UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

DESARROLLO DE FUNCIONALIDADES AL MÓDULO DE PROCESAMIENTO DEL SCADA UX

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Juan Carlos González Tamayo

Tutor: Ing. Ana Silvia Tellería Martínez

Junio 2011, Ciudad de La Habana

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis 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 _____.

Juan Carlos González Tamayo	Ing. Ana Silvia Tellería Martínez
Autor	Tutor

DATOS DE CONTACTO

Tutor: Ing. Ana Silvia Tellería Martínez

Edad: 27

Ciudadanía: Cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas

Categoría Docente: Instructor

E-mail: atelleria@uci.cu

Graduado de la UCI, con 5 años de experiencia en el tema de los sistemas SCADA.

DEDICATORIA

Dedicado especialmente a mi señora madre Zenaida Tamayo Torres quién estuvo conmigo cuando me anunciaban oficialmente mi entrada a la UCI, más no pudo llegar a hoy para verme ingeniero. Este logro es para ti mamá.

A mi padre Juan González quién ha sido padre y madre a la vez en los últimos 3 años.

A mi abuela Emiliana, madre de la familia.

A mi hermana Mirelis y a mi sobrino Alejandro.

A primo Roly y tía Eugenia.

A mis amigos de Las Tunas en especial a Carlos Jorge, Ariel, Oscar, Hernán, Andy, Jorge, Reinier, Raúl, a todos.

AGRADECIMIENTOS

A mis primos, tías y familia en general.

A mis amigos, todos. Desde los que conozco de primer año, hasta los que he conocido en el camino de la docencia, la producción, la FEU y en la vida de la UCI, en especial a Alexis, Calviño, Evelio, Alian, Adruban, Luis Mariano, Liván y su gente, a Liusba, Irenna, a la mulatísima Dayanis, a Yurisel, Dunia, Daisy, a Sandra, a Libeidy, Liudmila, Yoana, a Eva, Osvaldo y a todos los que aunque no mencione son mis amigos, gracias.

A mi tutora Ana Silvia quien me guió sin descanso en la realización de la tesis.

Al equipo de desarrollo con el que trabajo.

A la UCI y a la Revolución que la hizo posible aunque no perfectas, si perfectible.

A los muertos de mi felicidad.

RESUMEN

El sistema de supervisión y control SCADA UX, desarrollado en el Centro de Informática Industrial es un sistema distribuido compuesto por varios módulos, entre los que se encuentra el módulo de procesamiento. El SCADA UX no contaba con las funcionalidades necesarias para su empleo en el proyecto de acueductos que se construye en la ciudad de Santiago de Cuba. Carecía de mecanismos para la notificación de algunos tipos de situaciones excepcionales. No permitía procesar comandos de modificación de variables y operación de alarmas. Empleaba una representación poco coherente para el manejo de valores discretos del proceso bajo supervisión. Parte importante de estas carencias constituyen responsabilidad del módulo de procesamiento.

El presente trabajo se plantea como objetivo diseñar e implementar las funcionalidades necesarias en el módulo de procesamiento que posibiliten su aplicación como parte del SCADA UX en el proyecto de acueductos de la Ciudad Héroe.

Palabras clave: Alarmas, Comandos, Procesamiento, Punto Digital, SCADA, Supervisión

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

INTRODUCCIÓN

“La Informática Industrial surge de la necesidad por parte del hombre de automatizar ciertos procesos industriales, los cuales son más efectivos si se realizan mediante procesos automáticos que si son realizados por la mano del hombre, ya que este tipo de tareas requiere de precisión y exactitud. La gran mayoría de estos procesos tienen como objetivo contribuir con la industrialización, por ejemplo un proceso de fabricación de automóviles, donde robots programados realizan la mayor parte del trabajo, sin embargo existe la intervención de las personas en algún momento.” *(1)*

Dentro de las aplicaciones de informática industrial se tienen a los Sistemas de Supervisión, Control y Adquisición de Datos (SCADA), los cuales son aplicaciones de software diseñadas especialmente para controlar procesos productivos automatizados. Los SCADA proporcionan comunicación con dispositivos de campo encargados de obtener las variables del proceso, procesarlas y mostrarlas en la pantalla de un ordenador. *(2)*

En el Centro de Informática Industrial (CEDIN), perteneciente a la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), se desarrolla un sistema SCADA a partir de los requerimientos funcionales de un conjunto de empresas cubanas. Para la implementación de este SCADA, el cual es denominado en su versión genérica como SCADA UX se emplean únicamente tecnologías software libre, constituyendo una alternativa a las soluciones privativas y un paso más en el camino a la soberanía tecnológica en correspondencia con las normativas de CEDIN y la UCI que son promovidas también en el ámbito nacional.

Dicho sistema se desarrolla empleando una arquitectura distribuida. La cual consta de una serie de módulos interconectados a través de una capa de comunicaciones conocida como Middleware. Entre los módulos que lo componen se encuentran: el recolector de la información de campo, procesamiento, configuración, históricos y la interfaz hombre – máquina o HMI por sus siglas en inglés. *(3)*

El módulo de procesamiento tiene como funciones principales el procesamiento y publicación de información extraída de los dispositivos de campo que le llega a través del módulo de recolección. Esta información se analiza y transforma según las configuraciones establecidas y se realiza la publicación de los datos resultantes utilizando el middleware.

Los dispositivos de campo ofrecen información importante del proceso o parte del proceso bajo supervisión. Pueden ser equipos de comunicación, sensores (de presión, temperaturas, caudal, etc.), actuadores (motores, manipuladores de válvula, interruptores, etc.), entre otros. *(5)*

Un punto es un valor simple de entrada o salida que es controlado o supervisado por el sistema SCADA. Es una representación del valor que ofrece un dispositivo de campo, y se obtiene realizando operaciones de lectura sobre la memoria del mismo. Son conocidos también como variables. El valor que provee un manipulador de válvula indicando el flujo que se permite pasar por una tubería sería un punto para el SCADA. Estas variables pueden ser analógicas o digitales. *(5)* Las variables analógicas pueden tomar infinitos valores entre dos valores cualesquiera de su dominio, por ejemplo la temperatura. Las variables digitales toman valores dentro de un conjunto discreto, por ejemplo una lámpara encendida o apagada o una llave cerrada o abierta. *(7)*

Las alarmas representan condiciones anormales de un sistema o de uno de sus componentes. Por ejemplo, pudieran ser situaciones anómalas cuando el sistema detecta que no puede comunicarse con uno de los dispositivos de campo (Alarma de Falla de Comunicación). O cuando un tanque de combustible se llena, indicando que hay que cerrar la válvula que lo llena (Alarmas de Nivel). Las circunstancias mencionadas son comúnmente situaciones de alarma pero pueden no serlo en dependencia del proceso bajo supervisión. *(8)*

Entre los elementos que condicionan el alto impacto del SCADA UX se encuentra los despliegues que se llevan a cabo en las instalaciones relacionadas al sistema de acueductos de Santiago de Cuba, proyecto impulsado por la Revolución y que pretende entre otras metas llevar agua a miles de familias que prescinden de tan preciado tesoro. Este despliegue es denominado SCADA Aguas Santiago(AS).

El módulo de procesamiento del SCADA UX actual no podía ser utilizado en el Acueducto de Santiago de Cuba debido a que no soportaba aún la notificación de algunas situaciones anómalas que suelen presentarse en estos sistemas, como el fallo al ejecutar un comando de escritura o cuando un punto toma un valor que no es el establecido por los operarios del sistema. Tampoco les permitía forzar el valor de un punto a cambiar a un valor especificado por ellos, en algunos casos para realizar una acción correctiva al ser avisados sobre desvíos en el proceso. Además, eran mejorables los mecanismos para la

representación de valores de puntos correspondientes a dispositivos en los que solo es posible tomar un valor dentro de un conjunto discreto; por ejemplo una lámpara que puede estar encendida o apagada. Realizando una representación más coherente se contribuye con la usabilidad y eficiencia del sistema, entre otros parámetros de calidad de software.

¿Cómo obtener una versión del módulo de procesamiento del SCADA UX aplicable al proyecto de Acueductos de Santiago de Cuba?

El **objetivo general** del presente trabajo es desarrollar en el módulo de procesamiento del SCADA UX las funcionalidades de procesamiento de puntos digitales, alarmas de falla de ejecución de comando, alarmas de cambio de estado no comandado, alarmas de estado, comandos de escritura de punto y comandos de operación de alarmas.

Para solucionar el problema se plantea como **Objeto de Investigación** los sistemas SCADA y dentro de esta área de conocimientos se propone como **Campo de Acción** el procesamiento de comandos, alarmas y puntos digitales.

Para lograr los objetivos trazados se plantean las siguientes **Tareas de Investigación**:

- Definir los principales conceptos asociados al módulo de procesamiento para su utilización como base de la investigación a desarrollar.
- Realizar un análisis de la arquitectura, código y tecnologías empleadas en el módulo de procesamiento del SCADA Guardián del Alba (GALBA) y en otros SCADA, haciendo énfasis en las características comunes al SCADA UX con el objetivo de identificar técnicas que puedan ser empleadas en el desarrollo de las nuevas funcionalidades.
- Establecer y describir los requisitos funcionales y no funcionales que guíen el desarrollo a realizar en el módulo de procesamiento a partir de la problemática planteada.
- Diseñar las clases y relaciones necesarias para la incorporación de las funcionalidades identificadas sobre la arquitectura y diseño existentes.

- Implementar los nuevos requisitos funcionales previamente identificados del módulo de procesamiento, respondiendo tanto al diseño establecido como a los requisitos no funcionales que impacten directamente en el desarrollo.
- Diseñar y aplicar pruebas a la nueva versión del módulo de procesamiento para la validación del cumplimiento de los requerimientos establecidos.

Para la realización de la investigación se hizo necesario aplicar algunos **métodos de investigación científicos**.

De los métodos científicos teóricos se emplearon el **Analítico-Sintético** y la **Modelación.**

De los métodos científicos empíricos se emplearon las **Entrevistas** y el **Experimento**.

El método **Analítico-Sintético** se emplea para sintetizar y analizar toda la información adquirida en el proceso de investigación y a partir de los conceptos recopilados generar nuevos conceptos e ideas que sustenten la propuesta de solución presentada.

El método **Modelación** se emplea en el diseño de clases y algoritmos que intervienen en la solución de problema planteado permitiendo reproducirlos mediante diagramas, tablas y nuevos conceptos.

Las **Entrevistas** se realizan a asesores, arquitectos y desarrolladores pertenecientes al CEDIN. De estas se obtendrá toda la información posible sobre la experiencia previa en la implementación de un módulo de procesamiento.

El **Experimento** se utiliza para realizar pruebas, aislándose las funcionalidades a probar, y creando y modificando circunstancias propicias donde se puedan demostrar el correcto funcionamiento del procesamiento implementado en el presente trabajo. *(9)*

El presente trabajo está desarrollado con la siguiente estructura:

En el **Capítulo1 Procesamiento y Sistemas SCADA** se definen los principales conceptos asociados al estudio de los sistemas SCADA y al desarrollo del módulo de procesamiento entre otros elementos necesarios para describir la propuesta de solución.

