Universidad de las Ciencias Informáticas

Título: Aplicación de estándares de calidad de servicios para el Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA.

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

Autora: Yerenia Breña Pérez
Tutor: Ing. José Antonio Aragón Cáceres
Co-Tutor: Ing. Maikel Pérez Javier

Ciudad de la Habana, Junio 2011
"Año 53 del Triunfo de la Revolución"

Declaración de autoría

Declaro ser la única autora del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma de la Autora

Yerenia Breña Perez

_____

Firma del Tutor

Ing. José Antonio Aragon Cáceres

_____

Firma del Co-Tutor

Ing. Maikel Pérez Javier

DATOS DE CONTACTO

Nombre y apellidos del tutor: José Antonio Aragón Cáceres.

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas.

e-mail: jaaragon@uci.cu

Ingeniero graduado en la Universidad de las Ciencias Informáticas (UCI), con 5 años de experiencia en la producción de software, específicamente en el desarrollo de sistemas de comunicación distribuida para sistemas SCADA. Líder de la línea de desarrollo de Comunicaciones del Departamento de Construcción de Componentes del CEDIN.

Nombre y apellidos del co-tutor: Maikel Pérez Javier

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Informática.

e-mail: mperezj@uci.cu

Profesor Instructor con tres años de experiencia docente en la Universidad de las Ciencias Informáticas (UCI) y 6 años de experiencia en la producción de software, específicamente en el desarrollo del *middleware* del Sistema Supervisor de Procesos Automatizados (SCADA). Jefe de departamento del Departamento de Construcción de Componentes del CEDIN.

AGRADECIMIENTOS

Agradezco a la Revolución por brindarme la oportunidad de realizar un sueño, a mi familia por apoyarme siempre, a mis amigos, a los que por algún motivo no pueden estar presentes y especialmente a Toni por estar siempre a mi lado en todo momento y por su ayuda incondicional, a todos mil gracias.

Resumen

El presente trabajo muestra el desarrollo de un paquete (QoS Calidad de Servicio) dentro del servidor de comunicación con terceros del sistema SCADA Guardián del ALBA, proporcionando una vía efectiva de lograr seguridad y confiabilidad en la comunicación que se establece entre el SCADA y sistemas externos de diversos tipos, desarrollados por terceros que pudieran necesitar de la información manejada por este, relacionada con variables, eventos, alarmas y ejecución de comandos.

De manera general se comenzó por la investigación sobre la calidad de servicio en servicios web y los diferentes estándares que garantizan la seguridad y la confiabilidad en las comunicaciones entre servicios web. Se adquirió una noción general acerca del estado del arte de los diferentes estándares de calidad, para poder llevar a cabo una investigación acertada de los mismos.

Se seleccionaron los estándares a aplicar en el servidor, SSL (*Secure Socket Layer*) para lograr la seguridad y *WS-ReliableMessaging* para la confiabilidad. Se definieron los requerimientos funcionales y no funcionales que poseería el servicio, seguido por el modelado de la solución utilizando el patrón de arquitectura en tres capas. Finalmente se implementó el diseño definido y se realizaron pruebas las cuales arrojaron resultados satisfactorios que validaron las tecnologías utilizadas y el modelado propuesto.

Palabras clave

Calidad de servicio, confiabilidad, seguridad, SCADA.

Abstract

This work shows the development of a package (QoS Quality of Service) within the communication server with third applications that belongs to SCADA ALBA's Guardian system, providing an effective way to achieve security and reliability for the communication between the SCADA and some other external systems of different types, developed by third companies who may need the information that is managed by SCADA; this information can be related to variables, events, alarms and commands execution.

In general, it was started with the research about the service quality in Web services and the different standards that guarantee the safety and reliability in communications between web services. It was acquired a general idea about the state of the art of the different quality standards, in order to achieve a successful investigation about them. There were selected the standards to be implemented on the server: SSL (Secure Socket Layer) for the assurance of the security, and WS-ReliableMessaging for the reliability. There were defined the functional and nonfunctional requirements that would own the service, and after that, it was modeled the solution by using the three-layer architecture standard. Lastly, the definite design was implemented and tested; as a consequence, satisfactory results were obtained and then the selected technologies and the proposed model were validated.

Key words

Quality of service, reliability, security, SCADA

Índice

Índice de figuras.

Índice de tablas.

Introducción

En la actualidad los servicios web ofrecen el principal mecanismo para el transporte de datos entre ordenadores, ellos abren la puerta a nuevas oportunidades de negocios al poder compartir servicios con un alto grado de integración, además promueven la interoperabilidad, debido a que la interacción entre un proveedor y un solicitante de servicio está diseñada para que sea completamente independiente de la plataforma y el lenguaje en el que han sido desarrollados. También permiten la integración, donde solicitantes, proveedores y agentes actúan en conjunto para crear sistemas que son auto-configurables, adaptativos y robustos.

Un servicio web es una aplicación de software identificada por un identificador único de recurso (URI por sus siglas en ingles), cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML (1). Este soporta interacciones directas con otros agentes de software utilizando mensajes XML intercambiados mediante protocolos basados en Internet.

El uso de estos servicios ofrece nuevas características en el campo de control y automatización de procesos, que anteriormente eran difíciles o costosos de implementar en el control tradicional de sistemas, tales como el acceso remoto a la planta, el intercambio de información y la estandarización de software.

Con la proliferación de los servicios web como solución para la integración de aplicaciones de software, la calidad de servicio para los servicios Web se está convirtiendo en un requisito fundamental a tener en cuenta para la comunicación entre proveedores y consumidores de servicios.

En un servicio web, la calidad de servicio (QoS Quality of Service) se refiere principalmente a la calidad, tanto en los aspectos funcionales y no funcionales de este. La aplicación correcta de calidad de servicio a los servicios web proporciona rendimiento, fiabilidad, escalabilidad, capacidad, robustez, exactitud, integridad, accesibilidad, disponibilidad, interoperabilidad y seguridad contribuyendo a desarrollar productos cada vez más elaborados y acabados.

Actualmente la calidad de servicio en servicios web se ven reflejadas en diversas áreas como en las telecomunicaciones, para medir o predecir la calidad de la voz lo más exacta y eficiente posible, en los sistemas de control y supervisión para mejorar los tiempos de entrega de los datos y asegurar los mismos.

El sistema SCADA Guardián del ALBA (GALBA) es un conjunto de aplicaciones de software diseñado para la supervisión y control de la producción petrolera, desarrollado por especialistas

venezolanos en conjunto con el Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas. Este sistema está conformado por diferentes módulos, cada uno de ellos con una función específica, que se comunican entre sí a través del módulo encargado del transporte de datos o middleware.

Para la gestión de las comunicaciones entre el GALBA y sistemas tanto de gestión, como otros SCADAs, se desarrolló el módulo o Subsistema de Comunicación con Terceros, utilizando como tecnología de comunicación distribuida los Servicios Web antes mencionados. Esta solución permite a aplicaciones externas a nuestro sistema, el intercambio de variables, alarmas, eventos e interacción a través de comandos.

En la actualidad en la interacción de los sistemas externos con el Subsistema de Comunicación con Terceros no se garantiza:

- La persistencia en los mensajes, el acuse de recibo, reenvío de mensajes, la eliminación de mensajes duplicados y el orden de entrega de estos.
- La seguridad ante cualquier tipo de ataque, pues la mensajería en la capa de transporte no garantiza la confiabilidad en la entrega de los mensajes intercambiados.

Por todo lo expuesto anteriormente se identificó el siguiente **problema a resolver**:

¿Cómo lograr la comunicación del Subsistema de Comunicación con Terceros del Guardián del ALBA con sistemas externos de forma segura y confiable?

Del problema anterior, se define que el **Objeto de Estudio** de este trabajo se corresponde a la Calidad de Servicio en Servicios Web y el **campo de acción**, es el Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA. De acuerdo con el problema planteado la **Idea a Defender** es la siguiente:

Si se aplican estándares adecuados de calidad de servicios a los servicios web del Subsistema de Comunicación con Terceros del SCADA Guardián del ALBA, entonces se garantizará un subsistema fiable y robusto, en el cual los datos que se intercambiarán con sistemas externos no serán alterados y las transacciones que sean truncadas por algún factor externo, no alterarán el estado del subsistema, ni de las operaciones que se encuentre realizando el tercero; posibilitará además un mayor rendimiento y menor latencia por envío de datos, ya que se optimizará la cantidad de información a enviar mejorando así el desempeño de los servicios.

El **Objetivo General** que se desea alcanzar con este trabajo es el de seleccionar, e implementar, estándares de calidad de servicios que permitan la comunicación del Subsistema de Comunicación con Terceros con sistemas externos de forma segura y confiable.

Definiéndose las siguientes **Tareas de la Investigación** para lograr un mayor nivel de precisión en la implementación:

1. Estudiar los temas relacionados con estándares de calidad de servicios en servicios web.
2. Analizar los estándares de calidad de servicio de servicios web en sistemas SCADA.
3. Analizar las especificaciones y tecnologías actuales relacionas con calidad de servicio para la selección adecuada de una o varias de estas y su aplicación en el GALBA.
4. Evaluar y seleccionar tecnologías y componentes libres adecuados para la aplicación de los estándares de calidad de servicios seleccionados en el GALBA.
5. Modelar a través del diseño las funcionalidades relacionadas con la aplicación de los estándares de calidad para los servicios web del Subsistema de Comunicación con Terceros.
6. Implementar las funcionalidades relacionadas con la aplicación de los estándares de calidad de servicio para los servicios web del Subsistema de Comunicación con Terceros.
7. Probar las funcionalidades relacionadas con la aplicación de los estándares de calidad de servicio para los servicios web del Subsistema de Comunicación con Terceros.

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

A nivel teórico

- **Análisis-síntesis:** Utilizado para analizar las teorías y documentos generados por desarrolladores que usen herramientas para el tratamiento de aplicaciones distribuidas permitiendo la extracción de los elementos que se relacionen con la aplicación de los estándares de calidad de servicios a los servicios web del Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA.

- **Análisis histórico-lógico:** Utilizado para conocer, con mayor profundidad los antecedentes y las tendencias actuales referidas al origen, desarrollo y aplicación de los estándares de calidad de servicio en los servicios web.

A nivel empírico

Experimentos: Empleado en la elaboración de prototipos funcionales, con el objetivo de comprobar la efectividad de la aplicación de estándares de calidad de servicios en el Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA.

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

En el Capítulo 1: "Fundamentación Teórica", se describe el estado del arte de los temas relacionados con la calidad de servicios en servicios web y los estándares o soluciones existentes que implementan o permiten su aplicación.

En el Capítulo 2: "Selección de Tecnologías y modelado de la solución", se seleccionan los estándares de calidad de servicios para su posterior aplicación en el Subsistema de Comunicación con Terceros así como todas las herramientas que se usarán para su desarrollo. También se modela la solución propuesta.

En el Capítulo 3: "Implementación y Prueba", se muestra las vistas de implementación e implantación del Subsistema de Comunicación con Terceros después de aplicadas los estándares de calidad de servicio, así como los casos de pruebas realizados a este.

Capítulo 1: Fundamentación teórica.

En el presente capítulo se exponen los conceptos fundamentales relacionados con los sistemas SCADA, una introducción al SCADA GALBA, así como los módulos que los componen, haciendo énfasis en el Subsistema de Comunicación con Terceros y de las tecnologías utilizadas en su implementación. Se aborda el tema de la calidad en los servicios web, las organizaciones que estandarizan la calidad de servicio en estos y se hace un estudio de principales estándares que aplican a las categorías de calidad de servicio como son: la seguridad, confiabilidad y las transacciones.

1.1 SCADA

El término SCADA proviene de las siglas de *Supervisory Control And Data Adquisition* es decir: Adquisición de Datos y Control Supervisor. Un SCADA es una aplicación o conjunto de aplicaciones de software y hardware especialmente diseñada para funcionar sobre ordenadores en el control de la producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, entre otros) 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, entre otros. (2)

1.1.1 SCADA Guardián del ALBA

El SCADA Guardián del ALBA es una solución de supervisión y control para los pueblos del ALBA, orientado inicialmente hacia la industria petrolera, pero con la visión de solventar las necesidades de supervisión y control de procesos de otras industrias para los países del ALBA y contribuir a la soberanía tecnológica de nuestros pueblos. La información viaja desde los manejadores que acceden a los dispositivos hasta los despliegues y reportes que visualiza el operador, pasando por los módulos de adquisición, middleware e históricos, los cuales acoplados al servidor de seguridad y configuración proveen mecanismos de autenticación y control de acceso y adaptabilidad a diversos entornos. El sistema brinda la información a otros sistemas mediante los mecanismos de OPC y Comunicación con Terceros y ofrece métodos

para la creación de algoritmos y tareas de control avanzado a través de la plataforma de Aplicaciones.

El sistema SCADA GALBA posee varios módulos, cada uno con funcionalidades específicas como lo muestra la siguiente figura:

Figura 1: Arquitectura distribuida del SCADA Guardián del ALBA.

Interfaz hombre maquina: Es el módulo que se encarga de la visualización de la información al operador de la planta. Mediante este se puede ejercer control sobre el proceso de producción. La interfaz de usuario brinda un conjunto de funcionalidades primarias, entre ellas la generación de reportes, impresión, análisis de variables, graficación de la tendencia de indicadores, configuración de los drivers para la comunicación y acceso a las alarmas entre otros.

Adquisición: Módulo encargado de manejar todo lo referente a la recepción, procesamiento y distribución de los datos proveniente del campo en tiempo real y de esta forma hace posible la ejecución de toda la lógica de negocio referente al GALBA. Permite la ejecución de acciones de adquisición de datos del nivel de recolección, conversión de unidades y linealización de datos recolectados y procesamiento de variables calculadas.

Configuración: Este módulo tiene como finalidad administrar toda información que sirve como base de configuración al resto de los módulos del GALBA, garantizando la persistencia de dicha información.

Seguridad: Provee las funcionalidades necesarias para garantizar el trabajo autorizado por usuarios y módulos, además brinda las herramientas necesarias para la protección contra ataques maliciosos o involuntarios al sistema por parte de personas o recursos tales como fallas de corriente, problemas de red o servidores, entre otros.

Middleware: Es la capa de software, que se encarga de la comunicación entre los diferentes módulos que forman parte del sistema. Proporciona los mecanismos para el envío de puntos, alarmas, comandos, lotes de puntos, lotes de alarmas, eventos y bitácoras, así mismo, también es responsable de proveer una interfaz que permita a los módulos del sistema ejecutar funcionalidades presentes en otros módulos de manera sincrónica, mediante llamadas a procedimientos remotos.

Base de datos Histórica: Módulo encargado de almacenar la información del sistema con el objetivo de que esta pueda ser empleada luego en la generación de reportes, tendencias, gestión de producción. Contiene la información persistente de la recolección de datos de los dispositivos.

1.1.2 Subsistema de Comunicación con Terceros del GALBA

En términos de este documento la comunicación con terceros se refiere a la comunicación que se establece entre sistemas externos y un sistema SCADA con el objetivo de intercambiar información. El objetivo fundamental de este subsistema es ofrecer una solución mediante la cual los sistemas externos pueden acceder a los datos manejados por el sistema SCADA Guardián del ALBA e interactuar con él.

Estos terceros pueden ser otros sistemas SCADA, Sistemas de Control Distribuido (DCS por sus siglas en inglés), aplicaciones de gestión, gerenciales y de negocio. En la actualidad dentro de los terceros más comunes se encuentran los Sistemas de Ejecución de Manufactura (en inglés MES, *Manufacturing Execution Systems*), y los Sistemas de Planificación de Recursos de la Empresa (en inglés ERP, *Enterprise Resource Planning*). (3)

Para la implementación de este subsistema se tuvieron en cuenta varias tecnologías de las cuales se seleccionó la tecnología de Servicios Web, que constituyen una forma estándar de integrar aplicaciones utilizando estándares abiertos; esta tecnología permite además una gran interoperabilidad, ya que es independiente de la plataforma.

