Universidad de las Ciencias Informáticas
Facultad 5

Título: Estructura Base del Visualizador del SCADA Guardián del Alba.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Osmany Pérez Feria

Tutor: Ing Raudi Agdel Bacallao Sánchez

Cotutor: Ing Roberto Cárdenas Isla

Ciudad de la Habana, Abril 2011.
"Año 53 de la Revolución"

Declaración de Autoría

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencia Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes _____ del año 2011.

_____ _____

Firma del autor Firma del tutor

Datos de Contacto

Ing. Raudi Agdel Bacallao Sánchez

Graduado de Ingeniero en Ciencias Informáticas y profesor de la Universidad de Ciencias Informáticas (UCI), con 5 años de experiencia en el desarrollo de software.

e-mail: rabacallao@uci.cu

Ing. Roberto Cárdenas Isla

Graduado de Ingeniero en Ciencias Informáticas y profesor de la Universidad de Ciencias Informáticas (UCI), con 4 años de experiencia en el desarrollo de software.

e-mail: risla@uci.cu

Agradecimientos

En primer lugar quiero agradecer el desarrollo de este trabajo a mi mamá y a mi papá, por la preocupación y esfuerzos durante todos estos años, en los que me apoyaron para llegar hasta aquí.

A mi hermano y a lisi por su cariño y apoyo, por ser tan especiales para mí.

A Dieguito por quererlo tanto.

A mis tutores por sus valiosos consejos y ayuda.

A Ariel por toda su ayuda y preocupación constante.

A mi familia en forma general por su apoyo.

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.

Dedicatoria

A mis queridos padres por ser tan especiales para mí.

A Dieguito para que siga el ejemplo de su papá y su tío que lo adoran.

A mi hermano y a lisi por haber confiado tanto en mí.

A mis queridos abuelos por ser tan preocupados con mis estudios.

Resumen

El presente trabajo muestra el desarrollo de las funcionalidades básicas del visualizador del SCADA (sus siglas en español significan: Supervisión, Control y Adquisición de Datos), que debe ser capaz de representar los datos de forma gráfica en tiempo real a los operadores de la planta, guiar a la detección del origen de fallas y emitir control sobre los procesos que se llevan a cabo.

Primeramente se comenzó por el estudio de sistemas SCADA existentes, haciendo posible adquirir una noción general del funcionamiento de estos y en especial del módulo Interfaz Hombre-Máquina (HMI), lo cual permitió una mejor comprensión para realizar los mecanismos que hacen posibles la visualización de los procesos.

Luego se definieron los requerimientos funcionales y no funcionales a implementar en el visualizador, con los que se realizó el modelado de la solución. Seguidamente se presentó el diseño del sistema, para comenzar la implementación.

Palabras Claves:

SCADA, Sistema en tiempo real (STR), Operador, HMI, Requerimientos.

Índice de Contenido

Introducción

El control paso a paso de los procesos que se realizan en las grandes industrias ha sido de gran importancia para el hombre moderno. La rivalidad entre las empresas las ha llevado a tratar de mejorar los procesos industriales con fines de aumentar la calidad y los niveles de producción, por lo que se hizo necesario reemplazar la mano de obra humana por sistemas de control automatizados. Los primeros mecanismos de supervisión eran sencillos sistemas de telemetría que solo proporcionaban reportes de las condiciones de campo y brindaban parámetros a las unidades remotas. Con la llegada de los avances tecnológicos y el uso de la informática en conjunto de la automática hizo posible la automatización de los procesos industriales, lográndose realizar operaciones complejas y riesgosas sin la presencia directa de los humanos. En la cúpula de estos sistemas hoy en día se encuentra el SCADA.

Un SCADA no es más que una aplicación software diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo, presentando los datos a un operador en forma amigable, permitiendo controlar el proceso de forma automática desde la pantalla del ordenador.

Un SCADA incluye dentro de sus funcionalidades: bases de datos históricas, comunicación, seguridad, procesamiento de datos y una interfaz de usuario, llamada HMI por sus siglas en inglés (Human Machine Interface), siendo este último, el módulo que permite la comunicación entre los usuarios y los ordenadores. El HMI está compuesto por dos aplicaciones, ambiente de configuración y ambiente de ejecución. El ambiente de configuración permite configurar todos los módulos del SCADA que automatizarán los procesos industriales. El ambiente de ejecución es el encargado de mostrar todos los datos recolectados de los dispositivos de campo de forma amigable, para su supervisión y control.

La informatización de la sociedad surge como una de las estrategias de la Batalla de Ideas que libra nuestro pueblo, y la Universidad de las Ciencias Informática (UCI) está a la vanguardia en esta tarea de la Revolución. En la Facultad 5 se encuentra el Centro de Informática Industrial (CEDIN), área encargada de la automatización de los procesos industriales, entre los proyectos más importantes del centro se encuentra el SCADA

Guardián del Alba, desarrollado como una alternativa de soberanía tecnológica para la empresa venezolana PDVSA (Petróleos de Venezuela S.A).

La versión del módulo HMI que desarrolló la parte cubana en el SCADA Guardián del Alba, hoy en día se encuentra fuera del mercado venezolano, y unas de las razones es que se desarrolló con tecnologías que presentaban deficiencias para aplicaciones de este tipo, como: inexistencia de una estructura de datos capaz de actualizar los objetos gráficos de forma optima, no poseía mecanismos de eventos para el manejo de gráficos en dos dimensiones, el diseñador de interfaces de usuario es complejo de utilizar y solo expertos podían obtener la interfaz gráfica deseada en un lapso de tiempo aceptable. Por lo que los clientes exigieron un cambio de tecnología para volver a utilizar la versión cubana en el SCADA Guardián del ALBA.

Entre las dificultades que presenta este módulo, es que se intentó abstraer la presentación de forma que en un futuro fuese posible migrar a otras bibliotecas como Qt, Motif, u otro framework gráfico, lo que provocó que se tuviese que multiplicar el número de líneas de código para obtener una misma funcionalidad, además de la complejidad que conllevaba introducir cambios o incorporar nuevas funcionalidades, algo que hoy en día es necesario realizar, por las necesidades de nuevos clientes, no constando con una de las principales características con que debe constar todo SCADA, una arquitectura abierta, capaz de crecer y adaptarse fácilmente a las necesidades cambiantes de las empresas.

Otra de las dificultades es que los tiempos de compilación son elevados, ya que el código no está distribuido en la mayor cantidad de componentes independientes posibles, conjuntamente con la dependencia cíclica que existe entre los subsistemas, haciendo que las herramientas de construcción recompilen varias veces las mismas clases. Esto traía consecuencia que los desarrolladores se tardaran mucho más de lo que se esperaba para realizar las tareas asignadas, generando retrasos en las entregas planificadas en el proyecto.

Con la idea de dar solución a las deficiencias en algunos de los componentes desarrollados en el SCADA Guardián del Alba, se pretende desarrollar un nuevo módulo

Interfaz Hombre-Máquina, el cual está compuesto por dos sub-módulos, Edición y Ejecución.

De todo lo expuesto anteriormente se presenta como **Problema Científico**: "Visualización y control de los procesos industriales en el SCADA Guardián del Alba"**.**

La investigación tiene como **objeto de estudio** la interfaz hombre-máquina en procesos industriales, teniendo como **campo de acción** las funcionalidades básicas de los visualizadores de los SCADA.

Para darle solución al problema anteriormente planteado se traza como **objetivo general** desarrollar las funcionalidades básicas del visualizador del SCADA Guardián del Alba.

Como **idea a defender** se define:

Con esta investigación se prevé obtener un visualizador robusto y flexible, que permitirá controlar los procesos industriales de forma automática desde la pantalla del ordenador, capaz de adaptarse fácilmente a los diferentes procesos industriales.

Para darle cumplimiento al objetivo planteado se definieron las siguientes **tareas de investigación**:

- Elaboración del marco teórico a través del estudio del estado del arte para conceptualizar los conocimientos en el tema de los SCADA.
- Estudiar la implementación del HMI existente, para el análisis y diseño de la arquitectura actual.
- Elaboración de una arquitectura base flexible a partir del estudio de visualizadores existentes.
- Análisis y diseño de los requerimientos básicos para el visualizador del SCADA Guardián del Alba.
- Implementación de los requerimientos básicos del visualizador del SCADA Guardián del Alba.

Para darle cumplimiento a las tareas de investigación planteadas se llevarán a cabo varios métodos en la búsqueda y procesamiento de la información como son:

Métodos a nivel Teórico:

Analítico – Sintético: Durante la definición de las funcionalidades básicas necesarias del visualizador del SCADA Guardián del Alba.

Análisis sistémico: Durante el desarrollo del diseño mediante la elaboración de los diagramas de clases.

Modelación: Durante el desarrollo del diseño mediante la elaboración de los diagramas de secuencia.

Métodos a nivel Empírico:

Observación: Permitirá observar cómo se comporta la visualización y actualización del visualizador del SCADA Guardián del Alba.

Experimento: Se utilizará para la elaboración del mecanismo de actualización del visualizador de SACAD Guardián del Alba.

A continuación se describen los elementos más importantes identificados durante la confección del cuerpo del documento de la presente investigación:

Capítulo 1: Se desarrolla la fundamentación teórica del presente trabajo y se exponen las características de las tecnologías utilizadas.

Capítulo 2: Características del Sistema, se ofrece una visión práctica del sistema, exponiéndose los requisitos funcionales y no funcionales, además se determinan los casos de uso y se describen los más significativos.

Capítulo 3: Diseño e implementación, se diseña un sistema de clases, en correspondencia con las técnicas de la programación orientada a objetos, que luego son implementadas teniendo como resultado final un prototipo funcional del sistema propuesto.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se hace alusión al funcionamiento y a los módulos fundamentales de un SCADA para la gestión y control de datos, haciéndose énfasis en HMI. Además se hace una descripción de las principales tecnologías que se emplean en la construcción del software, incluyendo: el lenguaje de programación, framework, y las herramientas de desarrollo empleadas.

1.2 Generalidades de un sistema SCADA

El término SCADA usualmente se refiere a un sistema central que monitorea y controla un sitio completo o un sistema que se extiende sobre una gran distancia (kilómetros / millas). La instalación de un sistema SCADA necesita un hardware de señal de entrada y salida, sensores y actuadores, controladores, HMI (Interfaz Hombre Máquina), redes, comunicaciones, bases de datos, entre otros. Los sistemas SCADA mejoran la eficacia del proceso de monitoreo y control, proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas. Cuenta además con información (alarmas, históricos, paradas, etc.) de primera mano de lo que ocurre u ocurrió en el proceso. [1].

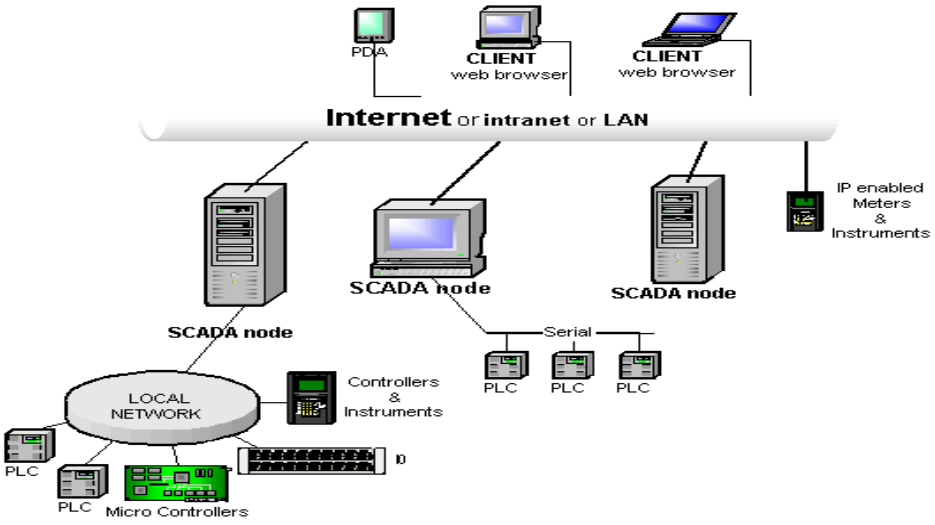


Fig. 1 Ejemplo del esquema de un SCADA.

Un SCADA debe cumplir tres **funciones principales:**

Adquisición de datos: significa que el sistema tiene la capacidad de obtener información desde el sistema de control, por ejemplo sobre valores de temperatura o presión y a diferencia de la "supervisión" los datos son registrados o almacenados para su posterior explotación.

Supervisión: significa poder observar o monitorear aquello que sucede en el proceso industrial, el equipo o la maquinaria, por ejemplo conocer (generalmente de una forma gráfica) si un motor se encuentra encendido o apagado.

Control: significa tener la posibilidad de ejecutar comandos; en otras palabras, poder enviar instrucciones hacia el sistema de control, por ejemplo ordenar al PLC (en español significa: Controlador Lógico Programable) que encienda o apague un motor. [4].

