Universidad de las Ciencias Informáticas

Título: “Desarrollo de la extensión para la configuración del módulo de adquisición de datos del SCADA Guardián del ALBA”.

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

Autor(es):

Daimeris Velasco Pérez

Yuniesky Orlando Vasconcelo Mir

Tutor(es):Ing. Angélica Cabrera González

Co-tutor: Ing. Evián Suárez Rodríguez

Ciudad de La Habana, 2010.

Año 52 de la Revolución

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 del Autor	Firma del Autor	Firma del Tutor

"... Si los jóvenes fallan, todo fallará. Es mi más profunda convicción que la juventud cubana luchará por impedirlo. Creo en ustedes."

Fidel Castro Ruz.

DATOS DE CONTACTO

Nombre y Apellidos: Angélica Cabrera González.
Edad: 23 años
Título: Ingeniero en Ciencias Informáticas.
Categoría: Adjunto a la Producción.
E-mail: acgonzalez@uci.cu
Graduada de Ingeniera Informática en el 2009 en la Universidad de las Ciencias Informáticas (UCI), con 1 año de experiencia en el desarrollo de software.

Nombre y Apellidos: Evián Suárez Rodríguez.
Edad: 25 años
Título: Ingeniero en Ciencias Informáticas.
Categoría:
E-mail: esrodriguez@uci.cu
Graduado de Ingeniero Informático en el 2009 en la Universidad de las Ciencias Informáticas (UCI), con 1 año de experiencia en el desarrollo de software.

AGRADECIMIENTOS

De Daimeris:

Aprovechando esta oportunidad quisiera decir gracias a todos los que han mostrado interés y preocupación por mí. En primer lugar a toda mi familia, en especial a mi mamá por su dedicación, por darme el beso de amor que nunca me ha faltado, que aunque hemos estado lejos durante estos 5 años siempre me ha dado su consejo y ánimo en el momento que más lo he necesitado, a mi papá por enseñarme que la palabra del triunfo es alcanzable mientras se estudia, por ser un ejemplo a seguir. Gracias a los dos por darme tanto cariño, amor, por su profunda confianza, por sus constantes llamadas en espera de alguna noticia, por el apoyo recibido durante mi formación profesional y a lo largo de toda mi vida, sin ustedes no lo hubiese logrado. Todo esto me hace sentir muy afortunada de tenerlos como padres, a mi hermanita por hacerme reír y ojalá el presente trabajo, le sirva de inspiración a coger una carrera universitaria, a mi abuelita por quererme tanto, por siempre tenerme presente y por darme siempre su consejo.

También quiero dar gracias a mi compañero de tesis por su paciencia, por siempre estar a mi lado cuando lo necesitaba, por su amistad.

A nuestra tutora por aclararnos las dudas, por su apoyo durante el desarrollo del presente trabajo, por su amistad, por su optimismo.

A mis compañeros y amigos que juntos hemos transitado por este camino y que de una forma u otra, me han ayudado en la realización de este trabajo.

Un agradecimiento especial a la Revolución por darme la oportunidad de estudiar en esta magnífica universidad.

De Vasconcelo:

Comienzo agradeciendo a mi mamá, porque siempre estuvo y está a mi lado con tanto amor, tanto cariño, apoyándome en todo y depositando una confianza enorme en mí; de la cual aprendí sobre la voluntad, el sacrificio y el amor a la familia. A mí papá el cual hizo de mi una persona trabajadora, honesta, honrada e independiente en el buen sentido. A mi hermano Yasi del cual aprendí a ser curioso, a querer aprender y cuestionarlo todo. A mi tío Eberto, el cual siempre me dio sabios consejos. En fin, a toda mi familia.

También quiero dar gracias a mi compañera de tesis por su paciencia, por haber siempre estado encima de mí para poder terminar la tesis.

A nuestra tutora y tutor por aclararnos las dudas, por su apoyo incondicional durante el desarrollo del presente trabajo, por su amistad, por su optimismo.

A mis compañeros y amigos que juntos hemos transitado por este camino y que de una forma u otra, me han ayudado en la realización de este trabajo, a mis hermanos del grupo y del kenpo.

A la Revolución Cubana por darme la oportunidad de estudiar, y desarrollarme.

DEDICATORIA

A mis padres y a mi abuela por tanto amor y entrega, por permitirme crecer, por inspirarme con tantos ejemplos de genialidad y sacrificio; y porque se han esmerado en mi formación.

A mi hermanita por quererla tanto.

Daimeris

A mi mamá, Olga, y a mi papá, Landy, por hacer y haber hecho posible mi existencia y, por darme todo el cariño, la fuerza y el sentido que impulsa y guía mi vida.

Yuniesky

RESUMEN

En un sistema de supervisión y control (como lo es el SCADA Guardián del ALBA) es vital la adquisición y el procesamiento de los datos del campo. Esta funcionalidad permite adquirir (obtener y almacenar) los datos de los dispositivos de control (PLCs, RTU, entre otros.) conectados al SCADA, así como controlar cuándo y con qué frecuencia se obtendrán dichos datos de acuerdo con los parámetros configurados.

Anteriormente en el Guardián del ALBA, el Editor de Configuración poseía un diseño y arquitectura poco flexible y no orientada a extensiones dinámicas (plugins), trayendo como consecuencias la modificación y re-compilación del código para configurar nuevos módulos adicionados al SCADA. Debido al cambio a una arquitectura robusta, flexible y orientada a extensiones del Editor, el sistema se ve afectado, porque no cuenta con la posibilidad de configurar el módulo de adquisición de datos. Sin esta extensión el Guardián del ALBA no sabe qué recolectar, ni cada qué tiempo, ni qué manejador se va a usar, ni qué procesamiento se le van hacer a esos datos.

Este trabajo tiene como objetivo obtener una extensión capaz de configurar el módulo adquisición de datos desde el Ambiente de Configuración (Editor) mediante una interfaz gráfica de usuario amigable que permita la interacción con los conceptos significativos del dominio del problema. Para alcanzar esta meta se describe una solución partiendo del diseño de software y se detalla en la implementación del sistema, validándola mediante la realización de las pruebas de unidad, viéndose de esta forma cumplido el principal objetivo del trabajo.

PALABRAS CLAVE

Configuración, Editor, Extensión, IHM, Módulo, SCADA.

Índice

ÍNDICE ..... **IX**

ÍNDICE DE FIGURAS **1**

INTRODUCCIÓN **2**

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA Y ESTADO DEL ARTE **7**

1.1 INTRODUCCIÓN 7
1.2 LOS SISTEMAS SCADA 7
1.2.1 Niveles de un sistema SCADA *8*
1.2.2 Funciones y Requisitos *10*
1.2.3 Componentes Hardware *12*
1.2.4 Componentes Software *13*
1.2.5 SCADA Guardián del ALBA *14*
1.2.6 Ventajas del SCADA Guardián del ALBA *14*
1.3 EJEMPLOS DE SISTEMAS SCADA 16
1.3.1 LabView *16*
1.3.2 SCADA Eros *16*
1.3.3 IHM/SCADA Paragon, de Nematron *16*
1.3.4 HYBREX (HybrexExpertSystem), de SIEMENS *17*
1.3.5 OpenSCADA *17*
1.3.6 FreeSCADA *18*
1.4 HERRAMIENTAS Y TECNOLOGÍAS A UTILIZAR 19
1.4.1 Proceso de Desarrollo y Gestión de Proyectos de Software del UCID *19*
1.4.2 UML *21*
1.4.3 Las herramientas CASE: Visual Paradigm *23*
1.4.4 Lenguaje de programación utilizado: C++ *23*
1.4.5 Técnica de programación: Programación Orientada a Objetos *24*
1.4.6 Entorno Integrado de Desarrollo: QtCreator *25*
1.4.7 Marco de trabajo: Qt *26*
1.4.8 Extensiones o Plugins *26*
1.4.9 Sistema Operativo GNU/Linux *26*
1.4.10 Distribución de Linux: Debian GNU/Linux *27*
1.5 CONCLUSIONES 28

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA **29**

2.1 INTRODUCCIÓN 29
2.2 PROCESO DE CONFIGURACIÓN DE LOS MÓDULOS DEL SCADA GUARDIÁN DEL ALBA 29
2.2.1 Editor de Configuración *29*
2.2.2 Configuración del Módulo de Adquisición de Datos *30*

2.3 PROPUESTA DE SOLUCIÓN 32
2.4 MODELO CONCEPTUAL 33
2.4.1 Descripciones de las entidades del Modelo Conceptual *35*
2.5 REQUERIMIENTOS DE LA SOLUCIÓN 35
2.5.1 Requisitos funcionales *35*
2.5.2 Requisitos no funcionales *38*
2.5.3 Prototipo de Interfaz de Usuario *40*
2.6 CONCLUSIONES 41

CAPÍTULO 3: DISEÑO DEL SISTEMA 42

3.1 INTRODUCCIÓN 42
3.2 DESCRIPCIÓN DE LOS ELEMENTOS DE LA ARQUITECTURA 42
3.2.1 Estilos arquitectónicos *42*
3.2.2 Patrones arquitectónicos *42*
3.3 DEFINICIONES DE DISEÑO QUE SE APLICAN 45
3.3.1 ¿Qué es un patrón? *45*
3.3.2 Patrones de Diseño GRASP *46*
3.3.3 Patrones de Diseño GOF *48*
3.4 DIAGRAMAS DE SECUENCIA Y DIAGRAMAS DE CLASES PARA CASA REQUISITO FUNCIONAL 48
3.5 DESCRIPCIÓN DE LAS CLASES 49
3.6 CONCLUSIONES 52

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA 53

4.1 INTRODUCCIÓN 53
4.2 DIAGRAMA DE COMPONENTES 53
4.2.1 Matriz de Integración de Componentes *55*
4.3 DIAGRAMA DE DESPLIEGUE 55
4.4 DISEÑOS DE CASOS DE PRUEBA 56
4.5 PRUEBAS DE SOFTWARE 56
4.5.1 Método de Prueba *57*
4.6 LISTADO DE LAS NO CONFORMIDADES 57
4.7 CONCLUSIONES 58

CONCLUSIONES 59

RECOMENDACIONES 60

TRABAJOS CITADOS 61

BIBLIOGRAFÍA 63

GLOSARIO DE TÉRMINOS 67

Índice de Figuras

Fig. 1 Niveles de un sistema SCADA. 9
Fig. 2 Diagrama de interrelación de alto nivel entre los subsistemas del Guardián del ALBA. 15
Fig. 3 Etapas del ciclo de vida del proyecto. 21
Fig. 4 Modelo conceptual. 34
Fig. 5 Prototipo de interfaz de usuario. 41
Fig. 6 Vista de paquetes de la extensión. 44
Fig. 7 Diagrama de clases general. 51
Fig. 8 Diagrama de componentes. 54
Fig. 9 Diagrama de despliegue. 56

Introducción

Ante las agresivas declaraciones hechas sobre Cuba el primero de Mayo del 2003 por el presidente norteamericano, quien situó a la Isla como uno de los 60 oscuros rincones del mundo donde podían realizar un golpe militar sin previo aviso; se tomó la decisión en un pleno del Partido celebrado en Julio de ese propio año de incrementar la capacidad defensiva del país, lo cual incluyó un grupo de medidas.

Fue en ese momento que el Ministro de las FAR le ordenó a la Industria Militar llevar a cabo la misión, que significó adentrarse en la producción y modernización de medios de combate. (1)

“Nos ha correspondido la misión de modernizar, perfeccionar y producir armamentos y técnica militar con altas cualidades combativas, como son la movilidad, el poder de fuego y la protección, entre otras, que se corresponden con nuestra doctrina militar de la Guerra de Todo el Pueblo”. (1)

Las Fuerzas Armadas Revolucionarias (FAR) de Cuba, institución militar básica del Estado con la misión fundamental de combatir al agresor, poseen una estructura, equipamiento y preparación que garantiza el cumplimiento de sus misiones combativas. Dentro de estas estructuras existen numerosas entidades las cuales necesitan supervisar y controlar a distancia una instalación, proceso, situación, o cualquier otro sistema. Estos sistemas están representados por una serie de dispositivos (PLC, APC, PIC) encargados de recolectar datos (temperatura, cantidad de fluido, humedad y movimiento, etc.), y ejecutar acciones (cerrar válvulas, apagar luces, entre otras), a través de sensores y actuadores incorporados respectivamente, para cumplir una tarea determinada. Las acciones pueden ejecutarse manual (con intervención humana, operador), y/o automática (sin intervención humana). Además, la información debe ser visualizada en tiempo real para observar el estado actual del proceso o situación, y almacenada para el análisis de la evolución histórica y acumulada.

Es por esto que en el año 2009 el centro de Modernización de la Técnica y el Armamento MTA perteneciente a la unidad 1722 decidió integrarse al equipo de desarrollo del SCADA Guardián del ALBA de la Universidad de las Ciencias Informáticas, para continuar desarrollando el sistema en conjunto. El sistema está compuesto por diferentes módulos, cada uno de ellos con una función específica y desarrollados usando herramientas y tecnologías de software libre; algunos de ellos son: los de procesamiento de datos, base de datos, comunicación, manejadores, seguridad y IHM. Este último está dividido en dos sub-módulos: el ambiente de configuración (Editor) y el ambiente de

ejecución (Run-Time). El ambiente de configuración engloba las utilidades relacionadas con la configuración del funcionamiento de los módulos que componen el sistema y, permite la creación de una aplicación para controlar y supervisar un proceso en particular. Mientras que el ambiente de ejecución es el encargado de mostrar al operador la aplicación creada.

Anteriormente, el ambiente de configuración poseía un diseño y arquitectura poco flexible y no orientada a extensiones dinámicas (plugins), trayendo como consecuencias la modificación y re-compilación del código para configurar nuevos módulos adicionados al SCADA. Debido al cambio a una arquitectura más robusta, flexible y orientada a extensiones del sub-módulo de configuración, el sistema se ve afectado, porque no cuenta con la posibilidad de configurar el módulo de adquisición de datos. Esta situación hace imposible definir las variables a supervisar y/o controlar, así como configurar el dispositivo donde estén las mismas, su dirección y la forma con la que se puede comunicar el sistema con el dispositivo, lo cual conduce a un mal funcionamiento del sistema.

Teniendo en cuenta todo lo anterior se plantea el siguiente **problema científico**: ¿Cómo lograr una solución para configurar el módulo de adquisición de datos del SCADA Guardián del ALBA a través del ambiente de configuración del propio SCADA?

Mediante la problemática planteada anteriormente podemos establecer el **Objeto de Estudio** de la presente investigación, la cual se centra en el proceso de configuración de los módulos del SCADA Guardián del ALBA, siendo el **Campo de Acción** el proceso de configuración del módulo de adquisición de datos del SCADA Guardián del ALBA.

Por lo tanto, la **idea a defender** sería: Si se desarrolla la extensión para el ambiente de configuración del SCADA Guardián del ALBA, se podrá configurar el módulo de adquisición de datos de dicho SCADA.

Planteada la idea a defender del presente trabajo, se pretende como **Objetivo General** desarrollar una extensión para el Editor del SCADA Guardián del ALBA usando herramientas y tecnologías de software libre, que permita configurar el módulo de adquisición de datos.

Tomando en cuenta lo antes mencionado se presentan los siguientes **Objetivos Específicos**:

- ✓ Analizar los sistemas de control, supervisión y adquisición de datos para comprender su funcionamiento, haciendo énfasis en el SCADA Guardián del ALBA.

- ✓ Realizar un análisis sobre el proceso de configuración de los módulos del SCADA Guardián del ALBA, para obtener una mejor comprensión de cómo se realiza este.
- ✓ Definir herramientas y tecnologías software a utilizar, para dar soporte a los procesos de modelación e implementación.
- ✓ Identificar los requisitos del sistema para garantizar una implementación satisfactoria de la extensión.
    - Identificar las variables, protocolos, alarmas y demás elementos de la recolección y adquisición de datos.
    - Identificar los parámetros a configurar.
    - Analizar la interfaz genérica de protocolos, para lograr un entendimiento de cuáles son los campos a configurar.
- ✓ Diseñar las funcionalidades de la extensión, para obtener más detalles e indicar lo que se debe programar.
- ✓ Implementar la extensión, para obtener el código fuente.
    - Aprender a utilizar el sistema de extensiones del marco de trabajo Qt.
    - Analizar el ejemplo de la extensión del Editor de Configuración.
        - Observar cuales son los requisitos que debe de cumplir la extensión para que pueda ser acoplado al configurador.
- ✓ Integrar la extensión con el editor de configuración del SCADA Guardián del ALBA, para comenzar a utilizarla.
- ✓ Diseñar y realizar las pruebas correspondientes a los requerimientos implementados, para encontrar defectos.
- ✓ Evaluar el resultado de las pruebas previamente realizadas, para valorar si se cumplió con los requerimientos.