Dentro de los principales servicios con que cuenta el Subsistema de Comunicación con Terceros se encuentran:

- **Acceso a variables:** Este servicio se encarga de garantizar a sistemas externos el acceso a las variables que se manipulan en el SCADA GALBA. Estas variables pueden ser de tipos simples o de tipos complejos. La información de dichas variables permite conocer el estado del sistema y su historia, de ahí la importancia de que sistemas de gestión o sistemas gerenciales puedan acceder a dicha información.
- **Acceso a alarmas y eventos:** Este servicio se encarga de garantizar el flujo de eventos y alarmas a través de todas las capas de la pirámide de control y supervisión, permitiendo tomar decisiones correctivas del estado del sistema.
- **Interacción a través de comandos:** Este servicio se encarga de definir la forma en que los clientes externos solicitan servicios de ejecución de comandos de manera segura a los servidores disponibles. Usualmente se utiliza este servicio para la escritura de variables ya que esta interacción debe contar con una mayor seguridad.

1.2 Servicios Web. Tecnologías básicas que lo componen.

Según el consorcio W3C los servicios web son aplicaciones de software identificada por un URI (Identificador único de recurso por sus siglas en español), cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML. (4) Un servicio Web soporta interacciones directas con otros agentes software utilizando mensajes XML intercambiados mediante protocolos basados en Internet. Predominantemente utiliza XML para etiquetar los datos, SOAP para transferir los datos, WSDL para describir los servicios disponibles y UDDI para listar que servicios están disponibles.

1.2.1 XML (eXtensible Markup Language)

Es un metalenguaje basado en marcas y etiquetas desarrollado por el World Wide Web Consortium (W3C) que se utiliza para crear otros lenguajes. Puede también definirse como un lenguaje de etiquetado extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. (5)

Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas y es muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

1.2.2 SOAP (Simple Object Access Protocol)

Es un protocolo de alto nivel de estandarización que permite el intercambio de mensajes entre aplicaciones, por lo que su función básica es definir un formato de mensajes estándar (basado en XML) que encapsula la comunicación entre aplicaciones. Los mensajes SOAP son independientes de cualquier sistema operativo o protocolo y puede ser transportado utilizando una variedad de protocolos de Internet.

Un mensaje SOAP se compone de un sobre (*envelope)* que contiene el cuerpo (*body*) del mensaje y cualquier información de cabecera (*header)* que se utiliza para describir el mensaje.

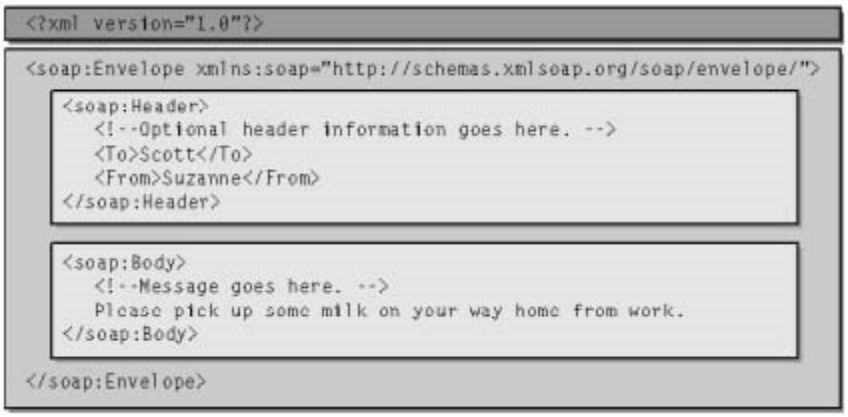

Figura 2: Estructura de un mensaje SOAP.

El elemento raíz del documento es el elemento ***Envelope***. La figura 2 contiene dos subelementos, ***Header y Body***. Un ejemplo de SOAP válido también puede contener otros elementos hijo en el sobre.

El elemento *envelope* puede contener un elemento *Header* (cabecera) opcional que contiene información sobre el mensaje. En la figura 2, la cabecera contiene dos elementos que describen a quien compuso el mensaje, y posible receptor del mismo.

El elemento *envelope* debe contener un elemento *Body* (cuerpo), el cual contiene la carga de datos del mensaje.

1.2.3 WSDL (Web Services Description Languages)

Es un estándar de descripción de servicios web, utiliza un documento con formato XML que detalla la forma en la cual los clientes externos pueden interactuar con el servicio web existente, los métodos que soportan y la sintaxis de los protocolos de comunicación (HTTP, SOAP) que utiliza. Un documento WSDL contiene información acerca de la interfaz, la semántica y los aspectos administrativos involucrados en una solicitud realizada a un servicio web.

1.2.4 UDDI (Universal Description Discovery and Integration)

Es un directorio distribuido y basado en Web que permite listar, buscar, describir y descubrir servicios web. Se asemeja en su funcionalidad a las páginas amarillas de un directorio telefónico. UDDI define estructuras de datos, permite a los desarrolladores encontrar información para escribir los clientes de los servicios, y posibilita a estos últimos preguntar al registro y obtener las referencias a los servicios de interés.

1.3 Calidad de Servicio

Es el grado en el que un servicio satisface las necesidades o requerimientos del consumidor, y en lo posible excederlos, lo que implica hacer las cosas necesarias bien y a la primera, con actitud positiva y espíritu de servicio.

1.3.1 Calidad de Servicio en Servicios Web

La calidad de servicio o ***QoS***, es una combinación de varias cualidades o propiedades de un servicios (6). Se trata de un conjunto de atributos no funcionales que pueden influir en la calidad del servicio prestado por un servicio Web.

Algunos ejemplos de los atributos de calidad de servicio son:

Disponibilidad: Es la probabilidad de que el sistema está en marcha y puede responder a los consumidores sus peticiones. Por lo general, es un poco paralelo a la fiabilidad y ligeramente frente a la capacidad.

Capacidad: Es el límite de solicitudes simultáneas que un servicio puede manejar. Cuando el número de solicitudes simultáneas excede la capacidad de un servicio, su disponibilidad y la fiabilidad disminuyen.

Fiabilidad: Es la capacidad de un servicio para realizar sus funciones requeridas.

Rendimiento: Es la medida de la velocidad para completar una solicitud de servicio. Es medido por la latencia (el tiempo transcurrido entre la llegada y la terminación de un servicio, petición del cliente), rendimiento (el número de solicitudes completadas en un período de tiempo) y tiempo de respuesta (el retraso de la solicitud de obtener una respuesta desde el servicio).

Costo: Es la medida del costo de solicitar un servicio. Se puede cobrar por el número de solicitudes de servicio, o podría ser una tarifa plana, cargos por un período de tiempo. De

manera que los requisitos de calidad de servicio para los servicios Web son tan importantes para ambos servicios (proveedores y consumidores).

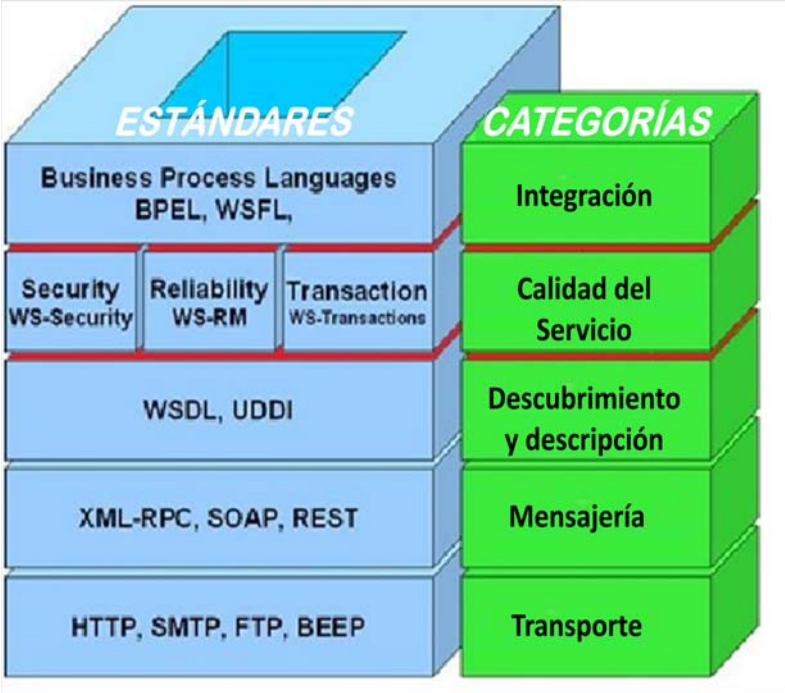

Figura 3: Estándares en Servicios Web.

1.3.2 Organizaciones que estandarizan las QoS

Cuando se habla de calidad del servicio en los servicios web se deben mencionar varias organizaciones que están involucradas en tan importante tarea como son: la W3C, OASIS, y WS-I. Estas organizaciones son las responsables de la arquitectura y reglamentación de todas las especificaciones y recomendaciones.

1.3.2.1 W3C

El Consorcio *World Wide Web* (W3C) se fundó en Octubre de 1994 para llevar a la web a su máximo potencial mediante el desarrollo de protocolos comunes que fomenten su evolución y aseguren su interoperabilidad. (7)

La W3C está compuesta por más de 400 organizaciones en todo el mundo y han contribuido con su esfuerzo a hacer de la Web una infraestructura robusta, escalable y adaptativa para el mundo de la información y ha logrado realizar más de medio centenar de especificaciones

técnicas orientadas a la infraestructura Web, además de proporcionar retroalimentación para el cliente y de contribuir en la estandarización de las tecnologías.

1.3.2.2 OASIS

OASIS (*Organization for the Advancement of Structured Information Standards*), fundada en 1993, es un consorcio global para el desarrollo, convergencia y uso de estándares en el comercio electrónico. Los miembros mismos establecen la agenda técnica usando un proceso abierto y ligero para las transacciones de negocio, publicaciones electrónicas, mapas temáticos e interoperabilidad dentro y entre mercados (7). Esta organización presenta una colección de referencias en líneas para estándares de lenguajes de marca interoperables.

1.3.2.3 WS-I

WS-I (*Web Services Interoperability*) es una organización para lograr la interoperabilidad de los servicios web sin importar la diferencia de plataformas, lenguaje de programación y aplicaciones. Su objetivo es dar respuesta a las necesidades de los clientes. Esta organización desarrolla herramientas, recursos y otras guías de apoyo para ayudar en la implementación de los servicios web. Esta comunidad se divide en grupos de trabajo donde cada grupo trabaja en un conjunto específico de entregables y está asociada a otras organizaciones como W3C y el IETF (*Internet Engineering Task Force*), que complementan su actual trabajo. (7)

1.3.3 Confiabilidad en servicios web

Cuando se trata sobre confiabilidad en servicios web, existen algunos aspectos a tener en cuenta que son fundamentales:

Garantía de entrega: Garantizar que toda la información que se enviará sea realmente recibida por el destinatario o informar de que ocurrió un error.

Eliminación de mensajes duplicados: garantizar que toda la información duplicada puede ser detectada y filtrada.

Orden de entrega de mensajes: La comunicación entre las partes se compone de varios intercambios de mensajes individuales. Este aspecto asegura que el intercambio de mensajes entre la aplicación receptora y el emisor se realice de manera ordenada.

Tolerancia a fallos: Asegura que toda la información prescrita por el protocolo está siempre disponible, independientemente de un posible fallo de la red o falla física del equipo de computo.

Sincronización de Estado: Si el intercambio de mensajes se cancela por cualquier razón, entonces es conveniente para ambos nodos establecer su estado como si no hubiera comunicación entre las partes.

Es común que las interacciones basadas en mensajes entre los servicios Web requieran confiabilidad, sean capaz de garantizar la mensajería incluso en el caso de la red, las aplicaciones o los fallos de componentes, y que incluyan mecanismos de persistencia y vuelvan a enviar la semántica.

Como es común en el mundo de los servicios web existe un espacio de normas que vienen a solucionar el problema:

- WS-Reliability
- WS ReliableMessaging

Estas dos normas son la competencia en cuanto a normas de confiabilidad en servicios web ya que están respaldadas por diferentes grupos de empresas encargadas de estandarizar la web. La confiabilidad en los servicios web se encuentra en la capa de comunicación en la pila de protocolos de los servicios web. La implementación de los estándares *WS-ReliableMessaging y WS-Reliability* garantizan de una forma u otra la confiabilidad en la entrega de los mensajes intercambiados en los servicios web.

1.3.3.1 WS-ReliableMessaging

A menudo es un requisito para dos servicios web que desean comunicarse, hacerlo de forma fiable en la presencia de componentes de software, sistema o fallos en la red. El objetivo principal de la especificación *(WS-ReliableMessaging)* es crear un mecanismo modular para la entrega fiable de mensajes. Esta especificación define un protocolo de mensajería para

identificar, rastrear y gestionar la entrega confiable de mensajes entre exactamente dos partes, una fuente y un destino. También define un enlace SOAP que es necesario para la interoperabilidad y permite además que enlaces adicionales puedan ser definidos, como la seguridad, entre otros.

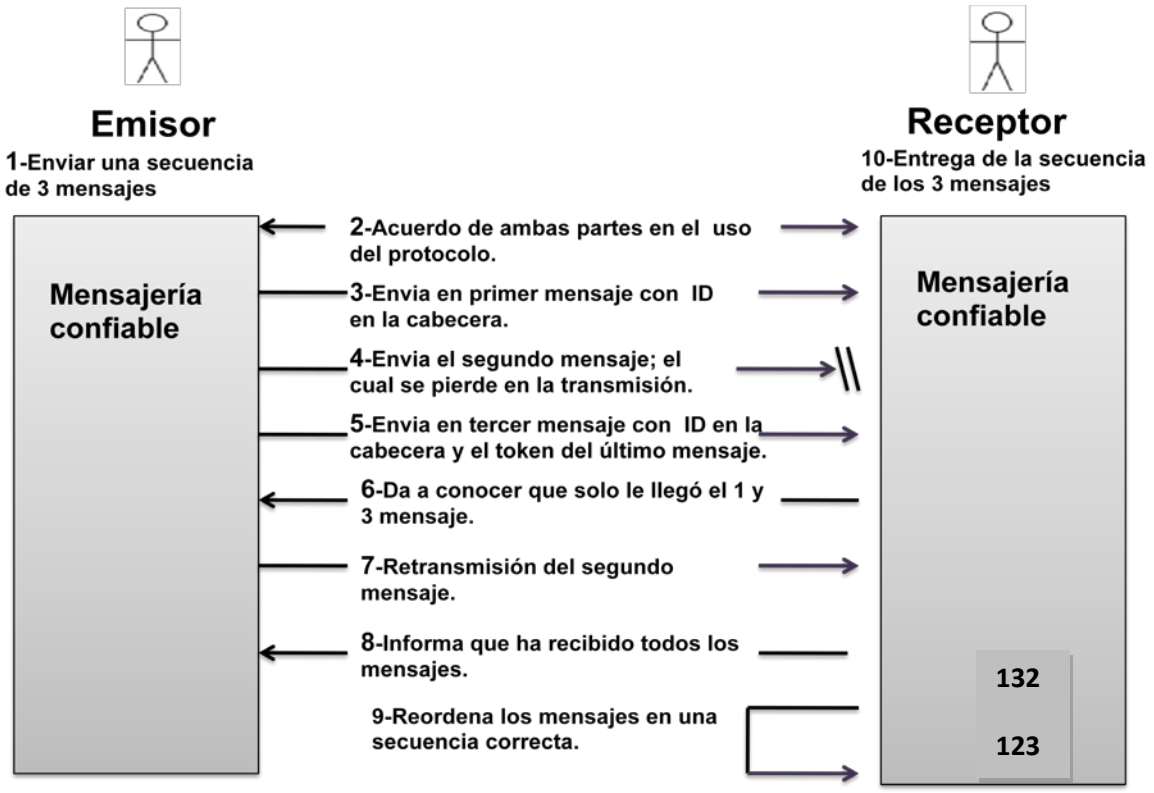

Figura 4 : Descripción de la mensajería confiable paso a paso.

1.3.3.2 WS-Reliability.

WS-Reliability es una especificación basada en SOAP que cumple con los requisitos de mensajería confiable crítico para algunas aplicaciones de Servicios Web. SOAP sobre HTTP no es suficiente como protocolo de mensajería de nivel de aplicación, pues se debe garantizar un cierto nivel de fiabilidad y seguridad. Esta especificación define la fiabilidad en el contexto de los actuales estándares de servicios Web y ha sido diseñada para su uso en combinación con otros protocolos complementarios y se basa en experiencias previas.

1.3.4 Seguridad en Servicios Web

La seguridad es un concepto de los más significativos entre aquellos servicios que comprenden la Calidad del Servicio (QoS) de los servicios Web. Según la ISO 7498-2(estándar que define un servicio de seguridad como el servicio proporcionado por un nivel de un sistema abierto que garantiza la seguridad de los sistemas abiertos o a las transferencias de datos en dichos sistemas), lo elemental que debe cumplir un servicio web es la confidencialidad, integridad, autenticidad de origen, no repudio y control de acceso (8). Estos son elementos que se deben de tener en cuenta para seleccionar un estándar de calidad referido a la seguridad de un servicio web.

