la existencia de las alarmas solicitadas a través de una suscripción, de no existir, actualiza la lista de los record para así empezar a recibir los valores de las alarmas existentes y las nuevas alarmas solicitadas usando el Middleware. También tiene la responsabilidad de actualizar los valores de las alarmas guardadas en los record de suscripciones a partir de recibir las alarmas desde el Middleware. Hereda de la interfaz AlarmCallback, re-implementando el método onAlarmArrived() y de esta manera puede decidir que hacer cada vez que arriba una alarma desde el SCADA. | |
| Nombre del Método | Descripción |
| doSubscription(std::map<UID, AlarmRecord*>* alarms):void | Este método define la lógica para verificar en el registro de suscripciones de alarmas la existencia de las alarmas pedidas en una nueva suscripción y así saber que alarmas de las que se disparan, se deben almacenar. De no estar registradas las alarmas se crea un nuevo registro y se empieza a escuchar nuevas alarmas además de las ya registradas. |
| getStore(): static AlarmStore* | Este método retorna una instancia de la propia clase que será usado en el momento de decirle al Middleware que desea recibir alarmas especificando el callback (función *getStore()*) que se va a ejecutar cuando llegue una alarma. Esta llamada se realiza en el método *receiveAlarm*() de la clase MiddlewareIO. |
| onAlarmArrived(value: Alarm):void | Este método es la implementación de método onAlarmArrived() de la interfaz AlarmCallback que brinda el Middleware para escuchar las alarmas que son publicadas en un determinado canal permitiendo ejecutar un callback |

| | |
|---|---|
| | cada vez que arriba una alarma. La lógica del método está orientada a verificar que el Id de la alarma que llega se corresponde con alguna de la alarmas registradas por suscripción para de esta manera poder actualizar el valor de la indicada. |

*Tabla 10: Descripción de la clase AlarmStore*

| ***Clase, AlarmRecord*** | |
|---|---|
| Esta clase tiene la responsabilidad de almacenar, mantener actualizada la lista de alarmas por suscripción y entregar la información de las mismas a la clase AlarmSuscription. | |
| Nombre del Método | Descripción |
| getData() : SCADA::Comm3::Alarm | Este método define la lógica para devolver una alarma de Comm3 creada a partir de una alarma del SCADA capturada a través de Middleware, en sí, a partir de la alarma guardada en el atributo _data. |
| upDate(value: Alarm):void | Este método verifica que no exista la alarma que se pasa por parámetros para asignársela al atributo _data de la clase. De de lo contrario elimina la alarma que se pasó por parámetros. |