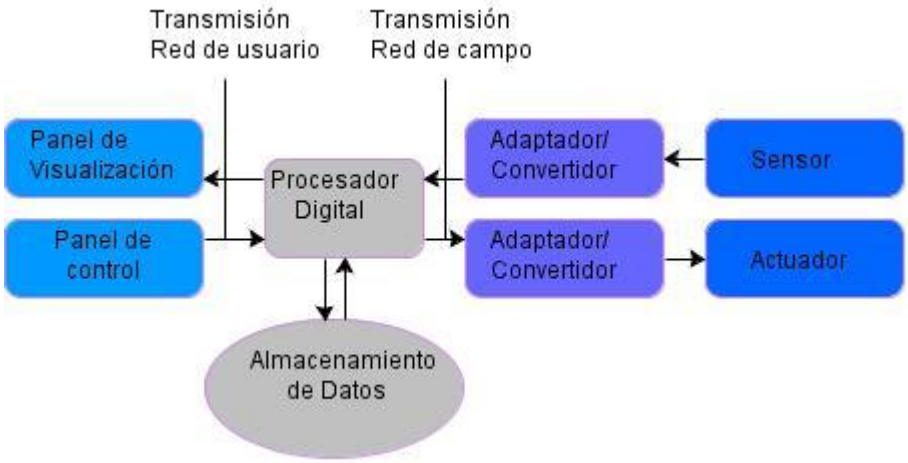


Fig. 2 Esquema de un Sistema de Adquisición, Supervisión y Control de Datos.

El usuario, mediante herramientas de visualización y control, tiene acceso al procesador digital, generalmente un ordenador, donde reside la aplicación de supervisión y control. La comunicación entre estos dos sistemas se suele realizar a través de redes de comunicación corporativas. El procesador digital capta el estado del sistema a través de los elementos sensores e informa al usuario a través de la herramienta HMI. Basándose en los comandos ejecutados por el usuario el procesador digital inicia las acciones pertinentes para mantener el control del sistema a través de los elementos actuadores.

1.2.1 Funciones más específicas de un SCADA:

- Transmisión de información con dispositivos de campo y otros PC.
- Gestión de datos con bajos tiempos de acceso.
- Representación gráfica de los datos. Interfaz del Operador o HMI.
- Explotación de los datos adquiridos para gestión de la calidad, control estadístico, gestión de la producción y gestión administrativa y financiera. [4]

1.3 Módulos de un SCADA distribuido

- **Bases de Datos Históricos:** Es el encargado de almacenar la información recibida desde el campo, así como la sucesión de alarmas y eventos**.** Esta información es de vital importancia para realizar cualquier tipo de análisis posteriores como, por ejemplo, diagnósticos o reportes.
- **Comunicación:** Este módulo representa la capa de software encargada de la comunicación entre los diferentes procesos distribuidos, de mediano y alto nivel.
- **Seguridad:** Permite a los usuarios autenticarse en el sistema, y de esta forma poder acceder sólo a los recursos que tiene asignado su rol. Posee herramientas para la protección ante ataques piratas, fallos eléctricos, problemas de red, entre otros.
- **Procesamiento de Datos:** Éste es el "centro neurológico" del sistema, el cual supervisa y recoge la información del resto de las subestaciones, bien sean otros ordenadores conectados (en sistemas complejos) a los instrumentos de campo o directamente sobre dichos instrumentos, algunas de las funciones que cumple son:
  - Interrogar de forma periódica a las RTU, y transmitirle consignas; siguiendo usualmente un esquema maestro-esclavo.
  - Actuar como interfaz al operador, incluyendo la presentación de información de variables en tiempo real, la administración de alarmas, y la recolección y presentación de información historiada.
  - Puede ejecutar software especializado que cumple funciones específicas asociadas al proceso supervisado por el SCADA. Por ejemplo, software para detección de pérdidas en un oleoducto.

1.4 Sistemas existentes

A continuación se muestra una tabla con una comparación entre los software SCADA más importantes existentes en el mercado (CXSupervisor, All-Done, Intouch, Vijeo Look y Win CC).

Software	**CX-Supervisor**	**All-Done**	**InTouch**	**Vijeo Look**	**Win CC**
Fabricante	Omron/Omron (UK)	Freixas i Ros, S.L./Freixas i Ros, S.L.(E)	Logitec, S.A./ Wonderware (USA)	Schneider Electric/ Schneider Electric (F)	Siemens/ Siemens
Drivers para PLC´s	OMRON: todo los PLC's	OMRON: Sysmac seric C MOELLER: PS4-200 SIEMENS: simatic S5,S7-200/300/400 TELEMECANIQUE	ALLENBRADLEY SIEMENS MODICON OPTO 22 SQUARE D OMRON	TELEMECANIQUE AEG MODICON MODICON SQUARE D	SIMATIC ALLEN-BRADLEY MITSUBISHI FETELEMECANIQUE UNI-TELWAYGE- FANUC MODICON OMRON serie C
Lenguajes deprogramación	Visual Basic/Java	Visual Basic	propio (basado en C)	VBA (Visual Basic for Application)	Visual Basic C ANSI-C
Control de usuarios	si	si	si	si	si
gestión de alarmas	Si	Si	Si	Si	Si
Adquisición de datos	cliente OPC	OPC Servidor/Cliente	OPC Servidor/ Cliente	OPC Factory Server (OFS)	OPC Servidor/Cliente

Tabla # 1: Comparación entre sistemas SCADA existentes.

En la tabla se puede observar las similitudes y diferencias que poseen entre unos y otros sistemas. Pudiéndose decir que todos convergen en una característica y es el uso de tecnologías propietarias, principalmente de Microsoft. A modo de diferencia la característica que más varía son los drivers utilizados para la conexión con el PLC.

1.5 El módulo Interfaz Hombre-Máquina

La Interfaz Hombre-Máquina o HMI ("Human Machine Interface") es el aparato que presenta los datos a un operador (humano) y a través de la cual éste controla el proceso. La industria de HMI nació esencialmente de la necesidad de estandarizar la manera de monitorear y de controlar múltiples sistemas remotos, PLC y otros mecanismos de control. Aunque un PLC realiza automáticamente un control pre-programado sobre un proceso, normalmente se distribuyen a lo largo de toda la planta, haciendo difícil recoger los datos de manera manual, los sistemas SCADA lo hacen de manera automática.

Históricamente los PLC no tienen una manera estándar de presentar la información al operador. La obtención de los datos por el sistema SCADA parte desde el PLC o desde otros controladores y se realiza por medio de algún tipo de red, posteriormente esta información es combinada y formateada. Un HMI puede tener también vínculos con una base de datos para proporcionar las tendencias, los datos de diagnóstico y manejo de la información así como un cronograma de procedimientos de mantenimiento, información logística, esquemas detallados para un sensor o máquina en particular, incluso sistemas expertos con guía de resolución de problemas. Desde cerca de 1998, virtualmente todos los productores principales de PLC ofrecen integración con sistemas HMI/SCADA, muchos de ellos usan protocolos de comunicaciones abiertos y no propietarios. Numerosos paquetes de HMI/SCADA de terceros ofrecen compatibilidad incorporada con la mayoría de PLC, incluyendo la entrada al mercado de ingenieros mecánicos, eléctricos y técnicos para configurar estas interfaces por sí mismos, sin la necesidad de un programa hecho a medida escrito por un desarrollador de software.

Aplicaciones que componen un HMI:

- **Ambiente de configuración:** Esta aplicación permite configurar varios procesos o partes de ellos, posibilitando al usuario definir el entorno de trabajo de su aplicación, según la disposición de pantallas requeridas y los niveles de acceso para los distintos usuarios. Dentro del módulo de configuración el usuario define las pantallas gráficas o de texto que va a utilizar, se seleccionan los drivers de comunicación que permitirán el enlace con los elementos de campo y la conexión o no en red de estos últimos. Estas configuraciones son las que más tarde utilizará

el visualizador para poner en ejecución la representación gráfica de la planta, fábrica o lugar donde ha sido instalado.

- **Ambiente de ejecución:** Proporciona al operador las funciones de control y supervisión de la planta. El proceso a supervisar se representa mediante sinópticos gráficos que cambian dinámicamente a diferentes formas y colores, según los valores leídos en la planta o en respuesta a las acciones del operador. Las funcionalidades principales con la que debe contar toda aplicación de este tipo son:

  - **Visualización de despliegue:** Es el encargado de mostrar de forma gráfica los procesos que se siguen en una planta. Esto se realiza a través de objetos gráficos que suelen dividirse en dos grupos, simples y complejos, los primeros se caracterizan por ser objetos estáticos, como circunferencias, líneas, imágenes, y los complejos se caracterizan por variar su estado en el transcurso del tiempo según los datos leídos en la planta, entre los principales se encuentran: las gráficas de tendencia, los relojes y las reglas.
  - **Visualización de alarmas:** Es una interfaz gráfica que concentra las condiciones de procesos críticas, medias y baja presentes en el sistema, cuyo objetivo primordial es guiar al operador a la detección del origen de la falla y supervisar la ejecución de las medidas de corrección automatizada o manual.
  - **Visualización de puntos:** Es la interfaz gráfica encargada de visualizar los detalles de un punto, tales como: nombre, calidad, valor, fecha, entre otros valores.
  - **Visualización de eventos:** Muestra las acciones que se han realizado en el sistema.

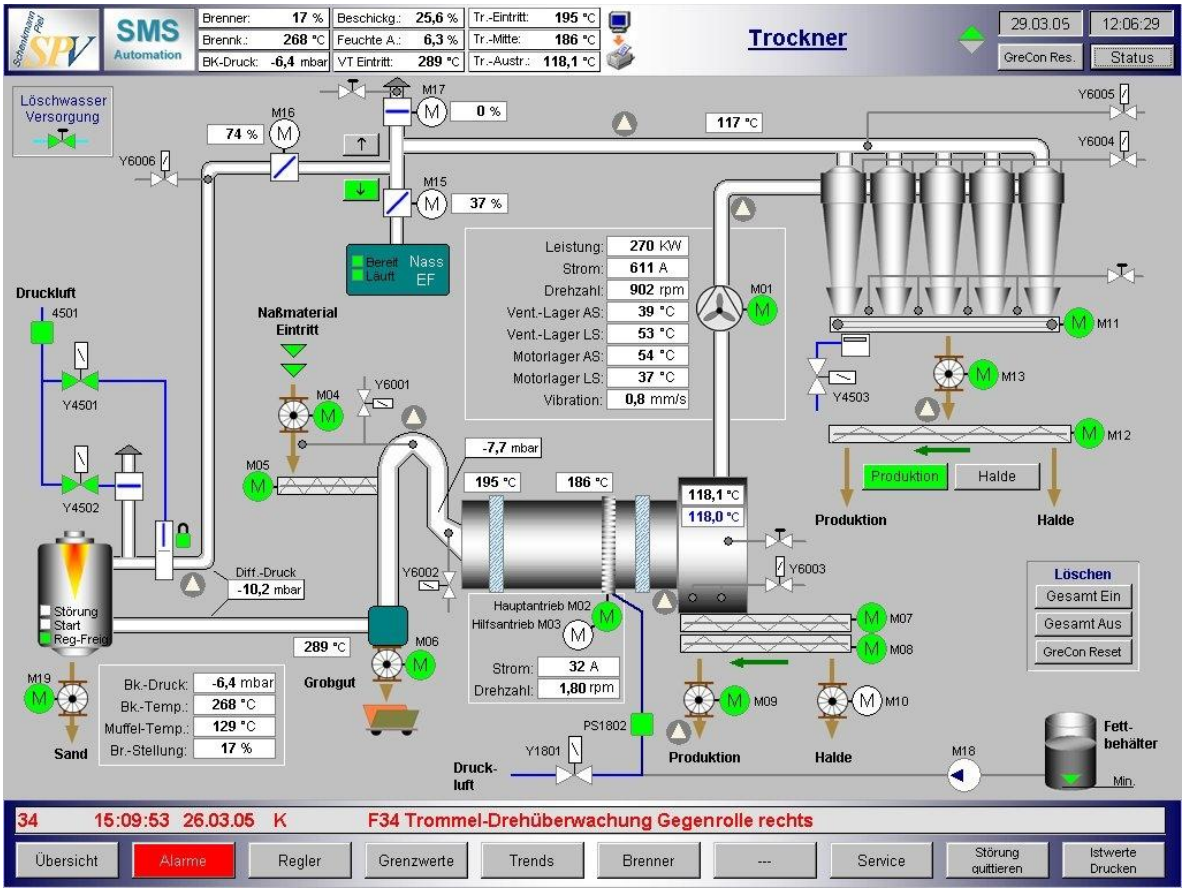


Fig. 3 Ejemplo del HMI del SCADA WinCC.

Como se observa, los visualizadores de hoy en día ofrecen a los operadores las más sofisticadas técnicas de supervisión y control pudiendo simular partes de la planta con los gráficos sinópticos haciendo sumamente intuitiva la operación de una planta específica.

1.6 Tendencias y tecnologías

En este epígrafe se realiza un análisis de las tendencias y tecnologías actuales en el campo del desarrollo de aplicaciones. Este proyecto se desarrolla a partir de las herramientas que brinda el software libre, buscando la independencia tecnológica que estas brindan al posibilitar la libertad de uso y distribución de los programas sin incurrir en litigios de licenciamiento o asuntos legales. Las tecnologías que se pusieron a consideración para la implementación de la aplicación fueron Python o C++ como lenguaje de programación, y GTK o Qt como Framework Gráfico. Es importante destacar

que cada una de las tecnologías que se mencionaron, fueron estudiadas en la Línea de Visualización del SCADA, escogiendo como lenguaje de programación C++ y Framework Gráfico Qt. A continuación se hace un análisis de las principales características de las tecnologías seleccionadas.

