Facultad 5

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

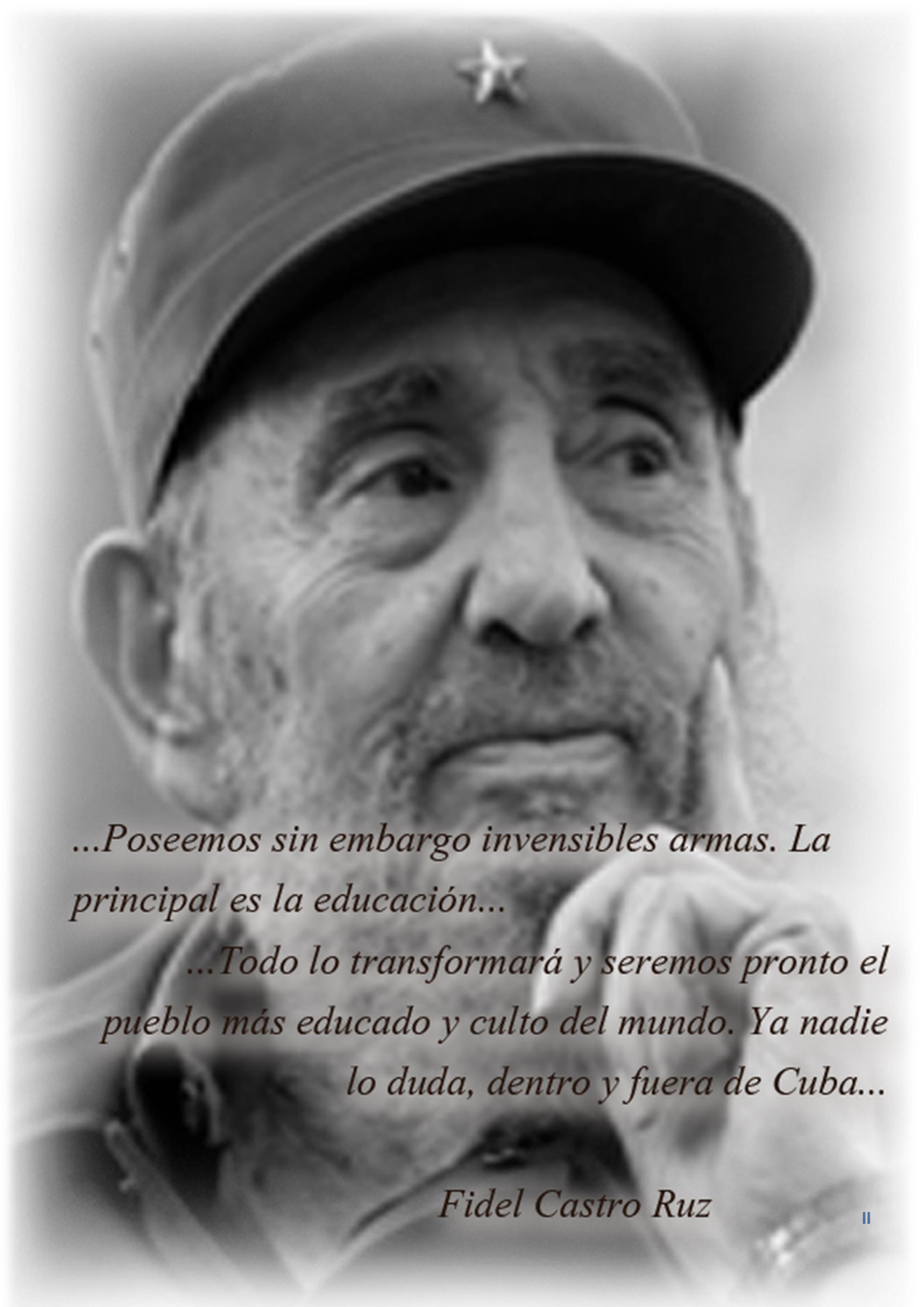
Título:

Mecanismo para la ejecución de acciones automatizadas en el HMI del SCADA SAINUX.

Autor:
Nayrobis Gonzalez Torres

Tutores:
Ing. Yoandri Matos De La Cruz
Ing. Ridel Oscar García Mora

La Habana, junio de 2015
"Año 57 de la Revolución"



...Poseemos sin embargo invensibles armas. La principal es la educación...

...Todo lo transformará y seremos pronto el pueblo más educado y culto del mundo. Ya nadie lo duda, dentro y fuera de Cuba...

Fidel Castro Ruz

Declaración de autoría

Declaro ser autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos del mismo, con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de _____ del año _____.

Nayrobis Gonzalez Torres

_____

Firma del Autor

Ing. Yoandri Matos De La Cruz

_____

Firma del Tutor

Ing. Ridel Oscar García Mora

_____

Firma del Tutor

Datos de contacto

Nombre y apellidos del tutor: Yoandri Matos De La Cruz.
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: Ingeniero en Ciencias Informáticas.
Correo electrónico: yoandri@uci.cu
Graduado en la Universidad de las Ciencias Informáticas (UCI), recién graduado en adiestramiento con 2 años de experiencia en la producción de software, específicamente en el desarrollo de sistemas SCADA.

Nombre y apellidos del tutor: Ridel Oscar García Mora.
Institución: Universidad de las Ciencias Informáticas (UCI).
Título: Ingeniero en Ciencias Informáticas.
Correo electrónico: rmora@uci.cu
Graduado en la Universidad de las Ciencias Informáticas (UCI), recién graduado en adiestramiento con 2 años de experiencia en la producción de software, específicamente en el desarrollo de sistemas SCADA.

Agradecimientos

A mi mamá, por todo lo que me ha enseñado durante estos años de mi vida, por ser incondicional, guía, amiga y compañera en los momentos buenos y malos.

A mi hermanito Carlitin que me da las fuerzas para seguir adelante solo con sus ocurrencias de niño.

A mi papá que a pesar de no ser el mejor padre se preocupa por mis problemas en la escuela.

A Carlos por ayudarme en todo lo que ha podido durante estos años de la carrera. Por ser mi compañero, por la paciencia que ha mantenido para tratar de que yo salga adelante de la mejor forma posible y, sobre todo, por aguantar mis malos humores. Eres y serás muy especial en mi vida.

Al profe Roberto Millet por su apoyo desde principios de la universidad y por confiar en mí.

A los profesores Lannie Octavio, José Antonio Aragón, Andy H. y al tribunal por colaborar con la corrección del documento y en mi preparación para la pre-defensa y defensa.

A mis tutores Yoandri y Ridel, principalmente a Yoandri por estar siempre conmigo y darme todo su apoyo.

A mis amigos que de alguna forma me dieron ánimo para que siguiera y no me rindiera. A Dianet, Osmel, los dos Alejandro, Javier A., y a mis compañeros del grupo 5501 y 5502 por estar en los momentos buenos y malos.

En fin, agradezco a todos los que han contribuido con la realización de este trabajo y con mi formación profesional.

Dedicatoria

Hay una persona que ocupa el primer lugar en mi vida y esa es mi mamá. Eres tú mami a quien quisiera comenzar dedicando todo mi esfuerzo y sacrificio durante todos estos años, porque me impulsas a luchar y a buscar mis sueños.

En segundo lugar, a mi hermano que es también mi fuente de inspiración, porque sé que cuando no estoy lloras y me extrañas, y cuando estoy no me sueltas y me sigues como si fuera tu segunda mamá.

Y a todas las personas que de alguna forma se regocijan de todos mis éxitos.

Resumen

Cada año se imponen nuevos criterios que requieren del aumento de los niveles de eficiencia y optimización de los procesos industriales. Uno de los mecanismos más importantes para la consecución de estos objetivos, es mediante el uso de los sistemas de supervisión, control y adquisición de datos (SCADA), para la automatización de los procesos que componen la cadena de producción. El Sistema de Automatización Industrial basado en GNU/Linux (SAINUX) es un software de tipo SCADA, distribuido y en proceso de desarrollo por especialistas del Centro de Informática Industrial (CEDIN), perteneciente a la Universidad de las Ciencias Informáticas (UCI). Entre los módulos que lo integran se encuentra el denominado Interfaz Hombre-Máquina (HMI). Este módulo no responde intuitivamente ante los posibles eventos que puedan surgir durante el proceso.

En el presente trabajo se muestra el desarrollo de un mecanismo que monitorea en todo momento los cambios de las mediciones del proceso y, ejecuta un conjunto de acciones programadas con el fin de visualizar la información correspondiente ante determinadas anomalías del proceso. Permite la especificación previa de determinadas condiciones y la configuración, a través de comandos, de las acciones que serán automatizadas. Así durante la supervisión del proceso, el propio sistema podrá mostrar los recursos implicados en el suceso y, el operador tendrá una rápida retroalimentación sobre la información del posible problema.

Palabras claves: Alarma, automatización, despliegue, evento, HMI, medición, SCADA.

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

Las nuevas exigencias de calidad y reducción de costos en el mercado global, han llevado a la informatización y optimización de los procesos en las industrias actuales. Estos son más accesibles y su seguimiento es posible mediante los sistemas de Supervisión, Control y Adquisición de Datos (SCADA, *Supervisory Control And Data Acquisition*). Dichos sistemas capturan la información de un proceso o planta industrial, la cual es utilizada para realizar una serie de análisis con los que se pueden obtener valiosos indicadores, que permiten una realimentación sobre un operador o sobre el propio proceso.

El Sistema de Automatización Industrial basado en GNU/Linux (SAINUX) es desarrollado en el Centro de Informática Industrial (CEDIN), perteneciente a la Universidad de las Ciencias Informáticas (UCI). Es un software de tipo SCADA, que integra funcionalidades de alto nivel que permiten la solución de aplicaciones de supervisión y control de procesos, utilizando para ello una arquitectura distribuida de módulos que posibilita escalar a aplicaciones de gran envergadura.

Uno de estos módulos es el Interfaz Hombre-Máquina (HMI, *Human Machine Interface*), encargado de representar en un ordenador los procesos que ocurren en campo en tiempo real; muestra los componentes implicados, los sensores, las estaciones remotas y el sistema de comunicación. Este permite dar al operador diferentes niveles de control en dependencia de sus niveles de privilegios. De esta manera el operador podrá estar en contacto directo con el sistema y, realizar la supervisión y control del proceso de forma general, mediante la observación de sus fenómenos característicos y la oportuna emisión de comandos y ajustes. La observación se hace posible por la obtención de datos adquiridos en el campo.

En este tipo de sistema, los procesos por lo general son dinámicos y, como tal, tienen eventos que lo caracterizan, dígase, cambios de estado de diversa naturaleza, comandados o accidentales. Existen los que son propios del desarrollo normal del proceso y, los que corresponden a, o señalan condiciones anormales. Estos últimos deben ser llevados al conocimiento del operador del sistema, en otras palabras, deben ser establecidas advertencias en forma de alarmas audibles y visibles.

Dada la variación constante de datos durante todo el proceso, implica que los operadores tengan que ejecutar tareas con mayor frecuencia ante los eventos ocurridos. El módulo HMI de SAINUX permite tener un control de las variables de producción y contar con información relevante de los distintos procesos en tiempo real. Ante los cambios de las mediciones que se chequean en campo, se ejecutan

animaciones de los objetos que se visualizan en los despliegues, e incluye la generación, visualización y manejo de alarmas. Posibilita además, programar la ejecución de acciones ante eventos del *mouse* sobre los objetos gráficos. Sin embargo, no es posible realizar otras acciones de forma automática que permitan el análisis y visualización del proceso durante su monitoreo.

Ante la ocurrencia de situaciones anómalas, dígase, la activación de una alarma o algún cambio fuera de rango de las variables de campo, el HMI tiene la limitante de no mostrar la información concreta, para que el operario sepa en qué área operacional está ocurriendo el problema. En una configuración real de un proceso industrial, donde se puede tener hasta más de cien representaciones gráficas en despliegues, gran cantidad de mediciones de campo y recursos del SCADA, se tiene que revisar la descripción del suceso, ver los recursos implicados y buscar dónde se brinda la información correspondiente de la anomalía del proceso, causando retrasos en la detección y corrección del posible problema. Por ejemplo, al generarse una alarma de un punto, el HMI indica la ocurrencia en el sumario de alarmas. El operario tiene que acceder a la descripción de la alarma y ver el recurso donde se generó y el tipo de alarma, abrir el sumario de puntos y buscar el estado del punto correspondiente. Incluso debe saber qué componente gráfico hace referencia al punto y en qué despliegue se está visualizando, todo esto de forma manual.

Atendiendo a la problemática planteada anteriormente, se define como **problema investigativo** a resolver: ¿Cómo automatizar acciones sobre el proceso de visualización en el HMI del SCADA SAINUX ante cambios de estado del sistema?

A partir del problema anterior se tiene como **objeto de estudio** la automatización de acciones en interfaces hombre- máquina en sistemas SCADA.

Para dar solución al mismo se establece como **objetivo general**: Desarrollar un mecanismo que monitoree el estado del sistema y ejecute acciones previamente configuradas sobre el HMI del SCADA SAINUX.

El trabajo se enmarca en el comportamiento de las interfaces hombre-máquina del SCADA SAINUX ante los cambios del proceso supervisado, representando su **campo de acción**.

Para lograr el cumplimiento del objetivo general de esta investigación, se definen las siguientes **tareas investigativas:**

- Elaboración del marco teórico de los elementos relativos al proceso de visualización durante la supervisión y control en sistemas SCADA, mediante un estudio del estado del arte sobre el tema.
- Definición de los requerimientos funcionales y no funcionales del HMI para la automatización de acciones ante los cambios de las mediciones del proceso.
- Diseñar e implementar un mecanismo para automatizar acciones en el HMI ante los cambios de las mediciones del proceso.
- Realización de pruebas con el objetivo de verificar el correcto funcionamiento de la solución propuesta.

Durante el desarrollo de la investigación, se aplicaron métodos científicos que posibilitaron la recopilación y análisis de la información necesaria para la confección de la solución propuesta.

Métodos Teóricos:

- **Análisis histórico-lógico:** Utilizado para conocer con mayor profundidad los antecedentes y las tendencias actuales de las herramientas y tecnologías.
- **Analítico-Sintético:** Utilizado para analizar los documentos, teorías y principales conceptos relacionados con los sistemas de supervisión y control, permitiendo la extracción de los elementos más importantes que hacen viable el comienzo de la investigación.
- **Modelación:** Para representar gráficamente las funcionalidades y elementos representativos de la propuesta realizada.

Métodos Empíricos:

- **Consulta de la información en todo tipo de fuentes**: Permitió la elaboración del marco teórico de la investigación.
- **Observación:** Se emplea para estudiar las características y comportamientos de las soluciones similares, permitiendo obtener información relevante sobre la automatización en las interfaces hombre-máquina de sistemas SCADA.
- **Aplicación de las pruebas de validación**: Permitió verificar la confiabilidad y validez del sistema.

El presente trabajo de diploma consta de: Resumen, Introducción, Desarrollo (dividido en tres Capítulos), Conclusiones y Recomendaciones.

Capítulo 1- Fundamentación teórica: Se analizan elementos teóricos de la investigación relacionados con el proceso de visualización durante la supervisión y control en sistemas SCADA. Conjuntamente se detallarán las herramientas y tecnologías seleccionadas para el desarrollo de la solución.

Capítulo 2- Presentación de la solución propuesta: Se describe el modelo de dominio, se especifican los requisitos funcionales y no funcionales, los actores y los casos de uso del sistema.

Capítulo 3- Implementación y pruebas: Se exponen los principales artefactos generados en el análisis, diseño y construcción de la solución, dígase: diagramas de clases, diagramas de componentes y otras características generales de la implementación. Además se realiza la comprobación funcional de la solución mediante la ejecución de pruebas.

Capítulo 1. Fundamentación teórica

Introducción

En el presente capítulo se muestran los conceptos fundamentales relacionados con los sistemas SCADA, características del SCADA SAINUX y, se hace énfasis en los elementos afines al módulo HMI. Además se describen las herramientas y tecnologías a utilizar en el desarrollo de la solución propuesta.

1.1. Sistemas SCADA

Los sistemas SCADA representan una alternativa eficiente al momento de monitorear y controlar procesos que se ejecuten en locaciones amplias, esto debido a la posibilidad de visualizar el estado total del sistema en un mismo sitio, evitando la necesidad de desplazarse por parte del operario o demás interesados al nivel de campo.

Un SCADA es un software de aplicación especialmente diseñado para funcionar sobre ordenadores de producción, lo que permite la comunicación con los dispositivos de campo (controladores autónomos y autómatas programables) y controlar el proceso de forma automática desde la pantalla del ordenador. Provee además de toda la información que se genera en el proceso productivo a diversos usuarios. Esto propicia la toma de decisiones operacionales apropiadas. De igual forma, ya que cuenta con información (alarmas, históricos, paradas) de primera mano de lo que ocurre u ocurrió en el proceso, permite la integración con otras herramientas que intervienen en el negocio. (1)

Los sistemas SCADA se utilizan en el control de oleoductos, sistemas de transmisión de energía eléctrica, yacimientos de gas y petróleo, redes de distribución de gas natural, subterráneos, generación energética (convencional y nuclear). (2)

Los primeros SCADA se caracterizaban por ser monolíticos. En la actualidad, las generaciones de este tipo de sistemas, tienden a ser distribuidos. Se entiende como un SCADA distribuido aquel sistema en que sus componentes tanto de hardware como de software, se encuentran conectados en una red de área local, cada uno de estos, con una función específica dentro del sistema. La idea a perseguir es que todo funcione como una única entidad física. Esto último se podría aproximar, si se logra una comunicación y coordinación adecuada entre los componentes. Lo anterior proporciona una guía más clara hacia el cumplimiento (lo mejor posible) del objetivo fundamental: lograr el control y supervisión de los procesos industriales.

De forma general se consideran como funciones básicas de un sistema SCADA la supervisión y control de instalaciones, procesamiento de información, generación de reportes, gestión de alarmas,

almacenamiento de información histórica, presentación de gráficos de tendencias y programación de eventos.

Los sistemas SCADA se ubican en los tres primeros niveles de la pirámide de automatización[1], donde los dos primeros niveles están compuestos por el hardware, mientras el tercer nivel lo compone la aplicación de software. El software SCADA es de vital importancia, principalmente, para labores de control de calidad y mantenimiento, pues brinda los datos necesarios para evaluar el estado del sistema. Es muy importante que el sistema SCADA se integre de forma adecuada y, cumpla con los requerimientos presentes en la pirámide. De la pirámide de automatización se puede extraer que los sistemas SCADA sirven de puente entre los niveles administrativos y, el nivel de planta en una industria. Estos sistemas brindan soporte a los niveles de producción y gestión, además sirven de integración entre las diversas áreas presentes en la industria. Estas características hacen de este tipo de sistemas una herramienta muy substancial en el contexto industrial. (3)

Los softwares de este tipo buscan primordialmente la adquisición de datos de un proceso por medio de unidades remotas conectadas a sensores, estos datos son almacenados en una base de datos ubicada en una unidad central, la cual se comunica con las unidades remotas. Los datos son presentados a los usuarios por medio de una interfaz, en donde se puede visualizar el estado del proceso, así como tomar acciones de control.

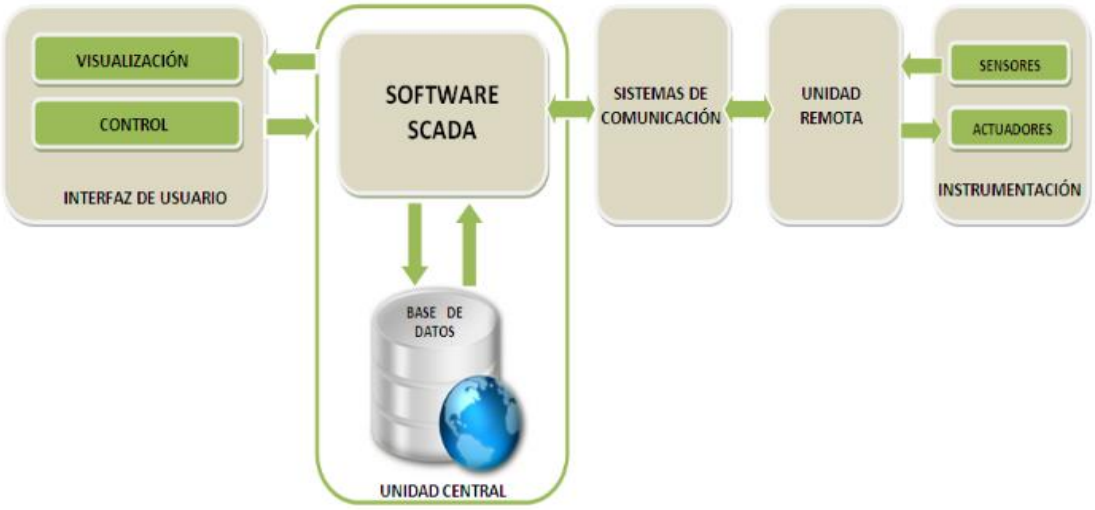


Figura 1 Esquema básico de los sistemas SCADA

[1] La pirámide de automatización CIM es un modelo que integra las diversas unidades presentes en la industria, indicando las jerarquías de las áreas decisorias así como las relaciones entre las diferentes unidades. Este modelo se utiliza como referencia al momento de automatizar un proceso, ya que brinda algunos lineamientos a seguir para que el proyecto se integre de forma adecuada a los diversos ámbitos presentes en la industria.