Para realizar dichos objetivos específicos se tienen en cuenta varios **Métodos de Investigación (Científicos)**.

Teóricos:

- ✓ **Analítico-Sintético:** Para el análisis de la bibliografía especializada permitiendo la extracción de los elementos más importantes que se relacionan con el objeto de estudio y para arribar a las conclusiones de la investigación, así como determinar las características de los requisitos implementados.
- ✓ **Histórico-Lógico:** Para determinar las tendencias actuales de desarrollo de los modelos y enfoques de configuración.
- ✓ **Modelación:** Para observar la organización e interacción de los elementos del diseño de la solución propuesta.

Empíricos:

- ✓ **Entrevista:** Usado para obtener información acerca de la investigación y para la realización de un buen levantamiento de requisitos del sistema.
- ✓ **Experimento:** Aplicado para comprobar el comportamiento de nuestra idea a defender en la realidad.

El presente trabajo de diploma está estructurado de la siguiente manera: resumen, introducción, cuatro capítulos de contenido, conclusiones, recomendaciones, referencias bibliográficas, bibliografía consultada, anexos y por último un glosario de términos donde se plasmarán los términos importantes que nos brindan información del trabajo realizado.

Estructuración del contenido de los capítulos:

- ✓ **Capítulo 1:** Fundamentación Teórica y Estado del arte.

  En este capítulo se definirán los principales conceptos que serán utilizados en el desarrollo del presente trabajo para tener un mayor entendimiento, donde se hará referencia a las generalidades de un sistema de control, supervisión y adquisición de datos, enunciando sus principales características, requisitos y funcionamiento. Además, se hará un análisis de las herramientas y tecnologías de software a utilizar.

- ✓ **Capítulo 2:** Características del Sistema.

  El objetivo de este capítulo es obtener una visión más específica del sistema, donde se definirán los requisitos funcionales y no funcionales que regirán el desarrollo de la solución al problema. Además, se identificarán los conceptos más significativos del dominio del problema, sus relaciones y los atributos que lo componen.

- ✓ **Capítulo 3:** Diseño del Sistema.

  En este capítulo se presentarán los diagramas de clases de diseño, donde se especificarán las responsabilidades de las clases y sus relaciones. Conjuntamente se expondrán los diagramas de secuencia por cada escenario de los requisitos funcionales y los patrones de diseño y arquitectónicos que se aplicarán.

- ✓ **Capítulo 4:** Implementación y Prueba.

  En este último capítulo se codificarán las funcionalidades, se mostrarán los diagramas de componente y de despliegue que serán realizados y por último se diseñarán los casos de prueba y se ejecutarán las pruebas de unidad para detectar los errores.

Capítulo 1: Fundamentación Teórica y Estado del Arte

1.1 Introducción

En este capítulo se brinda una explicación sobre las generalidades de un sistema de control, supervisión y adquisición de datos, el funcionamiento de éste, sus requisitos y los principales módulos de software y hardware que lo componen. También se dará una breve explicación de la arquitectura del SCADA Guardián del ALBA. Por último, se conocerán las tecnologías y herramientas que se emplearán durante el desarrollo de este trabajo.

Es fundamental consultar lo investigado en este capítulo, el entendimiento de lo expuesto en el mismo, permitirá dar solución a la problemática que se plantea en la introducción de este trabajo.

1.2 Los sistemas SCADA

Las siglas SCADA son un anglicismo que abrevia la denominación de “Supervisory Control And Data Acquisition”, lo cual significa: sistema de Control, Supervisión y Adquisición de Datos.

Se trata de un tipo de aplicaciones software especialmente diseñadas para funcionar sobre ordenadores personales que desempeñen funciones a nivel de control de producción, proporcionando al operador una interfaz amigable y muy potente con los distintos dispositivos empleados para el control del proceso: autómatas programables, reguladores digitales, máquinas de control numérico y otros PLC. Los sistemas SCADA capturan toda la información del sistema automatizado y la filtran, acondicionan y presentan al operador de una manera coherente con sus distintas naturalezas lo que permite llevar a cabo la supervisión del sistema automatizado fácilmente. (2)

Los sistemas SCADA mejoran la eficacia del proceso de monitoreo y control proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas. De igual forma, al contar con información (alarmas, históricos, entre otros) de primera mano de lo que ocurre u ocurrió en el proceso, permite la integración con otras herramientas del negocio como lo son intranets y ERP. (3)

De forma general, SCADA es un término que se utiliza para describir los sistemas de supervisión, control y gestión de la evolución de un proceso en una amplia gama de industrias. Por ejemplo: en los sistemas de gestión de agua, energía eléctrica, señales de tráfico, sistemas de control ambiental, y sistemas de fabricación. No se deben confundir los sistemas SCADA con los Sistemas de Control Distribuidos (Distributed Control Systems, DCS), pues en estos últimos todas las acciones de control se realizan de forma automática.

1.2.1 Niveles de un sistema SCADA

Un sistema SCADA está formado por una amplia gama de elementos que colaboran para llevar a cabo las tareas de supervisión, control y adquisición de datos. De ahí que puedan subdividirse para su mejor comprensión en 4 niveles o partes esenciales: (3)

1. Nivel de instrumentación: este nivel toma las variables físicas y las convierte en una señal equivalente que puede ser leída o interpretada por el hombre. El sistema SCADA maneja instrumentación de tipo electrónica donde la variable física se convierte en una señal eléctrica.
2. Nivel de RTU: las RTU (Remote Terminal Unit) o unidades terminales remotas, son dispositivos inteligentes con microprocesador que recogen, almacenan y procesan la información que viene de la instrumentación de campo y comandan elementos finales de control. Son programables, permitiendo alojar algoritmos de control y poseen un canal de comunicación para su interconexión.
3. Nivel de comunicaciones: es el encargado de tomar la información de las RTUs y transmitirla por el medio escogido hasta el centro de control. Dependiendo del costo, tamaño del sistema SCADA, distancias a las RTUs, disponibilidad del medio, la velocidad de transmisión y la confiabilidad requerida, se seleccionan los distintos soportes y medios de comunicación más apropiados.
4. Nivel de Gestión o Centro de control: está compuesto por un conjunto de computadoras, periféricos y software que realizan el procesamiento de las señales. Existe al menos una computadora con su hardware y software de base en la cual se mantiene ejecutando un software SCADA de cierta complejidad, que abarca diversas funciones. Usualmente existe también un equipo que sirve como interfaz de comunicaciones, cuya función es recibir la

información de los diferentes canales de comunicación, procesarla y agruparla para enviarla a las restantes computadoras mediante una red.

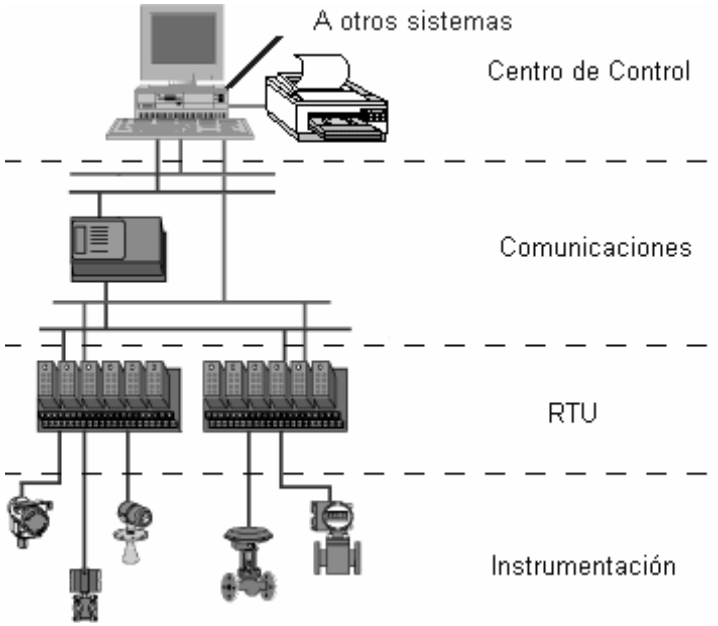


Fig. 1 Niveles de un sistema SCADA.

Como se puede apreciar a cada uno de estos niveles corresponden tanto elementos de software como de hardware, los cuales serán descritos en los siguientes epígrafes.

1.2.2 Funciones y Requisitos

Las principales funcionalidades que presenta un SCADA a sus usuarios, permitiéndole seguir y actuar sobre la evolución de un proceso son: (2)

- Adquisición de datos: para recoger, procesar y almacenar información relevante sobre la evolución del proceso.
- Supervisión: para observar desde un monitor la evolución del proceso.
- Control: para modificar la evolución del proceso (consignas, alarmas, recetas y menús) actuando por lo general a través de los equipos de control, aunque en ocasiones es posible la actuación directa sobre el proceso mediante las salidas conectadas.

La funcionalidad de Adquisición de Datos se encuentra al nivel de instrumentación y al nivel de RTU mencionados anteriormente, mientras que las funcionalidades de Supervisión y Control se encuentran al nivel de centro de control. Como se puede observar el nivel de comunicación es el encargado de relacionar a todas las funcionalidades.

A estas tres funciones básicas se pueden añadir otras funciones más avanzadas que permiten sacar el máximo provecho a la gran potencia de cálculo que proporcionan las PC de hoy en día, cuando son aplicados a tareas de supervisión: (2)

- Transmisión de información entre dispositivos de campo (pre-actuadores/actuadores y sensores) y otros equipos de la red corporativa de la empresa. La PC cumpliendo esta funcionalidad es una pieza clave en la integración vertical de la información de la empresa, ya que actúa como una pasarela entre el nivel de mando, regulación y el nivel de gestión empresarial.
- Posibilidad de procesar la información de proceso, conversiones a unidades de ingeniería, detección, procesamiento de alarmas y eventos del sistema.
- Visualización de los datos de campo y de sistema a través de Interfaces Hombre-Máquina, que representen los mímicos del proceso y permitan a los operadores actuar sobre la planta. No sólo es necesario que el sistema de supervisión permita representar gráficamente un

sinóptico del proceso que posibilite visualizar fácilmente la evolución del mismo, sino que además es muy interesante que el SCADA permita representar gráficos de tendencias de señales del proceso para poder analizar las causas que generan cuellos de botella en la producción, fallos inesperados, mal funcionamiento y desincronizaciones.

- Gestión de alarmas. El usuario necesita registrar los eventos que desencadenan una situación de alarma para poder prevenirlas y corregirlas. Posibilidad de crear paneles de alarmas, que exigen la presencia del operador para reconocer una parada o situación de alarma, con registro de incidencias.
- Ejecución de programas, que modifican la ley de control, o incluso anulan o modifican las tareas asociadas al autómata, bajo ciertas condiciones.
- Posibilidad de programación numérica, que permite realizar cálculos aritméticos de elevada resolución sobre la CPU del ordenador.
- Generación de históricos de señal de planta, que pueden ser volcados para su proceso sobre hojas de cálculo, reportes o aplicaciones de minería de datos. Esta característica es fundamental para poder alcanzar la integración vertical.
- Explotación e interpretación de los datos adquiridos. Se trata del conjunto de herramientas que permiten implementar un módulo de gestión de calidad, de control estadístico, o de gestión administrativa y financiera. Por lo general de este módulo se esperan complejas funcionalidades que se suelen implementar en herramientas auxiliares y no dentro del SCADA.
- Programación de rutinas de control. El empleo de un sencillo lenguaje de programación permite al usuario programar rutinas de control no críticas que de otra forma podrían saturar el ciclo de scan del autómata.
- Cálculos numéricos complejos. En ocasiones es necesario llevar a cabo cálculos numéricos que son más sencillos de programar empleando un lenguaje de programación de alto nivel que el lenguaje de programación del PLC.

Todos los sistemas SCADA deben cumplir varios requisitos básicos: (2)

- Deben ser sistemas de arquitectura abierta, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa.
- Deben permitir la comunicación con total facilidad y de forma transparente al usuario con el equipamiento de planta y con el resto de la empresa (redes locales y de gestión).
- Deben ser programas sencillos de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables de usuario.

Para realizar cada una de estas funcionalidades y cumplir con los requisitos, los sistemas SCADA se dividen en componentes software y hardware, los cuales abordaremos próximamente.

1.2.3 Componentes Hardware

El hardware empleado en un sistema SCADA es resistente a las extremas temperaturas, vibraciones, presión, voltajes, elementos químicos y otros agentes presentes en el campo. Independientemente de estas medidas de seguridad, se suele tener hardware de respaldo y se usan varios canales de comunicación.

El hardware involucrado en los sistemas SCADA se clasifica en cuatro grandes grupos coincidiendo con los niveles del SCADA mencionados anteriormente. Cada grupo cumple con funciones específicas que se describen a continuación: (4)

- La Unidad Central (MTU) es el ordenador principal del sistema, generalmente es una computadora personal que contiene la IHM. La MTU actúa como intermediaria con el operador e incluye la presentación de los datos y la administración de las alarmas en tiempo real. Se encuentra en el nivel de centro de control en un SCADA.
- Las Estaciones Remotas (RTU) están situadas en los nodos estratégicos del sistema gestionando y controlando las sub-estaciones, reciben los datos de los sensores de campo y le envían a los mismos los comandos que llegan desde la unidad central. Se encuentran en el nivel de RTU o nivel de automatización.

- La Red de Comunicación gestiona la información que se envía a la red de ordenadores. El tipo de implementación es variada y depende de las necesidades y del software escogido para el sistema. Debido a que los componentes pueden estar localizados en diversas áreas, se utilizan sistemas de tipo Red de Área Amplia (Wide Area Network, WAN).
- Los Instrumentos o Dispositivos de Campo son aquellos dispositivos que permiten tanto la supervisión y el control (PLC, controladores y actuadores) así como la adquisición de los datos (sensores) en el campo.

1.2.4 Componentes Software

Todos los sistemas SCADA, bien sean comerciales o hechos a medida, presentan una serie de módulos que permiten cubrir con las funcionalidades anteriormente descritas: (2)

- Configuración: permite al usuario definir el entorno de trabajo de su aplicación, según la disposición de pantallas requeridas y los niveles de protección de acceso para los distintos usuarios.
- Interfaz gráfica: proporciona al operador un módulo para la edición y configuración de las funciones de control y supervisión de la planta que han de tener una representación gráfica.
- Editor gráfico: permite al ingeniero la generación de las pantallas gráficas de supervisión mediante sinópticos gráficos almacenados en el ordenador de proceso y generados desde el editor incorporado en el SCADA o importados desde otra aplicación durante la configuración del sistema.
- Módulo de proceso: ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas. La programación se realiza por medio de bloques de programas en lenguaje de alto nivel (como C y Basic).
- Gestión y archivo de datos: se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo puedan tener acceso a ellos.
- Comunicaciones: se encarga de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre ésta y el resto de los elementos informáticos de gestión.

Cada uno de estos módulos representa una aplicación software encargada de cumplir con las funcionalidades mencionadas. Generalmente estos módulos se programan en un lenguaje de programación que pueda ser compilado o interpretado en varias plataformas hardware y Sistemas Operativos (SO), para darle una mayor libertad de elección de hardware y SO a los usuarios.

1.2.5 SCADA Guardián del ALBA

El SCADA “Guardián del ALBA”, para el cual se desarrolla el presente trabajo, es un sistema que permite la supervisión y control automático de los procesos industriales en el sector petrolero, aumentando la eficiencia y seguridad de esta industria. Esta aplicación, desarrollada por especialistas cubanos de la Universidad de las Ciencias Informáticas, venezolanos para Petróleos de Venezuela S.A. (PDVSA) y más recientemente por profesionales de las Fuerzas Armadas Revolucionarias (FAR) de Cuba.

1.2.6 Ventajas del SCADA Guardián del ALBA

Está dividido de forma general, en 10 sistemas que encajan en las definiciones de componentes software mencionados anteriormente, y además, que posibilitan la escalabilidad de las soluciones haciendo factible su utilización en una amplia gama de industria. La figura 2, representa la interrelación de alto nivel entre todos los subsistemas. Para un mayor entendimiento y explicación del funcionamiento y la arquitectura puede dirigirse a consultar la bibliografía(5).