En el **Capítulo 2 Análisis y Diseño** se identifican los requisitos funcionales y no funcionales para dar solución el problema planteado y se realiza el diseño de las clases que modelarán las nuevas funcionalidades.

En el **Capítulo 3 Implementación y Pruebas** se dan detalles sobre el proceso de implementación y se abordan los distintos casos de prueba diseñados y aplicados a la versión resultante del módulo de procesamiento.

CAPÍTULO 1: PROCESAMIENTO Y SISTEMAS SCADA

1.1 Introducción

En este capítulo se abordan conceptos esenciales relacionados con los sistemas SCADA haciendo énfasis en los elementos afines al módulo de procesamiento, como lo son el tratamiento de puntos, alarmas y comandos. De los SCADA se describirán sus usos y características relevantes y del módulo de procesamiento sus responsabilidades dentro del sistema SCADA UX.

Esta fundamentación teórica servirá como sustento a la comprensión del problema a resolver.

1.2 Procesamiento

Procesamiento es la aplicación sistemática de una serie de operaciones sobre un conjunto de datos, generalmente por medio de máquinas, para explotar la información que estos datos representan. *(10)*

En el caso de los sistemas SCADA este procesamiento se realiza sobre los puntos, alarmas y comandos que se obtienen o generan durante la ejecución del sistema.

1.3 Sistemas SCADA

Los sistemas SCADA son aplicaciones de software diseñadas especialmente para controlar una gran variedad de procesos industriales, proporcionando comunicación con los dispositivos de campo y controlando las variables y los procesos involucrados en la producción desde la pantalla de un ordenador. Además provee toda la información que se genera en el proceso productivo a diferentes usuarios: supervisores de control de calidad, mantenedores, asesores, operadores y otros. *(2)*

Los datos precisos y oportunos permiten la optimización de la operación de una planta y sus procesos. Un beneficio adicional es la obtención de mayor eficiencia, confiabilidad y lo más importante es que se logran realizar operaciones seguras. Todo esto trae como resultado una disminución de los costos en comparación con los primeros sistemas no automatizados. *(11)*

Estos sistemas son empleados en un amplio abanico de procesos industriales, por ejemplo la producción, refinación, manufactura, fabricación y generación de energía. También son utilizados en procesos de infraestructuras públicas como el tratamiento y distribución de agua, recogida y tratamiento de aguas

residuales, distribución y trasmisión de energía, gaseoductos y oleoductos, sistemas de sirenas de la defensa civil y en los grandes sistemas de comunicación. También se usan en estaciones espaciales, aeropuertos, barcos, edificios, etc. *(6)*

1.4 Sistemas SCADA existentes

En el mundo de los sistemas SCADA se encuentran una gran variedad de aplicaciones. Desde grandes sistemas privativos, muchos de los cuales se comercializan como soluciones completas acompañadas del hardware de supervisión y control hasta no menos importantes soluciones de código abierto que generalmente trabajan con hardware de un sinnúmero de fabricantes los que los hace en algunos casos más interoperables. Las plataformas más comunes sobre las que corren estos sistemas son Windows y GNU/Linux.

1.4.1 Sistemas SCADA privativos

Entre los sistemas SCADA más avanzados y reconocidos a nivel Internacional se pueden mencionar el Simantic WinCC (Windows Control Center), el InTouch HMI/SCADA, el Citect SCADA y el SCADA Pro de MeasureSoft.

InTouch es un SCADA de la rama Wonderware de la división de Administración de Operaciones de la compañía inglesa Invensys. Permite configurar alarmas y establecerles hasta 999 niveles de prioridad y hasta 8 niveles de jerarquía entre grupos de alarma con posibilidad de hasta 16 subgrupos para cada uno de ellos. Posibilita visualizar todas o un extracto de ellas de forma histórica y grabar en disco o imprimir en diferentes formatos personalizables. Las funciones de alarmas distribuidas incluyen reconocimiento global o selectivo, desplazamiento por la lista y visualización de alarmas procedentes de diferentes servidores en un único panel. *(12)*

ScadaPro de la compañía irlandesa Measurement Development Ltd. por su parte incluye un componente de Procesamiento y Gestión de alarmas cuyas características más importantes en el procesamiento de alarmas son:

- A cada canal se le pueden asignar condiciones de alarmas únicas para límites alto y bajo.

- Se evitan la ocurrencia de múltiples alarmas en un mismo canal empleando histéresis y retardo de alarmas.
- A cada condición de alarma se le puede asignar una prioridad en un rango de 1 a 255 y se le pueden asociar bloques de texto que son mostrados en caso de activarse la alarma.
- Varios canales de alarmas pueden estar vinculados a un canal de salida de alarmas común para propósitos de notificación, o para apagar partes importantes del proceso o la planta de forma automática.
- Las alarmas pueden ser reconocidas de forma independiente o en grupo. *(13)*

Estos sistemas tienen en común que son software privativo, altamente costosos y solo se ejecutan sobre plataforma Windows.

En el país también se han desarrollado algunos SCADA privativos como el Guardián del Alba (GALBA), el TELENUL y el EROS.

El Guardián del Alba (Galba) es desarrollado en Venezuela por un equipo compuesto por personal tanto venezolano como cubano, en el cual toman parte estudiantes y profesores del CEDIN.

Es un SCADA distribuido, comprendido por varios módulos. *(13)* A partir de un análisis del sistema en ejecución y su código fuente se comprobaron sus funcionalidades. Realiza procesamiento de puntos recolectados y calculados, tanto analógicos como digitales. Detecta la presencia de condiciones anormales y las notifica al HMI. Soporta tratamiento para los siguientes tipos de alarmas: nivel, tasa de cambio, falla de instrumento, falla de comunicación, falla de no variación en tiempo y cambio de estado. El sistema procesa comandos de escritura de puntos y de manipulación de alarmas.

Se ejecuta sobre Debian GNU/Linux versión 5 Lenny. *(13)*

SCADA Telenul es desarrollado por la Empresa de Ingeniería y Proyectos de la Electricidad (INEL) para el control de recerradores y seccionalizadores automáticos de industrias NULEC en la Unión Nacional Eléctrica. Permite telecontrolar y supervisar de forma remota los interruptores NULEC instalados en sus subestaciones y redes de distribución. El sistema corre sobre Windows y está programado en Delphi. *(14)*

EROS es un Sistema de Supervisión y Control de Procesos Industriales de la Empresa de Servicios de Computación, Comunicaciones y Electrónica del Níquel “Rafael Fausto Orejón Forment” (SERCONI) en Nicaro, Mayarí, provincia Holguín. Realiza variadas funciones dentro del entorno de la dirección de los procesos. Puede trabajar acoplado a diversos sistemas de colección de datos, como elemento único o formando parte de una red industrial. Tiene en cuenta todas las características de las variables medidas y permite realizar tratamiento estadístico y determinístico de las mismas. Permite la ejecución de comandos para el control manual y para realizar cambios de parámetros y acción de los reguladores. Su aplicación más importante es su uso en la vigilancia en tiempo real desde el Despacho Nacional de la Unión Eléctrica de la energía que consume cada uno de los 169 municipios del país. *(15)*

Estos sistemas al ser privativos no es posible tener acceso a su código fuente para ser estudiados por el personal del centro para tomar experiencia y verterla en nuestros desarrollos enfocados a la industria cubana. Solamente se tiene acceso al código del SCADA GALBA, el cual fue desarrollado en conjunto entre el CEDIN y PDVSA. Hay que mencionar además sus altos precios de adquisición, en especial los grandes SCADA internacionales que en muchas ocasiones vienen acompañados del hardware lo que los hace aún más prohibitivos para países sin un alto desarrollo económico como Cuba. El empleo de estos sistemas limita la soberanía tecnológica del país debido a que se deja el control de los procesos industriales claves en manos de un software desarrollado en muchos casos en países ideológicamente contrarios al proceso revolucionario cubano.

1.4.2 Sistemas SCADA software libre

Dentro de los sistemas SCADA con licencias de software libre y de código abierto se puede citar al SCADA Argos.

SCADA Argos es desarrollado por la Gerencia de Investigación y Desarrollo CVG Venalum en Venezuela.

“El SCADA está implementado en plataformas GNU/Linux y los componentes de software han sido desarrollados usando el lenguaje de programación C/C++, se utiliza XML como formato para los archivos de configuración, además de varias utilidades y librerías de software libre.” *(16)*

Los SCADAS software libre nos permiten a diferencia de los SCADA privativos el acceso a su código fuente para su estudio, modificación y reutilización.

1.5 SCADA UX desarrollado en el CEDIN

El SCADA UX es un sistema que integra funcionalidades que permiten la solución de aplicaciones de supervisión y control de procesos, utilizando para ello una arquitectura distribuida de módulos que permiten realizar el procesamiento en varios nodos independientes.

Se les llama módulos clientes a aquellos que para realizar su trabajo necesitan la información que brinda el núcleo de adquisición y procesamiento del SCADA. Algunos módulos clientes son el HMI e históricos.

El SCADA UX está compuesto por los siguientes módulos:

-Recolección: Tiene como función la adquisición de datos de los dispositivos de campo, en él se efectúa el manejo de los hilos requeridos para la adquisición y la comunicación con el módulo de procesamiento así como con los manejadores(drivers) de los dispositivos.

-Módulo de procesamiento: Presenta como principales características la recepción de datos provenientes del módulo de recolección, detección y manejo de alarmas y publicación a los clientes de la actualización de los puntos y la sucesión de alarmas y eventos.

-Middleware: Capa de comunicaciones que ofrece un conjunto de servicios a los módulos del SCADA para el envío asíncrono de la información relacionada con puntos, alarmas y comandos. Para ello hace empleo del paradigma publicación/suscripción haciendo posible el funcionamiento del SCADA en una arquitecta distribuida.

-HMI: módulo de visualización empleado para la interacción entre los operarios y el sistema. Muestra de una forma amena la información obtenida tras su adquisición y procesamiento. Posibilita la realización de acciones del operador como escritura de puntos y operación de alarmas.

-Históricos: se encarga de gestionar el almacenamiento ordenado de los datos generados en el SCADA permitiendo su posterior consulta por el propio SCADA u otras aplicaciones. *(17)*

El módulo de procesamiento interactúa fundamentalmente con el módulo de recolección para la adquisición de los datos provenientes del nivel de campo y con el middleware para la comunicación con el resto de los módulos. *(3)*

El módulo de recolección es el responsable de adquirir la información del nivel de campo empleando una representación interna de los puntos agrupados según su frecuencia de lectura configurada, para las operaciones de lectura y escritura de puntos hace uso de los manejadores (drivers). Interactúa directamente con el módulo de procesamiento, al cual le envía los datos recolectados para ser procesados. *(18)*

El Middleware se emplea para lograr la arquitectura distribuida que permite compartir el trabajo del sistema SCADA entre los módulos ya mencionados, ubicados en distintos servidores de la red. Emplea el paradigma de comunicación distribuida Publicador/Suscriptor.