1.2. Interfaz Hombre-Máquina en sistemas SCADA

Los sistemas HMI pueden pensarse como una “ventana” de un proceso. Esta ventana puede estar en dispositivos especiales como paneles de operador o en una computadora. Las señales del proceso son conducidas al HMI por medio de dispositivos como tarjetas de entrada/salida, PLC (Controladores lógicos programables) y RTU (Unidades de transmisión remota). Todos estos dispositivos deben tener una comunicación que entienda el HMI. (4)

Posee ciertas funciones específicas para facilitar la visualización de procesos, que conllevan a automatizar la gestión de procesos industriales. El empleo de esta interfaz es una estrategia automatizada e integrada para optimizar la operatividad.

Funciones de un software HMI:

- **Monitoreo**: Es la habilidad de obtener y mostrar datos de la planta en tiempo real. Estos datos se pueden mostrar como números, texto o gráficos que permitan una lectura más fácil de interpretar.
- **Supervisión**: Esta función permite junto con el monitoreo, la posibilidad de ajustar las condiciones de trabajo del proceso directamente desde la computadora.
- **Alarmas**: Es la capacidad de reconocer y reportar eventos excepcionales dentro del proceso. Las alarmas son reportadas basadas en límites de control preestablecidos.
- **Control**: Es la capacidad de aplicar algoritmos que ajustan los valores del proceso y así mantener estos valores dentro de ciertos límites. Control va más allá del control de supervisión, removiendo la necesidad de la interacción humana. Sin embargo la aplicación de esta función desde un software corriendo en una PC, puede quedar limitada por la confiabilidad que quiera obtenerse del sistema.
- **Históricos**: Es la capacidad de muestrear y almacenar en archivos, datos del proceso a una determinada frecuencia. Este almacenamiento de datos es una poderosa herramienta para la optimización y corrección de procesos.

La representación del proceso es una etapa básica de la monitorización. Las facilidades gráficas que incorporan los SCADA permiten aprovechar las propiedades de animación y combinaciones de colores, para la identificación no solo de los equipos de proceso, sino también de su estado (paro/marcha) y, la animación de sus contenidos de acuerdo con la evolución de las magnitudes medidas. El resultado es una interfaz dinámica y amigable que facilita la comprensión del proceso, la visualización interna del equipamiento y, la evolución del flujo del producto.

Es importante señalar, que la automatización de la tarea de vigilancia del proceso se logra en los entornos de monitorización mediante alarmas, usadas para detectar situaciones de comportamiento anómalas.

La interpretación visual de la información es otra de las herramientas utilizadas en la industria. La representación de información y valores a través de gráficas permite un análisis global de la situación. Existe también un registro con las muestras y las fechas de cada una, que no permite que haya confusiones con ningún registro previo.

1.3. Sistema de Automatización Industrial basado en GNU/Linux

SAINUX es un sistema distribuido en módulos que trabajan de manera conjunta, posibilitando el funcionamiento del sistema como un todo. Estos módulos se encuentran interconectados a través de un software para la distribución de los servicios en la red, conocido como software de comunicación entre aplicaciones. Esta filosofía permite obtener configuraciones escalables en dependencia de los requisitos que presente cada aplicación. Los subsistemas en los que se divide son los siguientes:

Figura 2 Arquitectura SCADA SAINUX

Comunicación: Es la capa de software que se encarga de la comunicación entre los diferentes módulos que forman parte del sistema. Tiene como finalidad proporcionar la capa de comunicación de alto nivel, tanto sincrónica, como asincrónica, para la comunicación de todos los módulos que conforman el sistema SCADA.

Adquisición: Es el encargado de la adquisición, recepción, procesamiento y distribución de los datos provenientes del campo.

Configuración: El servicio de configuración está formado por un grupo de componentes con tareas específicas y, una base de datos que contendrá las configuraciones de cada uno de los módulos que conformen el proyecto activo. Es el encargado de almacenar, persistir y suministrar la información base para el funcionamiento de los demás módulos del SCADA. Además, durante la configuración 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 se selecciona el puerto de comunicación sobre el ordenador y los parámetros de la misma.

Almacenamiento de datos históricos: Es el encargado de almacenar la información del sistema para que posteriormente pueda ser empleada, por ejemplo, en generación de reportes, tendencias o en gestión de producción. La base de datos histórica (BDH) contendrá la información persistente de los datos recolectados de los dispositivos.

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

HMI: Proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos almacenados en el ordenador de proceso y, generados desde el editor incorporado en el SCADA. O sea, este subsistema lo integran dos partes fundamentales, el ambiente de configuración (*Editor*) y el ambiente de ejecución (*Runtime*).

El primero de estos es un entorno de desarrollo integrado para crear, configurar y gestionar proyectos de configuración del SCADA SAINUX. Con el editor se podrán diseñar despliegues y configurar varios procesos o partes de ellos, se definen y gestionan las variables, los drivers, los comandos y variadas opciones adicionales. Este ambiente funciona como una aplicación de diseño tradicional, con la peculiaridad que los sinópticos se confeccionan a partir de objetos y primitivas básicas predefinidas, que se pueden agrupar, combinar, transformar, importar y exportar, entre otras acciones. (5)

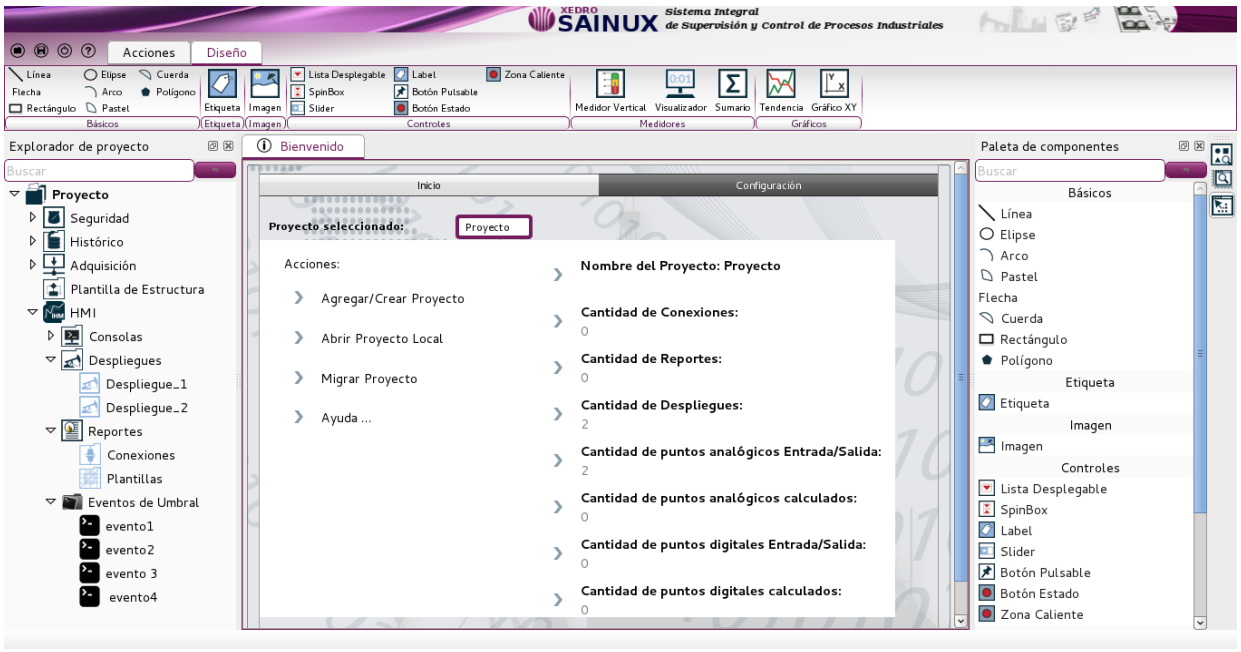


Figura 3 Ambiente de configuración del SCADA SAINUX

El ambiente de ejecución se puede comparar con un reproductor multimedia. Este se encarga de visualizar las animaciones y los objetos definidos en el editor, muestra lo que está ocurriendo en el campo en tiempo real, envía los comandos a las estaciones remotas, recibe los valores de las variables e interactúa con la mayoría de los operadores, pues se emplea para supervisar el proceso de manera directa. Al ser el módulo que se encarga de brindar el control total sobre el proceso de producción, la interfaz de usuario brinda un conjunto de funcionalidades primarias, entre ellas: la generación de reportes, impresión, análisis de variables, visualización de la tendencia de indicadores, configuración de los manejadores para la comunicación y acceso a las alarmas. (5)

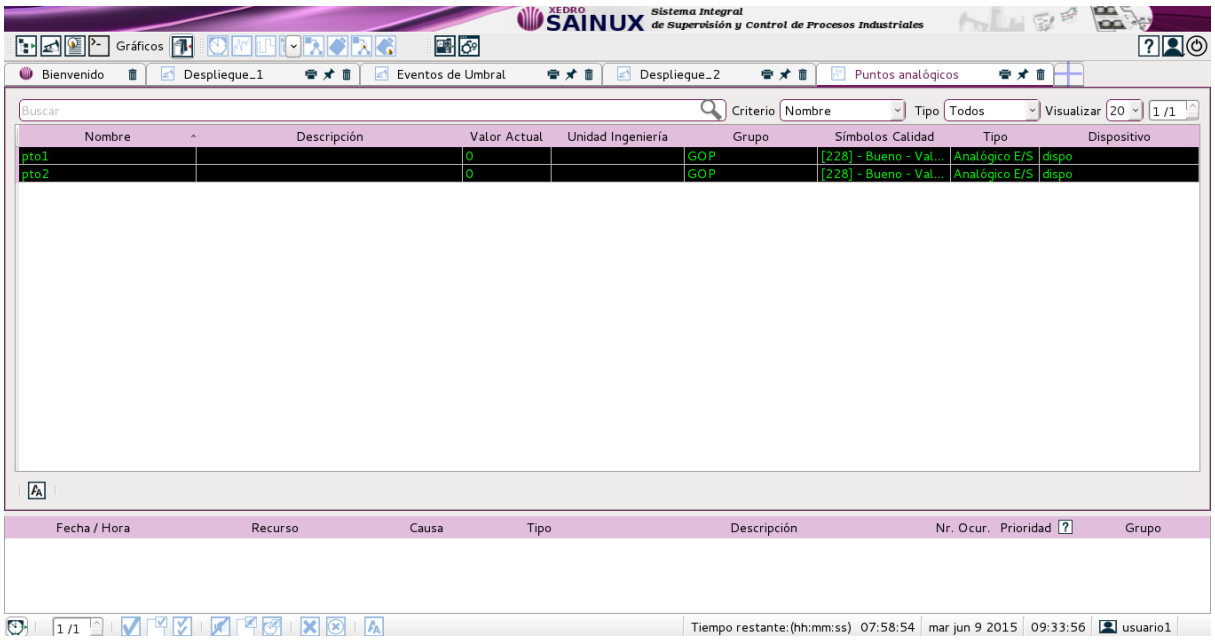


Figura 4 Ambiente de ejecución del SCADA SAINUX

1.4. Los procesos: supervisión y control

La complejidad de los procesos viene dada desde su propia naturaleza. Estos pueden contener condiciones, funciones y características muy diversas. De esta forma, diferentes instancias de un mismo SCADA estarán sujetas y, deberán responder a tal diversidad. Un sistema SCADA debe ser capaz de distinguir entre el desarrollo normal de un proceso y aquellas situaciones que requieren de la atención del operador, con el fin de que éste tome decisiones para corregir alguna situación anormal o, al menos, limitar en lo posible sus eventuales consecuencias.

Las diversidades pueden existir hasta en un mismo proceso. Esto es, debido a diferencias en las soluciones de ingeniería utilizadas en su implementación. Es necesario tratar situaciones y fenómenos de similares características también de manera distinta en lo que al control y la detección de irregularidades se refiere. Por otra parte, las exigencias de la supervisión y el control de un proceso pueden conducir a la selección de procedimientos distintos para tareas de análisis y visualización. Lo anterior, sucede aún cuando se aplican a situaciones o partes de los procesos que son iguales o muy similares. También, la misma situación en diferente contexto puede, si es anormal, tener una mayor criticidad en cuanto a sus eventuales consecuencias. (6)

Para que el sistema pueda tener utilidad para el operador, tiene que ser configurado respecto a las operaciones de análisis que se han de llevar a cabo. En este punto, hay que tomar en cuenta también las condiciones especiales del proceso y, su comportamiento deseado. Se debe proveer al operador

mecanismos, procedimientos e interfaces que le permitan atender los eventos de mayor importancia, sin excesos de información y, que garanticen llevar el sistema a lo que es considerado su operación normal.

Algunos paquetes de SCADA incorporan lenguajes de alto nivel, como *Visual Basic*, *C* o *Java,* que permiten programar tareas que respondan a eventos del sistema. Se pueden mencionar entre dichas tareas: enviar un correo electrónico al activarse una alarma concreta, un mensaje a un teléfono móvil del servicio de mantenimiento, o poner en marcha o detener partes del sistema en función de los valores de las variables adquiridas.

También se conocen los llamados scripts que permiten ejecutar comandos y operaciones lógicas basadas en criterios especificados. Estos pueden ser de varias clases: aplicación, ventana, tecla, condición o cambio de datos. En todos los casos, el scripts será leído y posteriormente ejecutado cuando se cumpla la condición previa del mismo (según la clase de script). (1)

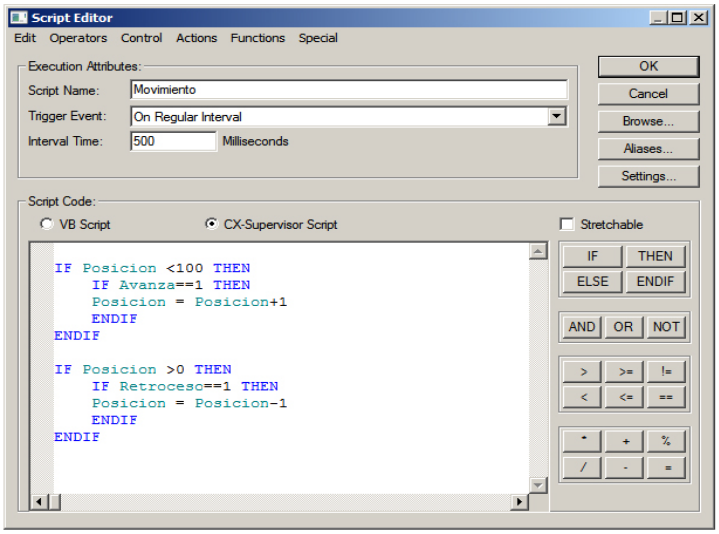


Figura 5 Ejemplo de programa script

1.5. Eventos y alarmas

En términos de informática, particularmente en el lenguaje de los sistemas, “evento” se denomina a los sucesos generados por el sistema. Según su tipo pueden ser procesados en su totalidad por el mismo sistema o a través de mecanismos que corren bajo su propio control en forma de mensajes. Por lo general, los eventos se dividen en dos grandes categorías:

- Eventos de usuario
- Eventos de sistema.

Se conoce como eventos de usuario, aquellas acciones que tienen su origen en el usuario del programa. Como ejemplos de este tipo serían acciones de pulsar un botón del ratón o una tecla. Es decir, estos tipos de eventos están siempre relacionados con alguna acción del usuario final del producto.

En el caso de eventos de sistema, son aquellos que se inician internamente o son invocados por mecanismos internos. Ejemplos de este tipo son: la generación de eventos a intervalos de tiempo predeterminados y, redibujado de objetos gráficos en ventanas. Por lo general no tienen su origen directo en ninguna acción del usuario.

Adicional a los eventos generados por usuarios y sistema, se tienen los generados por el proceso que está bajo supervisión y control. Ejemplo de este tipo serían la activación de alarmas o los cambios significativos en una característica asociada a una variable.

Los eventos indican situaciones que requieren de advertencia, manejo y/o almacenamiento para su posterior uso. Dentro de los más relevantes se encuentran las alarmas, que requieren de especial atención del operador o de los sistemas automáticos. Sin embargo existen otros eventos que se manejan con el objetivo de mejorar la calidad, producción y seguridad del sistema, así como facilitar el mantenimiento.

En resumen, un evento puede ser emitido por cualquier componente del sistema que requiera informar, en tiempo real o de forma histórica, cualquier eventualidad de importancia para la correcta operación y mantenimiento del sistema.

Dentro de los sistemas SCADA se incluyen como eventos los siguientes:

- **Comando:** Evento generado por una acción que el usuario proporciona al sistema, desde una interfaz que haya sido previamente definida para ello, por ejemplo, botones desde un despliegue o líneas de comando.
- **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.
- **Reconocimiento:** Evento generado por la acción de reconocimiento del usuario de una condición anormal originada en el proceso.
- **Inhibir alarma:** Evento generado por el usuario al ejecutar la acción operacional “Inhibir alarma” en el sistema.

- **Deshabilitar alarma:** Evento generado por el usuario al ejecutar la acción operacional "Deshabilitar alarma" en el sistema.

Las alarmas se basan en la vigilancia de los parámetros de las variables del sistema. Son los sucesos no deseables, ya que su aparición puede dar lugar a problemas de funcionamiento. Este tipo de sucesos requiere la atención de un operario para su solución antes de que se llegue a una situación crítica que detenga el proceso o, para poder seguir trabajando. El resto de situaciones, llámense normales, tales como puesta en marcha, paro, cambios de funcionamiento y consulta de datos, serán denominadas eventos del sistema o sucesos. (1)

Todo SCADA proporciona un sistema de notificación para informar al operador de las condiciones del proceso y del sistema. Este permite la visualización, registro e impresión de alarmas de proceso y eventos del sistema. Para visualizar las alarmas es preciso disponer de un visor en el cual, cuando se active la alarma, aparecerá toda la información relativa a la misma (hora y fecha, tipo de alarma, nombre, grupo y valores limites). Será necesario disponer de pulsadores de "enterado" para que el color del texto cambie indicando dicho reconocimiento de la alarma. Cuando se normalice el estado el mensaje dejará de visualizarse.

1.6. Eventos asociados a umbrales

Para adecuarnos a una terminología, se define la asociación de umbrales a comandos como "Eventos de Umbral".

Los Eventos de Umbral (*thresholds*) son la asociación de un umbral (lista de expresiones a evaluar) con una lista de comandos que se ejecutarán cuando se cumpla dicho umbral. Bajo esta definición, para que un evento de umbral se ejecute tiene, necesariamente, que tener umbral y comandos asociados, o sea, causas y acciones. (7)

La funcionalidad de adicionar eventos asociados a umbrales a nivel de proyecto, existe en algunos SCADA de las siguientes formas:

- **Desde el árbol de proyecto:** Crear un evento del tipo *Threshold*, o sea, un evento asociado a un umbral. Esto no es más que crear un umbral que se activa cuando la operación lógica sobre atributos de diferentes mediciones se convierte en verdadera; luego este umbral puede ser asociado a una lista de comandos definidos por el usuario. El conjunto de causas (Ecuaciones de Umbrales) que al cumplirse producen acciones o consecuencias (comandos) es lo que se conoce en otros SCADA como Evento.

- **Desde un componente gráfico:** De la misma forma que los eventos asociados a proyectos, los componentes gráficos pueden tener eventos asociados. Éstos pueden ser creados en el momento de la configuración del componente gráfico o usar otros eventos existentes en el proyecto.