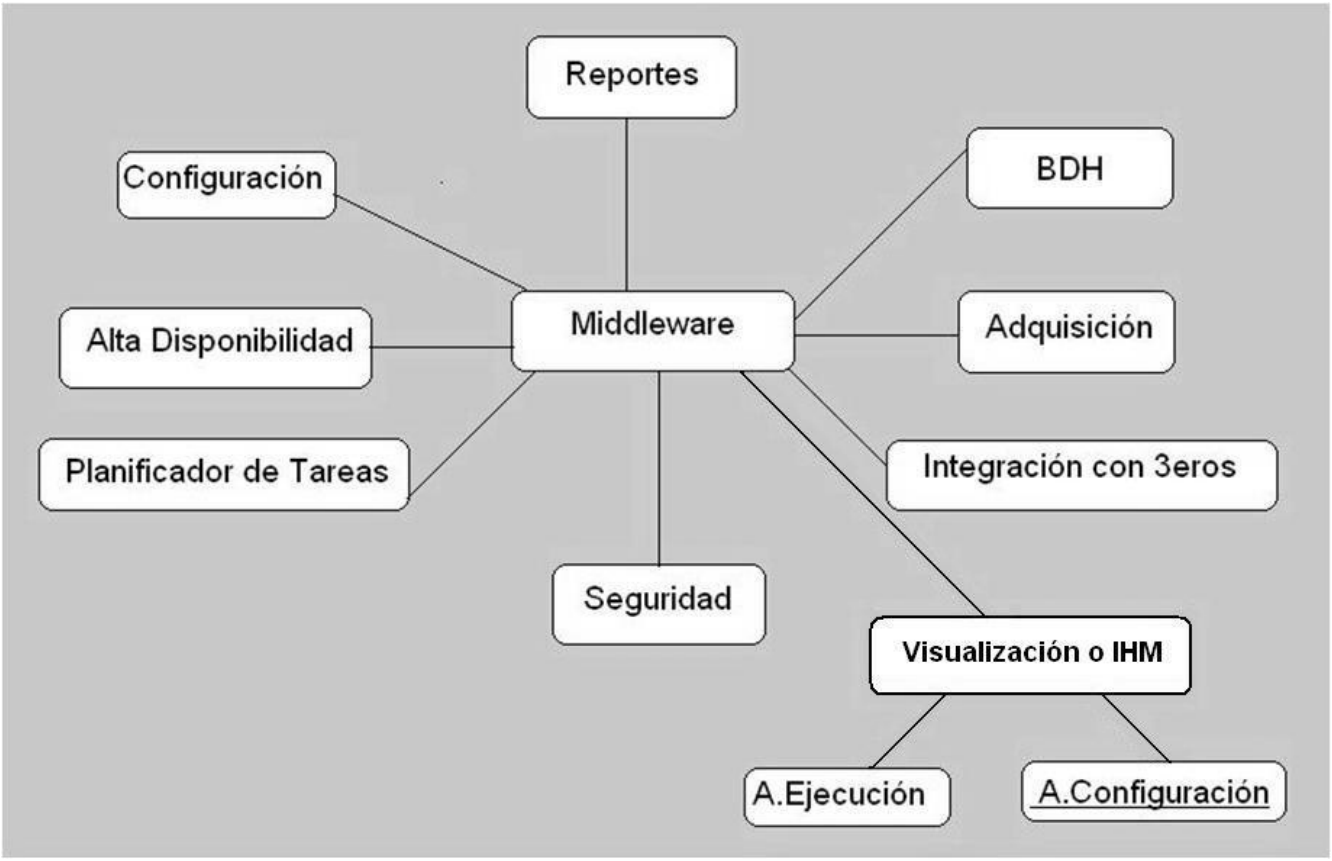


Fig. 2 Diagrama de interrelación de alto nivel entre los subsistemas del Guardián del ALBA.

Además, brinda alta integridad y confiabilidad garantizando su funcionamiento ininterrumpido de manera que se garantice la continuidad operacional y el flujo normal de los procesos de exploración, producción y transporte de crudo hasta las instalaciones aguas abajo. Utiliza software libre, con lo cual contribuye no sólo a la disminución de los costos sino también a elevar la soberanía tecnológica. (6)

El “Guardián del ALBA” es configurable y fácilmente extensible para su operación con nuevos dispositivos. Cuenta con una interfaz amigable que apoya la toma de decisión eficaz y un seguimiento efectivo de los procesos en la industria. Es muy flexible permitiendo la integración con otros sistemas, su adaptación y extensión.

1.3 Ejemplos de sistemas SCADA

Hoy en día existen varios sistemas SCADA, algunos muy reconocidos internacionalmente por el prestigio de sus proveedores. A continuación se mencionan algunos de los SCADA que se comercializan en el mundo:

1.3.1 LabView

Herramienta para la creación de sistemas SCADA. Es una herramienta diseñada especialmente para monitorizar, controlar, automatizar y realizar cálculos complejos de señales analógicas y digitales capturadas a través de tarjetas de adquisición de datos, puertos serie y GPIBs (Buses de Intercambio de Propósito General). Es un lenguaje de programación de propósito general pero con la característica de que es totalmente gráfico, facilitando de esta manera el entendimiento y manejo de dicho lenguaje para el diseñador y programador de aplicaciones tipo SCADA.

“Ofrece un ambiente de desarrollo gráfico con una metodología muy fácil de dominar por ingenieros y científicos. Con esta herramienta se pueden crear fácilmente interfaces de usuario para la instrumentación virtual sin necesidad de elaborar código de programación”. (7)

1.3.2 SCADA Eros

SCADA muy potente, con múltiples facilidades para operar y dirigir una amplia gama de procesos en cualquier industria. Muy amigable, fácil de operar y configurar, estable, probado, y es un sistema distribuido. Se ha podido observar su utilización y generalización en diferentes sectores de la industria cubana. El Sistema ha servido de base para muchos otros desarrollos. (8)

1.3.3 IHM/SCADA Paragon, de Nematron

Software poderoso y flexible, permite construir aplicaciones para una completa visualización del operador, supervisión, control y adquisición de datos (SCADA).

Debido a que las funciones para la reparación de errores se encuentran integradas en los módulos de control, IHM y SCADA, todas ellas comparten una sola base de datos, facilitando así la programación

y localización de errores. La misma base de datos creada para el sistema de control se usa para configurar entradas y salidas, pantallas de operador, adquisición de datos y otras aplicaciones. Soporta las normas abiertas como OPC, ActiveX y COM/DCOM, e incluye capacidades avanzadas de diagnóstico, por lo que se facilita el mantenimiento y la capacitación del personal técnico. (7)

1.3.4 HYBREX (HybrexExpertSystem), de SIEMENS

Herramienta de simulación que permite realizar cambios virtuales en la planta y observar sus resultados sin ningún riesgo. Está específicamente orientada a procesos de laminado en plantas siderúrgicas y se puede utilizar en cualquiera de las etapas del ciclo de vida de la planta, desde construcciones nuevas hasta plantas en procesos de optimización y modernización. (7)
Los SCADA anteriormente mencionados son totalmente propietarios.

También se han desarrollado algunos sistemas SCADA utilizando tecnologías libres, teniendo en cuenta el incremento del avance que cada día protagonizan el software que apuestan por el código abierto. A continuación se muestran algunos proyectos que han tenido notables resultados:

1.3.5 OpenSCADA

El sistema OpenSCADA es capaz de supervisar y controlar una máquina pequeña o de gran escala, brinda la arquitectura e ingeniería del sistema de control de proceso de forma fiable y segura.
El monitoreo remoto se facilita a través de LAN y de la comunicación segura de Internet.

Además, de esto permite una perfecta integración de los negocios, la documentación y las funcionalidades de archivo. El archivo de los datos de entrada tanto manual y/o automáticamente que se generan son confiables y pueden ser utilizados para los procesos de facturación.

OpenSCADA es actualmente empleado en el sistema de control de envío de una de las empresas líder de petróleo y gas.

La seguridad es una cuestión fundamental para todos los sistemas SCADA. Este ofrece la máxima seguridad en todas las capas del sistema correspondiente, lo que proporciona al mismo tiempo una gran flexibilidad.

Proporciona una plataforma SCADA abierta, esto significa que es independiente de las tecnologías, no se limita a un determinado Sistema Operativo.

Además, la integración es simple, es altamente configurable, lo que significa que la creación de un entorno de ejecución es fácil. El sistema es fácilmente extensible, y los entornos de prueba y de simulación son fáciles de configurar. Provee una arquitectura abierta y un entorno de desarrollo abierto.

El proyecto OpenSCADA consiste en módulos que en su conjunto crean el sistema SCADA completo. Los módulos se almacenan en bibliotecas dinámicas. Cada biblioteca puede contener un conjunto de módulos de varios tipos. (9)

1.3.6 FreeSCADA

Es un sistema basado en UNIX. El sistema FreeSCADA proporciona un entorno completo para el control del proceso y recogida de datos, es muy flexible y fácil de usar para la programación de las normas de control. Incluye una interfaz de usuario totalmente gráfica para la configuración y supervisión de procesos, utilidades de alarma y acceso remoto. El sistema de FreeSCADA también tiene un interfaz web.

Es un avanzado sistema de control de propósito general que puede aplicarse en una gran variedad de aplicaciones de control. Es especialmente adecuado para ámbitos como la automatización de edificios, control de medio ambiente y control de procesos industriales.

FreeSCADA alcanza un rendimiento que es superior a los PLC tradicionales basados en sistemas de control, y con menores costos. Por ejemplo, en la base de datos FreeSCADA registros de varios años de expansión se pueden almacenar en alta resolución de los recursos, mientras que los sistemas tradicionales a menudo son muy reducidos para los registros de datos. Tiene una capacidad de almacenamiento superior.

Es un sistema basado en componentes de código abierto. Está diseñado para controlar y/ o recoger datos para diversos procesos. Es muy fiable y tolerante a fallos. (10)

1.4 Herramientas y Tecnologías a utilizar

El desarrollo de software no fuera posible sin las herramientas y tecnologías que le dan soporte, pues de no existir estas el trabajo sería engorroso, tedioso y muy lento para la obtención de los resultados. Las metodologías y herramientas a utilizar para llevar a cabo la propuesta de solución son las siguientes:

1.4.1 Proceso de Desarrollo y Gestión de Proyectos de Software del UCID

Un proceso de desarrollo de software tiene como objetivo la producción eficiente de un producto de software que satisfaga los requisitos de un cliente con una planificación y una estimación de recursos predecibles. Los elementos de un proceso y sus relaciones deben responder Quién debe hacer Qué, Cuándo y Cómo.

Quién: Las personas participantes en el proyecto de desarrollo desempeñando uno o más roles específicos.

Qué: Un artefacto es producido por un rol como resultado del desarrollo de sus actividades. Los artefactos se especifican utilizando notaciones. Las herramientas apoyan la elaboración de artefactos.

Cómo y **Cuándo**: Las actividades son una serie de pasos que lleva a cabo un rol durante el proceso de desarrollo. El avance del proyecto está controlado mediante hitos que establecen un determinado estado de terminación de ciertos artefactos.

Este Proceso de Desarrollo y Gestión de Proyectos (a partir de aquí se abreviará como PDGP-UCID) es una metodología desarrollada por la Unidad de Compatibilización Integración y Desarrollo de Software para la Defensa, con el fin de llevar a cabo el desarrollo exitoso de software dentro de las FAR. A continuación se explica en que consiste.

El desarrollo de cualquier tipo de proyecto consiste en una serie de fases, muchas veces secuenciales conocidas como el ciclo de vida del proyecto y generalmente definen: qué trabajo técnico se hará en cada fase, cuando se generarán los entregables, como se revisarán, verificarán y

validarán en estas, quién está involucrado en cada fase y cómo controlar y aprobar cada una. Las principales características del ciclo de vida expuesto en PDGP-UCID son:

1. Las fases son secuenciales y su transferencia debe ser precedida por un proceso de revisión o liberación del Centro de Calidad y su aprobación en Consejo Técnico Formal.
2. El nivel del personal es bajo al comienzo, alcanza su nivel máximo en la fase de construcción y decae rápidamente cuando el proyecto se aproxima a su conclusión.
3. La participación de los interesados es alta en las etapas de Inicio y Modelación, baja en la etapa de Construcción y vuelve a subir en las etapas finales del proyecto.

En el PDGP-UCID el ciclo de vida del proyecto se considera como parte del ciclo de vida del producto, el cual representa un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en la adquisición, el desarrollo, la explotación y el mantenimiento de un producto.

Las etapas o fases permiten mejorar el planeamiento, ejecución y control del proyecto, se caracterizan por la conclusión y la aprobación de uno o más artefactos entregables siendo estos productos del trabajo, mensurables y verificables, correspondiendo algunos a los procesos de gestión de proyectos y otros al producto. La consecución exitosa de cada fase es indispensable para poder continuar con el proyecto. El ciclo de vida de un proyecto de software desarrollado basado en PDGP-UCID se descompone en el tiempo en cinco fases secuenciales que son: Inicio, Modelación, Construcción, Explotación Experimental y Despliegue. Al final de cada fase los representantes de los grupos de roles presentes en el proyecto realizan una evaluación para determinar si los objetivos se cumplieron y así presentar al Consejo Técnico Formal para su evaluación y dar paso o no a la fase siguiente. A continuación se muestran las etapas del ciclo de vida de los proyectos así como los principales artefactos de cada una de estas.

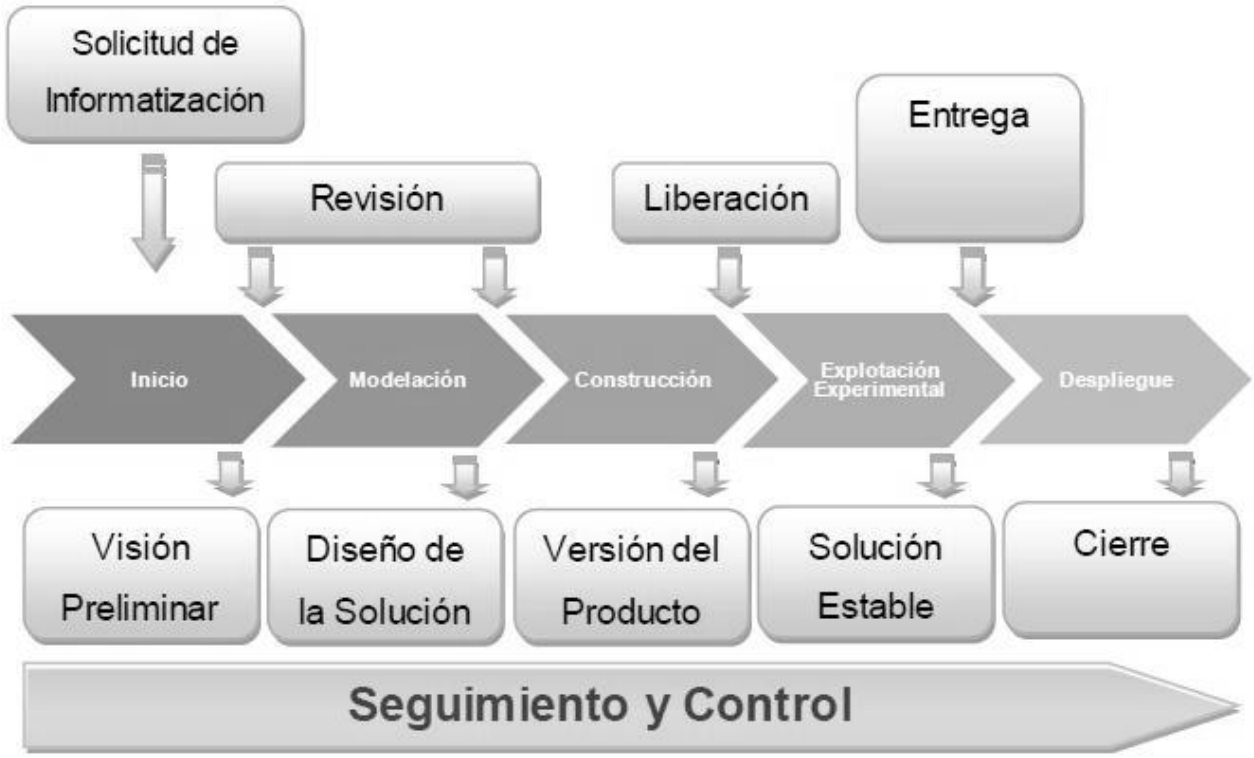


Fig. 3 Etapas del ciclo de vida del proyecto.

El modelo de desarrollo de software propuesto describe la secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones. Se logra con la combinación entre los modelos basado en Componentes, el Iterativo y el Incremental. (11)

1.4.2 UML

El modelado es una parte central de todas las actividades que conducen a la producción de un buen software, porque a través de él logramos comprender el sistema a desarrollar.

