FACULTAD 5

Centro de Informática Industrial

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

Título: Componentes gráficos para la representación de interruptores y selectores en el SCADA SAINUX.

Autor: **Javier Bravo Calzado**

Tutor: **Ing. Luis Andrés Valido Fajardo**

"La Habana, Mayo de 2016"

"Año 58 de la Revolución"

Declaración de Auditoría

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

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

Javier Bravo Calzado

Firma del Autor

ing. Luis Andrés Valido Fajardo

Firma del Tutor

Agradecimientos

Son muchas las personas a quienes debo agradecerle por su apoyo a lo largo de la carreara y especialmente en este trabajo:

A la revolución por darme la oportunidad de estudiar esta maravillosa carrera.

A mis padres, quienes son mi razón de ser, son una gran inspiración para mí. Gracias por estar siempre ahí para aconsejarme y guiarme por buenos caminos.

Especialmente a mi madre Amarylis Calzado Fuentes por serlo todo en mi vida.

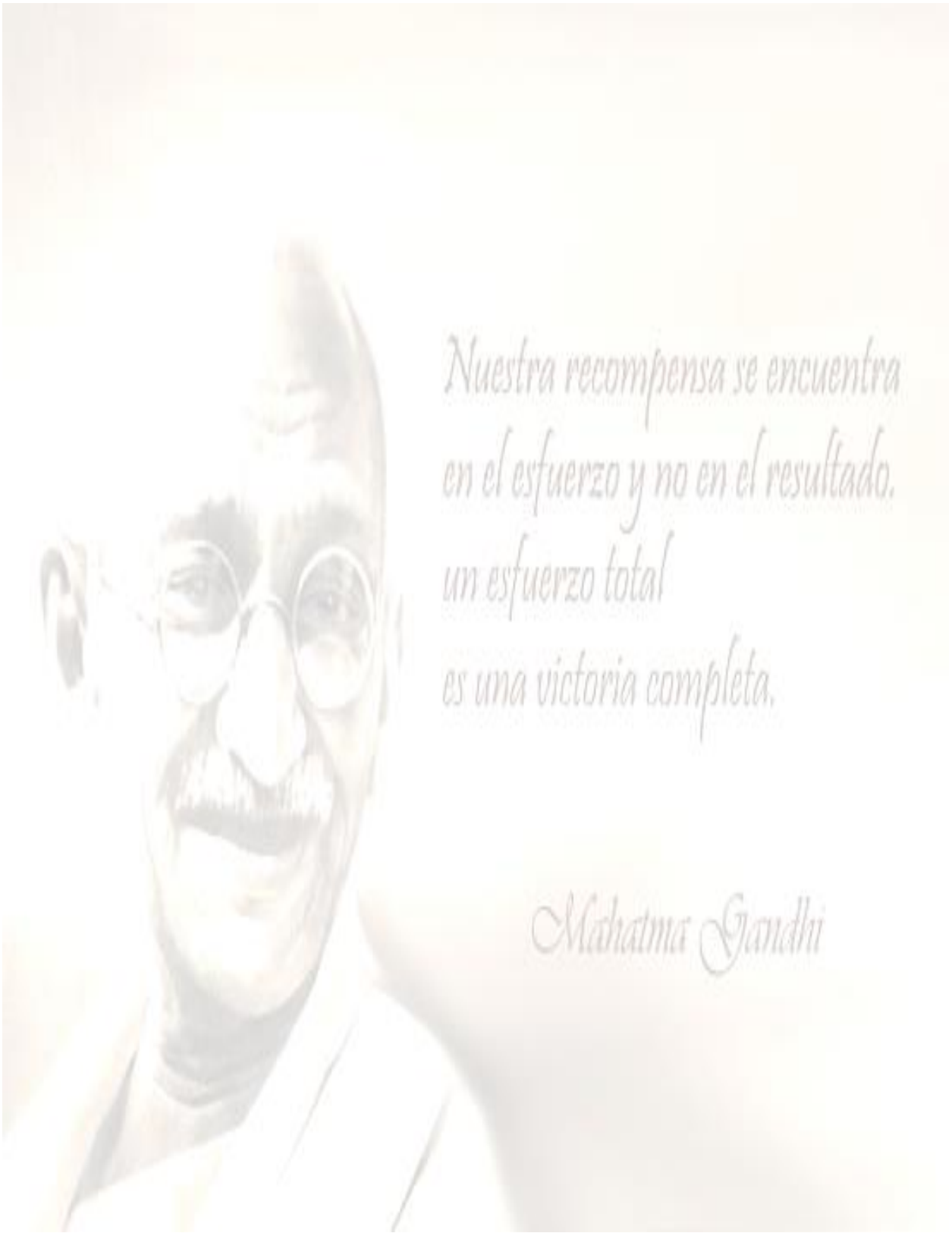
A mis abuelos, Elia y José Triniño por su cariño y constante preocupación, gracias por estar siempre pendientes de su nieto.

A las personas que compartieron conmigo estos años, gracias por todos los momentos que pasamos juntos.

A mi tuto Luis Valido por ser la persona que con sus gracias, regaños e insistencias, se convierte además de un gran guía inigualable, en un amigo de por vida.

A mis compañeros de la línea HMI que me han brindado apoyo y su ayuda en la realización de este trabajo de diploma y en otras actividades relacionadas con el mundo del software.

A mis verdaderos amigos que aunque no son muchos siempre están ahí para lo que necesite y me han brindado su amistad incondicionalmente.


Nuestra recompensa se encuentra
en el esfuerzo y no en el resultado.
un esfuerzo total
es una victoria completa.
Mahatma Gandhi

Resumen

En el mundo moderno la utilización de interruptores y selectores son empleados en diferentes campos de la sociedad entre los más frecuentes en industrias. Sus tipos y aplicaciones son innumerables, van desde un simple interruptor que apaga o enciende una bombilla, hasta un complicado selector de transferencia automático de múltiples capas, controlado por computadora. El Centro de Informática Industrial (CEDIN) donde se desarrollan varios productos entre los que se encuentra el SCADA SAINUX (Sistema de Automatización Industrial basado en GNU/LINUX), es un sistema distribuido, orientado a componentes y en proceso de desarrollo por el centro. Unos de los módulos que componen el SCADA SAINUX es la Interfaz Hombre Máquina el cual constan de dos ambientes: el ambiente de configuración (AC) y el ambiente de visualización (AV). Tanto para el AC como para el AV, la Interfaz Hombre Máquina provee una biblioteca de componentes gráficos (CG) que permiten recrear de una manera lo más real posible los proceso de campos. Los CG constan entre sus limitantes el no poseer un componente gráfico para la representación de interruptores y selectores. El presente trabajo tiene como objetivo principal el desarrollo de componentes gráficos para la representación de interruptores y selectores en la Interfaz Hombre Máquina del SCADA SAINUX. En el trabajo se utilizó Agile Unified Process como metodología de desarrollo, lenguaje de modelado UML, la herramienta CASE Visual Paradigm, el lenguaje de programación C++. Como resultado se logró ampliar la gama de componentes gráficos para representar dispositivos de tipo Interruptor y Selector.

Palabras claves: Componente gráfico, Interfaz Hombre Máquina, interruptores, selectores, SCADA.

Índice

Índice de Tablas

Índice de Ilustraciones

INTRODUCCIÓN

En poco tiempo las Tecnologías de la Información y las Comunicaciones (TIC) han evolucionado considerablemente, hasta el punto de volverse casi imprescindible para algunos sectores de nuestra sociedad. La automatización de los procesos industriales se ha convertido en un gran avance para el sector empresarial, con su uso, se han revolucionado las producciones a gran escala, reduciendo el peligro de catástrofes, además de obtener grandes ahorros en tiempo, dinero y recursos.

En las industrias se utilizan diferentes herramientas para la ejecución, monitoreo y control de los procesos industriales. Recientemente estas herramientas han ganado en complejidad y funcionalidad de forma exponencial, lo que ha permitido una disminución considerable en la carga de trabajo de los trabajadores en cuanto a tareas se refiere, así como la peligrosidad en las labores que realizan. En la actualidad, los procesos industriales se monitorean y controlan mediante los sistemas SCADA (Sistemas de Supervisión, Control y Adquisición de Datos). Estos sistemas proporcionan gran información de los procesos de forma oportuna para la toma de decisiones y centralizan el funcionamiento de una empresa en una reducida cantidad de estaciones de trabajo, mejorando altamente la eficacia de dichos procesos. (Penin, 2012)

En la Universidad de las Ciencias Informáticas (UCI) se implementa un esquema estructurado de estudio-trabajo, en el cual se instruye a los estudiantes tanto en su formación docente como en su vida laboral, basados en la integración de procesos fundamentales como la formación, investigación y la producción entorno a una temática de un centro de desarrollo. La universidad a su vez cuenta con un Centro de Informática Industrial (CEDIN), perteneciente a la facultad 5, el cual está desarrollando varios software para contribuir a la automatización de procesos industriales, destacando entre los productos el sistema SCADA SAINUX (Sistema de Automatización Industrial basado en GNU/LiNUX). Dicho sistema está compuesto por varios módulos que interactúan entre sí, comenzando el proceso por los controladores de dispositivos y finalizando en la Interfaz Hombre-Máquina (HMI). El HMI está compuesto por un ambiente de edición o editor, que es donde se realizan las configuraciones para la supervisión, y el ambiente de ejecución o visualizador, que permite al operador supervisar y controlar los procesos industriales.

El módulo de HMI en el SCADA SAINUX se encarga de representar, en un ordenador, los procesos que ocurren en el campo, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador total control. Este módulo

es el que permite al operador estar en contacto directo con el sistema, realizar la supervisión y el control del proceso en general.

El HMI constan de dos entornos: El entorno de configuración (EC), donde los mantenedores configuran la información específica del área que se desea supervisar y diseñan los despliegues, los cuales haciendo uso de los componentes gráficos permiten simular los procesos de campo; el entorno de visualización (EV) es donde el operador puede supervisar y controlar la configuración realizada en el EC, interactuando con los componentes gráficos para emitir control sobre el sistema, monitorear los cambios de estado de las variables, gestionar alarmas, generar reportes.

Tanto para el EC como para el EV el HMI provee una biblioteca de componentes gráficos (CG) que permiten recrear de una manera lo más real posible los procesos de campo. Estos gráficos pueden ser simples figuras geométricas como: línea, rectángulo, elipse, texto, polígono, polilínea, curvas; pero también muy complejas como los componentes de hardware utilizados en la industria que se desean supervisar. Dentro de estos componentes se encuentran los destinados para ejercer control sobre las variables presente en el proceso y que alteran el estado de dicho proceso, además permiten ejercer el control sobre procesos adyacentes y secundarios al proceso principal que se monitorea y controla desde un sistema SCADA como puede ser los sistemas de iluminación, ventilación y acceso de los locales donde se encuentra instalado el sistema SCADA SAINUX.

La biblioteca de componentes gráficos presenta deficiencias para proporcionarles a los mantenedores o encargados de realizar las configuraciones componentes gráficos que represente dichos elementos. Todo lo anterior expuesto trae como consecuencia.

1. La utilización de imágenes para representar dichos dispositivos, trae consigo la pérdida de calidad cuando esta se redimensiona.
2. La acción de realizar una operación de control sobres los dispositivos reales se torna engorrosa en sus variantes, la primera el mantenedor se ve obligado a conjugar la imagen con algún otro componente de control presente en la biblioteca (CG), mientras en la segunda el operador tendría que buscar la variable sobre la cual quiere actuar en el sumario y desde ahí ejercer la acción deseada.

Teniendo en cuenta la situación planteada con anterioridad se define el siguiente **problema de investigación:** ¿Cómo contribuir a la representación dispositivos de control de tipo interruptor y selector en la Interfaz Hombre Máquina de sistemas SCADA SAINUX?

Definiendo el autor como **objeto de estudio:** La representación de dispositivos de control de tipo interruptor y selector en la interfaz Hombre-Máquina de sistemas SCADA.

Siendo el **campo de acción:** La representación de dispositivos de control de tipo interruptor y selector en la Interfaz Hombre Máquina del sistema SCADA SAINUX.

En concordancia con lo anterior se propone como **objetivo general:** Desarrollar componentes gráficos que permitan elevar los grados de representación de interruptores y selectores en el proceso de visualización de la Interfaz Hombre Máquina del sistema SCADA SAINUX.

Para dar cumplimiento a los objetivos de esta investigación se definieron las siguientes **tareas investigativas:**

1. Elaboración del marco teórico de la investigación a través del estudio del estado del arte que existe actualmente sobre el tema.
2. Identificación de los principales interruptores y selectores utilizados y representados en sistemas SCADA.
3. Caracterización de los principales interruptores y selectores utilizados y representados en sistemas SCADA.
4. Realización del levantamiento de requisitos funcionales y no funcionales.
5. Implementación de componentes gráficos que permitirá la solución al problema planteado.
6. Realización de pruebas para validar el cumplimiento de los requerimientos.