"El paradigma **Publicador/Suscriptor** se utiliza para el intercambio de datos asíncrono y de alta frecuencia. Brinda interfaces de comunicación a los siguientes módulos: Procesamiento (PM), Visualización y Base de Datos Históricos (BDH). Esta versión implementa solo el intercambio de datos Push; el Publicador entrega los datos al canal de eventos y éste se los entrega al Consumidor. Las interfaces consisten en funciones de llamadas de retorno (callbacks), que implementan métodos para el envío y recepción de datos." *(3)*

1.5.1 Arquitectura del SCADA UX

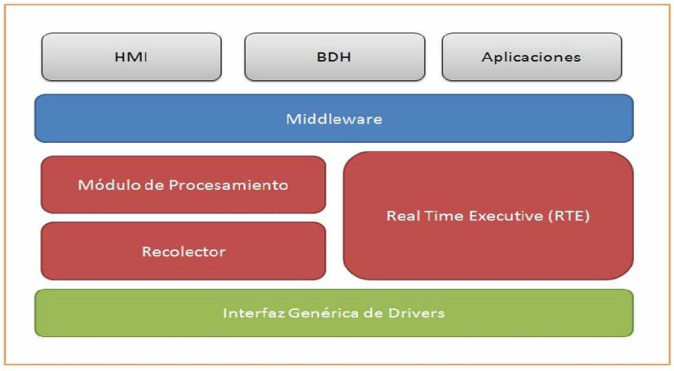

Ilustración 1 Arquitectura del SCADA UX

1.6 Módulo de procesamiento del SCADA UX

El módulo de procesamiento se encuentra en el nivel de Entrada-Salida. Almacena temporalmente la información adquirida por el módulo de recolección (valores, dirección, estados, calidad y marcas de tiempo de variables de proceso). Esta información es empleada en la detección de alarmas. Dota de un mecanismo de interfaz que provee una semántica para la interconexión entre el nivel de entrada salida y los niveles superiores a través del Middleware.

Es el encargado de manejar todo lo referente a la recepción, procesamiento y distribución de los datos provenientes del campo. Permite la ejecución de acciones como:

- Recepción de datos provenientes del módulo de recolección.
- Conversión lineal entre escalas de los datos recolectados.
- Procesamiento de variables calculadas.
- Detección y manejo de alarmas.
- Control de calidad de los datos recolectados.
- Publicación a los clientes la actualización de los puntos, sucesión de alarmas y eventos.
- Propagación a los niveles inferiores de los comandos enviados por los operadores o sistemas automatizados de control de procesos. *(3)*

1.6.1 Procesamiento de Alarmas

Las alarmas se almacenan en listas que se encuentran asociadas al punto al cual pertenecen. Las alarmas cuando se activan pasan a un espacio de memoria reservado conocido como sumario de alarmas, del cual no salen hasta tanto no se desactiven y sean reconocidas por los operadores del SCADA. *(19)*

Una alarma es activada cuando se cumple su condición de activación configurada. Su procesamiento consiste en pasarla al sumario de alarmas y publicarla en el Middleware, de manera que el HMI la reciba y

la muestre en pantalla a los operadores del proceso. En el caso contrario, una alarma se desactiva cuando se deja de cumplir la condición de activación configurada para la misma, estableciéndose el estado de la alarma a desactivada y se publica para que el HMI informe a los operadores que la situación no deseada dejó de existir. Un ejemplo de activación de alarma es cuando se vacía un tanque de combustible que siempre debe tener cierta capacidad ocupada, se activa una alarma que informa a los operadores de dicha situación los cuales en dependencia de las circunstancias específicas pueden abrir el abastecimiento de combustible para llenar el tanque. Una vez que el tanque deja de estar vacío esta alarma se desactivaría.

El módulo de procesamiento está dotado de un mecanismo de detección de alarmas asociadas a las variables almacenadas. Estas alarmas se procesan cada vez que llega un nuevo valor del punto comprobándose si este nuevo estado las activa (si se encuentra desactivada) o desactiva (si se encuentra activada). Se gestionan actualmente alarmas de nivel, tasa de cambio y falla de comunicación.

1.6.2 Alarmas presentes en el SCADA UX

Alarmas de nivel: Son las encargadas de verificar que los valores de los puntos analógicos se encuentren dentro de un rango de operación preestablecido.

Alarmas de Tasa de Cambio: La alarma de tasa de cambio monitorea que el cambio, en un tiempo determinado, no supere un valor previamente configurado:

La tasa de cambio se emplea usando la siguiente fórmula en la que variable.PVt1 es el valor actual del punto, variable.PVt0 es el valor del punto en la lectura anterior, variable.PV.timet1 es la estampa de tiempo actual y variable.PV.timet0 la estampa de tiempo de la lectura anterior.

Tasa de cambio = abs ((variable.PVt1 - variable.PVt0)/ (variable.PV.timet1-variable.PV.timet0)).

Alarmas de Falla de Comunicación: Esta alarma se procesa teniendo en cuenta el estado de las comunicaciones con los dispositivos del sistema. Una vez que se detecta que no es posible establecer comunicación con un dispositivo determinado se genera una alarma de este tipo.

1.6.3 Comandos presentes en el SCADA UX

Un comando es una instrucción u orden que el usuario proporciona a un sistema informático. En los sistemas SCADA los comandos son enviados por los módulos clientes al módulo de procesamiento. Este los recibe y procesa, ofreciendo una respuesta a los clientes indicando acuse de recibo.

Esto es posible debido a una interfaz que se implementa para la recepción de comandos y envío de respuestas de comandos. Lo que permite la interacción entre los módulos del SCADA mediante órdenes generadas ya sea por el propio sistema o por los operarios desde el HMI.

Actualmente se procesan comandos de sincronización y reconocimiento de alarmas. Los comandos de sincronización son usados para solicitar al módulo de procesamiento que publique el sumario de alarmas completo y los de reconocimiento de alarmas se emplean para notificar al sistema que ya el operador es consciente de la ocurrencia de una alarma en específico.

(3)

CAPÍTULO 2 - ANÁLISIS Y DISEÑO

2.1 Introducción

En este capítulo se expone el flujo de trabajo Análisis y Diseño realizado para modelar y dar respuesta al problema planteado. Se escogió realizar un Modelo de Dominio debido a que se adecúa mejor para la representación de conceptos, sus atributos y asociaciones con otros conceptos lo que permite modelar objetos y sus interacciones en el mundo real. Se desechó realizar Modelo de Negocio debido a que el módulo de procesamiento no interactúa directamente con personal humano.

2.2 Modelo de Dominio

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

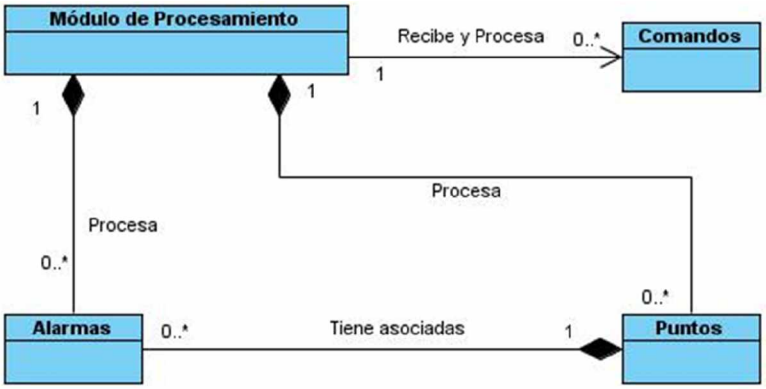

Ilustración 2 Modelo de Dominio

El **módulo de procesamiento** se encarga de procesar la información de los puntos provenientes del Recolector, así como las alarmas asociados a estos. También recibe comandos para realizar acciones concretas como sincronizar sumario de alarmas y reconocer alarmas.

Los **puntos** son la representación que emplea el sistema para procesar los valores que ofrecen los dispositivos de campo. Los puntos pueden ser analógicos o digitales.

Las **alarmas** son la representación utilizada para la notificación de comportamientos no deseados en el sistema.

Los **comandos** son el mecanismo que emplea el SCADA para emitir órdenes al módulo de procesamiento.

2.3 Captura de Requisitos

Se hace necesario realizar una captura de requisitos que recoja de forma coherente los nuevos requerimientos para el módulo de procesamiento. Estos requisitos tienen como objetivo definir qué funcionalidades hacen falta para para dar solución a los mismos.

2.3.1 Requisitos Funcionales

Los requisitos funcionales de un sistema describen lo que el sistema debe hacer. Definen con detalle la función del sistema, sus entradas, salidas, etc. *(20)*

Se identificaron como requisitos funcionales:

RF1. Procesar puntos digitales.

RF2. Procesar alarma de estado.

RF3. Procesar alarma de cambio de estado no comandado.

RF4. Procesar comando de escritura.

RF5. Procesar alarma de falla de ejecución de comando.

RF6. Procesar comandos de operación de alarmas: inhibir, desinhibir y reiniciar.

2.3.2 Requisitos No Funcionales

Los requisitos no funcionales son aquellos que se refieren a las propiedades emergentes de éste como fiabilidad, tiempo de respuesta, restricciones en el diseño e implementación, etc. *(20)*

Las nuevas funcionalidades que darán respuesta a los requisitos funcionales identificados deberán cumplir con los siguientes requisitos no funcionales:

Restricciones en el diseño y la implementación.

RNF1: Deben contar con un conjunto de patrones que permita la reutilización del código.

RNF2: El código debe cumplir con los estándares de codificación establecidos para el desarrollo del módulo de procesamiento.

RNF3: Deben mantener la capacidad del sistema de funcionar en los sistemas operativos Debian GNU/Linux y Ubuntu.

RNF4: Se deben implementar usando lenguaje de programación C++ en el entorno de desarrollo integrado Eclipse + Plugin C++. Herramientas estas definidas previamente para el desarrollo del Módulo.

RNF5: Se debe utilizar la metodología de desarrollo Open UP y herramienta de modelado Visual Paradigm.

Tecnologías y herramientas empleadas.

Las tecnologías y herramientas empleadas en el diseño e implementación de las nuevas funcionalidades son las definidas para el desarrollo del proyecto SCADA Aguas de Santiago. A continuación una breve descripción de cada una de ellas.

- Metodología de Desarrollo:

Open UP: Es un marco de trabajo para procesos de desarrollo de software que fue liberado por el Eclipse Process Framework (EPF). Entre sus características fundamentales se encuentran la preservación de la esencia del Proceso Unificado (Unified Process): desarrollo iterativo e incremental, desarrollo dirigido por casos de uso y centrado en la arquitectura. Open Up solo incluye lo fundamental sin dejar de ser completo y extensible (menos de 20 artefactos). *(21)*

- Herramientas de modelado:

Visual Paradigm: Es una herramienta CASE que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, generar código a partir de diagramas y generar documentación.

- Compilación (Compilador, Máquina Virtual, Intérprete):

Compilador GCC: Es una colección de compiladores del proyecto GNU para C, C++, Objective C y Fortran; es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario.

Compilador G++: Es un compilador para C++ desarrollado por el proyecto GNU. Es software libre y forma parte del GCC.

- Entornos de desarrollo integrados (IDE):

Eclipse: Es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto Eclipse llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.

- Lenguajes de programación:

Lenguaje C++: Es un lenguaje imperativo orientado a objetos derivado del C. Brinda todas las potencialidades del paradigma de programación orientado a objetos como abstracción, encapsulación, herencia y polimorfismo.