1.6.1 Lenguaje de programación C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. Su creación se basó sobre la necesidad de extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. Desde el punto de vista de los lenguajes orientados a objetos, es un lenguaje híbrido.

C++ está considerado por muchos como el lenguaje de programación más potente, debido a que permite el trabajo tanto a alto nivel como a bajo nivel, logrando gran eficiencia en tiempos de ejecución y bajo consumo de memoria en los programas desarrollados, algo que la mayoría de las aplicaciones requieren, siendo siempre una buena opción como lenguaje a utilizar en sistemas que necesitan alto rendimiento.

Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Algunas de las características que se pueden encontrar en C++ son:

- Programación orientada a objetos: La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real. Además, permite la reutilización del código de una manera más lógica y productiva.
- Portabilidad: Un código escrito en C++ puede ser compilado en casi todo tipo de ordenadores y sistemas operativos sin hacer apenas cambios.
- Programación modular: Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Además, esta característica permite unir código en C++ con código producido en otros lenguajes de programación como Ensamblador.

- Velocidad: El código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel y a la reducida medida del lenguaje.

C++ aporta un alto rendimiento en cuanto al uso de la memoria pues maneja de manera dinámica accediendo y eliminándola constantemente, lo cual no deja que se sobrecargue la pila de la máquina. También tiene como característica la declaración de funciones inline que posibilitan que el código de estas funciones se ejecute más rápido, evitando usar la pila para pasar parámetros; además se evitan las instrucciones de salto y retorno que implican costo en tiempo.

Existen gran cantidad de bibliotecas implementadas que permiten la reutilización del código fuente, disminuyen el tiempo de desarrollo y aumenta la productividad. La utilización de plantillas también permite la reutilización del código y la programación genérica.

Entonces se pude determinar que C++ consta con:

- Eficiencia en el manejo de la memoria y el procesamiento de datos (CPU).
- Aprovechamiento máximo con librerías.
- Reutilización de código fuente con plantillas. Permite la programación genérica.
- Manejo de errores.
- Mecanismos de serialización.
- Compiladores disponibles para todas las arquitecturas.
- Es un lenguaje orientado a objetos.
- Disponible para múltiples plataformas.
- Posibilidad de abstracción de datos.
- Es un lenguaje compilado.
- Permite trabajar tanto a alto como a bajo nivel.
- Es un lenguaje bastante maduro.

1.6.2 Framework gráfico Qt

Qt es un framework multiplataforma para el desarrollo de interfaces gráficas. Es utilizado en aplicaciones como KDE, Google Earth, Skype, Adobe Photoshop Album, VirtualBox y Opie. Es producido por la división de software Qt de Nokia, que entró en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008.

Qt está desarrollado de forma nativa en C++, pero ha sido portado para ser utilizado desde varios lenguajes de programación como Java, Python, C#, entre otros. Qt está disponible para varios sistemas operativos entre ellos: Windows, Linux y Mac OSx, teniendo un amplio respaldo en el sector empresarial y en las comunidades de desarrollo.

Incluye un amplio conjunto de widgets que proporcionan las funcionalidades estándar de interfaz gráfica de usuario, introduce una innovadora alternativa para la comunicación entre objetos, llamados "signals y slots". Puede soportar toda la funcionalidad de interfaz de usuario que requieren las aplicaciones modernas, tales como menús, menús contextuales, arrastrar y soltar, y barras de herramientas acoplables. También propone el patrón de diseño model/view (modelo/vista) para la representación gráficas de los datos con la separación de las funcionalidades introducidas por esta arquitectura, ofrece a los desarrolladores una mayor flexibilidad para personalizar la presentación de elementos y proporciona una interfaz de modelo estándar que permitir una amplia gama de fuentes de datos.

Hay que destacar que comenzó como un framework para el desarrollo de interfaces gráficas, pero ya el API de la biblioteca cuenta con tecnologías que facilitan el trabajo a los desarrolladores como son:

- Máquina de estados.
- Soporte para hilos.
- Sistema de pintado.
- Animaciones y multimedia.
- Vista para gráficos 2D.
- Vista para gráficos 3D.
- Integración de documentación en aplicaciones.

- Modelos dinámicos de objetos.
- Pruebas de unidad.
- Serialización y trabajo con DOM.
- Trabajo con extensiones.
- Contenedores genéricos.
- Estilos.
- Eventos y filtrado de eventos.

Además permite crear aplicaciones de bases de datos independientes de la plataforma que se utiliza. Incluye drivers nativos para Oracle, Microsoft SQL Server, Sybase Adaptive Server, IBM DB2, PostgreSQL TM, MySQL, Borland Interbase, SQLite, y bases de datos compatibles con ODBC. Incluye widget personalizado para la representación de los datos de las bases de datos.

Otras características son: la disponibilidad de código fuente, la excelente documentación organizada en un asistente (QtAssitant), y un editor para el diseño de formularios visualmente (QtDesigner). Qt se distribuye bajo un sistema de licenciamiento dual, que incluye una licencia comercial y una de código abierto bajo los términos de GNU Lesser General Public License (LGPL).

1.6.3 XML (Extensible Markup Language)

XML es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable. Es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores.

Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Entre sus principales ventajas se pueden mencionar:

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Se pueden comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, se pude tener una aplicación en Linux con una base de datos postgresql y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.
- Transformación de datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual se tiene flexibilidad para estructurar documentos.

1.7 Metodología de software

1.7.1 Proceso Unificado de Desarrollo

El Proceso Unificado de Rational o RUP (Rational Unified Process), es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El Proceso Unificado es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, para diferentes áreas de aplicación, tipos de organizaciones, niveles de competencia y tamaños de proyectos.

Provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible.

El Proceso Unificado usa el Lenguaje de Modelado Unificado (UML) en la preparación de todos los planos del sistema. De hecho, UML es una parte integral del Proceso Unificado, fueron desarrollados a la par.

Los aspectos distintivos del Proceso Unificado están capturados en tres conceptos claves: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Esto es lo que hace único al Proceso Unificado.

- **Dirigido por casos de uso:** Un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado de valor. Los casos de uso capturan los requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad completa del sistema. Este modelo reemplaza la tradicional especificación funcional del sistema. Sin embargo, los casos de uso no son solamente una herramienta para especificar los requerimientos del sistema, también dirigen su diseño, implementación y pruebas, es decir, dirigen el proceso de desarrollo.
- **Centrado en la arquitectura:** La arquitectura da una perspectiva del sistema completo, en la que todo el equipo de proyecto y usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- **Iterativo e Incremental:** El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones. Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

UML: Sus siglas en inglés (Unified Modeling Language), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. UML es un "lenguaje" para modelar y no un método o un proceso. [5]

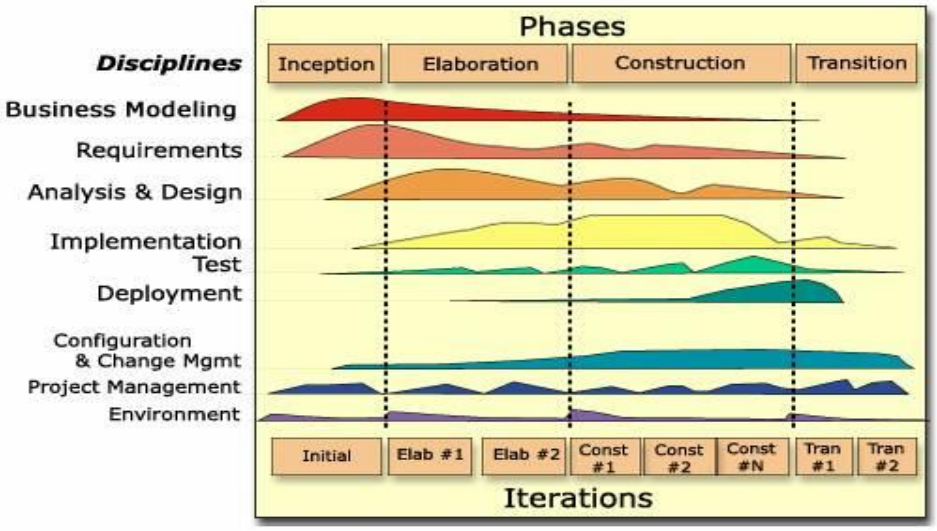


Fig. 4 Fases e iteraciones del Proceso Unificado.

1.8 Herramientas de desarrollo empleadas

1.8.1 IDE de programación Eclipse

Eclipse es un entorno de desarrollo integrado (IDE) que abarca todo el ciclo de desarrollo de software. El software en sus inicios fue desarrollado por IBM pero en la actualidad es mantenido por la Fundación Eclipse que es una organización independiente sin ánimos de lucro que fomenta una comunidad de código abierto y un conjunto de productos, capacidades y servicios complementarios, además tiene el apoyo de gran cantidad de innovadoras empresas, prestigiosas universidades y varias personas alrededor del mundo. Inicialmente eclipse se desarrolló para los programadores que

usaban el lenguaje Java pero en la actualidad existe soporte para varios lenguajes de programación como C/C++, Python, PHP entre muchos otros. También el software tiene herramientas para modelado de software, sistemas de gestión de bases de datos (DBMS), para la gestión de la configuración y el control de versiones, tiene soporte para CVS y Subversion. [6].

1.8.2 Herramienta CASE

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

- **Visual Paradigm**

Es una herramienta Case para UML, de fácil uso y completa, con soporte multiplataforma, posibilita una alta interoperabilidad con otras aplicaciones. La herramienta ayuda al equipo del desarrollo del software a maximizar y acelerar el desarrollo del software y contribuciones individuales. Apoya un gran número de idiomas en la generación de código y la ingeniería inversa en Java, C++, PHP, entre otros. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de clases. Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al momento, creando de forma simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso de la aproximación orientado a objeto. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros de equipo.

Capítulo 2: Características del Sistema

2.1 Introducción

En el presente capítulo se hace una descripción general de la propuesta del sistema así como los conceptos necesarios para el desarrollo de la solución. Se exponen los requisitos funcionales y no funcionales que regirán el desarrollo de la solución propuesta. A partir de los requerimientos se determinarán los casos de uso, que serán mostrados mediante diagramas y luego descritos para una mejor comprensión.

2.2 Propuesta del sistema

2.2.3 Descripción general de la propuesta de sistema

El sistema que se propone será la estructura base del visualizador del SCADA Guardián del Alba. Constará de: un mecanismo de Entrada/Salida capaz de abstraer al sistema de la lógica de comunicación con los distintas fuentes de datos, una consola gráfica que permitirá mostrar e interactuar de forma amigable con el operador, varios sumarios cuyo objetivo primordial es guiar a la detección del origen de la falla y supervisar la ejecución de las medidas de corrección automatizada o manual y un mecanismo de actualización de las distintos tipos de consolas gráficas. Este sistema no solo será un visualizador de escritorio, sino que posibilitará el uso de sus componentes como pueden ser los mecanismos de Entrada/Salida y procesamiento de datos por otros visualizadores como el visualizador web o el de móviles.

2.3 Especificación de los requisitos de software

2.4.1 Requisitos funcionales

Son capacidades o condiciones que el sistema debe cumplir para que las peticiones del usuario queden satisfechas. A continuación se muestra una tabla con los requisitos funcionales que se deben implementar al software, con una breve descripción y el nivel de complejidad.