Existen un conjunto de estándares a nuestro alcance que nos permiten garantizar los servicios de seguridad básicos y promueven la interoperabilidad de las soluciones facilitando y acelerando de este modo el acceso al mercado de los servicios Web como son, Secure Socket Layer, WS-Security, las especificaciones de seguridad de las tecnologías XML (XML Digital Signature, XML Encryption, XML Key Management System) sobre las cuales se profundizará a continuación.

1.3.4.1 SSL

Secure Socket Layer es un sistema de protocolos diseñado en 1994 por la empresa Nestcape Communcations Corporation, y está basado en la aplicación conjunta de Criptografía Simétrica, Criptografía Asimétrica (de llave pública), certificados digitales y firmas digitales para conseguir un canal o medio seguro de comunicación a través de Internet y proporciona servicios de seguridad a la pila de protocolos, encriptando los datos salientes de la capa de Aplicación antes de que estos sean segmentados en la capa de Transporte y encapsulados y enviados por las capas inferiores.(10)

SSL implementa un protocolo de negociación para establecer una comunicación segura a nivel de socked (nombre de máquina más puerto), de forma transparente al usuario y a las aplicaciones que lo usan. (10)

El protocolo SSL se encuentra en la pila OSI entre los niveles de TCP/IP y de los protocolos HTTP, FTP, SMTP, entre otros. Proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico, típicamente el RC4 o IDEA, y cifrando la clave de sesión de RC4 o IDEA mediante un algoritmo de cifrado

de clave pública, típicamente el RSA. (13) La clave de sesión es la que se utiliza para cifrar los datos que vienen del y van al servidor seguro. Se genera una clave de sesión distinta para cada transacción, lo cual permite que aunque sea reventada por un atacante en una transacción dada, no sirva para descifrar futuras transacciones. Proporciona cifrado de datos, autenticación de servidores, integridad de mensajes y, opcionalmente, autenticación de cliente para conexiones TCP/IP.

Cuando el cliente pide al servidor seguro una comunicación segura, el servidor abre un puerto cifrado, gestionado por un software llamado Protocolo SSL Record, situado encima de TCP. Será el software de alto nivel, Protocolo SSL Handshake, quien utilice el Protocolo SSL Record y el puerto abierto para comunicarse de forma segura con el cliente. (13)

1.3.4.2 WS-Security

La especificación de Seguridad de Servicios Web (*WS-Security*) fue publicada por OASIS, se describe como la estrategia de *Microsoft* para hacer frente a la seguridad dentro de un entorno de servicios Web. Se define un modelo integral de servicios de seguridad Web que soporta, integra y unifica varios modelos de seguridad popular, mecanismos y tecnologías (incluyendo las tecnologías clave simétrica y pública) de una manera que permite una variedad de sistemas para interactuar de forma segura en una plataforma y forma independiente del idioma. Características y extensión flexible para SOAP para aplicar seguridad a los servicios web. Esta especificación proporciona formatos de firma y cifrado de múltiples tecnologías y provee tres mecanismos principales: la capacidad de enviar señales de seguridad como parte de un mensaje, la integridad del mensaje y la confidencialidad del mensaje y proporcionando mejoras de estos así como la autenticación de mensajes individuales. WS-Security también proporciona un uso general, pero extensible, mecanismo para la asociación de tokens de seguridad con los mensajes, donde un token de seguridad (también token de autenticación o token criptográfico) es un dispositivo electrónico que se le da a un usuario autorizado de un servicio computarizado para facilitar el proceso de autenticación. (9)

1.3.4.3 XML Digital Signature

XML *Digital Signature* es una recomendación W3C desde el 2002 que define cómo aplicar firmas digitales sobre contenido XML y cómo representar, definiendo en un esquema XML, dicha información. Las firmas digitales son un mecanismo que garantiza la integridad de la

información y se muestran como una evidencia no repudiable del generador de la misma. En XML *Digital Signature* una firma digital puede ser aplicada a cualquier contenido digital incluyendo XML. Existen tres tipos de firmas: envolventes, envueltas y hermanas. Las dos primeras se aplican sobre información contenida dentro del mismo documento XML que transporta la firma digital. En las firmas hermanas el objeto de datos firmado y el elemento que representa la firma son hijos de un mismo padre de forma que no poseen relaciones padre-hijo entre sí. (7)

1.3.4.4 XML Encryption

W3C XML Encryption es una propuesta de recomendación desde el año 2002 y además de ser independiente de la tecnología, ofrece un modelo de procesamiento para cifrar, descifrar y representar los siguientes tipos de información:

- Documentos XML completos.
- Elementos XML únicos (y todos sus descendientes) dentro de un documento XML.
- El contenido de un elemento XML (algunos o todos los nodos hijos incluyendo todos sus descendientes) localizado dentro de un documento XML.
- Contenidos binarios arbitrarios ubicados fuera de un documento XML.

XML Encryption describe la estructura y la sintaxis de elementos XML para presentar información cifrada, resuelve los problemas de la confidencialidad de los mensajes SOAP en el intercambio entre los servicios Web por medios de la encriptación, este además especifica los pasos que se deben seguir a la hora de cifrar y descifrar documentos XML (o partes de ellos).
WS-Security usa el XML Encryption para que el contenido del mensaje se mantenga secreto para todos, con excepción del destinatario pretendido, generalmente a través de claves públicas encapsuladas en certificados digitales.

1.3.4.5 XML Key Management

XML Key Management System es una especificación sometida al proceso de estandarización del W3C que propone un formato de información así como los protocolos necesarios para convertir una infraestructura de Clave Pública (PKI) en un servicio Web de forma que se pueda:

- Registrar pares de clave privada/pública.
- Localizar claves públicas.
- Validar la clave.
- Revocación de clave.

- Recuperación de clave.

De esta forma toda la infraestructura PKI se extiende al mundo XML y permite delegar las decisiones de confianza a sistemas especializados simplificando el desarrollo de las aplicaciones cliente.

1.3.4.6 SAML (Security Assertion Markup Language)

SAML, especificación liberada por OASIS, es un marco de trabajo XML que permite el intercambio de información de seguridad entre servicios Web. Esta información de seguridad se materializa en forma de afirmaciones hechas por una autoridad SAML sobre un sujeto. Las afirmaciones SAML pueden contener tres tipos de información: autenticación, atributo, y decisión de autorización.

SAML define un protocolo petición/respuesta utilizado entre los sujetos y las autoridades SAML, el cual presenta características de flexibilidad, extensibilidad y además está diseñado para ser utilizado por otros estándares.

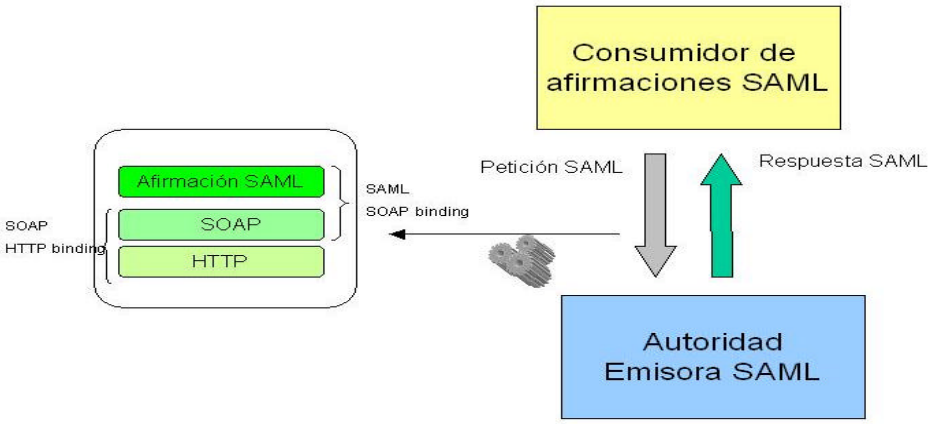

Figura 5: Funcionamiento de SAML.

1.3.4.7 XACML (eXtensible Access Control Markup Language)

XACML (estándar del consorcio OASIS) es un lenguaje, basado en XML, para definir políticas de control de acceso. Es decir, proporciona una sintaxis para gestionar el acceso a los recursos por parte de ciertos sujetos que desean realizar ciertas operaciones sobre ellos.

1.3.5 Transacciones

WS-Transaction es parte de una serie de especificaciones de un grupo industrial que incluye *a IBM, Microsoft y BEA Systems* la cual fue publicada en el 2002*.* La interfaz de *WS-Transaction* define lo que constituye una transacción y lo que va a determinar cuándo se ha completado con éxito. Cada transacción es parte de un conjunto global de actividades que constituyen un proceso de negocio que se lleva a cabo mediante la cooperación de servicios Web . (7)

El proceso de negocio en general se describe formalmente mediante el *Business Process Execution Language (BPEL). WS-Coordination* es una especificación complementaria que define el contexto y exactamente cómo se intercambia información durante el proceso de negocio.

Como tal, *WS-Transaction* especifica un protocolo estándar para la ejecución de transacciones en entornos de servicios Web:

- Transacciones atómicas o de corta duración en las que se conserva la idea de propiedades **ACID** (atomicidad, coherencia, aislamiento y durabilidad) en un sentido estricto, es decir, todas las operaciones dentro de la transacción se ejecutan con éxito o no se ejecuta ninguna (idea de protocolo en dos fases)
- Actividades de negocio o transacciones de larga duración en las que se flexibiliza las propiedades **ACID** de las transacciones, se invocan todas las operaciones a los participantes, los cuales tras su ejecución, informan al coordinador si han sido completadas con éxito. Si alguna falla, es necesario la ejecución de operaciones de compensación.

Capítulo 2: Selección de tecnología y modelado de la solución.

En el presente capítulo se exponen los requerimientos a implementar que permitirán la aplicación de estándares de calidad de servicio; también se analizan los estándares de calidad de servicio abordados previamente y se realiza la selección de los mismos para su uso en el Servidor de Comunicación con Terceros. Se selecciona además la metodología a utilizar, los lenguajes, tanto de programación como de modelado, y las herramientas que se emplearán en el desarrollo. Se muestra además, la arquitectura a utilizar, así como las descripciones de los paquetes del diseño arquitectónicamente significativos, incluyendo la elaboración de diagramas de clases del diseño y diagramas de secuencia con sus respectivas descripciones.

2.1. Requerimientos de software del Servidor de Comunicación con Terceros.

Los requisitos se clasifican en funcionales y no funcionales. Los funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que hacen al producto atractivo, usable, rápido y confiable.

A continuación se enuncian los requisitos con que contará el Subsistema de Comunicación con Terceros con la aplicación de estándares de calidad de servicio. Para su obtención se tuvieron en cuenta diversos criterios de personal especializado en el trabajo con sistemas SCADA, se tuvieron en cuenta además los criterios de los arquitectos y desarrolladores que trabajan en la línea de desarrollo de comunicaciones del Departamento de Construcción de Componentes del CEDIN. Además se estudiaron las aplicaciones de software privativo utilizadas para la gestión de los datos provenientes de sistemas SCADA.

2.1.1. Requisitos Funcionales.

A continuación se exponen la funcionalidad (RF) identificada en el proceso de adición de elementos de seguridad y confiabilidad al Subsistema de Comunicación con Terceros.

RF 1. El subsistema de Comunicación con Terceros deberá de permitir al externo o tercero al SCADA Guardián del ALBA un mecanismo para autenticarse con el servidor de Seguridad. (Si estos usuarios existen en la base de datos de seguridad)

2.1.2. Requisitos No Funcionales.

A continuación se exponen las características (RNF) referentes a estándares de calidad de servicio identificados en el proceso de conformación del Subsistema de Comunicación con Terceros del Guardián del ALBA:

RNF 1. Software

RNF 1.1 La implementación asociada a la aplicación de los estándares de calidad de servicios en el Subsistema de Comunicación con Terceros se debe desarrollar sobre el Sistema operativo GNU/Linux, distribución Debian, Kernel 2.6

RNF 2. Diseño e implementación

RNF 2.1 Para la aplicación de los estándares de calidad de servicios de la Comunicación con Terceros se debe utilizar el lenguaje de programación C++.

RNF 2.2 Para la aplicación de los estándares de calidad de servicios en el Subsistema de Comunicación con Terceros se debe utilizar como paradigma de programación, la Programación Orientada a Objetos.

RNF 3. Portabilidad

RNF 3.1 La aplicación de los estándares de calidad de servicios en el Subsistema de Comunicación con Terceros debe garantizar que los terceros puedan seguir accediendo a él sin importar la plataforma que estos utilicen.

RNF 4 Seguridad

RNF 4.1 En la comunicación entre el Subsistema de Comunicación con Terceros y sistemas externos se debe garantizar la confidencialidad, integridad, autenticidad de origen, no repudio y control de acceso.

RNF 5 Fiabilidad

RNF 5.1 El Subsistema de Comunicación con Terceros debe garantizar una comunicación confiable ante la presencia de componentes de software, sistemas o fallos en la red, para evitar las pérdidas de información en la comunicación con sistemas externos.

RNF 6 Rendimiento

RNF 6.1 El Subsistema de Comunicación con Terceros debe garantizar una alta velocidad de transmisión de datos para su utilización en aplicaciones de visualización y otras aplicaciones externas.

2.2. Selección de los estándares de calidad de servicio a utilizar en el Subsistema de Comunicación con Terceros.

En este punto de la investigación se hace necesario seleccionar los estándares de calidad de servicio que serán implementados en el Servidor de Comunicación con Terceros.

2.2.1. Estándar de seguridad a utilizar.

WS-Security proporciona un conjunto amplio de dispositivos de seguridad para aplicaciones de servicios Web, al basarse en estándares establecidos de la industria respecto a criptografía y cifrado y firmado de XML. Con el soporte generalizado, los estándares de seguridad en varias plataformas e infraestructuras de servicios Web, existe una buena interoperabilidad con la tendencia a mejorar en el tiempo. (11) A pesar de los beneficios, WS-Security también tiene algunas desventajas pues la configuración de WS-Security puede ser compleja y añade mucho volumen a los mensajes que se intercambian.

Para ver cómo podría impactar la aplicación del estándar WS-Security en el Subsistema de gComunicación con Terceros se realizaron pruebas de rendimiento a la extensión wsse (WS-Security) de gSOAP.

Los resultados de la prueba que se muestran en la figura 6 se basan en el envío y recepción de 100 y 1000 variables respectivamente entre un cliente y servidor. Cada secuencia de solicitudes se ejecutó varias veces con diferentes opciones de seguridad y sólo se muestra el mejor resultado arrojado en cada opción.

Las pruebas se ejecutaron en un sistema GNU/Linux Debian Squeeze de 32 bits® con procesador Intel Core 2 Duo E4500 2.20 GHz y 1GB de RAM, el cliente y el servidor operaron en máquinas diferentes, además se utilizó la versión 2.8 de la herramienta gSOAP para el desarrollo del servicio web de prueba.

Se ejecutaron las pruebas con cada una de las siguientes opciones de seguridad:

- texto plano : sin seguridad

- ssl : se usó HTTPS para conectar al servidor
- firma : firma de WS-Security
- cifrado : cifrado WS-Security
- firma-cifrado: firma y cifrado de WS-Security**.**

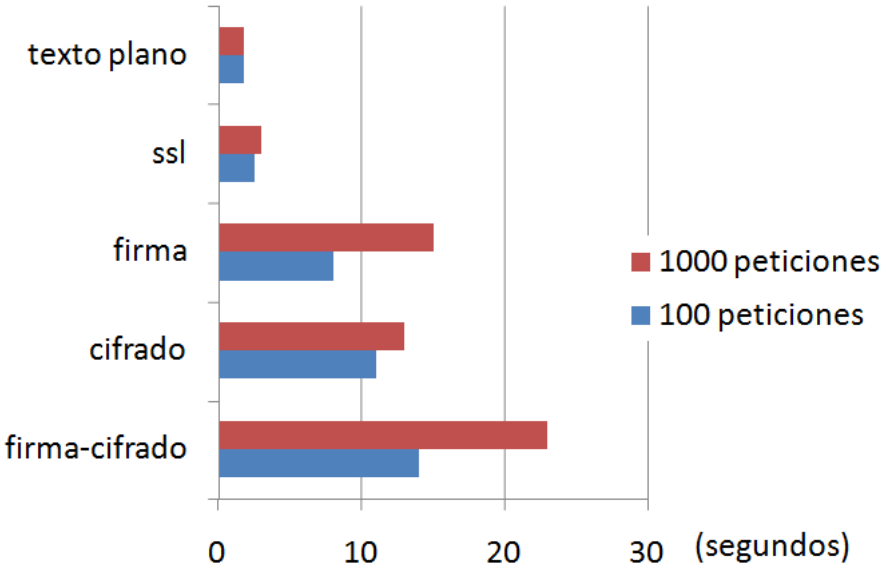

Figura 6: Pruebas de rendimiento a la implementación de gSOAP del estándar wsse.