1.7. Selección de herramientas y tecnologías

En el desarrollo de aplicaciones se incluyen varios programas de software, lenguajes de programación y tecnologías diferentes. Este proceso puede ser complicado y duradero, por lo que las herramientas disponibles pueden disminuir el tiempo de desarrollo y aumentar el desempeño de los desarrolladores y demás miembros del equipo.

1.7.1. Metodología de desarrollo

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros.

Una posible mejora es incluir en los procesos de desarrollo más actividades, más artefactos y más restricciones, basándose en los puntos débiles detectados. Sin embargo, el resultado final sería un proceso de desarrollo más complejo que puede incluso limitar la propia habilidad del equipo para llevar a cabo el proyecto. Otra aproximación es centrarse en otras dimensiones, como por ejemplo el factor humano o el producto software. Esta es la filosofía de las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad. (7)

La metodología a utilizar es AUP (*Agile Unified Process*). Se basa en una variación de la metodología Proceso Unificado Ágil en unión con el modelo CMMI-DEV v1.3; se puede decir que es una versión simplificada del Proceso Unificado de Rational (RUP, *Rational Unified Process*). Describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio, usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. AUP aplica técnicas ágiles incluyendo: (8)

- Desarrollo dirigido por pruebas.
- Modelado ágil.
- Gestión de cambios ágil.
- Refactorización de Base de Datos para mejor productividad.

La metodología AUP define 3 fases fundamentales, las cuales son expuestas a continuación:

- **Inicio**: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se hace un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y, decir si se ejecuta o no el proyecto.
- **Ejecución**: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y diseño, se implementa y se libera el producto. El producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
- **Cierre**: En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre de proyecto. (9)

Características principales de AUP:

- Iterativo e Incremental.
- Descomposición de un proyecto grande en mini-proyectos.
- Cada mini-proyecto es una iteración.
- Las iteraciones deben estar controladas.
- Cada iteración trata un conjunto de casos de uso.

Ventajas del enfoque iterativo:

- Detección temprana de riesgos.
- Administración adecuada del cambio.
- Mayor grado de reutilización.
- Mayor experiencia para el grupo de desarrollo.

1.7.2. Herramienta de modelado

El modelado es una parte central de todas las actividades que ayuda a visualizar, especificar, construir y documentar los artefactos de un sistema. El Lenguaje Unificado de Modelado (UML, *Unified Modeling*

Language) proporciona una forma estándar de escribir los planos de un sistema, procesos del negocio, funciones del sistema, clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes software reutilizables. (9)

1.7.3. Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*), en español, Ingeniería de Software Asistida por Computadora, son aplicaciones informáticas utilizadas en el proceso de desarrollo de software. Ayudan en la realización de tareas como: diseñar un proyecto, cálculo de costos, implementación de parte del código automáticamente a partir del diseño, compilación automática y documentación o detección de errores. (10)

Para el modelado será utilizado *Visual Paradigm* en su versión 8.0. Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis, diseño, construcción, pruebas y despliegue. Incluye como lenguaje de modelado UML. Permite diseñar todo tipo de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Esta herramienta se caracteriza fundamentalmente por su agilidad en el proceso de modelado, así como por ser muy flexible con una organización lógica y deducible de sus componentes.

1.7.4. Lenguaje de programación

El lenguaje de programación a utilizar para la implementación es C++ por ser el lenguaje utilizado en el desarrollo del sistema SCADA SAINUX y, así lograr una mejor integración de la solución con el producto final.

C++ está considerado por muchos como el lenguaje de programación más potente, dado que permite el trabajo tanto a alto nivel como a bajo nivel, logrando gran eficiencia en los tiempos de ejecución y bajo consumo de memoria en los programas desarrollados, aspectos que muchas aplicaciones requieren, siendo una opción factible como lenguaje a utilizar en sistemas que necesitan un alto rendimiento. Entre las principales características que brinda C++ se puede mencionar la programación orientada a objetos. La posibilidad de orientar la programación a objetos le permite al programador, diseñar aplicaciones desde un punto de vista más cercano a la vida real. Además, permite reutilizar el código de una manera más lógica y productiva. (12)

1.7.5. Framework Qt

El uso de un ambiente de trabajo (*framework*) para el desarrollo de la aplicación ayuda a construir una arquitectura sólida y robusta, brindando consistencia al código y facilitando la integración de nuevas funcionalidades. Entre las tecnologías en auge que proporcionan un juego de herramientas y

elementos gráficos para la creación de interfaces así como aplicaciones multiplataforma, se encuentra Qt. Es distribuido bajo los términos de GNU *Lesser General Public License*.

El API (*Application Programming Interface*) de la biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, uso de XML (*eXtensible Markup Language*), gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y otras herramientas para el manejo de ficheros, además de estructuras de datos tradicionales. (13) Otro de los elementos significativos de este *framework* es que permite una fácil integración con C++, para la construcción de aplicaciones orientada a objetos.

1.7.6. Entorno Integrado de Desarrollo QtCreator

Qt Creator es un IDE multiplataforma, distribuido bajo los términos de GNU *Lesser General Public License*. Combina edición, depuración, gestión de proyectos, localización y herramientas de compilación. Está diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil; posee un avanzado editor de código C++. El depurador visual para C++ es consciente de la estructura de muchas clases de Qt, lo que aumenta la capacidad de mostrar los datos con claridad. Constituye un entorno integrado para la creación y diseño de GUI para proyectos C++. Los formularios son totalmente funcionales y pueden ser previamente visualizados para asegurarse de que se verá y sentirá exactamente como lo pensó el usuario. (14)

1.7.7. eXtensible Markup Language (XML)

XML permite representar y hacer uso de las propiedades definidas en determinada estructura, posibilitando jerarquizar y estructurar la información, al describir los contenidos dentro del propio documento, así como la reutilización de partes del mismo.

El lenguaje XML consiste de una serie de reglas, pautas o convenciones para planificar formatos de texto con tales datos, de manera que produzcan archivos que sean fácilmente generados y leídos por el desarrollador, evitando los problemas más comunes como la falta de extensibilidad, de interoperabilidad entre plataformas o de soporte para universalizar su tratamiento. (15)

1.7.8. Sistema operativo

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. Por lo general, es mayoritariamente de software libre, aunque en ocasiones pueden incorporar aplicaciones o controladores propietarios. Una

gran parte de las herramientas básicas que completan el sistema operativo, vienen del Proyecto GNU (es un acrónimo recursivo que significa GNU No es Unix) de ahí el nombre: GNU/Linux. (16)

La distribución que se utiliza es Debian Wheezy en su versión 7.5, ya que es el SO que se utiliza en el centro CEDIN para el desarrollo del sistema SCADA SAINUX. Debian es una distro que ha demostrado su estabilidad y utilidad. Su distribución es libre y gratuita, tanto del SO como de las actualizaciones del mismo. Una de las grandes ventajas de Debian, es que posee miles de paquetes pre-compilados estables.

Conclusiones parciales

El análisis del proceso de supervisión y control permitió establecer las pautas para el funcionamiento del mecanismo a desarrollar, enfatizando en los elementos relacionados con la visualización en los sistemas SCADA. Los principales componentes tecnológicos para el desarrollo de la solución propuesta: metodología de desarrollo de software, lenguaje de programación, entorno de desarrollo, facilitan en conjunto alcanzar el objetivo general de la presente investigación.

Capítulo 2. Presentación de la solución propuesta

Introducción

En el presente capítulo se describen los procesos del negocio que intervienen en la realización de las actividades vinculadas al objeto de estudio. Se representan los conceptos principales del entorno del problema en un modelo de dominio. Se definen los requisitos funcionales y no funcionales. Además se describen los actores y casos de usos del sistema.

2.1. Modelo de dominio

El modelo de dominio o modelo conceptual, permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del problema. Representa las clases conceptuales del mundo real, no de componentes de software. Puede considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio. Aprovechando las oportunidades de los diagramas UML para representar conceptos, el modelo de dominio se presenta en forma de diagrama de clases donde figuran los principales conceptos y roles del sistema en cuestión. (17)

En la siguiente figura se muestra el modelo de dominio, utilizando para ello el lenguaje UML, donde cada clase representa un concepto significativo para el dominio del problema.

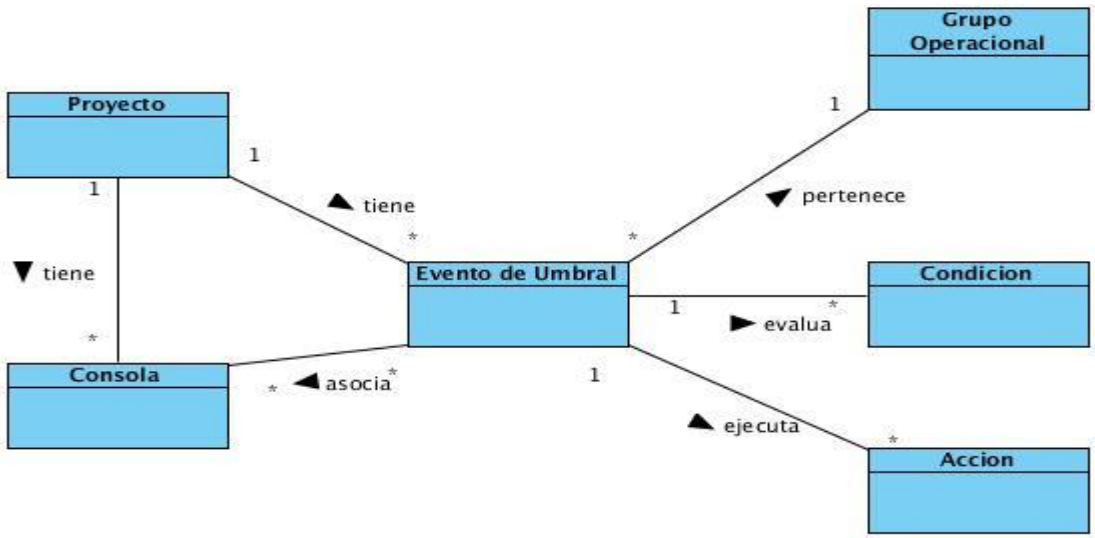


Figura 6 Modelo de dominio

Proyecto: Es lo que representa toda la información de configuración del SCADA y contiene los recursos y componentes implicados.

Evento de Umbral: Representa la configuración de un evento asociado a umbrales.

Consola: Representa la configuración de una consola HMI. Contiene los elementos necesarios para iniciar en una estación de trabajo el ambiente de ejecución del SCADA SAINUX.

Grupo Operacional: Entidad que representa un grupo operacional de privilegios para establecer los permisos que tiene un grupo de usuarios sobre los recursos configurables del SCADA.

Condición: Representa una ecuación lógica que define el umbral.

Acción: Es la que va a representar la configuración de determinada acción que podrá ser ejecutada cuando se cumpla el umbral del evento.

2.2. Requerimientos del sistema

Los requisitos o requerimientos definen el comportamiento del software. Se pueden encontrar dos tipos de requisitos: funcionales y no funcionales. Los requisitos funcionales no son más que las condiciones o capacidades que deben estar presentes en un sistema o componentes del sistema. Los requisitos no funcionales son aquellos que describen las cualidades, características y propiedades del producto. (18)

Todas las ideas que tengan los clientes, usuarios y miembros del equipo de proyecto acerca de lo que debe hacer el sistema, deben ser analizadas como candidatas a requisitos. La aprobación de requisitos es una actividad muy importante, pues un levantamiento de requisitos erróneos puede afectar los resultados esperados, además de provocar costos excesivos y pérdida de tiempo.

A continuación se enuncian los requerimientos funcionales y no funcionales que el sistema debe cumplir.

2.2.1. Requerimientos funcionales

En el ambiente de edición el sistema debe permitir configurar los eventos de umbrales a nivel de proyecto. Los requisitos correspondientes a este escenario son:

- **RF1:** Crear un evento de umbral.
- **RF2:** Eliminar un evento de umbral.
- **RF3:** Modificar las propiedades de un evento de umbral.
- **RF4:** Exportar eventos de umbral de un proyecto.
- **RF5:** Importar eventos de umbral de un proyecto.
- **RF6:** Configurar las condiciones de un evento de umbral.
- **RF7:** Configurar comandos a un evento de umbral.
- **RF8:** Asociar evento de umbral a una consola HMI.

Las funcionalidades que se deben configurar a través de comandos en los eventos de umbrales son:

- **RF9:** Terminar la visualización de un despliegue.
- **RF10:** Iniciar la visualización de un despliegue.
- **RF11:** Acoplar y desacoplar despliegue.
- **RF12:** Abrir un sumario de alarmas, despliegues, reportes, puntos analógicos, puntos digitales, eventos, dispositivos, sub-canales y estado del sistema.
- **RF13:** Ejecutar una aplicación externa al HMI.

En el ambiente de ejecución es donde se evaluarán los eventos de umbrales configurados y se ejecutarán las acciones correspondientes.

- **RF14:** Ejecutar los comandos asociados a un evento de umbral al cumplirse las condiciones del mismo.
- **RF15:** Activar y desactivar evento de umbral.
- **RF16:** Visualizar los eventos de umbrales en un sumario.

2.2.2. Requerimientos no funcionales

- **Características de apariencia o interfaz externa**
  - La interfaz debe ser de fácil comprensión en su funcionamiento, permitiendo la utilización del sistema sin mucho entrenamiento.
  - Interfaces uniformes y con los mismos colores y diseños.
  - Se debe garantizar que los colores de la interfaz de la aplicación sean claros.
  - Mensajes sin ambigüedades.
- **Características de restricciones en el diseño e implementación**
  - El sistema debe ser desarrollado en el lenguaje de programación C++.
  - Las interfaces gráficas de usuarios deben ser desarrolladas utilizando el marco de trabajo Qt.
- **Características de soporte**
  - El sistema debe ser capaz de dar las mismas salidas para diferentes ambientes de trabajo.
- **Características del hardware**
  - El sistema debe trabajar satisfactoriamente en máquinas con memoria RAM de 2 GB o superiores.
- **Características de usabilidad**
  - Facilidad de uso por parte del usuario, basta con una experiencia básica, media o avanzada en el trabajo con la aplicación.

2.3. Descripción del sistema propuesto

Para dar respuesta al problema identificado al inicio de este trabajo, y teniendo en cuenta los requerimientos planteados, la solución que se propone es el desarrollo de un mecanismo para la automatización de acciones en el HMI del SCADA SAINUX ante los cambios de las mediciones del proceso, a través de la configuración y ejecución de eventos de umbral a nivel de proyecto. Este debe permitir ejecutar automáticamente acciones previamente configuradas cuando se cumplan un conjunto de condiciones, que son evaluadas ante sucesos generados durante el monitoreo del proceso. Así se pueden configurar acciones a través de comandos para que muestren al operador la información requerida ante posibles estados del sistema.

En el ambiente de edición se gestiona la configuración de los eventos de umbral, incluyendo la definición de sus condiciones. También se establecen las acciones en forma de comandos que estarán asociadas al umbral. Se incluyen otras funcionalidades necesarias como la asociación de eventos a consolas HMI. Los eventos se adicionan como recursos configurables del SCADA, se establecen las funcionalidades para que puedan ser manipulados en los componentes del HMI que así lo requieran, como el Explorador de Proyecto y el Inspector de Propiedades.

En el ambiente de ejecución se activan o desactivan, según decida el usuario, los eventos asociados a la consola que se visualiza. El sistema chequea los eventos activos ante los cambios de las mediciones implicadas en las condiciones. Al evaluar el umbral, si las condiciones se cumplen se ejecutan los comandos asociados al evento. Se da la opción de visualizar todos los eventos de umbral en un sumario de recursos.

Las acciones que son configurables a través de comandos para ser ejecutadas al cumplirse el umbral en el ambiente de ejecución son:

- Apertura de un despliegue: Muestra los componentes gráficos que conforman un despliegue. Permite establecer el remplazo de los puntos genéricos que contenga el despliegue por puntos reales, así la visualización de un mismo despliegue puede representar diferentes escenarios. Se incluye también acoplar o desacoplar el despliegue que será visualizado.
- Cierre de un despliegue: Finaliza la visualización de un despliegue si este se encuentra abierto.
- Apertura de un sumario de recursos: Muestra en un sumario los recursos correspondientes al tipo de sumario que se establezca, dígase: despliegues, reportes, alarmas, puntos digitales, puntos analógicos, eventos, sub-canales, dispositivos y estado del sistema.
- Ejecutar una aplicación: Permite iniciar la ejecución de una aplicación externa al SCADA.

- Apertura de una gráfica de tendencia: Inicia una gráfica de tendencia con los puntos que son indicados en el ambiente de configuración. Los puntos implicados en las condiciones del umbral son establecidos previamente.
- Utilizando las facilidades que brinda UML, se representan los requisitos funcionales del sistema mediante un diagrama de casos de uso. Para ello se definen cuáles serían los actores que van a interactuar con el sistema y los casos de uso que van a representar las funcionalidades.

2.3.1. Descripción de los actores del sistema

Un actor es un rol que se juega dentro del sistema, que puede intercambiar información o puede ser un recipiente pasivo de información. Representa a un ser humano, a un software o a una máquina que interactúa con el sistema. (19)

Tabla 1 Actores del sistema

Actor	Descripción
	En el ambiente de configuración es quien configura todos los recursos del SCADA incluyendo los eventos de umbral. En el ambiente de ejecución es el encargado de supervisar todos los procesos en el SCADA SAINUX.
	En el ambiente de ejecución el propio sistema es quien monitorea los cambios que surjan, evalúa las condiciones de los eventos de umbral activos y ejecuta las acciones asociadas en caso que se cumplan las condiciones establecidas.

2.3.2. Modelo de casos de uso del sistema

El modelo de casos de uso del sistema es una representación de las funciones deseadas para el sistema y su entorno, y sirve como contrato entre el cliente y los desarrolladores. Se utiliza como entrada esencial para las actividades de análisis y diseño. (20)

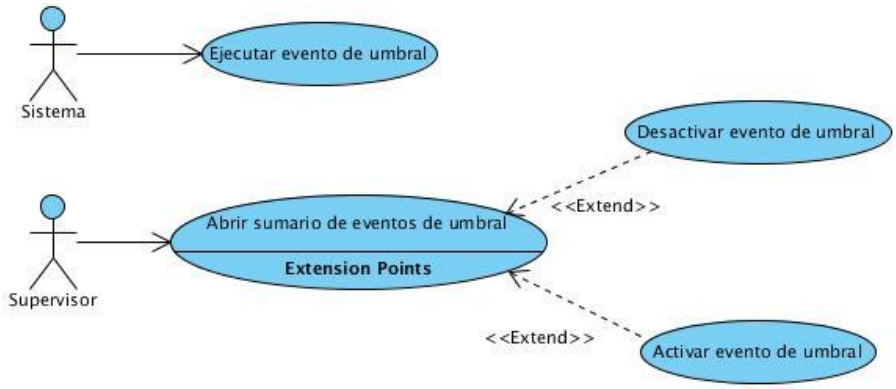


Figura 7 Diagrama de Caso de Uso ambiente de ejecución.

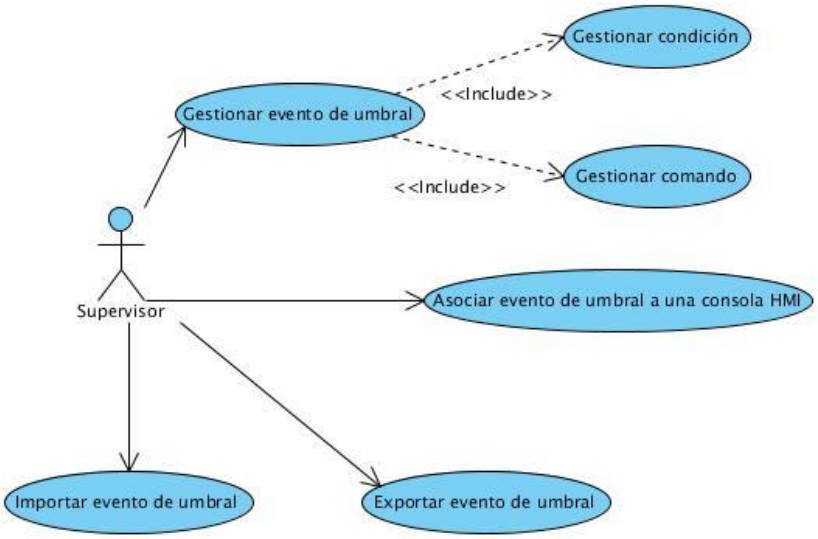


Figura 8 Diagrama de Caso de Uso ambiente de configuración.