Nº	Funcionalidad	Descripción	Complejidad
[R1.]	Visualizar el sumario de alarmas.	El sistema debe presentar una interfaz gráfica que concentre las condiciones de procesos críticas, medias y bajas presentes en el sistema y cuyo objetivo primordial es guiar al operador a la detección del origen de la falla y supervisar la ejecución de las medidas de corrección automatizada o manual. El sistema debe ser capaz de representar el sumario en modo contraído y en modo expandido. En caso de mostrar el sumario en modo de lista contraída debe mostrar al menos las cinco (5) últimas alarmas en su orden de aparición. El sumario en modo de lista expandida debe representar las alarmas ordenadas de acuerdo a la relación severidad, prioridad, reconocimiento. Los datos que se deben mostrar en las alarmas son: **Fecha y Hora:** Fecha y hora de ocurrencia de la alarma. **Recurso:** Recurso que genera la alarma. **Causa y Tipo:** La causa y tipo de alarma generada. **Descripción:** Descripción del evento que generó la alarma, debe incluir el valor que generó la alarma en los casos que aplique. **Nro. De Ocurrencias:** Número de ocurrencias de la alarma sin haber sido reconocida. **Prioridad:** Prioridad configurada a la Alarma, para las alarmas de falla de comunicación y falla de instrumento su prioridad es fija y es igual a la más baja que maneje el sistema. **Grupo:** Grupo operacional al cual pertenece el recurso que generó la alarma.	Alta

[R2.]	Operar alarmas.	Dentro del sumario de alarmas se pueden ejecutar las acciones que se enuncian. **Reconocer:** Ejecuta la acción de reconocimiento de la o las alarmas seleccionadas. **Reconocer Todas:** Ejecuta la acción de reconocimiento de todas las alarmas en el sumario, visibles o no, excepto las alarmas de severidad 1. **Silenciar Alarma:** Ejecuta la acción de silenciado de todas las alarmas en el sumario, en caso de que se genere una nueva alarma posterior a la acción, la misma no será afectada. **Silenciar Todas:** Ejecuta la acción de silenciado de la o las alarmas seleccionadas en el sumario, en caso de que se genere una nueva alarma posterior a la acción, la misma no será afectada.	Media
[R3.]	Filtrar de alarmas.	El sumario debe permitir el filtrado de la información, por los campos Fecha, Recurso, Grupo y tipo, estos filtros se aplican solo al sumario en modo extendido.	Baja
[R4.]	Visualizar el sumario de puntos	El sistema debe presentar una interfaz gráfica con los siguientes datos del punto: Nombre, Calidad, Valor, Descripción y Grupo.	Media
[R5.]	Autenticar usuario.	El módulo de visualización debe permitir la integración con el módulo de seguridad, y la seguridad será basada en la autenticación por clientes y control de acceso a los recursos del sistema.	Alta
[R6.]	Emitir control.	El sistema debe permitir que el operador pueda manejar la información relacionada con las operaciones y ejecutar todos los comandos que su nivel de acceso le permita.	Alta
[R7.]	Actualizar consola gráfica.	El sistema debe garantizar el refrescamiento de datos en los despliegues.	Alta
[R8.]	Actualizar sumario de alarmas.	El sistema debe garantizar el refrescamiento de datos en el sumario de	Media

		alarmas.	
[R9.]	Actualizar sumario de puntos.	El sistema debe garantizar el refrescamiento de datos en el sumario de puntos.	Media
[R10.]	Navegar entre despliegues.	El sistema debe ser capaz de mostrar los despliegues seleccionados por el operador.	Media
[R11.]	Cargar configuración.	El sistema debe ser capaz de cargar la configuración del proyecto.	Alta
[R12.]	Adquisición de campo	El sistema de ser capaz de adquirir la información recogida en el campo.	Alta

Tabla # 2: Requisitos Funcionales.

2.4.2 Requisitos no funcionales

Son propiedades o cualidades que el producto debe tener, debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. A continuación se enuncian los diferentes requisitos no funcionales.

Usabilidad:

RNF 1: El sistema debe proveer una forma de Interacción de la Interfaz basada en ventanas.

RNF 2: El sistema debe proveer una capacidad de escalabilidad de los despliegues a la resolución de la pantalla en tiempo de ejecución.

RNF 3: El sistema debe posibilitar el manejo de teclas funcionales para despliegues.

RNF 4: Se debe diseñar el despliegue de alarmas de forma que no interfiera la visualización de despliegues del proceso y viceversa.

RNF 5: Se debe posibilitar en la interfaz de administración de alarmas la capacidad de visualizar las alarmas por colores dependiendo de su grado de criticidad, su valor y fecha en que sucedió el evento.

Rendimiento y disponibilidad del sistema:

RNF 6: Se debe garantizar el tiempo de acceso a las diferentes ventanas y despliegues del sistema en menos de dos segundos.

Requerimientos de portabilidad:

RNF 7: El sistema debe funcionar en sistemas de la familia GNU/Linux.

Requerimiento de confiabilidad:

RNF 8: Se debe garantizar que el módulo esté disponible las 24 horas del día.

RNF 9: Las actividades de mantenimiento, supervisión y edición del sistema no deben interferir en el correcto desempeño del resto del sistema, ni afectar la ejecución de la aplicación.

Requerimientos del tiempo de respuesta para adquisición y procesamiento de datos:

RNF 10: 5 milisegundos para acceso a los datos.

RNF 11: 1-2 segundos para el refrescamiento de los datos en el despliegue de las consolas. Tiempo que demora el dato desde que sale de módulo de recolección hasta que se visualiza en la consola.

2.5 Casos de uso del sistema

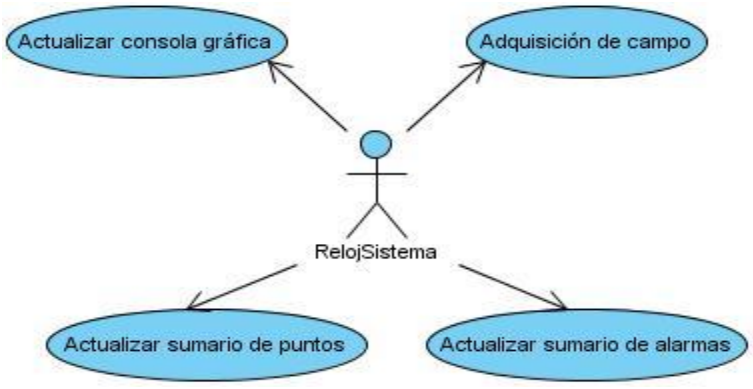


Fig. 6 Casos de uso del reloj del sistema.

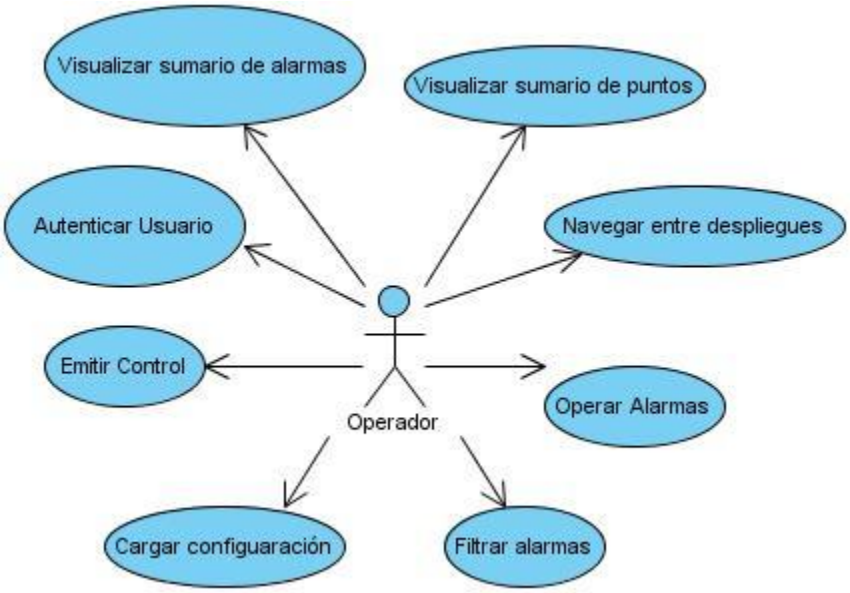


Fig. 5 Casos de uso del operador.

2.6 Definición de los actores del sistema

Actor	Descripción
Operador	Persona encargada directamente del control y supervisión de los procesos operacionales y de la ejecución y envío de los reportes durante su guardia. El ambiente del Operador es la Sala de Control, este es el responsable de las consolas de operación, que están conectadas a los servidores SCADA, los cuales pueden encontrarse fuera de la sala de control.
RelojSistema	Es el encargado directamente de llevar el lapso de tiempo en que se deben recolectar los datos del sistema y de emitir una señal para realizar la actualización de las distintas consolas gráficas.

2.7 Descripción de los casos de uso

2.7.1 CU Visualizar el sumario de alarmas

<table>
<tr><td>Nombre del caso de uso</td><td colspan="2">Visualizar el sumario de alarmas.</td></tr>
<tr><td>Actores</td><td colspan="2">Operador</td></tr>
<tr><td>Propósito</td><td colspan="2">Mostrarle al operador las alarmas.</td></tr>
<tr><td colspan="3">Resumen: El caso de uso comienza cuando el operador va al menú y escoge la opción, ver sumario de alarmas, el sistema muestra el sumario.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF 1</td></tr>
<tr><td colspan="3">Curso normal de los eventos</td></tr>
<tr><td>Acción del actor</td><td colspan="2">Respuesta del sistema</td></tr>
<tr><td>1- El operador escoge la opción en el menú de la ventana principal, ver sumario de alarmas.</td><td colspan="2">2- El sistema muestra el sumario con las alarmas existentes en el sistema.

Los datos que se mostrarán son:

Fecha y Hora: Fecha y hora de ocurrencia de la alarma.

Recurso: Recurso que genera la alarma.

Causa y Tipo: La causa y tipo de alarma generada.

Descripción: Descripción del evento que generó la alarma.

Nro. De Ocurrencias: Número de ocurrencias de la alarma sin haber sido reconocida.

Prioridad: Prioridad configurada a la alarma, para las alarmas de falla de comunicación y falla de instrumento su prioridad es fija y es igual a la más baja que maneje el sistema.

Grupo: Grupo operacional al cual pertenece el recurso que generó la alarma.</td></tr>
</table>

2.7.2 CU Operar alarmas

Nombre del caso de uso	Operar alarmas
Actores	Operador
Propósito	Reconocer, Silenciar, Reconocer Todas, Silenciar Todas.
Resumen: El caso de uso comienza cuando el operador decide realizar una operación dentro del sumario de alarmas, que puede ser: Reconocer, Silenciar, Reconocer todas, Silenciar todas. El sistema da una respuesta en dependencia de la acción seleccionada.	
Referencias	RF 2
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
a) Si el actor desea reconocer una alarma ir a la sección “Reconocer”. b) Si el actor desea silenciar una alarma ir a la sección “Silenciar”. c) Si el actor desea reconocer todas las alarmas ir a la sección “Reconocer todas”. d) Si el actor desea silenciar todas las alarmas ir a la sección “Silenciar todas”.	
Sección: “Reconocer”	
Acción del actor	Respuesta del sistema
1- El operador va al sumario de alarmas, selecciona la alarma, y	2- El sistema ejecuta la tarea reconocer alarma.

pulsa el botón reconocer alarma.	3- Muestra el mensaje del reconocimiento de la alarma.
Sección: “Silenciar”	
Acción del actor	Respuesta del sistema
1- El operador va al sumario de alarmas, selecciona la alarma, y pulsa el botón silenciar alarma.	2- El sistema ejecuta la tarea silenciar alarma.
Sección: “Reconocer todas”	
Acción del actor	Respuesta del sistema
1- El operador va al sumario de alarmas y pulsa el botón reconocer todas las alarmas.	2- El sistema ejecuta la tarea reconocer todas la alarmas, si existen alarmas críticas en el panel no se reconocen. 3- Muestra el mensaje del reconocimiento de todas las alarmas.
Sección: “Silenciar todas”	
Acción del actor	Respuesta del sistema
1- El operador va al sumario de alarmas y pulsa el botón silenciar todas las alarmas.	2- El sistema ejecuta la tarea silenciar todas la alarmas.

2.7.3 CU Filtrar alarmas

Nombre del caso de uso	Filtrar alarmas
Actores	Operador
Propósito	Filtrar las alarmas en el sumario expandido de alarmas por: campos Fecha, Recurso, Grupo y tipo.

Resumen: El caso de uso inicia cuando el operador decide filtrar las alarmas, el sistema filtra las alarmas y las muestra.	
Referencias	RF 3
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- El operador selecciona la columna por la que desea realizar el filtrado. 2- El operador entra el valor del campo por el que va realizar el filtrado.	3- El sistema realiza el filtrado de las alarmas. 4- El sistema muestra solo las alarmas filtradas.

2.7.4 CU Visualizar el sumario de puntos

Nombre del caso de uso	Visualizar el sumario de puntos.
Actores	Operador
Propósito	Mostrarle al operador los puntos.
Resumen: El caso de uso comienza cuando el operador va al menú y escoge la opción, ver sumario de puntos, el sistema muestra el sumario de puntos.	
Referencias	RF 4
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- El operador escoge la opción en el menú de la ventana principal, ver	2- El sistema muestra el sumario con los puntos activos en el sistema. Los datos que se mostraran en los puntos son: Nombre, Calidad, Valor,

sumario de puntos.	Descripción y Grupo.

2.7.5 CU Autenticar usuario

Nombre del caso de uso	Autenticar usuario
Actores	Operador
Propósito	Autenticación de los usuarios del sistema.
Resumen: Este caso de uso inicia cuando el usuario inicia el sistema, el usuario completa los campos usuario y contraseña, y indica realizar la autenticación, el sistema muestra la ventana principal.	
Referencias	RF 5
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- El operador inicia el sistema. 3- El operador entra los datos, usuario y contraseña. 4- El operador indica al sistema realizar la autenticación.	2- Muestra la interfaz “Autenticación” 5- Verifica que el usuario y la contraseña sean correctos. 6- El sistema muestra la ventana principal.
Flujo alternativo	
Acción del actor	Respuesta del sistema

	3.1- Si el usuario y la contraseña no son válidos, el sistema emite un mensaje “usuario o contraseña incorrectos”.

2.7.6 CU Actualizar la consola gráfica

Nombre del caso de uso	Actualizar de la consola gráfica.
Actores	Reloj del sistema.
Propósito	Actualizar los objetos gráficos presentes en la consola gráfica.
Resumen: El caso de uso inicia cuando transcurre un período de tiempo y el reloj del sistema hace un pedido de variables, luego actualiza los objetos gráficos mostrados en el despliegue.	
Referencias	RF 7
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Transcurre un período de tiempo y el reloj del sistema realiza un pedido de variables.	2- El sistema recepciona los datos. 3- Actualiza los objetos gráficos mostrados en el despliegue.