Durante la investigación se llevan a cabo varios **métodos y técnicas** en la búsqueda y procesamiento de la información como son:
Métodos Teóricos.

- **Analítico-sintético:** Para el estudio de los conceptos empleados en los sistemas SCADA, analizando todos los documentos elaborados, para la extracción de los elementos más importantes.
- **Histórico-lógico:** Para la comprensión de los antecedentes y las tendencias actuales referidas la evolución en el mundo de los sistemas SCADA.
- **Modelación:** Permitió la elaboración de múltiples diagramas para un mejor entendimiento del problema y solución.

Métodos Empíricos.

- **Observación**: Se puso en práctica este método para conocer el funcionamiento existente en los despliegues del SCADA SAINUX mediante el comportamiento de los dispositivos de campo en las propiedades de los objetos gráficos y sumarios empleados para la toma de decisiones de los operadores.
- **Consulta Bibliográfica:** Empleada para consultar las fuentes de información relacionados con los tipos de los sumarios y objetos gráficos visualizados en los entornos web de los HMI en sistemas SCADAS.

El presente documento está estructurado en tres capítulos:

Capítulo 1. Fundamentación teórica: se definen los conceptos y principios utilizados durante toda la investigación, y se presentan los argumentos teóricos que responden a las técnicas a emplear y el porqué de la necesidad de tener componentes gráficos de tipo de control en el HMI del SCADA SAINUX.

Capítulo 2. Análisis y diseño dela solución: se analiza e implementa la arquitectura propuesta y se explican las principales funcionalidades. Tomándose como base la metodología Proceso Unificado Ágil (AUP[1]) para guiar el proceso de desarrollo.

Capítulo 3. Implementación y prueba: se detallan algunos aspectos de la etapa de implementación del componente gráfico, al igual se detallan las pruebas realizadas para garantizar la robustez de la aplicación desarrollada.

[1] Por sus siglas en inglés (Agile Unified Process).

CAPÍTULO 1

Fundamentación Teórica

1.1 Introducción

Este capítulo tiene como objetivos definir y elaborar un marco teórico donde se expongan temas fundamentales que sustenten la investigación; además de hacer una valoración de las principales tecnologías y herramientas que serán empleadas para dar solución al problema planteado, definiendo las características que las hacen relevantes ante otras de su tipo. Se tratan aspectos relacionados con los sistemas SCADA, haciendo una breve historia relacionada con la evolución de estos sistemas.

1.2 Definición de SCADA

El objetivo principal de la automatización es disminuir la intervención del operador humano en la ejecución de los procesos de producción de las empresas. Con vista a cumplir esa exigente meta se han desarrollado en los últimos años algunos sistemas denominados SCADA. (Penin, 2012)

Los sistemas SCADA, comprenden las soluciones de aplicación que requieren de la captura de información de un proceso o planta industrial, la cual es utilizada para realizar una serie de análisis o estudios con los que se pueden obtener valiosos indicadores de los procesos para mejorar la eficacia del monitoreo y control del proceso y de la toma de decisiones operacionales apropiadas. A su vez se comunican con los dispositivos de campo y controlan el proceso de forma casi automática desde la pantalla de un ordenador u otra tecnología de comunicación. Otras de las características de los SCADA es la adquisición y el almacenamiento de información en bases de datos de tiempo real, representación gráfica de los procesos de manera animada, creación de gráficos de tendencia a partir de los valores de las variables en el tiempo. (Dagoberto Monteros, 2004.)

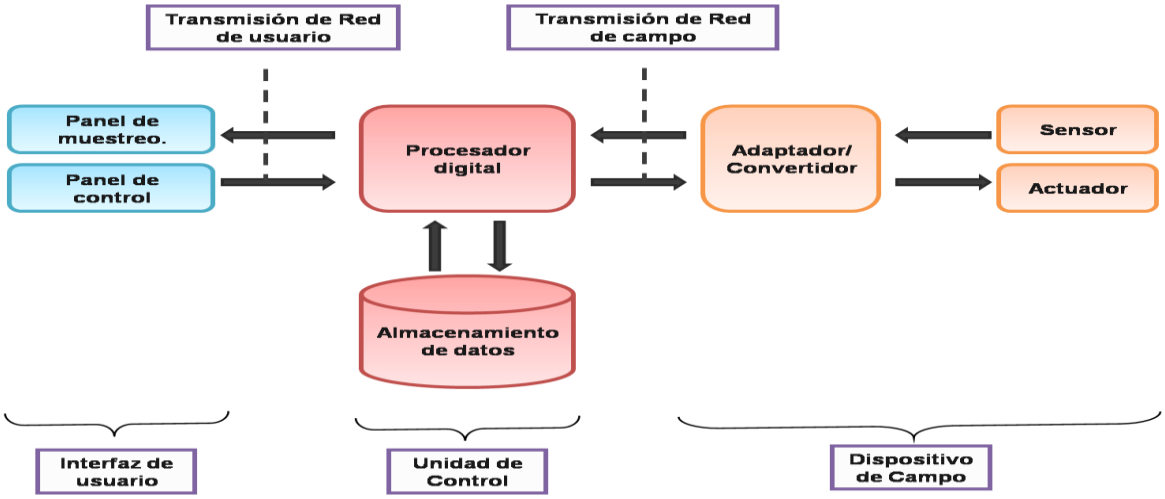


Ilustración 1: Estructura básica de un SCADA

1.2.1 Funciones principales que debe cumplir un SCADA

Entre las funciones principales de un sistema SCADA se encuentran:

- Adquisición de datos: para recoger, procesar y almacenar la información recibida.
- Supervisión: para observar desde un monitor la evolución de las variables de control.
- Control: para modificar la evolución del proceso, actuando bien sobre los reguladores autónomos básicos (consignas, alarmas, menús, etc.) o directamente sobre el proceso mediante las salidas conectadas. (Dagoberto Monteros, 2004.)

1.2.2 Módulos del SCADA SAINUX

Los bloques de software o módulos que intervienen en las actividades de adquisición, supervisión y control de los sistemas SCADA SAINUX se explican a continuación:

Configuración: Permite al usuario definir el entorno de trabajo de su SCADA, adaptándolo a la aplicación particular que se desea desarrollar. (B.O.D. Servicio de Biodinternet, 2015)

Interfaz hombre-máquina: Proporciona al operador las funciones de control y supervisión de procesos. El proceso se representa mediante sinópticos[2] gráficos almacenados en el ordenador de proceso y generados desde el editor incorporado al sistema SCADA. Tradicionalmente estos sistemas consistían en paneles compuestos por indicadores y comandos, tales como luces pilotos, indicadores digitales y analógicos, registradores, pulsadores, selectores y otros que se interconectaban con la máquina o proceso. En la actualidad, dado que las máquinas y procesos en general están implementadas con controladores y otros dispositivos electrónicos que dejan disponibles puertas de comunicación, es posible contar con sistemas de HMI mucho más poderosos y eficaces, además de permitir una conexión más sencilla y económica con el proceso o máquinas. (2015)

Adquisición: Ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas. La programación se realiza por medio de bloques de programa en lenguaje de alto nivel (como C, Basic, etc.). Presenta como principales características la adquisición de datos del nivel de recolección en tiempo real, el procesamiento de variables calculadas, la detección y el manejo de alarmas, la conversión de unidades, el control de calidad de los datos recolectados, la publicación a los clientes de los puntos y la sucesión de alarmas y eventos. (Morales.)

Base de Datos Históricos: Se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos. (Yepes, 2001)

Comunicaciones o Middleware: Se encarga de la transferencia de información entre los diferentes servicios del SCADA y el resto de elementos informáticos de gestión. (2003)

[2] Se denomina cuadro sinóptico a un esquema que refleja en modo gráfico y con palabras la estructura que un determinado texto utiliza para desarrollar un tema.

Seguridad: Permite a los usuarios autenticarse[3] en el sistema, y de esta forma poder acceder sólo a los recursos que tiene asignado su rol[4]. Posee herramientas para la protección ante ataques piratas, fallos eléctricos y problemas de red. (CCN-STIC-480, 2010)

1.2.3 Módulo Interfaz Hombre – Máquina del SCADA SAINUX

El HMI del sistema SCADA SAINUX está compuesto por dos ambientes de trabajo, el de Configuración o Editor que permite administrar la configuración del proyecto y sus recursos; y el de Ejecución o Visualizador que permite ejercer control y supervisar los recursos que han sido configurados con anterioridad en el ambiente de configuración. A continuación se describen algunas de las funcionalidades que permiten estos ambientes:

- El entorno de configuración permite crear un proyecto y administrar la configuración de sus recursos (Seguridad, Histórico, Adquisición y HMI). Posibilita la creación y el diseño de los despliegues que se visualizarán en el ambiente de Ejecución utilizando para ello los componentes necesarios para representar la información al operador que son asociados a su vez con los puntos(valores configurados para adquirir información del campo) que representan, dando la posibilidad el sistema de modificar todos los recursos que se definan. Permite la creación y edición de reportes, la definición de los usuarios bajo un esquema de privilegios identificado y la configuración de los grupos donde se almacenarán los datos del sistema.
- El entorno de visualización permite adquirir datos del campo en forma continua, almacenar y mostrar información en forma de sumarios y gráfica, ejercer control y supervisión de los despliegues configurados en el ambiente de configuración, así como visualizar las alarmas que se detecten y afecten el comportamiento del proceso supervisado. Permite la autenticación de usuarios en el

[3] Autenticarse: Acción que permite verificar la identidad digital de un usuario.

[4] Rol: Responsabilidad asumida por una persona en el equipo de trabajo.

sistema. Crea una sesión para cada usuario autenticado, con sus respectivos privilegios sobre la herramienta y maneja los tiempos de expiración de la sesión.

Como se observa, los visualizadores de hoy en día ofrecen a los operadores las más sofisticadas técnicas de supervisión y control, brindado la posibilidad de simular las operaciones que se realizan en una planta industrial con los gráficos sinópticos, haciendo sumamente intuitivo la operación de una planta específica.

1.3 Componentes Gráficos del HMI SAINUX

En el SCADA SAINUX además de la figuras geométricas básicas (rectángulo, línea, círculo, etc.) son necesarios una serie de componentes gráficos que le permitan al usuario obtener aplicaciones para la supervisión y el control de los procesos industriales .Estos componentes se van agrupar en paletas de acuerdo a la función que pueden realizar dentro de un despliegue.

- ✓ **Gráficos:** Son aquellos componentes que permiten representar el estado de los valores de una variable para realizar un análisis estadístico, entre ellos se encuentran el gráfico XY y el gráfico de barra.
- ✓ **Miscelánea:** Reúne elementos como la imagen, el diodo emisor de luz (LED) y el ventilador/extractor.
- ✓ **Básicos:** Formado por figuras simples como líneas, flechas, elipses, rectángulos y polígonos.
- ✓ **Medidores:** Conformado por todos aquellos elementos que posibilitan la visualización del estado de los valores que obtienen las variables durante todo el proceso, entre los que se encuentran el visualizador y el medidor vertical.
- ✓ **Controles**: Compuesto por componentes que permiten ejercer control sobre el sistema tales como botón pulsable, botón de estado y lista desplegable.

1.4 Interruptores y selectores

En la actualidad los interruptores y selectores son empleados en: refinerías, industrias petroquímicas, procesos de exploración, plantas de extracción de petróleo y gas, tratamiento de gas industrias productos inflamables, ambientes marinos altamente corrosivos, uso interior y exterior etc.

Los interruptores y selectores son comúnmente utilizados en circuitos eléctricos, los mismos se pueden representar en varios tipos de diagramas o esquemas como son:

- **Representación multifilar**

  Se representan todos los conductores con sus conexiones a los distintos elementos que intervienen en un circuito o en una instalación eléctrica. Son los que se utilizan normalmente en los montajes de los circuitos eléctricos. (Barcojo)

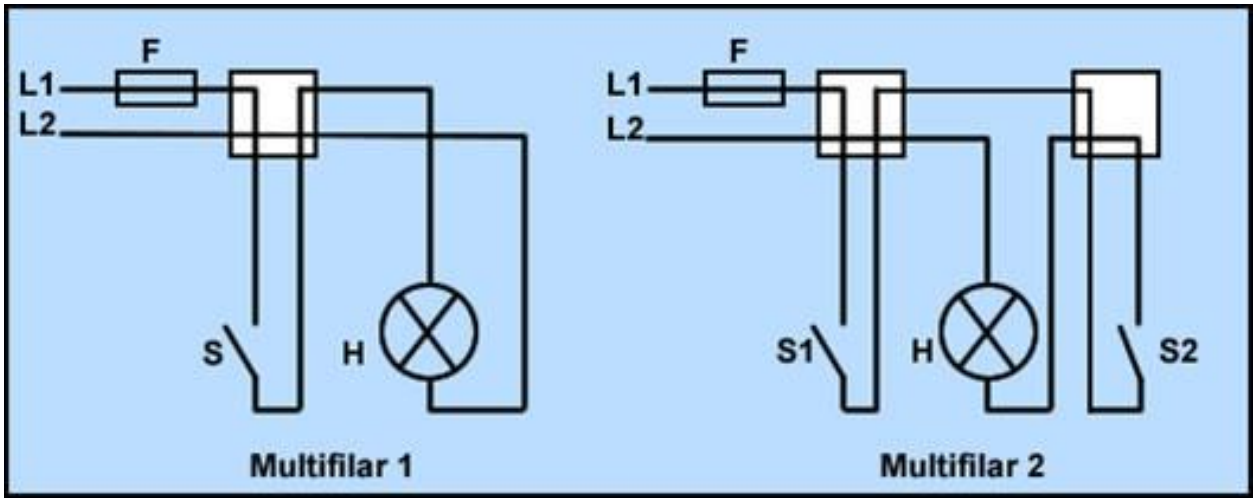


Ilustración 2: Esquema multifilar de un circuito eléctrico.