- Framework o Componentes:

Biblioteca Estándar de C++: La biblioteca estándar es una colección de clases y funciones escritas en el núcleo del lenguaje. La biblioteca estándar proporciona varios contenedores genéricos, funciones para utilizar y manipular esos contenedores, funciones objeto, cadenas y flujos genéricos (incluyendo E/S interactiva y de archivos) y soporte para la mayoría de las características del lenguaje. La biblioteca estándar de C++ también incorpora la ISO C90 biblioteca estándar de C. *(3)*

2.4 Especificación de requisitos

A continuación se realiza la especificación de los requisitos previamente identificados con el objetivo de facilitar el diseño e implementación de las funcionalidades que les deben dar solución.

Tabla 1 Especificación de Requisito Procesar Puntos Digitales.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF1.	Procesar Puntos Digitales	Una variable digital es aquella que solamente puede tomar un determinado número de valores. Las variables digitales más empleadas son las binarias, es decir aquellas que toman solamente un estado de dos posibles. Algunos ejemplos son: una lámpara encendida o apagada, una llave cerrada o abierta, un circuito con tensión o sin tensión. Estas variables se representan en el sistema con puntos digitales, cuyo valor o estado es una palabra de un máximo de 4 bits, la cual puede segmentarse de forma tal que cada segmento (uno o más bits) puede estar asociado a una dirección específica del dispositivo que lo emite. Estos puntos al igual que los analógicos pueden ser leídos y modificados con comandos para estas operaciones. Como precondiciones se tiene que el sistema debe estar iniciado a partir de un archivo XML de configuración válido generado por el configurador gráfico, listo para procesar información proveniente del Recolector y conectado al Middleware. Los puntos	Media	Alta

		digitales a procesar deben estar configurados con todos los atributos necesarios para su procesamiento. Un punto digital puede tener configuradas ninguna o varias alarmas de estado y ninguna o una alarma de cambio de estado no comandado. Estas alarmas se procesan en cada nueva lectura del punto. El requisito comienza cuando el módulo de recolección envía las lecturas de puntos digitales. El módulo de procesamiento recibe las lecturas, actualiza su representación de los puntos y procesa todas las alarmas asociadas a cada punto digital. El módulo de procesamiento publica o envía a través del Middleware los puntos digitales recibidos para uso de los restantes módulos del SCADA. Luego de ejecutado el procesamiento el sistema debe quedar de la siguiente manera: La representación de los puntos en el módulo de procesamiento queda actualizada con sus nuevos valores, y sus alarmas correspondientes activas o desactivas según su análisis. Los puntos quedan publicados en el Middleware.		

Tabla 2 Especificación de Requisito Procesar alarma de estado.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF2.	Procesar alarma de estado.	Una alarma de estado es una alarma que se activa cuando el punto digital al cual pertenece toma el estado que ella representa indicando que el dispositivo se encuentra en un estado anormal. Como precondiciones se tiene que el sistema debe estar iniciado a partir de un archivo XML de configuración válido generado por el configurador gráfico, con su representación en memoria de los puntos digitales configurados con sus respectivas alarmas de estado configuradas. El módulo debe estar listo para procesar información proveniente del recolector y conectado al Middleware. Las alarmas a procesar deben estar habilitadas y desinhibidas. Se recibe desde el recolector información de lectura de un punto digital. Se validan que el punto recibido desde el recolector exista en la colección de puntos cargada en la configuración. Se verifica si el nuevo estado del punto coincide con algún estado asociado a una alarma de estado. Si coincide el nuevo estado con una alarma se activa la alarma de estado.	Media	Alta

		Una alarma de estado se desactiva cuando el punto tiene un estado diferente al que representa la alarma. Si la alarma es activada se publica en el Middleware para uso de los restantes módulos del SCADA. La representación de las alarmas en el módulo de procesamiento queda actualizada con sus nuevos estados en caso de haber cambiado a partir de las nuevas lecturas de su punto al cual están asociadas. Las alarmas quedan publicadas en el Middleware.		

Tabla 3 Especificación de Requisito Procesar alarma de cambio de estado no comandado.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF3.	Procesar alarma de cambio de estado no comandado.	La alarma de cambio de estado no comando consiste en notificar si un punto tiene un valor el cual no fue establecido a través de un comando de escritura. El sistema debe estar iniciado a partir de un archivo XML de configuración válido generado por el configurador gráfico, con su representación en memoria de los puntos digitales configurados con sus alarmas configuradas. El módulo debe estar listo para procesar información proveniente del Recolector y conectado al Middleware. Las alarmas a procesar deben estar configuradas, habilitadas y desinhibidas. Se recibe desde el Recolector información de un	Media	Media

		punto. Se verifica que el punto mantenga el último valor que fue escrito por el sistema a través de un comando de escritura. Si hubo un cambio en el valor no coincidente con el último escrito por el sistema se activa la alarma de cambio de estado no comandado. Esta alarma no tiene condición de desactivación, solo necesita ser reconocida para ser eliminada del sumario. La alarma se publica en el Middleware para uso de los restantes módulos del SCADA. La representación de las alarmas en el módulo de procesamiento queda actualizada con sus nuevos estados en caso de haber cambiado a partir de las nuevas lecturas de su punto al cual están asociadas. Las alarmas quedan publicadas en el Middleware.		

Tabla 4 Especificación de Requisito Procesar comandos de escritura.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF4.	Procesar comando de	Los comandos de escritura son empleados para modificar el estado de un punto. Estos puntos para ser escritos deben tener configurada la escritura.	Alta	Alta

	escritura.	La escritura de puntos digital se puede realizar de dos formas: una de manera completa (escritura simultánea) donde se escriben todos los bits de la variable, y la otra (escritura individual) donde se especifica el segmento que se verá afectado. El sistema debe estar iniciado a partir de un archivo XML de configuración válido generado por el configurador gráfico. Debe estar conectado al Middleware y suscrito al tópico de comandos. Se recibe un comando de escritura proveniente del HMI a través del Middleware. Se envía un comando de respuesta notificando la recepción del comando. Si se trata de un punto digital se debe separar el segmento del valor. El segmento indica la zona de memoria donde se escribirá el valor. Si se trata de un punto analógico se toma el valor, se linealiza en caso de tener configurada linealización para llevarlo a unidades manejables por el dispositivo. Se ordena la escritura al recolector y se programa un chequeo para comprobar que el comando fue ejecutado en el tiempo establecido. El procesamiento del comando termina con la respuesta del recolector, que en caso de haber sido exitosa la escritura se envía una respuesta de comando exitosa, en caso contrario la respuesta de		

		comando debe indicar la causa. Si se notifica que se acabó el tiempo para la escritura se envía una respuesta de comando indicándolo. El punto al que se le cambió el valor debe quedar establecido con el nuevo valor.		

Tabla 5 Especificación de Requisito Procesar alarma de falla de ejecución de comando.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF5.	Procesar alarma de falla de ejecución de comando.	La alarma de falla de ejecución de comando se emplea para notificar cuando un comando de escritura falló en su ejecución. El sistema debe estar iniciado a partir de un archivo XML de configuración válido generado por el configurador gráfico, con su representación en memoria de los puntos digitales configurados con sus respectivas alarmas de estado configuradas. El módulo debe estar listo para procesar información proveniente del Recolector y conectado al Middleware. Las alarmas a procesar deben estar habilitadas y desinhibidas. Se recibe un comando de escritura proveniente del HMI a través del Middleware. Se procesa el comando. Si el recolector en su respuesta informa que hubo error en la escritura se activa la alarma de falla de	Media	Media

		ejecución de comando y se publica en el Middleware. Esta alarma no tiene condición de desactivación, solo necesita ser reconocida para ser eliminada del sumario.		

Tabla 6 Especificación de Requisito Procesar comandos de operación de alarmas: reiniciar, inhibir y desinhibir.

Nº	Nombre	Descripción	Complejidad	Prioridad para cliente
RF6.	Procesar comandos de operación de alarmas: reiniciar, inhibir y desinhibir	Los comandos de operación de alarma se emplean para modificar el estado de las alarmas. El comando de inhibir se emplea para suspender el procesamiento de una alarma. El comando de desinhibir se emplea para continuar el procesamiento de una alarma inhibida. El comando de reiniciar se emplea para establecer todos los atributos de una alarma a sus valores por defecto. Como precondiciones se tiene que el módulo de procesamiento debe estar iniciado y suscrito al Middleware. Comienza cuando se recibe un comando con id 36, 41 o 42, se procesa el comando extrayendo su ID para determinar qué tipo de operación se desea ejecutar: reiniciar, inhibir o desinhibir. La alarma a operar debe estar configurada, habilitada y	Media	Media

		desinhibida. Cuando el comando tiene ID 36 se trata de un comando de reiniciar alarma, cuya ejecución consiste en establecer todos los atributos de estas como número de ocurrencias y el estado de la alarma a sus valores por defecto. Cuando el comando tiene ID 41 se trata de un comando de inhibir alarma, consiste en establecer el bit de estado que indica inhibición a 1. Cuando el comando tiene ID 42 se trata de un comando de desinhibir alarma, consiste en establecer el bit de estado que indica inhibición a 0. Al recibir el comando se envía un acuse de recibo indicando si se recibió correctamente o si hubo errores en los parámetros del comando.		