Como se puede observar en la gráfica, el cifrado con SSL proporciona casi el mismo rendimiento que se obtiene sin protección, aunque su rendimiento es mejor en los mensajes mayores que en los menores. Por otro lado, el uso de WS-Security produce caídas significativas de rendimiento. En el caso de la combinación de firma y cifrado, el tiempo de prueba es más que 2.100% más largo que el tiempo sin protección.

La caída de rendimiento de WS-Security en combinación con XML Signature y del XML Encryption se debe a dos aspectos fundamentales:

- El trabajo intenso, de generar resúmenes y cifrar los datos. Esa parte del trabajo es necesaria independientemente del método de implementación que se use.

- WS-Security además añade mucho volumen a las cabeceras de mensajes SOAP. Ese volumen añadido tiene un impacto significativo en el rendimiento cuando se envían los datos entre el cliente y el servidor en una red, por lo que mientras mayores son los mensajes, más lento será el intercambio

Por todo lo señalado anteriormente y los resultados arrojados por las pruebas de rendimiento a la tecnología se decidió utilizar SSL para el cifrado de las comunicaciones y garantizar la seguridad en la comunicación entre el Servidor de Comunicación con Terceros y los clientes o terceros.

2.2.2. Estándar de confiabilidad a utilizar.

WS-ReliableMessaging

Proporciona un sistema de reconocimiento flexible en el que los receptores pueden transmitir de manera eficiente la gama de mensajes que tienen (y no) han recibido, también proporciona un mecanismo eficaz de pedido para asegurarse de que los receptores pueden procesar los mensajes en el mismo orden en que fueron enviados, incluso en la cara de reordenación debido a las retransmisiones.

WS-ReliableMessaging no está vinculado a los protocolos de transporte por lo que la vida útil de una conversación puede tener un período o lapso de tiempo en (días, semanas), incluso cuando uno o ambos sistemas se reinicia. Esto permite que las conversaciones que se suspendieron a medio camino (por ejemplo, para permitir el mantenimiento del sistema) se puedan reanudar sin necesidad de retransmitir toda la conversación. Esta especificación no pone ninguna restricción arbitraria en el número de mensajes pendientes en tránsito; dos extremos pueden tener varios mensajes de confianza "en vuelo" en cualquier dirección con una conexión de transporte subyacente único, por lo que la sobrecarga de mensajes para *WS-ReliableMessaging* es muy baja.

WS-ReliableMessaging aprovecha la especificación WS-Addressing que proporciona un mecanismo flexible y extensible para hacer frente a los mensajes y los servicios Web llamada referencia de extremo, y en lugar de enviar un mensaje de confirmación por cada mensaje recibido, permite al destino reconocer de forma acumulativa cada mensaje que ha recibido en un solo elemento, el control compacto. Este elemento de control se pueden enviar en su propio mensaje o incluido en un mensaje posterior. El estado de los mensajes enviados entre los

extremos se puede determinar por medio de un contenedor de secuencias que tiene propiedades estatales y seguimiento incluido.

Con *WS-ReliableMessaging* se puede obtener un mayor grado de fiabilidad para la comunicación de red porque los puntos finales permiten crear y terminar las secuencias de mensaje, un acuse de recibo se envía cada vez que se envía un mensaje y la retransmisión de mensajes se realiza para los mensajes que no fueron recibidos. Al final de la transmisión de datos, un recibo de garantía de reconocimiento se otorga por cada mensaje que se recibe entre los extremos con la eliminación de duplicados y el ordenamiento y la retransmisión de mensajes perdidos.

WS-Reliability

El envío de mensajes en WS-Reliability es individual, debe utilizar un identificador de grupo único. Un acuse de recibo debe ser enviado en respuesta a cada mensaje de error, mientras que WS-ReliableMessage el destino de RM debe enviar un acuse de recibo por cada mensaje recibido de la Fuente RM sin importar si el mensaje tenga error o no.

WS-Reliability contrata el productor y el consumidor de mensajes en todo el ciclo de enviar un mensaje de confianza. El productor especifica el modo de respuesta que se requiere de un consumidor y se mantiene activa durante todo el proceso hasta que un acuse de recibo llega a su destino.

Para la selección del estándar de confiabilidad, se utilizo una matriz de decisión que efectúa una comparación de alternativas contra puntajes.

A continuación se exponen una serie de criterios utilizados en la matriz de decisión ver (Anexo I) que fueron evaluados para cada especificación a fin de realizar los análisis apropiados y seleccionar la mejor opción:

1. Facilidad de implementación: este criterio pretende evaluar cuán 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: 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. *Integración con gSOAP:* la herramienta a utilizar en este caso gSOAP debe posibilitar una integración con los estándares seleccionados a implementar.

4. *Soporte 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.

5. Rendimiento*:* Para la selección de los estándares correcto es necesario tener en cuenta cuál de ellos no afecta la velocidad de trasmisión de mensajes, donde el rendimiento en la transmisión después de aplicado el estándar seleccionado sea similar a su anterior velocidad o mejor.

Atendiendo a los cinco criterios utilizados y a través de la información arrojada por la matriz de decisión que se muestra en la siguiente figura se utilizará el estándar *WS-ReliableMessaging* para garantizar la confiabilidad en la interacción del Subsistema de Comunicación con Terceros y los clientes o terceros.

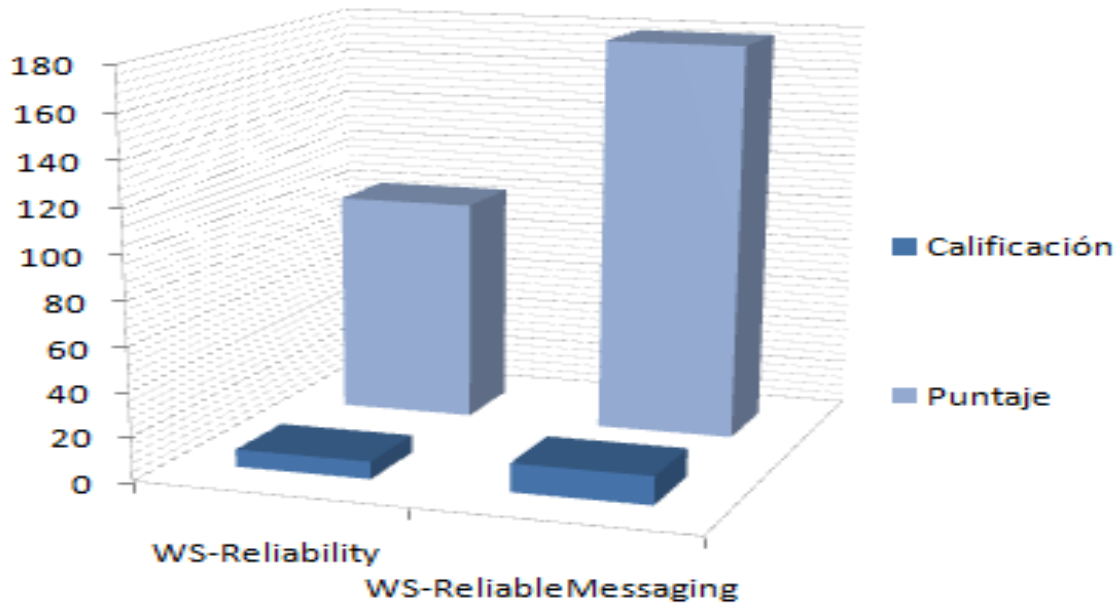

Figura 7: Matriz de selección comparando los estándares de confiabilidad de Servicios Web.

2.3. Metodología, lenguajes y herramientas de desarrollo utilizadas.

La calidad de un producto de software es determinada en gran medida por la calidad del proceso utilizado para desarrollarlo y mantenerlo, por lo que a continuación se aborda acerca de la metodología de desarrollo, lenguaje de programación, herramienta y lenguaje de modelado, así como la herramienta de desarrollo a utilizar en la aplicación de estándares de calidad de servicios Web.

2.3.1. Metodología de desarrollo.

La metodología que guiará el desarrollo será RUP (Rational Unified Process), la cual fue utilizada en el desarrollo del subsistema de Comunicación con Terceros, fue desarrollada en 1998 por Grady Booch, Ivar Jacobson y James Rumbaugh. Se caracteriza por ser dirigida por Casos de Uso, centrada en la arquitectura, iterativa e incremental, además proporciona una visión completa de la construcción del producto. (12) RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al culminarse cada una de las iteraciones. Es más apropiada para proyectos de gran envergadura, como el SCADA Guardián del ALBA, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas.

2.3.2. Lenguaje de Modelado.

El lenguaje de modelado a utilizar será UML (Unified Modeling Language) es desde finales de 1997 un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software. (12) Es válido destacar que UML es un lenguaje de modelado 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.(12)

2.3.3 Herramienta CASE.

Se utilizará para el desarrollo de los artefactos generados por el diseño: *Visual Paradigm*, la cual es una herramienta CASE que utiliza UML, como lenguaje de modelado. Esta herramienta es colaborativa, por lo que soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o pdf, y permite control de versiones. Visual Paradigm para UML es apoyado por un conjunto de lenguajes tanto en la generación del código como en la Ingeniería Inversa dentro de los que se encuentra C + +. Para maximizar la interoperabilidad de sus productos de otras aplicaciones, *Visual Paradigm* introdujo la importación y exportación de modelos de proyecto desde o hasta un formato XML. Los usuarios y proveedores de tecnología pueden integrar *Visual Paradigm* en cada uno de sus modelos para utilizarlos en sus soluciones con un mínimo esfuerzo.

2.3.4. Lenguaje de programación.

El lenguaje de programación a utilizar en el desarrollo será C++, el cual fue creado en la década de 1980 por Bjarne Stroustrup como extensión del lenguaje C. Este lenguaje es utilizado ampliamente en la industria del software. En este caso se cuenta con varias razones para seleccionarlo como lenguaje base de la solución propuesta. El conocimiento que la gran mayoría del equipo de desarrollo tiene de este lenguaje y el hecho de que la concepción del Guardián del ALBA desde su definición ha sido el desarrollo en C++ influyeron en gran medida para su selección, además los recursos con que cuenta son muy útiles en la rama de la automatización. Dentro de las características validas a destacar de este lenguaje se pueden mencionar que C++ es un lenguaje híbrido, multiplataforma, orientado a objetos e imperativo. Usa tipos de datos fuertes y estáticos y la eficiencia en tiempo de ejecución de C/C++ está por encima de todos los lenguajes de programación de alto nivel. (3)

2.3.5. Entorno de desarrollo.

El entorno de desarrollo a utilizar será Eclipse, un proyecto de software libre, multiplataforma que fue desarrollado por IBM y en la actualidad lo mantiene la Fundación Eclipse. En sus inicios este IDE se desarrolló para los programadores que utilizaban el lenguaje Java. Actualmente emplea extensiones para agregar funcionalidades en dependencia de las necesidades del desarrollador, gracias a estas extensiones se ha extendido el soporte de Eclipse hasta lenguajes como C/C++, Phyton, PHP y otros. Además permite utilizar lenguajes de procesado de texto, aplicaciones de red, Sistemas de Gestión de Bases de Datos. Brinda soporte para Sistemas de Control de Versiones e incluye extensiones para realizar pruebas de unidad y para casi cualquier cosa, entre ellos se encuentran los de herramientas de revisión de código. (3)

2.3.6 Generador de documentación

Para la documentación del código fuente se utilizará el paquete *Doxygen* que permite generar la documentación correspondiente a una aplicación. Puede generar archivos en HTML y/o un manual de referencia a partir de un grupo de ficheros fuente documentados. Se decide utilizar Doxygen ya que contiene un sistema de documentación para C++ que es el lenguaje de programación que se utiliza en el Subsistema de Comunicación con Terceros. Además, brinda la facilidad de que la documentación se extrae directamente de las fuentes, lo que hace mucho más fácil mantener la consistencia con el código fuente.

2.3.7 gSOAP, herramienta de desarrollo de Servicios Web.

gSOAP, es una alternativa de código abierto para todos aquellos programadores de C++, que desean disfrutar de las bondades de los Servicios Web. Proporciona interoperabilidad, protocolos de seguridad, y Transferencia a través de HTTP (S), TCP / UDP y otros protocolos de transporte.

La herramienta gSOAP en su versión 2.8, 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++. (3)Se distribuye con tres tipos de licencia: GNU GPL, código abierto público gSOAP y «comercial». Es una herramienta multiplataforma que permite trabajar con SOAP, WSDL, UDDI y con otras tecnologías como *WS-Addressing y WS-Security*, *WS-Reliably Messaging* por lo cual será utilizado en la implementación de estándares de calidad a los servicios web del subsistema de Comunicación con Terceros del GALBA.

2.3.7.1. Extensiones de gSOAP.

Los plug-ins o extensiones en gSOAP son un mecanismo de extensión de las funcionalidades de gSOAP. Cuando la extensión se registra con gSOAP, tiene pleno acceso a la configuración de tiempo de ejecución y las devoluciones de llamadas a funciones gSOAP. Dentro de las extensiones más utilizadas se pueden mencionar.

- La extensión para WS-Security.
- La extensión para WS-ReliableMessaging.
- La extensión para WS-Addressing.

2.4. Subsistema de Comunicación con Terceros.

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

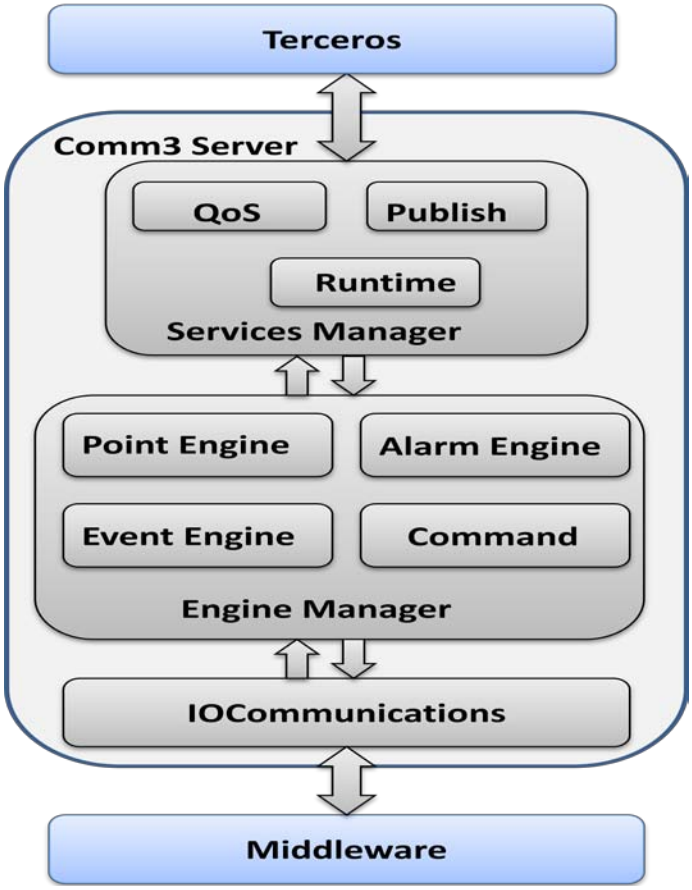

Figura 8: Subsistema de Comunicación con Terceros.

Terceros: Hace referencia a toda aquella aplicación externa incluyendo otros sistemas SCADA que requieran intercambiar información con el sistema SCADA Guardián del ALBA, obteniendo información relacionada con variables, alarmas y eventos.

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

- El paquete *Runtime* es el encargado de inicializar el servidor, garantizando que esté listo para responder a las peticiones enviadas desde las aplicaciones externas. Agrupa el conjunto de elementos de diseño que definen la lógica para iniciar el subsistema de

comunicación con terceros y la lógica de implementación de los servicios que puede invocar el cliente.