- **Representación unifilar**

  Un esquema o diagrama unifilar es una representación gráfica de una instalación eléctrica o de parte de ella. El esquema unifilar se distingue de otros tipos de esquemas eléctricos en que el conjunto de

conductores de un circuito se representa mediante una única línea, independientemente de la cantidad de dichos conductores. Típicamente el esquema unifilar tiene una estructura de árbol. (Barcojo)

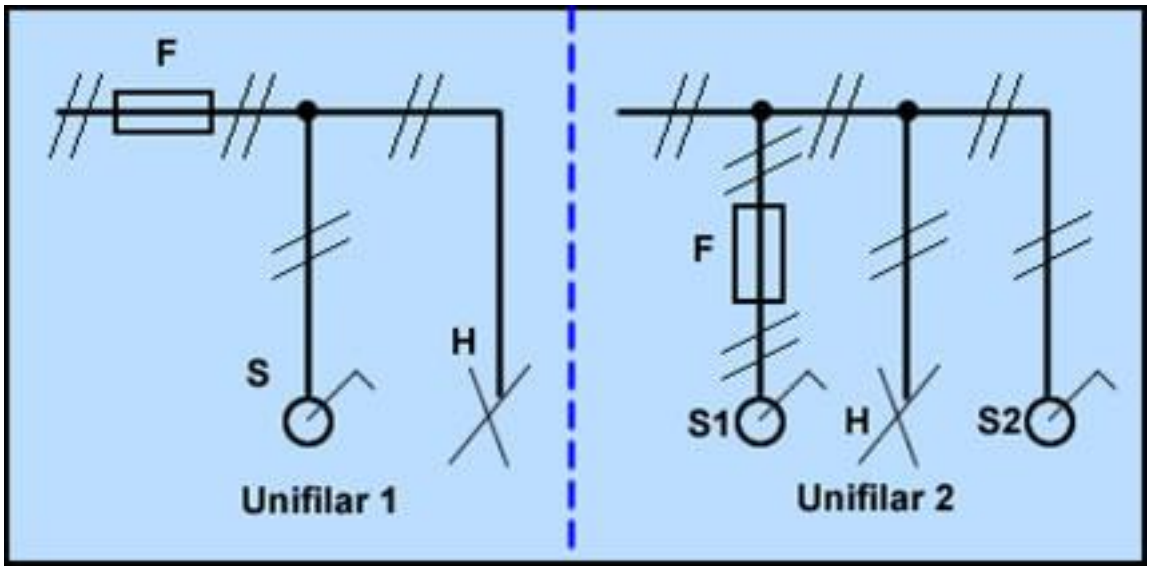


Ilustración 3: Esquema unifilar de un circuito eléctrico.

- **Representación funcional**

  Un esquema funcional muestra las funciones del sistema de forma gráfica y con algunas aclaraciones en texto. En otras palabras, muestra los procesos que se llevan a cabo en un sistema. Por eso también es llamado modelo de procesos. Debe mostrar también las entradas al sistema (de datos, materiales, energía, etc) y las salidas del sistema (productos, datos, energía, etc.). (Barcojo)

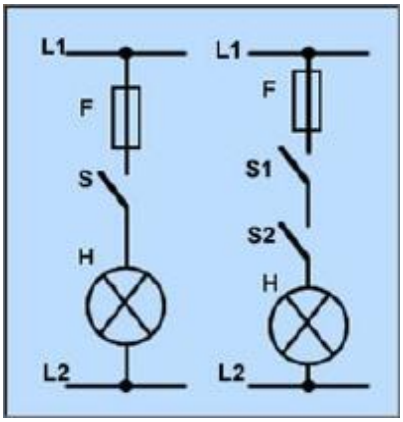


Ilustración 4: Esquema funcional de un circuito eléctrico.

1.4.1 Interruptor

Un interruptor independientemente en el diagrama o esquemas representado, su simbología es la misma la cual se muestra en la siguiente imagen:

Un interruptor eléctrico es un dispositivo que permite desviar o interrumpir el curso de una corriente eléctrica. En el mundo moderno sus tipos y aplicaciones son innumerables, van desde un simple interruptor que apaga o enciende una bombilla, hasta un complicado selector de transferencia automático de múltiples capas, controlado por computadora.

Su expresión más sencilla consiste en dos contactos de metal inoxidable y el actuante. Los contactos, normalmente separados, se unen mediante un actuante para permitir que la corriente circule. El actuante es la parte móvil que en una de sus posiciones hace presión sobre los contactos para mantenerlos unidos. (1998-2016)

1.4.2 Selector

Un selector independientemente en el diagrama o esquemas representado, su simbología es la misma la cual se muestra en la siguiente imagen:

Un selector tiene varias posiciones de accionamiento manual. Un selector por lo general, está ajustado por un pomo o mango, y puede tener retenes para mantener en una posición dada. Se utiliza, por ejemplo, en los dispositivos o instrumentos con múltiples funciones, rangos, o modos de funcionamiento. Un selector de este tipo es generalmente rotatorio. También se llama conmutador. Además es un dispositivo eléctrico o electrónico que permite modificar el camino que deben seguir los electrones. Son típicos los manuales, como los utilizados en las viviendas y en dispositivos eléctricos, y los que poseen algunos componentes eléctricos o electrónicos como el relé. Se asemejan a los interruptores en su forma exterior, pero los selectores a la vez que desconectan un circuito, conectan otro (1998-2016)

1.5 Selección de herramientas y tecnologías

La selección de la tecnología a utilizar se realiza mediante un estudio de las diferentes opciones existentes. Parte de este estudio, fue realizado en años anteriores, por parte de la dirección del proyecto, por lo que el desarrollo del presente trabajo, sigue la línea trazada. Dos ejemplos son la selección del entorno de desarrollo integrado y el sistema operativo a utilizar.

1.5.1 Gráficos Vectoriales Escalables (SVG)

Los gráficos vectoriales escalables (SVG, por sus siglas en inglés) son un lenguaje abierto que permite crear gráficos vectoriales basados en XML tanto estáticos como animados. A pesar de ser un lenguaje vectorial permite crear imágenes complejas, lo cual garantiza su selección para el diseño de componentes. (ALEGSA, 1998-2016)

Es un estándar de la W3C (World Wide Web Consortium, en inglés) que define una gramática XML para describir gráficos de dos dimensiones. Cualquier programa, tales como un navegador que reconozca XML

pueden mostrar la imagen usando la información proporcionada en el formato SVG. La parte del término "gráficos escalable" hace hincapié en que las imágenes SVG se pueden ampliar fácilmente, a diferencia de las imágenes especificadas en gráficos de mapa de bits. Así, el formato SVG permite la visualización de una imagen sobre una pantalla de ordenador de cualquier tamaño y resolución, ya sea una pequeña pantalla LCD en un teléfono celular o una gran pantalla CRT en una estación de trabajo. Además de la facilidad de la reducción del tamaño y la ampliación, SVG permite que los textos dentro de imágenes para ser reconocidas como tales, por lo que el texto puede ser localizado por un motor de búsqueda y fácilmente traducible a otros idiomas. Otra ventaja potencial sobre los formatos de imagen estándar de la web (GIF y JPEG) es su tamaño, en comparación con una imagen de mapa de bits, una imagen SVG puede ser mucho menor lo que disminuye el tiempo de carga. Los gráficos SVG pueden ser interactivos y dinámicos, con animaciones que pueden ser definidas y activas de modo declarativo o por medio de scripts. (ALEGSA, 1998-2016)

1.5.2 Editor Gráfico

Inkscape es un editor de gráficos vectoriales de código abierto, con capacidades similares a Illustrator. Es un ambiente ideal para editar los gráficos desarrollados en la aplicación, en este caso la creación del componente gráfico de control de tipo Interruptor y Selector, generados con programas que tengan la opción de exportar en formatos .eps, .pdf, .svg o algún otro formato vectorial. (Mad)

Es un ambiente ideal para editar los gráficos desarrollados en la aplicación, en este caso la creación de los componentes gráficos de control de tipo Interruptor y Selector. El Inkscape también soporta meta-datos Creative Commons, edición de nodos, capas, operaciones complejas con trazos, vectorización de archivos gráficos, texto en trazos, alineación de textos, edición de XML directo y mucho más. Puede importar formatos como Postscript, EPS, JPEG, PNG, y TIFF y exporta PNG así como muchos formatos basados en vectores. (Mad)

1.5.3 Lenguaje de programación

El lenguaje de programación será el C++ por ser en lenguaje en el cual está desarrollado SCADA SAINUX y así lograr una mejor compatibilidad de la solución con el SAINUX. C++ está considerado por muchos como

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

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

Existen gran cantidad de bibliotecas implementadas que permiten la reutilización del código fuente, disminuyen el tiempo de desarrollo y aumenta la productividad. La utilización de plantillas también permite la reutilización del código y la programación genérica. Además de ser el lenguaje nativo sobre el cual está desarrollado Qt. (Oliag)

1.5.4 Marco de trabajo

Para la representación de interruptores y selectores se hace necesario la utilización de un marco de trabajo (framework, en inglés), el cual se utiliza de base para la organización y desarrollo del software. Un marco de trabajo puede incluir soporte de programas, bibliotecas, lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar los diferentes componentes de un proyecto. (CORPORATION, 2008)

El marco de trabajo utilizado es Qt en su versión 4.8.2, el cual es multiplataforma para el desarrollo de aplicaciones, las cuales pueden ser con o sin interfaz gráfica. Para el desarrollo de este trabajo se hace uso de esta biblioteca debido a que utiliza el lenguaje de programación C++ de forma nativa y es precisa para realizar todas las interfaces gráficas necesarias. (CORPORATION, 2008)

Qt incluye un amplio conjunto de componentes gráficos que proporcionan las funcionalidades estándar de interfaz gráfica de usuario, introduce una innovadora alternativa para la comunicación entre objetos, llamados señales y ranuras. Puede soportar las funcionalidades de interfaz de usuario que requieren las aplicaciones modernas, tales como menús, menús contextuales, arrastrar, soltar, y barras de herramientas.

Propone el patrón de diseño modelo/vista para la representación gráfica de los datos, con la separación de las funcionalidades introducidas por esta arquitectura. El marco de trabajo Qt ofrece a los desarrolladores una mayor flexibilidad para personalizar la presentación de elementos y proporciona una interfaz de modelo estándar que permite una amplia gama de fuentes de datos. (CORPORATION, 2008)

1.5.5 Entorno de Desarrollo Integrado (IDE)

Para el desarrollo de este producto se requiere de un entorno de desarrollo integrado (IDE, por sus siglas en inglés), el cual puede denominarse como un entorno de programación que consiste en un editor de código y un compilador. (Raydel Raúl Viñolo Sosa, 2012)

El IDE seleccionado para el desarrollo de la aplicación es Qt Creator en su versión 2.5.0. Qt Creator es un IDE multiplataforma para el desarrollo de aplicaciones que pueden o no tener interfaz gráfica. Este se centra en proporcionar características que ayudan a los nuevos usuarios del IDE a aprender y comenzar a desarrollar rápidamente. (Raydel Raúl Viñolo Sosa, 2012)

Existen otras características importantes que presenta este IDE como la disponibilidad de código fuente, la excelente documentación organizada que provee en el QtAssistant y un editor para el diseño de formularios denominado QtDesigner. (Raydel Raúl Viñolo Sosa, 2012)

Qt Creator cuenta con:

- ✓ Un editor de código con soporte para C++.
- ✓ Herramientas para la rápida navegación por el código.
- ✓ Resaltado de sintaxis y auto-completado de código.
- ✓ Control estático de código y estilo a medida.
- ✓ Soporte para refactorización de código.
- ✓ Paréntesis coincidentes y modos de selección.