*Tabla 11: Descripción de la clase AlarmRecord*

**Diagramas de Secuencia**

**Realizar suscripción a un grupo de alarmas**

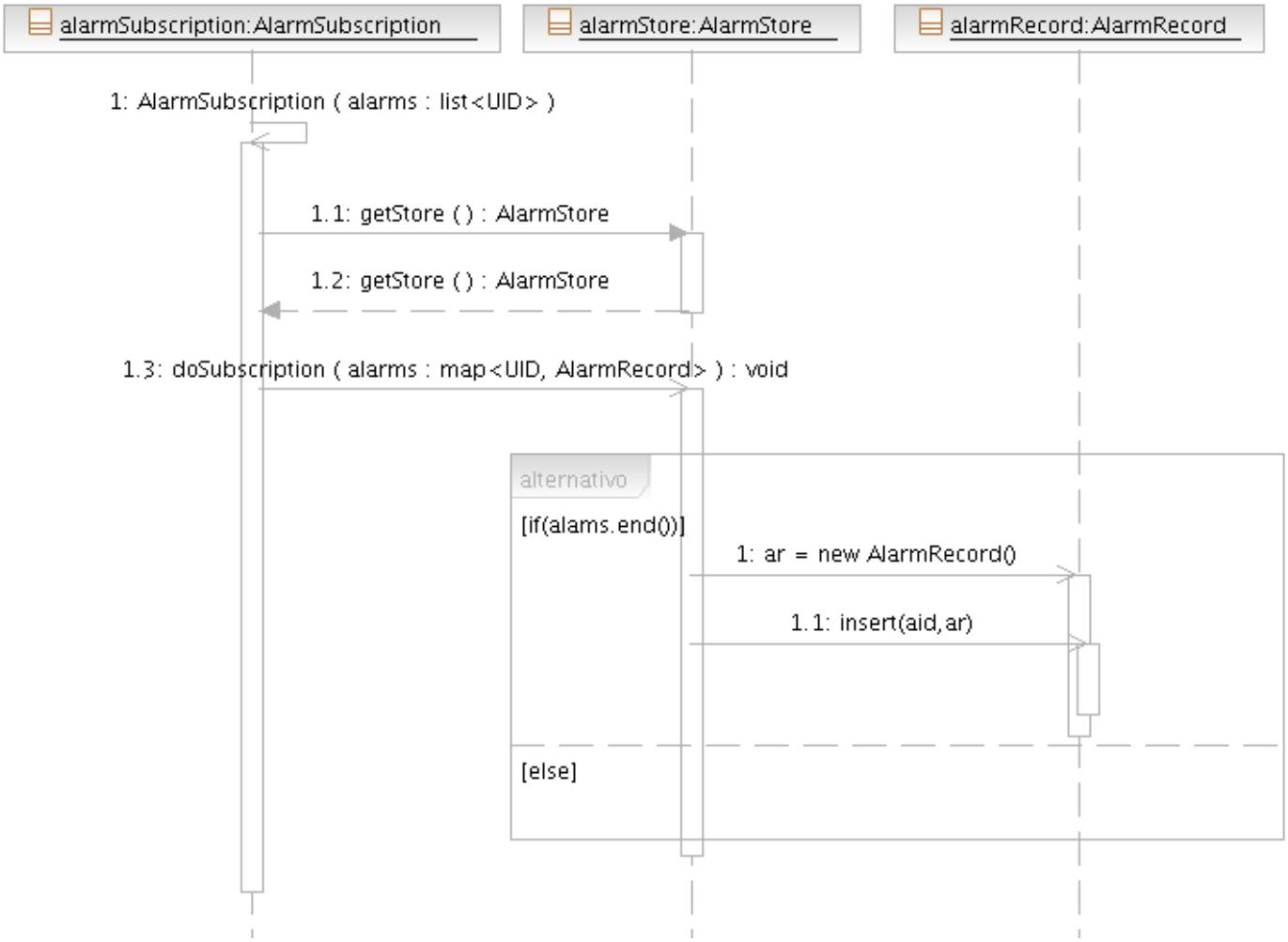


*Figura 20: Diagrama de Secuencia Realizar Suscripción a las Alarmas*

**Recibir alarmas usando las interfaces del Middleware**

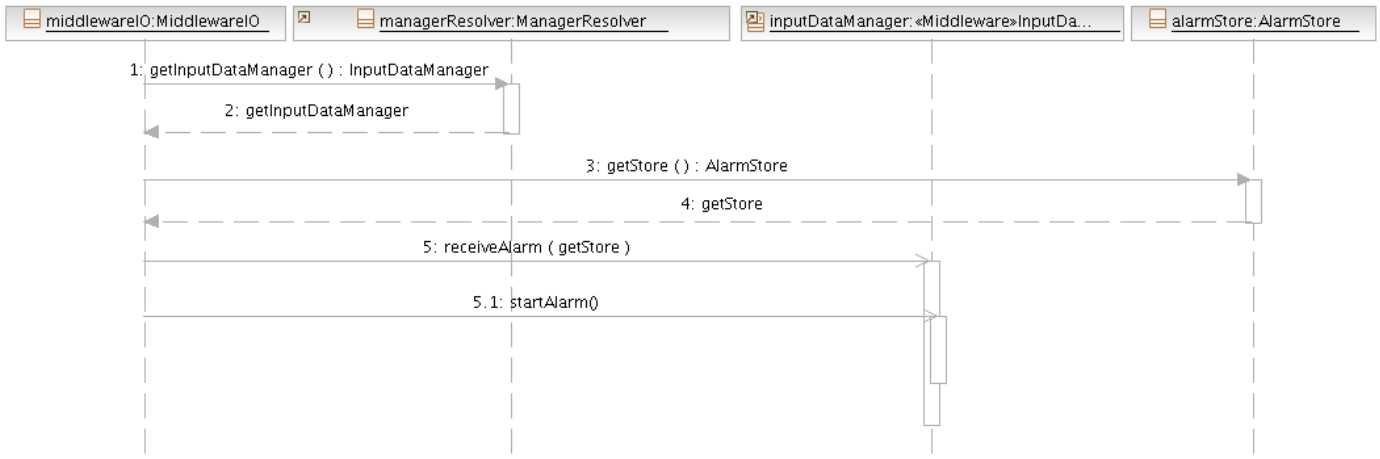


*Figura 21: Diagrama de Secuencia Recibir Alarmas usando el Middleware*

**Registrar alarmas del SCADA en el Servicio de Alarmas de Comm3**

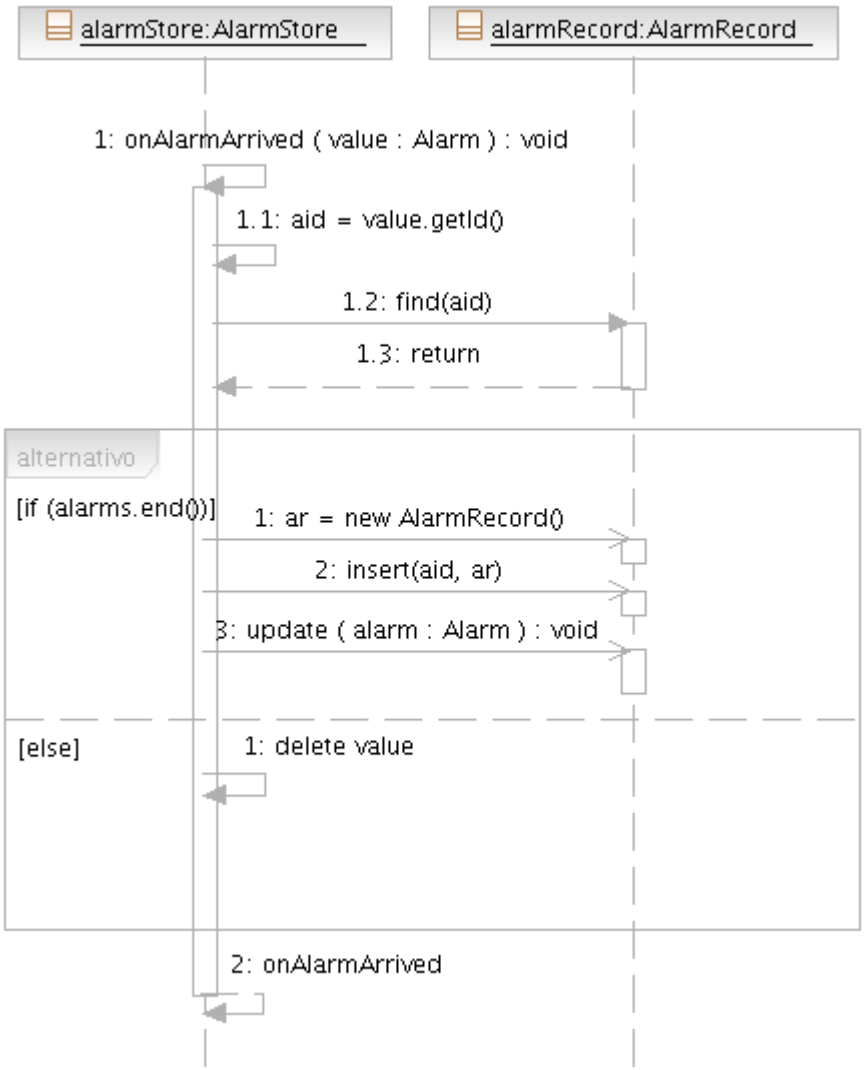


*Figura 22: Diagrama de Secuencia Registrar Alarmas del SCADA en el Servicio*

**Solicitar lista de alarmas para una suscripción**

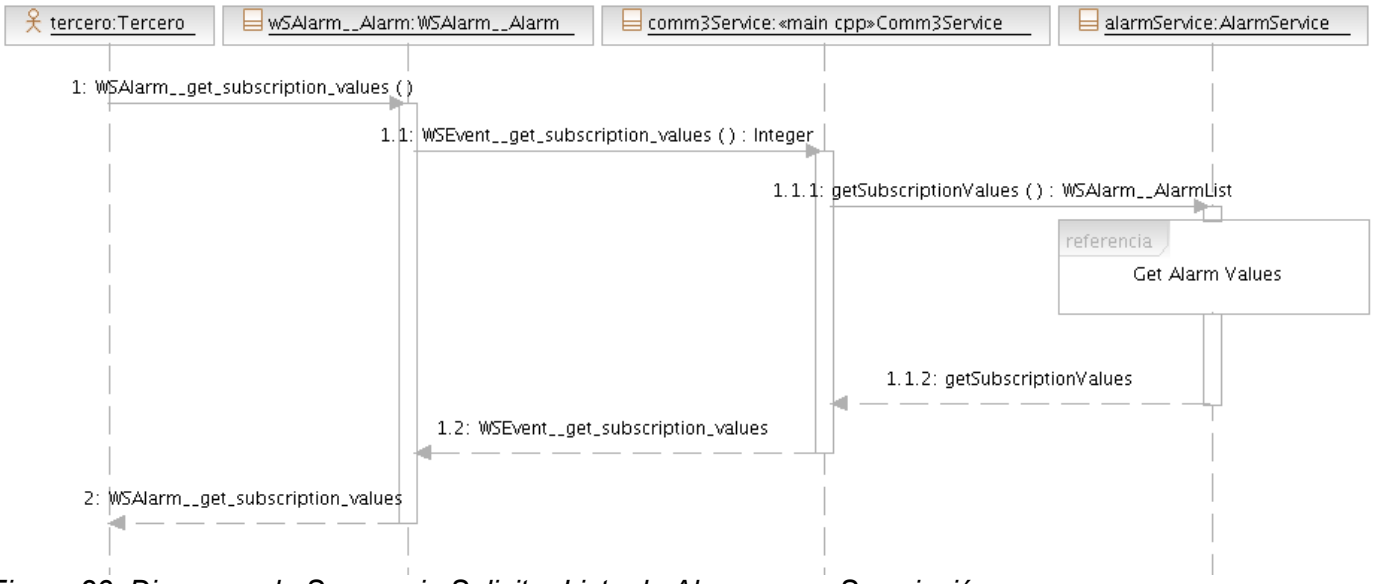


*Figura 23: Diagrama de Secuencia Solicitar Lista de Alarmas por Suscripción*

**Obtener alarmas registradas en el Servicio de Alarmas**

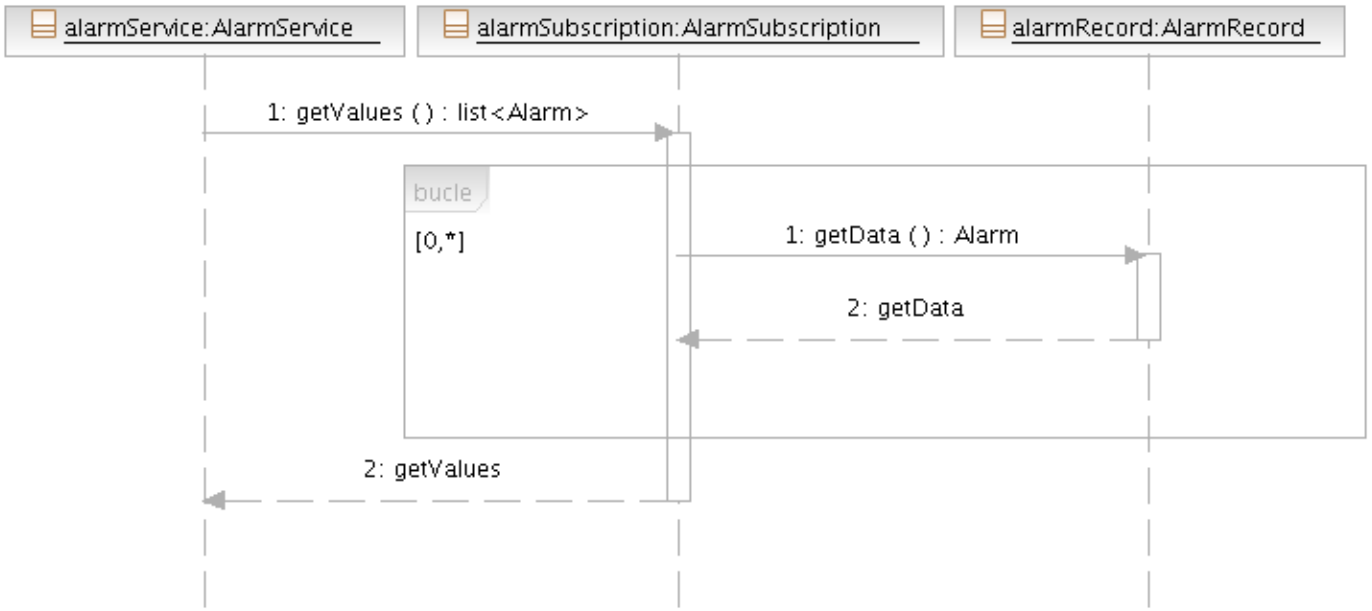


*Figura 24: Diagrama de Secuencia Obtener Alarmas del Servicio*

### 2.6.2.4. Paquete Event Engine

En este paquete se encuentran las clases que definen la lógica para la suscripción y acceso a los eventos del sistema. Estas son las clases que se implementarán para resolver las suscripciones y acceso a la información del SCADA relacionada con los eventos ocurridos, usando los mecanismos de comunicación propuestos por el Middleware. Las funcionalidades se inician en el momento que se establece la comunicación SOAP entre el cliente (tercero) del servicio y el servicio de eventos. Las clases de este paquete de diseño tienen la responsabilidad de registrar las suscripciones, los eventos que llegan a través de Middleware según la suscripción, cuando estos son solicitados y cada vez que cambian, así como de entregar al servicio la información solicitada por los terceros.

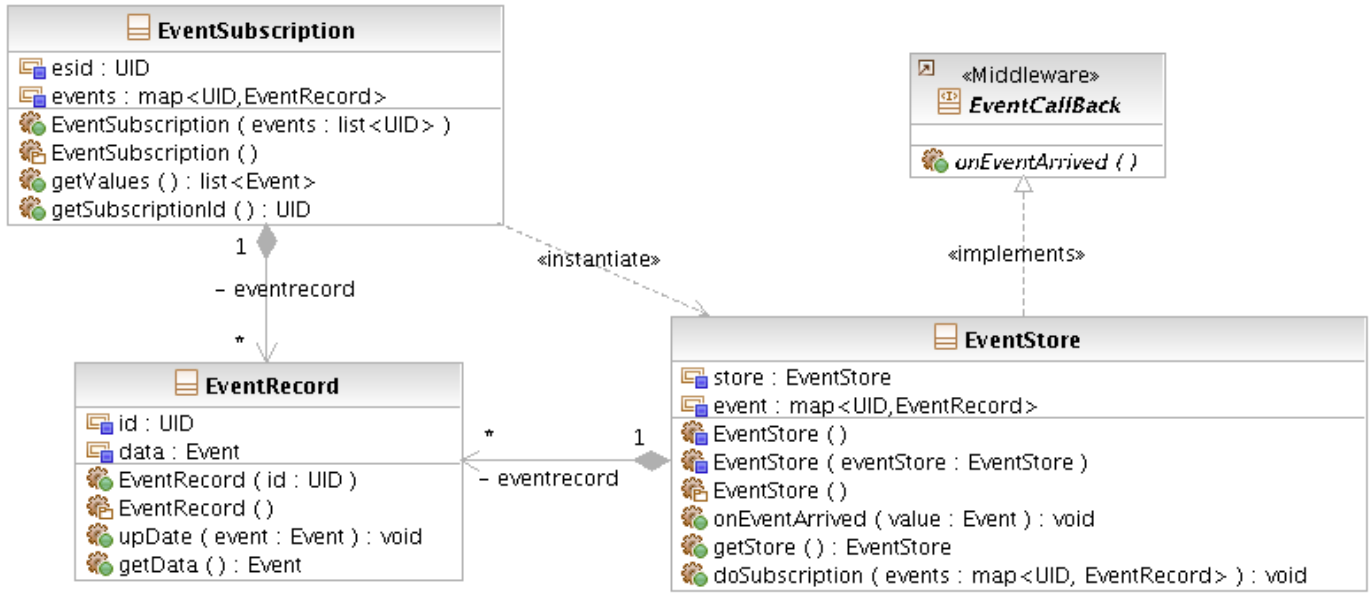


*Figura 25: Diagrama de Clases del Paquete Event Engine*

**Descripción de las clases y sus métodos**

| ***Clase, EventSubscription*** | |
|---|---|
| Esta clase define la lógica para manejar las suscripciones realizadas desde la clase EventService que implementa el método de suscripción del servicio web para eventos. Esta clase tiene la responsabilidad de registrar las suscripciones usando la clase EventStore, entregando posteriormente a la clase EventService el identificador de la suscripción que asignó el sistema al cliente y se encarga además de entregar los valores actualizados de los eventos para cada suscripción, y para ello depende las funcionalidades de la clase EventRecord. | |
| Nombre del método | Descripción |
| EventSubscription(events: std::list<UID>) | Este es el constructor de la clase y define la lógica para registrar la suscripción a los eventos según la lista de Id pasada por parámetros. Para ello pide al Store la lista de Record(registros) 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. También asigna un Id de suscripción a cada solicitud hecha por los clientes. |
| getSubscriptionId(): UID | Este método permite a la clase EventService obtener el Id de suscripción que fue a asignado al cliente y entregárselo a este. |

*Tabla 12: Descripción de la clase EventSubscription*

| ***Clase, EventStore*** | |
|---|---|
| Esta clase tiene la responsabilidad de verificar en EventRecord la existencia de los eventos solicitados a través de una suscripción, de no existir, actualiza la lista de los record para así empezar a recibir los valores de los eventos existentes y los nuevos eventos solicitados usando el Middleware. También tiene la responsabilidad de actualizar los valores de los eventos guardados en los record de suscripciones a partir de recibir los eventos desde el Middleware. Hereda de la interfaz EventCallback, re-implementando el método onEventArrived() y de esta manera puede decidir que hacer cada vez que arriba un evento desde el SCADA. | |
| Nombre del Método | Descripción |
| doSubscription(std::map<UID, EventRecord*>* events):void | Este método define la lógica para verificar en el registro de suscripciones de eventos la existencia de los eventos pedidos en una nueva suscripción y así saber que eventos de los que se disparan, se deben almacenar. De no estar registrados estos eventos se crea un nuevo registro y se empieza a escuchar nuevos eventos además de los ya registrados. |
| getStore(): static EventStore* | Este método retorna una instancia de la propia clase que será usada en el momento de decirle al Middleware que desea recibir eventos especificando el callback (función *getStore()*) que se va a ejecutar cuando llegue un evento. Esta llamada se realiza en el método *receiveEvent*() de la clase MiddlewareIO. |
| onEventArrived(value: Event):void | Este método es la implementación de método onEventArrived() de la interfaz EventCallback que brinda el Middleware para escuchar los eventos que son publicados en un determinado canal permitiendo ejecutar un callback cada vez que arriba un evento. La lógica del método está orientada a verificar que el Id del evento que llega se corresponde con alguno de los eventos registrados por suscripción para de esta manera poder actualizar |

| | |
|---|---|
| | el valor del indicado. |

*Tabla 13: Descripción de la clase EventStore*

| ***Clase, EventRecord*** | |
|---|---|
| Esta clase tiene la responsabilidad de almacenar, mantener actualizada la lista de eventos por suscripción y entregar la información de los mismos a la clase EventSuscription. | |
| Nombre del Método | Descripción |
| getData() : SCADA::Comm3::Event | Este método define la lógica para devolver un evento de Comm3 creado a partir de un evento del SCADA capturado a través de Middleware, en sí, a partir del evento guardado en el atributo _data. |
| upDate(value: Event):void | Este método verifica que no exista evento que se pasa por parámetros para asignárselo al atributo _data de la clase. De de lo contrario se elimina el evento que se pasó por parámetros. |