2.7.7 CU Actualizar sumario de alarmas

Nombre del caso de uso	Actualizar sumario de alarmas.
Actores	Reloj del sistema.
Propósito	Actualizar los datos de las alarmas existentes y visualizar las nuevas alarmas.

Resumen: El caso de uso inicia cuando transcurre un período de tiempo y el reloj del sistema emite una señal para que se actualicen las alarmas existentes en el sumario.	
Referencias	RF 8
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Transcurre un período de tiempo y el reloj del sistema emite la señal para actualizar las alarmas.	2- El sumario hace el pedido de las alarmas existentes en el sistema. 3- Actualiza la vista que visualiza las alarmas.

2.7.8 CU Actualizar sumario de puntos

Nombre del caso de uso	Actualizar sumario de puntos.
Actores	Reloj del sistema.
Propósito	Actualizar los datos de los puntos activos en el sistema.
Resumen: El caso de uso inicia cuando transcurre un período de tiempo y el reloj del sistema emite una señal para que se actualicen los puntos activos en el sistema.	
Referencias	RF 9
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- Transcurre un período de tiempo y el reloj del sistema emite la señal para actualizar los puntos activos en el	2- El sumario hace el pedido de los puntos activos existentes en el sistema. 3- Actualiza la vista que visualiza los puntos.

sistema.	

2.7.9 CU Navegar despliegue

Nombre del caso de uso	Navegar entre despliegues
Actores	Operador
Propósito	Que el usuario pueda visualizar los distintos despliegues existentes en el sistema.
Resumen: El caso de uso inicia cuando el operador decide cambiar el despliegue que se está visualizando, el operador indica el despliegue a mostrar, y el sistema lo gestiona y lo muestra.	
Referencias	RF 10
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- El operador selecciona la acción cambiar despliegue. 3- El operador selecciona el despliegue a visualizar.	2- El sistema muestra los despliegues a los que tiene permiso. 4- El sistema inicia la actualización de los recursos existentes en el despliegue. 5- El sistema muestra el despliegue.

2.7.10 CU Cargar configuración

Nombre del caso de uso	Cargar configuración
Actores	Operador

Propósito	Cargar la configuración del proyecto
Resumen: El caso de uso inicia cundo el operador inicia el sistema y termina con la construcción del proyecto.	
Referencias	RF 10
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
1- El operador inicia el sistema. 3- Selecciona el archivo xml que desea cargar.	2- Muestra una ventana con los directorios existentes. 4- Verifica que el archivo pueda ser cargado. 5- Construye el proyecto.
Flujo alternativo	
Acción del actor	Respuesta del sistema
	4.1- Se notifica un mensaje de error.

2.7.11 CU Adquisición de campo

Nombre del caso de uso	Adquisición de campo.
Actores	Reloj del sistema.
Propósito	Proveer al sistema de toda la información que presentan los dispositivos de campo.
Resumen: El caso de uso inicia cuando transcurre un período de tiempo y el reloj del sistema señala que se deben recolectar todos los datos de campo.	
Referencias	RF 11
Curso normal de los eventos	
Acción del actor	Respuesta del sistema

1- Transcurre un período de tiempo y el reloj del sistema señala que se deben recoger la información de campo.	2- El sumario hace el pedido de datos recibidos. 3- Actualiza los administradores de cada tipo de dato según correspondan.

2.8 Conclusiones

En este capítulo se comenzó a profundizar en el desarrollo de la propuesta de solución, obteniéndose una lista de las funcionalidades que debe tener el sistema, las mismas fueron representadas mediante diagramas de casos de uso, y luego fueron descritas todas las acciones que realiza el actor y el sistema en general. A partir de aquí se puede comenzar a construir el sistema, cumpliendo con todos los requerimientos y las funcionalidades que se consideraron en este capítulo.

Capítulo 3: Descripción de la Solución Propuesta

3.1 Introducción

En este capítulo se hace una descripción de las principales características que corresponden a la implementación del visualizador del SCADA Guardián del Alba. En el primer epígrafe se muestra la arquitectura y los patrones utilizados. Posteriormente se expondrán los diagramas de clase de diseño separado por paquetes y funcionalidades con la descripción de las clases que lo conforman.

3.2 Arquitectura de la solución

"El diseño arquitectónico define la relación entre los elementos estructurales principales del software, los patrones de diseño que se pueden utilizar para lograr los requisitos que se han definido para el sistema, y las restricciones que afectan a la manera en que se pueden aplicar los patrones de diseño arquitectónicos" [3].

Los patrones arquitectónicos expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos; así como ayudan a especificar la estructura fundamental de una aplicación. [2]

La arquitectura del visualizador del SCADA Guardián del Alba es una solución orientada al patrón arquitectónico Modelo-Vista. Este patrón describe una forma de desarrollar aplicaciones software separadas en dos capas, posibilitando un alto nivel de encapsulamiento de las responsabilidades y reduce al máximo el acoplamiento.

- **Modelo:** Es la representación específica de la información con la cual el sistema opera, la lógica de negocios y sus mecanismos de persistencia.
- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

La principal ventaja que posee es que las alteraciones de una de las capas no tienen por qué afectar a las demás, posibilitando que en caso de cambios solo se ataque a la capa requerida sin tener que revisar las demás.

También se hizo uso del patrón de diseño Comando, una de las principales ventajas que presenta es la reutilización de código, pues cada comando encapsula la lógica para realizar una tarea determinada; haciendo posible que cualquiera que esté interesado en realizar dicha tarea solo tiene que invocar el comando adecuado.

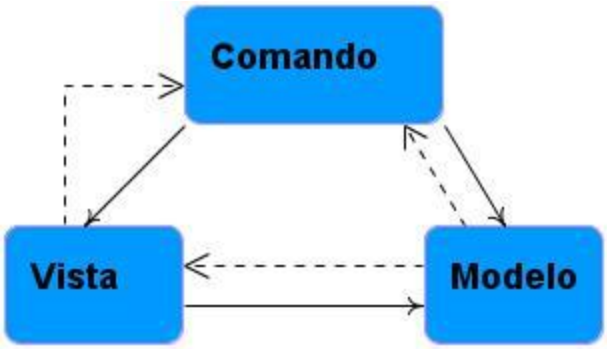


Fig. 7 Diagrama de cómo se relacionan el Modelo-Vista-Comando

El diagrama de la figura muestra la relación entre el modelo, la vista y el comando. Las líneas sólidas indican una asociación directa, y las punteadas una indirecta.

3.3 Diseño del sistema

El sistema está compuesto por una serie de paquetes que conforman su arquitectura, a continuación se presentan distintos diagramas, que muestran la solución planteada.

3.3.1 Paquete Base

En este Paquete se encuentran las clases sobre las cuales se soporta parte del sistema. Como principales elementos de este Paquete se encuentran las clases System, Viewer, AlarmManager y Builder.

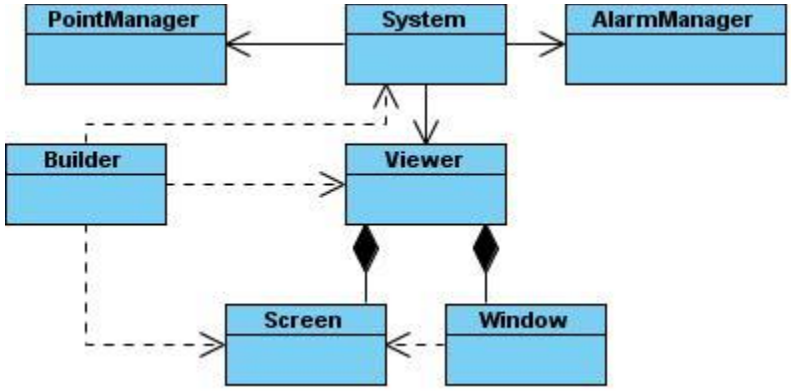


Fig. 8 Diagrama del paquete Base.

Nombre: System	
Descripción: Clase encargada de modelar el sistema.	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
viewer	Viewer
pointManager	PointManager
acquisitionModel	AcquisitionModel
alarmManager	AlarmManager
Para cada responsabilidad:	
Nombre:	PointManager* getPointManager ()const
Descripción:	Devuelve el administrador de puntos del sistema.
Nombre:	AlarmManager* getAlarmManager ()const
Descripción:	Devuelve el administrador de alarmas del sistema.
Nombre:	AcquisitionModel* getAcquisitionModel () const
Descripción:	Devuelve los datos persistentes en el sistema.

Tabla # 3: Descripción de la clase de Diseño "System".

Nombre: Viewer	
Descripción: Es la clase principal que contiene los despliegues y las ventanas donde se mostraran.	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
windowsCollection	WindowsCollection
screensCollection	ScreensCollection
Para cada responsabilidad:	
Nombre:	Window* createMainWindow ()
Descripción:	Crea una nueva ventana principal y devuelve una referencia.
Nombre:	void* addScreen (Screen *screen)
Descripción:	Adiciona un nuevo objeto de tipo Screen en la colección.
Nombre:	Screen* getScreen(unsigned int key) const;
Descripción:	Devuelve el objeto de tipo Screen que tenga la llave pasada por parámetros.
Nombre:	const ScreensCollection& getScreensCollection() const;
Descripción:	Devuelve la colección de objetos de tipo Screen.

Tabla # 4: Descripción de la clase de Diseño "Viewer".

Nombre: AlarmManager	
Descripción: Clase encargada de la administración de las alarmas en el sistema.	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
alarmsCollection	AlarmsCollection
silentAlarms	QSet<AlarmId>
mutex	QMutex

alarmSoundManager	AlarmSoundManager
Para cada responsabilidad:	
Nombre:	void add(Alarm *Alarm);
Descripción:	Adiciona una nueva alarma en la colección de alarmas activas.
Nombre:	bool existAlarm(Feature::AlarmId id);
Descripción:	Verifica si existe una alarma activa con el identificador pasado por parámetro
Nombre:	IO::AlarmData* getAlarm(Feature::AlarmId id);
Descripción:	Retorna la alarma activa que machee con el identificador pasado por parámetro.
Nombre:	bool isVisualize(Alarm *AlarmData);
Descripción:	Retorna verdadero si la alarma se va a visualizar, en caso contrario retorna falso.
Nombre:	bool isMomentary(Feature::AlarmTypeIdAlarmTypeId);
Descripción:	Retorna verdadero si la alarma es momentánea.
Nombre:	IO::AlarmData* convertToAlarmData(const Alarm *Alarm);
Descripción:	Convierte una alarma proveniente del middleware a una alarma del sistema para mostrar en el sumario de alarma.
Nombre:	void AlarmManager::soundSeverity()
Descripción:	Busca la alarma de menor severidad y define el nuevo sonido a reproducir.
Nombre:	void soundSeverityAlarm(IO::AlarmData *alarmData);
Descripción:	Define si el sonido debe ser cambiado por la llegada de una nueva alarma.
Nombre:	void soundMode(bool play, AlarmSeverityId severity);
Descripción:	Cambia el sonidoa emitir en el administrador de sonido.
Nombre:	bool isAlarmSilent(AlarmIdalarmId);
Descripción:	Retorna verdadero si la alarma con el identificador pasado por parámetro está silenciada, en caso contrario retorna

	falso.
Nombre:	AlarmsCollection getAlarmsCollection();
Descripción:	Retorna las alarmas activas en el sistema.
Nombre:	AlarmsCollection getFiveLastAlarms();
Descripción:	Retorna las últimas cinco alarmas activas en el sistema.
Nombre:	void silenceAlarm(AlarmId alarmId);
Descripción:	Silencia la alarma con el identificador pasado por parámetros.

Tabla # 5: Descripción de la clase de Diseño "AlarmManager".

Nombre: Builder	
Descripción: Es la clase que lee el XML para construir el proyecto.	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
file	QFile
Para cada responsabilidad:	
Nombre:	void loadPlugin();
Descripción:	Carga el plugin graphics.
Nombre:	void start();
Descripción:	Inicia la lectura del archivo XML, para la construcción del proyecto
Nombre:	readProyect(QXmlStreamReader&in);
Descripción:	Inicia la lectura del proyecto.
Nombre:	void readNodeCollection(QXmlStreamReader& in);
Descripción:	Es el encargado de leer la colección de nodos del archivo XML.

Nombre:	void readNode(QXmlStreamReader& in);
Descripción:	Lee un nodo del archivo XML.
Nombre:	void readModuleCollection(QXmlStreamReader& in);
Descripción:	Lee la colección de módulos del archivo XML.
Nombre:	void readScreenCollection(QXmlStreamReader& in);
Descripción:	Lee la colección de objetos de tipo Screen y los almacena en la instancia de tipo Viewer.
Nombre:	Screen* readScreen(QXmlStreamReader& in);
Descripción:	Lee el XML y devuelve un objeto de tipo Screen.
Nombre:	void readGraphicsCollection(QXmlStreamReader& in, Screen *screen);
Descripción:	Lee la colección de gráficos del archivo XML.
Nombre:	GraphicObject* readGraphic(QXmlStreamReader& in);
Descripción:	Lee un gráfico del archivo XML.