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML) es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. (12)

El Lenguaje Unificado de Modelado es el lenguaje que permite la modelación de sistemas de software con tecnología orientada a objetos más conocido y utilizado en la actualidad. Es un lenguaje gráfico que cuenta con un grupo de diagramas, los cuales son utilizados para visualizar, especificar, construir y documentar un sistema de software en cada una de las etapas por las que tiene que pasar. Indica que es lo que supuestamente hará el sistema, pero no como lo hará. Para la solución propuesta se utilizará UML 2.0.

1.4.3 Las herramientas CASE: Visual Paradigm

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son las aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas contribuyen de manera directa en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (13)

Para el modelado de los artefactos y diagramas generados a lo largo del ciclo de vida del proyecto se decidió emplear Visual Paradigm en su versión 3.4, pues su uso está muy estandarizado a nivel mundial y constituye una herramienta multiplataforma[1] muy madura y acabada.

Visual Paradigm para UML es una herramienta profesional fácil de utilizar, que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a la rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, la realización de ingeniería tanto directa como inversa, generar el código desde diagramas y generar la documentación automáticamente en varios formatos como Web o Formato de Documento Portable (pdf) y el control de versiones. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto. Proporciona también abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. La única desventaja es que es una herramienta propietaria, pero si distribuyen copias gratuitas para la educación. (14)

1.4.4 Lenguaje de programación utilizado: C++

C++ es la evolución de C adaptada a la programación orientada a objetos. Se encuentra entre los más usados para desarrollar aplicaciones gráficas en 3D, por ser los que con más velocidad ejecutan el código, en general puede llegar a ser un lenguaje tan rápido como C (el más rápido después del

[1] Sistema que puede ejecutarse en diversos Sistemas Operativos.

Lenguaje Ensamblador). Además, tiene algunas cuestiones más pulidas como un control más estricto en el manejo de tipos de datos, y otras características que ayudan a la programación libre de errores. (14)
Es considerado como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel.

Este lenguaje de programación posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución.
- Embeber código ensamblador.

Entre las principales ventajas del C++ como lenguaje de programación orientado a objetos se encuentran:

- C++ es un lenguaje de propósito general, o sea, que con el se puede programar cualquier cosa, desde sistemas operativos y compiladores hasta aplicaciones de bases de datos y procesadores de texto.
- Los programas escritos en C++ tienen la ventaja de que podrán ejecutarse en cualquier máquina y bajo cualquier sistema operativo.
- Es un lenguaje multi-nivel, es decir, se puede usar para programar directamente el hardware o para crear aplicaciones tipo Windows.
- Es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

1.4.5 Técnica de programación: Programación Orientada a Objetos

Con el uso de esta técnica de programación se pretende simplificar la modificación y extensión del software para lograr una mayor reutilización del mismo, permite una fácil comprensión debido a que la estructura del software y del problema a resolver están relacionadas directamente. Aplicando diseño

orientado a objetos, se crea software resistente al cambio y escrito con economía de expresión. Esta técnica es la más usada actualmente por programadores. (15)

1.4.6 Entorno Integrado de Desarrollo: QtCreator

Los Entornos Integrados de Desarrollo, IDE (del inglés: Integrated Development Environment) constituyen programas compuestos por un conjunto de herramientas para los programadores que facilitan la escritura de programas. Pueden dedicarse en exclusivo a un sólo lenguaje de programación o bien, pueden utilizarse para varios. Los IDE facilitan un ambiente de trabajo amigable. Para el desarrollo de la extensión se decidió usar QtCreator como IDE.

QtCreator ofrece un ambiente de desarrollo completo para la creación de aplicaciones Qt. Es una herramienta ligera y multiplataforma, con un enfoque estricto hacia las necesidades de los desarrolladores de aplicaciones Qt. Las principales características son:

1. El editor avanzado de C++, para escribir, editar y navegar por el código fuente sin utilizar el ratón (mouse).
2. Interfaz gráfica para la depuración, que incrementa la percepción de la estructura de clases de Qt.
3. Integración con QtDesigner para editar los archivos de interfaz gráfica de usuario.
4. La herramienta Localizador (Locator) para la navegación rápida por archivos de proyecto, funciones, clases e informaciones de ayuda.
5. Integración con QtHelp, facilitando el acceso a la ayuda de la API Qt (tipos de datos, funciones) de Qt a través de una ayuda sensible al contexto.
6. Uso de diferentes sistemas para el control de versiones como Git, Subversion, CVS and Perforce.
7. Construir y ejecutar proyectos Qt con la herramienta multiplataforma qmake.

Estos son los elementos principales tenidos en cuenta para su elección.

1.4.7 Marco de trabajo: Qt

Qt es un marco de trabajo (framework) escrito en C++ para el desarrollo de aplicaciones de Interfaz Gráfica de Usuario, el cual sigue la filosofía de software “escriba una vez, compile donde quiera” (write once, compile anywhere).

Además, amplía las posibilidades del lenguaje C++ con una extensa biblioteca de clases, de uso intuitivo y dividida en módulos, en los cuales se puede encontrar desde programación de interfaces gráficas de usuarios hasta programación de redes, procesamiento de textos enriquecidos, acceso a datos almacenados en varios de los gestores de bases de datos más populares. Incorpora herramientas para escribir aplicaciones robustas y en el menor tiempo posible como el Diseñador Qt (QtDesigner). A esto se le suma su extensa base de datos de ejemplos listos para usar. Qt está disponible bajo licencia LGPL, permitiendo el desarrollo de Software Libre, y si se desea, software comercial no libre, en ambos casos sin vernos obligados al pago de licencias o derechos.

1.4.8 Extensiones o Plugins

Las extensiones son pequeños programas que por si solos no tienen ninguna utilidad, debido a que ellos son totalmente dependientes de otros programas más complejos y funcionales. Además, son específicos de una aplicación, y deben cumplir con las exigencias y restricciones de programación de esta, para poder ser reconocidos y usados por la misma.

Una de las tareas del presente trabajo, es realizar una extensión para el ambiente de configuración de procesos del SCADA Guardián del ALBA, el cual está implementado con el lenguaje de programación C/C++ y utilizando el marco de trabajo Qt.

1.4.9 Sistema Operativo GNU/Linux

Software libre es la denominación del software que brinda libertad a los usuarios sobre un producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software; de modo más preciso, se refiere a cuatro libertades

de los usuarios del software: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo que puede ayudar a otros; de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie (para la segunda y última libertad mencionadas, el acceso al código fuente es un requisito previo). Una distribución de Linux es una variante de ese sistema operativo (SO) que incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones hogareñas, empresariales y para servidores. Pueden ser exclusivamente de software libre, o también incorporar aplicaciones o controladores propietarios. (16)

Este proyecto será desarrollado a partir de las herramientas que brinda el software libre. Buscando la independencia y soberanía tecnológica que este brinda al posibilitar la libertad de uso y distribución de los programas sin incurrir en discusiones de licenciamiento o asuntos legales.

1.4.10 Distribución de Linux: Debian GNU/Linux

Debian es un SO libre, utiliza el núcleo Linux (el corazón del sistema operativo), pero la mayor parte de las herramientas básicas que completan el sistema operativo vienen del Proyecto GNU, de ahí el nombre: GNU/Linux. También es una comunidad conformada por desarrolladores y usuarios. Debian nace como una apuesta por separar en sus versiones el software libre del software no libre. El modelo de desarrollo del proyecto es ajeno a motivos empresariales o comerciales, siendo llevado adelante por los propios usuarios, aunque cuenta con el apoyo de varias empresas en forma de infraestructuras. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuir comercialmente este software mientras se respete su licencia. Debian es la única distribución importante de GNU/Linux mantenida solamente por voluntarios, lo que permite la actualización de los paquetes de software y la participación abierta de todos aquellos que deseen colaborar. Constantemente hay personas que se unen a Debian para participar aportando su granito de arena, no solamente haciendo paquetes de programas, sino colaborando en el Servidor Web, traduciendo documentación de Debian, documentando fallos, o ayudando a los usuarios a través de las listas de correo que mantiene la comunidad. (16)

Se decidió usar Debian porque es una distribución (distribución de GNU/Linux) de desarrollo muy estable, por lo que los paquetes que se desarrollen en él y quieran ejecutarse utilizando cualquier

distribución siempre serán estables. A diferencia de otras distribuciones tiene un magnífico soporte de estabilidad en las aplicaciones (no requieren ser compiladas en la máquina que las esté usando).

1.5 Conclusiones

Hasta el momento se les ha dado solución a varios objetivos específicos que se propusieron en la introducción del presente trabajo de diploma.

- Se analizaron los sistemas de control, supervisión y adquisición de datos.
- Se realizó un análisis sobre el proceso de configuración de los módulos del SCADA Guardián del ALBA.
- Se definieron las herramientas y tecnologías que se van a utilizar.

Capítulo 2: Características del Sistema

2.1 Introducción

En este capítulo se obtiene una visión más específica del sistema, donde se explica cómo se lleva a cabo la configuración de los módulos y la adquisición de datos en el Guardián del ALBA. Además, se identifican los conceptos del dominio del problema. Y por último se exponen los requisitos funcionales y no funcionales que regirán el desarrollo de la solución al problema.

2.2 Proceso de Configuración de los módulos del SCADA Guardián del ALBA

El proceso de configuración de los módulos del SCADA Guardián del ALBA es realizado en dos etapas fundamentales. La primera, dentro del subsistema de visualización por el Ambiente de Configuración o Editor de Configuración, donde el operador puede definir a través de una interfaz gráfica los elementos asociados al proceso (módulos, variables,…) que desea monitorizar y controlar; y la segunda, realizada por el subsistema de configuración, encargado de suministrar y almacenar la configuración de los módulos antes realizada en el Editor.

A continuación se dará una breve explicación de como ocurre este proceso en el Editor de Configuración. Para un esclarecimiento sobre el funcionamiento del subsistema de configuración y más detalles del Editor consultar la bibliografía (17).

2.2.1 Editor de Configuración

El ambiente de configuración o Editor, permite a un mantenedor definir el ambiente de trabajo SCADA, adaptándolo mejor a la aplicación particular que se desee desarrollar. En el ambiente de configuración actual, se pueden crear varios nodos y cada nodo puede contar con varios módulos, como el módulo IHM donde se crean los despliegues sobre los cuales se configuran los objetos gráficos (widgets) necesarios para representar la información al operador.

Se lleva a cabo también la asociación de estos objetos gráficos (widgets) con las variables que representan y se permite modificar las propiedades de los elementos gráficos y los despliegues. Se configuran los módulos de históricos. Se posibilita la configuración completa del módulo de seguridad, donde se administran los grupos de transferencia a histórico, los perfiles y usuarios. También se configura el módulo de configuración, donde se especifica la forma en que debe trabajar el servicio que distribuye la configuración centralizada que se encarga de distribuir los datos necesarios para el funcionamiento de todos los módulos de forma individual. (17)

Después de que todos estos elementos y módulos son configurados, o se modifica una configuración, se envía al Módulo de Configuración a través del Middleware[2] para que sea distribuida a sus destinatarios y comience a funcionar el sistema con la nueva configuración.

2.2.2 Configuración del Módulo de Adquisición de Datos

En un sistema de supervisión y control, es vital la adquisición y el procesamiento de los datos del campo. Esta funcionalidad permite adquirir (obtener y almacenar) los datos de los dispositivos de control (PLCs, RTU, entre otros.) conectados al SCADA. El sistema controlará cuándo y con qué frecuencia se obtendrán los datos de los dispositivos de campo de acuerdo con los parámetros configurados.

La adquisición de datos no es más que, recoger, procesar y almacenar, en tiempo real, información relevante sobre la evolución del proceso productivo. En la arquitectura del Guardián del ALBA, la adquisición se divide en dos partes fundamentales: una, la recolección sin procesamiento, de la cual se encarga el recolector usando los manejadores o drivers; y la otra, del procesamiento, que es la llamada BDTR (Base de Datos en Tiempo Real).

Los manejadores o drivers, se encargan de la adquisición de los datos traduciendo los protocolos de campo a un protocolo genérico (Interfaz Genérica) que utiliza el SCADA Guardián del ALBA. Como regla, se programan en módulos independientes al sistema para evitar la re-compilación del sistema antes de ser agregados los nuevos protocolos. Cada manejador es específico para un protocolo

[2]En este contexto Middleware hace referencia capa de software intermedia que permite la comunicación a través del protocolo de control de transporte (TCP), entre las estaciones remotas (RTU) y las estaciones maestras (MTU).

determinado y lleva una configuración diferente, pero tiene un sistema que expone los parámetros que necesitan configurarse.

El recolector es responsable de adquirir y modificar puntos, asociados a dispositivos de campo, utilizando para ello los manejadores de protocolo. La planificación de las tareas de lectura se hace en función del tiempo de consulta de cada bloque de interrogación. La información adquirida por los manejadores usados por el recolector, es entregada a la Base de Datos de Tiempo Real.

La Base de Datos Tiempo Real está encargada de manejar todo lo referente a la recepción, procesamiento[3] 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 Guardián del ALBA.

Anteriormente, el Editor de Configuración poseía un diseño y arquitectura poco flexible y no orientada a extensiones dinámicas (plugins), trayendo como consecuencias la modificación y re-compilación del código para configurar nuevos módulos adicionados al SCADA. Debido al cambio a una arquitectura más robusta, flexible y orientada a extensiones del Editor, el sistema se ve afectado, porque no cuenta con la posibilidad de configurar el módulo de adquisición de datos.

Sin esta extensión el Guardián del ALBA no sabe qué recolectar, ni cada qué tiempo, ni qué manejador se va a usar para recolectar la información del campo, ni qué procesamiento se le van hacer a esos datos.

[3] Conversión de datos a unidades de ingeniería estándar y normalización de datos.

2.3 Propuesta de Solución

Para la configuración de los despliegues, el SCADA Guardián del ALBA posee un Editor, el cual mediante extensiones permite que distintos módulos sean configurados, pero no cuenta con una extensión capaz de configurar la adquisición de datos, es por eso que se hace inevitable desarrollarla.

Esta extensión debe ser capaz de configurar la recolección, de la siguiente manera:

1. Primero: se configuran los protocolos que se quieren usar para la adquisición de datos del campo. Esto no es más que configurar cada driver o manejador implementado en el SCADA, se debe hacer dinámico, porque son muchos manejadores y debe servir para el desarrollo futuro de más manejadores, es decir, se debe ajustar a ello. La configuración aquí depende del tipo de protocolo[4], cada protocolo tiene una configuración particular.
2. Segundo: se configuran las variables o puntos que se quieren leer o escribir y se le asocia a un protocolo en particular, porque las variables se adquieren por un protocolo en particular. Las variables tienen como configuración (Frecuencia, tipo de dato y dirección de lectura y escritura).
3. Tercero: se configura el procesamiento que se le va a hacer a esa variable. Esto sí es fijo y es igual para cualquier variable, debido a que los procesamientos son limitados y se sabe cuántos tipos de procesamientos se le hacen a dichas variables.

No se logró identificar procesos del negocio ya que no se tenían fronteras bien establecidas donde se pudieran ver claramente, quienes eran las personas que lo iniciaban, quienes eran los beneficiados con cada uno de los procesos, además quienes eran las personas que desarrollaban las actividades en cada uno de estos. Como no se alcanzó lo planteado entonces se decidió identificar conceptos, darle definiciones a estos conceptos y relacionarlos en un modelo conceptual.

[4] En este contexto los protocolos se utilizan para hacer referencia tanto a los protocolos a nivel físico (RS-232)del modelo OSI como a los del nivel de aplicación (Modbus)

2.4 Modelo Conceptual

Como parte de la investigación del dominio del problema (la configuración de la adquisición de datos) se desarrolla el modelo conceptual, el cual permite representar cada uno de los conceptos significativos (cosas del mundo real), las relaciones existentes y los atributos que los componen. En la figura 4 se muestra una representación gráfica del modelo a través de un diagrama de clases UML, y luego se explican cada uno de sus conceptos y los atributos de estos conceptos. Para un mayor entendimiento puede ir al anexo 1 donde se pueden encontrar imágenes de cada paquete por separado.