*Tabla 14: Descripción de la clase EventRecord*

**Diagramas de secuencia del Paquete Event Engine**

**Realizar suscripción a un grupo de eventos**

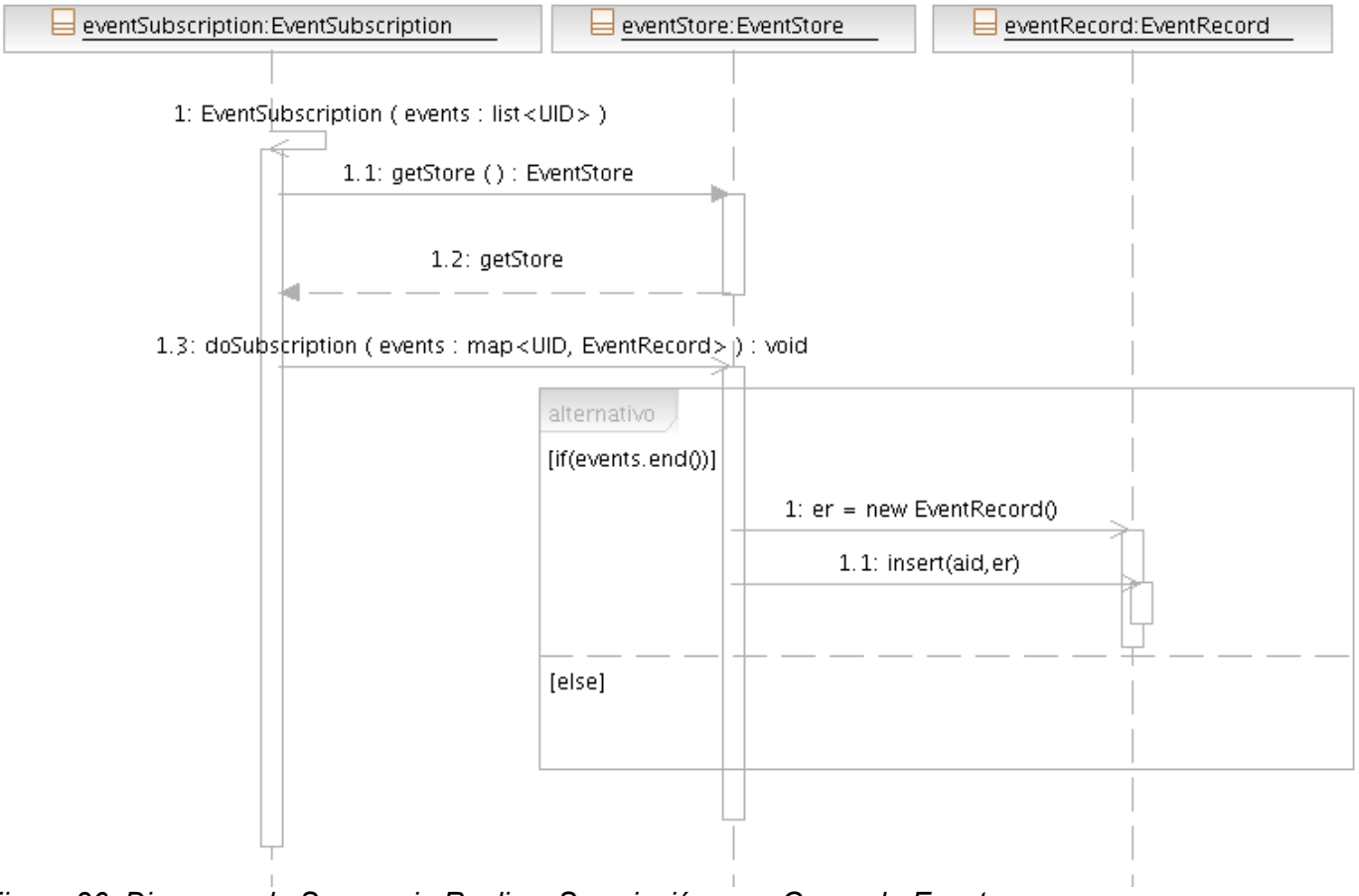


*Figura 26: Diagrama de Secuencia Realizar Suscripción a un Grupo de Eventos*

**Recibir eventos usando las interfaces del Middleware**

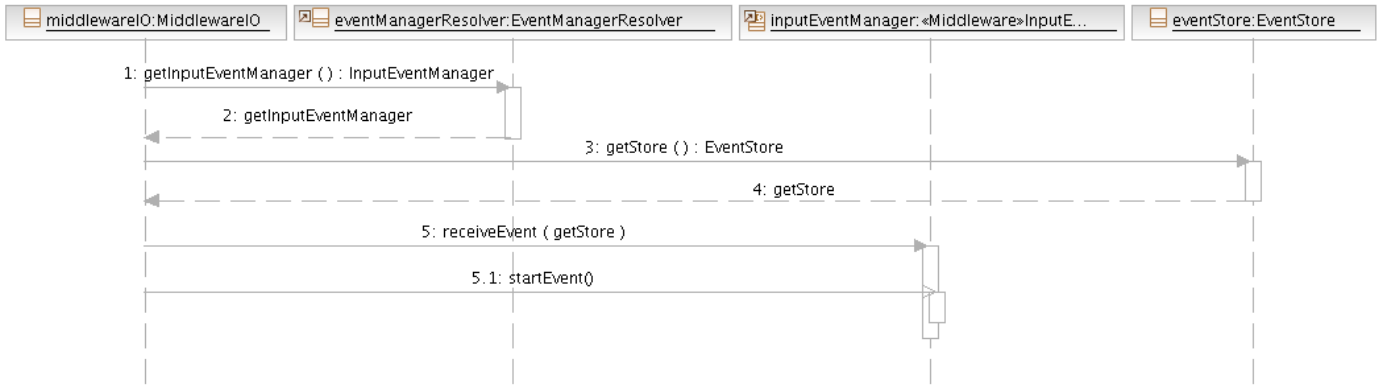


*Figura 27: Diagrama de Secuencia Recibir Eventos usando el Middleware*

**Registrar eventos del SCADA en el Servicio de Eventos**

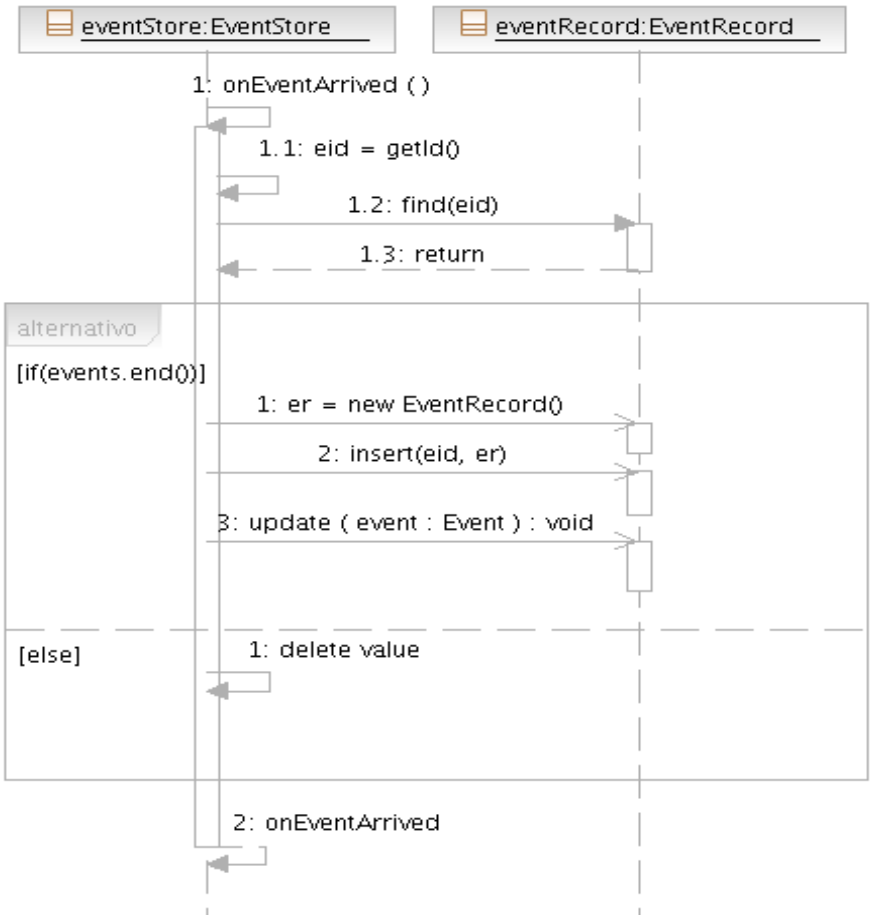


*Figura 28: Diagrama de Secuencia Registrar Eventos del SCADA en el Servicio*

**Solicitar lista de eventos para una suscripción**


*Figura 29: Diagrama de Secuencia Solicitar una Lista de Eventos*

**Obtener eventos registrados en el Servicio**

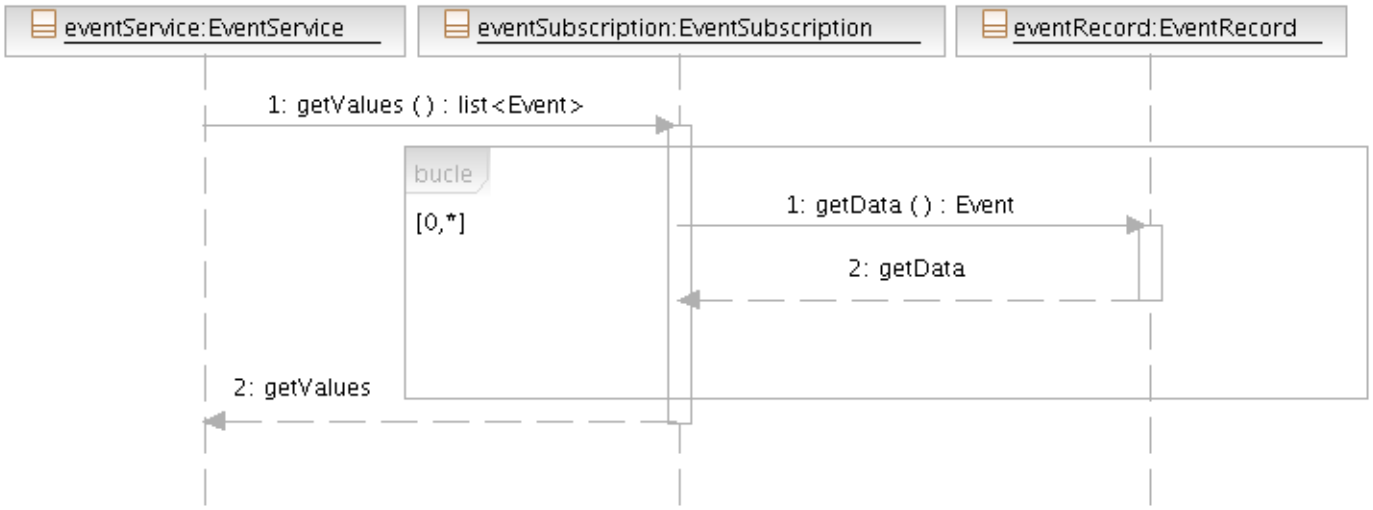


*Figura 30: Diagrama de Secuencia Obtener Eventos registrados en el Servicio*

### 2.6.2.5. Paquete IO Communication

Este paquete contiene las clases que definen la lógica para solicitar a través del Middleware los datos relacionados con alarmas y eventos. Todo se logra a partir de usar las interfaces del Middleware y especificar el tipo de dato que desea escuchar, como un cliente más del Middleware del Guardián del ALBA.

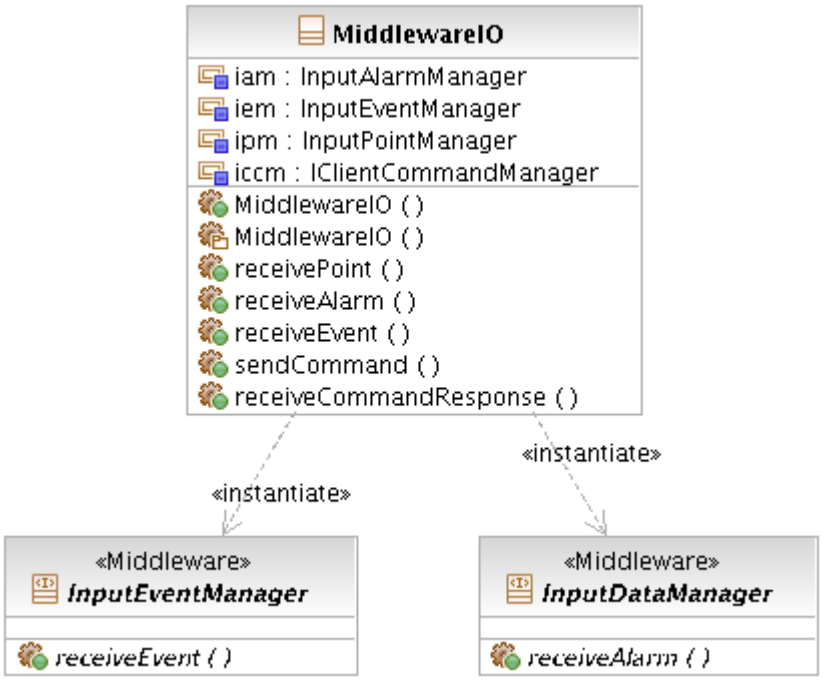


*Figura 31: Diagrama de Clases del Paquete IO Communication*

**Descripción de las clases y sus métodos**

| ***Clase, MiddlewareIO*** | |
|---|---|
| Esta clase tiene como responsabilidad 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 servicio de alarmas. También define la lógica para recibir diferentes tipos de datos usando las interfaces del Middleware, entre ellos las alarmas disparadas en el SCADA. | |
| Nombre del Método | Descripción |

| | |
|---|---|
| receiveAlarm(callback: AlarmCallback) | Este método define la lógica para especificar al Middleware que se desea recibir alarmas, también se especifica el callback, función que se va a ejecutar cuando el sistema reciba una alarma usando el método onAlarmArrived(). |

*Tabla 15: Descripción de la clase MiddlewareIO*

*Las interfaces InputEventManager e ImputDataManager pertenecen al subsistema de comunicación del SCADA Guardián del ALBA, Middleware.*

**Paquete Global**

Este paquete encapsula los diferentes tipos de datos definidos por el servicio de eventos de Comm3 y sus relaciones.