2.5 Modelo de Diseño

2.5.1 Diagrama de Paquetes

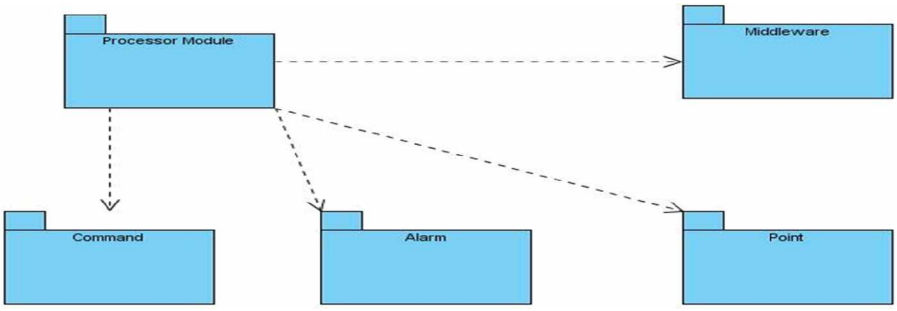

Ilustración 3 Diagrama de Paquetes

2.5.2 Diagrama de clases

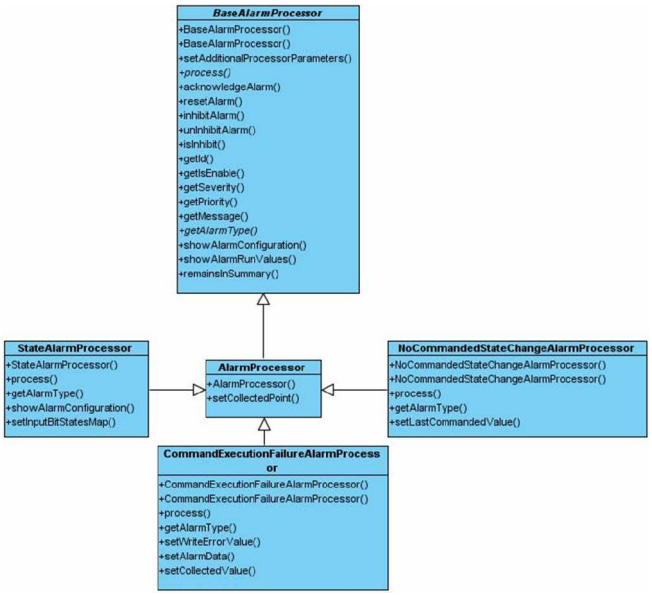

Ilustración 4 Diagrama de la jerarquía de clases de las alarmas

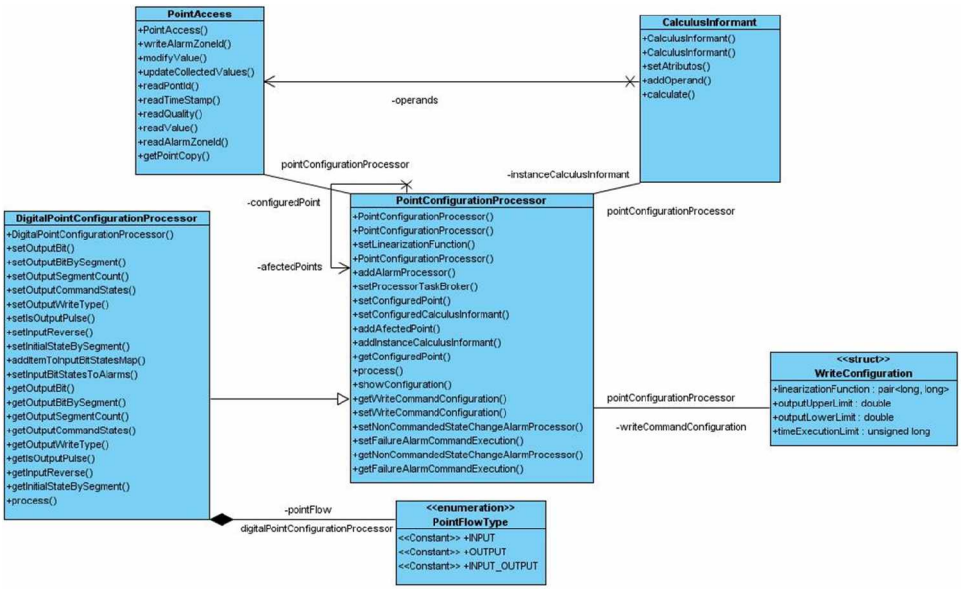

Ilustración 5 Diagrama de la jerarquía de clases de los puntos

2.5.3 Descripción de clases

Clase CommandExecutionFailureAlarmProcessor

La clase **CommandExecutionFailureAlarmProcessor** tiene como propósito representar la forma de manejar una situación no deseada consistente en la ocurrencia de un error al ejecutar un comando de escritura sobre un punto.

Descripción

La clase **CommandExecutionFailureAlarmProcessor** tienen como atributos el writeError, en el cual se guarda el valor del parámetro writeError recibido del recolector, el cual nos indica si es distinto de 0 que hubo error en la escritura, el collectedValue que almacena el valor que tiene el punto luego de ejecutada la escritura y el controlValue en el cual se guarda el valor que debería tener el punto. Esta clase tiene una relación de especialización/generalización con la clase AlarmProcessor de la cual hereda los atributos y métodos que definen a cualquier tipo de alarma. El método principal de esta alarma es process, el cual chequea si hubo error en la escritura o si el valor del dispositivo luego de la escritura difiere del que fue ordenado a escribir, en cualquiera de los dos casos se activa la alarma. Esta clase da respuesta al RF5 Procesar alarma de falla de ejecución de comando. Se ubica como una alarma en la clase PointConfigurationProcessor empleando composición.

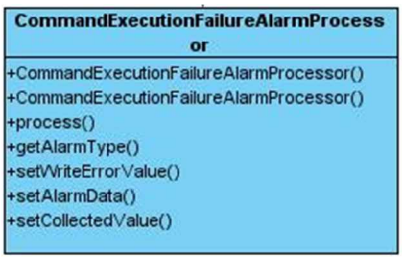

Ilustración 6 Diagrama de Clase CommandEcecutionFailureAlarmProcessor

Clase NoCommandedStateChangeAlarmProcessor

Esta clase tiene como propósito representar la forma de manejar una situación no deseada consistente en un cambio en el valor de un punto si haber sido comandado por el sistema.

Descripción

La clase **NoCommandedStateChangeAlarmProcessor** tienen como atributo lastCommandedValue que es el último valor que fue escrito empleando un comando de escritura, el cual se emplea para comparar con la última lectura efectiva del punto y activarse en caso de que sean diferentes. Se activa cuando la última lectura del punto difiere del último valor comandado. No tiene condición de desactivación, por esto para eliminarla del sumario solo es necesario que el operador del sistema reconozca la alarma. Hereda los atributos comunes a todas las alarmas de la clase AlarmProcessor. Implemente el método process para chequear. Da respuesta al RF3 Procesar alarma de cambio de estado no comandado. Se ubica como una alarma en la clase PointConfigurationProcessor empleando composición.

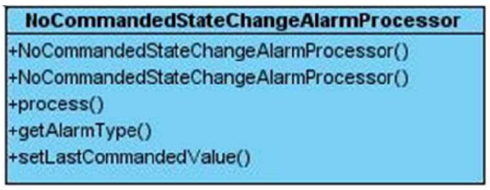

Ilustración 7 Diagrama de Clase NoCommandedStateChangeAlarmProcessor

Clase StateAlarmProcessor

Esta clase tiene como propósito representar un estado de un punto digital que de ser configurado indique el estado de un dispositivo.

Descripción

La clase **StateAlarmProcessor** tienen como atributo alarmState, el cual representa un estado que de coincidir con el del punto al que está asociada se activa la alarma. Esta alarma solo se encuentra presente en la clase DigitalPointConfigurationProcessor en modo composición. Hereda las funcionalidades comunes a todas las alarmas de la clase AlarmProcessor y su método process se encarga para comprobar si el punto tiene o no el estado de alarma, en caso positivo se activa. Si la alarma se encuentra

activa y el estado actual del punto difiere del de la alarma esta se desactiva. Da respuesta al RF2 Procesar alarma de estado.

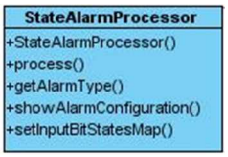

Ilustración 8 Diagrama de Clase StateAlarmProcessor

Clase DigitalPointConfigurationProcessor

Esta clase **DigitalPointConfigurationProcessor** tiene como propósito representar las características inherentes a un punto digital en memoria, como el procesamiento de las alarmas de estado.

Descripción

La clase **DigitalPointConfigurationProcessor** tienen como atributos principales el value que es el estado actual del punto, un diccionario con los pares llave estado numérico valor estado textual, el cual se emplea para determinar el estado textual del punto para ser comparado con el estado de las alarmas de estado. Esta clase hereda los atributos comunes a todos los puntos de PointConfigurationProcessor. Y sus alarmas de estado se almacenan en un diccionario conformado por los pares, id de las alarmas y alarma como tal. Esta clase da respuesta al RF1 Procesar puntos digitales.

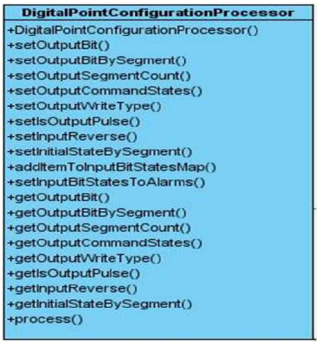

Ilustración 9 Diagrama de Clase DigitalPointConfigurationProcessor

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

En este capítulo se abordan los flujos de trabajo de Implementación y Pruebas realizado en el desarrollo de las funcionalidades. Se muestra el diagrama de componentes desarrollado para dar solución al problema planteado.

3.2 Implementación

En la fase de implementación es donde se realiza la implementación de las clases y objetos en ficheros fuente, binarios y ejecutables. Obteniéndose como resultado una aplicación ejecutable que incluye las funcionalidades definidas en la captura de requisitos funcionales.

3.2.1 Diagrama de componentes

Las clases que se obtienen en el diseño se hacen físicas mediante componentes. Los componentes representan módulos de software (código fuente, código binario, ejecutables, shared object(SO)). Para ilustrar mejor las dependencias entre los componentes se elabora el diagrama de componentes. El diagrama de componentes incluye fundamentalmente componentes de código fuente y ejecutables.

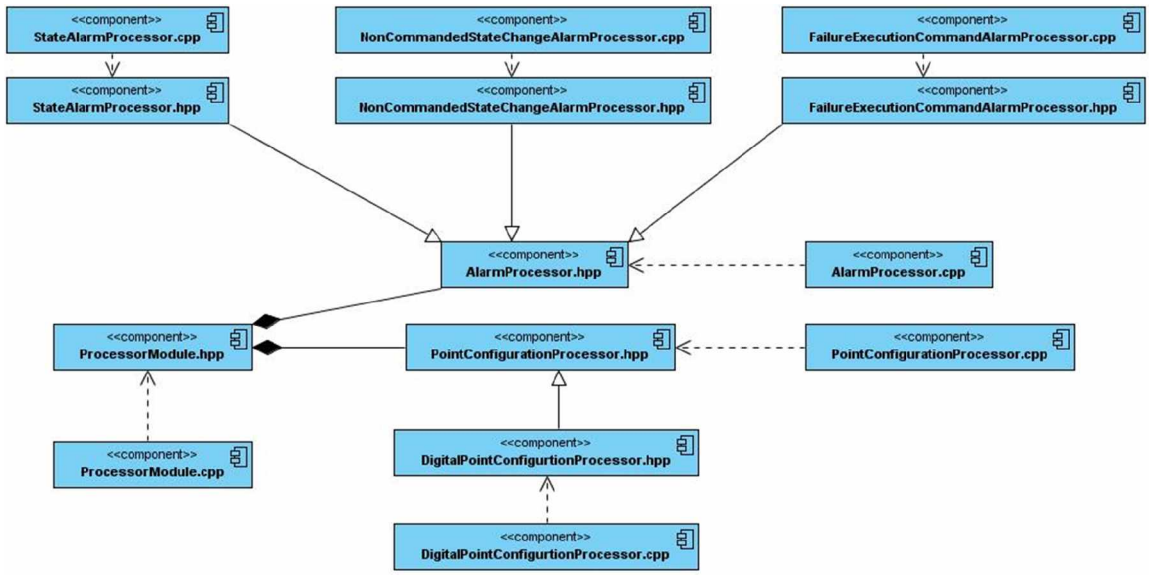

Ilustración 10 Diagrama de Componentes

3.2.2 Algoritmos críticos

A continuación se describe el algoritmo seguido para procesar el comando de escritura de punto debido a su alta criticidad.

El comando de escritura se recibe a través del Middleware verificándose que todos sus parámetros sean correctos. Luego se envía una respuesta de comando para dar acuse de recibo al módulo emisor del comando, en esta respuesta se indica si el comando es válido o no. Posteriormente se programa una operación de escritura que cuando es ejecutada se comprueba que el punto a escribir tenga configurada la operación de escritura, si tiene configurada linealización se le aplica esta al valor a escribir. Se planifica la escritura en el recolector, además de programar una verificación del tiempo de ejecución del comando. Una vez recibida la respuesta del recolector se procesa la alarma de falla de ejecución de comando la cual se activa si el comando no pudo ser ejecutado. El algoritmo finaliza enviando una respuesta de comando indicando la ejecución exitosa o no del comando.