Tabla # 6: Descripción de la clase de Diseño "Builder".

3.3.2 Paquete View

En este paquete se encuentran las clases encargadas de visualizar datos, como pueden ser alarmas o puntos. Las clases principales en este paquete son: Summary, AlarmSumary y PointSumary.

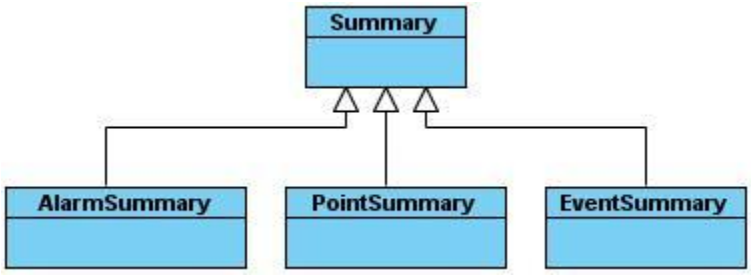


Fig. 9 Diagrama del paquete View.

Nombre: Summary	
Descripción: Clase base para todos los sumarios del sistema.	
Tipo de clase:	
Atributo:	**Tipo:**
tableView	TableView
paginator	SortFilterPaginator
tableModel	QAbstractTableModel
Para cada responsabilidad:	
Nombre:	void nextPage();
Descripción:	Muestra la página siguiente.
Nombre:	void backPage();
Descripción:	Muestra la página anterior.
Nombre:	void numberRowsChanged(int option);
Descripción:	Cambia la cantidad de columnas que se están visualizando.
Nombre:	void textFilterChanged(const QString& text);
Descripción:	Filtra el contenido por una columna determinada.
Nombre:	void searchParameterChanged(int option);
Descripción:	Cambia la columna por la que se realiza el filtrado.

Tabla # 7: Descripción de la clase de Diseño "Summary".

Nombre: AlarmSummary	
Descripción: Hereda de la clase Summary.	
Tipo de clase:	
Atributo:	**Tipo:**
Para cada responsabilidad:	

Nombre:	void silenceAlarmAct();
Descripción:	Silencia la alarma seleccionada.
Nombre:	void silenceAllAlarmAct();
Descripción:	Silencia todas las alarmas existentes en el sumario.
Nombre:	void recognizeAlarm();
Descripción:	Reconoce la alarma seleccionada.
Nombre:	void recognizeAllAlarm();
Descripción:	Reconoce todas las alarmas existentes en el sumario.

Tabla # 8: Descripción de la clase de Diseño "AlarmSummary".

Nombre: PointSummary	
Descripción:	
Tipo de clase:	
Atributo:	**Tipo:**
Para cada responsabilidad:	
Nombre:	void showPoint();
Descripción:	Reconoce el tipo de punto que se quiere visualizar y en dependencia del tipo manda a ejecutar la ventana adecuada.
Nombre:	void showDigitalPoint(SCADA::ACQUISITION::DigitalPoint*digitalPoint);
Descripción:	Ejecuta la ventana para visualizar los detalles de un punto digital.
Nombre:	void showAnalogocPoint(SCADA::ACQUISITION::AnalogicPoint *analogicPoint);
Descripción:	Ejecuta la ventana para visualizar los detalles de un punto analógico.

Tabla # 9: Descripción de la clase de Diseño "PointSummary".

3.3.3 Subsistema Entrada/Salida

Representa como se recibe la información desde los diferentes módulos del SCADA al visualizador. La clase IOManager es la encargada de administrar la entrada/salida de datos al sistema, y lo hace a través de las implementaciones concretas del middleware que son las responsables de recibir el flujo de datos. En este diagrama se puede ver la implementación del patrón Fábrica Abstracta que posibilita la recepción de datos de distintos tipo de middleware, como por ejemplo ICE o SOCKET. Para la actualización de las variables en los despliegues la clase IOManager es la responsable de informarle a los objetos activos de tipo Screen (representan los despliegues) los nuevos valores provenientes del middleware. Cada objeto de tipo Screen actualiza sus objetos gráficos a través de objetos InputSlot, que están conectados a dichos objetos.

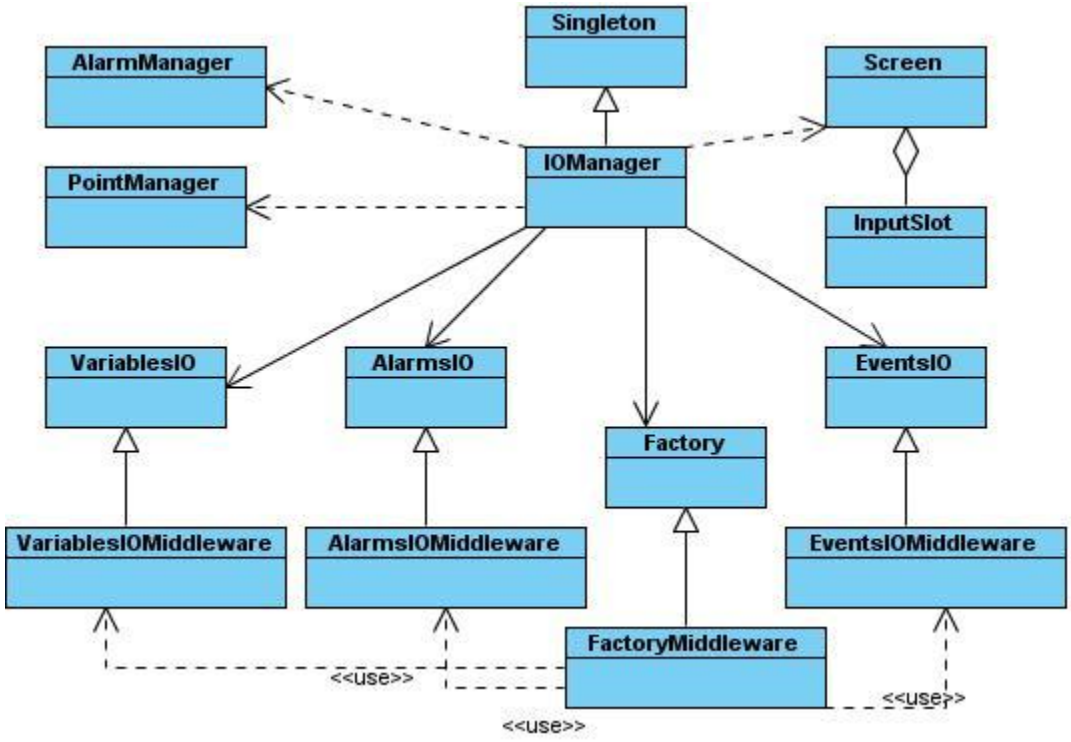


Fig. 10 Diagrama general del mecanismo deentrada/salida.

Nombre:IOManager
Descripción: Clase encargada de la administración de la Entrada/Salida en el visualizador.

Tipo de clase: Controladora	
Atributo:	**Tipo:**
factory	Factory
alarmsIO	AlarmsIO
commandIO	commandIO
variableIO	VriableIO
connection	Connection
stopRetrieve	bool
mutex	QMutex
waitCondition	QWaitCondition
Para cada responsabilidad:	
Nombre:	void connectRTDBModule()
Descripción:	Crea la conexión para iniciar la recepción de datos desde la base de datos en tiempo real.
Nombre:	void retrieveInput()
Descripción:	Inicia la recolección de datos desde la bases de datos en tiempo real.
Nombre:	void run()
Descripción:	Es donde se realizan los pedidos de los lotes de datos recibidos.
Nombre:	void nextVariableBatch()
Descripción:	Pide el próximo lote de variables recibido.
Nombre:	void nextAlarmsBatch()
Descripción:	Pide el próximo lote de alarmas recibidas.
Nombre:	void updateScreens (Variable *var)
Descripción:	Actualiza a todos los objetos de tipo Screen activo, con el nuevo valor de la variable recibida.
Nombre:	void sendCommand(CommandRequest *command);

Descripción:	Envía comandos al resto de los módulos del SCADA.

Tabla # 10: Descripción de la clase de Diseño "IOManager".

Nombre: Variable	
Descripción: Almacena el valor recogidos en el campo para una variable.	
Tipo de clase: Entidad	
Atributo	**Tipo**
id	VarId
value	VariableValue
timestamp	Timestamp
Para cada responsabilidad:	
Nombre:	const VarIdgetId ( )
Descripción:	Retorna el valor de atributo id
Nombre:	virtual VariableValue getValue() const
Descripción:	Retorna el valor del atributo valor
Nombre:	virtual void setValue(VariableValue value)
Descripción:	Asigna un nuevo valor al atributo valor.
Nombre:	Timestamp getTimestamp() const
Descripción:	Retorna el valor del atributo timestamp.

Tabla # 11: Descripción de la clase de Diseño "Variable".

Nombre: VariableIO
Descripción: Interfaz que debe implementar todas las clases que manejen información de entrada y salida.
Tipo de clase: Interfaz

Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	virtual constVariableCollection& getNextBatch( ) = 0

Tabla # 12: Descripción de la clase de Diseño "VariableIO".

3.4 Diagramas de las funcionalidades más importantes del sistema

3.4.1 Descripción diagrama "Mecanismo adquisición variables"

Este diagrama representa el mecanismo de actualización de las variables en la consola gráfica: IOManager es la clase encargada de proveer los datos a los despliegues activos (objetos de tipo Screen), lo hace a través de una instancia de la clase VariablesIO, que es la interfaz que debe implementarse de acuerdo al tipo de middleware que se utilice. Los objetos de tipo Screen tienen una colección de InputSlot, cada objeto de este tipo tiene el identificador de la variable a la que representa y está conectado a todos los objetos gráficos interesados en la actualización.

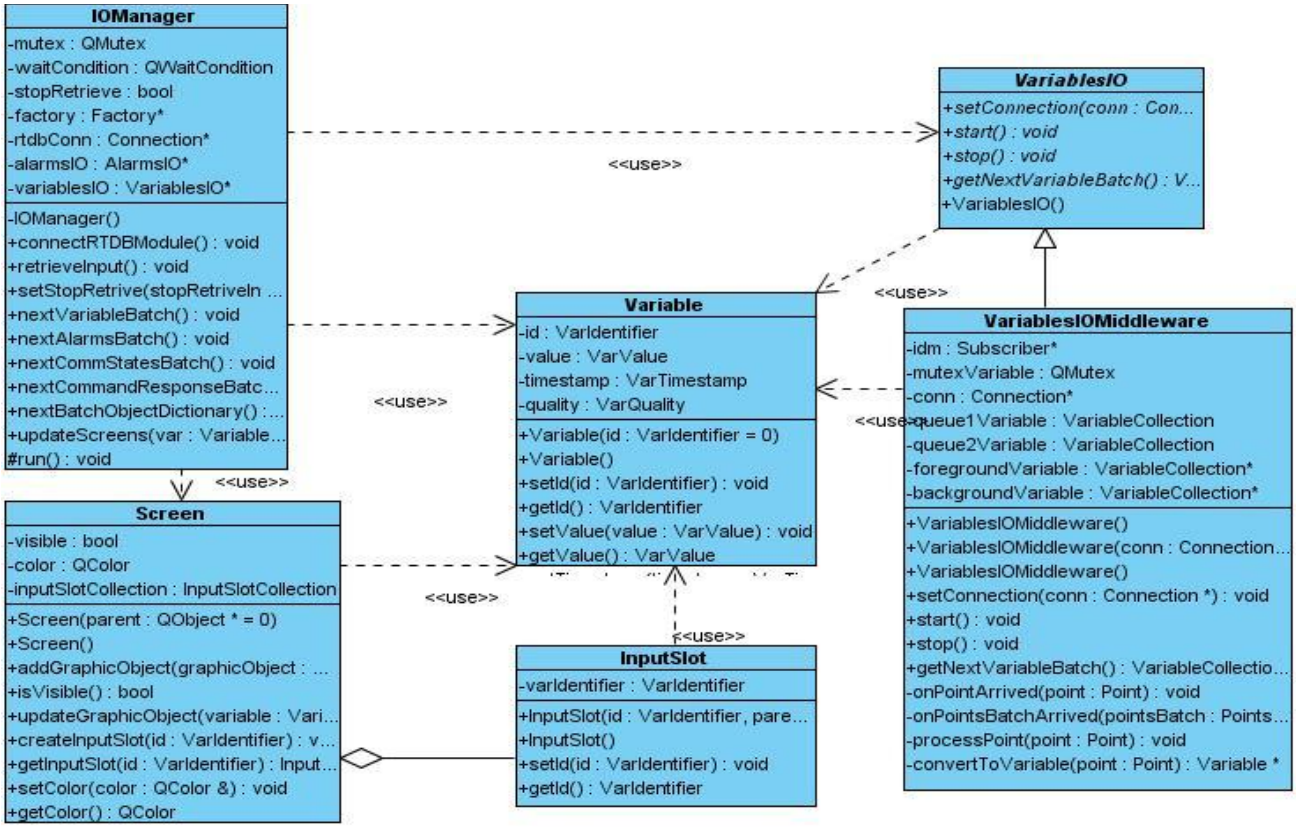


Fig. 11 Mecanismo adquisición de variables.