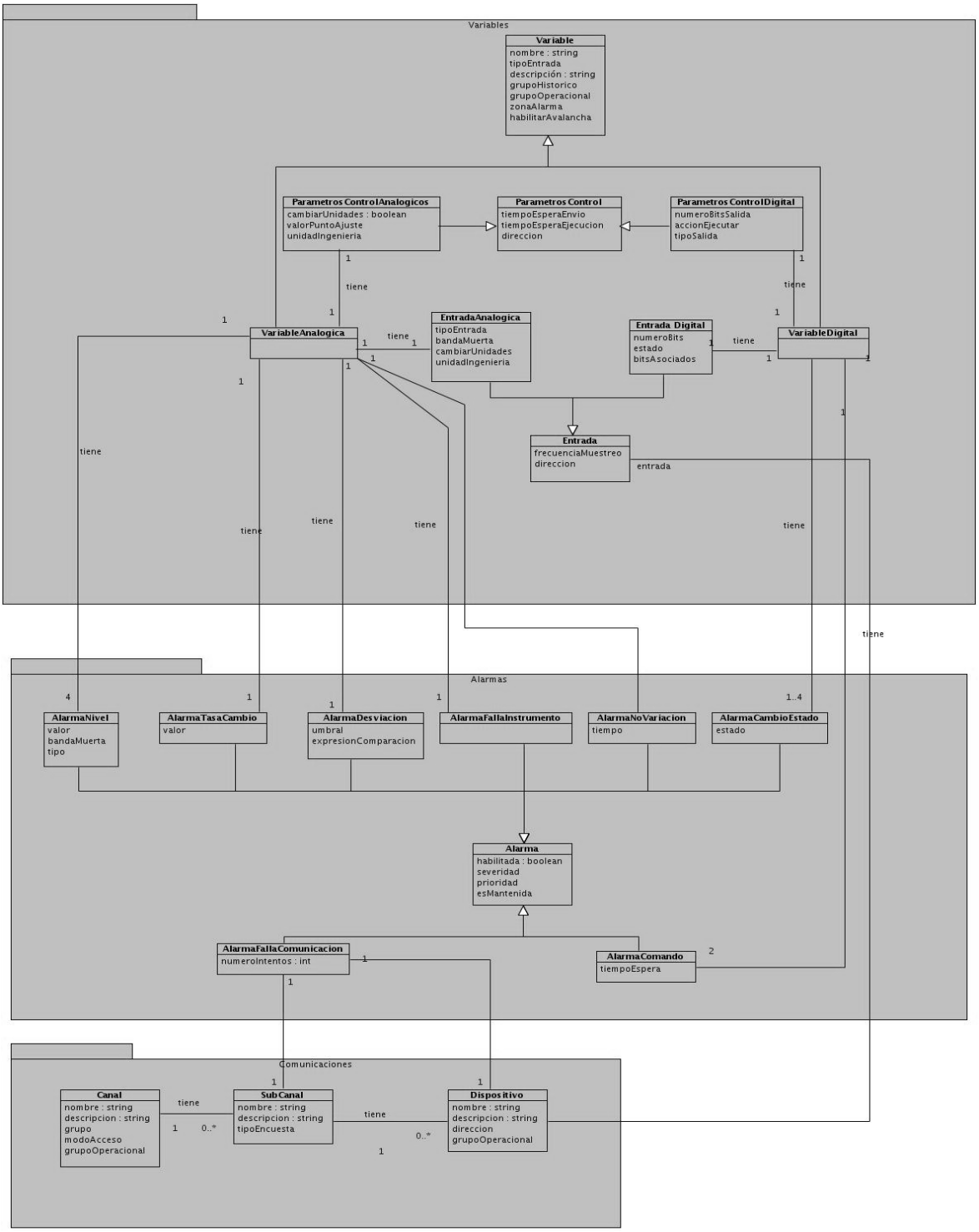


Fig. 4 Modelo conceptual.

2.4.1 Descripciones de las entidades del Modelo Conceptual

Las descripciones de las entidades del modelo conceptual presentado anteriormente pueden ser vistas en el anexo 2. Para ello se utilizan las plantillas de la metodología PDGP-UCID.

2.5 Requerimientos de la solución

El propósito de la definición de requisitos es especificar las condiciones o capacidades que el sistema debe cumplir y las restricciones bajo las cuales debe operar, logrando un entendimiento entre el equipo de desarrollo y el cliente, y especificando las necesidades reales de forma que satisfaga sus expectativas.

2.5.1 Requisitos funcionales

Los requisitos funcionales constituyen las capacidades o condiciones que el sistema debe tener, o sea, las utilidades expuestas por la solución para los usuarios, los cuales deben redactarse en el lenguaje del cliente. En este caso el usuario será el Mantenedor o el Administrador de las configuraciones de los despliegues del Guardián del ALBA. A continuación se listan estas funcionalidades:

1. Administrar variables digitales: el sistema debe permitir gestionar la información de las variables digitales correspondientes a un despliegue a través de una interfaz gráfica de usuario.
    1.1. Agregar una variable digital.
        1.1.1. Definir la información general de las variables como el nombre, la descripción, el grupo de privilegios al que está asociada y el tipo de entrada.
        1.1.2. Definir los parámetros de entrada como el dispositivo de control, la frecuencia de muestreo, el número de bits de entrada, la dirección de entrada, el estado y los bits asociados
        1.1.3. Definir los parámetros de control como la dirección, el tiempo de espera por envío de comando, el tiempo de espera por ejecución de comando, el número de bits de salida, la acción a ejecutar y el tipo de salida
    1.2. Modificar los parámetros de una variable.
    1.3. Eliminar una variable.

2. Administrar variables analógicas: el sistema debe permitir gestionar la información de las variables digitales correspondientes a un despliegue a través de una interfaz gráfica de usuario.
   2.1. Agregar una variable analógica.
      2.1.1. Definir la información general de las variables como el nombre, la descripción, el grupo de privilegios al que está asociada y el tipo de entrada.
      2.1.2. Definir los parámetros de entrada como el dispositivo de control, la frecuencia de muestreo, la dirección de entrada, el estado, el tipo de entrada, la banda muerta, cambiar unidades de campo a unidades de ingeniería y las unidades de ingeniería
      2.1.3. Definir los parámetros de control como el tiempo de espera por envío de comando, el tiempo de espera por ejecución de comando, la dirección de salida, cambiar unidades de campo a unidades de ingeniería, las unidades de ingeniería y el valor del punto de ajuste.
   2.2. Modificar los parámetros de una variable analógica.
   2.3. Eliminar una variable analógica.

3. Administrar canales: el sistema debe permitir gestionar la información de los canales correspondientes a un despliegue a través de una interfaz gráfica de usuario.
   3.1. Agregar un canal.
      3.1.1. Introducir el nombre, la descripción, el grupo de privilegios al que está asociado el canal y el modo de acceso disponible
   3.2. Modificar los parámetros de un canal.
   3.3. Eliminar un canal.

4. Administrar sub-canales: el sistema debe permitir gestionar la información de los sub-canales correspondientes a un despliegue a través de una interfaz gráfica de usuario.
   4.1. Agregar un sub-canal.
      4.1.1. Introducir el nombre, la descripción y el tipo de encuesta.
   4.2. Modificar los parámetros de un sub-canal.
   4.3. Eliminar un sub-canal.

5. Administrar dispositivos de control: el sistema debe permitir gestionar la información de los dispositivos correspondientes a un despliegue a través de una interfaz gráfica de usuario.
   5.1. Agregar un dispositivo.

5.1.1. Introducir el nombre, la descripción, el grupo de privilegios, y la dirección del controlador.

5.2. Modificar los parámetros de un dispositivo.

5.3. Eliminar un dispositivo.

6. Configurar alarma de desviación: el sistema debe permitir definir y configurar la alarma de desviación programable para variables analógicas a través de una interfaz gráfica.

   6.1. Definir los valores.

7. Configurar alarma de falla de instrumento: el sistema debe permitir definir y configurar la alarma de falla de instrumento asociada a una variable analógica a través de una interfaz gráfica.

   7.1. Definir los valores.

8. Configurar alarmas de nivel: el sistema debe permitir definir y configurar todos los parámetros comunes de las alarmas de nivel asociadas a una variable analógica.

   8.1. Definir los valores.

9. Configurar alarmas de tasa de cambio: el sistema debe permitir definir y configurar la alarma de tasa de cambio asociada a una variable analógica, a través de una interfaz gráfica.

   9.1. Definir los valores.

10. Configurar alarmas de no variación en el tiempo: el sistema debe permitir definir y configurar la alarma de no variación con el tiempo asociado a una variable analógica, a través de una interfaz gráfica.

    10.1. Definir los valores.

11. Configurar alarmas de cambio de estado: el sistema debe permitir definir y configurar las alarmas de estados asociadas a una variable digital a través de una interfaz gráfica.

    11.1. Definir los valores.

12. Configurar alarmas de falla de ejecución de comando: el sistema debe permitir definir y configurar las alarmas de falla de ejecución de comando asociadas a un variable digital a través de una interfaz gráfica.
    12.1. Definir los valores.

13. Configurar alarmas de falla de comunicación: el sistema debe permitir definir y configurar las alarmas de falla de comunicación asociadas a un dispositivo de control o a un sub-canal a través de una interfaz gráfica.
    13.1. Definir los valores.

Las descripciones de los requisitos funcionales del sistema se pueden encontrar en el anexo 3. Para ello se utilizan las plantillas de la metodología PDGP-UCID.

2.5.2 Requisitos no funcionales

Los requerimientos no funcionales especifican las propiedades o cualidades que debe tener la solución a desarrollar. Representan las características que hacen al producto atractivo, usable, rápido o confiable. Estos requisitos pueden marcar la diferencia entre un producto bien aceptado y otro con poca aceptación. A continuación se listan estas características:

Requerimientos de Usabilidad

- Podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.

Requerimientos de Restricciones en el diseño e implementación

- El sistema debe de ser desarrollado en el lenguaje de programación C++.
- Las interfaces gráficas de usuarios deben de ser desarrolladas utilizando el marco de trabajo Qt.
- El sistema debe de cumplir con las especificaciones de las extensiones del ambiente de configuración del Guardián del ALBA.
- El sistema debe de ser compilado a una biblioteca dinámica.

Requerimientos de ayuda y documentación

- Cada una de las etapas del ciclo de vida del proyecto deberá contar con una documentación según la metodología establecida.

Requerimientos de Apariencia o interfaz externa

- La interfaz debe ser de fácil comprensión en su funcionamiento permitiendo la utilización del sistema sin mucho entrenamiento, es decir, de fácil uso y con rápida respuesta del sistema.
- La interfaz contará con menús desplegables que faciliten y aceleren su utilización.
- Interfaces uniformes y con los mismos colores y diseños.
- Se debe garantizar que los colores de la interfaz de la aplicación sean claros.
- Imágenes claras y con la correcta visualización de su contenido.
- Mensajes sin ambigüedades.

Requerimientos de licencias y patentes

- Se deben utilizar herramientas libres.

Requerimientos de Portabilidad

- El sistema debe funcionar en sistemas de la familia GNU/Linux y Windows.

Requerimientos Políticos culturales

- Debe respetar los términos empleados normalmente, por los especialistas en el tema de la esfera que se automatiza.

Requerimientos Legales

- Todos los derechos sobre la propiedad intelectual de los diseños, entregables y manuales que se generen serán propiedad del cliente.
- Se certificará contra acta de aceptación del cliente de la etapa concluida con la calidad esperada, esto quedará debidamente aprobado dentro de los anexos del contrato que se firme.
- El mantenimiento correctivo (errores en la implantación) debe ser resuelto sin costo adicional.
- Se garantizará la protección de los datos, privacidad y derecho a la información.

Requerimientos de Seguridad

- Cuando se intente realizar cualquier acción irreversible, existirá una opción de advertencia antes realizar dicha acción.

Requerimientos de Software

- Se debe contar con el Editor de Configuración del SCADA Guardián del ALBA.

Requerimientos de Hardware

- Procesador: 1.00 GHz.
- Memoria RAM: 128 MB.
- Disco Duro: 20 MB.

2.5.3 Prototipo de Interfaz de Usuario

Los prototipos de interfaz de usuario ayudan a identificar, comunicar y probar un producto antes de crearlo. La realización de los mismos, logra un entendimiento común entre el cliente y los desarrolladores. Estos constituyen una presentación de la interfaz del producto que representan las funcionalidades que se han definido; de manera que permita que el usuario verifique que el sistema va a satisfacer sus necesidades. En la figura 5 se muestra una representación del prototipo de interfaz de usuario.

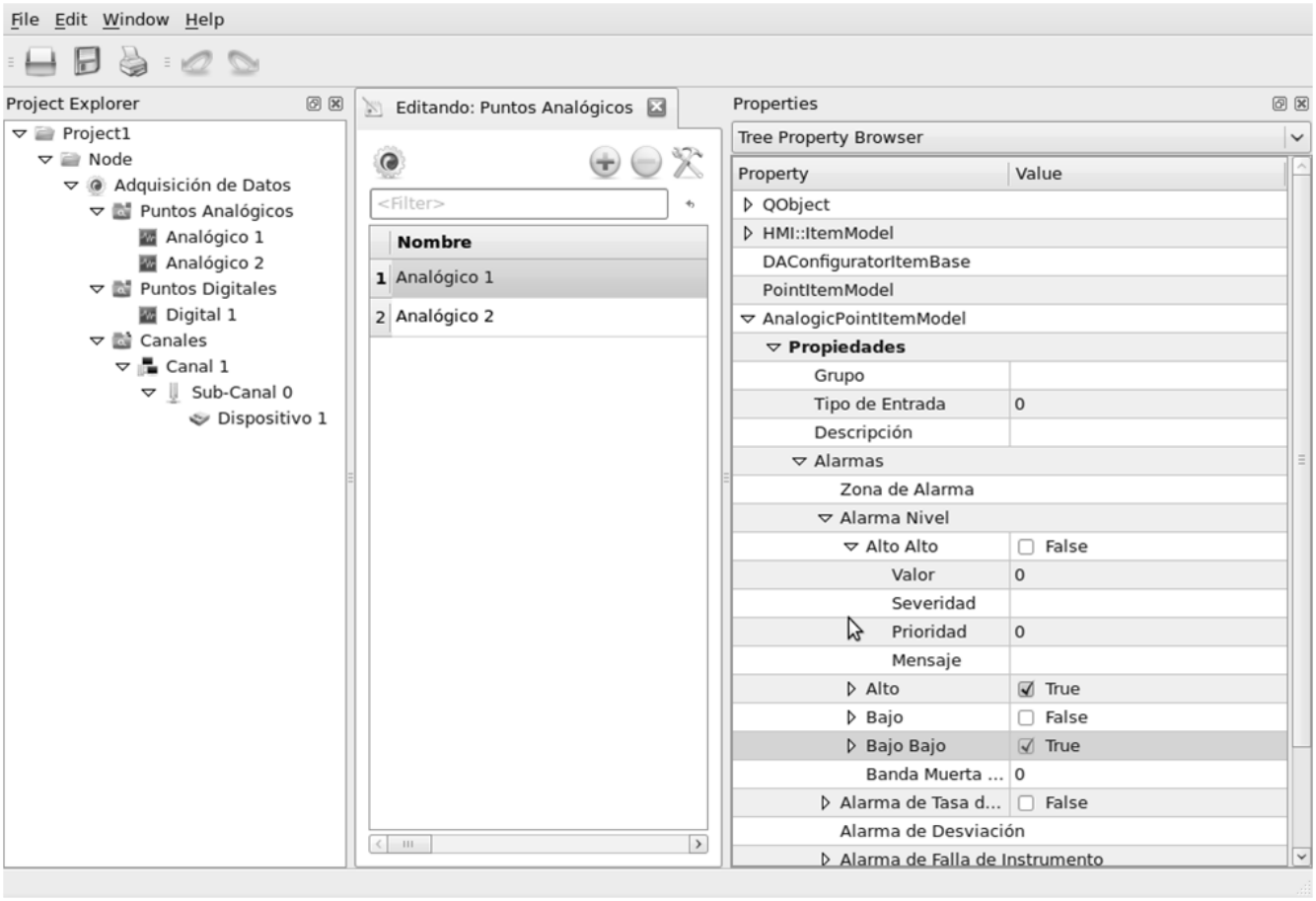


Fig. 5 Prototipo de interfaz de usuario.

2.6 Conclusiones

En este capítulo se ha logrado entender a profundidad el dominio del problema, así como se han identificado los conceptos alrededor del mismo. Todo lo anterior nos permitió definir cuales son las funcionalidades que debe tener el sistema software a implementar y avanzar hacia el siguiente capítulo.