- El paquete *QoS* es el encargado de aplicar elementos de calidad de servicio (seguridad y confiabilidad) a la comunicación que establece el servidor de comunicación con terceros con sistemas externos.

- El paquete *Publish* es el encargado de publicar en un nodo UDDI la definición y descripción de los servicios web del servidor de comunicación con terceros para que posteriormente aplicaciones externas puedan localizar y utilizar esta descripción en la implementación de aplicaciones que se comunique con el GALBA.

Engine Manager: Se define la lógica para las invocaciones de datos del tipo puntos, alarmas, eventos y comandos. Este paquete está estructurado por un grupo de paquetes (*Point Engine, Alarm Engine, Event Engine y Command Engine*) con vistas a obtener una mejor organización del sistema y los elementos que lo componen. Se definen además los mecanismos de autenticación y control de acceso a los recursos, por cada llamada que llega de los clientes, se verifica que los datos solicitados se corresponden con los recursos a los cuales tienen permiso.

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

***Middleware*:** Representa al middleware del GALBA y las conexiones a establecer con este para el acceso a los datos, la comunicación con el servidor de seguridad y la obtención de la configuración.

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

El Subsistema de Comunicación con Terceros está dividido en tres capas como se muestra en la figura 10. La arquitectura tres capas posibilita que el mantenimiento y mejoras a la solución que se proponga sea más fácil de manejar, debido al bajo acoplamiento y alta cohesión entre las capas. Al implementar esta arquitectura se obtiene escalabilidad, tolerancia a fallos y un mejor rendimiento de la solución propuesta. El uso de capas ayuda a controlar y encapsular la complejidad y permite obtener una solución de fácil reutilización por otros sistemas que tengan la necesidad de contar con una solución de comunicación con aplicaciones externas. Además, posibilita el trabajo colaborativo, es decir, diferentes personas pueden trabajar en el desarrollo

u optimización de capas diferentes con una necesidad mínima de comunicación entre ellos debido a que las interacciones de la capa *Application* con la capa *Business*, y la de la capa *Business* con la capa *Communication* es mínima y se realiza a través de interfaces bien definidas.

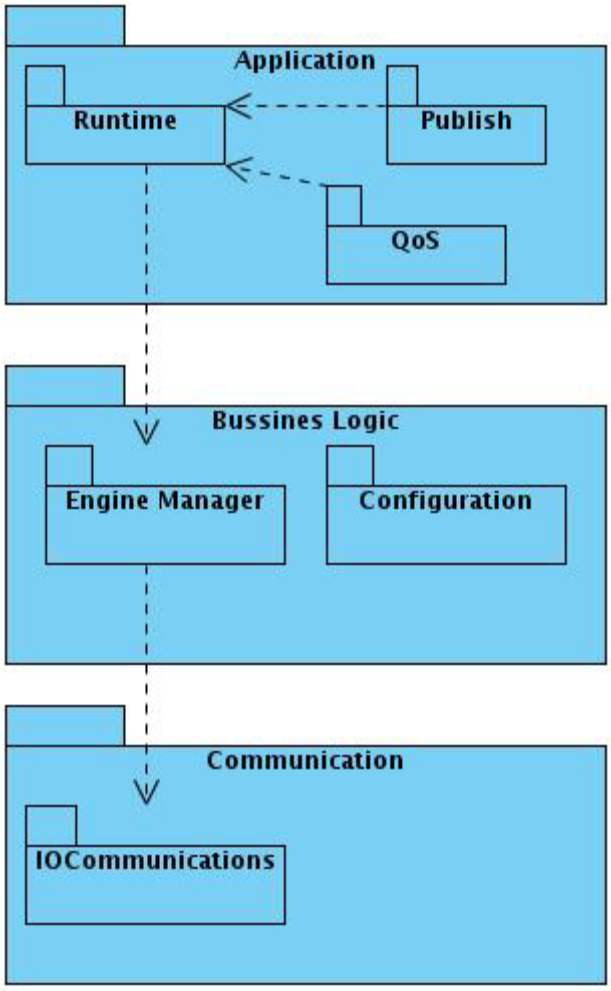

Figura 9: Vista lógica de la arquitectura del Subsistema de Comunicación con Terceros.

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

***Application* (Capa de presentación o aplicación):** Esta capa tiene la responsabilidad de exponer las interfaces de comunicación que van a permitir a las aplicaciones externas solicitar información de puntos, alarmas y eventos y la interacción a través de comandos. Las responsabilidades de esta capa están divididas en paquetes. El paquete *Runtime,* que encapsula las clases que definen la lógica para inicializar el servidor de Comm3 que incluye inicializar la comunicación vía SOAP*, Publish,* que publica la descripción de los servicios web y

QoS que le agrega calidad a la comunicación. *Application* depende de la capa de negocio para dar respuesta las peticiones de los clientes.

***Business Logic* (Capa de Lógica de Negocio):** Esta capa tiene la responsabilidad de resolver las peticiones de los terceros invocadas desde la capa superior, define la lógica de negocio relacionada con las suscripciones, acceso a la información del GALBA, invocación de comandos y los mecanismos de seguridad. Las responsabilidades están divididas en las clases contenidas en varios paquetes, entre ellos: el paquete *Comm3*, el paquete *EngineManager*, que encapsula las funcionalidades relacionadas con la suscripción y acceso a la información del GALBA y el paquete *Configuration* encargado de obtener y almacenar la configuración del servidor de Comm3.

***Communication* (Capa de Comunicación):** Esta capa tiene como principal responsabilidad realizar las peticiones a los módulos del GALBA a través de las interfaces de comunicación del *middleware*. Las responsabilidades están divididas en el paquete *IO Communication*, encargado de gestionar las comunicaciones a través del *middleware*.

2.6. Paquetes de diseño arquitectónicamente significativos.

A continuación se muestran los diagramas de clases del diseño de los paquetes de diseño.

2.6.1. Paquete QoS

En este paquete se encuentran las clases que contienen todas las funcionalidades relacionadas con la utilización del *pluging wsrm* para garantizar la confiabilidad y la biblioteca g*soapssl*++ la que garantizará todo lo relacionada con la seguridad en la comunicación del subsistema de comunicación con terceros y los externos.

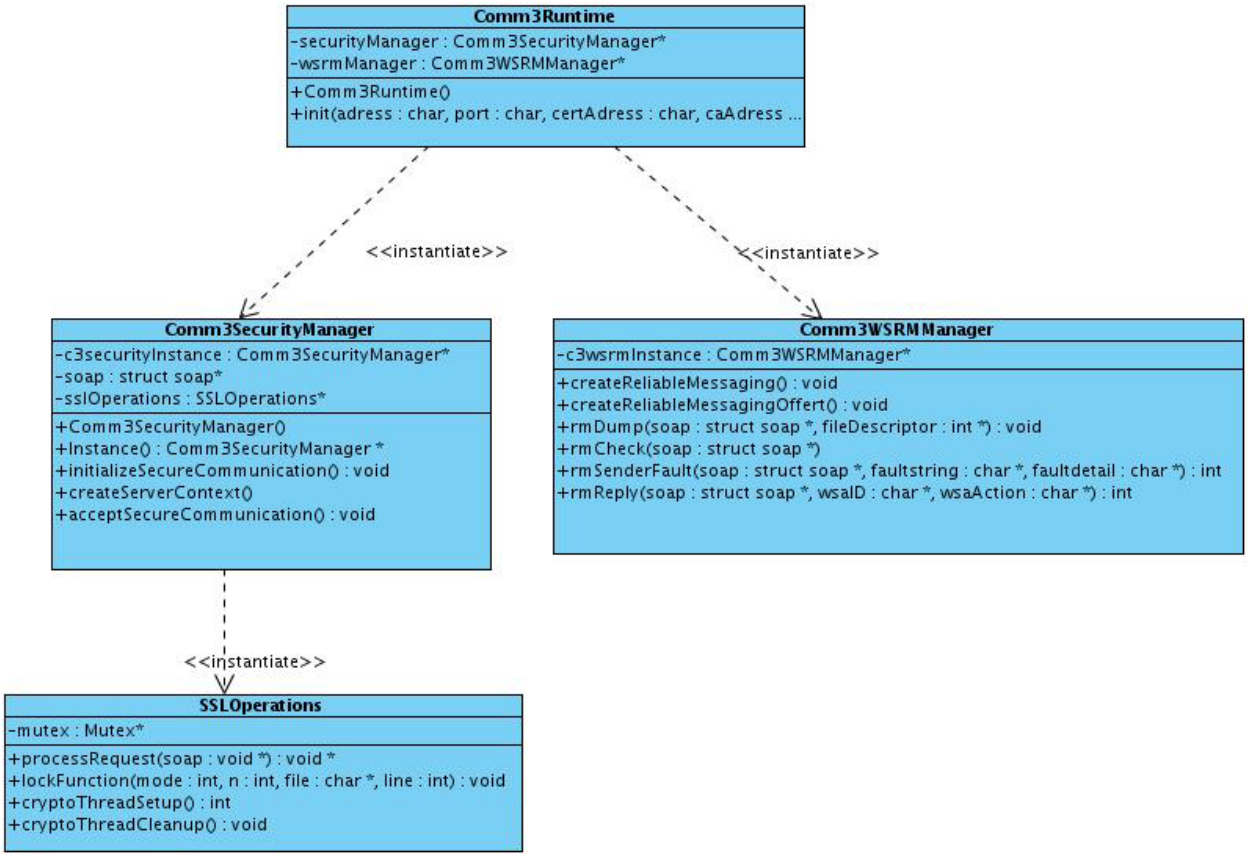

Figura 10: Diagrama de Clases del Paquete QoS.

2.6.1.1. Descripción de clases y métodos del paquete QoS

Comm3WSRMManager	
Descripción de la clase: Clase encargada de gestionar todas las operaciones que posibiliten la comunicación entre el Servidor de Comunicación con Terceros y los clientes o terceros de manera confiable. Esta clase es instanciada desde la clase Comm3Runtime para poder utilizar sus funcionalidades.	
Atributos	
`c3wsrmInstance``: Comm3WSRMManager*`	Instancia de la clase Comm3WSRMManager
Métodos	
`rmCheck``(soap: struct soap*)`	Funcionalidad encargada de detectar la utilización de WS-Addressing. Esta función debe de ser llamada en cada operación que

	soporte WS-ReliableMessaging.
rmDump(soap: struct soap*,fileDescriptor: int)	Función utilizada cuando el plugin se registra en gSOAP.
rmSenderFault(soap : struct soap*, faultstring : char*, faultdetail: char*): int	Funcionalidad encargada de gestionar el envío de fallas del funcionamiento. El fallo se devuelve al remitente (cliente o tercero).
rmReply (soap : struct soap*, wsaID : char *,wsaAction : char *)	Funcionalidad que se ejecuta cada vez que un servicio web va a enviar alguna respuesta.

Tabla 1: Descripción de la clase Comm3WSRMManager.

Comm3SecurityManager	
Descripción de la clase: Clase encargada de gestionar todas las operaciones que posibiliten la comunicación entre el Servidor de Comunicación con Terceros y los clientes o terceros de manera segura utilizando SSL para la encriptación de los mensajes que viajan por la red. Esta clase es instanciada desde la clase Comm3Runtime para poder utilizar sus funcionalidades.	
Atributos	
c3securityInstance: Comm3SecurityManager*	Instancia de la clase Comm3SecurityManager
soap: struct soap*	Instancia del entorno de ejecución de gSOAP necesario para la mayoría de las operaciones que realiza esta clase.
sslOperations: SSLOPerations*	Instancia de la clase SSLOPerations
Métodos	
Comm3SecurityManager()	Constructor de la clase Comm3SecurityManager
Instance(): Comm3SecurityManager*	Funcionalidad que permite acceder a una única instancia de la clase Comm3SecurityManager.
initializeSecureCommunications(address : char*, port : char *, certAdress : char *, caAdress: char *, password : char***)**	Funcionalidad que inicializa de forma segura el servidor de Comunicación con Terceros.
createServerContext(): void	Esta función inicializa el contexto SSL en el

	servidor. El archivo server.pem clave es la clave privada del servidor concatenado con su certificado.
acceptSecureCommunication(): void	Soporte de SSL independiente. EL servicios Web de gSOAP se activa llamando soap_ssl_accept.

Tabla 2: Descripción de la clase Comm3SecurityManager.

SSLOperations	
Descripción de la clase: Como el subsistema de comunicación con terceros es una aplicación multiproceso, se tiene que llamar a las funcionalidades CRYPTO_thread_setup() y CRYPTO_thread_cleanup(). Estas rutinas son necesarias para aplicaciones de subprocesos múltiples que utilizan SSL.	
Atributos	
mutex : Mutex*	Semáforo para manejar secciones críticas.
Métodos	
processRequest(void*): void *	Funcionalidad utilizada para levantar un hilo de ejecución independiente por cada petición de un cliente o tercero
lockFunction(mode : int, n : int, file : char *, line : int): void	Funcionalidad que bloquea un semáforo.
cryptoThreadSetup(): int	Inicializa un hilo de ejecución.
cryptoThreadCleanup(): void	Elimina un hilo de ejecución.

Tabla 3: Descripción de la clase SSLOperations.

2.6.1.2. Diagramas de Secuencia del paquete QoS.

El siguiente diagrama de secuencia muestra como se inicia la comunicación entre el Servidor de Comunicación con Terceros y los externos usando comunicación segura mediante SSL.

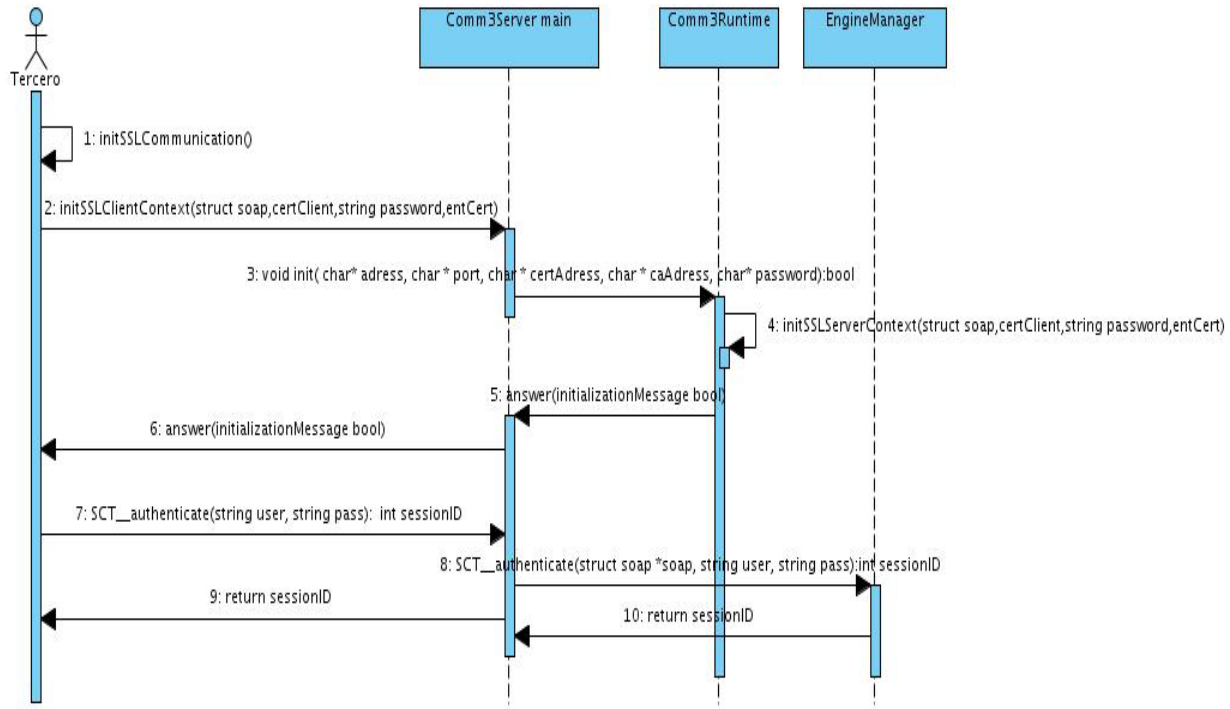

Figura 11: Diagrama de Secuencia: Iniciar comunicación segura del paquete QoS.