3.4.2 Descripción diagrama "Mecanismo adquisición alarmas"

Representa el mecanismo de adquisición de alarmas: IOManager es la clase encargada de proveer los datos a los objetos interesados, para la recolección de las alarmas, posee una instancia de la clase AlarmsIO, que es la interfaz que debe implementarse de acuerdo al mecanismo de acceso a datos. Además la clase IOManager es la encargada de almacenar las alarmas que se recolectan, en la clase AlarmManager, responsables de procesarlas y guardarlas para el posterior uso.

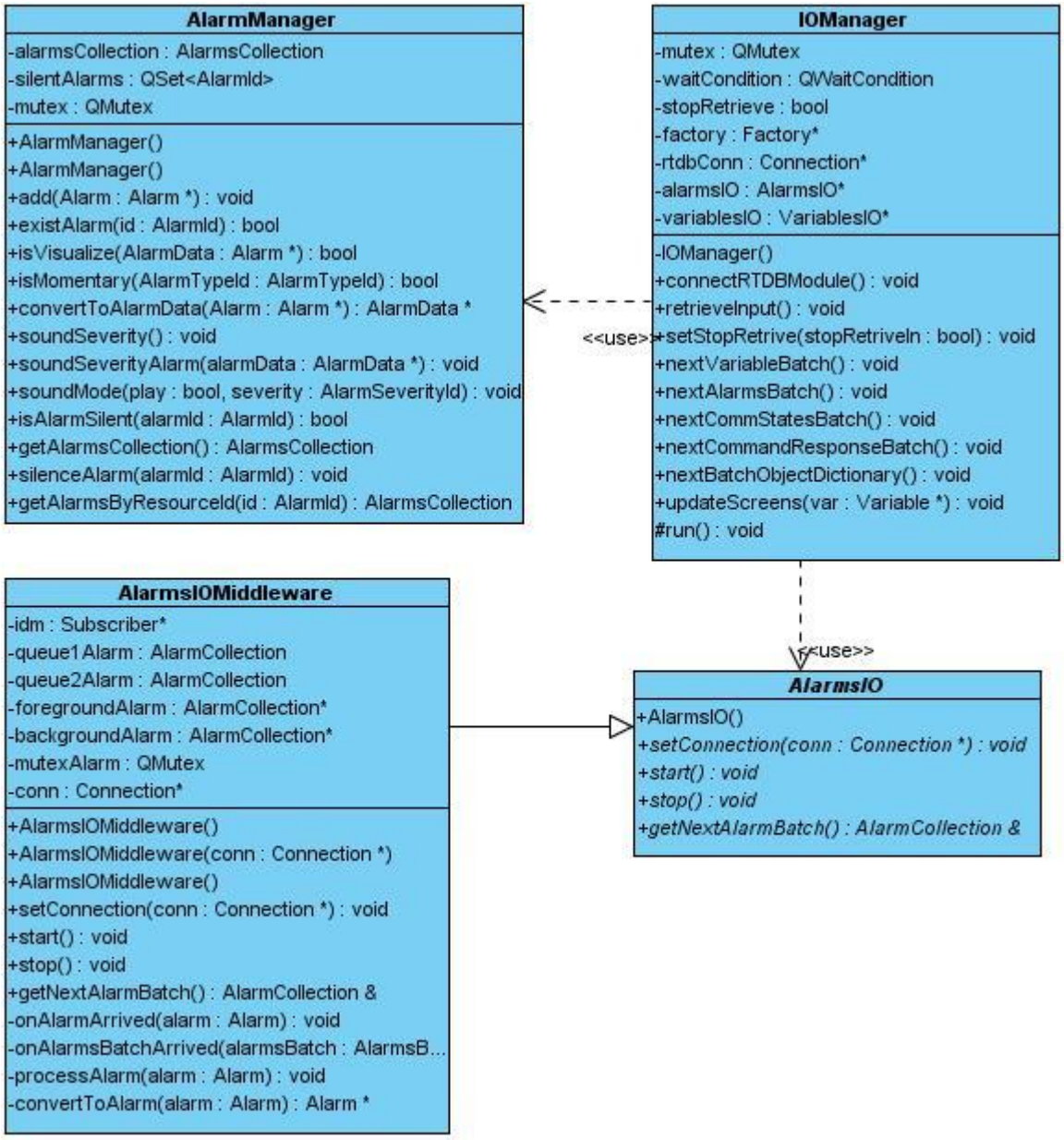


Fig. 12 Mecanismo adquisición de alarmas.

3.4.3 Descripción diagrama "Sumarios de alarmas"

El diagrama representa una vista del sumario de alarmas. Summary es una clase que presenta todas las características comunes para los sumarios, como filtrado, paginado, mecanismos de ordenamiento entre otras características; AlarmSummary es la clase que define al Sumario de Alarmas, presenta acciones propias como silenciar, reconocer,

silenciar todas y reconocer todas, para realizar estas acciones, al igual que la recolección de datos, utiliza la clase AlarmManager que es la encargada del procesamiento y almacenamiento de las alarmas.

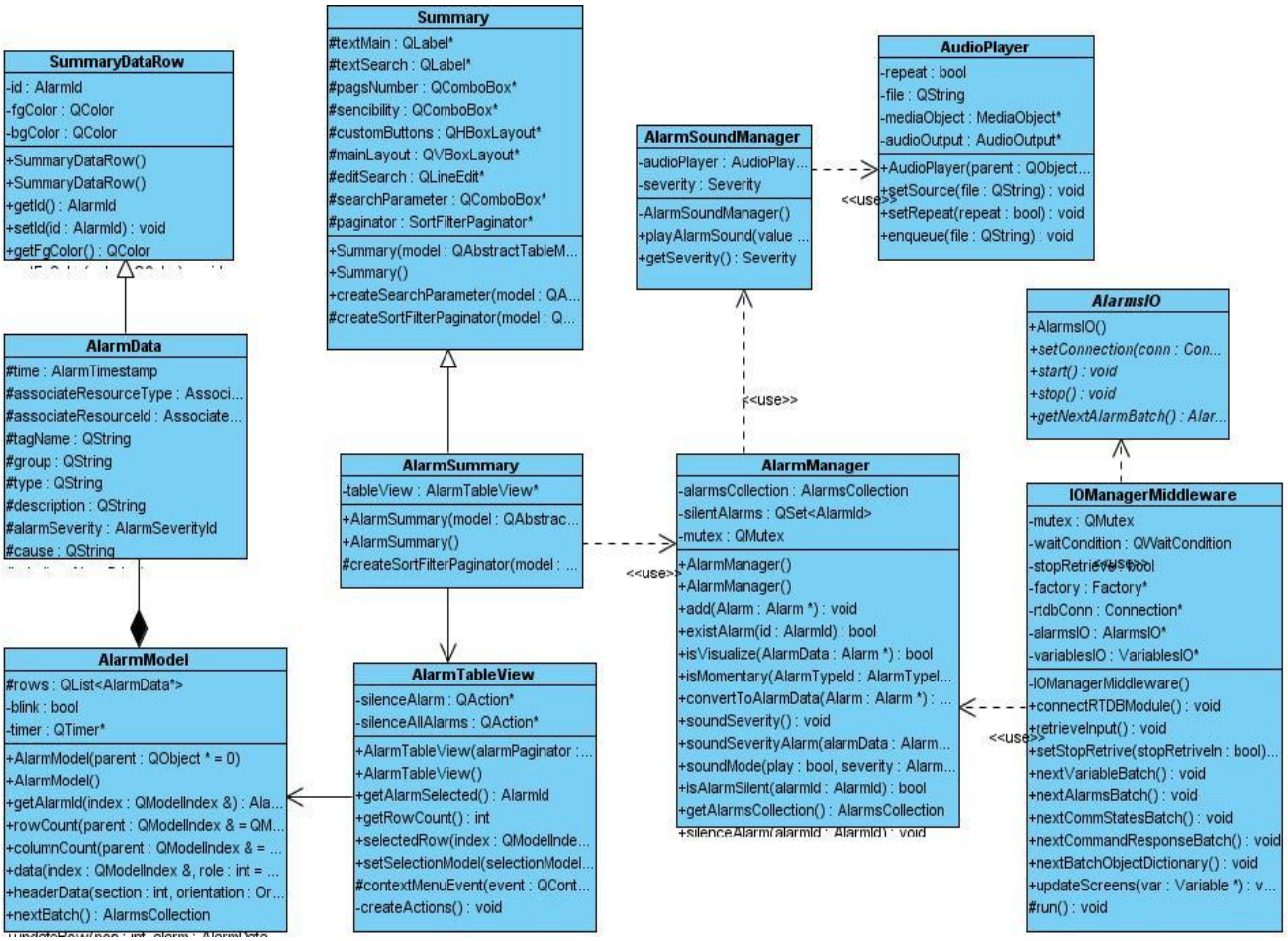


Fig. 13 Diagrama del sumario de alarmas.

3.5 Conclusiones

En el capítulo se procedió con la realización del diseño e implementación del sistema, donde se expuso la arquitectura y los patrones utilizados. Se presentaron los diagramas de clases, donde se mostraron las relaciones de las diferentes clases que componen nuestro sistema y una descripción de cada una de ellas.

Conclusiones

Luego del estudio realizado se logró el desarrollo del sistema que servirá como base para futuras labores de implementación, además, con este trabajo, se brinda una guía para familiarizar a implementadores, clientes y usuarios finales con los conceptos y actividades relacionadas con la visualización de sistemas SCADA.

Con el estudio de las aplicaciones con características similares a la que se desarrolla se obtuvieron diseños equivalentes a los deseados, los cuales sirvieron como punto de partida para obtener una arquitectura flexible, que permita la modificación o incorporación de nuevos requerimientos.

La modelación del sistema se desarrolló siguiendo la metodología RUP, y se utilizó el lenguaje de modelado UML para la modelación de las diferentes fases por las que transitó el desarrollo de este trabajo. Se definieron los requerimientos del software, tanto funcionales como no funcionales, estructurándose además, el modelo de casos de uso del sistema. Luego se realizó el diseño a través de diagramas de clases para finalmente proceder con la implementación, obteniéndose una aplicación que permite controlar los procesos industriales de forma automática desde la pantalla del ordenador, capaz de adaptarse fácilmente a los diferentes procesos industriales.

Los objetivos propuestos al comienzo de este trabajo se cumplieron satisfactoriamente, además se han incluido una serie de recomendaciones que deben tenerse en cuenta para el trabajo futuro.

Recomendaciones

- Continuar el desarrollo de este sistema y agregarle nuevas funcionalidades adecuándolo aun más a las demandas de los usuarios haciéndolo más útil y provechoso.
- Utilizar el sistema de Entrada/Salida y envío de comandos por otros visualizadores como el web o el de móviles.
- Realizar optimizaciones de rendimiento al sistema.

Referencias Bibliográficas

1. Sehara, Yossel Luis. Implementación de un modelo para la configuración de un sistema SCADA. Ciudad de La Habana: julio, 2008.

2. Pressman, Roger S. "Ingeniería de Software. Un enfoque práctico." 5ta Edición (traducción de la edición original en inglés "SoftwareEngineering. A practical approach"), Editorial Mac Graw Hill, Madrid, España, 2001.

3. Shaw, M. y Garlan, D. "Software Architecture". Editorial Prentic-Hall, New York, E.U.A. 1996.

4. Rodriguez Penin, Antonio. Sistemas SCADA. Madrid, Marcombo, 2007.

5. Booch, G., Rumbaugh, J., Jacobson, I. El Proceso Unificado de Desarrollo de Software. España, Addison Wesley, 2000.

6. Fundación Eclipse, Proyecto Eclipse, www.eclipse.org/projects/project_summary.php?projectid=eclipse, 2011.

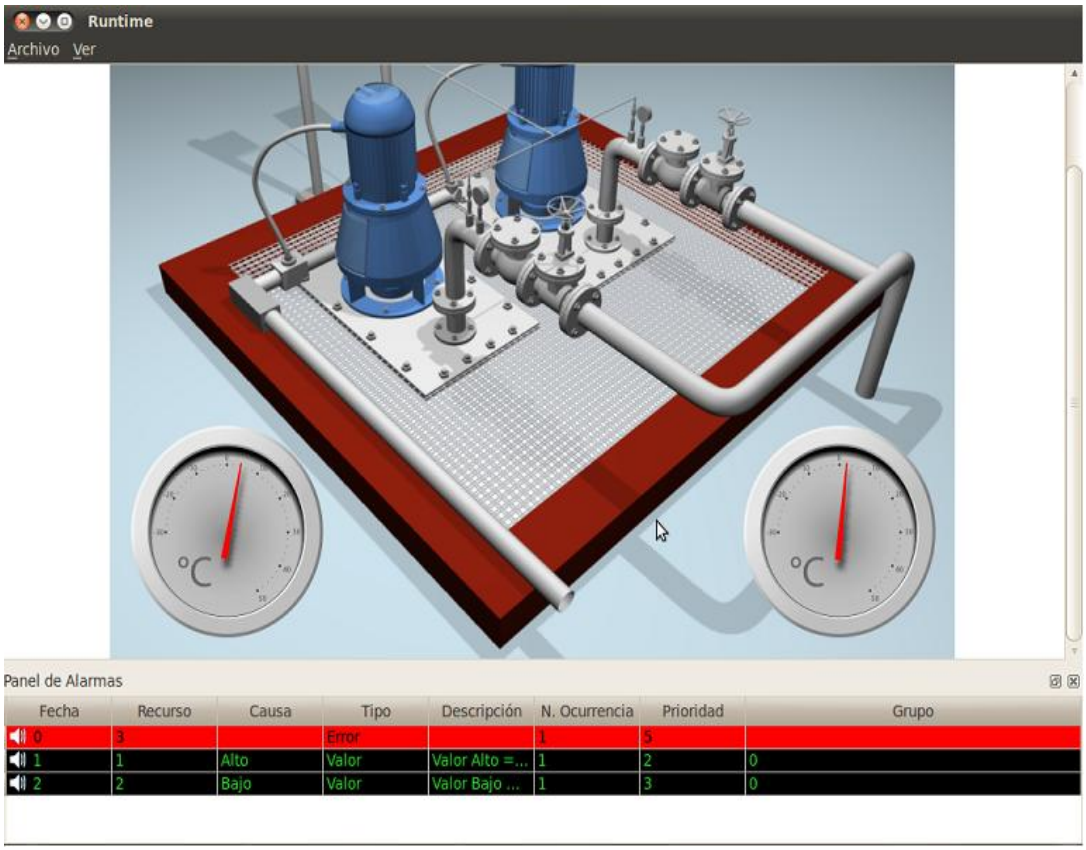
7. http://www.tiempo-real.org/tiempo-real-definicion