2.3.3. Descripción textual de los casos de uso del sistema

Los casos de uso del sistema son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema. (20)

A continuación se describen detalladamente algunos de los casos de uso más importantes del sistema:

Tabla 2 CU Gestionar evento de umbral

Objetivo	El objetivo del CU es gestionar los eventos de umbral del proyecto activo.
Actores	Supervisor.

Resumen	El CU permite adicionar, eliminar y configurar las propiedades de un evento de umbral.
Referencias	RF1, RF2, RF3
Prioridad	Crítico
Precondiciones	Exista un proyecto activo con al menos un GOP.

Flujo de eventos	
Flujo Normal de eventos	
Actor	**Sistema**
1. El CU inicia cuando el supervisor da clic en el ítem “Eventos de Umbral” del árbol de proyecto.	2. Muestra en el menú la opción: Agregar evento.
	3. Termina el CU.
Sección 1: “Adicionar evento”	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona la opción “Agregar evento”.	2. Muestra una ventana solicitando la siguiente información: • Nombre • Descripción • Grupo Operacional • Condiciones • Comandos
3. Introduce los datos de • Nombre • Descripción. 4. Da clic en el botón que se establece para el GOP.	5. Muestra en una ventana el sumario de GOP del proyecto.
6. Da doble clic en un GOP del sumario.	7. Establece al evento el GOP seleccionado.
8. Da clic en el botón que se establece para las condiciones.	9. Muestra la ventana de colección de condiciones. (Ver CU2).
10. Da clic en el botón que se	11. Muestra la ventana de asociar comandos. (Ver

establece para los comandos.	CU3).
12. Da clic en el botón “Aceptar”. (Alterno 1)	13. Agrega el evento al proyecto. 14. Muestra el ítem del evento agregado en el árbol de proyecto. 15. Ir al paso 3 del flujo normal de eventos.

Flujos Alternos

Evento 1

Actor	Sistema
1. Da clic en el botón “Cancelar”.	2. Ir al paso 3 del flujo normal de eventos

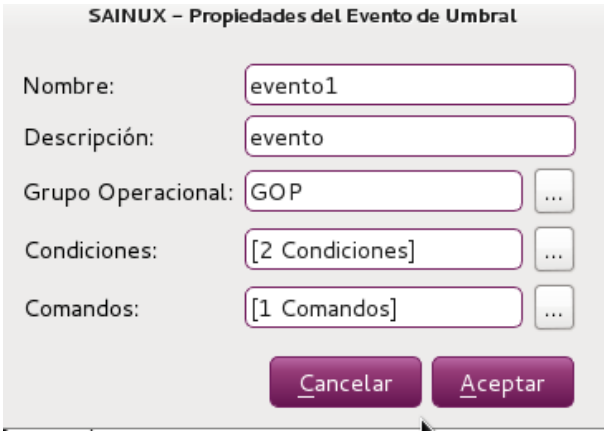


Figura 9 Prototipo de Interfaz Gestionar evento de umbral

Sección 2: “Eliminar”

Flujo de eventos

Actor	Sistema
1. Selecciona un evento del árbol de proyecto. 2. Da clic derecho.	3. Muestra un menú con las opciones • Propiedades • Eliminar
4. Selecciona la opción Eliminar.	5. Muestra en una ventana el mensaje “¿Está seguro que desea eliminar el evento de umbral?”.
6. Da clic en el botón “Aceptar”. (Alterno 1)	7. Elimina el evento del proyecto. 8. Elimina el ítem del evento del árbol de proyecto. 9. Ir al paso 3 del flujo normal de eventos.

Flujos Alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón “Cancelar”.	2. Ir al paso 3 del flujo normal de eventos

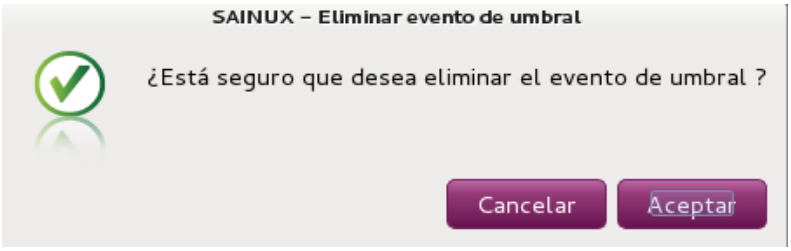


Figura 10 Prototipo de Interfaz Eliminar evento de umbral

Sección 3: “Modificar propiedades de un evento”	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona un evento del árbol de proyecto. Y se presiona clic derecho.	2. Muestra un menú con las opciones Propiedades Eliminar
3. Selecciona la opción Propiedades.	4. Muestra en una ventana la información del evento: • Nombre • Descripción • Grupo Operacional • Condiciones • Comandos
5. Configura los datos del evento. 6. Da clic en el botón “Aceptar”. (Alterno 1)	7. Actualiza las propiedades del evento. 8. Ir al paso 3 del flujo normal de eventos.
Flujos Alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón “Cancelar”.	2. Ir al paso 3 del flujo normal de eventos

Relaciones	**Caso de uso del cual depende**	

	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		

Tabla 3 CU Exportar evento de umbral

Objetivo	El objetivo del CU es exportar los eventos de umbral de un proyecto.
Actores	Supervisor
Resumen	El CU permite exportar a un fichero en disco los eventos de un proyecto.
Referencias	RF4
Prioridad	Secundario
Precondiciones	Exista un proyecto activo.
Flujo de eventos	
Flujo Normal de eventos	
Actor	**Sistema**
1. El CU inicia cuando el supervisor da clic derecho en el ítem “Eventos de Umbral” en el árbol de proyecto.	2. Muestra un menú con las opciones: Agregar evento Importar Exportar Eliminar todo
3. Selecciona la opción “Exportar”.	4. Muestra la ventana Exportar Eventos de Umbral. (alterno 1) 5. Muestra los eventos de umbral del proyecto en un sumario.
6. Selecciona los eventos que se quieren exportar. 7. Da clic en el botón Siguiente.	8. Solicita el nombre y ruta del archivo donde serán guardados los eventos.
9. Selecciona la ruta y escribe el nombre de archivo. 10. Da clic en el botón Siguiente.	11. Muestra los eventos seleccionados, la ruta y fichero establecido.
12. Da clic en el botón Exportar.	13. Guarda los eventos seleccionados en el fichero. 14. Termina el CU.
Flujos Alternos	

Evento 1	
Actor	**Sistema**
	1. Si no existen eventos de umbral en el proyecto, se muestra el mensaje: “No existen recursos a exportar.”
2. Da clic en el botón Aceptar.	3. Ir al paso 14 del flujo normal de eventos.

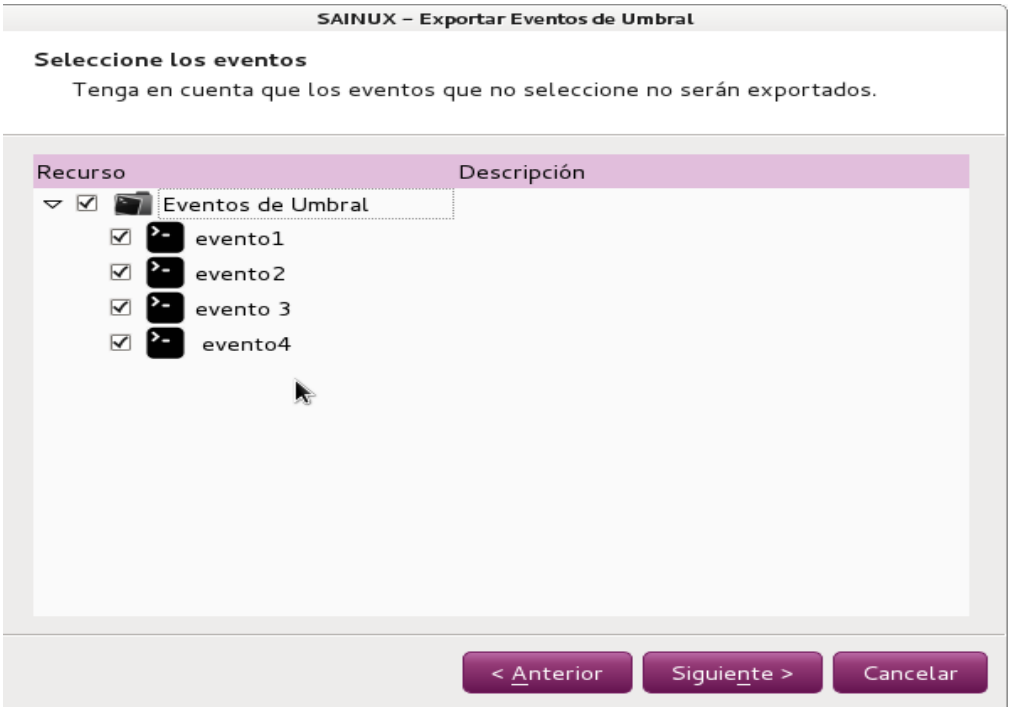


Figura 11 Exportar eventos de umbral

Relaciones	**Caso de uso del cual depende(padre)**	
	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		

Tabla 4 CU Asociar evento de umbral a una consola HMI

Objetivo	El objetivo del CU es asociar los eventos de umbral de un proyecto a una consola HMI.
Actores	Supervisor.

<table>
<tr><td>Resumen</td><td colspan="2">El CU permite asociar los eventos de un proyecto a una consola HMI.</td></tr>
<tr><td>Referencias</td><td colspan="2">RF8</td></tr>
<tr><td>Prioridad</td><td colspan="2">Critico</td></tr>
<tr><td>Precondiciones</td><td colspan="2">Exista un proyecto activo.</td></tr>
<tr><td colspan="3">Flujo de eventos</td></tr>
<tr><td colspan="3">Flujo Normal de eventos</td></tr>
<tr><td>Actor</td><td colspan="2">Sistema</td></tr>
<tr><td>1. El CU inicia cuando el supervisor selecciona la pestaña “Eventos de Umbral” en la ventana de propiedades de HMI.</td><td colspan="2">2. Muestra los eventos de umbral disponibles del proyecto y los que están asociados a la consola HMI.</td></tr>
<tr><td>3. Selecciona los eventos a asociar de la lista de eventos disponibles.
4. Da clic en el botón de asociación.</td><td colspan="2">5. Elimina de la lista de eventos disponibles los seleccionados y los agrega a la lista de eventos asociados.</td></tr>
<tr><td>6. Da clic en el botón Aceptar. (Alterno 1)</td><td colspan="2">7. Asocia a la consola HMI los eventos de umbral que fueron seleccionados.
8. Termina el CU.</td></tr>
<tr><td colspan="3">Flujos Alternos</td></tr>
<tr><td colspan="3">Evento 1</td></tr>
<tr><td>Actor</td><td colspan="2">Sistema</td></tr>
<tr><td>1. Da clic en el botón Cancelar.</td><td colspan="2">2. Ir al paso 8 del flujo normal de eventos.</td></tr>
</table>

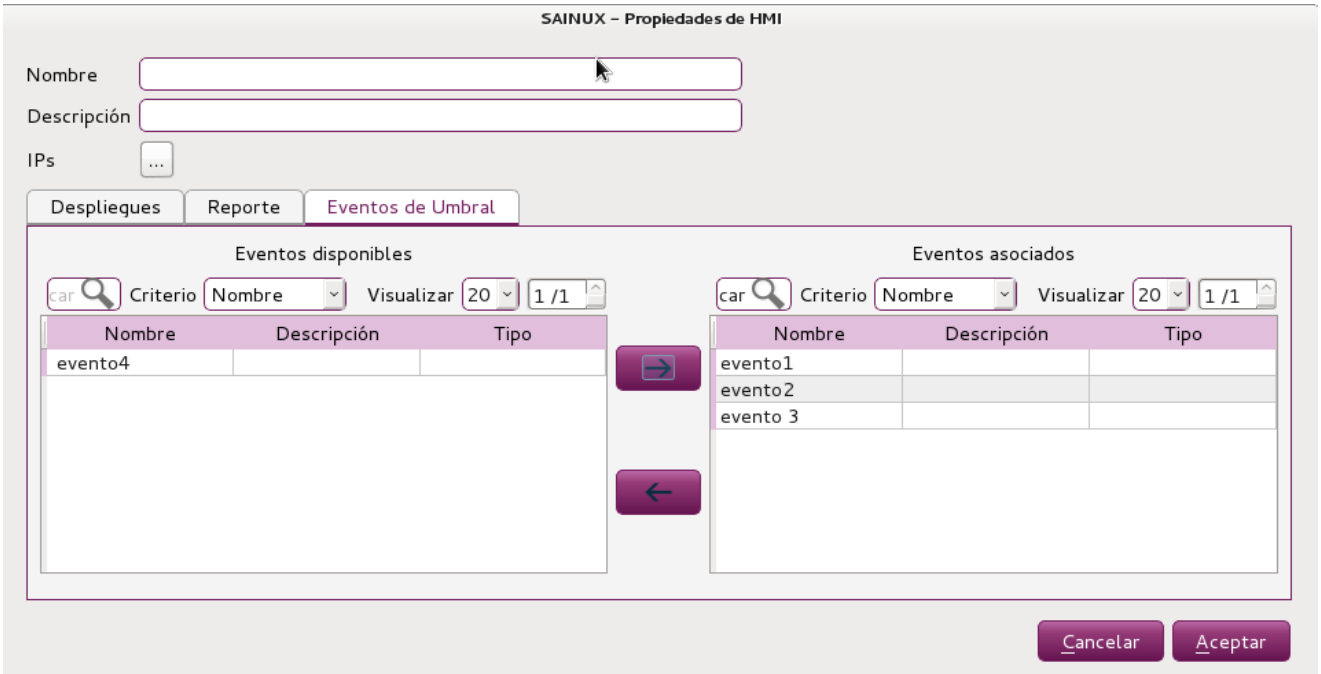


Figura 12 Asociar evento de umbral a una consola HMI

Relaciones	**Caso de uso del cual depende(padre)**	
	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		

Conclusiones parciales

Con la realización de un estudio detallado del negocio, el modelamiento del mismo, la definición de los requisitos funcionales y no funcionales que el sistema debe cumplir, la descripción y representación gráfica de los casos de uso del sistema se contribuyó a: alcanzar una mayor calidad del producto final y un alto grado de satisfacción del cliente, constituir las bases para enmarcar el análisis y diseño de la solución y establecer una comprensión entre desarrolladores y clientes para una mayor satisfacción y cumplimiento de sus necesidades.

Capítulo 3. Implementación y Pruebas

Introducción

En este capítulo se expondrá el proceso de construcción del mecanismo propuesto anteriormente. Para ello se definirá la arquitectura, se construirán los modelos de diseño de clases, se seleccionarán los patrones de diseño a emplear y, se aplicaran las pruebas pertinentes para validar el funcionamiento de la solución.

3.1. Arquitectura

La arquitectura de software constituye un pilar fundamental en la construcción de cualquier solución informática, dos acercamientos a este tema lo constituyen las siguientes definiciones:

- “La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos, el ambiente, los principios que orientan su diseño y evolución.” (21)
- “La Arquitectura de Software de un sistema de programa o computación, es la estructura de las estructuras del sistema, la cual comprende los componentes del software, las propiedades de esos componentes visibles externamente, y las relaciones entre ellos.” (22)

Se define la arquitectura como una vista estructural de alto nivel, que ocurre muy tempranamente en el ciclo de vida del software y, define el estilo adecuado para cumplir con los requerimientos no funcionales del sistema. Además, en el proceso de desarrollo y construcción del software juega un papel fundamental. Posibilita al mismo tiempo, el análisis de las relaciones e interacciones entre los distintos elementos que conforman un sistema, antes de elaborar el diseño definitivo y comenzar a implementar.

Debido a las características del módulo HMI, se utilizan en este varios estilos arquitectónicos con el propósito de aprovechar las ventajas que brinda cada uno. Los estilos arquitectónicos a tener en cuenta son: orientada a componentes y Modelo-Vista-Controlador (MVC, *Model-View-Controller*).

Los sistemas de software orientados a componentes se basan en principios definidos por una ingeniería de software específica. En un principio, se planteaba como una modalidad que extendía o superaba la tecnología de objetos. Las arquitecturas orientadas a componentes están teniendo éxito en los sistemas orientados a extensiones. En la mayoría de los casos los componentes terminan siendo formas especiales de bibliotecas que admiten acoplamientos en caliente con el sistema, necesitan ser registrados y no requieren que sea expuesto el código fuente de la clase. De las varias

definiciones de componentes que existen, Clemens Alden Szyperski proporciona una que es bastante operativa. Plantea que: “un componente de software es una unidad de composición con interfaces especificadas contractualmente y dependencias del contexto explícitas” (23)

El estilo de arquitectura basado en componentes presenta las siguientes características:

- Es un estilo de diseño para aplicaciones compuestas de componentes individuales.
- Pone énfasis en la descomposición del sistema en componentes lógicos o funcionales que tienen interfaces bien definidas.
- Define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades. (24)

En la siguiente figura se muestran algunos componentes del HMI en los cuales se especificaron y desarrollaron funcionalidades para alcanzar el objetivo de la solución.

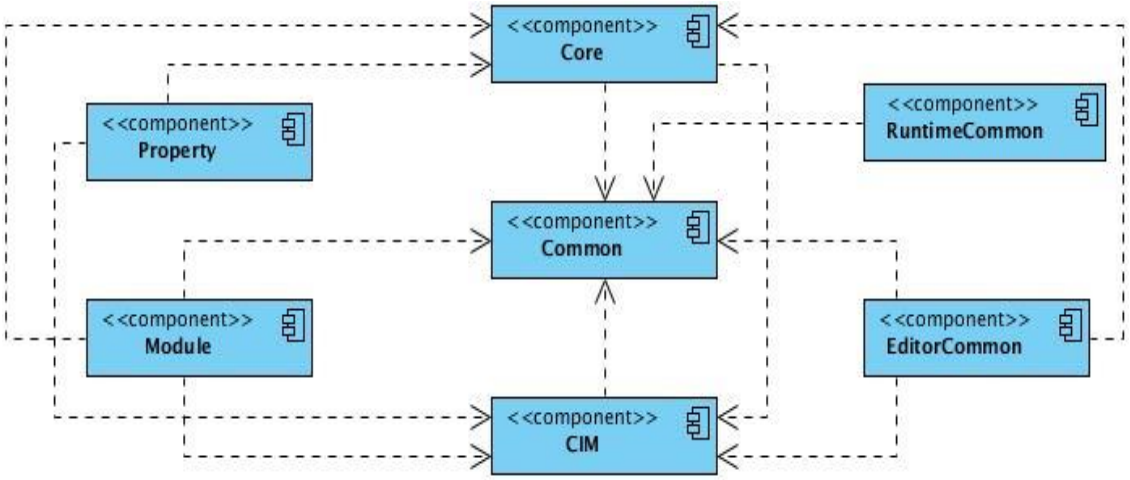


Figura 13 Componentes en el HMI

El patrón conocido como MVC separa el modelado del dominio, la presentación y, las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

- **Modelo:** El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- **Vista:** Maneja la visualización de la información.
- **Controlador**: Interpreta las acciones, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual.