El siguiente diagrama de secuencia muestra la interacción del cliente o tercero con el Servidor de Comunicación con Terceros del GALBA para solicitar valores de puntos utilizando un mensaje (ACK acuse de recibo) que se envía para confirmar que un mensaje o un conjunto de mensajes han llegado.

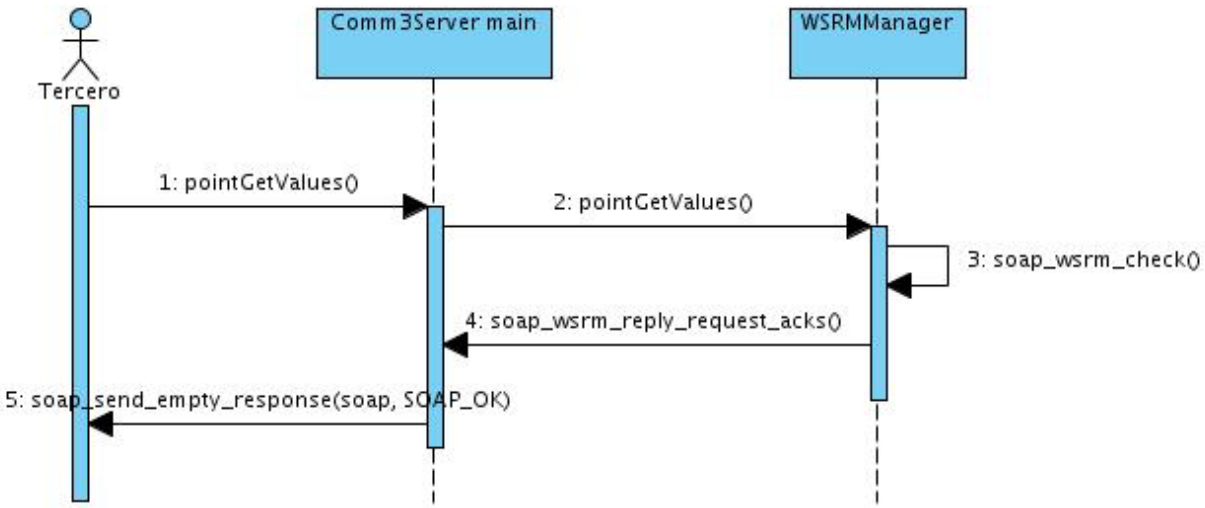

Figura 12 : Diagrama de Secuencia: Obtener valores de puntos utilizando wsrm.

2.6.2. Paquete *Runtime*. Solo elementos relacionados con la interacción con el Servidor de Seguridad.

En este paquete se encuentran las clases que contienen todas las funcionalidades relacionadas con la inicialización del Servidor de Comunicación con Terceros. Entiéndase por inicialización, el inicio de la comunicación con los terceros, la comunicación que se establecerá entre Comm3 y el Servidor de Configuración para obtener la configuración y entre Comm3 y el Servidor de Seguridad para autenticarse y poder determinar el acceso de cada usuario a los recursos. En este epígrafe se abordara principalmente la comunicación con el Servidor de Seguridad del SCADA Guardián del ALBA.

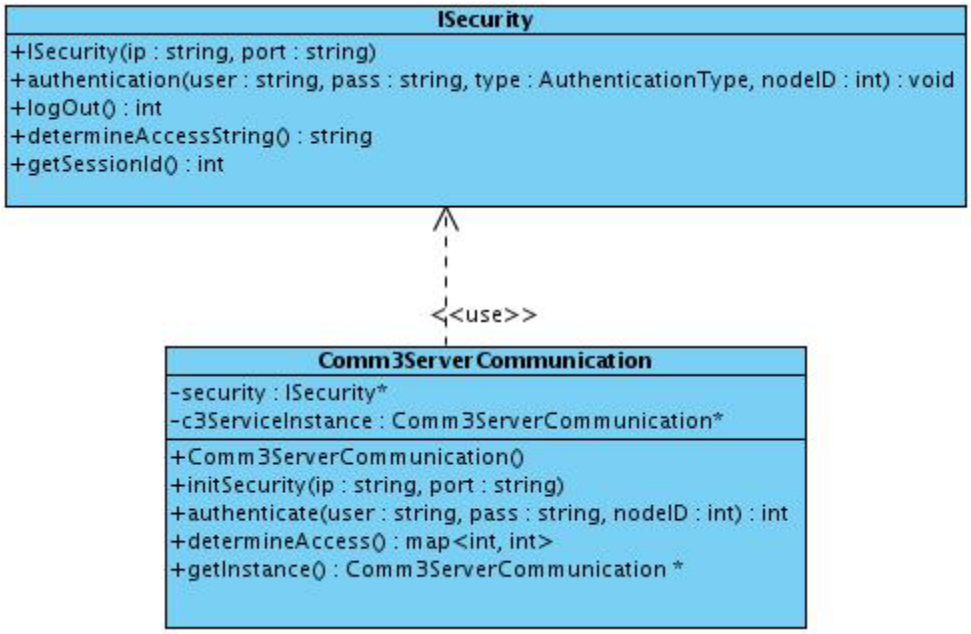

Figura 13: Diagrama de Clases que muestra la interacción con el Servidor de Seguridad mediante la biblioteca *ISecurity*.

2.6.2.1. Descripción de clases y métodos relacionados con la interacción con el Servidor de Seguridad.

ISecurity	
Descripción de la clase: Clase con todas las funcionalidades que ofrece el Servidor de Seguridad a sus clientes.	
Métodos	
`ISecurity``(ip: string,port: string)`	Constructor que recibe la dirección del servidor traducidos en ip y puerto.
`authentication``(user:string, password:string,type:AuthenticationType,nodeID:int): void`	Método encargado de la autenticación de los usuarios ya sea por medio de usuario y contraseña o por huellas digitales.
`logOut``(): int`	Método encargado de eliminar la sesión de usuario del sistema.
`determineAccessString``(): string`	Carga en la sesión todos los permisos asociados al usuario.
`getSessionId``(): int`	Permite obtener el identificador de la sesión. Retorna el identificador de sesión 0, en caso de no estar activa.

Tabla 4: Descripción de la clase ISecurity.

Comm3ServerCommunication	
Descripción de la clase: Clase que agrupa las funcionalidades relacionadas con la inicialización de la comunicación entre el Servidor de Comunicación con Terceros y la capa de comunicaciones del SCADA GALBA el Servidor de Seguridad y el Servidor de Configuración.	
Atributos	
c3ServiceInstance:Comm3ServerCommunication*	Instancia de la clase Comm3ServerCommunication.
security: Isecurity*	Instancia de la clase *ISecurity* a través de la cual se acceden a las funcionalidades del servidor de Seguridad.
Métodos	
Comm3ServerCommunication()	Constructor por defecto de la clase Comm3ServerCommunication.
initSecurity(ip: string,port: string)	Método encargado de establecer la comunicación con el servidor de Seguridad por una dirección ip y un puerto determinado.
authenticate(user:string, password:string,nodeID:int): void	Método encargado de la autenticación de los usuarios (terceros) en el servidor de Seguridad mediante usuario y contraseña.
determineAccessString():map<int, int>	Método que almacena en un mapa cada recurso con los permisos que tiene asociados.
getInstance():Comm3ServerCommunication*	Permite obtener una única instancia de la clase Comm3ServerCommunication.

Tabla 5: Descripción de la clase *Comm3ServerCommunication*.

2.6.2.2. Diagramas de Secuencia relacionados con la interacción con el Servidor de Seguridad.

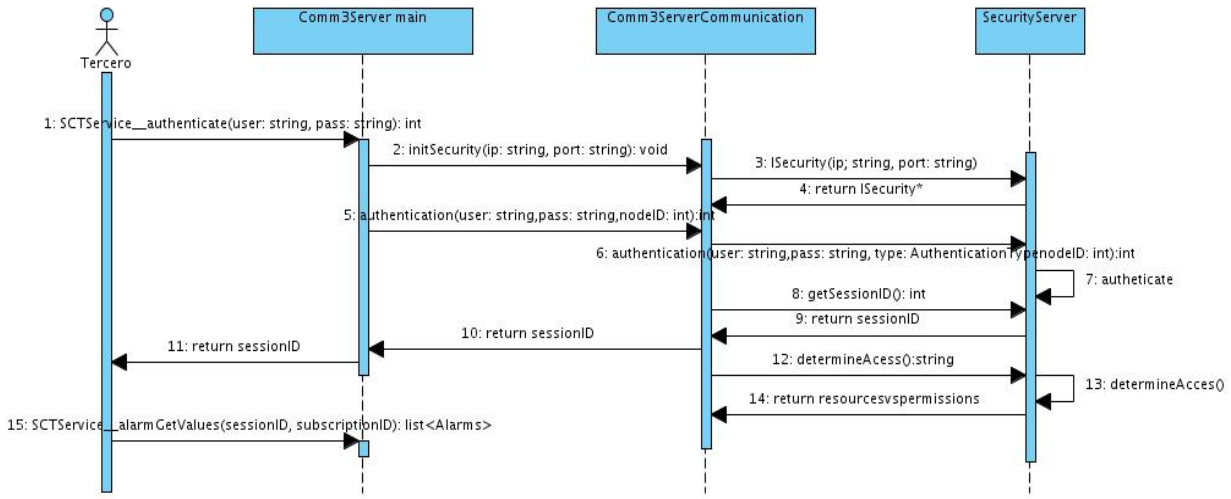

Figura 14: Diagrama de Secuencia: Interacción con el Servidor de Seguridad.

Capítulo 3: Implementación y pruebas.

En este capítulo se abordará sobre la implementación del paquete QoS y la implementación de la interacción con el servidor de seguridad. Se muestran los diagramas de despliegue y de componentes, el estilo de código a utilizar, y las pruebas para validar el funcionamiento de este nuevo paquete.

3.1 Implementación

3.1.1 Estándar de codificación

A la hora de programar es necesario seguir un estilo. Este permitirá revisar, mantener y actualizar el código de una manera más sencilla y ordenada. Seguir estas normas también evitará caer en errores y malas prácticas que dificultan la comprensión de las líneas de código.

A continuación se muestra el estilo de código utilizado en el Subsistema de Comunicación con Terceros. Este estilo se definió para ser utilizado por todos los subsistemas del sistema SCADA Guardián del ALBA

- **Nombres.**

• Los nombres de las clases son sustantivos singulares.

• Los nombres deben reflejar el ¿qué? y no el ¿cómo?

• Los nombres no deben revelar detalles de implementación.

• Escoger nombres lo suficientemente largos para ser expresivos, pero evitando manejar nombres que dificulten la labor de implantación.

• Evitar nombres que permitan una interpretación subjetiva (evitar ambigüedad y asegurar abstracción).

• Evitar redundancia no repitiendo nombres de clases en sus elementos.

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

• Dado que los nombres generalmente son el producto de concatenar varias palabras, se debe emplear mayúscula para el inicio de cada palabra y minúscula para el resto de las letras para el caso de los nombres de métodos y funciones. Para el caso de los nombres de variables y atributos debe aplicarse la misma convención, con excepción de la primera letra del nombre, la cual debe ser en minúscula.

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

• Los nombres de constantes deben contener solo letras mayúsculas.

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

• Los nombres de los métodos son frases que incluyen verbos.

• Los nombres de los atributos y parámetros son frases con sustantivos.

• Evitar el uso de nombres idénticos para distinto propósito.

- **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.

- **Documentación y Comentarios.**

• En el código debe documentarse en forma explicativa los pasos que se van ejecutando.

• Emplear oraciones completas al documentar el código.

• Documentar mientras se programa.

• Documentar cualquiera cosa que no sea obvia en el código.

• Documentar eliminación de errores y cambios sobre el código.

• Al modificar el código se deben actualizar todos los comentarios y documentación asociada.

• Documentar cada rutina agregando: nombre del desarrollador, fecha, parámetros de entrada, valores de retorno, precondiciones, pos-condiciones, dependencia con otros métodos o funciones y descripción general del algoritmo. Además, de realizarse cambios al código, debe indicarse el nombre de la persona que realizó el cambio, la fecha y la descripción del cambio, comenzando desde el o los cambios más recientes.

• Evitar agregar comentarios al final de líneas de código, salvo en el caso de declaraciones. En este caso tales comentarios deben estar alineados.

• Antes de la entrega de la aplicación, eliminar todos los comentarios superfluos y/o temporales con la finalidad de evitar confusiones en su mantenimiento.

- **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 proporcionen el valor de tal variable, para mantener el encapsulamiento.

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

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

• Liberar apuntadores de manera explícita.

• Emplear i, j, k, l, p, q, r para contadores en ciclos.

• Comentar siempre las llaves que cierran.

• Emplear al máximo operadores del tipo: +=, *=, /=, -=, ++, --, entre otros.

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

• Emplear correctamente los tipos de ciclos: si es al menos una vez usar do-while, si es ninguna o más veces usar while-do, y si se conoce el número exacto de ciclos usar for.

• Inicializar todas las variables.

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

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

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

3.1.2 Estándar de documentación

Los formatos brindados por la herramienta Doxygen empleada para la documentación del sistema se explican a continuación:

- **Estilo de bloques de documentación.**

Se adopta el estilo de bloques de documentación JavaDoc, el cual consiste en un bloque de comentario de estilo C, los asteriscos que se encuentran en la línea de la mitad son opcionales.

```
/**
* Texto
*/
```

Figura14: Estilo de bloques de documentación.

- **Descripción breve con el comando @brief.**

Para hacer una descripción breve se adopta el uso del comando \brief o @brief en el bloque de comentarios ya descrito. La acción de este comando termina al final de un párrafo, de tal manera que la descripción detallada sigue después de una línea vacía, tal como lo muestra el ejemplo:

```
/**
 * @brief Descripción breve.
 * Continuación de la descripción breve.
 *
 * La descripción detallada comienza aquí, nótese
 * que se debe dejar una línea en blanco para lograr
 * tener las dos descripciones (breve y detallada)
 */
```

Figura 15: Descripción breve con el comando @brief.

Es válido aclarar que si no se utiliza el comando @brief *Doxygen* tomará la descripción hecha en el bloque como una descripción detallada y no habrá descripción breve.

- **Descripción de argumentos y métodos.**

A la hora de hacer una descripción de los argumentos de métodos y funciones ésta se hará en línea, es decir, luego de la declaración de cada uno de los argumentos se da una breve descripción de cada uno de ellos, en el siguiente ejemplo se muestra el formato a utilizar:

```
int multFunction ( int c , /**<Primer operando de la operación */
                   int d , /**<Segundo operando de la operación */
                   int e /**<Tercer operando de la operación */ );
```

Figura 16: Descripción de argumentos y métodos.

- **Documentación de tipos de datos.**

Para describir la función y los elementos que componen los tipos de datos definidos se pueden utilizar los formatos especificados en los apartados anteriores. A continuación se muestra un ejemplo:

```
/**
* @brief Enumerado de los tipos de datos admitidos
*
* Descripción más detallada de la función de este tipo de dato.
*/
enum DataTypes{ INTEGER, /**<Puede ser un valor de tipo entero */
                DOUBLE, /**<Puede ser un valor de tipo double */
                STRING /**<Puede ser un valor de tipo string */ };
```

Figura 17: Documentación de tipos de datos.

Observamos que se genera una descripción breve seguida de una descripción más detallada para la función, luego se explica brevemente el valor de retorno de la misma y la descripción de los parámetros usando el formato de documentación en línea.

- **Comandos @author y @date.**

Es importante que se especifique el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @author y @date para el nombre del autor y la fecha respectivamente, en el siguiente ejemplo se puede ver la acción de estos comandos:

```
/**@brief Descripción breve
 *
 * Aquí comienza la descripción detallada
 * @author Yerenia Breña Perez ybrena@estudiantes.uci.cu
 * @date 14-04-2011
 */
```

Figura 18: Comandos @author y @date.

- **Comando @see.**

Existe otro comando útil que permite hacer referencias a otras clases o métodos cuando esto sea necesario, es importante usar este comando en la documentación a la hora de implantar los métodos o funciones propias de alguna clase, este comando es @see y su uso se muestra en el siguiente ejemplo:

```
/** * Constructor de la clase
* @see Test::Test()
*/
Test1();
```

Figura 19: Comando @see.

Esto generará en la documentación para el constructor de la clase de prueba Test1 una referencia hacia la documentación existente para el constructor de la clase de prueba Test.
Si se quiere hacer una referencia desde cualquier estructura hacia la documentación completa de una clase ya documentada, solo se debe utilizar el comando @see seguido del nombre de la clase.