Pseudocódigo de la tarea Procesar Comando de Escritura:

Si los parámetros del comando son correctos.

Se extrae del comando el ID del punto a escribir y se busca su procesador.

Se genera un ID para la solicitud de escritura.

Si el punto es analógico y tiene configurada escritura

Se linealiza el valor a escribir.

Se establece el valor a escribir en la alarmas de falla de ejecución de comando y de cambio de estado no comandado asociadas al punto.

Se almacena una solicitud de escritura.

Se planifica una tarea de escritura en el recolector.

Se planifica una comprobación para determinar si el comando fue ejecutado en el tiempo especificado.

Sino si el punto es digital.

Se separa el segmento del valor a escribir.

Se establece el valor a escribir en la alarmas de falla de ejecución de comando y de cambio de estado no comandado asociadas al punto.

Se almacena una solicitud de escritura.

Se planifica una tarea de escritura en el recolector.

Se planifica una comprobación para determinar si el comando fue ejecutado en el tiempo especificado.

Sino

Se establece el result de la respuesta de comando a Error de Comando General.

Sino

Se establece el result de la respuesta de comando a Parámetro de Comando Inválido.

Se envía la respuesta de comando.

3.2.3 Integración con el Middleware

El módulo de procesamiento para la realización de sus funciones de recepción y envío de información emplea los siguientes métodos para la interacción con el resto de los módulos, consistentes en la publicación o recepción de puntos y alarmas y recepción de comandos.

SendPoint(Punto): Método para enviar la información de un punto.

SendPointBatch(ListaPuntos): Método para enviar un batch de puntos.

SendAlarm(Alarma): Método para enviar la información de una alarma.

ReceiveCommand(Comando): Método para recibir comandos provenientes de otros módulos.

SendCommandResponse(Comando): Método para enviar una respuesta de comando.

3.2.4 Patrones de Diseño

Para la implementación de las funcionalidades descritas se emplearon los patrones de diseño método plantilla y adaptador. El método plantilla se emplea en la clase padre BaseAlarmProcessor el cual define un método que indica los pasos a seguir la activación de todas las alarmas e internamente emplea métodos implementados a nivel de clases hijas. Por su parte el patrón adaptador se usa para compatibilizar el mecanismo de envío y recepción de información del módulo de procesamiento con la interfaz del middleware.

3.3 Pruebas

Las pruebas son un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática. El objetivo de la etapa de la prueba de componentes es descubrir defectos probando componentes de programas individuales que pueden ser funciones, objetos o componentes reutilizables. (22)

A las funcionalidades implementadas se le realizaron pruebas unitarias de caja negra.

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

Las pruebas de caja negra se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una “Caja Negra” cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. No obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable se selecciona un conjunto de ellas sobre las que se realizan las pruebas. (23)

A continuación se documentan los casos de pruebas realizados a las funcionalidades implementadas en el presente trabajo.

Tabla 7 Pruebas al procesamiento de puntos digitales.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Procesar puntos digitales.	EC 1.1: Procesar un punto digital.	Esta funcionalidad permite procesar un punto digital, lo cual incluye carga de configuración, recepción de valores y procesamiento de sus alarmas asociadas.	Se carga en tiempo de configuración los atributos de un punto digital. Se comienzan a recibir las lecturas del punto. Los puntos son publicados en el Middleware para uso de los módulos restantes. Si se activa alguna alarma de estado es publicada también.

ID del escenario	Escenario	ID	Tipo	Respuesta del sistema	Resultado de la prueba
EC 1.1	Procesar un punto digital.	197 772 21	Digital Point	El sistema debe publicar puntos digitales indicando la correcta configuración de los mismos y procesamiento que incluye lectura, escritura y procesamiento de alarmas.	Se le da como entrada al módulo de procesamiento un XML de configuración con información de un punto digital. Se comienzan a recibir en el cliente de prueba las lecturas del punto digital. Se envían comandos de escritura y

					se comienzan a recibir las lecturas con los nuevos valores del punto y una alarma de estado indicando sobretorque.

Tabla 8 Pruebas a la alarma de estado.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Procesar alarmas de estado.	EC 2.1: Probar activación de alarma de estado.	Esta funcionalidad permite activar una alarma de cambio de estado no comandado para notificar que un punto tiene actualmente un valor que no fue establecido por un comando de escritura.	Se ordena a escribir un valor a un punto recibido en el cliente de pruebas. Como es su primera vez se establece ese como valor comandado del punto. Se modifica el valor del punto en el simulador de Modbus. El módulo de procesamiento recibe el nuevo valor de punto del recolector. Debido a que no coincide con el último valor escrito a través del sistema, se activa la alarma. La alarma es publicada con estado 9 indicando que está activada.
	EC 2.2: Probar desactivación de	Esta funcionalidad permite desactivar una alarma de	Se envía un comando de escritura al punto para modificar el estado

	alarma de estado reconocida.	estado para notificar que se dejó de cumplir la condición que la activó.	que está provocando la situación de alarma. Se procesa la alarma y al comprobar que ya su estado no coincide con el que tiene el punto ahora se desactiva. Se publica la alarma con el estado de desactivada.
	EC 2.3: Probar desactivación de alarma de estado no reconocida.	Esta funcionalidad permite desactivar una alarma de estado para notificar que se dejó de cumplir la condición que la activó.	Se envía un comando de escritura al punto para modificar el estado que está provocando la situación de alarma. Se procesa la alarma y al comprobar que ya su estado no coincide con el que tiene el punto ahora se desactiva. Se publica la alarma con el estado de desactivada.
	EC 2.4 Probar reconocimiento de alarma de estado activa.	Esta funcionalidad permite reconocer una alarma de estado para notificar que el operador del sistema conoce de su existencia. La alarma debe mantenerse en el sumario luego de ser reconocida debido a que aún no está desactivada.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 29, indicando que es un comando de reconocimiento de alarma.

			Se envía un comando de respuesta como acuse de recibo del comando. Se establece el estado de la alarma a reconocido. Si la alarma ya está desactivada es eliminada del sumario. En caso contrario permanece en el sumario hasta ser desactivada.
	EC 2.5 Probar reconocimiento de alarma de estado desactivada.	Esta funcionalidad permite reconocer una alarma de estado para notificar que el operador del sistema conoce de su existencia. La alarma debe ser eliminada del sumario debido a que ya se ha desactivado y ha sido reconocida.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 29, indicando que es un comando de reconocimiento de alarma. Se envía un comando de respuesta como acuse de recibo del comando. Se establece el estado de la alarma a reconocido. Si la alarma ya está desactivada es eliminada del sumario. En caso contrario permanece en el sumario hasta ser desactivada.

ID del escenario	Escenario	ID	Tipo	Estado	Respuesta del sistema	Resultado de la prueba
EC 2.1	Probar activación de alarma de estado.	25777322	StateAlarm	1	El sistema debe publicar la alarma con estado 9, indicando que el punto al cual pertenece la alarma tomo un el estado de alarma.	Se envía un comando de escritura a un punto digital forzándolo a tomar un estado que provoque la activación de la alarma. Cuando llega al módulo de procesamiento la lectura con el nuevo valor del punto se procesan las alarmas de estado y se activa la alarma que indica SobreTorque.
EC 2.2	Probar desactivación de alarma de estado reconocida.	25777322	StateAlarm	9	El sistema debe publicar la alarma con estado 1, indicando que la alarma se desactivó.	Se envía un comando de reconocimiento de alarma a la alarma de estado. Se recibe una respuesta de comando indicando el acuse de recibo del comando. La alarma llega al cliente de pruebas con estado 25 indicando que la alarma se encuentra activa y

						reconocida. Se envía luego un comando de escritura cambiando el valor del punto. La alarma llega al cliente de pruebas con estado 1, indicando que se desactivó. Es eliminada del sumario de alarmas.
EC 2.3	Probar desactivación de alarma de estado no reconocida.	25777322	StateAlarm	9	El sistema debe publicar la alarma con estado 1, indicando que la alarma se desactivó.	Se envía un comando de escritura cambiando el valor del punto. La alarma llega al cliente de pruebas con estado 1, indicando que se desactivó. Al no estar reconocida aún la alarma permanece en el sumario de alarmas esperando ser reconocida.

EC 2.4	Probar reconocimiento de alarma de estado activa.	25777322	StateAlarm	9	El sistema debe publicar la alarma con estado 25, indicando que la alarma se encuentra activa y reconocida.	Se envía un comando de reconocimiento de alarma a la alarma de estado. Se recibe una respuesta de comando indicando el acuse de recibo del comando. La alarma llega al cliente de pruebas con estado 25 indicando que la alarma se encuentra activa y reconocida. La alarma permanece en el sumario esperando a desactivarse.
EC 2.5	Probar reconocimiento de alarma de estado desactivada.	25777322	StateAlarm	1	El sistema debe publicar la alarma con estado 1, indicando que la alarma se encuentra desactivada y reconocida.	Se envía un comando de reconocimiento de alarma a la alarma de estado. Se recibe una respuesta de comando indicando el acuse de recibo del comando. La alarma llega al cliente de pruebas con estado 1 indicando que la alarma se encuentra desactiva y

						reconocida. La alarma es eliminada del sumario.

Tabla 9 Pruebas a la alarma de cambio de estado no comandado.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Procesar Alarma de Cambio de Estado No Comandado.	EC 3.1: Probar activación de alarma de cambio de estado no comandado.	Esta funcionalidad permite activar una alarma de cambio de estado no comandado para notificar que un punto tiene actualmente un valor que no fue establecido por un comando de escritura.	Se ordena a escribir un valor a un punto recibido en el cliente de pruebas. Como es su primera vez se establece ese como valor comandado del punto. Se modifica el valor del punto en el simulador de Modbus. El módulo de procesamiento recibe el nuevo valor de punto del recolector. Debido a que no coincide con el último valor escrito a través del sistema, se activa la alarma. La alarma es publicada con estado 9 indicando que está activada.

	EC 3.2: Probar reconocimiento de alarma de cambio de estado no comandado.	Esta funcionalidad permite reconocer una alarma de cambio de estado no comandado para notificar que el operador del sistema conoce de su existencia.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 29, indicando que es un comando de reconocimiento de alarma. Se envía un comando de respuesta como acuse de recibo del comando. Se establece el estado de la alarma a reconocido. La alarma es eliminada del sumario.

ID del escenario	**Escenario**	**ID**	**Tipo**	**Estado**	**Respuesta del sistema**	**Resultado de la prueba**
EC 3.1	Probar activación de alarma de cambio de estado no comandado	25777322	NoCommandesStateChangeAlarm	1	El sistema debe publicar la alarma con estado 9, indicando que ocurrió un cambio de estado no comandado en un punto.	Se envía un comando de escritura a un punto previamente recibido en el cliente de pruebas. Para que se establezca como último valor escrito por comandos.