1.5.6 Lenguaje Unificado de Modelado (UML)

Se selecciona como lenguaje el UML, debido a que se emplea para visualizar, especificar, construir y documentar los artefactos de la solución propuesta. Es un lenguaje estándar, fácil de aprender, y ofrece un conjunto de notaciones y diagramas estándar para mostrar la solución propuesta desde varias perspectivas. Está especialmente diseñado para apoyar un estilo de desarrollo iterativo e incremental y presenta tecnología orientada a objetos. Ayuda al usuario a entender la realidad de la tecnología y tiene una notación gráfica muy expresiva que permite representar todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, hasta la implementación y configuración con los diagramas de despliegue. (Rumbaugh, 2005)

1.5.7 Visual Paradigm

Una herramienta CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadoras) es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases. Las herramientas CASE abarcan todos los pasos del desarrollo del software, y también aquellas actividades generales que se aplican a lo largo del proceso. Además facilitan el uso de las distintas metodologías propias de la ingeniería del software incluyendo AUP. (Paradigm, 2013)

Se seleccionó como herramienta CASE el Visual Paradigm porque soporta el ciclo de vida completo del desarrollo de software, permite modelar los diagramas de la solución propuesta, realizar ingeniería tanto directa como inversa, a partir de un modelo relacional en SQL Server, MySQL, PostgreSQL, es capaz de desplegar todas las clases asociadas a las tablas (siguiendo el patrón de diseño Una Clase-Una Tabla). Soporta múltiples usuarios trabajando sobre el mismo proyecto, permite el control de versiones y generar la documentación automáticamente en formatos como web o PDF. Además este software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. A su vez es una herramienta libre y multiplataforma. (Paradigm, 2013)

1.5.8 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. Pueden ser exclusivamente de software libre, o también incorporar aplicaciones o controladores propietarios. Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU (acrónimo que significa GNU No es Unix); de ahí el nombre: GNU/Linux. (SPI, 1997-2015)

La distribución seleccionada es Debian, específicamente la versión 7.0 Squeeze. Debian o Proyecto Debian es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en software libre. El sistema se encuentra pre compilado, empaquetado y en un formato .deb para múltiples arquitecturas de computadoras y para varios núcleos. (SPI, 1997-2015)

Nació como una apuesta por separar en sus versiones el software libre del software no libre. El modelo de desarrollo del proyecto es ajeno a motivos empresariales o comerciales, siendo llevado adelante por los propios usuarios, aunque cuenta con el apoyo de varias empresas en forma de infraestructuras. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuirlo comercialmente mientras se respete su licencia. (SPI, 1997-2015)

1.5.9 Herramienta de control de versiones

La herramienta de control de versiones que se utilizará es Apache Subversión (SVN) en su versión 1.6. SVN es una herramienta de control de versiones basada en un repositorio cuyo funcionamiento se asemeja al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache. (CollabNet, 2000)

Es seleccionada esta herramienta para el desarrollo de la aplicación ya que utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones, optimizando así al máximo el uso de espacio en disco. SVN permite al usuario crear, copiar y borrar carpetas con la misma facilidad con la que lo haría si estuviese en su disco duro local. Dada su flexibilidad, es necesaria la aplicación de buenas prácticas para llevar a cabo una correcta gestión de las versiones del software generado. (CollabNet, 2000)

Subversión puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Debido a que la investigación se encuentra bajo el control de versiones, no existe razón para que la calidad de la misma sea afectada. Esta herramienta permite recuperar versiones antiguas y ver el historial de cambios de un sistema de archivos. (CollabNet, 2000)

1.6 Metodología de desarrollo de software

Todo desarrollo de software es muy riesgoso además de ser difícil de controlar, y si no se emplea una metodología que guíe este proceso, los resultados que se obtendrán serán clientes insatisfechos y desarrolladores aún más descontentos. Hoy día en todo el mundo, se proponen diferentes metodologías en dependencia del tiempo de vida y la complejidad del proyecto que se vaya a desarrollar. Una Metodología de Desarrollo de Software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Las metodologías, dadas sus características, se enmarcan en dos grandes grupos, las metodologías tradicionales o pesadas" y las metodologías ágiles. La diferencia más notable entre estos dos grupos es que mientras las metodologías tradicionales intentan obtener los resultados apoyándose principalmente en la documentación ordenada y persiguen la consecución de un proyecto cuya planificación está bien definida, las metodologías ágiles tienen como base de sus resultados la comunicación e interacción directa con todos los usuarios involucrados en el proceso además están pensadas para pequeños grupos de personas. (2008)

1.6.1 Metodología AUP-UCI

En la UCI actualmente existen 14 centros productivos, cada uno de estos centros se dedica al desarrollo de software y/o servicios empleando el uso de diferentes metodologías de desarrollo entre robustas y ágiles. A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad. Las diferencias entre estas metodologías no radica únicamente en los productos de trabajos

que proponen o en sus roles, sino en su forma de planificar el proyecto y realizar las estimaciones del tiempo. Factor determinante en la culminación exitosa de todo desarrollo de software. Para lograr erradicar los problemas detectados, se decide escoger una metodología para ser adaptada a lo que ya la Universidad ha estado proponiendo como ciclo de vida de los proyectos, sin alejarse de lo que hasta el momento se ha trabajado e introducir la menor cantidad de cambios posibles. (Sánchez)

Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3. Se logra un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. De ahí que se escoge la utilización de dicha metodología para guiar el proceso de desarrollo de la solución propuesta, cumpliendo con los estándares de calidad definidos por la universidad. (Sánchez)

Esta 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. Se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. Describe un enfoque simple del desarrollo del software usando técnicas y conceptos ágiles. Algunas técnicas usadas por AUP incluyen el desarrollo orientado a pruebas, modelado y gestión de cambios ágiles y refactorización de base de datos para mejorar la productividad. (Sánchez)

1.6.2 Fases de AUP-UCI

1. Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.

3. Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. Transición: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

1.6.3 Disciplinas de APU-UCI

AUP define 7 disciplinas (4 ingenieriles y 3 de gestión de proyectos), las disciplinas son:

1. **Modelo.** El objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se abordan en el proyecto, y determinar una solución viable para resolver el problema de dominio. Agrupa los flujos de trabajos de Modelado de negocio, Requisitos y Análisis y Diseño.
2. **Implementación.** El objetivo de esta disciplina es transformar su modelo (s) en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas.
3. **Prueba.** El objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificando que se cumplan los requisitos.
4. **Despliegue.** El objetivo de esta disciplina es la prestación y ejecución del sistema y que el mismo este a disposición de los usuarios finales.
5. **Gestión de configuración.** El objetivo de esta disciplina es la gestión de acceso a herramientas de su proyecto. Esto incluye no sólo el seguimiento de las versiones con el tiempo, sino también el control y gestión del cambio para ellos.
6. **Gestión de proyectos.** El objetivo de esta disciplina es dirigir las actividades que se lleva a cabo en el proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, etc), coordinación con el personal y los sistemas fuera del alcance del proyecto para asegurarse de que es entregado a tiempo y dentro del presupuesto.
7. **Entorno.** El objetivo de esta disciplina es apoyar el resto de los esfuerzos por garantizar que el proceso sea el adecuado, la orientación (normas y directrices), y herramientas (hardware, software, etc) estén disponibles para el equipo según sea necesario.

1.7 Conclusiones parciales

En este capítulo se realizó un esbozo de las principales características de los sistemas SCADA, haciendo énfasis en el módulo HMI del SCADA SAINUX específicamente en los componentes gráficos. Se expusieron las tecnologías a usar en la solución como son el SVG e Inkscape para el diseño gráfico de los componentes, UML y el Visual Paradigm como herramienta CASE ya que se emplean para visualizar, construir y documentar los artefactos del sistema, C++ como lenguaje de programación empleado en el desarrollo del producto, Qt, marco de trabajo utilizado para la construcción de las interfaces gráficas, QtCreator, entorno de desarrollo integrado usado para la implementación de los componentes, las cuales se corresponden con las utilizadas en el desarrollo del HMI del SCADA SAINUX para lograr una mejor integración entre la solución propuesta en este trabajo y el módulo HMI. Como metodología de desarrollo de software se escogió AUP-UCI por ser la definida tanto en la universidad como en el centro para guiar el proceso de desarrollo de software.

CAPÍTULO 2

Análisis y diseño de la solución

2 Introducción

El presente capítulo se describirá las actividades que se realizaran en los procesos de análisis y diseño de la solución propuesta; proceso que será guiado por la metodología de desarrollo AUP-UCI. En el mismo se realizará el modelo de dominio donde se describirán las entidades que intervienen, con el objetivo de facilitar la comprensión de los principales conceptos que se utilizarán en el proceso de negocio identificado. Se expondrá los artefactos más importantes que describen el flujo normal de eventos que ocurren en el sistema, se realizará una descripción de la solución propuesta, planteándose los requisitos funcionales y no funcionales, el diagrama de casos de uso del sistema y la descripción de los casos de uso críticos.

2.1 Modelo de Dominio

El modelo de dominio es “la representación visual de los conceptos u objetos más importantes de un negocio, sus características y las relaciones entre dichos conceptos. Es el mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes. Es un diccionario visual del dominio del problema.” (S. Pressman, 2001)

A continuación se presentan las clases del dominio perteneciente a la solución según el siguiente modelo de dominio.

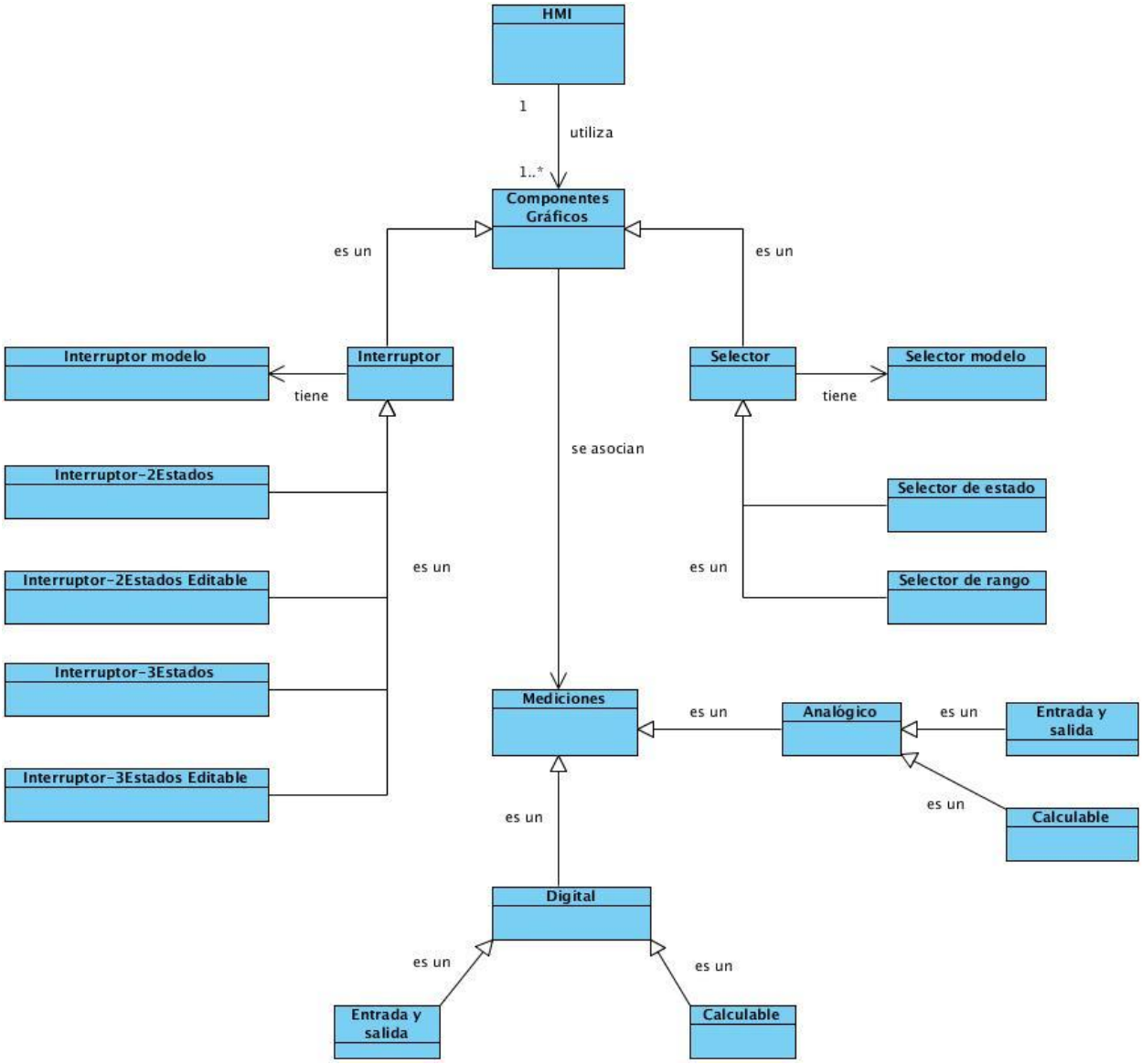


Ilustración 5: Modelo de dominio.

El modelo de dominio los conceptos que lo integran:

- ✓ **HMI:** Interfaz encargada de mostrar en un ordenador los procesos que ocurren en el campo.