3.1.3 Diagrama de componentes

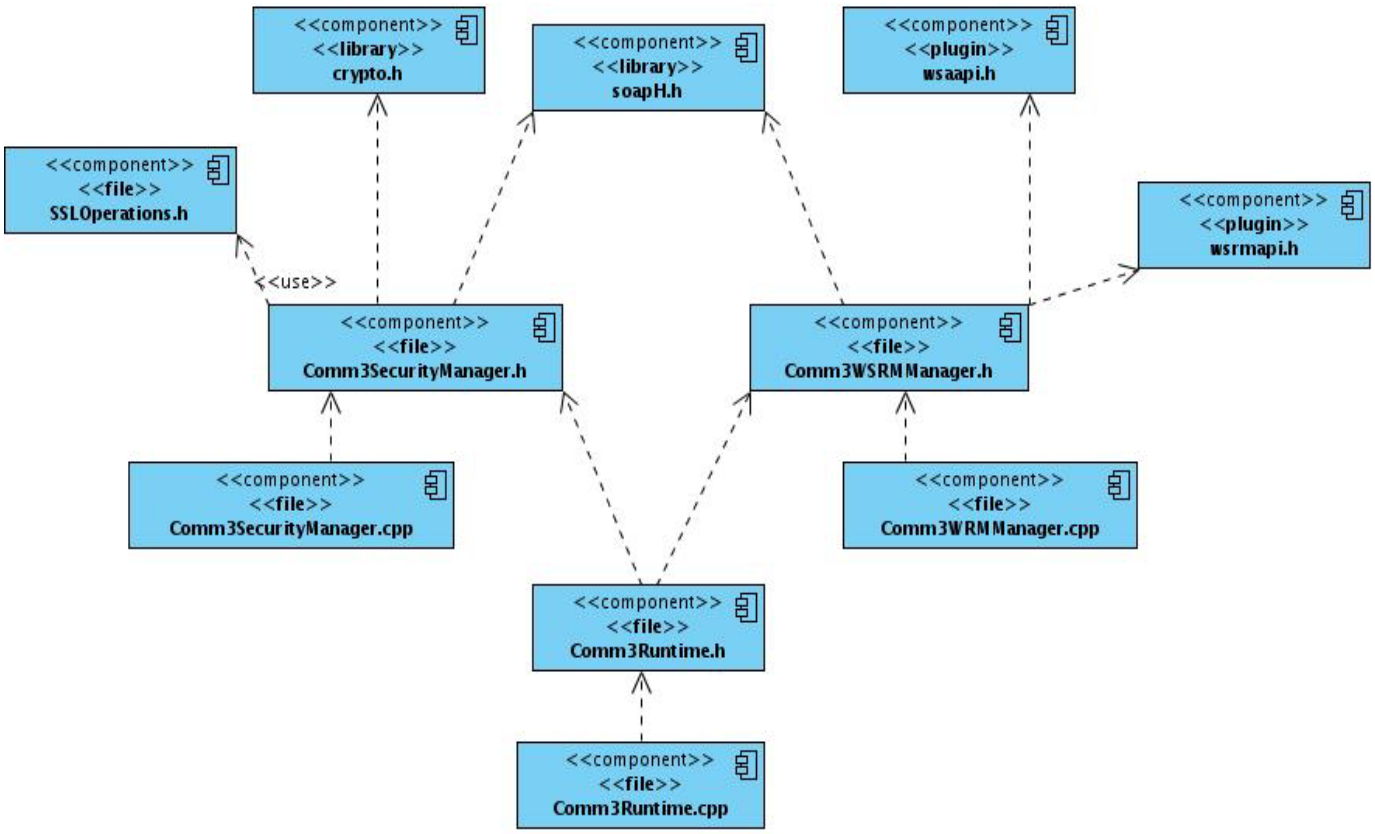

Figura 20: Diagrama de componentes del paquete QoS.

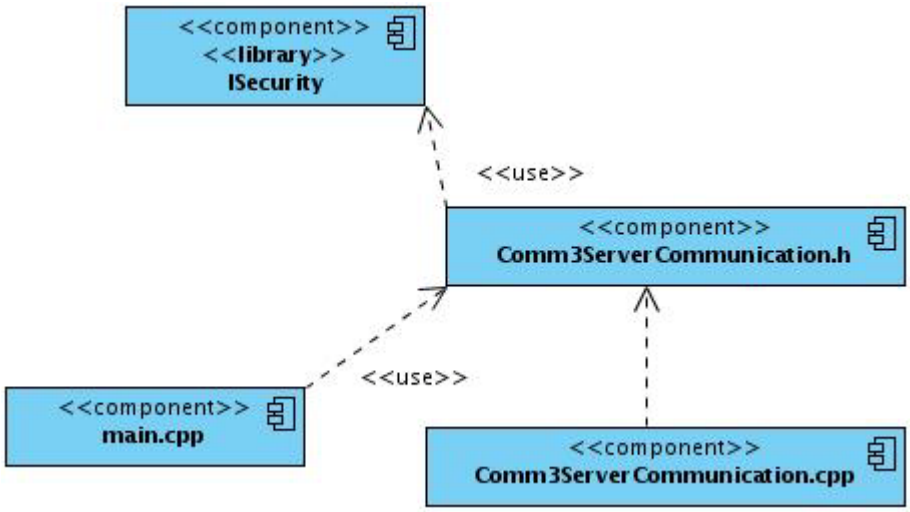

Figura 21: Diagrama de Componentes de la interacción del Servidor de Comunicación con Terceros con el Servidor de Seguridad.

3.1.4 Diagrama de despliegue

La implantación o despliegue del sistema, satisfacen la solución planteada y muestra a nivel de nodos físicos la comunicación entre el servidor de Comunicación con Terceros y los clientes o terceros. A continuación se muestra la vista de despliegue en la siguiente figura.

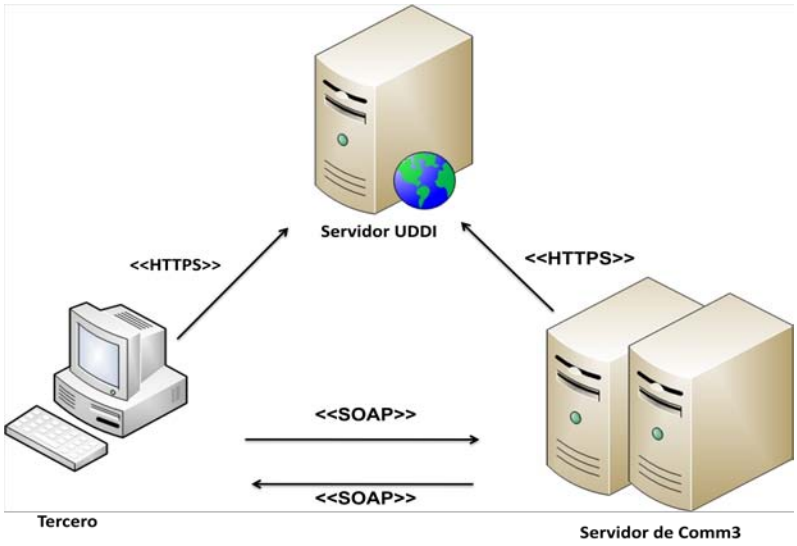

Figura 22: Diagrama de despliegue del Servidor de Comunicación con Terceros.

A continuación se expone una breve descripción de los nodos y sus conexiones:

- **Servidor UDDI:** En este nodo reside un servidor o registro UDDI con las descripciones de los servicios brindados por el servidor de Comunicación con Terceros para poder ser localizados por los potenciales usuarios (terceros). En dicho registro se aportarán datos sobre la empresa, los Servicios Web que se ofrecen y demás, también la descripción de las interfaces de uso de cada Servicio Web (WSDL). Cuando algún consumidor solicite dicho Servicio Web, el servidor UDDI le re-direccionará la URI proporcionada por el SCADA Guardián del ALBA.
- **Tercero:** En este nodo radica la aplicación personalizada que requiere acceder a la información del sistema SCADA Guardián del ALBA a través de los servicios web y para ello se comunica por HTTPS con el Servidor UDDI para acceder a la descripción de los servicios que este expone. Cuando el cliente realiza una petición acerca de cualquier servicio y obtiene su descripción, este es capaz de comunicarse con el Subsistema de Comunicación con Terceros a través del protocolo SOAP.
- **Servidor de Comunicación con Terceros:** En este nodo está instalado el Servidor de Comunicación con Terceros como parte del GALBA el cual puede estar instalado en una o varias computadoras. Este servidor inicia su funcionamiento, publicando la descripción de sus servicios en el registro UDDI mediante el protocolo HTTPS y posteriormente comienza a escuchar peticiones de los terceros a través del protocolo SOAP.

El servidor de Comunicación con Terceros no es una aplicación con interfaz gráfica sino un proceso que se pone en ejecución cuando comienza el funcionamiento del GALBA. Este proceso para garantizar la mensajería confiable necesita de una múltiple ejecución, preferiblemente en computadoras diferentes para sí ocurre un problema con el CPU no afecte el funcionamiento de este.

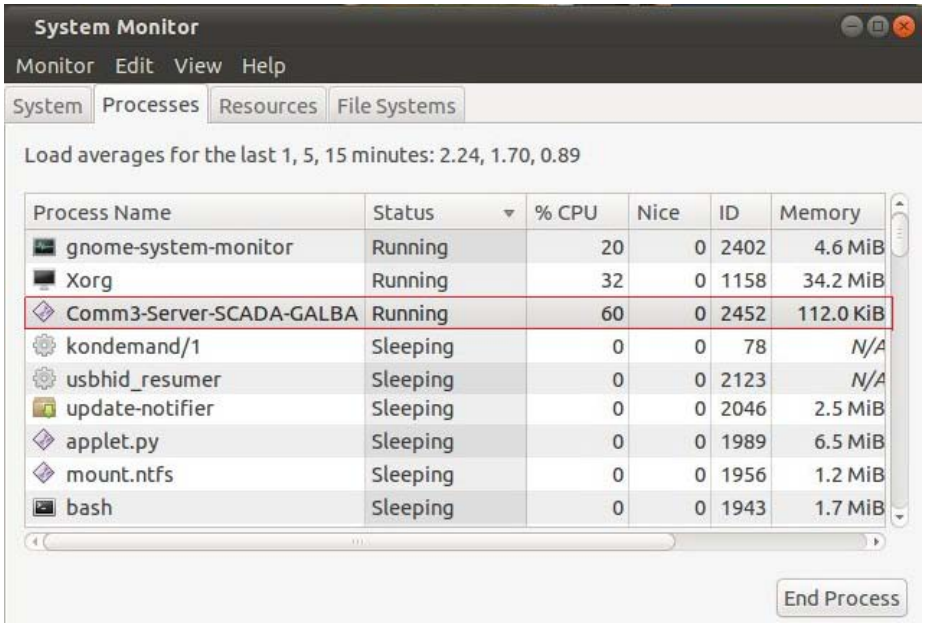

Figura 23: Proceso en ejecución del Servidor de Comunicación con Terceros.

3.2 Pruebas

La realización de pruebas es una actividad en la cual un sistema o componente es ejecutado bajo ciertas condiciones o requerimientos específicos, cuyos resultados son observados, registrados y evaluados. Uno de los niveles de pruebas que existen son las pruebas de sistema, las cuales tienen como objetivo fundamental verificar el software ejerciendo como un todo. A continuación se detallan los casos de prueba elaborados para las funcionalidades con las que cuenta el Servidor de Comunicación con Terceros. Mediante la ejecución de estos casos de prueba se podrá comprobar el correcto funcionamiento del sistema después de haber adicionado los elementos que permiten la seguridad y la confiabilidad en la comunicación.

A continuación se muestran las funcionalidades que agrupan cada uno de los casos de prueba:

#	**Diseño de caso de prueba**	**Funcionalidades relacionadas**
DCP 1	Suscripción y recepción de puntos	Suscripción al servicio de intercambio de puntos. Recepción de puntos.
DCP 2	Suscripción y recepción de alarmas	Suscripción al servicio de intercambio de alarmas.

		Recepción de alarmas.
DCP 3	Suscripción y recepción de eventos	Suscripción al servicio de intercambio de eventos. Recepción de eventos.
DCP 4	Ejecución de comandos	Ejecución de comando de escritura de variable. Ejecución de comando de reinicio de alarma. Ejecución de comando de reconocimiento de alarma. Ejecución de comando de reconocimiento de inhibir alarma. Ejecución de comando de reconocimiento de deshabilitar e inhibir alarma.
DCP 5	Autenticar con el servidor de Seguridad	Autenticación.

Tabla 6: Diseño de casos de prueba y funcionalidades relacionadas.

3.2.1 Ambiente de pruebas

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

Recursos Físicos

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

Recursos Lógicos

- El sistema operativo en el cual se desarrollaron las pruebas fue Debian 6.0 (squeeze), Kernel Linux 2.6.26-1-686.
- La capa de comunicaciones utilizada fue la implementada utilizando la tecnología de sockets.

- La versión del servidor de Seguridad utilizado fue la 1.0 (Revisión 1271)
- La versión del servidor de Configuración utilizado fue la 1.0 (Revisión 1271)
- La velocidad de la red con que se contó fue de 100 megabits por segundo.

3.2.2 Diseño de Casos de Prueba.

Nombre del caso de prueba:	Suscripción y recepción de puntos	
Entrada	**Resultado esperados**	**Resultados obtenidos**
Caso de Prueba #1		
Suscribir un tercero al servicio de puntos con un identificador de sesión inválido.	Acción fallida. El servidor de comunicación con terceros retorna un identificador de suscripción con valor cero.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #2		
Suscribir un tercero al servicio de puntos con un identificador de sesión válido.	El servidor de comunicación con terceros retorna un identificador de suscripción con válido (mayor que cero).	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #3		
Realizar petición de obtener valores de puntos con un identificador de suscripción inválido.	Acción fallida. El servidor de comunicación con terceros retorna una lista vacía de sin valores de puntos.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #4		
Realizar petición de obtener valores de puntos con un identificador de suscripción válido.	El servidor de comunicación con terceros retorna una lista con todos los valores de los puntos solicitados.	Prueba exitosa. Se obtienen los resultados esperados.

Tabla 7: DCP Suscripción y recepción de puntos.

Nombre del caso de prueba:	Suscripción y recepción de alarmas	
Entrada	**Resultado esperados**	**Resultados obtenidos**
Caso de Prueba #1		
Suscribir un tercero al servicio de alarmas con un identificador de sesión inválido.	Acción fallida. El servidor de comunicación con terceros retorna un identificador de suscripción con valor cero.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #2		
Suscribir un tercero al servicio de alarmas con un identificador de sesión válido.	El servidor de comunicación con terceros retorna un identificador de suscripción válido (mayor que cero).	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #3		
Realizar petición de obtener valores de alarmas con un identificador de suscripción inválido.	Acción fallida. El servidor de comunicación con terceros retorna una lista vacía de sin valores de alarmas.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #4		
Realizar petición de obtener valores de alarmas con un identificador de suscripción válido.	El servidor de comunicación con terceros retorna una lista con todos los valores de las alarmas solicitados.	Prueba exitosa. Se obtienen los resultados esperados.

Tabla 8: DCP Suscripción y recepción de alarmas.

Nombre del caso de prueba:	Suscripción y recepción de eventos	
Entrada	**Resultado esperados**	**Resultados obtenidos**
Caso de Prueba #1		
Suscribir un tercero al servicio de eventos con un identificador de sesión inválido.	Acción fallida. El servidor de comunicación con terceros retorna un identificador de suscripción con valor cero.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #2		

Suscribir un tercero al servicio de eventos con un identificador de sesión válido.	El servidor de comunicación con terceros retorna un identificador de suscripción con válido (mayor que cero).	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #3		
Realizar petición de obtener valores de eventos con un identificador de suscripción inválido.	Acción fallida. El servidor de comunicación con terceros retorna una lista vacía de sin valores de eventos.	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #4		
Realizar petición de obtener valores de eventos con un identificador de suscripción válido.	El servidor de comunicación con terceros retorna una lista con todos los valores de los eventos solicitados.	Prueba exitosa. Se obtienen los resultados esperados.

Tabla 9: DCP Suscripción y recepción de eventos.