Bibliografía

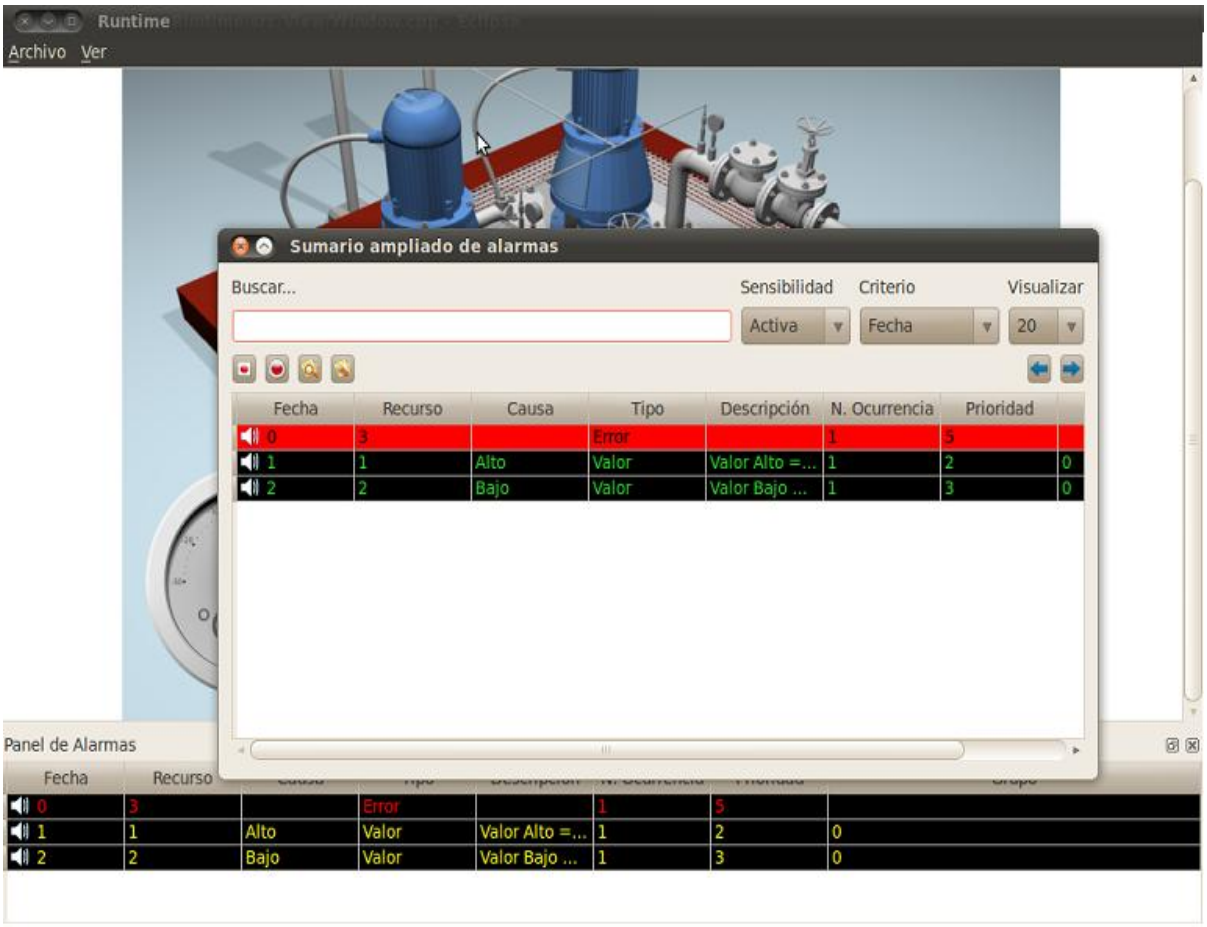
1. Stroustrup, Bjarne. The C++ Programming Language (Third Edition). s.l.: Addison-Wesley, 1997.
2. Rodríguez Penin, Aquilino. 2006. Sistemas SCADA, 2006.
3. Villalobos Ordaz, Gustavo, y otros. 2006. Medición y Control de Procesos Industriales. 2006.
4. Pérez Feria Jordenys. Análisis de la arquitectura del módulo Interfaz Hombre Máquina del SCADA Nacional: s.n., 2008.
5. Booch, Grady. OBJECT-ORIENTED ANALYSIS AND DESIGN. California, ADDISON-WESLEY, 1998. 543 pág.
6. Progea. MoviconProgrammer Guide. Modena – Italia.
7. http://www.cplusplus.com/.
8. http://www.visual-paradigm.com/.
9. http://qt.nokia.com/.
10. http://www.progea.com/.
11. http://www.foxboroscada.com/home_english.htm.
12. http://automatas.org/redes/scadas.htm.
13. http://www.aptitudcapacitacion.com/.
14. http://free-scada.sourceforge.net/.

Anexos

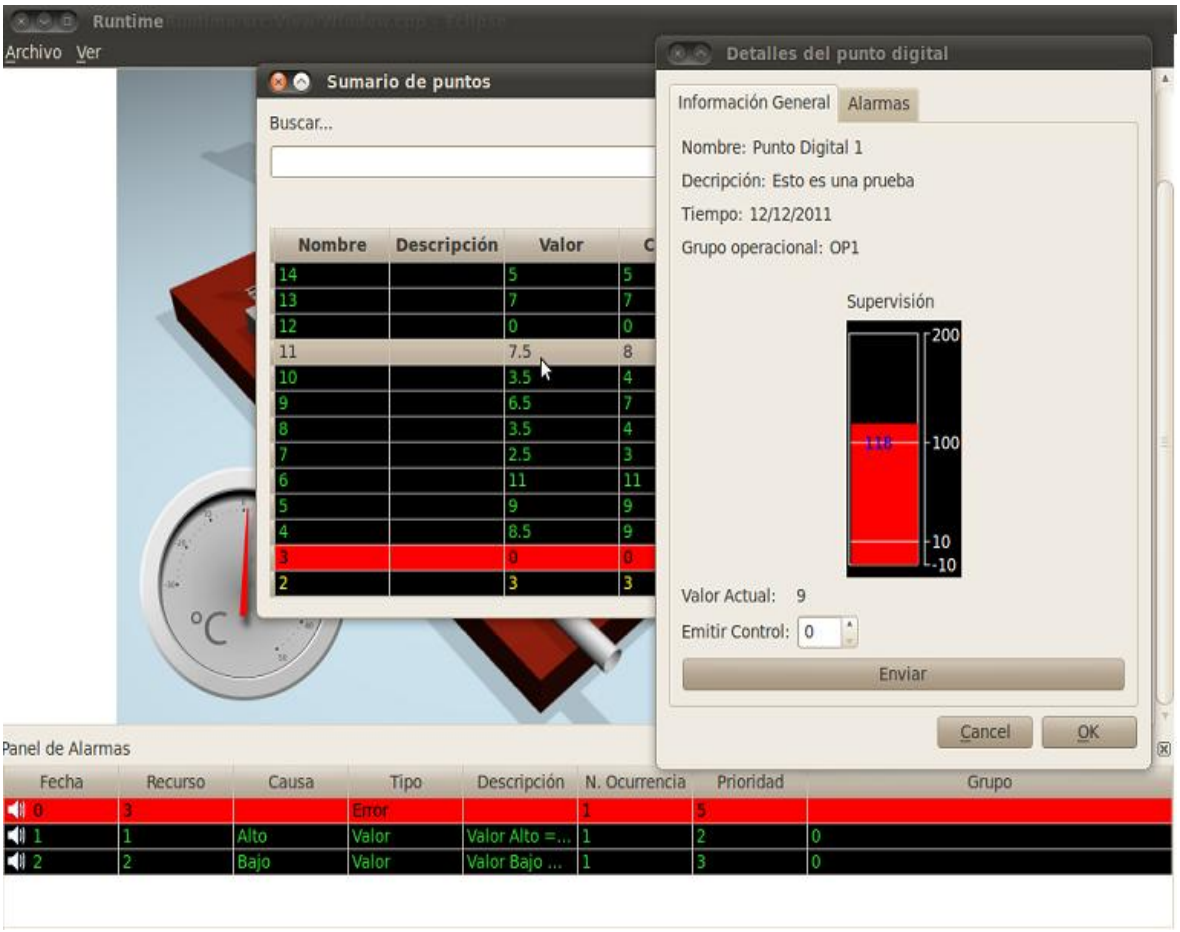
Anexo 1:



Anexo 2:


Runtime
Archivo Ver
Sumario ampliado de alarmas
Buscar...
Sensibilidad
Criterio
Visualizar
Activa
Fecha
20
Fecha Recurso Causa Tipo Descripción N. Ocurrencia Prioridad
0 3 Error 1 5
1 1 Alto Valor Valor Alto =... 1 2 0
2 2 Bajo Valor Valor Bajo ... 1 3 0
Panel de Alarmas
Fecha Recurso
0 3 Error 1 5
1 1 Alto Valor Valor Alto =... 1 2 0
2 2 Bajo Valor Valor Bajo ... 1 3 0

Anexo 3:


Runtime
Archivo Ver
Sumario de puntos
Buscar...
Nombre Descripción Valor
Detalles del punto digital
Información General
Alarmas
Nombre: Punto Digital 1
Decripción: Esto es una prueba
Tiempo: 12/12/2011
Grupo operacional: OP1
Supervisión
200
100
10
-10
Valor Actual: 9
Emitir Control: 0
Enviar
Cancel
OK
Panel de Alarmas
Fecha Recurso Causa Tipo Descripción N. Ocurrencia Prioridad Grupo
0 3 Error 1 5
1 1 Alto Valor Valor Alto =... 1 2 0
2 2 Bajo Valor Valor Bajo ... 1 3 0

Glosario de Términos

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

Adquisición de datos: Consiste en recolectar un conjunto de variables medidas en forma física a través de diferentes elementos (sensores, transductores...) para ser convertidas en digital y que puedan ser utilizadas por el computador.

Arquitectura del sistema: Describe cómo las funcionalidades del mismo se distribuyen sobre un número de componentes lógicos y cómo estos componentes se interrelacionan.

Autenticar: Verificación de la identidad de una persona o de un proceso para acceder a un recurso o poder realizar determinada actividad, así como también para detectar la veracidad del origen de un mensaje electrónico.

Evento: Conjunto de acciones que proceden de la ejecución o activación de otra acción y que ha sido registrada, la combinación de estas acciones también pueden dar como resultado otro evento en particular o una serie de eventos.

Multiplataforma: Sistema que puede ejecutarse en diversos sistemas operativos y tipos de hardware.

Sistema distribuido: Es un sistema desarrollado de forma modular (por componentes) que permite el funcionamiento de los módulos en nodos diferentes, garantizando la comunicación entre ellos.

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

Requerimiento: En ingeniería de software, se define como "una condición o capacidad que un sistema debe cumplir".

Control: Se refiere al conjunto de acciones que permiten modificar el estado actual de los actuadores en campo a través del envío de comandos.

Comandos: Acciones que se llevarán a cabo en tiempo de ejecución del SCADA y que influirán sobre los recursos del sistema (despliegues, variables, alarmas, usuarios). Los comandos definen funcionalidades, utilizadas comúnmente en la creación de aplicaciones de supervisión sin la necesidad de programación de scripts, ejemplo de comandos son: abrir un despliegue, mostrar un menú, asignar un valor a una variable etc. Los comandos puedes ser clasificados según su funcionalidad como: comandos de variable (cambiar un variable), comandos de sistema (ejecutar un aplicación externa), comandos de usuario (abrir sesión), comandos de despliegues (mostrar un despliegue), comandos de alarmas (reiniciar, reconocer), comandos de reporte (mostrar un reporte).

Punto: Los puntos se refieren 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, digital, texto y vector.

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.

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, aun cuando no tenga conexión a las capas superiores del control.