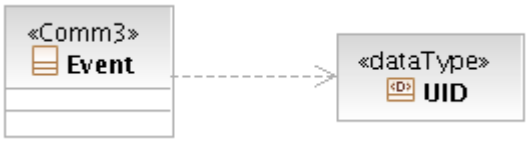


*Figura 32: Diagrama de Clases del Paquete Global*

# CAPITULO 3: CONSTRUCCIÓN DE LA SOLUCION PROPUESTA

El presente capítulo presenta el servicio de acceso a alarmas y eventos en términos de componentes, es decir, archivos de código fuente, scripts, binarios y similares, con el objetivo de mostrar los componentes que se obtienen de implementar las clases y los subsistemas definidos en el diseño. También describe la vista de implantación de los servicios y los casos de pruebas implementados para validar la solución implementada.

## 3.1. Vista de implementación

El principal objetivo de la implementación de los servicios de Comm3 para el intercambio de alarmas y eventos es proveer mecanismos de comunicación entre el SCADA y las aplicaciones externas permitiendo el acceso a la información del Guardián del ALBA relacionada con alarmas y eventos. Con el fin de lograr estos objetivos y brindar las funcionalidades requeridas por el cliente, la implementación se realiza basada en servicios web como una solución universal, usando la herramienta gSOAP que permite implementar servicios web en C++. Esta implementación logra la exposición de las funcionalidades del SCADA en forma de servicios, específicamente el servicio de terceros para el acceso a las alarmas y eventos del proceso y el sistema.

## 3.2. Modelo de implementación

Para lograr un modelo de comunicación abstracto para los clientes, se divide la implementación en tres niveles o capas de abstracción, cada nivel encapsula una parte importante del sistema, permitiendo modificar una de ellas, sin afectar la arquitectura, y en menor medida la interacción con el cliente. Cada nivel tiene una responsabilidad específica en el sistema, pero solo una está en constante interacción con otras aplicaciones o clientes, el nivel de aplicación o de interfaces.

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 (lib) y en algunos casos ejecutables o binarios (.exe).

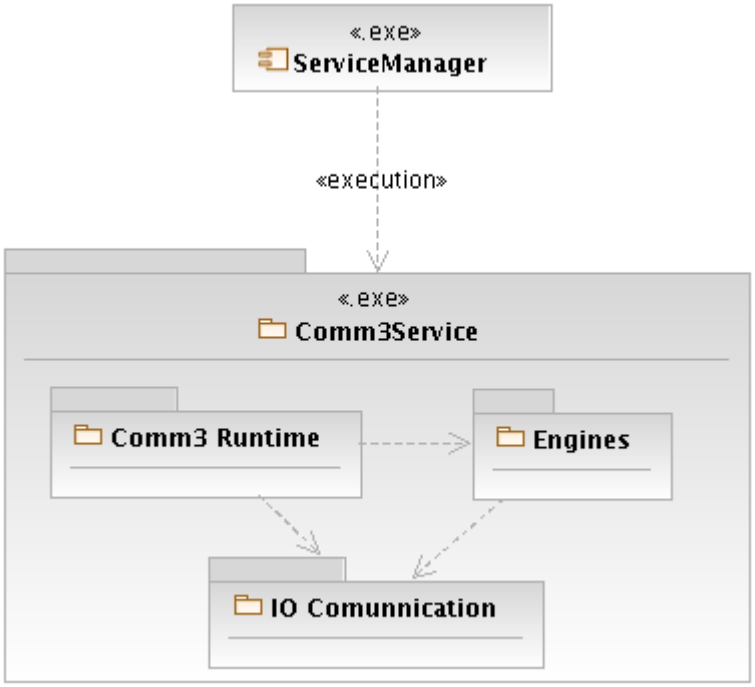


*Figura 33: Modelo de Implementación del Subsistema de Comm3*

### 3.2.1. Descripción de componentes principales

***ServiveManager.exe:*** es el componente encargado de publicar los servicios web en una dirección HTTP que al ser invocados por los terceros, transmite una solicitud al servidor de Comm3, Comm3Service(en este caso, a los servicios de alarmas(*Alarm Service*) y de eventos(*Event Service*) ), a través del protocolo SOAP. Este componente representa la capa de presentación (aplicación) en la arquitectura definida.

***Comm3Service.exe:*** representa el servidor de Comm3 (en este caso los servicios de alarmas y eventos) que implementa los mecanismos de integración con terceros, suscripciones y acceso a los recursos del Guardián del ALBA como sus principales funcionalidades. Este componente encapsula los

componentes *Comm3 Runtime* y *Engines* representantes de la Capa de Negocio de la arquitectura y IO Communication con las funcionalidades relacionadas con la comunicación con el Middleware.

### 3.3. Modelo de implementación por capas

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. El presente apartado describe los componentes que implementan las clases relacionadas con el servicio de alarmas y eventos.

A continuación se presentan los componentes implementados a partir de las clases diseñadas en las capas de Negocio y Comunicación, la capa de presentación no es objetivo del trabajo.

#### 3.3.1. Capa de Lógica de Negocio

Contiene los componentes que implementan los mecanismos de subscripción y acceso a la información del SCADA usando los servicios web. Está representada por los componentes agrupados en los paquetes Comm3 Runtime y Engines. Se presentan los Engines para Alarmas y para eventos.

#### 3.3.1.1. Componentes agrupados en Comm3 Runtime

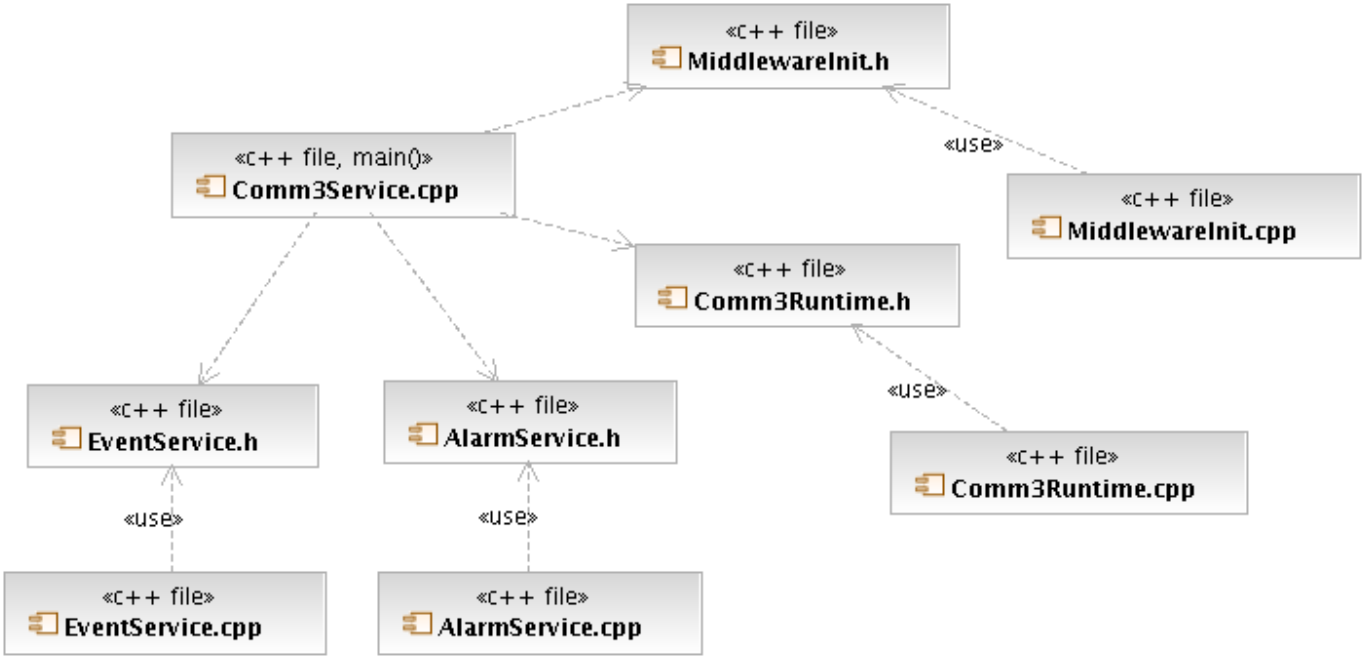


*Figura 34: Diagrama de Componentes del Paquete Comm3 Runtime*

### 3.3.1.2. Componentes agrupados en el Paquete Alarm Engine

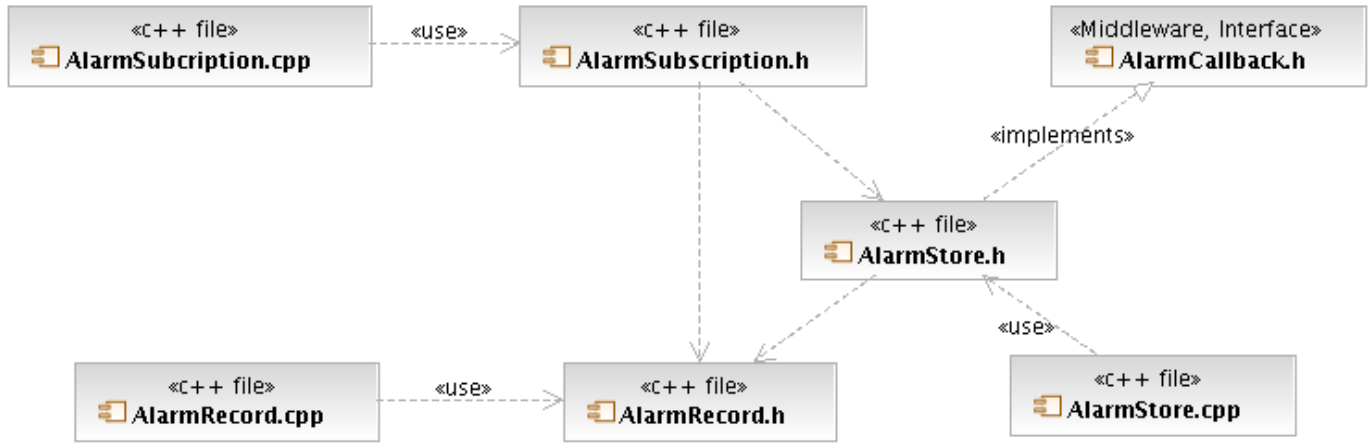


*Figura 35: Diagrama de Componentes del Paquete Alarm Engine*

### 3.3.1.3. Componentes agrupados en el Paquete Event Engine

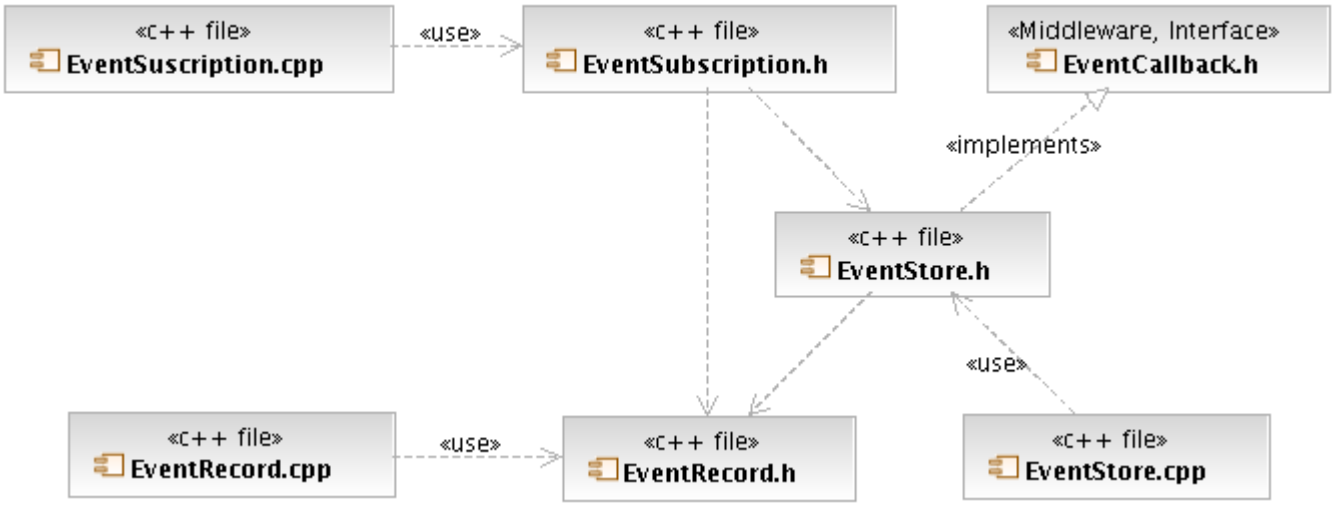


*Figura 36: Diagrama de Componentes del Paquete Event Engine*

### 3.3.2. Capa de comunicación

Contiene los componentes que implementan los mecanismos de comunicación para conectar el subsistema de Comm3 (los servicios de alarmas y eventos) con el resto de los módulos del Guardián del ALABA usando el Middleware. Esta capa está representada por los componentes agrupados en IO Communication.