Capítulo 3: Diseño del Sistema

3.1 Introducción

En el presente capítulo se presentará una descripción de los elementos de la arquitectura; los diagramas de interacción, específicamente los de secuencia por cada escenario de los requisitos funcionales, así como el diagrama general de clases del diseño, la descripción de las clases implicadas y los patrones de diseño aplicados.

3.2 Descripción de los elementos de la arquitectura

La arquitectura es el esqueleto o base de una aplicación. Se necesita una arquitectura para comprender el sistema, organizar el desarrollo y hacer evolucionar el sistema.

3.2.1 Estilos arquitectónicos

Un estilo arquitectónico define un conjunto de principios que le dan forma o rigen el diseño del sistema a desarrollar. Estos principios van encaminados a como interactúan y están estructurados los componentes o módulos del sistema, así como las responsabilidades de cada uno.

La arquitectura de un sistema de software casi nunca está atada a un solo estilo arquitectónico, sino que es más bien una combinación de estos estilos los que dan como resultado la arquitectura final del sistema. En el presente trabajo se asumió como principal el estilo de Presentación Separada, cuyo objetivo es separar la lógica de manejo de las interacciones del usuario con las interfaces gráficas y de los datos que son usados durante estas interacciones, permitiendo el simple mantenimiento de las interfaces gráficas.

3.2.2 Patrones arquitectónicos

Un patrón arquitectónico al igual que un estilo arquitectónico define un conjunto de principios para el diseño de la arquitectura, pero difieren de estos en que el alcance es menor, pues solo se concentran

en un aspecto de la arquitectura (por ejemplo el manejo de la concurrencia y la distribución). Los patrones se usan en conjunto con un estilo arquitectónico para determinar la forma de la estructura general del sistema.

La elección del estilo de Presentación Separada, y las características mencionadas a lo largo del trabajo conducen a la combinación de 2 patrones arquitectónicos que son variaciones en el modo de funcionar del legendario Modelo Vista Controlador (MVC). Primeramente se dará una breve introducción del MVC y los principales roles que puede tener una clase (componentes) dentro de este patrón, para luego hacer llegar los patrones utilizados.

El MVC divide una aplicación en tres componentes o roles principales:

- Modelo: es la representación específica de la información con la cual el sistema opera, es decir, sobre la cual funciona la aplicación. No depende de ninguna vista y de ningún controlador.
- Vista: maneja la visualización de la información. Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: controla el flujo entre la vista y el modelo(los datos). Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Es el encargado de realizar la lógica específica del negocio.

Teniendo en cuenta los roles mencionados anteriormente se pueden explicar los patrones que se combinaron para desarrollar la extensión:

- Modelo de Presentación: plantea una forma de almacenar el estado y el comportamiento de una presentación (vista) de forma independiente de los controles utilizados en una Interfaz Gráfica de Usuario. En esencia el patrón sugiere tener una clase independiente de la vista que almacene los datos no específicos del dominio y, los comportamientos de la interfaz de usuario llamado modelo de presentación (controlador), pero sin ninguno de los controles u objetos usados en dicha interfaz. (18)

- Vista Pasiva: en este patrón la vista juega un rol completamente pasivo y no es responsable de actualizarse desde el modelo. Como resultado se tiene toda la lógica de la vista en el controlador,

y la inexistencia de dependencias entre las vistas y los modelos. Por lo tanto el controlador es el encargado de mediar entre las vistas y los modelos. (19)

En la solución propuesta se combinan todos estos roles, dando como resultados los siguientes: las vistas, que jugarán el mismo rol del patrón Vista Pasiva; los modelos de presentación, que realizarán las funciones como el rol del controlador en el patrón Vista Pasiva y el patrón Modelo de Presentación; y por último los modelos del dominio, que se desempeñaran como simples modelos. La siguiente figura muestra un diagrama de paquetes confeccionados con elementos de UML y esboza como queda organizada las dependencias entre los paquetes que representan cada uno de los roles antes mencionados. Para una mayor comprensión de la estructuración de las clases puede consultar el anexo 4.

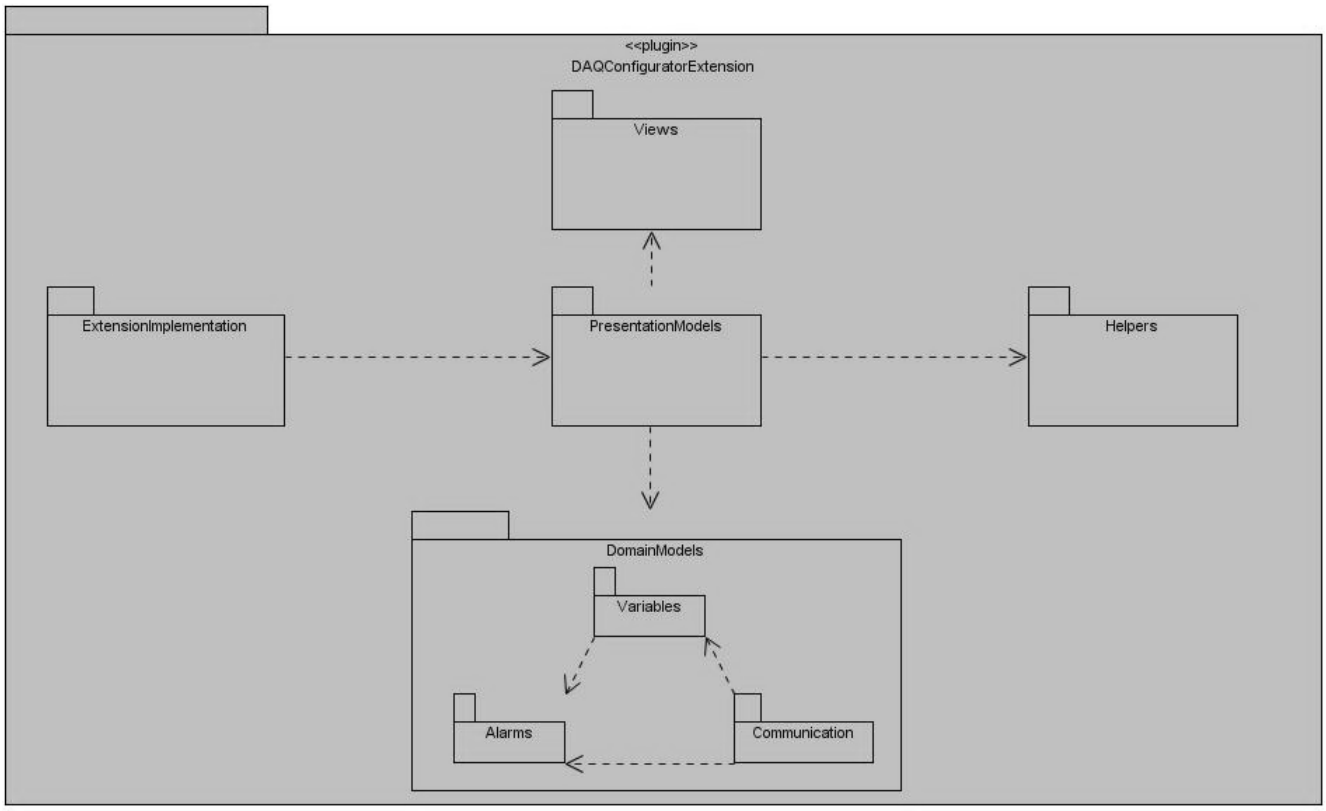


Fig. 6 Vista de paquetes de la extensión.

3.3 Definiciones de diseño que se aplican

3.3.1 ¿Qué es un patrón?

Un sistema bien estructurado está lleno de patrones, pero ¿qué es un patrón? Un patrón:

- Es una pareja de problema/solución con un nombre, que codifica (estandariza) buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. (20)

  Amplio repertorio de principios generales basados en la experiencia que guían la creación de un software.
- Es una solución a un problema en un contexto.
- Codifica conocimiento específico acumulado por la experiencia en un dominio.

Existen varias categorías o clasificaciones de patrones entre las que se encuentran los patrones arquitectónicos, patrones de diseños, entre otros. En esta sección se hará una descripción de cuales fueron los patrones de diseños utilizados en la solución.

Primeramente, un patrón de diseño es una abstracción de una solución en un alto nivel que constituye la respuesta a un problema de diseño no trivial efectiva y reusable.

Cuando se tienen conocimientos sobre el comportamiento y los beneficios de los patrones de diseño, uno de los errores más comunes que suele cometerse es querer aplicarlos en todo momento en el sistema que se esté desarrollando, provocando así su uso excesivo, muchas veces sin realizar un previo análisis para determinar si sería efectivo realmente y aportaría la reusabilidad, extensibilidad y mantenimiento que se necesita en la solución desarrollada.

Con el propósito de lograr la simplicidad y eficiencia en el sistema, luego de un estudio profundo para no incurrir en el error antes mencionado, se seleccionaron un conjunto de patrones de diseño que se utilizarán en el desarrollo de la extensión.

3.3.2 Patrones de Diseño GRASP

Los patrones GRASP (Patrones Generales de Asignación de Responsabilidades) son patrones de diseño que se usan para asignar responsabilidades a una clase. Se pueden destacar 5 patrones básicos (principales) y 4 patrones adicionales a aplicar en el diseño Orientado a Objetos. A continuación se muestran los más utilizados en la solución propuesta, con una breve descripción y ejemplos donde son aplicados.

- Experto: Asignar una responsabilidad al más competente en información, la clase cuenta con la información necesaria para cumplir la responsabilidad. Es el principio básico de asignación de responsabilidades que suele utilizarse en el diseño Orientado a Objetos. Es el patrón que más se usa para asignar responsabilidades.
  Utilización: Este patrón se puede observar en cada una de las clases pertenecientes a los modelos de presentación, pues estos son los encargados de llevar a cabo validaciones de los datos a ser almacenados en los objetos representantes de los modelos del dominio. Por ejemplo cuando en una instancia de la clase ChannelPresentationModel se le solicita que se elimine, ella es la encargada de decidir si se elimina o no.

- Bajo Acoplamiento: Asignar una responsabilidad para mantener un engranaje pobre. Es un principio que se debe recordar durante las decisiones de diseño; es la meta principal que es preciso tener siempre presente. Soporta el diseño de clases más independientes. Asigna las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible.
  Utilización: Las clases de las vistas y los modelos del dominio, son totalmente independientes unas de otras, o sea, no existe una relación directa entre estas clases debido a la separación en capas que hace el patrón arquitectónico Vista Pasiva, y del uso del patrón Indirección, por lo que se ve un bajo acoplamiento.

- Alta Cohesión: Asignar una responsabilidad de modo que la unión se mantenga a gran escala. Asignar a las clases responsabilidades que trabajen sobre una misma área de aplicación y que no tengan mucha complejidad. Mejoran la claridad y facilidad con que se entiende el diseño.

Utilización: Las clases de las vistas se encargan solamente de visualizar los datos, las clases de los modelos de presentación se encargan de validar los datos, almacenarlos en las instancias de las clases de los modelos de dominio; y las clases de los modelos de dominios se encargan de contener los datos.

- Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Asigna la responsabilidad del manejo de mensajes de los eventos de un sistema a una clase.
  Utilización: Este patrón es ampliamente usado en las clases de los modelos de presentación, las cuales son las encargadas de manejar los eventos generados por los usuarios sobre las vistas, por ejemplo Adicionar Canal, Eliminar Canal, entre otras.

- Creador: Asignarle a una clase la responsabilidad de engendrar (crear) una nueva instancia de otra clase; este patrón ayuda a identificar quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases.
  Utilización: Unas de las clases expertas en crear, son las clases pertenecientes a los modelos de presentación y que heredan de la clase DAQContainerPresentationModel, puesto que ellas son las encargadas de reaccionar antes los eventos de adicionar, y por lo tanto, deben responder ante ellos con la creación de un objeto del tipo que ellas contienen.

- Polimorfismo: Asignar las responsabilidades del comportamiento a los tipos en que el comportamiento presenta variantes cuando por el tipo varían las alternativas o comportamientos afines.
  Utilización: Se utiliza en las clases pertenecientes a los modelos de presentación que pueden contener otros modelos de presentación como hijos. El polimorfismo viene dado porque todas estas clases heredan de la clase abstracta DAQContainerPresentationModel, cuyo método abstracto onAdd() debe ser reimplementado, por ejemplo ChannelContainerPresentationModel reimplementa este método para crear un canal.

- Indirección: Asignar la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios y estos no terminen directamente acoplados.

Utilización: Este patrón se utiliza en los modelos de presentación pues estos actúan como intermediarios entre las vistas y los modelos del dominio.

3.3.3 Patrones de Diseño GOF

Los patrones GOF (Gand of Four, Banda de los Cuatro) son patrones de diseño que definen una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Estos patrones se dividen en tres categorías: los creacionales, los estructurales y los de comportamiento. Seguidamente se expondrán los patrones empleados:

- Instancia única (Singleton): este patrón consiste en garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella. (21)
  Utilización: se utiliza en la clase DriverInspector, pues solo es necesario que exista una instancia de este tipo y muchas otras puedan acceder a ella sin muchas complicaciones.

- Composite: este patrón permite construir objetos complejos componiendo de forma recursiva objetos similares en una estructura de árbol. En conclusión tiene como propósito componer objetos en estructuras jerárquicas para representar jerarquías parte-todo, donde los clientes pueden tratar de la misma forma a los objetos individuales o compuestos. (21)
  Utilización: este patrón es usado en las clases de los modelos de presentación pues estas pueden contener otros modelos de presentación como hijos.

3.4 Diagramas de secuencia y diagramas de clases para casa requisito funcional

Los diagramas de secuencia muestran las interacciones entre los objetos, ordenadas en secuencias temporales durante un escenario concreto, mientras que los diagramas de clases del diseño tienen como objetivo mostrar la estructura de la solución planteada en un lenguaje de programación específico.

Con el objetivo de explicar mejor la estructura y el comportamiento del sistema, se elaboraron los diagramas de secuencia por cada uno de los escenarios y de clases por cada requisito funcional. Para ver todos estos diagramas consulte el anexo 5.

3.5 Descripción de las clases

Existe una traza directa desde los requisitos funcionales del capítulo 2, a clases del diseño. Por cada requisito se generan 3 clases para desempeñar cada unos de los roles mencionados anteriormente, o sea, una clase para la vista, otra para el modelo de presentación (controlador) y otra para el modelo del dominio (modelo). Por ejemplo para el requisito funcional Administrar Canal se generaron las siguientes clases: ChannelView (vista de canal), ChannelPresentationModel (modelo de presentación de canal) y ChannelDomainModel (modelo de dominio de canal).

Debido a características similares presentadas en las vistas, se decidió crear una clase base llamada BaseView (vista base), la cual define controles y demás elementos básicos a todas las vistas. Por ejemplo, todas poseen botones para aceptar, cancelar o aplicar, los cuales utilizando el mecanismo de señales (signals) y ranuras (slots) del marco de trabajo Qt son conectados a los modelos de presentación correspondientes para realizar las operaciones requeridas.

Los modelos de presentación poseen un grupo de características y comportamientos que son comunes a todos los modelos de presentación por lo cual se platea una relación de generalización-especialización entre los modelos de presentación especializados y un modelo de presentación base como mecanismo de reutilización de código. Las generalizaciones están representadas por dos clases generales: DAQBasePresentationModel y DAQContainerPresentationModel.

DAQBasePresentationModel es la clase base de todos los modelos de presentación y hereda de la clase ItemModel del paquete IHM, permitiendo mediante esta herencia exponer en el explorador de proyectos el contenido visual -como los iconos- y, cada una de las acciones mostradas en el menú contextual desplegado cuando se ejecuta un clic derecho sobre un nodo del explorador de proyectos. En DAQBasePresentationModel se encuentran además, las ranuras (slots) que recibirán la notificación de las señales de los botones definidos en la vista base (BaseView) y que siguen un comportamiento similar para todos y, con ayuda de los métodos polimórficos "getView()",

"getIncrustedView()", "getDomainModel()", "store()" y "retrive()" encargados de crear o instanciar las vistas, los modelos de dominios, y como salvar y recuperar los datos respectivamente, se pueden generalizar estos comportamientos.

DAQContainerPresentationModel es la clase base de los modelos de presentación que contienen otros modelos de presentación, como es el ejemplo de la clase ChannelPresentationModel (modelo de presentación de canales que pueden contener SubChannelPresentationModel). Debido a la relación de herencia establecida con DAQBasePresentationModel y esta a su ves con ItemModel y, como esta última representa una estructura de árbol general (o sea, que cada nodo puede contener varios hijos y cada hijo puede contener más hijos y así sucesivamente), pues entonces se hizo idóneo colocar una ranura virtual pura o abstracta pura para ser reimplementada por los modelos de presentación contenedores y que sea invocada en cada uno de los eventos adicionar, además de colocar los elementos necesarios a mostrar en los menús desplegados cuando se ejecuta un clic derecho sobre estos nodos contenedores. En el siguiente diagrama UML se puede observar todo lo anteriormente dicho.