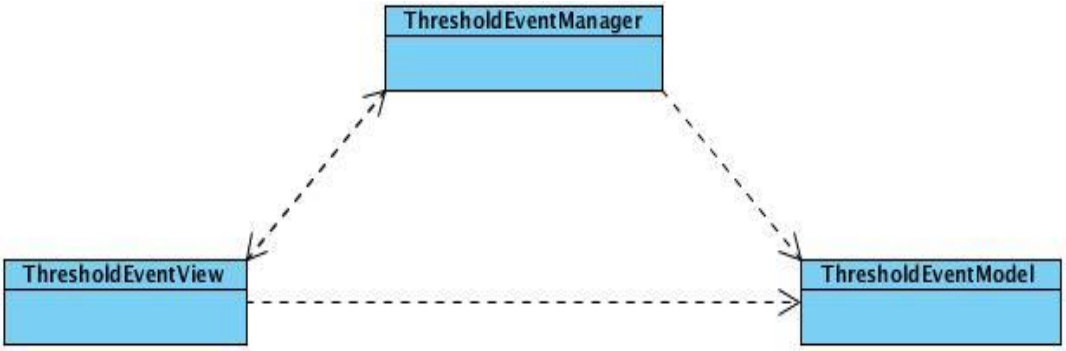


Figura 14 Uso de Modelo-Vista-Controlador

La arquitectura Modelo-Vista es una variación del patrón de diseño MVC que propone Qt, procedente de Smalltalk[2], que a menudo se utiliza en la construcción de interfaces de usuario. El modelo se comunica con la fuente de datos, proporcionando una interfaz para los otros componentes de la arquitectura. La naturaleza de la comunicación depende del tipo de fuente de datos y de la manera de aplicar el modelo. La vista obtiene los índices del modelo desde el propio modelo, que es donde se encuentran las referencias a los elementos de los datos. Mediante estos índices la vista puede recuperar los datos y manipularlos. (25)

En las vistas estándar de Qt, un delegado conoce los elementos de los datos. De esta manera, cuando un elemento es editado, el delegado se comunica con el modelo directamente a través de sus índices.

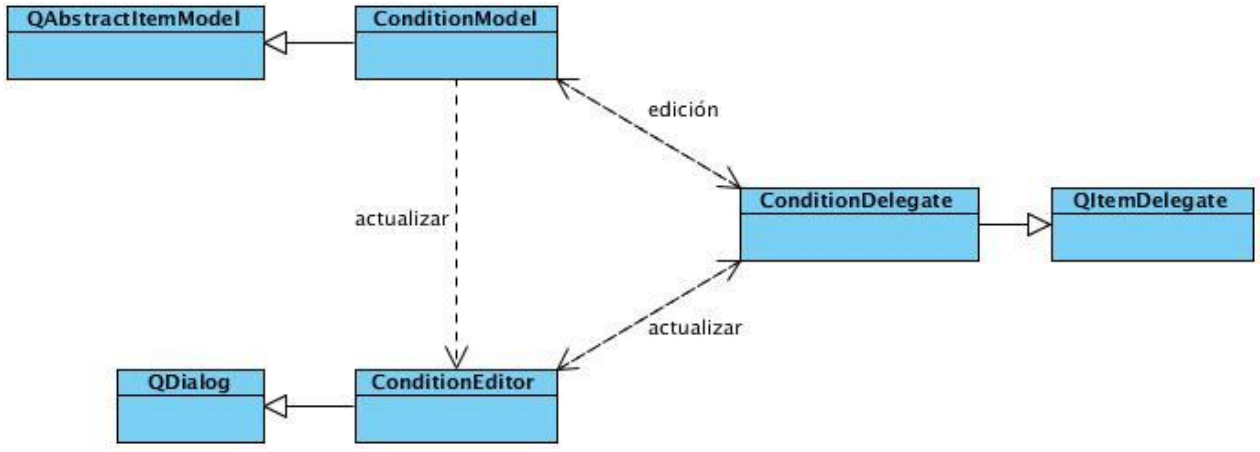


Figura 15 Uso de Modelo-Vista de Qt

[2] Lenguaje de programación que permite realizar tareas mediante la interacción con un entorno de objetos virtuales. Metafóricamente, se puede considerar que un Smalltalk es un mundo virtual donde viven objetos que se comunican mediante el envío de mensajes.

3.2. Patrones de diseño

Los patrones de diseño de software son los que nos permiten describir fragmentos de diseño y reutilizar ideas de diseño, ayudando a beneficiarse de la experiencia de otros. Los patrones dan nombre y forma a heurísticas abstractas, reglas y buenas prácticas de técnicas orientadas a objetos. Además el uso de patrones proporciona una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos. Así la incorporación de un nuevo programador, no requerirá conocimiento de lo realizado anteriormente por otro. (25)

A continuación se describen un conjunto de patrones utilizados en el diseño e implementación de la solución propuesta.

3.2.1. Patrones GRASP

Los patrones GRASP (*General Responsibility Assignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. (26)

Experto: El propósito de este patrón es asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. La responsabilidad de la creación de un objeto o la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). (25)

Creador: Crear una nueva instancia por la clase que tiene la información necesaria para realizar la creación del objeto. Usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase, o contiene o agrega la clase.

Controlador: Asignar la responsabilidad de gestionar un mensaje de un evento del sistema a una clase que represente una de estas dos opciones: representa el sistema global, dispositivo o subsistema (controlador de fachada), o representa un escenario de caso de uso en el que tiene lugar el evento del sistema (controlador de caso de uso o de sesión). (25)

3.2.2. Patrones Gof

Los patrones de diseño GoF (*Gang of Four*), término que hace referencia a los cuatro autores del libro "*Design Patterns: Elements of Reusable Object-Oriented Software*": Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides, constituyen una serie de posibles soluciones a problemas que suelen ser comunes en el desarrollo de software. Se clasifican en tres categorías basadas en su propósito:

- **Comportamiento:** Los patrones de comportamiento están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.
- **Estructurales:** Los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.
- **Creacionales:** Los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.

Observador: La idea principal detrás del patrón de comportamiento *Observer* es que existe una entidad con estados cambiantes y una o más entidades observándola. Los observadores esperan a que la entidad observada les informe de un cambio de estado a través de un evento que puede ser de su interés, por lo que los observadores se registran con la entidad observada. (9) El patrón Observador se basa en el polimorfismo, y protege al emisor del conocimiento de la clase específica de objetos, y número de objetos, con los que se comunica cuando genera un evento. Es usado para suscribir los eventos (*ThresholdEvent*) a las mediciones (*Measurement*) que definen las ecuaciones del umbral.

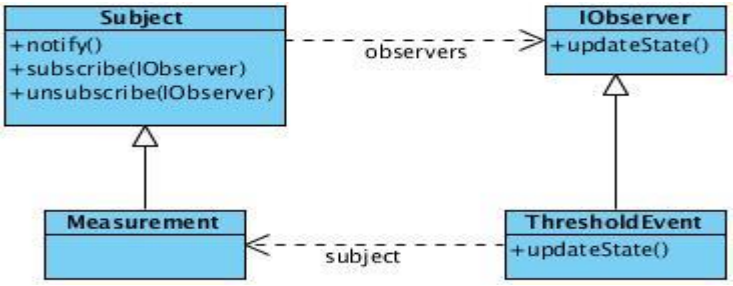


Figura 16 Aplicación del patrón observador

Fábrica abstracta: Este patrón tiene que ver con el proceso de creación. Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas. Abstrae la implementación de las clases fábrica de familia de objetos. Es usado en el inspector de propiedades para implementar la creación del editor que permite el acceso a los datos de: colecciones de condiciones (*ConditionCollection*), comandos (*CommandEventCollection*) y mediciones (*MeasurementEventCollection*).

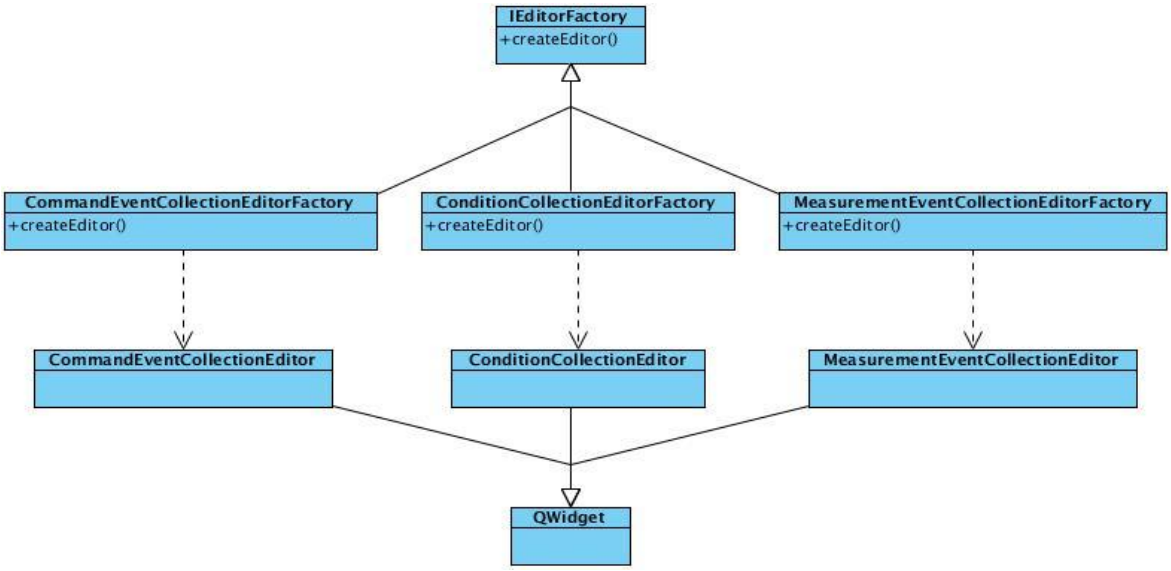


Figura 17 Aplicación del patrón fábrica abstracta

Comando: Permite encapsular en un objeto la acción que satisface una petición. Promueve una separación entre dicha acción y la petición que resuelve, lo cual redunda en una mayor flexibilidad en todo lo relativo a la acción. Es usado en la implementación de funcionalidades para gestionar los eventos de umbral, ejemplo: adicionar evento (*AddThresholdEventCommand*), eliminar evento (*DeleteThresholdEventCommand*), exportar eventos (*ExportThresholdEventsCommand*).

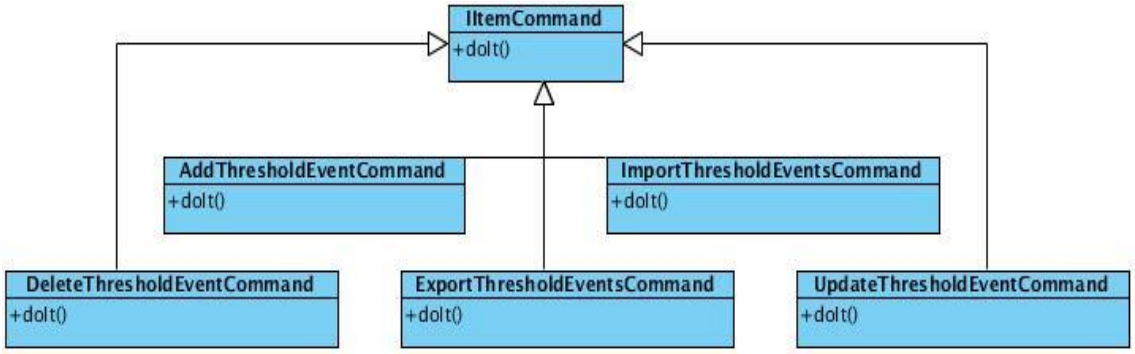


Figura 18 Aplicación del patrón comando

3.3. Modelo de diseño

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida. En el diseño se modela el sistema y se le da forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos.

3.3.1. Diagrama de clases del diseño

El diagrama de clases del diseño (DCD) describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. A diferencia de las clases conceptuales del modelo del dominio, las clases de diseño muestran las definiciones de las clases de software en lugar de los conceptos del mundo real. (20)

Para una mejor organización los diagramas de clases del diseño se han estructurado por componentes del HMI. A continuación se muestran los principales resultados obtenidos.

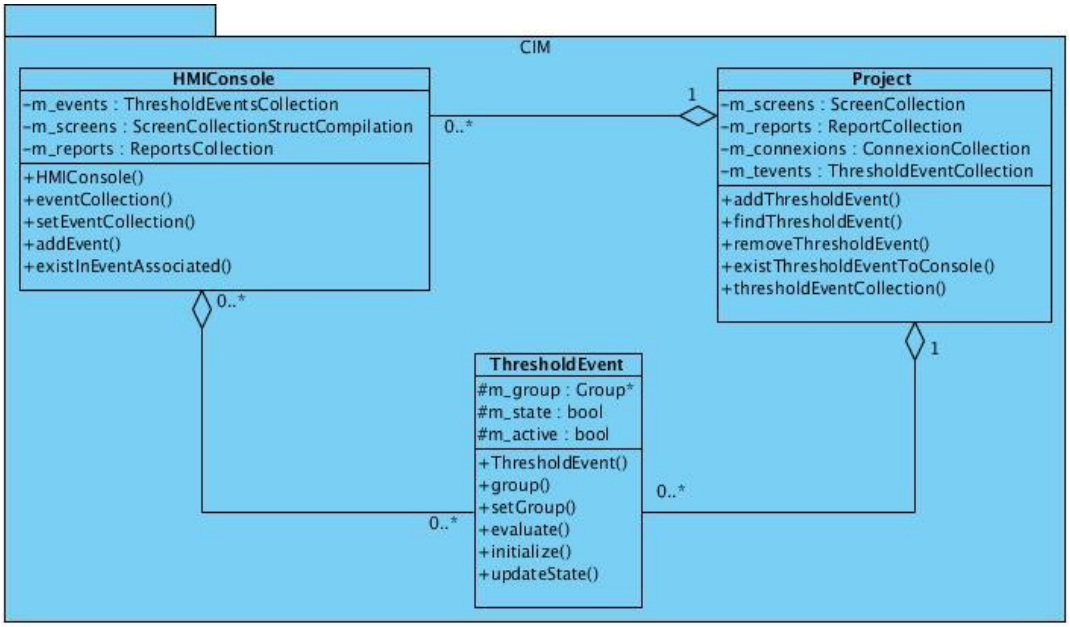


Figura 19 Diagrama de Clases de Diseño CIM

El paquete CIM (*Common Information Model*) contiene las clases del modelo de datos que definen las entidades del SCADA.

Project: Clase que representa una configuración del SCADA. Contiene todos los recursos implicados como: despliegues, reportes, conexiones, consolas HMI. Esta entidad contendrá también los eventos de umbral.

HMIConsole: Representa las propiedades de una consola HMI. Contiene los despliegues y reportes que les son asociados. Los eventos de umbral también serán asociados a consolas HMI para que en modo *Runtime* cada estación de trabajo solo monitorice aquellos eventos implicados con sus recursos.

ThresholdEvent: Clase que define el recurso evento de umbral.

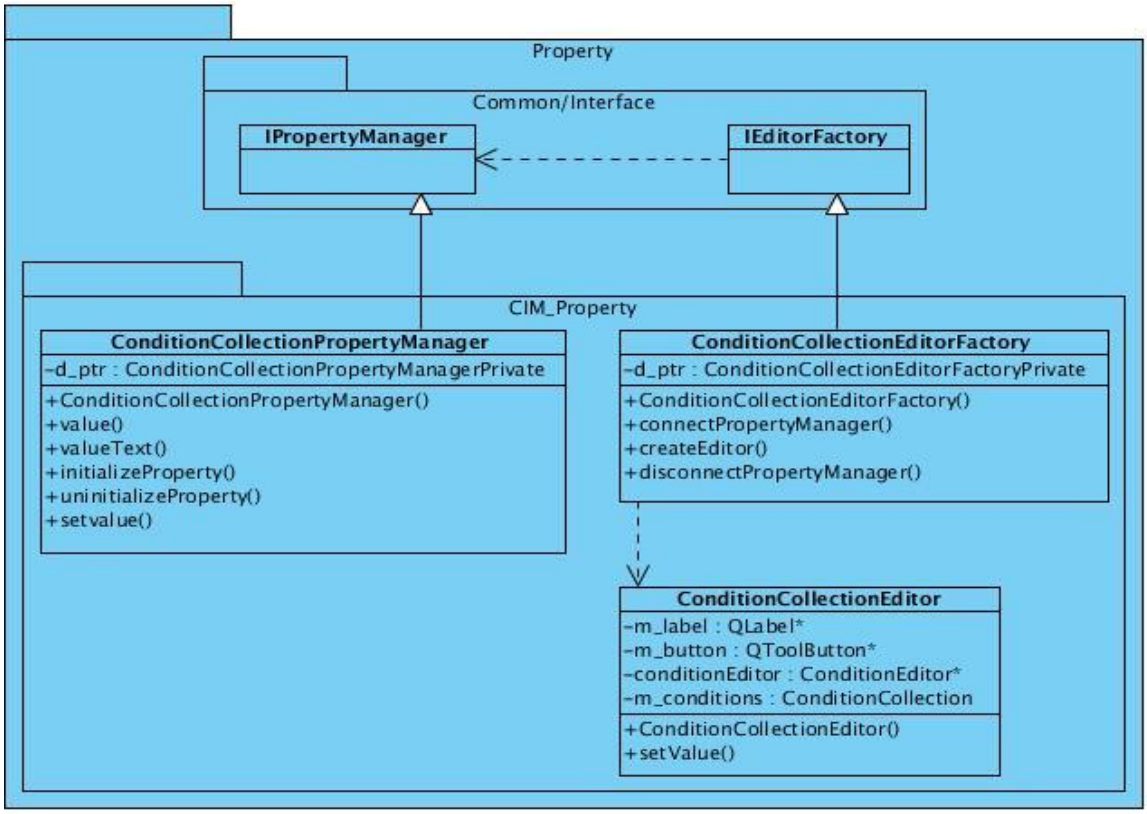


Figura 20 Diagrama de Clases de Diseño Property

El paquete Property contiene las clases que permiten el manejo de los diferentes tipos de datos en el Inspector de Propiedades del HMI. El diagrama anterior muestra un ejemplo de las entidades principales para lograr este propósito con el tipo de datos que representa una colección de condiciones. En el desarrollo de la solución fue necesario implementar este mecanismo para los datos de colección de condiciones, colección de comandos y colección de mediciones. Esto permite poder configurar los eventos de umbrales a través del Inspector de Propiedades.

ConditionCollectionEditor: Clase que muestra la colección de condiciones y permite el acceso a ellas.

ConditionCollectionEditorFactory: Como resultado del patrón fábrica abstracta esta clase es quien construye la entidad ConditionCollectionEditor.

ConditionCollectionPropertyManager: Clase que gestiona el valor de la propiedad de tipo colección de condiciones.

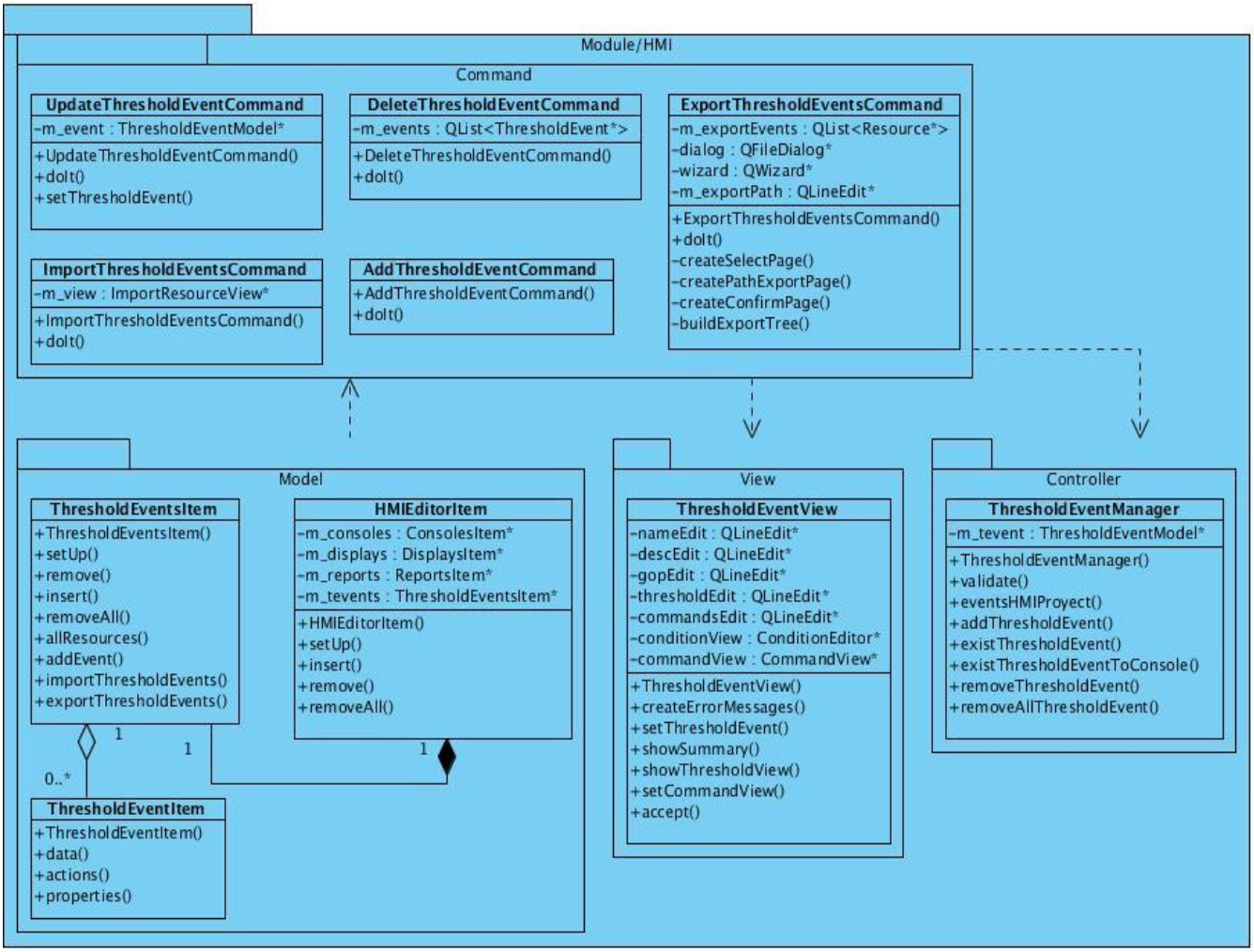


Figura 21 Diagrama de Clases de Diseño Module HMI

El paquete Module HMI contiene las entidades que corresponden al módulo HMI.