- ✓ **Componente gráfico:** Representa de manera visual una serie de datos por medio de figuras o signos.
- ✓ **Mediciones:** Entidad encargada de medir el fluido que pasa por el componente.
- ✓ **Digital:** Su objetivo es representar mediante un reloj digital la medición de flujo presente en el componente.
- ✓ **Analógica:** Encargada de representar mediante un reloj Analógico la medición de flujo presente en la red.
- ✓ **Interruptor:** Entidad base de la cual heredan cada uno de los componentes que serán visualizados de tipo interruptor.
- ✓ **Selector:** Entidad base de la cual heredan cada uno de los componentes que serán visualizados de tipo selector.
- ✓ **Interruptor Modelo:** Entidad Modelo donde se guarda la configuración que tendrá el interruptor y brinda la información necesaria para que un componente gráfico pueda visualizarse.
- ✓ **Selector Modelo:** Entidad Modelo donde se guarda la configuración que tendrá el selector y brinda la información necesaria para que un componente gráfico pueda visualizarse.
- ✓ **Interruptor -2Estados:** Encargada de representar el interruptor en forma de dos estados posibles.
- ✓ **Interruptor -2Estados Editable:** Encargada de representar el interruptor en forma de dos estados editable posibles.
- ✓ **Interruptor -3Estados:** Encargada de representar el interruptor en forma de tres estados posibles.
- ✓ **Interruptor -3Estados Editable:** Encargada de representar el interruptor en forma de tres estados editable posibles.
- ✓ **Selector de estado:** Entidad encargada de representar el estado del selector.
- ✓ **Selector de rango:** Entidad encargada de representar el rango del selector.

2.2 Análisis de la solución

La presente investigación está orientada al desarrollo de componentes gráficos de control de tipo Interruptor y Selector, capaz de ampliar la gama de componentes, en la biblioteca de componentes gráficos del módulo HMI (Interfaz Hombre Máquina) del SCADA SAINUX. La solución debe ser capaz de realizar transformaciones al componente gráfico sin que la misma pierda calidad de imagen al ser redimensionada con la utilización de la herramienta SVG, así como dicho componente de control puede tomar varios valores de estados, e incluso pueden ser editados por el usuario final a la forma que le sea más útil en la industria.

2.3 Especificación de requisitos

Conocer los requisitos del sistema que se va a desarrollar es el primer paso en el desarrollo de su implementación, de esta manera se reducirá el número de cambios que habrá que realizar en el producto debido a los cambios en sus requisitos. A partir del modelo del dominio presentado se realizó el levantamiento de los requisitos, los cuales están divididos en dos tipos: los funcionales y los no funcionales. A continuación se especifican los requisitos definidos para el desarrollo del sistema propuesto.

2.3.1 Requisitos funcionales

Los requisitos funcionales (RF) del sistema propuesto se identificaron de acuerdo a las capacidades o condiciones que este debe cumplir. Los mismos expresan una especificación detallada de cómo reacciona el sistema a una entrada particular y cómo se comporta ante situaciones particulares, es decir: sus entradas, salidas y excepciones. (Pressman, 2001)

Teniendo en cuenta las funcionalidades que el sistema debe cumplir, se especificaron los siguientes requisitos:

Para el ambiente de **Edición–Configuración** se determinaron los siguientes requisitos funcionales, comunes para ambos grupos de componentes:

RF 1. El sistema debe permitir asociar medición al componente. Estas mediciones pueden ser digital o analógicas según el tipo de componente.

RF 2. El sistema debe permitir configurar correspondencia entre los valores discretos de la variable con los colores deseados por cada valor del componente.

RF 3. El sistema debe permitir definir nombre del componente

RF 4. El sistema debe permitir configurar la dimensión del componente.

RF 5. El sistema debe permitir configurar la opacidad del objeto del componente.

RF 6. El sistema debe permitir definir la descripción del componente

RF 7. El sistema debe permitir configurar posición del componente

RF 8. El sistema debe permitir configurar color que representa el estado indefinido

RF 9. El sistema debe permitir configurar color del indicador

RF 10. El sistema debe permitir configurar color de borde

RF 11. El sistema debe permitir editar correspondencia entre los valores discreto de la medición, con los colores deseados por cada valor.

Para el selector de rango:

RF 12. .El sistema debe permitir mostrar y ocultar el valor

Para el selector de 16 estados:

RF 13. El sistema debe permitir mostrar y ocultar la escala

Para el ambiente de visualización:

RF 14. . El sistema debe permitir la ejecución de comando de escritura sobre las mediciones asociada al componente.

RF 15. El sistema debe permitir representar en el componente, el estado en el que se encuentra la medición asociada.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) son las propiedades o cualidades que el sistema debe tener, para que sea un producto atractivo, usable, rápido y confiable. (Pressman, 2001)

Existen múltiples categorías para clasificar los requisitos no funcionales, siendo las siguientes las más representativas acumulando un total de 7 RNF para el sistema propuesto.

Requerimientos de software

- El sistema debe funcionar en el Sistema Operativo Debian, en su versión 7.

Requerimientos de Hardware

El sistema debe ser ejecutado en computadoras que tengan como requerimientos mínimos.

- Microprocesador: dual core a 1.5 GHz.
- RAM: 2 GB

Restricciones en el diseño y la implementación

- El sistema se implementará utilizando el lenguaje de programación C++, en su versión del 98.
- El sistema se utilizará el IDE Qt Creator en su versión 2.5.0.
- El sistema se empleará el marco de trabajo Qt, en su versión 4.8.2.

Requerimientos de usabilidad

- El sistema debe proporcionar una interfaz gráfica intuitiva y fácil de entender.

Requerimientos de rendimiento

- En el ambiente de despliegue solo se admiten hasta 100 componentes.

2.4 Historias de Usuarios

Es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. (Cardozzo, 2014)

Historia de Usuario	
Número: 1	**Nombre de Historia:** El sistema debe permitir asociar medición al componente.
Programador: Javier Bravo Calzado	**Iteración Asignada:** 1

Prioridad en Negocio: Alta	**Tiempo estimado:** 0.8 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	**Tiempo real:** 0.8 semana
Descripción: Su objetivo es asignar una medición al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde podemos seleccionar la medición deseada.	

Tabla 1: Historia de Usuario 1

La descripción de las demás historias de usuario se encuentra en el Anexo 1

2.4.1 Desarrollo de Iteraciones

Durante el transcurso de las iteraciones se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de ser necesario. Como parte de este plan, se descomponen las HU en tareas de programación o ingeniería, asignando a un equipo de desarrollo (o persona), responsable de su implementación. Teniendo en cuenta la planificación realizada en el capítulo anterior, se llevó a cabo el desarrollo del sistema en seis iteraciones, obteniéndose un mecanismo capaz de solucionar el problema planteado en este trabajo. A continuación se detallan cada una de las iteraciones.

2.4.2 Iteración I

Historias de Usuarios	Tiempo de implementación (semanas)	
	Estimación	Real
Asociar medición al componente	0.8	0.8

Configurar correspondencia entre los valores discretos de l variable con los colores deseados por cada valor del componente.	0.8	0.8
Definir nombre del componente	0.2	0.2
Configurar la dimensión del componente	0.6	0.6
Configurar la opacidad del objeto del componente	0.6	0.6
Definir la descripción del componente	0.4	0.4
Configurar posición del componente	0.4	0.4
Configurar color que representa el estado indefinido	0.8	0.8
Configurar color del indicador	0.8	0.8

Tabla 2: Tabla de Iteraciones

2.4.3 Iteración II

Historias de Usuarios	**Tiempo de implementación (semanas)**	
	Estimación	**Real**
Configurar color de borde	0.8	0.8
Editar correspondencia entre los valores discreto de la medición, con los colores deseados por cada valor	0.8	0.8
Mostrar y ocultar el valor	0.8	0.8

Mostrar y ocultar la escala	0.8	0.8
Permitir ejecución de comando de escritura sobre las mediciones asociada al componente	0.8	0.8
Representar en el componente, el estado en el que se encuentra la medición asociada	0.8	0.8

Tabla 3: Tabla de Iteraciones

2.5 Diagrama de Paquetes

Un diagrama de paquetes en el Lenguaje Unificado de Modelado, representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones. (2007) La solución propuesta esta agrupada en el paquete HMI, el cual se subdivide en dos paquetes, SAINUX Interface y HMI Base. La primera agrupa todas las entidades que son utilizadas para la interface del HMI en SAINUX, dentro del sub paquete Extension se encuentra el paquete Graphics Plugins en el cual se encuentra las entidades que representan las paletas de componentes pero para la interfaz de SAINUX, en la segunda se encuentra el paquete HMI Base, es donde se encuentran todas las entidades y paquetes bases para el desarrollo del módulo HMI, independiente de su interfaz. Dentro del paquete Extension se encuentra el sub paquete Graphics Plugins el cual contiene las entidades que representa las paletas de componentes que están agrupada en sub paquetes de acuerdo a la clasificación del componente en este caso Interruptor y Selector. En el paquete Graphics se encuentra el sub paquete Graphic donde se agrupan las entidades que representan a los selectores e interruptores que de acuerdo a su función son ubicadas en los paquetes View y Model.

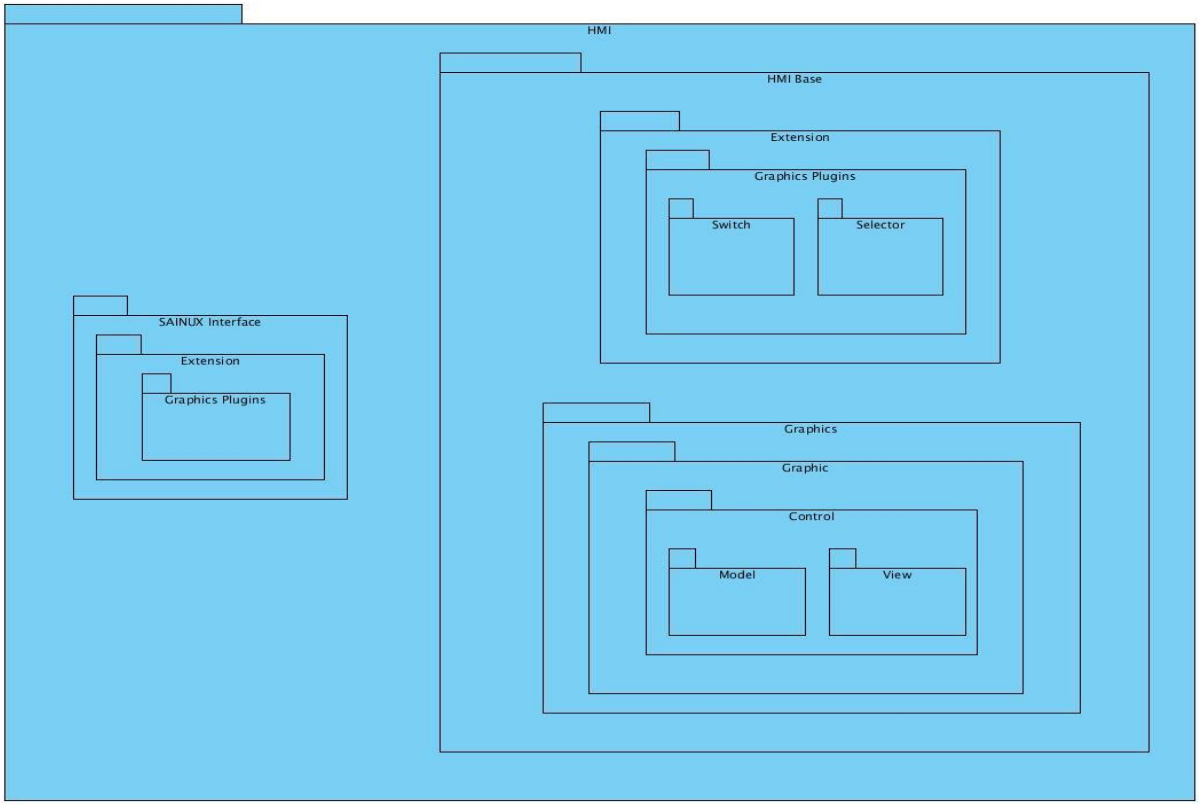


Ilustración 6: Diagrama de Paquetes.

2.6 Diseño de clases

El diseño es un proceso de resolución de problemas cuyo objetivo es encontrar y describir una forma:

- Para implementar los requisitos funcionales del sistema.
- Respetando las restricciones impuestas por los requisitos no funcionales.
- Ajustándose a los principios generales de calidad.