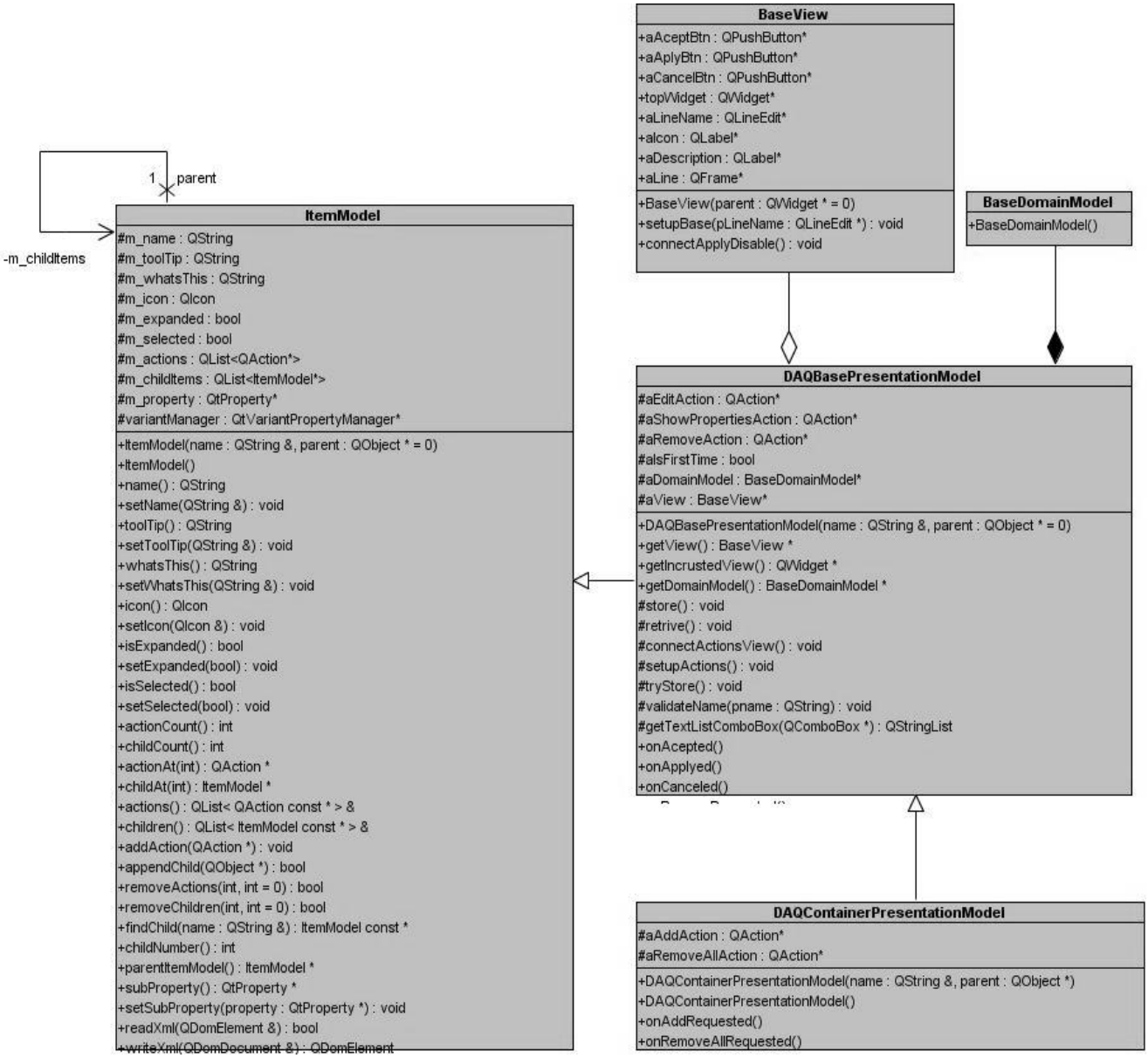


Fig. 7 Diagrama de clases general.

De los modelos de dominio solo mencionar que representan simples estructuras contenedoras de los datos o la información con la cual se trabaja en la aplicación. La más compleja es la de los sub-canales y dispositivos debido a que los parámetros a configurar están almacenados en las bibliotecas de enlace dinámico (Shared Object, Dinamyc Link Library) que son implementaciones de los manejadores del Guardián del ALBA, las cuales permiten ser inspeccionadas y así extraer toda la meta información de sus parámetros como: nombre del parámetro, tipo de dato, valor por defecto,

valor mínimo, valor máximo, descripción y una expresión regular para validar el valor que le es asignado al parámetro.

Las descripciones de las clases del diseño, así como sus principales atributos y funcionalidades pueden ser consultadas en el anexo 6.

3.6 Conclusiones

Mediante este capítulo se ha logrado desarrollar el diseño, donde su principal objetivo era realizar los diagramas de clases, así como la descripción de las mismas y los diagramas de secuencia correspondientes a cada uno de los escenarios definidos en los requisitos funcionales. Permitiendo esto dar paso a la realización posterior de la implementación y a la realización de las pruebas.

Capítulo 4: Implementación y Prueba

4.1 Introducción

En el presente capítulo se mostrará el diagrama de componentes y la matriz de integración. Esta etapa constituye la transición del diseño de clases a la creación de componentes físicos. Además, se realizarán los diseños de casos de prueba y se dará a conocer el método que se utilizó para realizar las pruebas correspondientes y los resultados que se obtuvieron al efectuar las mismas.

4.2 Diagrama de Componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente, se realizan por partes. Cada diagrama describe un apartado del sistema y se representa como un grafo de componentes software unidos por medio de relaciones de dependencia.

UML tiene dos vistas de componentes: vista de caja negra y vista de caja blanca. La vista de caja negra muestra los componentes desde una perspectiva exterior, mientras que, la vista de caja blanca muestra como un componente realiza una funcionalidad especificada por las interfaces que provee. A continuación se muestra una vista de caja blanca con los componentes fundamentales utilizados y creados en el desarrollo del presente trabajo.

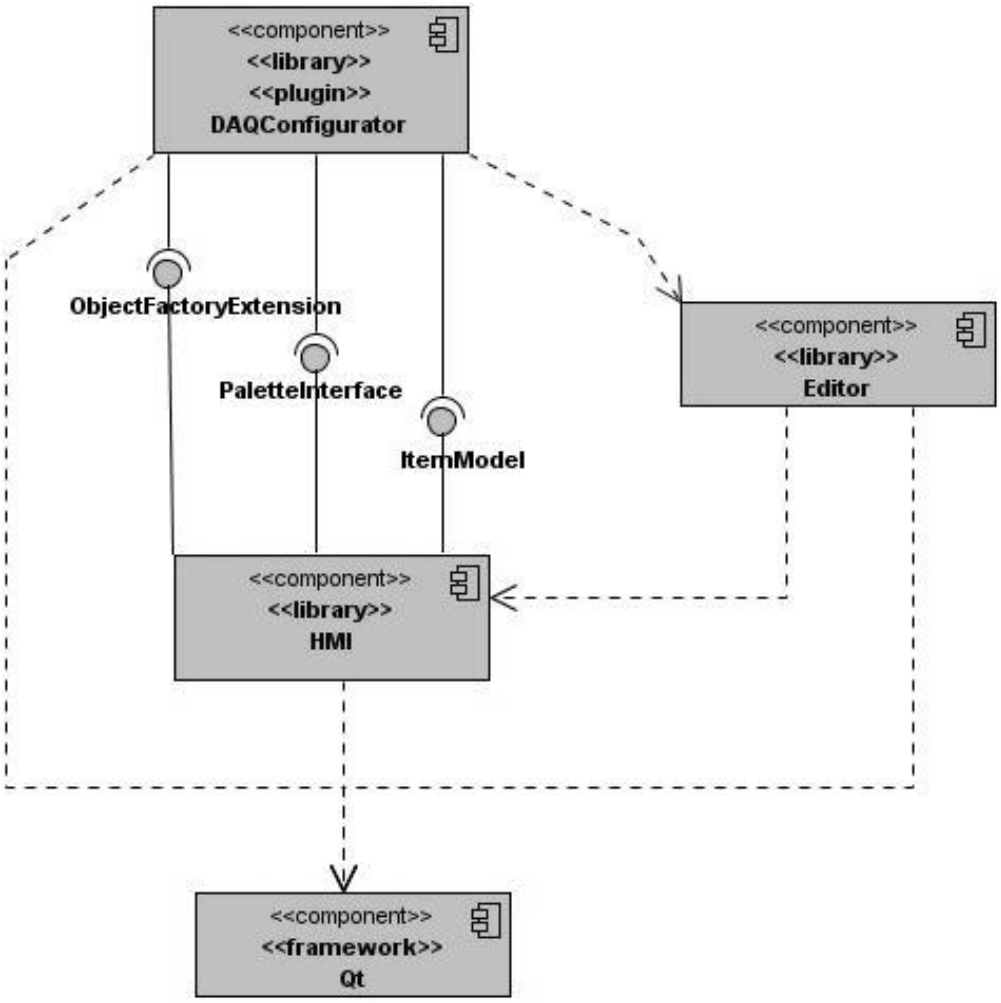


Fig. 8 Diagrama de componentes.

El componente DAQConfigurator representa la extensión desarrollada que después de ser adicionada al editor de configuración permite la configuración del subsistema de adquisición de datos del SCADA Guardián del ALBA. Para que DAQConfigurator funcione y pueda ser reconocido por el Editor necesita implementar la interfaz ObjectFactoryExtension, la cual define el contrato y el punto de entrada a la extensión.

Para registrar la extensión en la fábrica de extensiones del Editor se hace a través del punto de extensión o interface ObjectFactoryExtension. Además, para que cuente con una representación gráfica en la sección “Módulos” de la paleta de componentes del Editor se implementa la interfaz

PaletteInterface y, para ser observable en el explorador de proyectos se implementa la interfaz ItemModel que representa cada uno de los nodos del explorador de proyectos.

4.2.1 Matriz de Integración de Componentes

La siguiente matriz contiene todos los componentes definidos en el subsistema y especifica los servicios que consume el componente interno DAQConfigurator de los componentes externos IHM, Editor y framework Qt.

	Componentes Externos		
Componentes Internos	**IHM**	**Editor**	**Framework Qt**
DAQConfigurator	ObjectFactoryExtension PaletteInterface ItemModel	Utilización de la clase filtro para filtrar los elementos del modelo.	Creación de la Interfaz Gráfica de Usuario.

4.3 Diagrama de Despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Este diagrama es útil para ilustrar la arquitectura física de un sistema. A continuación se muestra una representación gráfica donde se encuentra el Editor situado en la PC cliente con la extensión adicionada.

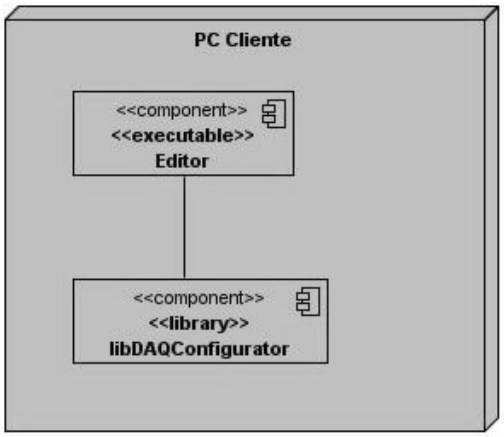


Fig. 9 Diagrama de despliegue.

4.4 Diseños de Casos de Prueba

Con los diseños de casos de prueba realizados, se pretende demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Para ver estos diseños remitirse al anexo 7.

4.5 Pruebas de Software

Estas pruebas constituyen una etapa imprescindible durante el proceso de desarrollo del software, pues permiten detectar y corregir el máximo de errores posibles antes de la entrega al cliente del software desarrollado, el éxito de las mismas puede mejorar la percepción de calidad del usuario final y lograr su satisfacción. (11)

Dentro de estas pruebas se encuentran las pruebas de unidad. La cual está enfocada a los elementos más pequeños del software. Mediante ésta sólo una unidad es probada como tal, como una clase o un subsistema. Consiste en una prueba estructural (o **caja blanca**) y una prueba de especificación (o **caja negra**).

4.5.1 Método de Prueba

Para la realización de las pruebas se utilizó el método de caja negra, el cual se centra principalmente en los requisitos funcionales del software y no consideran la estructura interna del programa; estas pruebas son hechas sin el conocimiento interno del producto. Se llevan a cabo sobre la interfaz del software y permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.

Para desarrollar las pruebas de caja negra existen varias técnicas pero las que se usaron fueron las siguientes; la técnica de “Partición de Equivalencia” y la de “Valores Extremos”, las cuales permiten:

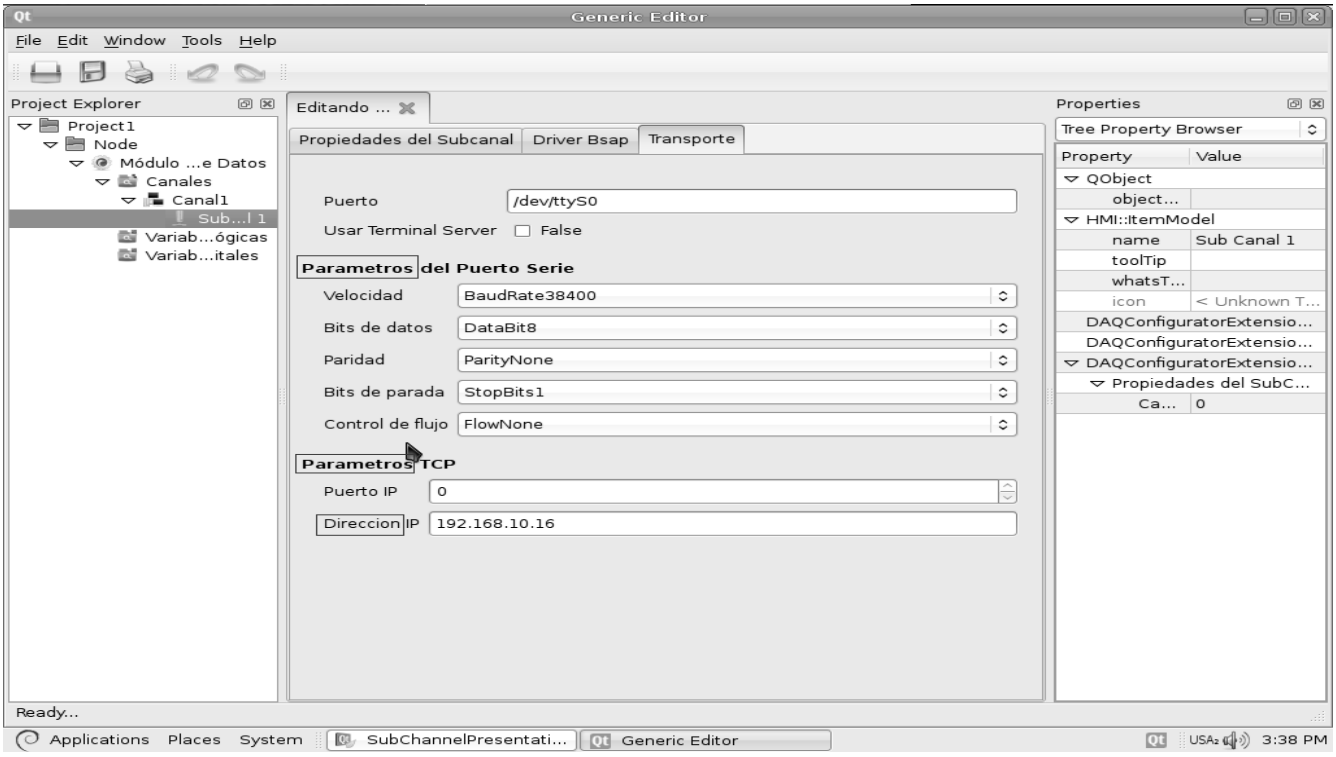
- Examinar los valores válidos e inválidos de las entradas existes en el software.
- Descubrir de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.
- Llevar a la elección de casos de prueba que ejerciten los valores límites o extremos.
- Detectar que los campos no se queden vacíos y que los valores introducidos no sean muy grandes.