ThresholdEventItem: Clase que constituye un ítem en el árbol del Explorador de Proyecto para representar un evento de umbral.

ThresholdEventsItem: Clase que constituye un ítem en el árbol del Explorador de Proyecto para representar un nodo contenedor de todos los eventos de umbral del proyecto. Es decir, contiene todos los ThresholdEventItem.

HMIEditorItem: Clase que constituye un ítem en el árbol del Explorador de Proyecto para representar el nodo correspondiente al módulo HMI. Contiene todos los nodos que representen sus recursos.

ThresholdEventView: Clase que representa la ventana de edición de un evento de umbral.

ThresholdEventManager: Clase que define un manejador de los datos de eventos de umbral.

AddThresholdEventCommand: Clase que define la acción de adicionar un evento de umbral al proyecto.

DeleteThresholdEventCommand: Clase que define la acción de eliminar un evento de umbral del proyecto.

ExportThresholdEventsCommand: Clase que define la acción de exportar la configuración de los eventos de umbral del proyecto a un fichero externo.

ImportThresholdEventsCommand: Clase que define la acción de importar la configuración de eventos de umbral al proyecto desde un fichero externo.

UpdateThresholdEventCommand: Clase que define la acción de actualizar la configuración de un evento de umbral.

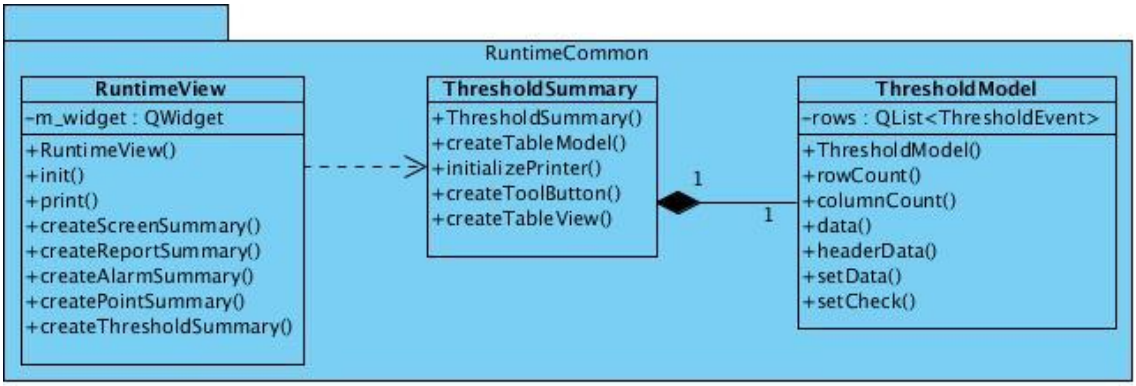


Figura 22 Diagrama de Clases de Diseño RuntimeCommon

El paquete RuntimeCommon contiene las clases necesarias en el ambiente de ejecución o Runtime.

RuntimeView: Clase que representa un componente desplazable para todas las vistas a mostrar en modo runtime.

ThresholdSummary: Clase que define el sumario de eventos de umbral.

ThresholdModel: Clase que define el modelo para el sumario de eventos de umbral.

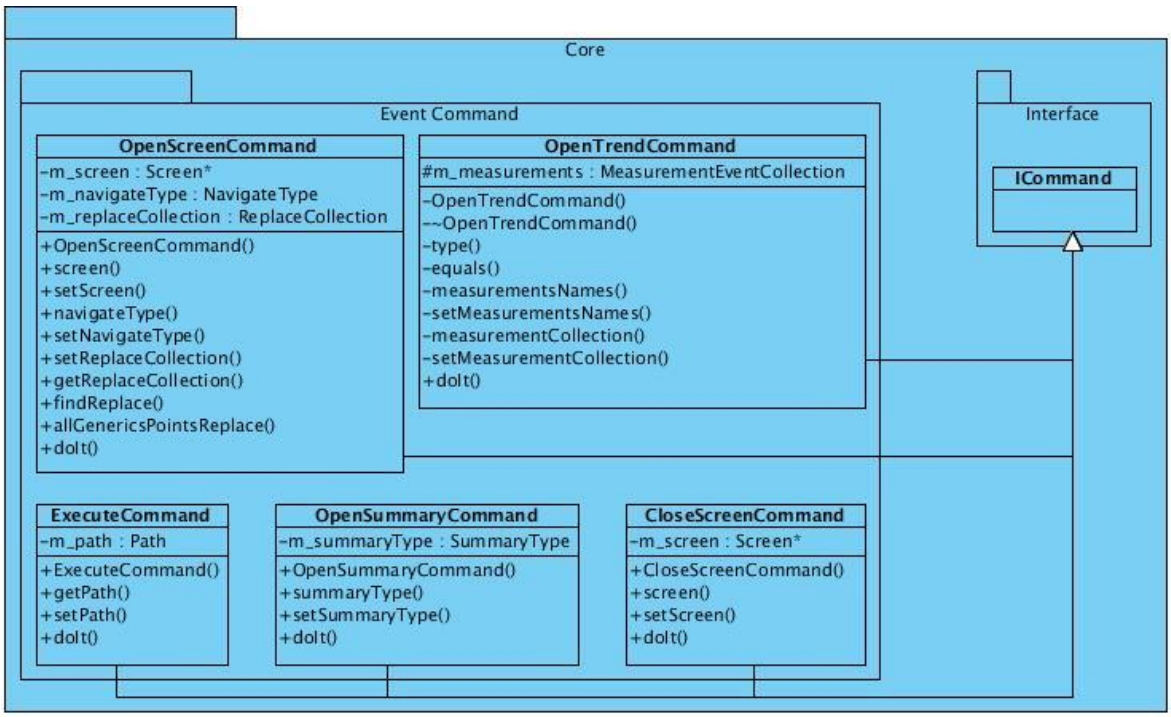


Figura 23 Diagrama de Clases de Diseño Core 1

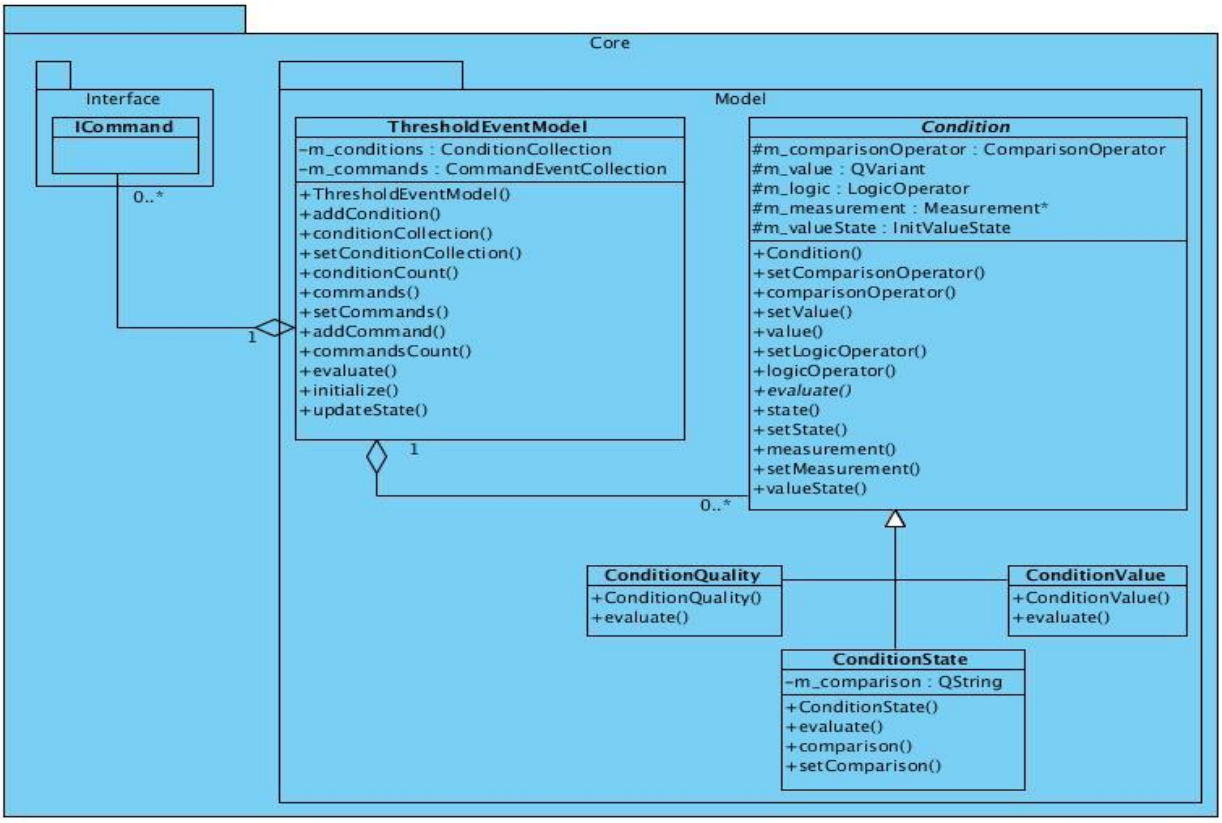


Figura 24 Diagrama de Clases de Diseño Core 2

El paquete Core representa el núcleo del sistema. Contiene las clases que brindan funcionalidades básicas para el HMI.

CloseScreenCommand: Clase que define los elementos configurables para la acción de cerrar un despliegue.

ExecuteCommand: Clase que define los elementos configurables para la acción de ejecutar una aplicación externa al SCADA.

OpenScreenCommand: Clase que define los elementos configurables para la acción de abrir un despliegue.

OpenSummaryCommand: Clase que define los elementos configurables para la acción de abrir un sumario de recursos.

OpenTrendCommand: Clase que define los elementos configurables para la acción de abrir una tendencia.

ThresholdEventModel: Clase que define el modelo de un evento de umbral. Contiene las condiciones que definen el umbral y la colección de comandos asociados que serán ejecutados.

3.4. Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. Este artefacto describe cómo se implementan los componentes, congregándolos en subsistemas organizados en capas y jerarquías, y señala las dependencias entre éstos. (17)

Para conformar el modelo de implementación se realizaron varios diagramas fundamentales, los diagramas de componentes y el diagrama de despliegue, los cuales se describen a continuación.

3.4.1. Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus elementos. Los componentes representados pueden ser datos, archivos, ejecutables, código fuente y directorios. (27)

A continuación se muestran los diagramas de componentes obtenidos en la implementación de la solución propuesta.

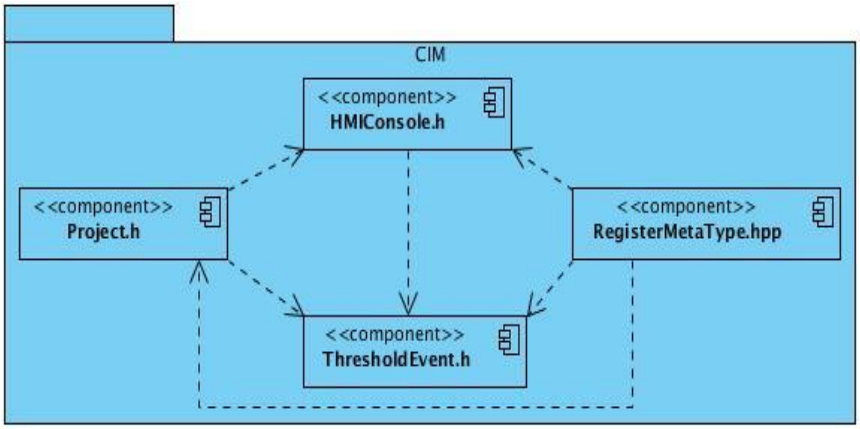


Figura 25 Diagrama de componente CIM

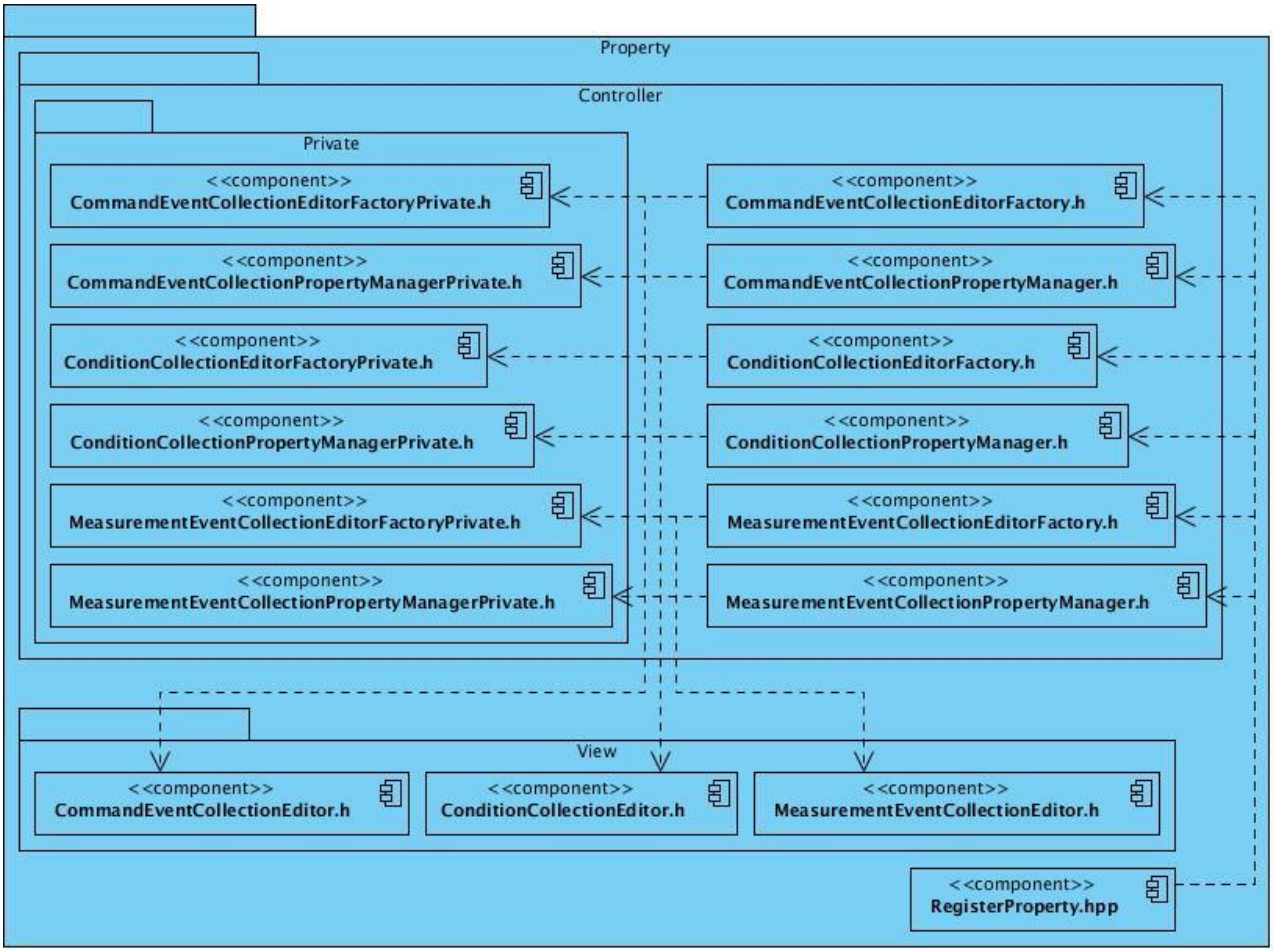


Figura 26 Diagrama de componente Property

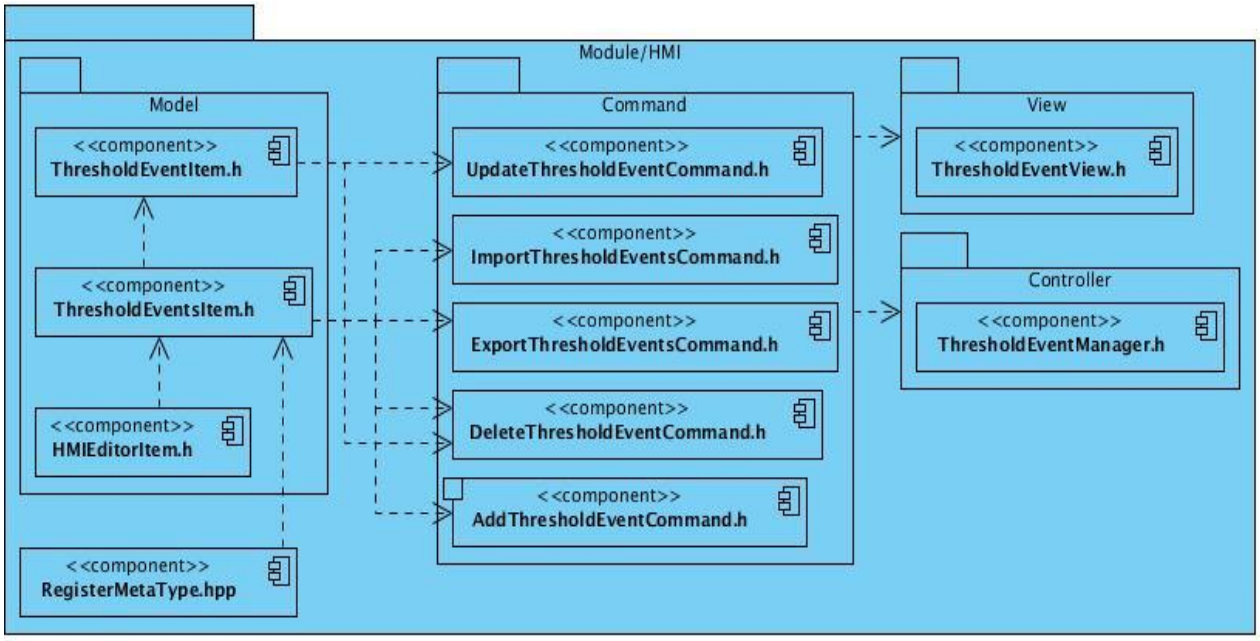


Figura 27 Diagrama de componente Module HMI

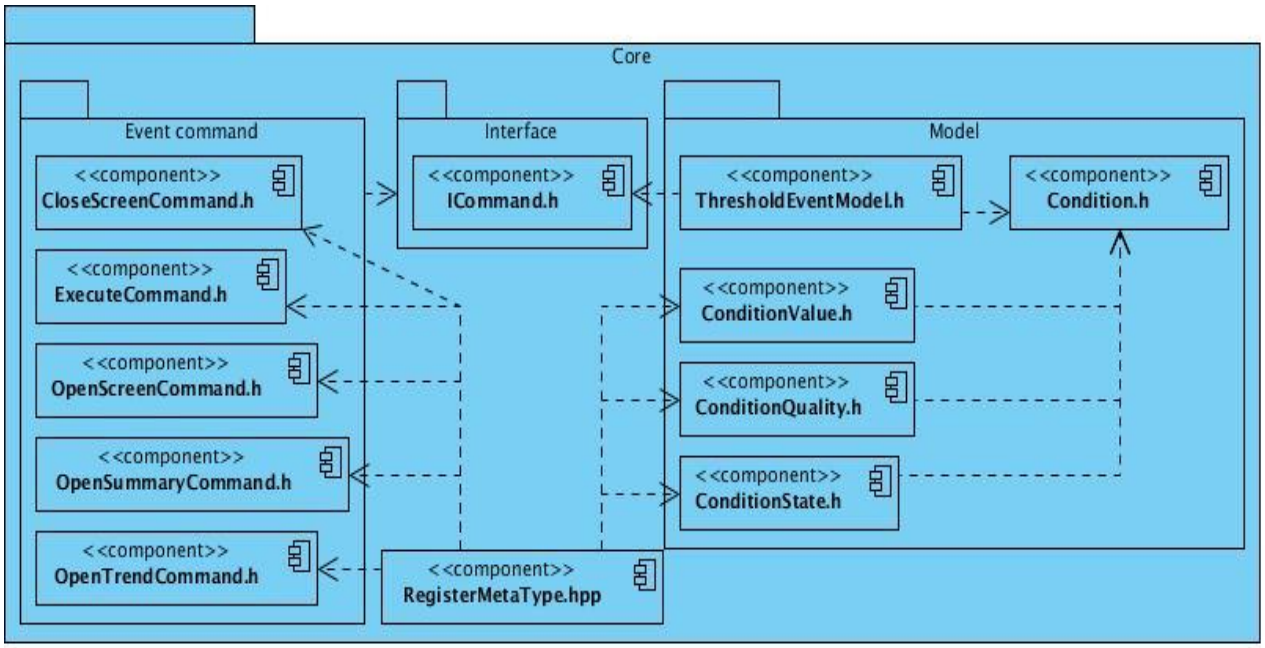


Figura 28 Diagrama de componente Core

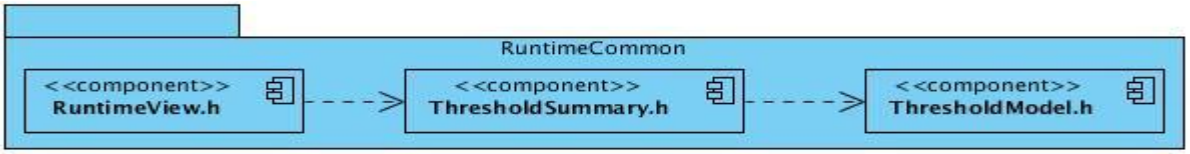


Figura 29 Diagrama de componente RuntimeCommon

3.4.2. Modelo de despliegue

En todo proyecto se hace necesario mostrar la arquitectura del sistema desde el punto de vista del despliegue de los artefactos del software. Estos artefactos representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. Se pueden mencionar los archivos ejecutables, bibliotecas, esquemas de bases de datos y archivos de configuración.