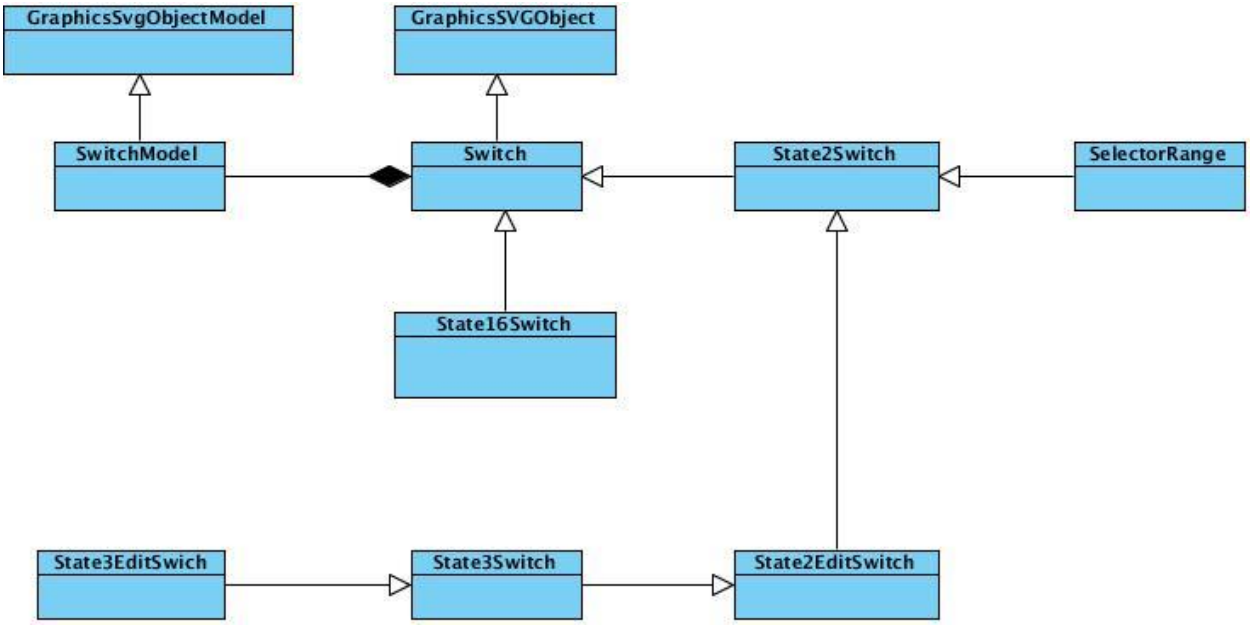


Ilustración 7: Diseño de Clases de Componentes Gráficos

La representación de los diseños de clases se encuentra en el Anexo 2

El diseño de clase los conceptos que lo integran:

- **GraphicsSvgObjectModel**: Entidad base para todas las entidades de tipo modelo asociada a componentes gráficos que en su representación utilizan SVG.
- **SwitchModel**: Entidad modelo para almacenar todos los datos referentes a los interruptores.
- **GraphicsSvgObject**: Entidad base para todos los componentes gráficos que utilizan en su representación SVG.
- **Switch**: Entidad base para el desarrollo de cualquier dispositivo de tipo interruptor
- **State16Switch**: Entidad base para el desarrollo de cualquier dispositivo de tipo interruptor de 16 estados.
- **State2Switch**: Entidad base para el desarrollo de cualquier dispositivo de tipo interruptor de 2 estados
- **SelectorRange**: Entidad modelo para la representación de cualquier dispositivo de tipo selector de rango.

- **State2EditSwitch**: Entidad base para el desarrollo de cualquier dispositivo de tipo interruptor de 2 estados editable.
- **State3Switch**: Entidad base para el desarrollo de cualquier dispositivo de tipo interruptor de 3 estados.
- **State3EditSwitch**: Entidad base para el desarrollo de cualquier dispositivo de tipo interruptor de 3 estados editable.

2.7 Diagrama de secuencia

Un diagrama se secuencia describe la dinámica del sistema, la cual resulta difícil de modelar en un único diagrama. También describe las interacciones entre un grupo de objetos mostrando de forma secuencial el envío de mensajes entre ellos, así como los flujos de datos intercambiados durante el envío. Durante la recepción de un mensaje, los objetos se vuelven activos ejecutan el método del mismo nombre; por lo que un envío representa una llamada a un método. (Laurent DEBRAUWER, 2013)

A continuación se muestran algunos de los diagramas diseñados para la solución propuesta:

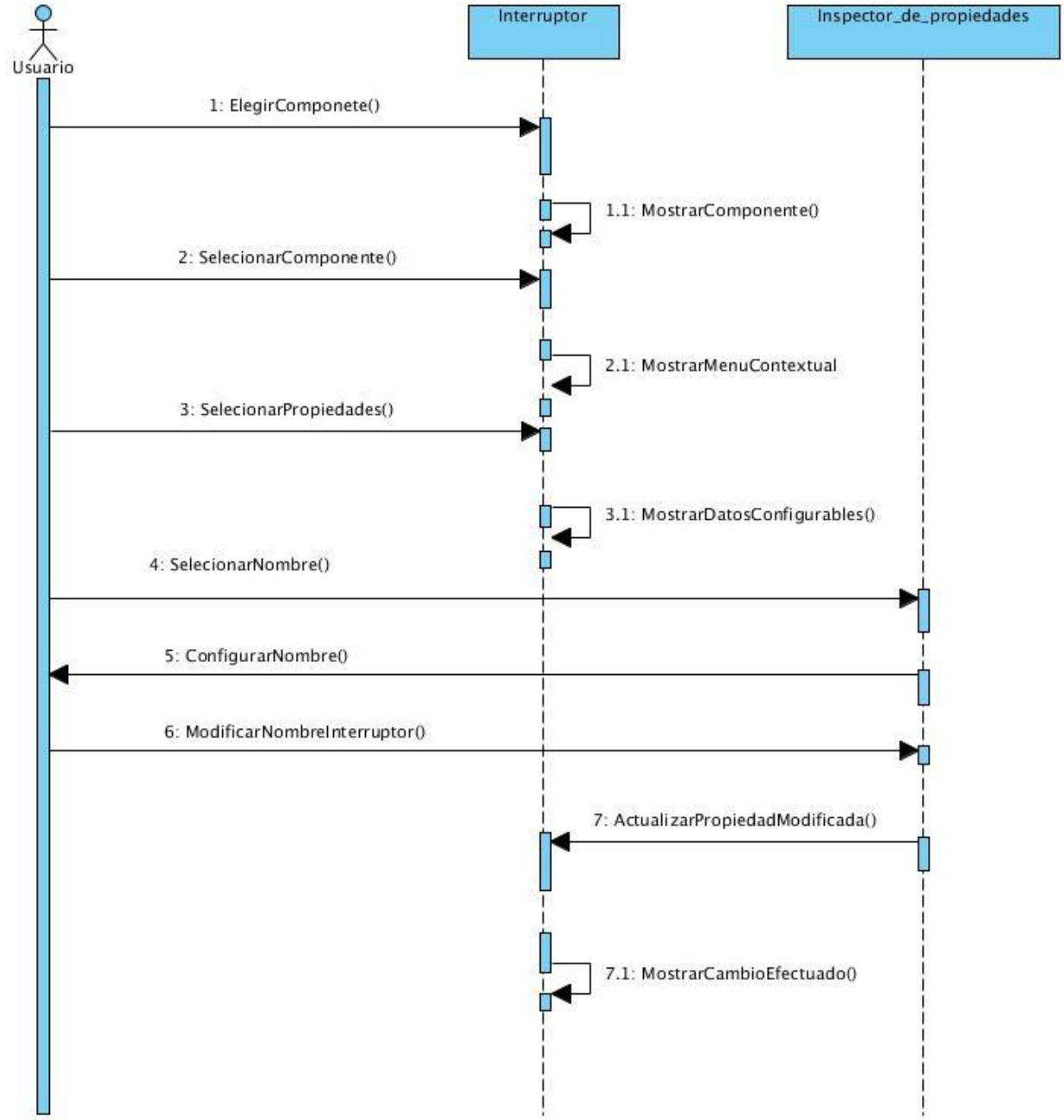


Ilustración 8: Diagrama de secuencia

Los demás diagramas se encuentran en el ANEXO 3

2.8 Patrón de Arquitectura

La selección de un patrón arquitectónico es una decisión fundamental de diseño en el desarrollo de un sistema de software ya que provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. (Jacobson Ivar, 1999)

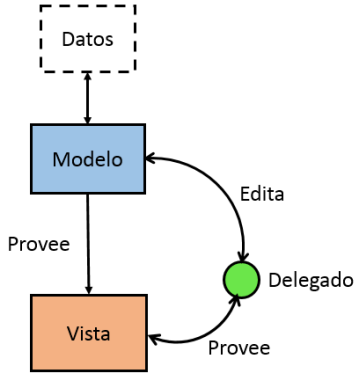


Ilustración 9: Patrón de Arquitectura de Software Modelo-Vista.

La arquitectura Modelo/Vista es una adaptación de la arquitectura Modelo/Vista/Controlador en la que la vista y el controlador se combinan para separar la forma en que los datos se almacenan de la forma en que son presentados al usuario, lo cual provee un marco de trabajo más simple basado en los mismos principios. (2016)

El modelo se comunica con una fuente de datos, proveyendo una interfaz para otros componentes en la arquitectura. La naturaleza de la comunicación depende del tipo de la fuente de datos y la forma en la que se implementa el modelo. (2016)

La vista obtiene índices modelos del modelo, los cuales son la referencia a los objetos de datos y le permite recolectar a estos en la fuente de datos. (2016)

En las vistas estándares, un delegado transforma los dato de manera que puedan ser interpretados de una mejor forma por los usuarios; cuando este objeto se edita, el delegado se comunica directamente con el modelo utilizando sus índices. (2016)

Por lo general las clases en esta arquitectura son separadas en tres grupos: modelos, vistas y delegados, los cuales se comunican a través de señales y aberturas. Cada uno de estos componentes está definido por clases abstractas que proveen interfaces amigables. Estas clases están destinadas a proveer un set completo de funcionalidades esperadas por otros componentes, lo que permite la especialización de alguno de ellos. (2016)

El patrón de arquitectura modelo-vista se ve expresado en la solución de tal forma que para la implementación de los componentes gráficos de control de tipo Interruptor y Selector, se crearon clases para la representación de la vista y el modelo como son; en caso de la vista se diseñaron las siguientes clase; Switch, State2EditSwitch, State2Switch y en caso del modelo fueron; SwitchModel, SelectorModel etc.

2.9 Patrones de Diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software, es decir brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Esto provee las siguientes ventajas: reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad informática, eficiencia y consistencia del diseño, y proporciona un considerable ahorro en la inversión. (Teniente López, 2004)

A continuación se describe la importancia de los patrones de diseño y como se aplican en el desarrollo de la estructura del sistema.

Entre los patrones GRASP utilizados en la definición del sistema se encuentran:

- **Experto:** Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (Gamma). De forma general el diseño del sistema se basa en asignar a cada clase la responsabilidad que solo ellas

pueden realizar, pues cada una cuenta con la información necesaria para llevarlas a cabo. Por lo que todas las clases del sistema son expertas.

- **Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos (Gamma). En el diseño del sistema se encuentra la clase Switch, la cual representa gráficamente los interruptores, esta se encuentra compuesta por la clase SwitchModel encargada de guardar todos los datos referentes a los interruptores como lo son: valor del estado, color del estado entre otras. En esta relación de composición se evidencia el patrón Creador, pues es la clase Switch responsable por la creación de la clase SwitchModel.
- **Bajo Acoplamiento:** El uso de los patrones Experto, Creador y Controlador contribuyen al bajo acoplamiento entre las clases del sistema. Este patrón se tuvo presente debido a la importancia que se le atribuye a realizar un diseño de clases independientes que puedan soportar los cambios de una manera fácil y que a su vez permitan la reutilización.
- **Alta Cohesión:** Se diseñaron las clases de forma tal que contengan las mínimas responsabilidades necesarias y colaboren con otras para llevar a cabo una tarea. Este patrón permitirá tener clases fáciles de mantener, entender y reutilizar.

Los patrones GoF utilizados fueron:

- **Observador** (Observer): Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos. El uso de este patrón se ve reflejado en las funcionalidades **updateState()**, donde este método se encarga de avisar a los demás componentes cuando las mediciones asociadas a este realizan algún cambio.
- **Plantilla** (Template Method): Este sencillo patrón resulta útil en casos en los que podamos implementar en una clase abstracta el código común que será usado por las clases que heredan de ella, permitiéndoles que implementen el comportamiento que varía mediante la reescritura (total o parcial) de determinados métodos. Este patrón se ve reflejado en los métodos **paintRuntime()**, **loadSVG()** y **paintEdition()**.

2.10 Conclusiones

En el capítulo presentado se definió el modelo del dominio, el cual refleja el punto de partida de la solución propuesta. Se especificaron además los requisitos del sistema que permitieron identificar las funcionalidades con las que este contará y que darán respuesta a las necesidades del usuario, las mismas fueron representadas mediante historias de usuarios, y luego fueron descritas. Se confecciono el Diagrama de Paquetes para representar las dependencias entre los paquetes que componen el modelo, el Diseño de Clases para los Componentes Gráficos y las Paletas de Componentes. Teniendo todos los requerimientos y las funcionalidades que se consideraron en este capítulo, se puede dar comienzo a construir el sistema previsto anteriormente.