#### 3.3.2.1. Componentes agrupados en el Paquete IO Communication

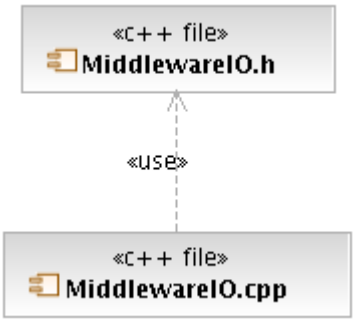


*Figura 37: Diagrama de Componentes del Paquete IO Communication*

### 3.4. Estilo de código utilizado

La codificación siguiendo un estilo o estándar de código garantiza el desarrollo de un producto siguiendo un estilo de codificación único y uniforme.

### 3.4.1. Definición y usabilidad de los 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 implantación.
- Escoger nombres lo suficientemente largo para ser expresivos, pero evitando manejar nombres que dificulten la labor de implantación.

- Evitar nombre que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).
- Evitar redundancia no repitiendo nombre 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 abreviaciones. 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 rehúso de nombres para distinto propósito.

### 3.4.2. 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.

### 3.4.3. Documentación y comentarios

- En el código debe documentarse en forma explicativa los pasos que se van ejecutando.
- Emplear oraciones completas al documentar 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, post 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.

### 3.4.4. Codificación

- Se establece un tamaño de identificación 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 provean 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: +=, *=, /=, -=, ++, --, etc.
- 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 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.*

### 3.5. Vistas más significativas del código

Este epígrafe muestra fragmentos de códigos esenciales para el correcto funcionamiento de lo servicios de Comm3 para el intercambio de alarmas y eventos.

#### 3.5.1. Recepción de alarmas usando las interfaces de Middleware

Este fragmento de código se encarga de recibir alarmas del SCADA a través de Middleware, buscar en el arreglo asociado a las alarmas la existencia de las mismas, si ya existen se actualiza su valor, si no se encuentra se adiciona al arreglo siempre cuando sean alarmas previamente solicitadas por los terceros. La clase a la cual pertenece este método se basa en el patrón *Singleton*, lo que garantiza que solo se pueda obtener una sola instancia de la clase. El uso de este patrón es de vital importancia porque en esta clase se almacenan todas las alarmas que arriban desde SCADA a través del Middleware y así se garantiza un solo punto de acceso a esta información.

```
void AlarmStore::onAlarmArrived(SCADA::Middleware::Alarm* value)
```

```
{
	//actualizar los records de alarmas.
	//buscar el id en la lista.
	UID pid = value->getAlarmID();

	std::map<UID,AlarmRecord*>::iterator it;
	it = this->alarms.find(pid);

	//actualizar el valor de la alarma.
	if (it == this->alarms.end()) // no existe.
	{
#ifdef NOCOM3CONFIG
		//adicionar la alarma al registro actual.
		AlarmRecord* pr = new AlarmRecord(pid);
		alarms.insert(std::make_pair<UID,AlarmRecord*>(pid,pr));

		pr->upDate(value);
#else//desechar la alarma
		//notificar llegada de una alarmas no configurada
		delete value;
#endif
	}
	else
	{
		it->second->upDate(value);
	}
}
```

### 3.5.2. Realizar suscripción a las alarmas del SCADA

Este fragmento de código muestra la implementación de método *doSubcription()* encargado de realizar las suscricpiones a las alarmas. Este implementa esta operación y el control de la concurrencia para el control de acceso a la memoria compartida.

```
void AlarmStore::doSubscription(std::map<UID, AlarmRecord*>* subscript)
{
	std::map<UID, AlarmRecord*>::iterator it, myPIt;
	it = subscript->begin();

	for (; it!= subscript->end();) {
		UID pid = it->first;
```

```
                myPIt = this->alarms.find(pid);

                if (myPIt == alarms.end()) //la alarma no esta registrada.
                {
                        //TODO: Esto es lo que se debe hacer con configuracion:
                        //      Eliminar la alarma de la solicitud y no incrementar
                        //      el iterador.

                        //TODO: Esto es lo que se hace pues no hay config:
                        //      Adicionar la alarma al registro de alarmas global.

                        AlarmRecord* pr = new AlarmRecord(pid);

                        alarms.insert(std::make_pair<UID,AlarmRecord*>(pid,pr));

                        alarms[pid] = pr;
                        it->second = pr;

                        ++it;
                }
                else
                {
                        it->second = myPIt->second;
                        ++it;
                }
        }
}
```

*Las vistas más significativas del código referente al Servicio de Eventos son similares y con igual lógica que las codificaciones anteriores pero para el tipo de dato Evento.*

### 3.6. Vista de Despliegue

La implantación, despliegue del sistema, satisface la solución planteada y muestra la comunicación entre los servicios de Comm3 y los terceros a través de los servicios web. En la figura se puede ver el despliegue de la solcuión en nodos físicos identificando los diferentes nodos y la forma de conexión entre ellos.

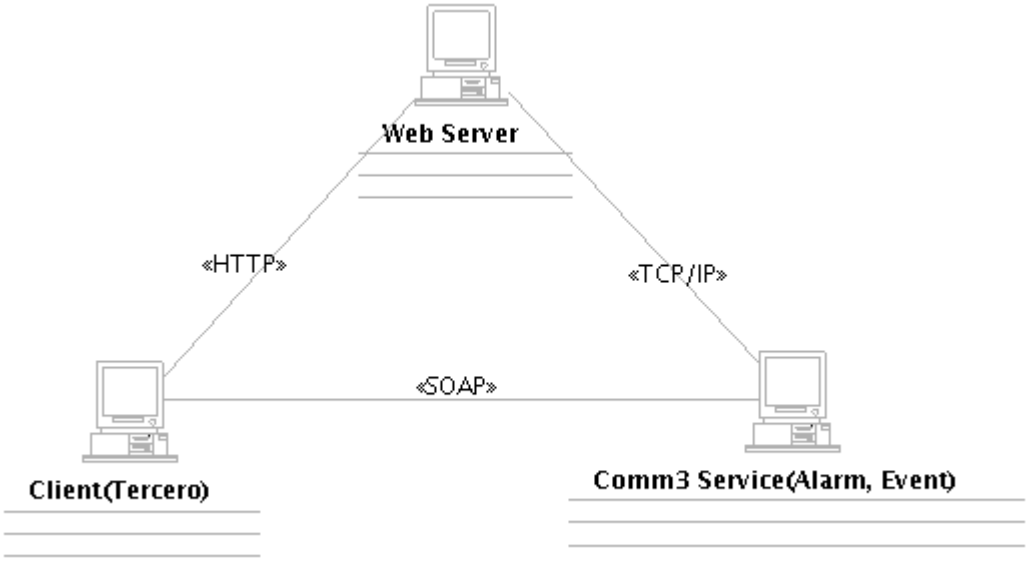


*Figura 38: Vista de Despliegue de los Servicios de Alarmas y Eventos*

### 3.6.1. Descripción de los nodos físicos

***Web Server (Servidor Web):*** En este nodo radica la instalación del Administrador de Servicios (*Services Manager*) basado en un servidor web encargado de administrar y exponer los servicios web en una dirección HTTP a completa disposición de los clientes o aplicaciones externas.

***Comm3 Services (Servicios de Alarmas y Eventos):*** en este nodo se encuentra instalado subsistema de Comm3 del SCADA Guardián del ALBA, en este caso en particular, los servicios de Alarmas y Eventos. Este se comunica por TCP/IP con el Servidor Web para registrar los servicios que maneja. Este nodo recibe peticiones SOAP desde el Cliente y envía respuestas vía SOAP.

***Client (Cliente o Tercero):*** En este nodo radica la instalación de una aplicación personalizada que requiere acceder a la información del SCADA a través de los servicios web y para ello se comunica por HTTP con el Servidor Web para acceder a los servicios que este expone. Después de conocer la descripción de los servicios web existentes puede realizar invocaciones a los servicios de alarmas y eventos vía SOAP.

### 3.7. Ejecución de las pruebas

Las pruebas de software constituyen una parte importante de cualquier desarrollo de software garantizando la calidad del producto final desarrollado. Los métodos de pruebas más utilizados son los de Caja blanca y Caja Negra. Los métodos de caja blanca están orientados a probar el código y su funcionamiento haciendo necesario que los probadores conozcan el código y la lógica interna del mismo; las pruebas de caja negra evalúan las funcionalidades a nivel de interfaz, los resultados de las entradas y salidas de datos al sistema, en fin, como se comporta la aplicación ante la interacción con los usuarios u otros sistemas.

A continuación se muestran los casos de prueba de caja negra confeccionados para probar las principales funcionalidades de los servicios de alarmas y eventos especificadas como requisitos.

#### 3.7.1. Ambiente de pruebas

A continuación se muestran los datos de las máquinas y el sistema operativos utilizados en la ejecución de las pruebas.

**Recursos Físicos**

- Procesador Intel(R) Core(TM)2 Duo CPU E4500 @ 2.20GHz
- Tarjeta Madre Intel
- Memoria RAM de 1GB
- Red alámbrica o cableada

**Recursos Lógicos**

- Debian 5 (Lenny), Kernel Linux 2.6.26-1-686
- Middleware versión c2.2.1
- Cliente de Middleware para envío de alarmas, *MiddlewareAlarmTestOutput*
- Cliente de Middleware para envío de eventos, *MiddlewareEventTestOutput*
- Código del Servicio de Alarmas, *Comm3AlarmService*
- Código del Servicio de Eventos, *Comm3EventService*

- Código del cliente de Comm3 para Alarmas, *Comm3AlarmReceiverGui*
- Código del cliente de Comm3 para Eventos, *Comm3EventReceiverGui*

### 3.7.2. Diseños de casos de prueba

En esta sección se muestran los casos de pruebas diseñados para probar las funcionalidades más críticas de los servicios de alarmas y eventos y los resultados de su aplicación.

#### 3.7.2.1. CPR 1: Realizar subscripción de alarmas

**Descripción**

El caso de prueba permite comprobar la realización exitosa de una suscripción a un grupo de alarmas en el Servicio de Alarmas de Comm3. Un cliente solicita la suscripción a una lista de alarmas pasando por parámetros la lista de identificadores, los que son almacenados en servicio, y este proporciona al cliente un identificador de la suscripción realizada.

**Flujo central**

- Se obtiene la descripción del servicio de alarmas.
- Se crea un cliente de este servicio web.
- Se selecciona la opción subscribir a una lista de alarmas.
- El cliente recibe el identificador de la suscripción realizada.

**Condiciones de Ejecución**

- Deben estar ejecutándose los servicios de Middleware, NameService (Servicio de Nombres), NotifyService (Servicio de Notificación), ChannelRegistry (Servicio para registro de Canales) y un cliente de Middleware para el envío de alarmas.
- Se debe ejecutar el Servicio de Alarmas a apuntando al Middleware en ejecución y posteriormente el cliente de Comm3, ambos en diferentes máquinas para garantizar comunicaciones de forma remota en la ejecución de las pruebas.

**Resultados**

| Caso # | Clases Válidas | Clases Inválidas | Resultados Esperados | Resultados Obtenidos | Observaciones |
|---|---|---|---|---|---|
| 1 | Se registran las 10 alarmas a las que el cliente desea suscribirse y posteriormente realiza la suscripción. | | 1. La aplicación debe mostrar una tabla en la interfaz preparada para recibir información de las 10 alarmas solicitadas.<br><br>2. Debe mostrar también un mensaje con los identificados de suscripción asignado por el servicio al cliente. | 1. Se observó que la interfaz quedó preparada para recibir las 10 alarmas solicitadas.<br><br>2. La aplicación mostró el identificador de la suscripción realizada. | |

*Tabla 15: Resultados de las pruebas para suscripción de alarmas*

#### 3.7.2.2. CPR 2: Obtener valores de las alarmas para una suscripción

**Descripción**

El caso de prueba permite comprobar la realización exitosa durante la solicitud de los valores de alarmas a partir de una suscripción en el Servicio de Alarmas de Comm3. Un cliente solicita una lista de alarmas, pasando por parámetros el identificador de la suscripción, y el servicio le devuelve los valores actualizados de las alarmas siempre y cuando estas existan o hayan sucedido en el SCADA Guardián del ALBA.

**Flujo central**

- Se obtiene la descripción del servicio de alarmas.
- Se crea un cliente de este servicio web.
- Se selecciona la opción subscribir a una lista de alarmas.
- El cliente recibe el identificador de la suscripción realizada.