Los diagramas de despliegue se utilizan para mostrar la vista estática de un sistema, mostrando las relaciones físicas entre los componentes de hardware y software en el sistema final. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP.

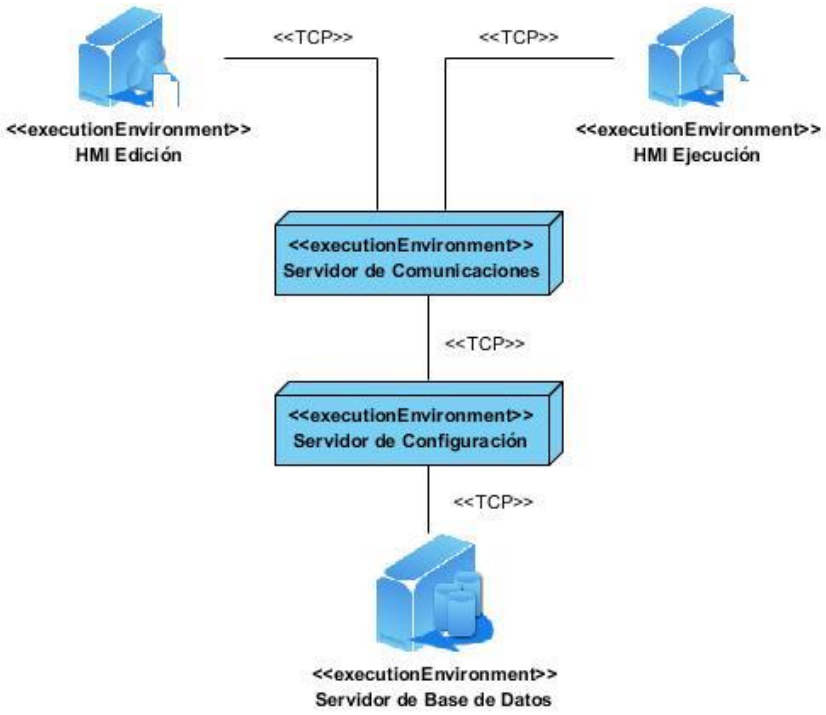


Figura 30 Diagrama de despliegue

El diagrama de despliegue anterior está compuesto por los nodos necesarios, para todo el proceso de configuración y ejecución de un proyecto del SCADA SAINUX. En el nodo del HMI Edición se hace referencia al ambiente de edición, donde se incluye la configuración de los eventos de umbrales. El nodo HMI Ejecución hace referencia al ambiente de ejecución donde se supervisa el proceso, aquí se monitorizan y evalúan los eventos de umbral. Se usa el servidor de comunicaciones para enviar las solicitudes requeridas al servidor de configuración, y a su vez este interactúa con los datos del servidor de base de datos, todas estas conexiones se realizan el protocolo TCP/IP.

3.5. Estándar de código empleado

Los diferentes estándares de codificación, o estilos de programación como también se les llama, definen la estructura y apariencia física del código, facilitando su lectura, comprensión y mantenimiento. Para la implementación de los diferentes aspectos que intervienen en este mecanismo, se utilizó como guía el documento de referencia: Estándares de codificación para C++. (28) Al hacer uso de este, se pretende fomentar las buenas prácticas, y definir estándares para mejorar la comunicación del equipo de trabajo y la gestión del conocimiento en general.

3.6. Pruebas de la solución

La disciplina de prueba es imprescindible para garantizar la calidad y el adecuado funcionamiento de un software. Algunos autores como Krutchen, Pressman, Pfleger, Cardoso y Sommerville afirman que el proceso de Ejecución de Pruebas debe ser considerado durante todo el ciclo de vida de desarrollo del software, para así obtener un producto de alta calidad. Su éxito dependerá del seguimiento de una estrategia de prueba adecuada. La estrategia de prueba de software integra un conjunto de actividades que describen los pasos que hay que llevar a cabo en un proceso de prueba, tomando en consideración cuánto esfuerzo y recursos se van a requerir, con el fin de obtener como resultado una correcta construcción del software. (29) Para asegurar el éxito de las pruebas es imprescindible realizar casos de pruebas que tengan probabilidad de descubrir errores en el sistema.

A la aplicación desarrollada se le aplicaron pruebas de alto nivel, entre estas se encuentran las pruebas de validación que son de suma importancia, ya que es la que proporciona la información de si se satisfacen o no todos los requisitos funcionales. Para la realización de esta prueba se usa exclusivamente el método de prueba de Caja Negra (*Black-Box Testing*). Estas pruebas están centradas en los requerimientos funcionales del software. Permiten derivar conjuntos de condiciones de entrada que ejerciten completamente los requerimientos funcionales de un programa. Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar.

3.6.1. Casos de prueba

Un caso de prueba permite detallar la forma en que se va a probar el sistema, incluyendo los datos de entrada con los que se realizará la prueba correspondiente, las condiciones de ejecución y resultados obtenidos. Deben verificar si el producto satisface los requerimientos del usuario, tal y como se

describe en las especificación de los requerimientos y si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño. (20)

A continuación se muestran los casos de prueba realizados a la aplicación, teniendo en cuenta la ejecución de cada caso de uso del sistema.

Tabla 5 Caso de prueba CU Gestionar eventos de umbral

Sección 1 "Adicionar evento"		
Entrada	**Resultados**	**Condiciones**
El usuario especifica los datos para adicionar un evento de umbral: • Nombre • Descripción • Grupo Operacional • Condiciones • Comandos	El evento es adicionado y visualizado satisfactoriamente al árbol de proyecto.	• El campo Nombre no puede estar vacío. • Debe de existir al menos un Grupo Operacional.
Sección 2 "Eliminar"		
Entrada	**Resultados**	**Condiciones**
Seleccionar en el Explorador de proyecto el nodo que hace referencia a un evento de umbral.	El evento es eliminado satisfactoriamente mostrando el mensaje de confirmación correspondiente.	• Exista previamente un evento creado y visualizado en el árbol de proyecto.
Sección 3 "Modificar propiedades de un evento"		
Entrada	**Resultados**	**Condiciones**
El usuario especifica los datos que desea modificar de un evento de umbral: • Nombre • Descripción • Grupo Operacional • Condiciones • Comandos	El evento es modificado y visualizado satisfactoriamente al árbol de proyecto.	• Exista previamente un evento creado y visualizado en el árbol de proyecto. • El campo Nombre no puede estar vacío.

Tabla 6 Caso de prueba CU Exportar eventos de umbral

Sección 1 "Exportar"

Entrada	Resultados	Condiciones
El usuario especifica la ruta donde será almacenado el fichero de salida.	El evento de umbral es exportado satisfactoriamente en la ruta dada al sistema y en el formato adecuado (*.ev).	• Debe de existir al menos un evento de umbral creado en el sistema.

Tabla 7 Caso de prueba CU Asociar evento de umbral a una consola HMI

Sección 1 "Asociar evento de umbral a una consola HMI"		
Entrada	**Resultados**	**Condiciones**
El usuario deberá seleccionar al menos un evento de umbral y dar clic en la opción que indique asociar dicho evento a la consola.	El sistema elimina el/los evento/s de umbral seleccionado/s de la lista de eventos de umbral disponibles en el sistema. El sistema adiciona el/los evento/s de umbral seleccionado/s a la lista de eventos de umbral asociados a la consola.	• Debe de existir al menos una consola creada en el sistema. • Debe de existir al menos un evento de umbral creado en el sistema.

A continuación se describe las no conformidades detectadas en cada una de las iteraciones y cuáles de estas fueron resueltas satisfactoriamente.

Cada incidencia detectada en el desarrollo de la solución propuesta fue resuelta a raíz del trabajo sistémico del desarrollador. Se detectaron un total de 7 No Conformidades (NC) en la primera iteración, 5 en la segunda, 3 en la tercera y ninguna en la cuarta, las cuales se dividieron en Significativas y No significativas. El resultado de las pruebas realizadas al prototipo funcional propuesto fue satisfactorio, ya que se pudo comprobar, que las entradas coinciden, y las respuestas del sistema fueron las esperadas y, al mismo tiempo cumple con las descripciones de los casos de uso definidos con anterioridad. Con las pruebas aplicadas se comprobó que el sistema cumple con el objetivo planteado al inicio de este trabajo.

Iteración 1:

Se probaron 4 CU (Gestionar evento de umbral, Gestionar condición, Gestionar comando y Asociar evento de umbral a una consola HMI). En la ejecución de las pruebas se detectaron 7 NC, de las cuales 3 fueron Significativas.

- Al dar clic en la opción eliminar no permitía eliminar las condición seleccionada.
- No permitía asociar un punto a la condición.
- No se mostraba las propiedades del comando, lo que impedía configurar el mismo para poder ser adicionado.

Iteración 2:

Se probaron 3 CU (Exportar evento de umbral, Importar evento de umbral y Ejecutar evento de umbral). En la ejecución de las pruebas se detectaron 5 NC, de las cuales 2 fueron Significativas:

- En la funcionalidad de exportar permitía pasar al siguiente escenario sin haber seleccionado al menos un evento.
- Cuando se carga el fichero en el formato adecuado, el sistema no reconoce ningún recurso importado.

Iteración 3:

Se probó 3 CU (Abrir sumarios de evento de umbral, Activar evento de umbral y Desactivar evento de umbral). En la ejecución de las pruebas se detectaron 2 NC, de las cuales 1 fue Significativa.

- Al abrir un sumario de eventos de umbral no se listaban los eventos existentes en el sistema.

Iteración 4:

- Se probaron los 10 CU mencionados anteriormente. En la ejecución de las pruebas no se detectaron NC.

En la figura 31 se muestra lo planteado anteriormente.

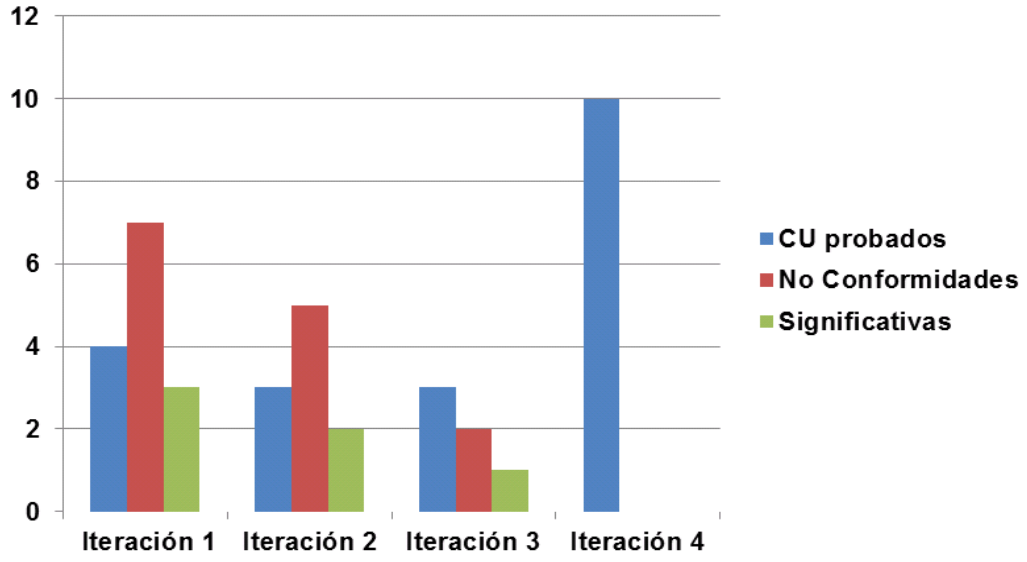


Figura 31 Gráfica de pruebas

Conclusiones parciales

Durante este capítulo quedaron evidenciadas las características principales del mecanismo desarrollado como solución al problema planteado, a través de las vistas de los diagramas expuestos. Los patrones de diseño utilizados ayudaron eficientemente a lograr un adecuado rendimiento del sistema. La realización del modelo de implementación permitió detallar los componentes creados en el desarrollo de la solución y la relación entre ellos. Mediante las pruebas ejecutadas a la solución, se comprobaron las funcionalidades identificadas durante el proceso de desarrollo del software.

Conclusiones generales

Con la realización de este trabajo se concluye que:

- Se dotó al SCADA SAINUX, de un mecanismo que posibilita monitorear constantemente el estado de las mediciones del proceso y, ejecutar una serie de acciones programadas como respuesta ante posibles eventos y/o anomalías del sistema.
- Permitió automatizar la visualización de la información implicada en los posibles sucesos del proceso.
- Permitió reducir las acciones manuales que tiene que hacer un operador ante determinado estado del sistema para la toma de decisión y la acción con el propósito de corregir posibles anomalías durante la supervisión y control del proceso.
- Aumentó el grado de sensibilidad del sistema ante determinados eventos.
- La validación mediante pruebas de caja negra demostró la robustez de la solución y permitió corregir las deficiencias encontradas.

Recomendaciones

Durante el desarrollo del mecanismo que se propone se han identificado ciertas mejoras que podrían implementarse en un futuro, en aras de darle una mayor efectividad y utilidad al producto obtenido. Es por ello que se recomienda para versiones posteriores, tener en cuenta los siguientes puntos:

- Permitir la configuración de otras acciones sobre el HMI ante los eventos que ocurren, como generación de reportes, abrir gráficos XY, forzar el apagado o encendido de un punto, mostrar detalles del punto, entre otras.
- Aplicar técnicas de optimización de código a las nuevas funcionalidades con el objetivo de mantener y mejorar el código desarrollado.
- Evaluar las condiciones del umbral teniendo en cuenta los niveles de precedencia de los operadores lógicos.

Referencias bibliográficas

1. **Rodríguez Penin, Aquilino.** *Sistemas SCADA 2da Edición.* Barcelona : Marcombo SA, 2007. ISBN 978-84-267-1450-3.

2. **Hungerford, Janice y York, Danetta.** *[En línea] http://www.gasindustries.com/articles/gijul01c.htm.*

3. **A. Boyer, Stuart.** *Scada: Supervisory Control And Data Acquisition. EUA : International Society of Automation, 2009. ISBN:1936007096.*

4. **Ruíz Lizama, Edgar, Inche Mitma, Jorge Luis y Chung Pinzás, Alfonso Ramón.** *Biblioteca digital Arístides Rojas. Desarrollo de una interfaz hombre máquina orientada al control de procesos. [En línea] 2008. http://bibliodar.mppeuct.gob.ve/?q=doc_categoria/SCADA.*

5. **Hernández, Judeli.** *Implementación de objetos gráficos para el desarrollo de despliegues operacionales en el sector comercio y suministro del SCADA Nacional del PDVSA. Mérida, Venezuela : s.n : s.n., 2008.*

6. **Bailey, David y Wright, Edwin.** *Practical SCADA for Industry. Oxford : Newnes, 2003. ISBN 07506-58053.*

7. **Cañizares, Julio Rubén.** *Eventos asociados a umbrales (thresholds). PDVSA, Lagunillas : s.n., 2014.*

8. **Letelier, Patricio, y otros.** *Metodologías Ágiles en el Desarrollo de Software. 2003.*

9. **Rodríguez Sánchez, Tamara.** *Metodología de desarrollo para la actividad productiva de la UCI. Programa de mejora. 2014-2015.*

10. **...Craig., Larman.** *UML y Patrones. Introducción al análisis y diseño Orientado a Objetos.s.l.:. s.l. : Prentice Hall.*

11. **Asensio, Menendez-Barzanallana, Rafael.** *Herramientas CASE. Ingeniería desoftware. .s.l.: Informática Aplicada a la gestión Pública. 2011.*

12. **Stroustrup., Bjarne.** *The C++ Programming Language 3rd Edition. [En línea] 1997. https://www.google.com.cu/?gws_rd=cr,ssl&ei=I7dLVbmzOM7IsASb9IDIBA#q=Stroustrup%2C+Bjarne.The+C%2B%2B+Programming+Language+%28Third+Edition%29:Addison-Wesley%2C+1997..*

13. **The Qt Company Ltd.** ***Qt Documentation. .*** *[En línea] 2015. http://doc.qt.io/qt-4.8/qt-overview.html..*

14. Qt Creator. ***Qt Project.*** *[En línea] 2012. http://qt-project.org/wiki/Category:Tools::QtCreator_Spanish..*

15. Bartolome Sints, Marco. ***XML:Lenguaje de Marcas Extensible. .*** *[En línea] mayo de 2013. www.mdibre.org....*

16. Linuxsalto. ***Linuxsalto.*** *[En línea] http://linuxsalto.org/introduccion-al-software-libre.*

17. Presman, , Roger. *Ingeniería de Software un Enfoque Practico.6ta Edicion.*

18. Chaves, Michael. *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de Software. 2005.*

19. González Cornejo, José Enrique. *El Lenguaje de Modelado Unificado (UML). DocIRS. [En línea] http://www.docirs.cl/uml.htm..*

20. Pressman, Roger. *Ingeniería de Software un Enfoque Práctico. 5ta Edición.*

21. Hilliard, Rich. *Recommended Practice for Architectural Description of Software-Intensive Systems. [En línea] 14 de noviembre de 2000. http://www.enterprise-architecture.info/Images/Documents/IEEE%201471-2000.pdf..*

22. et al, Len Bass. *Software Architecture in Practice (2nd Edition). [En línea] abril de 2003. http://www.tedfelix.com/software/bass2003.html.*

23. K Qian, Wang. *Component-oriented programming. 2005.*

24. Reynoso, Carlos y Kicillof, Nicolás. *Estilos y Patrones en la Estrategia de Arquitectura de Microsof.*

25. Aragón Cáceres, José Antonio y Cepero Nuñez, Mariela. *Arquitectura del sistema de automatización industrial SAINUX. 2015.*

26. C., Larman. ***UML Y PATRONES.*** *Una introducción al analisis y diseño orientado a objetos y al proceso unificado.Segunda edición. s.l. : PEARSON EDUCACIÖN, S.A., Madrid,, 2003.*

27. Grosso, Andrés. *Prácticas de Software. Patrones GRASP. [En línea] 2011. http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.*

28. Sommerville, Ian. *Ingeniería de Software. 2005.*

29. UCI. ***0120_1*** *Estándares de codificación para C++ Proyecto SCADA Guardián del ALBA. . Cuba : s.n.*

30. Pressman, R. *Ingeniería del Software:Un enfoque Práctico.McGraw Hill. 2002.*

31. Chacón, David, Dijort, Oscar y Castrillo, Jacinto. *UPCOpenCourseware. Supervisión y control de procesos. [En línea] 2002. http://ocw.upc.edu/sites/default/files/materials/15012628/40194-3452.pdf.*

32. ***Ingenieriadel software. Guión Visual Paradigm para UML. .*** *. [En línea] Curso 2013-2014. http://www.libreriapastor.com/file/visual-paradigm/.*

33. de la Horra Diaz, Abdelaziz. *Desarrollo SCADA Guardián del ALBA. Especificación de Eventos. 2011.*

34. Márquez, Abel. *Descripción funcional de requerimientos. Asociación de un despliegue a una alarma.* 2013.

Anexos

Tabla 8 CU Gestionar condición

Objetivo	El objetivo del CU es gestionar las condiciones de un evento de umbral.
Actores	Supervisor
Resumen	El CU permite crear, eliminar y configurar las condiciones de un evento de umbral.
Referencias	RF6,
Prioridad	Critico
Precondiciones	Exista un proyecto activo con al menos un GOP.
Flujo de eventos	
Flujo Normal de eventos	
Actor	**Sistema**
1. El CU inicia cuando el supervisor da clic en el botón que se establece para las condiciones en la ventana de Evento de Umbral.	2. Muestra la ventana Colección de condiciones, con las opciones de Adicionar y Eliminar.
	3. Termina el CU.
Sección 1: "Adicionar"	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona la opción "Adicionar".	2. Adiciona una fila a la tabla de condiciones con los siguientes datos: • Punto • Tipo • Comparación • Valor • Condición
3. Da clic en el campo "Punto".	4. Muestra en una ventana el sumario de puntos del proyecto.
5. Da doble clic en un punto del sumario.	6. Establece el punto seleccionado en la condición.