CAPÍTULO 3

Implementación y Pruebas

3 Introducción

En el presente capítulo se describe la fase de implementación a través de la descripción detallada de las diferentes iteraciones realizadas durante la etapa de construcción del sistema basándose en las historias de usuarios definidas en el capítulo anterior. La fase de prueba es reflejada con los resultados obtenidos de las diferentes pruebas de aceptación diseñadas para probar los requisitos funcionales descritos anteriormente; pruebas las cuales son especificadas en los finales de este capítulo.

3.1 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. (Hernández, 2013)

3.2 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. (2008)

Para visualizar la estructura general del sistema se generó el diagrama de componente, el cual permitió describir el modelo de implementación y las relaciones entre los elementos de dicho modelo.

UML define cinco estereotipos estándar que se aplican en los componentes:

- Ejecutable: componente que se puede ejecutar.
- Biblioteca: biblioteca de objetos estática o dinámica.
- Tabla: componente que representa una tabla de base de datos.
- Archivo: componente que representa un documento que contiene código fuente o datos.

- Documento: componente que representa un documento. (2008)

A continuación se muestra el diagrama de componentes generado para el componente Interruptor y Selector para el HMI del SCADA SAINUX.

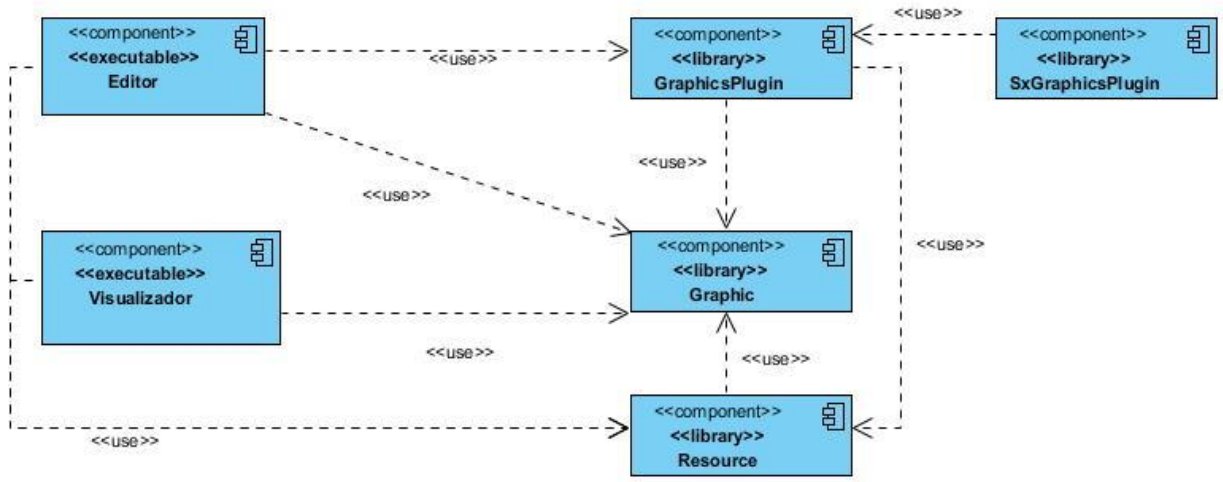


Ilustración 10: Diagrama de componentes

- GraphicsPlugins: Biblioteca contiene los Plugins pertenecientes a los objetos gráficos.
- Graphics: Biblioteca donde se encuentran agrupadas las entidades encargadas de representar los componentes gráficos, que serán utilizados en la representación.
- SXGraphic Plugins: Interfaz específica de la biblioteca Graphics Plugins para los componentes de SAINUX.
- Editor: Aplicación de escritorio que permite configurar todos los recursos asociados al proceso que se desea supervisar en el SCADA SAINUX.
- Visualizador: Permite visualizar, monitorear y controlar los recursos asociados a los procesos que se supervisan con el SCADA SAINUX.
- Resource: Biblioteca donde se encuentran agrupadas cada una de las entidades encargadas de brindar las imágenes, los iconos y los diseños de los componentes en SVG.

3.3 Estándar de codificación

Uno de los instrumentos que facilitan la tarea de asegurar la calidad del software es la adopción de estilos y estándares de codificación. El uso de estos estándares tiene innumerables ventajas entre ellas lograr un estilo de código homogéneo asegurando su legibilidad y proveer una guía para el encargado del mantenimiento/actualización del sistema, con código claro y bien documentado. Además ayuda a mejorar el proceso de codificación haciéndolo en gran medida eficiente y en muchos casos reutilizables. (Vernal)

A continuación se exponen los diferentes estilos de codificación que se utilizaron en la implementación del sistema:

- ✓ Para hacer una descripción breve se adopta el uso del comando @brief.
- ✓ Es importante especificar el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @autor y @date.
- ✓ El código será escrito en inglés y la documentación en español.
- ✓ Los atributos de las clases deben comenzar con la letra m seguido de guion bajo y a continuación el nombre del atributo, si existen atributos compuestos la segunda palabra debe comenzar con mayúscula.

Ejemplo: m_valuesState

- ✓ Los parámetros que recibe una función deben comenzar con guion bajo.

Ejemplo: void setShowValues (bool _showValues)

3.4 Diagrama de Despliegue

El diagrama de despliegue se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes. (Pearson., 2005)

- ❖ Permiten modelar la disposición física o topología de un sistema.
- ❖ Muestra el *hardware* usado.
- ❖ Muestra las conexiones físicas entre el hardware y las relaciones entre componentes.

A continuación se muestra el diagrama de despliegue modelado para el componente gráfico a desarrollar.

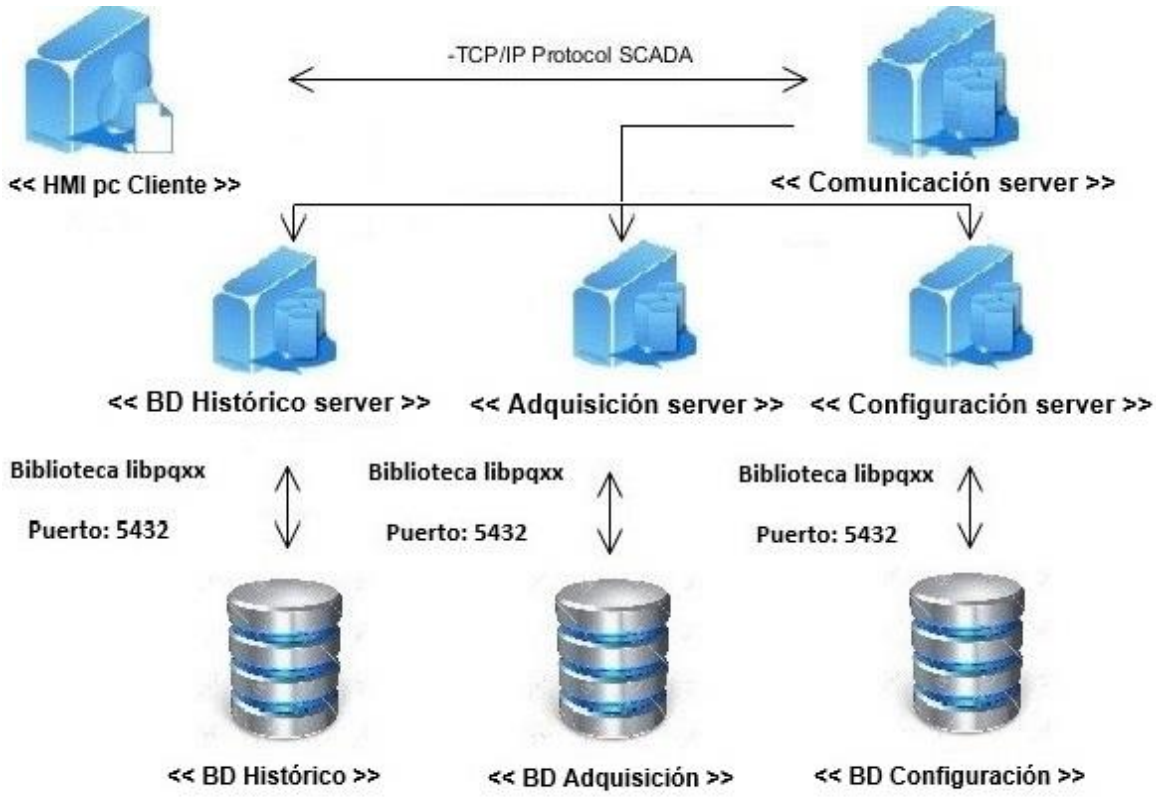


Ilustración 11: Diagrama de Despliegue

En el diagrama de despliegue se encuentra el nodo HMI pc cliente que es la pc donde se encuentra instalado el HMI, los cuales se encuentran desplegados el Ambiente Configuración y el Ambiente Edición, el mismo se comunica con los nodos de servidor Adquisición, Servidor Base de Datos Históricos y Servidor Configuración: Estos módulos se ejecutan en distintas Pc que pueden servir como servidores poniendo en evidencia la arquitectura distribuida que presenta el SCADA SAINUX, conectándose con sus respectiva base de datos, utilizando la biblioteca ibpqxx por el puerto 5432 . La conexión entre estos módulos se realiza utilizando comunicación TCP/IP con el protocolo SCADA desarrollado por el centro. La solución se ve enmarcada en el módulo de HMI del SCADA SAINUX.

CAPÍTULO 3

Implementación y Pruebas

3.5 Pruebas de software

El desarrollo de Sistemas de software implica la realización de una serie de actividades predispuestas a incorporar errores (en la etapa de definición de requerimientos, de diseño, de desarrollo,). Debido a que estos errores se deben a nuestra habilidad innata de provocar errores, tenemos que incorporar una actividad que garantice la calidad del software. (Glenford J. Myers, 2004)

En la etapa de prueba del software se crean una serie de casos de prueba que intentan "destruir" el software desarrollado. La prueba requiere que se descarten ideas preconcebidas sobre la "calidad o corrección" del software desarrollado. La prueba es un proceso de ejecución de un programa con la intención de descubrir un error. (Glenford J. Myers, 2004)

Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces

El objetivo es diseñar casos de prueba que, sistemáticamente, saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo.

La prueba no puede asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software. (Glenford J. Myers, 2004)

3.5.1 Niveles de pruebas de software

- *Pruebas Unitarias*: Tiene como objetivo detectar errores en los datos, lógicas, algoritmos.
- *Pruebas de Integración*: Tiene como objetivo detectar errores de interfaz y relaciones entre componentes.
- *Pruebas Funcionales*: Tiene como objetivo detectar errores en la implementación de requerimientos.
- *Pruebas del Sistema*: Tiene como objetivo detectar fallas en el cubrimiento de los requerimientos.
- *Pruebas de Aceptación*: Tiene como objetivo detectar fallas en la implementación del sistema.

CAPÍTULO 3

Implementación y Pruebas

Unas de las disciplinas de APU-UCI es el uso de las pruebas, el objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificando que se cumplan los requisitos. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones.

La metodología APU-UCI divide las pruebas en dos grupos: pruebas validación y pruebas de aceptación. Las pruebas validación está enfocada en hacer lo que el usuario realmente quiere, además de determinar la corrección del producto final respecto a las necesidades de ese usuario y las pruebas de aceptación están destinadas a verificar que al final de cada iteración las Historias de Usuario cumplen con la funcionalidad asignada y satisfagan las necesidades del cliente. Las pruebas de aceptación son más importantes que las pruebas de validación dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente, por esto el cliente es la persona adecuada para diseñar las pruebas de aceptación.

3.5.2 Pruebas de aceptación

Las Pruebas de Aceptación (PA) fueron creadas en base a las Historias de Usuarios (HU), en cada ciclo de la iteración del desarrollo. El cliente especificó uno o diversos escenarios para comprobar que una historia de usuario fue correctamente implementada. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, se indicó el orden de prioridad de resolución. (Letelier)

Precisando un poco más:

- PA describe un escenario (secuencia de pasos) de ejecución o uso del sistema desde la perspectiva del cliente.
- Puede estar asociada a requisitos funcionales o no funcionales
- Un requisito tiene una o más PA asociadas.

- Las PA cubren desde escenarios típicos/frecuentes hasta los más excepcionales. (Letelier)

Prueba de Aceptación	
Número: 1	**Historia de usuario:** 1
Nombre: El sistema debe permitir asociar medición al componente.	
Descripción: Su objetivo es asignar una medición al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde el operador puede seleccionar la medición deseada.	
Condiciones de Ejecución: El usuario debe comprobar que al asignar una medición, el componente, tome dicha dimensión asociada.	
Entradas/ Pasos de Ejecución: • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar la opción Mediciones. • Seleccionar la posible medición que desee que tome.	
Resultado esperado: EL componente toma la medición deseada.	

Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.

Tabla 4: Caso de Prueba Aceptación

La descripción de las restantes pruebas de aceptación se encuentra en el anexo 4

Sistema	HU	Iteración	NC	Cerrada	No Procede
Interruptores y Selectores	15	1ra	9	9	0
		2da	6	6	0
		3ra	0	0	0

Tabla 5: Resultado de las iteraciones.