- El cliente Se selecciona la opción Iniciar Lectura de alarmas para la suscripción anterior y este termina recibiendo la información de las alarmas solicitadas.

**Condiciones de Ejecución**

- Deben estar ejecutándose los servicios de Middleware, NameService (Servicio de Nombres), NotifyService (Servicio de Notificación), ChannelRegistry (Servicio para registro de Canales) y un cliente de Middleware para el envío de alarmas.
- Se debe ejecutar el Servicio de Alarmas a apuntando al Middleware en ejecución y posteriormente el cliente de Comm3, ambos en diferentes máquinas para garantizar comunicaciones de forma remota en la ejecución de las pruebas.

**Resultados**

| Caso # | Clases Válidas | Clases Inválidas | Resultados Esperados | Resultados Obtenidos | Observaciones |
|---|---|---|---|---|---|
| 1 | 1. Se registran las 10 alarmas a las que el cliente desea suscribirse y realiza la suscripción.<br><br>2. Posteriormente se inicia la lectura usando las opciones que brinda la interfaz. | | 1. La aplicación debe mostrar una tabla en la interfaz preparada para recibir información de las 10 alarmas solicitadas.<br><br>2. Debe mostrar también un mensaje con el identificador de suscripción asignado por el servicio al cliente.<br><br>3. La interfaz de mostrar en la tabla los valores actualizados de las alarmas solicitadas. | 1. Se observó que la interfaz quedó preparada para recibir las 10 alarmas solicitadas.<br><br>2. La aplicación mostró el identificador de la suscripción realizada.<br><br>3. La aplicación mostró la lectura en una tabla de los valores actualizados de las alarmas pedidas en la suscripción. | |

*Tabla 16: Resultados de las pruebas para lectura de alarmas por suscripción*

*El **Anexo IV,** muestra los casos de prueba relacionados con las funcionalidades del Servicio de Integración con Terceros para el intercambio de Eventos.*

## CONCLUSIONES

Al término de la presente investigación para el desarrollo de los Servicios de Integración con Terceros para el intercambio de información relacionada con Alarmas y Eventos del sistema SCADA Guardián del ALBA, se concluye que:

- Los Servicios Alarmas y Eventos desarrollados permiten a las aplicaciones externas comunicaciones efectivas y seguras con el SCADA Guardián del ALBA posibilitando el acceso a la información relacionada con alarmas y eventos.
- Se realizaron un grupo de pruebas de funcionalidad que permitieron validar la implementación de los servicios logrando un producto acorde con las exigencias del cliente y listo para ser implantado en el SCADA Guardián del ALBA.
- Con las investigaciones e implementaciones realizadas, relacionadas con los Servicios de Terceros para Alarmas y Eventos se logró entregar varios compromisos pactados con la parte venezolana que aportaron al país una cantidad de 80 mil dólares aproximadamente.
- La implementación de la solución basada en tecnologías libres representa un gran aporte a la soberanía tecnológica de Venezuela permitiendo ahorrar miles de dólares por concepto de compra de licencias de software.

## RECOMENDACIONES

- Integrar los Servicios de Alarmas y Eventos con los servicios de Variables, Seguridad y Comandos del módulo de Comunicación con Terceros.
- Implementar mecanismos de tratamiento de errores (excepciones) en los servicios implementados.
- Hacer un estudio sobre la posibilidad de implementar los servicios sobre la base de una arquitectura completamente orientada a servicios como es el caso de SOA.
- Implementar los Servicios de Alarmas y Eventos usando una tecnología Middleware como ICE para lograr una solución de comunicación con terceros más robusta y segura y con mecanismos de transferencia de la información en tiempo real y a una gran velocidad.
- Implantar la solución en otros sistemas buscando identificar nuevas funcionalidades y validar la portabilidad de la misma.

## REFERENCIAS BIBLIOGRÁFICAS


1. **ABB. 2009.** OPC Foundation. *OPC Unified Architecture.* [En linea] 2009. [Citado: Febrero 19, 2009.] http://www.opcfoundation.org/Products/ProductDetails.aspx?CM=1&RI=8931&CU=13.
2. **Apter. 2009.** Alternativas Productivas en Tecnologías. *Sistemas de Planificación de Recursos, ERP.* [En linea] 2009. [Citado: Abril 24, 2009.] http://www.apterca.com/component/content/article/54.
3. **Bentek, Systems. 2009.** SCADA and Telemetry Solutions. *Internet and Web-based SCADA.* [En linea] 2009. [Citado: Marzo 12, 2009.]
4. **Booch, Grady, Jacobson, Ivar and Rumbaugh, James. 2007.** *El Lenguaje Unificado de Modelado.* s.l. : ADDISON-WESLEY, 2007. 978-84-7829-087-1.
5. **—. 1999.** *El Proceso Unificado de Desarrollo de Software.* s.l. : ADDISON-WESLEY, 1999. ISBN 84-7829-036-2.
6. **Charrouf, Raed. 2009.** Proyecto SCADA Nacional. *Especificación de Sumario de Alarmas.* [En linea] 2009. [Cited: Marzo 10, 2009.]
7. **—. 2009.** Proyecto SCADA Nacional. *Especificación de Eventos.* [En linea] 2009. [Citado: Marzo 10, 2009.]
8. **Departamento de Ingeniería de Sistemas, Automaticas.** Escuela Superior de Ingenieros de Bilbao. *Comunicaciones Industriales.* [En linea] [Citado: Febrero 17, 2009.] http://www.disa.bi.ehu.es/spanish/ftp/material_asignaturas/Laboratorio%20de%20Comunicaciones.
9. **Ferrari, Juan Pablo. 2005.** Monografías. *Sistemas de control distribuido.* [En linea] 2005. [Citado: Enero 17, 2008.]
10. **gSOAP, Toolkit. 2009.** *The gSOAP Toolkit for SOAP Web Services and XML-Based Applications.* [En linea] 2009. [Citado: Mayo 19, 2009.] http://www.cs.fsu.edu/~engelen/soap.html.
11. **Henning, Michi y Spruiell, Mark. 2009.** *Distributed Programming with Ice.* [En linea] 2009. [Citado: Mayo 11, 2009.] http://www.zeroc.com/download/Ice/3.3/Ice-3.3.1.pdf.
12. **Herrera Vázquez, Moisés. 2008.** *Introducción a la arquitectura del "Guardián del ALBA".* [En linea] 2008. [Citado: Febrero 5, 2009.]
13. **Herrera Vázquez, Moisés and Pérez Javier, Maikel. 2008.** AIT-DST Mérida Venezuela, UCI y UCLV. Proyecto SCADA Guardián del ALBA. *Especificación de Comunicación con Terceros.* [En linea] 2008. [Citado: Febrero 10, 2009.]

14. **Massaro, Simone. ICONICS Inc, 2008. 2008.** Plant Engineering. *What is OPC UA and how does it affect your world?* [En linea] Mayo 5, 2008. [Citado: Abril 12, 2009.] http://www.plantengineering.com/article/183110What_is_OPC_UA_and_how_does_it_affect_your.
15. **MatrikonOPC, 2009. 2009.** *Gestión de Eventos.* [En linea] 2009. [Citado: Abril 20, 2009.] http://www.matrikonopc.es/products/opc-event-management/index.aspx.
16. **OPC Foundation, 2002. 2002.** Industry Standard Specification. Final Release. *Alarms and Events Custom Interface.* [En linea] Octubre 2, 2002. [Citado: Febrero 5, 2009.] https://www.opcfoundation.org/Login.aspx?PageState=100.
17. **OPC Foundation, 2009. 2009.** *Qué es OPC?* [En linea] 2009. [Citado: Febrero 20, 2009.] http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC.
18. **PDVSA. 2009.** *Sitio oficial de Petróleos de Venezuela.* [En linea] Abril 2, 2009. [Citado: Mayo 20, 2009.] http://www.pdvsa.com/.
19. **Pérez, Fedel. 2007. 2007.** Escuela Técnica Superior de Ingeniería de Bilbao. Departamento de Ingeniería de Sistemas y Automática. *OPC. Conceptos Fundamentales.* [En linea] Febrero 22, 2007. [Citado: Enero 25, 2009.]
20. **Progea, Movicon. 2008.** Movicon 11. *The innovative XML based Scada/HMI.* [En linea] 2008. [Citado: Abril 19, 2009.] http://www.progea.com/software-automation-scada/movicon-11/movicon-11-xml-based.html.
21. **Sepúlveda, José. 2006. 2006.** Revista Electroindustria. MES. *Sistemas de Ejecución de Manufactura.* [En linea] Octubre 2006. [Citado: abril 24, 2009.] http://www.emb.cl/electroindustria/articulo.mv?xid=166&tip=7.
22. **Sistemas, SCADA,2008. 2008.** *Fundamento Teórico.* [En linea] 2008. [Citado: Febrero 21, 2009.] http://www.alfinal.com/Temas/sistemascada.shtml.
23. **Villa, David. 2008.** Escuela Superior de Informática. Universidad de Castilla-La Mancha. Arquitectura de Redes. *Implementación libre del estándar de DDS (Data Distribution Service) con ZeroC Ice.* [En linea] Octubre 29, 2008. [Citado: Marzo 16, 2009.] http://arco.esi.uclm.es/es/.
24. **Villarreal, Elizabeth. 2009. 2009.** Proyecto SCADA Nacional. *Glosario de Términos.* [En linea] 2009. [Citado: Mayo 10, 2009.]
25. **Wikipedia. 2009.** *Servicios Web.* [En linea] 2009. [Citado: Abril 8, 2009.] http://es.wikipedia.org/wiki/Servicios_Web.

26. **Wikipedia, Electrónica de contro. 2008.** *Electrónica de control. SCADA.* [En linea] 2008. [Citado: Febrero 5, 2009.]
27. **ZeroC, Labs. 2009.** *ZeroC's implementation of Ice.* [En linea] 2009. [Citado: Marzo 16, 2009.] http://www.zeroc.com/labs/index.html.
28. **ZeroC, ZeroC™. 2009.** The Home of Ice. *The Internet Communication Engine(Ice).* [En linea] 2009. [Citado: Febrero 19, 2009.] http://www.zeroc.com/ice.html.

## BIBLIOGRAFÍA


1. *A Framework for Service-Oriented Computing with C and C++ Web Service Components, ACM Transactions on Internet Technologies.* **Engele, Robert A. van. 2008.** 12, 2008, Vol. 8.
2. **AdVosol. 2009.** Advanced OPC Solutions. *OPC XML Webservices.* [En linea] 2009. [Citado: Mayo 20, 2009.] http://www.advosol.us/c-2-opc-xml-webservices.aspx.
3. **Dagoberto, Montero, B, Barrantes David and M, Quirós Jorge. 2008.** Portal de Automatización Industria. *Introducción a los sistemas de control, supervisor y de adquisición de datos (SCADA).* [En linea] 2008. [Citado: Noviembre 15, 2008.] http://www.infoplc.net/Documentacion/Docu_SCADA/infpPLC_net_Introduccion_Sistemas_SCADA.html.
4. **Foundation, Eclipse. 2009.** *Eclipse IDE for C/C++ Developers.* [En linea] 2009. [Citado: Abril 27, 2009.] http://www.eclipse.org/downloads/moreinfo/c.php.
5. **IBM. 2009.** *RSA: Rational Software Architect.* [En linea] 2009. [Citado: Abril 28, 2009.] http://www-01.ibm.com/software/awdtools/architect/swarchitect/index.html.
6. **IBM, Rational. 2009.** Rational Software Architect Standard Edition. *Features and benefits .* [En linea] 2009. [Citado: Mayo 15, 2009.] http://www-01.ibm.com/software/awdtools/swarchitect/standard/features/.
7. **J. J. Domínguez Jiménez, A. Estero Botaro, I. Medina Bulo, M. Palomo Duarte y F. Palomo Lozano. 2008.** *El reto de los servicios Web para el software libre.* [En linea] 2008. [Citado: Febrero 20, 2009.]
8. **Javier, Maikel Pérez. 2009.** Proyecto SCADA Guardián del ALBA. *Selección de Tecnologías para el desarrollo del Subsistema de Comunicación con Terceros.* [En linea] 2009. [Citado: Febrero 23, 2009.]
9. **Lozano, Carlos de Castro y Romero, Morales Cristóbal.** Universidad de Córdoba. *Asignatura: Interfaz Hombre-Máquina, Introducción a SCADA.* [En linea] [Citado: Diciembre 20, 2008.] http://www.uco.es/grupos/eatco/automatica/ihm/descargar/scada.pdf.
10. **Maikel Pérez Javier, José Antonio Aragón Cáceres. 2009.** Proyecto SCADA Guardián del ALBA. *Documento de entrega formal del prototipo de Subsistema de Comunicación con Terceros.* [En linea] 2009. [Citado: Febrero 23, 2009.]