	.					Se modifica el punto en el simulador Modbus. Cuando llega al módulo de procesamiento la nueva lectura y al no coincidir con el último valor escrito por comandos, se activa la alarma, la cual es publicada para uso de los restantes módulos.
EC 3.2	Reconocer alarma de cambio de estado no comandado.	25777322	FailureExecutionCommandAlarm	9	El sistema debe publicar la alarma con estado 1, indicando que la alarma se desactivó.	Se envía un comando de reconocimiento de alarma a una alarma de cambio de estado no comandado recibida en el cliente de pruebas. Se recibe una respuesta de comando indicando el acuse de recibo del comando. La alarma llega al cliente de pruebas con estado 1 indicando que la alarma ya está fuera del

						sumario.

Tabla 10 Pruebas al comando de escritura.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Procesar Comando de escritura de punto.	EC 4.1: Procesar comando de escritura de punto analógico.	Esta funcionalidad permite modificar el estado de un punto digital.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 24, indicando que es un comando de escritura. Se envía un comando de respuesta como acuse de recibo del comando. Se comprueba que el comando tenga configurada la escritura. Se separa el segmento del valor. Se ordena la escritura al recolector. Se programa un chequeo de tiempo de ejecución del comando. Se envía un comando de respuesta indicando que la acción fue

			ejecutada.
	EC 4.2: Procesar comando de escritura de punto analógico.	Esta funcionalidad permite modificar el estado de un punto analógico.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 24, indicando que es un comando de escritura. Se envía un comando de respuesta como acuse de recibo del comando. Se comprueba que el punto tenga configurada la escritura. Si el punto tiene configurada linealización se linealiza el valor a escribir. Se ordena la escritura al recolector. Se programa un chequeo de tiempo de ejecución del comando. Se envía un comando de respuesta indicando que la acción fue ejecutada.

ID del escenario	Escenario	ID	Tipo	Valor	Respuesta del sistema	Resultado de la prueba

EC 4.1	Procesar comando de escritura de punto analógico.	19777217	DigitalPoint	0	El sistema debe publicar el punto escribir el punto con su nuevo valor en la próxima lectura programada. Debe enviar dos respuestas de comando, una de acuse de recibo del comando y otra para notificar la ejecución exitosa o no del comando.	Se envía un comando de escritura a un punto previamente recibido en el cliente de pruebas. Se recibe un comando de respuesta indicando la recepción del comando en el módulo de procesamiento. Se recibe en el cliente de prueba el punto con el nuevo valor escrito. Se recibe además la respuesta de comando indicando ejecución exitosa (CommandSuccess).
EC 4.2	Procesar comando de escritura de punto analógico.	19777218	AnalogicPoint	0	El sistema debe publicar el punto escribir el punto con su nuevo valor en la próxima lectura programada. Debe enviar dos respuestas de comando,	Se envía un comando de escritura a un punto previamente recibido en el cliente de pruebas. Se recibe un comando de respuesta

					una de acuse de recibo del comando y otra para notificar la ejecución exitosa o no del comando.	indicando la recepción del comando en el módulo de procesamiento. Se recibe en el cliente de prueba el punto con el nuevo valor escrito. Se recibe además la respuesta de comando indicando ejecución exitosa (CommandSuccess).

Tabla 11 Pruebas a la alarma de falla de ejecución de comando.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Procesar alarma de falla de ejecución de comando.	EC 5.1: Activar alarma de falla de ejecución de comando.	Esta funcionalidad permite activar esta alarma para notificar que un comando de escritura falló en su ejecución.	El Módulo de Procesamiento recibe un comando de escritura a través del Middleware. Se procesa el comando de escritura. En la respuesta del recolector se verifica que si hubo error en la escritura o el valor del punto luego de la escritura difiere del valor que se ordenó escribir entonces se

			activa esta alarma.
	EC 5.2: Reconocer alarma de falla de ejecución de comando.	idad permite reconocer esta alarma para notificar que el operador del sistema conoce de su existencia.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 29, indicando que es un comando de reconocimiento de alarma. Se envía un comando de respuesta como acuse de recibo del comando. Se establece el estado de la alarma a reconocido. La alarma es eliminada del sumario.

ID del escenario	Escenario	ID	Tipo	Estado	Respuesta del sistema	Resultado de la prueba
EC 5.1	Activar alarma de falla de ejecución de comando.	25777322	Command Execution FailureAlarmProcesso	1	El sistema debe publicar la alarma con estado 9, indicando que falló el comando.	Se envía un comando de escritura a un punto previamente recibido en el cliente de pruebas. Se recibe un comando de respuesta indicando la recepción del comando en el módulo de

			r			procesamiento. Se fuerza a que ocurra un error de escritura, activándose la alarma de falla de ejecución de comando.
EC 5.2	Reconocer alarma de falla de ejecución de comando.	25777322	FailureExecution Command Alarm	9	El sistema debe publicar la alarma con estado 1, indicando que la alarma se desactivó.	Se envía un comando de reconocimiento de alarma a una alarma de falla de ejecución de comando recibida en el cliente de pruebas. Se recibe una respuesta de comando indicando el acuse de recibo del comando. La alarma llega al cliente de pruebas con estado 1 indicando que la alarma ya está fuera del sumario.

Tabla 12 Pruebas al Comando de Inhibir Alarmas.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Procesar Comando de inhibir alarmas.	EC 6.1: Procesar comando de inhibir alarma activa y no inhibida.	Esta funcionalidad permite inhibir el estado de una alarma activa y no inhibida indicando que no puede ser notificada.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 41, indicando que es un comando para inhibir alarma. Se localiza la alarma solicitada. Se establece el bit que indica inhibición en su estado a 1. Se propaga la alarma con su nuevo estado. Se envía un comando de respuesta indicando que la acción fue ejecutada.
	EC 6.2: Procesar comando de inhibir alarma activa e inhibida.	Esta funcionalidad permite inhibir el estado de una alarma activa e inhibida indicando que no puede ser notificada.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 41, indicando que es un comando para inhibir alarma. Se localiza la alarma solicitada y si

			ya está inhibida no se hace nada. Se envía un comando de respuesta indicando que la acción no fue ejecutada debido a que la alarma ya estaba inhibida.
	EC 6.3: Probar comando de inhibir alarma desactivada e inhibida.	Esta funcionalidad permite inhibir el estado de una alarma desactivada e inhibida indicando que no puede ser notificada.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 41, indicando que es un comando para inhibir alarma. Se localiza la alarma solicitada y si ya está inhibida no se hace nada. Se envía un comando de respuesta indicando que la acción no fue ejecutada debido a que la alarma ya estaba inhibida.
	EC 6.4: Probar comando de inhibir alarma desactivada y no inhibida.	Esta funcionalidad permite inhibir el estado de una alarma desactivada y desinhibida indicando que no puede ser notificada.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 41, indicando que es un comando para inhibir alarma.

			Se localiza la alarma solicitada. Se establece el bit que indica inhibición en su estado a 1. Se propaga la alarma con su nuevo estado. Se envía un comando de respuesta indicando que la acción fue ejecutada.

ID del escenario	**Escenario**	**ID**	**Tipo**	**Estado**	**Respuesta del sistema**	**Resultado de la prueba**
EC 6.1	Procesar comando de inhibir alarma activa y no inhibida.	25777222	LevelAlarm	9	El sistema debe publicar la alarma con estado 11 y un comando de respuesta notificando que se ejecutó correctamente el comando.	Se recibe en el cliente de prueba la alarma en cuestión con estado 11. Se recibe además la respuesta de comando indicando ejecución exitosa (CommandSuccess).
EC 6.2	Procesar comando de inhibir alarma activa e	25777222	LevelAlarm	11	El sistema debe publicar la alarma con estado 11 y un comando de	Se recibe en el cliente de prueba la alarma en cuestión con estado 11. Se recibe además la

	inhibida.				respuesta.	respuesta de comando indicando ejecución exitosa (CommandSuccess).
EC 6.3	Probar comando de inhibir alarma desactivada e inhibida.	25777222	LevelAlarm	3	El sistema debe publicar la alarma con estado 3 y un comando de respuesta.	Se recibe en el cliente de prueba la alarma en cuestión con estado 3. Se recibe además la respuesta de comando indicando ejecución exitosa (CommandSuccess).
EC 6.4	Probar comando de inhibir alarma desactivada y desinhibida.	25777222	LevelAlarm	1	El sistema debe publicar la alarma con estado 3 y un comando de respuesta.	Se recibe en el cliente de prueba la alarma en cuestión con estado 3. Se recibe además la respuesta de comando indicando ejecución exitosa (CommandSuccess).

Tabla 13 Pruebas al Comando de desinhibir Alarmas.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Procesar Comando de desinhibir	EC 7.1: Procesar comando de desinhibir alarma	Esta funcionalidad permite desinhibir el estado de una alarma	El Módulo de Procesamiento recibe un comando a través del Middleware.

alarmas.	activa e inhibida.	activa y no inhibida indicando que puede ser notificada nuevamente.	Se verifica que el ID del comando sea 42, indicando que es un comando para desinhibir alarma. Se localiza la alarma solicitada. Se establece el bit que indica inhibición en su estado a 0. Se propaga la alarma con su nuevo estado. Se envía un comando de respuesta indicando que la acción fue ejecutada.
	EC 7.2: Procesar comando de desinhibir alarma activa y desinhibida.	Esta funcionalidad permite inhibir el estado de una alarma activa e inhibida indicando que puede ser notificada nuevamente.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 42, indicando que es un comando para inhibir alarma. Se localiza la alarma solicitada y si ya está inhibida no se hace nada. Se envía un comando de respuesta indicando que la acción no fue ejecutada debido a que la alarma ya estaba inhibida.
	EC 7.3: Probar comando de inhibir alarma desactivada y	Esta funcionalidad permite inhibir el estado de una alarma desactivada e inhibida	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 42,

	desinhibida.	indicando que puede ser notificada nuevamente.	indicando que es un comando para inhibir alarma. Se localiza la alarma solicitada y si ya está inhibida no se hace nada. Se envía un comando de respuesta indicando que la acción no fue ejecutada debido a que la alarma ya estaba inhibida.
	EC 7.4: Probar comando de inhibir alarma desactivada e inhibida.	Esta funcionalidad permite inhibir el estado de una alarma desactivada y desinhibida indicando que puede ser notificada.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 42, indicando que es un comando para inhibir alarma. Se localiza la alarma solicitada. Se establece el bit que indica inhibición en su estado a 0. Se propaga la alarma con su nuevo estado. Se envía un comando de respuesta indicando que la acción fue ejecutada.

ID del escenario	**Escenario**	**ID**	**Tipo**	**Estado**	**Respuesta del sistema**	**Resultado de la prueba**

EC 7.1	Procesar comando de desinhibir alarma activa e inhibida.	25777222	LevelAlarm	11	El sistema debe publicar la alarma con estado 9 y un comando de respuesta notificando que se ejecutó correctamente el comando.	Se recibe en el cliente de prueba la alarma en cuestión con estado 9. Se recibe además la respuesta de comando indicando ejecución exitosa (CommandSuccess).
EC 7.2	Procesar comando de desinhibir alarma activa y desinhibida.	25777222	LevelAlarm	9	El sistema debe publicar la alarma con estado 9 y un comando de respuesta.	Se recibe en el cliente de prueba la alarma en cuestión con estado 9. Se recibe además la respuesta de comando indicando ejecución exitosa (CommandSuccess).
EC 7.3	Probar comando de inhibir alarma desactivada y desinhibida.	25777222	LevelAlarm	1	El sistema debe publicar la alarma con estado 1 y un comando de respuesta.	Se recibe en el cliente de prueba la alarma en cuestión con estado 1. Se recibe además la respuesta de comando indicando ejecución exitosa

						(CommandSuccess).
EC 7.4	Probar comando de inhibir alarma desactivada e inhibida.	25777222	LevelAlarm	3	El sistema debe publicar la alarma con estado 1 y un comando de respuesta.	Se recibe en el cliente de prueba la alarma en cuestión con estado 1. Se recibe además la respuesta de comando indicando ejecución exitosa (CommandSuccess).