7. Establece los campos • tipo • comparación • valor • condición 8. Da clic en el botón “Aceptar”. (Alterno 1)	9. Adiciona la condición al evento de umbral. 10. Ir al paso 3 del flujo normal de eventos.

Flujos Alternos	
Evento 1	
Actor	**Sistema**
1. Da clic en el botón “Cancelar”.	2. Ir al paso 3 del flujo normal de eventos

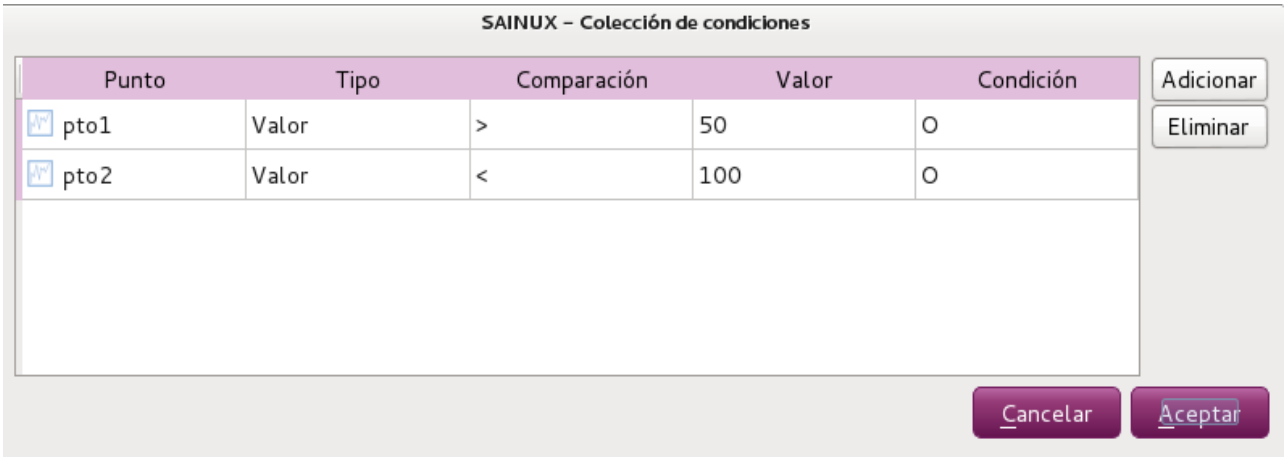


Figura 32 Prototipo de Interfaz Gestionar condición

Sección “Eliminar”	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona una condición de la lista de condiciones. 2. Da clic en el botón eliminar.	3. Elimina la fila seleccionada de la lista de condiciones.
4. Da clic en el botón “Aceptar”. (Alterno 1)	5. Elimina la condición del evento de umbral. 6. Ir al paso 3 del flujo normal de eventos.
Flujos Alternos	
Evento 1	
Actor	**Sistema**

1. Da clic en el botón “Cancelar”.		2. Ir al paso 3 del flujo normal de eventos
Relaciones	**Caso de uso del cual depende(padre)**	
	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		

Tabla 9 CU Gestionar comando

Objetivo	El objetivo del CU es gestionar los comandos de un evento de umbral.	
Actores	Supervisor	
Resumen	El CU permite crear, eliminar y configurar los comandos de un evento de umbral.	
Referencias	RF7, RF9, RF10, RF11, RF12, RF14	
Prioridad	Critico	
Precondiciones	Exista un proyecto activo con al menos un GOP.	
Flujo de eventos		
Flujo Normal de eventos		
Actor	**Sistema**	
1. El CU inicia cuando el supervisor da clic en el botón que se establece para los comandos en la ventana de Evento de Umbral.	2. Muestra la ventana Asociar comandos, con las opciones de • Adicionar • Eliminar.	
	3. Termina el CU.	
Sección 1: “Asociar comando”		
Flujo de eventos		
Actor	**Sistema**	
1. Selecciona la opción “Adicionar”.	2. Aparece una lista desplegable con los comandos que se pueden adicionar. Las opciones de comandos son las siguientes: a) Cerrar despliegue b) Ejecutar aplicación	

	c) Abrir despliegue d) Abrir sumario e) Abrir tendencia
3. Da doble clic en el comando que desea adicionar	4. Muestra en la parte derecha de la ventana la propiedad y el valor para cada comando.
5. Si escoge la opción se introduce una descripción y se escoge el despliegue que se desea cerrar. 6. Da clic en el botón “Aceptar”. (Alterno 1) 7. Si escoge la opción b se introduce una descripción y se escoge la ruta donde está la aplicación que se desea ejecutar. 8. Da clic en el botón “Aceptar”. (Alterno 1) 9. Si escoge la opción c se introduce una descripción, se escoge un despliegue, el tipo de navegador y la cantidad de reemplazos. 10. Da clic en el botón “Aceptar”. (Alterno 1) 11. Si escoge la opción d se introduce una descripción y se escoge el tipo de sumario ya sea: • Sumario de despliegue • Sumario de reportes • Sumario de alarmas • Sumario de puntos analógicos • Sumario de puntos digitales • Sumario de eventos • Sumario de sub-canales • Sumario de dispositivos	13. Adiciona el comando al evento de umbral. 14. Ir al paso 3 del flujo normal de eventos.

• Sumario de estado del sistema 12. Si escoge la opción e se introduce una descripción y se seleccionan las mediciones (puntos).	
Flujos Alternos	**Flujos Alternos**
Evento 1	**Evento 1**
Actor	**Actor**
1. Da clic en el botón “Cancelar”.	2. Ir al paso 3 del flujo normal de eventos

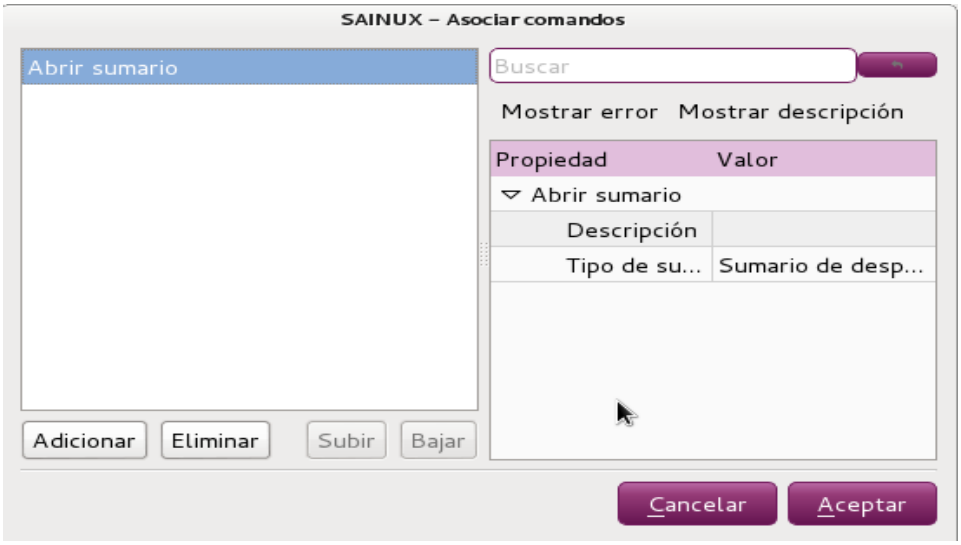


Figura 33 Prototipo de Interfaz Gestionar comando.

Sección 2: “Eliminar”	
Flujo de eventos	
Actor	**Sistema**
1. Selecciona un comando de la lista desplegable. 2. Da clic en el botón eliminar.	3. Elimina el comando de la lista.
4. Da clic en el botón “Aceptar”. (Alterno 1)	5. Elimina el comando del evento de umbral. 6. Ir al paso 3 del flujo normal de eventos.
Flujos Alternos	
Evento 1	
Actor	**Sistema**

1. Da clic en el botón “Cancelar”.		2. Ir al paso 3 del flujo normal de eventos
Relaciones	**Caso de uso del cual depende**	
	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		

Tabla 10 CU Importar evento de umbral

Objetivo	El objetivo del CU es importar los eventos de umbral de un proyecto.	
Actores	Supervisor	
Resumen	El CU permite importar un fichero en disco los eventos de un proyecto.	
Referencias	RF5	
Prioridad	Secundario	
Precondiciones	Exista un proyecto activo.	
Flujo de eventos		
Flujo Normal de eventos		
Actor		**Sistema**
1. El CU inicia cuando el supervisor da clic derecho en el ítem “Eventos de Umbral” en el árbol de proyecto.		2. Muestra un menú con las opciones: Agregar evento Importar Exportar Eliminar todo
3. Selecciona la opción “Importar”.		4. Muestra la ventana Importar Eventos de Umbral. (alterno 1)
5. Se selecciona el archivo que se desea importar. 6. Da clic en el botón Finalizar.		7. Se abre los eventos seleccionados en el fichero. 8. Termina el CU.
Flujos Alternos		
Evento 1		
Actor		**Sistema**
		1. Si no existen eventos de umbral en el proyecto,

	se muestra el siguiente mensaje: “No existen recursos a importar.”
2. Da clic en el botón Aceptar.	3. Ir al paso 14 del flujo normal de eventos.

Figura 34 Importar evento de umbral

Relaciones	**Caso de uso del cual depende**	
	CU Incluidos	
	CU Extendidos	
Requisitos no funcionales		

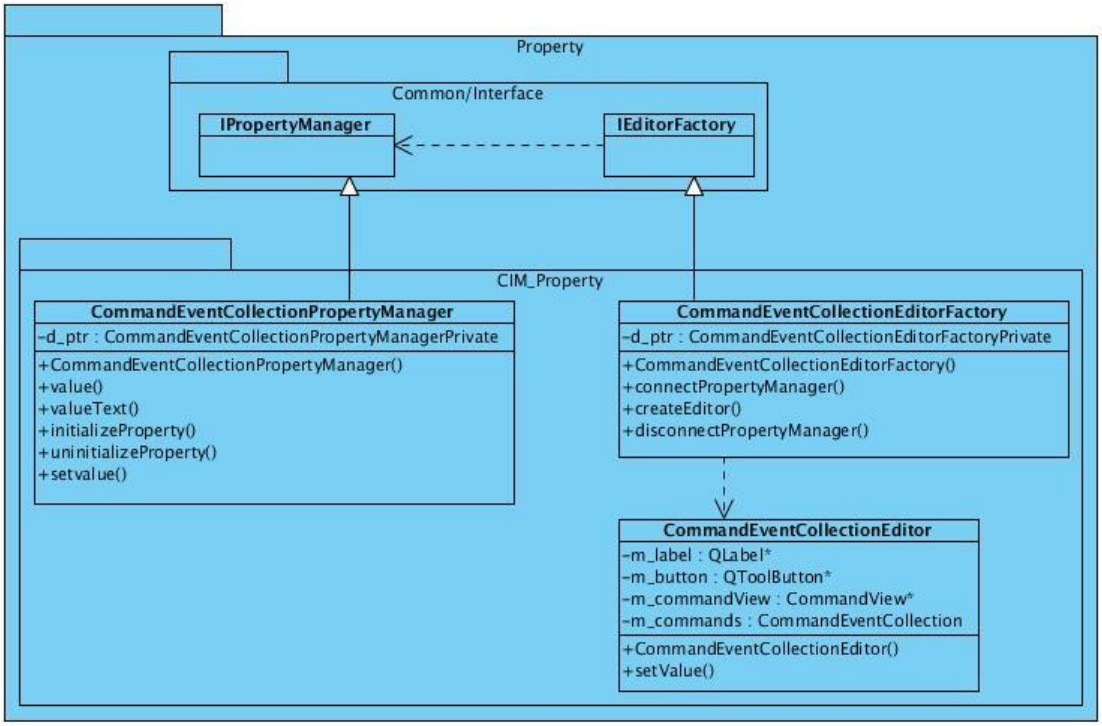


Figura 35 Diagrama de Clases de Diseño Property

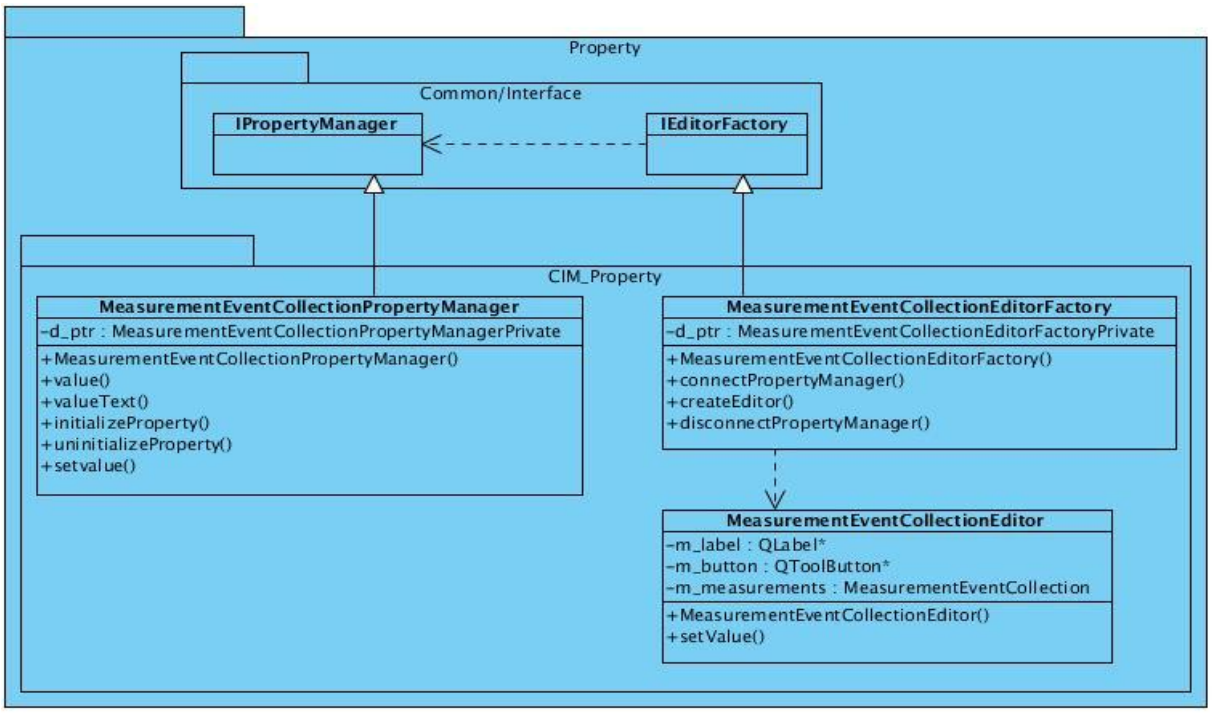


Figura 36 Diagrama de Clases de Diseño Property

Tabla 11 Caso de prueba CU Gestionar condición

Sección 1 "Adicionar"		
Entrada	**Resultados**	**Condiciones**
El usuario especifica los datos. • Punto • Tipo • Comparación • Valor • Condición	La condición fue adicionada satisfactoriamente.	• Debe existir previamente al menos un punto.
Sección 2 "Eliminar"		
Entrada	**Resultados**	**Condiciones**
Seleccionar en la ventana de las condiciones el nodo que hace referencia a una condición.	Se eliminar satisfactoriamente del evento de umbral.	• Debe de existir al menos una condición asociada al evento de umbral.

Tabla 12 Caso de prueba CU Gestionar comando

Sección 1 "Adicionar"		
Entrada	**Resultados**	**Condiciones**
• Si escoge la opción Cerrar despliegue se introduce una descripción y se escoge el despliegue que quiero cerrar. • Si escoge la opción Ejecutar aplicación se introduce una descripción y escojo la ruta donde está la aplicación que se quiera ejecutar. • Si escoge la opción Abrir despliegue se introduce una descripción, se escoge un despliegue, el tipo de	El comando fue adicionado satisfactoriamente al evento de umbral.	• Debe existir al menos un comando creado en el sistema. • Debe existir al menos un despliegue.

navegador y cantidad de reemplazos. • Si escoge la opción Abrir sumario se introduce una descripción y se escoge el tipo de sumario ya sea: ✓ Sumario de despliegue. ✓ Sumario de reportes. ✓ Sumario de alarmas. ✓ Sumario de puntos analógicos. ✓ Sumario de puntos digitales. ✓ Sumario de eventos. ✓ Sumario de sub-canales. ✓ Sumario de dispositivos. ✓ Sumario de estado del sistema. • Si escoge la opción Abrir tendencia se introduce una descripción y se selecciona las mediciones (puntos).		
Sección 2 "Eliminar"		
Entrada	**Resultados**	**Condiciones**
Seleccionar en la ventana de los comandos el nodo que hace referencia a un comando.	El comando se elimina satisfactoriamente del evento de umbral.	• Debe de existir al menos un comando adicionado al evento de umbral.

Tabla 13 Caso de prueba CU Importar evento de umbral

Sección 1 "Importar"		
Entrada	**Resultados**	**Condiciones**
La ruta en donde está guardado el fichero de	El evento de umbral es importado satisfactoriamente	• Debe de existir al menos un fichero de evento de

entrada.	desde la ruta donde está guardado en el sistema y en el formato adecuado (*.ev).	umbral guardado.

Glosario de términos

Supervisión: Conjunto de acciones desempeñadas con el propósito de asegurar el correcto funcionamiento del proceso incluso en situaciones anómalas. Función que permite al operador observar en un monitor la evolución en tiempo real de los parámetros de funcionamiento del proceso.

Monitorización: Sistema de vigilancia y asistencia al operario en el que se cumplen solo algunas de las etapas de supervisión.

Control: Función del sistema SCADA encargada de realizar las tareas que permiten mantener al sistema dentro de unos estrechos márgenes gracias a las posibilidades de actuación sobre el proceso.

Software: Es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.

Área operacional: Espacio físico donde se lleva a cabo el proceso productivo o parte del mismo de determinada industria.