11. **Modesti, Mario R.** Sistemas de supervisión y monitoreo. *Introducción a los sistemas SCADA.* [En linea] [Citado: Enero 20, 2009.] http://www.profesores.frc.utn.edu.ar/industrial/sistemasinteligentes/UT4/4.htm.
12. **Morales, Carlos Andres. 2008.** Universidad Nacional de Colombia. *Estado del Arte: Servicios Web.* [En linea] 2008. [Citado: Abril 10, 2009.] http://camoralesma.googlepages.com/articulo2.pdf.
13. **Multitarea, Multiprograma &.** [En linea] [Citado: Abril 10, 2009.] http://www.zator.com/Cpp/E1_7b.htm..
14. **Penín, Aquilino Rodríguez. 2007.** *SISTEMAS SCADA - GUÍA PRÁCTICA.* s.l. : MARCOMBO, S.A, 2007. 9788426714558.
15. **Ramadan Fan, Saudi Aramco, Dhahran, Dr. Lahouari Cheded, Dr. Onur Toker. 2008.** An XML Web Services Approach to SCADA Systems Design. *King Fahd University of Petroleum and Minerals. .* [En linea] 2008. [Citado: Enero 23, 2009.]
16. **W3C, World Wide Web,2009. 2009.** *Web Services Activity. .* [En linea] 2009. [Citado: Marzo 26, 2009.] http://www.w3.org/2002/ws/.
17. **ZetCode. 2009.** *C/C++ development in Eclipse IDE.* [En linea] 2009. [Citado: Abril 27, 2009.] http://zetcode.com/articles/eclipsecdevelopment/.
18. **Zhang, Robert A. van Engelen y Wei. 2008.** *Identifying Opportunities for Web Services Security Performance Optimizations, to appear in the proceedings of the IEEE Services Computing Conference (SCC),.* [En linea] 2008. [Citado: Marzo 28, 2009.]

## ANEXOS

**Anexo I**: Matriz de decisión elaborada para la evaluación de tecnologías candidatas para implementar cada uno de los servicios del Módulo de Comm3 del SCADA Guardián del ALBA. Esta matriz permite la toma de decisiones durante la fase de desarrollo de la arquitectura de la aplicación. Los elementos como, criterios, opciones, importancia y puntajes fueron analizados durante el desarrollo del trabajo.

**¿Cómo calificar una opción?**

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

*Tabla 18: Tipos de Calificaciones*

### Evaluación de las alternativas

| | | ALTERNATIVAS | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Modelo de Decision | | 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 | 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 19: Evaluación Alternativa*

### Detalles del los puntajes otorgados

| Criterio | Alternativa | Justificación |
|---|---|---|
| Facilidad de implementación. | OPC | El puntaje arrojado es porque el costo de implementar cada unos de los estándares OPC puede ser muy alto más cuando se necesitan otras herramientas para dar una solución adecuada. La evaluación se basa en los criterios del equipo de desarrollo de driver OPC del Guardián del ALBA. |
| Facilidad de Implementación. | OPC UA | OPC UA es la selección ideal, es una especificación que describe todas las funcionalidades que se deben exponer a los terceros, permitiendo además, exponer estas funcionalidades como |

| | | |
|---|---|---|
| | | servicios web, sin embargo, tiene como principal desventaja que esta especificación es propietaria y no se puede obtener para de manera libre implementar los servicios y/o funcionalidades que especifica. Es por eso que arrojó un puntaje de 0 en este aspecto. |
| Facilidad de implementación | Web Services | El puntaje que arrojó la evaluación de este criterio se debe a que los WS son fáciles de implementar, usar y manejar. Puede ser más fácil su implementación si contamos con herramientas como gSOAP para implementar los servicios. Esta evaluación se basa en la experiencia alcanzada por el equipo de desarrollo en la implementación de un prototipo para acceso a variables. |
| Facilidad de implementación | ICE | El puntaje alcanzado en este criterio está dado porque cualquier tecnología Middleware con las prestaciones de ICE o similares se puede decir que es compleja, sin embargo, si el equipo de desarrollo tiene conocimiento sobre la tecnología el puntaje podría ser mucho mayor. Este resultado está avalado por la experiencia en el desarrollo de pruebas de conceptos, de rendimiento y un prototipo funcional basado en ICE. |
| Documentación disponible y al alcance del personal. | OPC | El puntaje que arrojó este criterio radica en que existe bastante documentación disponible para OPC y se puede comprobar en las siguientes direcciones: www.opcfoundation.org http://www.matrikonopc.com/downloads...sts/index.aspx |

| | | |
|---|---|---|
| | | http://www.matrikonopc.com/resources/opc-tutorials.aspx<br><br>Existen muchas otras. |
| Documentación disponible y al alcance del personal. | OPC UA | Este criterio obtiene un puntaje de 0 porque actualmente es propietaria, solo accesible por los miembros de OPC Foundation lo que significa que no tiene documentación disponible. |
| Documentación disponible y al alcance del personal. | Web Services | El puntaje de este criterio se debe a que existe mucha documentación sobre servicios web en toda la red. Sobre la herramienta gSOAP que permite implementar servicios web en C++ también podemos encontrar toda la documentación requerida. Para comprobar algunos link puede visitar:<br><br>http://www.w3.org/TR/wsdl<br><br>http://www.w3.org/TR/soap12-part1/<br><br>http://uddi.org/pubs/ProgrammersAPIv2.htm<br><br>http://www-106.ibm.com/developerworks/webservices/<br><br>http://www.w3.org/TR/ws-arch.<br><br>http://www.ws-i.org/<br><br>http://www.cs.fsu.edu/~engelen/soap.html |
| Documentación disponible y al alcance del personal. | ICE | El puntaje que arrojó este criterio para ICE tiene que ver con que ICE es una tecnología nueva pero muy usada y tiene bastante documentación disponible en su página |

| | | |
|---|---|---|
| | | principal. También existe un grupo de empresas y proyectos que usan ICE y ponen a disposición mucha documentación.<br><br>http://www.zeroc.com/ice.html<br><br>http://arco.esi.uclm.es/es<br><br>http://crysol.org/en/node/619<br><br>http://www.laboratoryformicroenterprise.org/iter/doc/icetests/html/index.html |
| Posibilidad de crear soluciones universales. | OPC | Aunque OPC es el estándar más utilizado en la comunicación con dispositivos de control y con sistemas externos no es del todo una solución universal porque 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. Obliga a los terceros a comunicarse usando un cliente OPC. Por todo lo anterior obtiene el puntaje especificado en la matriz. |
| Posibilidad de crear soluciones universales. | OPC UA | En este criterio OPC UA obtiene el máximo de puntaje porque esta tecnología unifica en una sola especificación todos los estándares OPC y ya no se basa en DCOM sino en una arquitectura SOA que puede ser implementada con servicios web. Define los wsdl de cada uno de los estándares lo que la convierte en una solución universal e independiente de la plataforma. También existen pasarelas entre OPC UA y OPC DA lo que eliminaría problemas |

| | | |
|---|---|---|
| | | de compatibilidad con las versiones anteriores. |
| Posibilidad de crear soluciones universales. | WS | Esta tecnología obtiene un máximo de puntaje en este aspecto. Los servicios web representan una solución universal porque resuelven los problemas de interoperabilidad encontrados en tecnologías como CORBA, DDS, etc. Permiten que aplicaciones que se encuentren en diferentes sistemas operativos e implementadas en diferentes lenguajes de programación puedan comunicarse usando el protocolo SOAP. La base de esta evaluación está en la documentación disponible en la web. |
| Posibilidad de crear soluciones universales. | ICE | El puntaje es bastante alto porque a pesar de representar una solución completamente universal se pueden adoptar soluciones que nos lleven a ese punto. Es multiplataforma lo que permite comunicar aplicaciones ICE en Windows con aplicaciones ICE en Linux y además permite la comunicación con distintos ORB(característica que no cumple ninguna otra tecnología de este tipo), brinda wrapper para varios lenguajes de programación lo que permite entregar wrapper de los lenguajes que hablan los clientes garantizando la abstracción de la tecnología utilizada. El puntaje otorgado se basa en la documentación existente sobre ICE y en la experiencia alcanzada en el desarrollo de las pruebas a la tecnología. |
| Desarrollo de aplicaciones | OPC | El puntaje que alcanza en este |

| | | |
|---|---|---|
| distribuidas. | | criterio está dado porque OPC es una tecnología creada para funcionar de forma distribuida, donde a un servidor se pueden conectar varios clientes y los mismos estar remotamente y desde cualquier escalón en la estructura de una red. La evaluación se basa en el driver OPC desarrollado en el SCADA y en la documentación existente en la web. |
| Desarrollo de aplicaciones distribuidas. | OPC UA | El puntaje que arrojó este criterio se debe a que OPC UA es una tecnología creada para funcionar de forma distribuida, donde a un servidor se pueden conectar varios clientes y los mismos estar remotamente y desde cualquier escalón en la estructura de una red y además puede existir comunicación ínter-empresas por su arquitectura SOA. |
| Desarrollo de aplicaciones distribuidas. | Web Services | Este puntaje significa que los servicios web garantizan la distribución de las aplicaciones en diferentes niveles de la red y si algunas de las aplicaciones falla no afecta al resto porque el centro de la comunicación son los servicios y de esta manera permite las comunicaciones distribuidas. Este puntaje se puede comprobar en el prototipo de comunicación con terceros. |
| Desarrollo de aplicaciones distribuidas. | ICE | En este punto ICE alcanza el puntaje más alto porque ICE es una tecnología Middleware y su razón de ser es la gestión de las comunicaciones distribuidas usando diferentes mecanismos de comunicación y garantizando la interoperabilidad entre aplicaciones, entre otros |

| | | |
|---|---|---|
| | | servicios interesantes. Esta puntuación se basa en la documentación disponible y en el documento de pruebas. |
| Consumo de recursos. | OPC | El puntaje que arrojó este criterio se puede comprobar en los resultados de las pruebas expuestas en el documento de Pruebas de OPC del SCADA. |
| Consumo de recursos. | OPC UA | Este criterio no es medible en el caso de OPC UA porque no tenemos acceso a la especificación por lo que no se pueden implementar los componentes necesarios para probar este aspecto. |
| Consumo de recursos. | Web Services | El puntaje que arrojó este criterio se puede comprobar en el documento de pruebas del Prototipo de Comm3. |
| Consumo de recursos. | ICE | Este puntaje se debe a que el consumo de recursos usando ICE es aceptable y se pueden comprobar los resultados en el documento de pruebas de comparación entre ICE y TAO. |
| Soporte por parte de los desarrolladores. | OPC | El puntaje que arrojó este criterio se debe a que existe soporte por parte de OPC Foundation. Existen estándares (especificaciones) liberadas y otras que aún son propietarias. |
| Soporte por parte de los desarrolladores. | OPC UA | El puntaje en este criterio es porque siempre y cuando seamos miembros de OPC Foundation vamos a tener el soporte garantizado. Es una especificación propietaria. |
| Soporte por parte de los desarrolladores. | Web Services | El puntaje en este criterio se define porque existe soporte por parte de las organizaciones OASIS y W3C que son los comités responsables de la arquitectura y reglamentación de |

| | | |
|---|---|---|
| | | los servicios web. |
| Soporte por parte de los desarrolladores. | ICE | El puntaje que arrojó este criterio se debe a que ICE es una tecnología nueva que está disponible bajo licencia GPL y licencia Comercial y su principal soporte es de los desarrolladores de ICE de la empresa ZeroC. |
| Transmisión de un elevado número de variables (1000, 5000 y 10000). | OPC | El puntaje que arrojó este criterio se debe a que la transferencia de datos en OPC es sobresaliente y se puede comprobar en los resultados de las pruebas expuestos en el documento: dir |
| Transmisión de un elevado número de variables (1000, 5000 y 10000). | OPC UA | Este criterio no se puede evaluar porque es imposible realizar las pruebas pertinentes cuando no contamos con la especificación de OPC UA. |
| Transmisión de un elevado número de variables (1000, 5000 y 10000). | Web Services | El puntaje que arrojó este criterio se debe a que la transferencia de datos con servicios web es aceptable si tenemos en cuenta que lo que se transmite por la red es un xml. Se puede comprobar en los resultados de las pruebas del prototipo de comunicación con terceros. |
| Transmisión de un elevado número de variables (1000, 5000 y 10000). | ICE | El puntaje que arrojó este criterio se debe a que el performance de ICE en cuanto a la transferencia de eventos es Excelente. Esta tecnología permite transferir hasta 60000 puntos del SCADA de forma remota y en 8 segundos usando pura tecnología sin optimizar los servicios y sin usar políticas de calidad de servicio. Los resultados se pueden obtener en el documento de pruebas de rendimiento. |

*Tabla 20: Evaluación de Criterios por Alternativas*

**Comparación de Opciones Vs Puntajes**

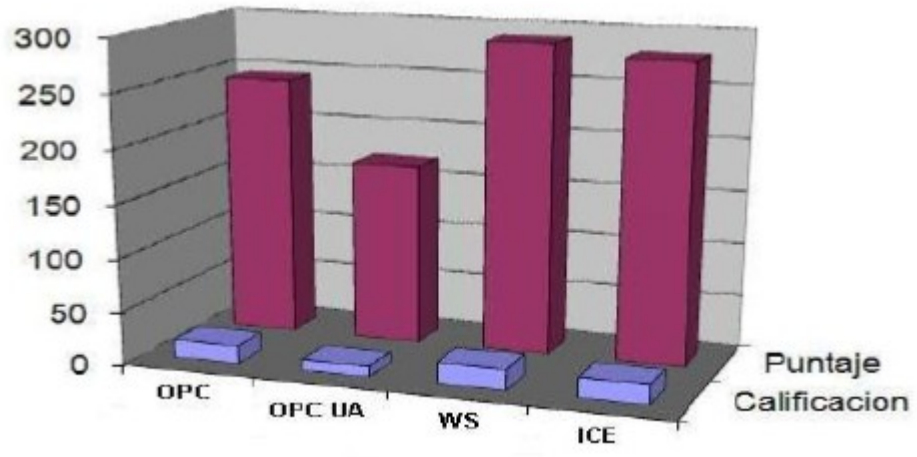


*Figura 39: Comparación de Opciones Vs Puntajes*

**Anexo II:** Descripción de los paquetes de diseño *Service Manager* y Services

**Paquete Services Manager**

Este paquete contiene las clases que definen la lógica para registrar y exponer las funcionalidades del SCADA como servicios Web, permitiendo a los terceros conocer los servicios expuestos y sus descripciones y al servidor de Comm3 registrar los servicios que implementa.