4.6 Listado de las No Conformidades

A continuación se muestra una tabla con los errores que fueron detectados a la hora de realizar las pruebas.

No Conformidades	Acciones Correctivas
Errores de taxonomía. Se detectaron palabras secundarias con mayúsculas.	Resueltas en el momento en que se detectaron.
Errores de ortografía. Se detectaron palabras sin tildes.	No procede. El cliente así lo solicitó.

En la interfaz perteneciente a la funcionalidad Administrar Dispositivo el campo nombre aceptaba caracteres inválidos.	Resuelta en el momento en que se detectó.



4.7 Conclusiones

Con el desarrollo de este capítulo se han mostrado los diagramas de componentes y de despliegue propios de nuestra solución. Conjuntamente se realizaron las pruebas unitarias las cuales arrojaron resultados de importancia considerable. En general las pruebas de unidad ofrecieron valiosa información lo que ayudó a la corrección de los errores identificados y a implementar un producto con mayor calidad.

Conclusiones

Con el desarrollo de la extensión se ha logrado que el Sistema “SCADA Guardián del Alba” cuente con la posibilidad de configurar el módulo de Adquisición de Datos. Para obtener el software resultante se transitó por varias etapas: primero la investigación y selección de las tecnologías a emplear, luego la captura de los requisitos, en tercer lugar el diseño propuesto, seguido la implementación y por último se realizaron las pruebas de unidad.

En sentido general se obtuvieron resultados satisfactorios que dieron cumplimiento a los objetivos propuestos al inicio del presente trabajo de diploma:

- Luego de un profundo estudio sobre los sistemas SCADA se logró dominar al máximo todo lo referente al tema.
- Luego de analizar el proceso de configuración se logró comprender como se realiza este.
- Se definieron las herramientas y tecnologías software a utilizar.
- Se identificaron los requisitos del sistema para logar una solución más viable.
- Se obtuvo el diseño de la extensión mediante la aplicación de los patrones arquitectónicos y de diseño.
- Se logró implementar la extensión.
- Se integró la extensión desarrollada con el Editor de Configuración.
- Se realizaron las pruebas para validar los requerimientos implementados.
- Se evaluó el resultado de las pruebas diseñadas, donde se demostró que los requisitos se habían cumplido.

Recomendaciones

Tomando como base la investigación realizada y los resultados obtenidos durante la realización de este trabajo, se recomienda lo siguiente:

- Estudiar constantemente la evolución de la arquitectura del Editor con el objetivo de realizarle mejoras a la extensión ya que es un proceso iterativo e incremental.
- Comenzar a utilizar la extensión en el SCADA Guardián del Alba.

Trabajos citados

1. **Galá, María de las Nieves.** Unión de la Industria Militar: Modesta, pero eficiente. *Trabajadores.* [En línea] 2009. [Citado el: 2 de junio de 2009.] http://www.trabajadores.cu/news/union-de-la-industria-militar-modesta-pero-eficiente.

2. **González, Victor M.** *Curso de Supervisión y Control de Procesos.* s.l. : Universidad de Oviedo, 2003.

3. **Jorge Riverón, Osmany y Yera Calleja, Hilario.** *Modelación de un generador de informes para sistemas de supervisión, control y adquisición de datos.* Ciudad de la Habana : s.n., 2007.

4. **David Bailey BEng.** *Practical SCADA for Industry.* Primera Edición. Newnes : IDC Technologies, 2003. pág. 298.

5. **ALBA, Guardián del.** *Introducción a la Arquitectura de Guardián del ALBA.* Ciudad Habana : s.n., 2008.

6. Informatica 14th International Convention and FAIR 2011. *Guardián del Alba: Sistema de Supervisión y Control de procesos industriales para los pueblos del Alba.* [En línea] 2009. [Citado el: 19 de febrero de 2010.] http://www.informaticahabana.com/?q=en/node/787.

7. Autómatas Industriales. *SCADA.* [En línea] 2 de marzo de 2006. [Citado el: 3 de febrero de 2010.] http://www.automatas.org/redes/scadas.htm.

8. **Dr. Trujillo Codorniú, Rafael, y otros.** *Desarrollo de Sistema de Supervisión y Control de Procesos EROS V.5.8.*

9. Open SCADA. [En línea] [Citado el: 10 de 2 de 2010.] http://openscada.org/.

10. Raditex Control AB FreeSCADA. [En línea] [Citado el: 10 de 2 de 2010.] http://www.freescada.com.

11. **Defensa, Unidad de Compatibilización, Integración y Desarrollo de Software para la.** *Proceso de Desarrollo y Gestión de Proyectos de Software.* Ciudad Habana : s.n., 2009.

12. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley.

13. **Pérez Feria, Jordenys.** *Análisis de la arquitectura del módulo Interfaz Hombre Máquina del SCADA Nacional.* Ciudad de la Habana : s.n., 2008.

14. **Benítez Herrera, Alejandro y Saurín Ojeda, Gelson Rafael.** *Módulo para el comportamiento autónomo de autos en un Entorno Virtual urbano.* Ciudad de la Habana : s.n., 2008.

15. **González Abad, Israeldis.** *Diseño e Implementación del Sistema de Gestión de Información de los Recursos de la Facultad 3.* Ciudad de la Habana : s.n., 2008.

16. **Sehara Driggs, Yosell Luis.** *Implementación de un modelo para la configuración de un sistema SCADA.* Ciudad de la Habana : s.n., 2008.

17. **ALBA, Guardian del.** *Módulo de Configuración. Interfaz Gráfica de Configuración.* Ciudad de la Habana : s.n., 2008.

18. **Fowler, Martin.** Presentation Model. [En línea] [Citado el: 30 de 4 de 2010.] http://www.martinfowler.com/eaaDev/PresentationModel.html.

19. —. Pasive View. [En línea] 1555. [Citado el: 15 de 4 de 2010.] http://www.martinfowler.com/eaaDev/PasiveView.html.

20. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. 970-1 7-0261-1.

21. Patrones de Diseño. [En línea] García Molina, Jesús. [Citado el: 10 de 5 de 2010.] http://www.slidefinder.net/p/patrones_dise%C3%B1o_jes%C3%BAs_garc%C3%ADa_molina/7849657.

Bibliografía

1. **Galá, María de las Nieves.** Unión de la Industria Militar: Modesta, pero eficiente. *Trabajadores.* [En línea] 2009. [Citado el: 2 de junio de 2009.] http://www.trabajadores.cu/news/union-de-la-industria-militar-modesta-pero-eficiente.

2. **González, Victor M.** *Curso de Supervisión y Control de Procesos.* s.l. : Universidad de Oviedo, 2003.

3. **Jorge Riverón, Osmany y Yera Calleja, Hilario.** *Modelación de un generador de informes para sistemas de supervisión, control y adquisición de datos.* Ciudad de la Habana : s.n., 2007.

4. **David Bailey BEng.** *Practical SCADA for Industry.* Primera Edición. Newnes : IDC Technologies, 2003. pág. 298.

5. **ALBA, Guardián del.** *Introducción a la Arquitectura de Guardián del ALBA.* Ciudad Habana : s.n., 2008.

6. Informatica 14th International Convention and FAIR 2011. *Guardián del Alba: Sistema de Supervisión y Control de procesos industriales para los pueblos del Alba.* [En línea] 2009. [Citado el: 19 de febrero de 2010.] http://www.informaticahabana.com/?q=en/node/787.

7. Autómatas Industriales. *SCADA.* [En línea] 2 de marzo de 2006. [Citado el: 3 de febrero de 2010.] http://www.automatas.org/redes/scadas.htm.

8. **Dr. Trujillo Codorniú, Rafael, y otros.** *Desarrollo de Sistema de Supervisión y Control de Procesos EROS V.5.8.*

9. Open SCADA. [En línea] [Citado el: 10 de 2 de 2010.] http://openscada.org/.

10. Raditex Control AB FreeSCADA. [En línea] [Citado el: 10 de 2 de 2010.] http://www.freescada.com.

11. **Defensa, Unidad de Compatibilización, Integración y Desarrollo de Software para la.** *Proceso de Desarrollo y Gestión de Proyectos de Software.* Ciudad Habana : s.n., 2009.

12. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar.** *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley.

13. **Pérez Feria, Jordenys.** *Análisis de la arquitectura del módulo Interfaz Hombre Máquina del SCADA Nacional.* Ciudad de la Habana : s.n., 2008.

14. **Benítez Herrera, Alejandro y Saurín Ojeda, Gelson Rafael.** *Módulo para el comportamiento autónomo de autos en un Entorno Virtual urbano.* Ciudad de la Habana : s.n., 2008.

15. **González Abad, Israeldis.** *Diseño e Implementación del Sistema de Gestión de Información de los Recursos de la Facultad 3.* Ciudad de la Habana : s.n., 2008.

16. **Sehara Driggs, Yosell Luis.** *Implementación de un modelo para la configuración de un sistema SCADA.* Ciudad de la Habana : s.n., 2008.

17. **ALBA, Guardian del.** *Módulo de Configuración. Interfaz Gráfica de Configuración.* Ciudad de la Habana : s.n., 2008.

18. **Fowler, Martin.** Presentation Model. [En línea] [Citado el: 30 de 4 de 2010.] http://www.martinfowler.com/eaaDev/PresentationModel.html.

19. Pasive View. [En línea] 1555. [Citado el: 15 de 4 de 2010.] http://www.martinfowler.com/eaaDev/PasiveView.html.

20. **Larman, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999. 970-1 7-0261-1.

21. Patrones de Diseño. [En línea] García Molina, Jesús. [Citado el: 10 de 5 de 2010.] http://www.slidefinder.net/p/patrones_dise%C3%B1o_jes%C3%BAs_garc%C3%ADa_molina/7849657.

22. **Rodriguez Penin, Antonio.** *Sistemas SCADA.* Madrid : Marcombo, 2007. pág. 447.

23. **Pérez Pupo, Iliana y Argota Vega, Irina Elena.** *Desarrollo del flujo de requisitos para el subsistema de Gráficos Vectoriales del producto SCADA Nacional.* Ciudad de la Habana, : s.n., Mayo 2007.

24. **Michael Leslie, P. Eng.** *Computer Applications in Power Systems - SCADA.* s.l. : IEEE Ottawa Section, 2003.

25. **González Barahona, Jesús, Seoane Pascual, Joaquín y Robles, Gregorio.** *Introducción al Software Libre.* Primera. Barcelona : Eureca Media, 2003. pág. 340.

26*An Introduction to Design Patterns in C++ with Qt 4*Prentice Hall2006656

27*C++ GUI Programming with Qt 4*Prentice Hall2006560

28**Nokia Corporation***Qt Assistant*

29. *Control de Acceso.* [En línea] 23 de Abril de 2008. http://control-accesos.es/scada/¿que-es-un-sistema-scada.

30. **Bridón Danger, Yordanis y Moreno Borges, Adrian Carlos.** Interfaz asíncrona para la comunicación con los Controladores Lógicos Programables utilizando el Protocolo Industrial EtherNet/IP. Ciudad de la Habana, Cuba : s.n., 2008.

31. **Melgarejo, Yailyn Fernández.** Servicios de Integración con Terceros para el intercambio de Alarmas y Eventos que ocurran en el proceso y el sistema SCADA Guardián del ALBA. 2009.

32. **Malaga., Universidad de.** http://www.lcc.uma.es. *Lenguajes y Ciencias de la Computacion.* [En línea] 2007. http://www.lcc.uma.es/~eat/services/i_socket/i_socket.html#link1..

33. **Alvarez, Victor Manuel.** Adaptación e Implementación de Comunicación Inalámbrica para un Sistema de Pruebas de Mar. *sitio Web National Instrument.* [En línea] 2009. http://sine.ni.com/cs/app/doc/p/id/cs-12218.

34. **Barrero, Ernesto Leyva.** Implementación del módulo de diseño de reportes para el SCADA Guardián del Alba. Ciudad de la Habana, Cuba : s.n., 2009.

35. **Carletti, Eduardo J.** Sensores - Conceptos generales. *sitio Web Robots.* [En línea] Mayo de 2009. http://robots-argentina.com.ar/Sensores_general.htm.

36. **Casanova Peláez, Pedro, y otros.** *DISEÑO DE UN SISTEMA DE TELEMEDIDA Y TELECONTROL.* España : s.n., 2008.

37. **Hernández, Luis Enrique García.** Framework para el desarrollo de manejadores de dispositivos, para el proyecto SCADA Guardián del ALBA. Ciudad de la Habana, Cuba : s.n., 2009.

38. **Marcelo.** Sitio Web Definición ABC. *Definición de Sensor.* [En línea] 14 de Septiembre de 2008. http://www.definicionabc.com/motor/sensor.php .

39. **Ponce, Pedro.** Desarrollo de un Invernadero Innovador y Portatil Utilizando Desarrollo Grafico de Sistemas. *sitio Web National Instrument.* [En línea] 2008. http://sine.ni.com/cs/app/doc/p/id/cs-12337.

40. **Pozo, Antonio Cedeño.** Módulos de adquisición y análisis para la interacción con dispositivos de campo en un SCADA. *Trabajo de Diploma.* Ciudad de la Habana, Cuba : s.n., Abril de 2009.

41. **Rodríguez, G., y otros.** *SISTEMA DE ADQUISICIÓN DE SEÑALES PARA WINDOWS SADW2.* Ciudad de La Habana : s.n., 2001.

42. **Sánchez, Kiel Le.** *Tarjeta de Adquisición de Datos.* Ciudad de la Habana : s.n.

43. **Walter, Julio.** Medición de Parámetros Ambientales para Laboratorios de Metrología. *sitio Web National Instrument.* [En línea] 2008. http://sine.ni.com/cs/app/doc/p/id/cs-12300.

44. **Defensa, Unidad de Compatibilización Integración y Desarrollo De Software para la.** *Proceso de Desarrollo y Gestión de Proyectos de Software (1ra Versión).* Ciudad Habana : s.n., 2009.

45. **Ivar JACOBSON, Grady RUMBAUGH, James BOOCH.** *El Proceso Unificado de Desarrollo de Software.* Ciudad Habana : s.n., 2002.

46. **Mañas, José A.** Pruebas. *http://www.lab.dit.upm.es.* [En línea] 1994. http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm.

Glosario de términos

Consola de operación: Sitio de trabajo del operador y que sirve de interfaz gráfica directa entre éste y el Sistema SCADA.

Controlador Lógico Programable (PLC): Dispositivos electrónicos usados en automatización industrial para realizar estrategias de control básicas. Por su robustez y características sencillas de control, están cercanas al proceso, permitiendo ejecutar las tareas básicas del control, aún cuando no tenga conexión a las capas superiores del control.

Despliegue: Es el área de configuración de un proyecto, donde se representan los elementos gráficos, sus animaciones en dependencia de los valores que registran sus variables, los gráficos de tendencia, toda la representación visual de un SCADA.

Dispositivo de campo: Son los elementos físicos que miden, monitorean y, en algunos casos almacenan los datos de las variables del proceso. Estos dispositivos no se conectan directamente al SCADA.

GUI: Graphical User Interface (Interfaz gráfica de usuario) es un método para facilitar la interacción del usuario con el ordenador a través de la utilización de un conjunto de imágenes y objetos pictóricos (iconos, ventanas) además de texto.

Interfaz Hombre Máquina (IHM): Consiste en herramientas para la supervisión y control del proceso (consolas de operación y generador de despliegue).

Módulo: Es una parte de un programa de ordenador. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará una de dichas tareas (o quizás varias en algún caso).

Punto: Se refiere a las direcciones de memoria que se configuran en el SCADA y que referencian a una variable capturada por un dispositivo de campo, o una variable del proceso. Pueden ser de tipo analógico o digital.

SCADA: Sistema que se encarga de capturar y manipular la información referente a un proceso automatizado, esta información es utilizada para la realización análisis de indicadores y en la retroalimentación sobre algún operador.

Sensor: Es un dispositivo capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas.

Sinópticos: Que toma la forma de un objeto.

UCID: Unidad de Compatibilización, Integración y Desarrollo de Software para la Defensa.

Widget (Objeto gráfico): Objeto gráfico que puede contener gráficos vectoriales o imágenes raster, (imágenes de mapas de bits) y presenta un conjunto de propiedades que pueden ser editadas y asociadas a puntos (variables) del SCADA o a expresiones que contengan variables del sistema. Por medio de los objetos gráficos se pueden crear animaciones en función de los valores de los puntos asociados. Los widgets pueden agruparse para formar nuevos objetos gráficos más complejos. Un ejemplo de widget puede ser un contenedor de textos.