Nombre del caso de prueba:	Ejecución de comandos	
Entrada	**Resultado esperados**	**Resultados obtenidos**
Caso de Prueba #1		
Escribir valor	Cambio en el valor del punto en el SCADA	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #2		
Reiniciar alarma	Reiniciar alarma en el SCADA	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #3		
Reconocer alarma	Reconocer alarma en el SCADA	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #4		
Inhibir alarma	Inhibir alarma en el SCADA	Prueba exitosa. Se obtienen los resultados esperados.

Caso de Prueba #5		
Inhabilitar e inhibir alarma	Inhabilitar e inhibir alarma en el SCADA	Prueba fallida. El comportamiento es el siguiente: cuando el servidor de Comm3 envía este comando, el SCADA muestra que recibió el comando mostrando los atributos y los diferentes parámetros del mismo, todo de forma correcta sin embargo el SCADA no envía respuesta alguna hacia el servidor de Comm3 y no ejecuta dicho comando en su contexto (Módulo de Adquisición).

Tabla 10: DCP Ejecución de comandos.

Nombre del caso de prueba:	Autenticar con el servidor de Seguridad	
Entrada	**Resultado esperados**	**Resultados obtenidos**
Caso de Prueba #1		
Invocar al método de autenticación con sintaxis en usuario o contraseña incorrecta.	Acción fallida. El servidor de comunicación con terceros retorna un identificador de sesión con valor -1 y el mensaje asociado: "Sintaxis en usuario o password incorrecta."	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #2		
Invocar al método de autenticación con usuario o contraseña incorrecta.	Acción fallida. El servidor de comunicación con terceros retorna un identificador de sesión con valor -2 o -3 y el mensaje asociado: "Usuario o password incorrecto."	Prueba exitosa. Se obtienen los resultados esperados.
Caso de Prueba #3		
Invocar al método de autenticación sin estar en ejecución el servidor de Seguridad.	Acción fallida. El servidor de comunicación con terceros retorna un identificador de sesión con valor -100 y el mensaje asociado: "Error de comunicación con seguridad."	Prueba exitosa. Se obtienen los resultados esperados.

Caso de Prueba #4		
Invocar al método de autenticación con usuario y contraseña correcta.	El servidor de comunicación con terceros retorna un identificador de sesión con valor mayor que cero y el mensaje asociado: "Sesión iniciada correctamente.	Prueba exitosa. Se obtienen los resultados esperados.

Tabla 11: DCP Autenticar con el servidor de Seguridad.

3.2.3 Evaluación de los resultados.

Como conclusión de estas pruebas, de un total de 26 casos, 21 resultaron satisfactorios y 5 demostraron problemas del software. Todos los errores detectados fueron solucionados, lo cual demuestra que el proceso de validar la funcionalidad del paquete QoS en el Servidor de Comunicación con Terceros concluye de manera exitosa.

CONCLUSIONES

Al término de la presente investigación para el desarrollo del paquete de calidad de servicio del Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA, se concluye que:

- WS-Security es el estándar de seguridad más completo pero en cuestiones de rendimiento no es óptimo.
- El estándar SSL proporciona elementos de seguridad claves para la comunicación entre servicios web sin afectar de manera agresiva el rendimiento.
- El estándar WS-RM ofrece garantía de la entrega de los mensajes y se integra fácilmente con la herramienta gSOAP posibilitando la confiabilidad en la comunicación entre el Subsistema de Comunicación con Terceros del sistema SCADA Guardián del Alba y sistemas externos.
- El Servidor de Comunicación con Terceros garantiza la seguridad y la confiabilidad en la comunicación que se establece entre los sistemas externos y el servidor de comunicación con terceros del sistema SCADA Guardián del ALBA, posibilitando el acceso a puntos, alarmas, eventos e interacción a través de comandos.

- El Servidor de Comunicación con Terceros, con la adición del paquete de calidad de servicio implementado continua permitiendo una total interoperabilidad debido a que utiliza los Servicios Web, tecnología basada en la transmisión de texto, independiente del lenguaje de programación y del sistema operativo en el que se desarrolle el cliente o tercero.

RECOMENDACIONES

- Utilizar las modificaciones y funcionalidades adicionadas al Servidor de Comunicación con Terceros del Guardián del ALBA en el SCADA-UX realizado en el CEDIN.
- Modificar el mecanismo de manejo de excepciones en el Subsistema de Comunicación con Terceros para que se adecue a los requerimientos de mensajería confiable.

Realizar un estudio de mecanismos de optimización de *WS-Security* para su aplicación en un futuro en el Servidor de Comunicación con Terceros.

Referencias Bibliográficas.

1. **ROMERO, JOHN JAIRO MEDEZ.** ANALISIS COMPARATIVO DE LAS PLATAFORMAS J2EE Y .NET APLICADO AL DESARROLLO DE WEB SERVICES. [En línea] NOVIEMBRE de 2008. [Citado el: 25 de 11 de 2010.] http://www.scribd.com/doc/12759942/ANALISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET-APLICADO-AL-DESARROLLO-DE-WEB-SERVICES.

2. SCADA. [En línea] 2 de marzo de 2006. http://www.automatas.org/redes/scadas.htm.

3. **José Antonio Aragón Cáceres, Beatriz Llanes Jiménez.** Biblioteca. [En línea] Mayo de 2009. [Citado el: 16 de noviembre de 2010.] http://bibliodoc.uci.cu/TD/TD_2554_09.pdf

4. **Comparative of the platforms J2EE and .NET for development of Web Services. 2011.** Scribd. [Online] 2011. http://www.scribd.com/doc/12786595/Articulo-ANALISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET-.

5. **Isaza E., Gustavo A.** *Estándares de seguridad basados en XML.* madrid : s.n., 2007. 2.

6. IEEE Xplore Digital Library. [Online] 6, 2002. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1067740&abstractAccess=no&userType=inst.

7. **C. Gutiérrez, E. Fernández-Medina,M. Piattini.** *Seguridad en Servicios Web.* La Mancha, España : Universidad de Castilla-La Mancha, 2005. DIAB-05-01-2.

8. **GALÁN, LUIS MENGUAL.** *ARQUITECTURAS DE SEGURIDAD.* Madrid, España : s.n., 2010.

9. **indaganda.** INDAGANDA. [En línea] 2011. http://www.indaganda.com/definir/token.

10. **MIGUEL GARCIA FEAL, JOSE LUIS CHOUCIÑO FERREIRO. 2008.** *Seguridad en Internet SSl.* 2008.

11. **IBM,Dennis Sosnoski.** developerWorks. [En línea] 7 de 7 de 2009. [Citado el: 15 de 11 de 2010.] http://www.ibm.com/developerworks/ssa/library/j-jws6/index.html.

12. **Software de Supervisión y Control. GRAELLS, JORDI AYZA. 2003**. s.l. : Revista Automática e Instrumentación, 2003, Vol. 344.

13**. 2005.** Blogia. [Online] 2005. http://curso_sin2.blogia.com/2005/072002-secure-socket-layer-capa-de-conexion-segura-ssl-.-por-nelson-guillen.php.

Bibliografía Consultada.

1. **W3C.** QoS for Web Services: Requirements and Possible Approaches. [En línea] W3C, 25 de 11 de 2003. [Citado el: 20 de 11 de 2010.] http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

2. **OASIS.** OASIS Security Services (SAML) TC. [Online] 2011. [Cited: 1 9, 2011.] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

3. **Robin Cover, OASIS.** Security Assertion Markup Language (SAML). [Online] 2 23, 2010. [Cited: 12 1, 2010.] http://xml.coverpages.org/saml.html.

4. **BEA Systems, IBM, Microsoft Corporation.** *Web Services Reliable Messaging Protocol.* 2005.

5. **OASIS Standard.** OASIS. [En línea] 15 de noviembre de 1004. [Citado el: 11 de 1 de 2011.] http://docs.oasis-open.org/wsrm/ws-reliability/v1.1.

6. **Microsoft.** WS-ReliableMessaging Specification Index Page. [En línea] 2011. http://msdn.microsoft.com/en-us/library/ms951271.aspx.

7. **Pat Romanski, Udayan Banerjee, Liz McMillan, Elizabeth White, Salvatore Genovese.** SOAWorldMagazine. [Online] 2011. [Cited: 1 9, 2011.] http://soa.sys-con.com/node/39769.

8. **Attapattu, Rajith.** redhat. [Online] septiembre 28, 2006. http://www.redhat.com/magazine/023sep06/features/web_services/?intcmp=bcm_edmsept_007.

9.El reto de los servicios Web para el software libre. **J. J. Domínguez Jiménez, A. Estero Botaro, I. Medina Bulo, M. Palomo Duarte y F. Palomo Lozano. 2008**. 2008.

10.**BOOCH, GRADY, JACOBSON, IVAR, RUMBAUGH, JAMES. 2007**. El Lenguaje Unificado de Modelado. Manual de Referencia. s.l. : ADDISON-WESLEY, 2007. 978-84-7829-087-1.

11. **2005.** Blogia. [En línea] 2005. http://curso_sin2.blogia.com/2005/072002-secure-socket-layer-capa-de-conexion-segura-ssl-.-por-nelson-guillen.php.

12. Colsa, Luis Enrique Corredera de. 2005. *Seguridad en XML.* 2005.

13. Comparative of the platforms J2EE and .NET for development of Web Services. 2011. Scribd. [En línea] 2011. http://www.scribd.com/doc/12786595/Articulo-ANALISIS-COMPARATIVO-DE-LAS-PLATAFORMAS-J2EE-Y-NET-.

14. 2008. kioskea.net. [En línea] octubre de 2008. http://es.kioskea.net/contents/crypto/ssl.php3.

15. MIGUEL GARCIA FEAL, JOSE LUIS CHOUCIÑO FERREIRO. 2008. *Seguridad en Internet SSl.* 2008.

16. Pedro J. Álvarez, José Ángel Bañares. 2004. *Estándares relacionados con la Coordinación de Servicios.* Berlin : s.n., 2004.

Anexos:

Anexo I: Matriz de Decisión elaborada para la selección del estándar de Confiabilidad a utilizar en el Subsistema de Comunicación con Terceros del sistema SCADA Guardián del ALBA.

La matriz de decisiones permite la toma de decisiones durante la fase de selección de estándares.

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

En resumen, una matriz de decisión es una herramienta de decisiones utilizada por los desarrolladores de software como parte de sus herramientas a la hora de seleccionar un estándar a utilizar.

Elementos de la Matriz de Decisión:

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

Criterios:	Opciones:	Importancia:	Puntajes:
Desarrollar una jerarquía de criterios de decisión, también conocido como modelo de decisión.	Determinar las opciones, también llamado soluciones o alternativas.	Asignar un valor a cada criterio sobre la base de su importancia en la decisión final, generalmente es un valor comprendido entre 1 y 20.	Tasa de cada opción en una relación de escala mediante la calificación a cada criterio.

Tabla 12: Elementos de la matriz de decisión.

Como calificar una opción?	
Calificación	**Descripción**
0	No Aceptable
1	Poco Aceptable
2	Aceptable
3	Sobresaliente
4	Exelente

Tabla 13: Valores cuantitativos de las calificaciones a los estándares de confiabilidad.

		ALTERNATIVAS			
Modelo de Decisión		WS-Reliability		WS-ReliableMessaging	
Criterios	Importancia	Calificación	Puntaje	Calificación	Puntaje
Facilidad de implementación	10	2	20	2	20
Documentación disponible	5	1	5	2	10
Integración con gSOAP	20	1	20	4	80
Soporte	13	2	26	3	39
Rendimiento	15	2	30	2	30
Total	63	8	101	13	179
Puntaje = Valor * Importancia					
Importancia = Rango entre 1-20					

Tabla 14: Modelo de decisión de la matriz de selección de estándares de confiabilidad para la implementación en el Subsistema de Comunicación con Terceros.

Anexo II: Pasos para crear certificados auto firmado con OpenSSL.

1. Ejecutar el script **root.sh** pasarle como argumento el fichero de configuración de openssl denominado openssl.cnf.

```
$ sh root.sh openssl.cnf
* Create the self-signed root CA and root's certificate: root.pem cacert.pem
* Distribute the cacert.pem to clients to authenticate your server
* Enter a (new) secret pass phrase when requested
* Enter it again when prompted
* You need the pass phrase later to sign the client and server key files
* Enter your company name as the Common Name (e.g. genivia.com) when requested
* The root CA will expire after three years (1095 days)
Generating a 1024 bit RSA private key
.....++++++
writing new private key to 'rootkey.pem'
Enter PEM pass phrase: Contraseña de la llave privada
Verifying - Enter PEM pass phrase: Verificación de la contraseña anterior
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
```

If you enter '.', the field will be left blank.

Country Name (2 letter code) [CU]: **Se escribe en dos letras el identificador del país o se utiliza el valor por defecto [CU] presionando ENTER.**
State or Province Name (full name) [Habana]: **Se escribe el nombre del estado o provincia o se utiliza el valor por defecto [Habana] presionando ENTER.**
Locality Name (eg, city) [Habana]: **Se escribe de la localidad o ciudad o se utiliza el valor por defecto [Habana] presionando ENTER.**
Organization Name (eg, company) [UCI]: **Se escribe el nombre de la empresa o se utiliza el valor por defecto [UCI] presionando ENTER.**
Common Name (eg, YOUR name) [ybrena]: **Se escribe el nombre de usuario que va a manipular la clave en el servidor de Comm3 o se utiliza el valor por defecto [ybrena] presionando ENTER.**
Email Address [ybrena@estudiantes.uci.cu]: **Se escribe la dirección de correo del usuario que va a manipular la clave en el servidor de Comm3 o se utiliza el valor por defecto [ybrena@estudiantes.uci.cu] presionando ENTER.**
Signature ok
subject=/C=CU/ST=Habana/L=Habana/O=UCI/CN=ybrena/emailAddress=ybrena@estudiantes.uci.cu
Getting Private key
Enter pass phrase for rootkey.pem:**Se repite la contraseña de la clave privada.**
subject=/C=CU/ST=Habana/L=Habana/O=UCI/CN=ybrena/emailAddress=ybrena@estudiantes.uci.cu
issuer=
/C=CU/ST=Habana/L=Habana/O=UCI/CN=ybrena/emailAddress=ybrena@estudiantes.uci.cu
notBefore=May 9 18:14:33 2011 GMT
notAfter=Jul 8 18:14:33 2014 GMT

2. Ejecutar el script **cert.sh** pasarle como argumento el fichero de configuración de openssl denominado openssl.cnf y el nombre del certificado de la siguiente manera:

   $ sh cert.sh openssl.cnf server

3. Con la ejecución de los pasos anteriores se generaron los ficheros **server.pem y cacert.pem** que corresponden al certificado y la entidad certificadora. Los ficheros deben copiarse en la dirección /etc/scadanac/comm3.d/.

El resto de los archivos generados deben estar almacenados en un directorio con permisos limitados.

De la misma manera que el servidor se procede a generar certificados para las aplicaciones externas autorizadas a comunicarse con el SCADA GALBA.

GLOSARIO

ACK: en comunicaciones entre computadores, es un mensaje que se envía para confirmar que un mensaje o un conjunto de mensajes han llegado. Si el terminal de destino tiene capacidad para detectar errores, el significado de ACK es "ha llegado y además ha llegado correctamente".

IBM: International Business Machines, es una empresa multinacional estadounidense que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.

OSI: Open system interconnection es un modelo de interconexión de sistemas abiertos.
Es un modelo de red descriptivo creado por la Organización Internacional para la Estandarización en el año 1984. Es decir, es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones.

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

RC4: El RC4 es un algoritmo de cifrado en flujo creado en secreto por Ron Rivest en 1987 para su compañía RSA (Rivest-Shamir-Adleman). Es usado en varios productos comerciales como, Secure Sockets Layer (SSL).

IDEA: El IDEA (Algoritmo Internacional de Cifrado de Datos) es un algoritmo de encriptado de clave secreta diseñado por los suizos, utiliza una clave de 128 bits, lo que por el momento lo hace inmune a los ataques de fuerza bruta así como al criptoanálisis diferencial para su desencriptado.

RSA: (Rivest, Shamir y Adleman) es un sistema criptográfico de clave pública desarrollado en 1977. En la actualidad, RSA es el primer y más utilizado algoritmo de este tipo y es válido tanto para cifrar como para firmar digitalmente.

MD5: (Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un algoritmo de reducción criptográfico de 128 bits se utilizan extensamente en el mundo del software para proporcionar la seguridad de que un archivo descargado de internet no se ha alterado.