Tabla 14 Pruebas al Comando de Reiniciar Alarmas.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo central**
Procesar Comando de reiniciar alarmas.	EC 8.1: Procesar comando de reiniciar alarma del sumario.	Esta funcionalidad permite reiniciar todos los atributos de una alarma el estado de una alarma dejándola en su estado inicial.	El Módulo de Procesamiento recibe un comando a través del Middleware. Se verifica que el ID del comando sea 36, indicando que es un comando para desinhibir alarma. Se localiza la alarma solicitada. Se reinician todos los atributos de la alarma

			quedando el estado con valor 1. Se propaga la alarma con su nuevo estado. Se envía un comando de respuesta indicando que la acción fue ejecutada.

ID del escenario	Escenario	ID	Tipo	Estado	Respuesta del sistema	Resultado de la prueba
EC 8.1	Procesar comando de reiniciar alarma del sumario.	25777222	LevelAlarm	9	El sistema debe publicar la alarma con estado 1 y un comando de respuesta notificando que se ejecutó el comando.	Se recibe en el cliente de prueba la alarma procesada con estado 1. Se recibe la respuesta de comando indicando ejecución exitosa (CommandSuccess).

CONCLUSIONES

Cumplimentados los objetivos del presente trabajo podemos arribar a las siguientes conclusiones:

- El módulo de procesamiento es indispensable para el funcionamiento del sistema SCADA UX.
- Las funcionalidades de procesamiento de puntos digitales, de alarmas de estado, de cambio de estado no comandado y de falla de ejecución de comando proveen de nuevos mecanismos de supervisión de situaciones no deseadas al módulo de procesamiento, así como el procesamiento de comandos de escritura y de operación de alarmas otorga mayor capacidad de interacción con el mismo.
- Con el resultado del presente trabajo se da cumplimento a los requisitos planteados por el proyecto SCADA Aguas de Santiago correspondientes al módulo de procesamiento.
- La nueva versión del módulo de procesamiento contribuye a que el SCADA UX sea más aplicable y competitivo.

RECOMENDACIONES

Con el objetivo de mejorar las funcionalidades implementadas en particular y el módulo de procesamiento en general se plantean las siguientes recomendaciones:

- Estudiar y aplicar técnicas de optimización de código a las nuevas funcionalidades con el objetivo de mantener y mejorar el código desarrollado.
- Desarrollar nuevas funcionalidades al módulo de procesamiento de manera que sea más completo.
- Estudiar y evaluar la factibilidad de ejecutar en hilos separados el procesamiento de puntos y alarmas con el objetivo de hacer un mayor aprovechamiento del microprocesador y hacerlo escalable.

REFERENCIAS BIBLIOGRÁFICAS

1. ***Baldiviezo Tórrez, Diego Efraín.*** *Informática Industrial.* http://www.buenastareas.com. *[En línea] 2009. http://www.buenastareas.com/ensayos/Informatica-Industrial/70972.html.*

2. ***Salazar Videaux, Leonel y Pérez-Alejo Neyra, Raúl.*** Módulo de Visualización de Gráficos Vectoriales para aplicaciones SCADA. *s.l. : UCI, 2007.*

3. ***Rubinos Carvajal, Alejandro Manuel.*** Arquitectura de software v2.0 Proyecto SCADA AS. *2010.*

4. ***Management, Emersson Process.*** *Field Devices.* Field Devices. *[En línea] 2010. http://www2.emersonprocess.com.*

5. ***SCADA Systems.*** *SCADA Systems.* SCADA Systems. *[En línea] 2009. http://www.scadasystems.net/.*

6. ***Gómez, Valeria Elena y Paz, Daniel Alfredo.*** *Comunicación Digital.* Comunicación Digital. *[En línea] 2006. http://www.dav.sceu.frba.utn.edu.ar/homovidens/gomezgomez_paz/PROYECTIN/PAGINA/.*

7. Conferencia 3 Base de Datos de Tiempo Real en el “GUARDIÁN DEL ALBA”. ***Herrera Vázquez, Moisés, Hernández, Yaneisy y Fardales Pérez, Jaime.*** *2008.*

8. El proceso de investigación científica. ***Hernández León, Rolando Alfredo y Coello González, Sayda.*** *La Habana : Editorial Universitaria, 2011.*

9. ***Real Academia Española.*** *Buscador Drae.* Buscador Drae. *[En línea] Real Academia Española, 2011. http://buscon.rae.es/draeI/.*

10. ***Bailey, David y Wright, Edwin.*** Practical SCADA for Industry. *Londres : IDC Tecnologies, 2003.*

11. ***Logiteksa.*** *Logiteksa Wonderware.* Logiteksa Wonderware. *[En línea] 2010. http://www.logiteksa.com/wonderware/.*

12. ***MeasureSoft.*** *MeasureSoft SCADA Pro.* MeasureSoft SCADA Pro. *[En línea] 2009. http://www.measuresoft.com/products/scadapro-server/.*

13. ***Herrera Vázquez, Moisés, Hernández, Yaneisy y Fardales Pérez, Jaime.*** Sistema Supervisor Guardián del ALBA. Introducción a la Arquitectura del Guardián del ALBA. *2008.*

14. TELENUL, SCADA de Supervisión y Telecontrol de Redes de Distribución, diseñado para recerradores y seccionalizadores NULEC. ***Otero Durán, María Elena.*** *2009.*

15. ***SERCONI.*** Ficha Técnica SCADA Eros. *2010.*

16. PROYECTO ARGOS, UN SCADA EN SOFTWARE LIBRE. ***Piña, Alejandro.*** *2009.*

17. ***Sardiñas García, Yuniel.*** Herramienta de respaldo, restauración y mantenimiento para el módulo de Base de Datos de Históricos del SCADA Guardián del ALBA. *s.l. : UCI, 2010.*

18. ***Herrera Vázquez, Moisés, Hernández, Yaneisy y Fardales Pérez, Jaime.*** Recolección y Manejadores en el Guardián del ALBA. *2008.*

19. ***Peralta González, Yunior y Tellería Martínez, Ana Silvia.*** Módulo de adquisición para un sistema SCADA. *2011.*

20. ***Sommerville, Ian.*** Ingeniería del software. *s.l. : Addison-Wesley, 2005.*

21. Introducción a Open Up. ***Jornada, Garcilaso.*** *s.l. : MUG Bs As, 2008.*

22. ***S. Pressman, Roger.*** Ingeniería de Software: un enfoque práctico. *s.l. : Mc Graw Hill, 2005.*

23. ***Juristo, Natalia, Moreno, Ana M. y Vegas, Sira.*** TÉCNICAS DE EVALUACIÓN DE SOFTWARE. *2006.*

24. Ayuda del Rational Unified Process.

25. ***Hernández León, Rolando Alfredo y Coello González, Sayda.*** *El proceso de investigación científica.* El proceso de investigación científica. *Ciudad de la Habana : Unversitaria, 2011.*

26. ***Chavarría Meza, Luis Eduardo.*** SCADA System´s & Telemetry. *2007.*

27. Alarm management for pipelines. ***Mark, McTavish.*** *Canadá : Matrikon inc., 2007.*

BIBLIOGRAFÍA

1. ***Salazar Videaux, Leonel y Pérez-Alejo Neyra, Raúl.*** Módulo de Visualización de Gráficos Vectoriales para aplicaciones SCADA. *s.l. : UCI, 2007.*

2. ***Rubinos Carvajal, Alejandro Manuel.*** Arquitectura de software v2.0 Proyecto SCADA AS. *2010.*

3. ***Herrera Vázquez, Moisés, Hernández, Yaneisy y Fardales Pérez, Jaime.*** Sistema Supervisor Guardián del ALBA. Introducción a la Arquitectura del Guardián del ALBA. *2008.*

4. Conferencia 3 Base de Datos de Tiempo Real en el "GUARDIÁN DEL ALBA". ***Herrera Vázquez, Moisés, Hernández, Yaneisy y Fardales Pérez, Jaime.*** *2008.*

5. El proceso de investigación científica. ***Hernández León, Rolando Alfredo y Coello González, Sayda.*** *La Habana : Editorial Universitaria, 2011.*

6. ***Bailey, David y Wright, Edwin.*** Practical SCADA for Industry. *Londres : IDC Tecnologies, 2003.*

7. TELENUL, SCADA de Supervisión y Telecontrol de Redes de Distribución, diseñado para recerradores y seccionalizadores NULEC. ***Otero Durán, María Elena.*** *2009.*

8. ***SERCONI.*** Ficha Técnica SCADA Eros. *2010.*

9. PROYECTO ARGOS, UN SCADA EN SOFTWARE LIBRE. ***Piña, Alejandro.*** *2009.*

10. ***Herrera Vázquez, Moisés, Hernández, Yaneisy y Fardales Pérez, Jaime.*** Recolección y Manejadores en el Guardián del ALBA. *2008.*

11. ***Peralta González, Yunior y Tellería Martínez, Ana Silvia.*** Módulo de adquisición para un sistema SCADA. *2011.*

12. ***Sommerville, Ian.*** Ingeniería del software. *s.l. : Addison-Wesley, 2005.*

13. Introducción a Open Up. ***Jornada, Garcilaso.*** *s.l. : MUG Bs As, 2008.*

14. ***S. Pressman, Roger.*** Ingeniería de Software: un enfoque práctico. *s.l. : Mc Graw Hill, 2005.*

15. ***Hernández León, Rolando Alfredo y Coello González, Sayda.*** *El proceso de investigación científica.* El proceso de investigación científica. *Ciudad de la Habana : Unversitaria, 2011.*

16. ***Chavarría Meza, Luis Eduardo.*** SCADA System´s & Telemetry. *2007.*

17. Alarm management for pipelines. ***Mark, McTavish.*** *Canadá : Matrikon inc., 2007.*

GLOSARIO DE TÉRMINOS

Calidad de Software: es la aptitud de un producto o servicio para satisfacer las necesidades del usuario.

Eficiencia: capacidad de lograr el efecto que se desea o se espera.

Modbus: es un protocolo de comunicaciones basado en la arquitectura maestro/esclavo o cliente/servidor, diseñado en 1979 por Modicon para su gama de controladores lógicos programables (PLCs).

Software libre: software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

Software privativo: cualquier programa informático en el que el usuario tiene limitaciones para usarlo, modificarlo o redistribuirlo.

Usabilidad: facilidad con que las personas pueden utilizar una herramienta particular o cualquier otro objeto fabricado por humanos con el fin de alcanzar un objetivo concreto.