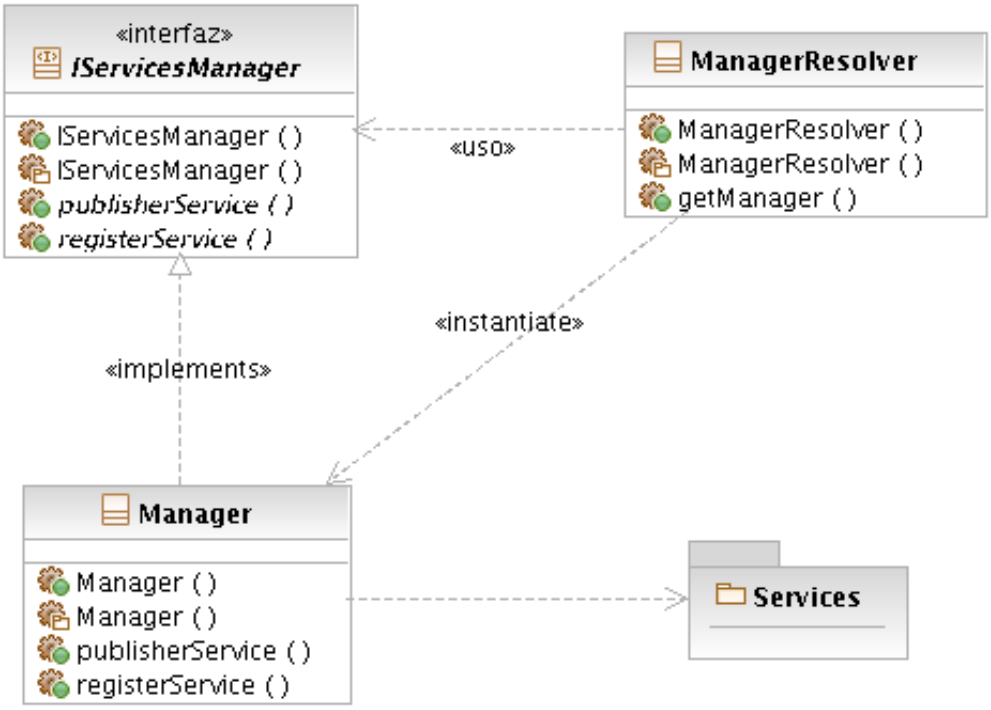


*Figura 40: Diagrama de Clases del Paquete Services Manager*

**Descripción de Clases y Métodos**

| ***Clase, ManagerResolver*** | |
|---|---|
| Esta clase juega el rol de una fábrica de manejadores de servicios (*Services Manager*). 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 logramos ejecutar los métodos de la implementación, en este caso, los métodos definidos en Manager. *Se basa en el patrón "factory method".* | |
| **Nombre del Método** | **Descripción** |
| *getManager():IServiceManager* | 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 217: Descripción de la clase ManegerResolver*

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

*Tabla 18: Descripción de la Interfaz IServiceManager*

| ***Clase, Manager*** | |
|---|---|
| Esta clase implementa la interfaz IServicesManager, define la lógica para publicar y registrar servicios. | |
| **Nombre del Método** | **Descripción** |
| *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 mismo método desde la internar 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 mismo método desde la interfaz IServicesManager. |

*Tabla 19: Descripción de la Interfaz Manager*

**Paquete Services**

Este paquete contiene las clases que definen la lógica de descripción de los servicios Web en C++, y sus operaciones.

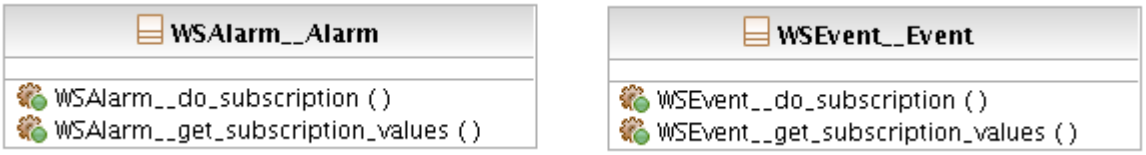


*Figura 41: Diagrama de Clases del Paquete Services*

**Descripción de Clases y Métodos**

| ***Clase, WSAlarm_Alarm*** | |
|---|---|
| Esta clase describe el servicio Web para acceso a las alarmas, y las operaciones soportadas por él. | |
| **Nombre del Método** | **Descripción** |
| *WSAlarm__do_subscription()* | En la descripción del servicio Web este es el método definido para invocar una suscripción a una lista de alarmas. |
| *WSAlarm__get_subscription_values()* | En la descripción del servicio este es el método definido para solicitar los valores de las alarmas ocurridas en el Guardián del ALBA. |

*Tabla 20: Descripción de la Clase WSAlarm__Alarm*

| ***Clase, WSEvent_Event*** | |
|---|---|
| Esta clase describe el servicio Web para acceso a los eventos, y las operaciones soportadas por él. | |
| **Nombre del Método** | **Descripción** |
| *WSEvent__do_subscription()* | En la descripción del servicio Web este es el método definido para invocar la suscripción a una lista de eventos. |
| *WSEvent__get_subcription_values()* | En la descripción del servicio este es el método definido para solicitar los valores de los eventos ocurridos en el Guardián del ALBA. |

*Tabla 21: Descripción de la Clase WSEvent__Event*

**Anexo III:** Utilización de las herramientas proporcionadas por gSOAP.

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

**wsdl2** es un paseador 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.

Opciones utilizadas con el compilador soapcpp2:
-C genera el código de la parte cliente
-S genera el código de la parte servidor
-L no genera soapClientLib/soapServerLib, los cuales están especificamente destinados a construir bibliotecas cliente/servidor de forma estática y dinámica.

Opciones utilizadas con el parseador wsdl2:
-s para deshabilitar la opción de importar por defecto el uso de vectores de la Standard Library.

Construcción de los esqueletos de los Servicios de Alarmas y Eventos:

- soapcpp2 -S -L WSAlarm.h
- soapcpp2 -S -L WSEvent.h

Con las líneas de comando anteriores se generan los ficheros wsdl que serán expuestos en una dirección HTTP para que los clientes (terceros) puedan consumir los servicios.

Construcción de los stub del cliente para los Servicios de Alarmas y Eventos:

- wsdl2 -s WSAlarm.wsdl
- wsdl2 -s WSEvent.wsdl

Posteriormente se generan los ficheros cabeceras y todo queda listo para implementar un cliente de los servicios.

- soapcpp2 -C -L WSAlarm.h
- soapcpp2 -C -L WSEvent.h

Si se desea desarrollar un cliente en otro sistema operativo o lenguaje de programación se pueden usar otras herramientas como Apache Axis y NetBeans IDE.

**Anexo IV:** Resultados de las pruebas del Servicio de Eventos

**CPR 3: Realizar subscripción a los eventos**

**Descripción**

El caso de prueba permite comprobar la realización exitosa de una suscripción a un grupo de eventos en el Servicio de Eventos de Comm3. Un cliente solicita la suscripción a una lista de eventos pasando por parámetros la lista de los identificadores, los que son almacenados en el servicio, y este proporciona al cliente un identificador de la suscripción realizada.

**Flujo central**

- Se obtiene la descripción () del Servicio de Eventos.
- Se crea un cliente de este servicio web.
- Se selecciona la opción subscribir a una lista de eventos.
- El cliente recibe el identificador de la suscripción realizada.

**Condiciones de Ejecución**

- Deben estar ejecutándose los servicios de Middleware, NameService (Servicio de Nombres), NotifyService (Servicio de Notificación), ChannelRegistry (Servicio para resgistro de Canales) y un cliente de Middleware para el envío de eventos.
- Se debe ejecutar el Servicio de Eventos apuntando al Middleware que está en ejecución y posteriormente el cliente de Comm3, ambos en diferentes máquinas para garantizar comunicaciones de forma remota en la ejecución de las pruebas.

**Resultados**

| Caso # | Clases Válidas | Clases Inválidas | Resultados Esperados | Resultados Obtenidos | Observaciones |
|---|---|---|---|---|---|
| 1 | Se registran los 10 eventos a las que el cliente desea suscribirse y posteriormente realiza la suscripción. | | 1. La aplicación debe mostrar una tabla en la interfaz preparada para recibir información de los 10 eventos solicitados.<br><br>2. Debe mostrar también un mensaje con el identificador de suscripción asignado por el servicio al cliente. | 1. Se observó que la interfaz quedó preparada para recibir los 10 eventos solicitados.<br><br>2. La aplicación mostró el identificador de la suscripción realizada. | |

*Tabla 22: Resultados de las pruebas para suscripción de eventos*

**CPR 4: Obtener valores de los eventos para una suscripción**

**Descripción**

El caso de prueba permite comprobar la realización exitosa durante la solicitud de los valores de eventos a partir de una suscripción en el Servicio de Eventos de Comm3. Un cliente solicita una lista de eventos, pasando por parámetros el identificador de la suscripción, y el servicio le devuelve los valores actualizados de los eventos siempre y cuando estas existan o hayan sucedido en el SCADA Guardián del ALBA.

**Flujo central**

- Se obtiene la descripción () del servicio de eventos.
- Se crea un cliente de este servicio web.
- Se selecciona la opción subscribir a una lista de eventos.
- El cliente recibe el identificador de la suscripción realizada.
- El cliente selecciona la opción Iniciar Lectura de eventos para la suscripción anterior y este termina recibiendo la información de los eventos solicitados.

**Condiciones de Ejecución**

- Deben estar ejecutándose los servicios de Middleware, NameService (Servicio de Nombres), NotifyService (Servicio de Notificación), ChannelRegistry (Servicio para registro de Canales) y un cliente de Middleware para el envío de eventos.
- Se debe ejecutar el Servicio de Eventos apuntando al Middleware en ejecución y posteriormente el cliente de Comm3, ambos en diferentes máquinas para garantizar comunicaciones de forma remota en la ejecución de las pruebas.

**Resultados**

| Caso # | Clases Válidas | Clases Inválidas | Resultados Esperados | Resultados Obtenidos | Observaciones |
|---|---|---|---|---|---|
| 1 | 1. Se registran los 10 eventos a los que el cliente desea suscribirse y se realiza la suscripción.<br><br>2. Posteriormente se inicia la lectura usando las opciones que brinda la interfaz. | | 1. La aplicación debe mostrar una tabla en la interfaz preparada para recibir información de los 10 eventos solicitados.<br><br>2. Debe mostrar también un mensaje con el identificador de suscripción asignado por el servicio al cliente.<br><br>3. La interfaz debe mostrar en la tabla los valores actualizados de los eventos solicitados. | 1. Se observó que la interfaz quedó preparada para recibir los 10 eventos.<br><br>2. La aplicación mostró el identificador de la suscripción realizada.<br><br>3. La aplicación mostró en una tabla los valores actualizados de los eventos pedidas en la suscripción. | |

*Tabla 23: Resultados de las pruebas para lectura de eventos por suscripción*

## 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.

**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.

**Comm3:** Susbistema de Comunicación con Terceros.

**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.

**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.

**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.

**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).

**.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.

**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 el 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.

**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.