Después de realizar las pruebas funcionales utilizando el tipo de prueba de aceptación, se comprobó el correcto funcionamiento de la aplicación desarrollada. Cada problema detectado en el desarrollo del sistema fue resuelto a raíz del trabajo continuo del desarrollador, con un total de 9 no conformidades encontradas en la primera iteración, 6 en la segunda y ninguna en la tercera, las cuales se dividieron en significativas y no significativas.

Las no conformidades encontradas pueden clasificarse según el tipo de incidencias:

1. De presentación: No se visualiza toda la información que se desea mostrar y fueron localizados varios errores ortográficos siendo estos generalmente de acentuación.
2. De ejecución: Durante el despliegue del sistema fueron detectadas diferentes fallas, siendo estas ocasionadas porque el sistema no realizaba todos los procedimientos definidos.

A continuación se representa lo expuesto anteriormente a través de la siguiente figura.

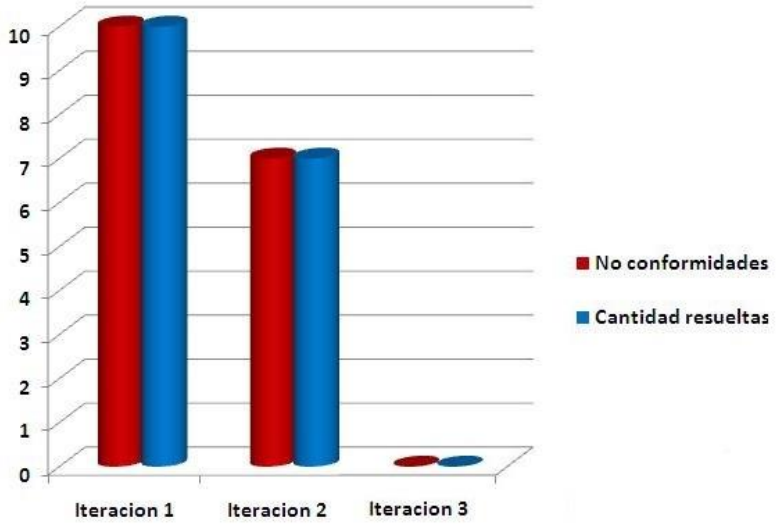


Ilustración 12: Pruebas del sistema

3.6 Conclusiones Parciales

En el capítulo expuesto se diseñó el modelo de implementación correspondiente a los componentes de tipo Interruptor y Selector para el HMI del SCADA SAINUX, para ello se realizó el diagrama de componentes en el que se definieron los componentes necesarios para el correcto funcionamiento. Una vez terminado el flujo de trabajo implementación se pasó al flujo de trabajo de prueba, donde se aplicaron los tipos de prueba de aceptación. Cada dificultad detectada en el desarrollo del sistema fue resuelta, con un total de 15 no conformidades detectadas durante 3 iteraciones, que fueron satisfactoriamente resueltas.

Conclusiones Generales

Conclusiones Generales

Con la realización del siguiente trabajo se logró ampliar la gama de componentes gráficos en la biblioteca de componentes gráficos del SCADA SAINUX. Una vez concluida la presente investigación, se logró:

- La elaboración de un marco teórico acerca de los principales conceptos de un sistema SCADA, así como Interruptores y Selectores.
- Desarrollar componentes gráficos que permitieron elevar los grados de representación de interruptores y selectores en el proceso de visualización de la Interfaz Hombre Máquina del sistema SCADA SAINUX.
- Desarrollar un mecanismo que permite la ejecución de comandos sobre las mediciones de los componentes gráficos.

Recomendaciones

Recomendaciones

- Se recomienda seguir ampliando la gama de componentes gráficos de control de tipo Interruptor y selector en el SCADA SAINUX.
- Incorporar los componentes gráficos desarrollados en el visor web, que se desarrolla en el centro.

Bibliografías

[En línea] http://www.debian.org.

[En línea] http://www.altova.com/es/umodel/uml-package-diagrams.html..

2016. **[En línea] 2016. http://doc.qt.io/qt-4.8/model-view-programming.html..**

DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. ***Definición de SVG.*** **[En línea] [Citado el: 27 de enero de 2016.] http://www.alegsa.com.ar/Dic/svg.php..**

B.O.D. Servicio de Biodinternet, guía de acceso al módulo de configuración. 2015. **Servicio de Biodinternet, guía de acceso al módulo de configuración. [En línea] 25 de octubre de 2015. http://www.bod.com.ve/media/26317/Guía_del_Módulo_Configuración.pdf..**

Barcojo, Alfonso Romero. **resumen inicicacion circuitos basicos. [En línea] [Citado el: 1 de Junio de 2016.] www.tecnojose.es/resumen%20inicicacion%20circuitos%20basicos.**

Cardozzo, Daniel Ramos. 2014. **Desarrollo de Sotware. 2014.**

CORPORATION, NOKIA. Qt. 2008. **[En línea] 2008. http://qt.nokia.com/products/developer-tools/..**

Dagoberto Monteros, David B. Barrantes y José M. Quirós. 2004.. ***Introducción a los sistemas de control, supervisión y adquisición de datos (SCADA). .*** **2004.**

2003. **Emersonprocess.** ***Módulo de Comunicación (serie ROC 800).*** **[En línea] Octubre de 2003. http://www.documentation.emersonprocess.com/groups/public/documents/specification_sheets/d301583x012.pdf..**

Eumed. ***Eumed.net.*** **[En línea] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm..**

Gamma. **Elements of Reusable Object-Oriented Software.**

Bibliografías

Glenford J. Myers, John Wiley. 2004. **The Art of Software. s.l. : 2, 2004.**

Group, ABB. *Módulo de Seguridad.* [En línea] http://www.abb.com/product/es/9AAC170130.aspx..

Hernández, Leovigilda. 2013. **[En línea] 1 de Junio de 2013. http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html..**

1998-2016. **IEC 60669-1. *Switches for household and similar fixed-electrical installations.* [En línea] 1998-2016. http://www.simbologia-electronica.com/simbolos-electricos-electronicos/simbolos-interruptores-electricos.htm.**

Jacobson Ivar, Booch Grady y Rumbaugh James. 1999. **El Proceso Unificado de Desarrollo de Software. 1999.**

Laurent DEBRAUWER, Fien VAN DER HEYDE. 2013. ***UML 2: Iniciación, ejemplos y ejercicios corregidos.* Barcelona : Ediciones ENI, 2013. ISBN 2746079941, 9782746079946.**

Letelier, Patricio. **Pruebas de Aceptación como Pruebas de Aceptación como conductor del Proceso Software. [En línea] http://in2test.lsi.uniovi.es/repris/.**

Mad, Freepress S. Coop. Freepress S. ***Inkscape-software-libre-para-diseño-vectorial.***

Morales., Carlos de Castro Lozano y Cristóbal Romero. ***Introduccion a SCADA. .***

Oliag, Sergio Talens. **Curso de Programación en C++. [En línea] http://www.uv.es/~sto/cursos/c++/curso95.pdf..**

Paradigm, Visual Paradigm. Visual. 2013. **UML CASE for software development. Visual Paradigm.UML CASE for software development. [En línea] 3 de septiembre de 2013. www.visual-paradigm.com/product/vpuml..**

Bibliografías

Pearson., Sommerville . s.l. :. 2005. **Ingeniería del Software. 2005.**

Penin, A.R. s.l. : Marcombo,ISBN 9788426716477. 2012. ***Sistemas SCADA.* 2012.**

Pressman, S. 2001. **2001.**

Raydel Raúl Viñolo Sosa, Alexander Roquero Figueroa. 2012. **Sistema Gestor de Proceso de Media v2. [En línea] julio de 2012. http://publicaciones.uci.cu/index.php/SC | seriecientifica@uci.cu..**

Rumbaugh, James, Jacobson, Ivar y Booch, Graddy. 2005. ***El lenguaje Unificado de modelado.* Estados Unidos : Addison-Wesley, 2005. 1.**

S. Pressman, Roger. 2001. **Ingeniería del Software.Un enfoque práctico. [aut. libro] Darrel Ince. 2001.**

Sánchez, Tamara Rodríguez. **Metodología de Desarrollo para la Actividad Peroductiva de la UCI. [En línea] http://excriba.prod.uci.cu/page/context/shared/sharedfiles/Metodologiauci.pdf..**

2015. **scribd. [En línea] 16 de octubre de 2015. http://es.scribd.com/doc/284387390/2-4-Sistema-HMI#scribd..**

2008. **Scribd. *Scribd.com.* [En línea] 2008. http://www.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue..**

Subversion. *Apache Subversion.* [En línea] https://subversion.apache.org/..

Teniente López, Ernest. 2004. **Diseño de sistemas de software en UML. Universidad Politécnica de Catalunya : s.n., 2004.**

Vernal, Francisco. **[En línea] http://www.inf.utfsm.cl/~visconti/xp/Guia_Estandares_Codificacion_2.doc..**

Bibliografías

Yepes, José López. 2001. **Redalyc.org. *Las Bases de Datos Históricas.* [En línea] septiembre de 2001. [Citado el: 25 de Diciembre de 2015.] http://www.redalyc.org/articulo.oa?id=16100905..**

Anexo 1

Historia de Usuario	
Número: 1	Nombre de Historia: Asociar medición al componente.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Alta	Tiempo estimado: 1 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 1 semana
Descripción: Su objetivo es asignar una medición al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde el operador puede seleccionar la medición deseada.	

Tabla 6: Historia de Usuario 1

Historia de Usuario	
Número: 3	Nombre de Historia: Definir nombre del componente
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Baja	Tiempo estimado: 0,2 semana

Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0,2 semana
Descripción: Su objetivo es asignar un nombre al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, donde se puede asignar el nombre deseado al componente, haciendo clic izquierdo encima del campo que está al lado de Nombre.	

Tabla 7: Historia de Usuario 3

Historia de Usuario	
Número: 4	Nombre de Historia: Configurar la dimensión del componente.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Media	Tiempo estimado: 0,6 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0,6 semana
Descripción: Su objetivo es asignar una dimensión al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde se puede seleccionar el alto y el ancho que se desea que tenga el componente, definido en el rango (- 2147483647; 2147483647).	

Tabla 8: Historia de Usuario 4

Historia de Usuario	
Número: 5	Nombre de Historia: Configurar la opacidad del objeto del componente.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Alta	Tiempo estimado: 0.6 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0.6 semana
Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde se puede seleccionar la opacidad del componente que la misma está dada en un intervalo de 0 – 100 fijo.	

Tabla 9: Historia de Usuario 5

Historia de Usuario	
Número: 6	Nombre de Historia: Definir la descripción del componente.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Baja	Tiempo estimado: 0.4 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0.4 semana

Descripción: Su objetivo es asignar una descripción al componente. Al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde se puede efectuar una breve descripción del componente seleccionado.

Tabla 10: Historia de Usuario 6

Historia de Usuario	
Número: 7	Nombre de Historia: Configurar posición del componente.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Baja	Tiempo estimado: 0.4 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0.4 semana
Descripción: Su objetivo es asignar una posición al componente. Al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, donde está la opción posición, en la que permite trasladar el objeto sobre los eje X y Y, definido en el rango (- 2147483647; 2147483647) que se quiere mostrar el componente en el despliegue.	

Tabla 11: Historia de Usuario 7

Historia de Usuario	
Número: 8	Nombre de Historia: Configurar color que representa el estado indefinido.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Alta	Tiempo estimado: 0.8 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0.8 semana
Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde puede seleccionar el color indefinido que quiere que tome el componente, al dar clic izquierdo sobre Color Indefinido se despliega tres combinaciones de colores, rojo, verde y azul el cual le podemos asignar valores de 0 – 255, o simplemente damos clic izquierdo al lado del campo de Color indefinido y se muestra una ventana que le da la opción de escoger el color básico que desee.	

Tabla 12: Historia de Usuario 8

Historia de Usuario	
Número: 9	Nombre de Historia: Configurar color del indicador.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Alta	Tiempo estimado: 0.8 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0.8 semana

Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde puede seleccionar el color indicador que quiere que tome el componente, al dar clic izquierdo sobre Color Indicador se despliega cuatros combinaciones de colores, rojo, verde, azul y transparencia el cual le podemos asignar valores de 0 – 255, o simplemente damos clic izquierdo al lado del campo de Color Indicador y se muestra una ventana que le da la opción de escoger el color básico que desee.

Tabla 13: Historia de Usuario 9

Historia de Usuario	
Número: 10	Nombre de Historia: Configurar color de borde.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Alta	Tiempo estimado: 0.8 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0.8 semana
Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde puede seleccionar el color borde que quiere que tome el componente, al dar clic izquierdo sobre Color Borde se despliega cuatros combinaciones de colores, rojo, verde, azul y transparencia el cual le podemos asignar valores de 0 – 255, o simplemente damos clic izquierdo al lado del campo de Color borde y se muestra una ventana que le da la opción de escoger el color básico que desee.	

Tabla 14: Historia de Usuario 10

Historia de Usuario	
Número: 13	Nombre de Historia: Mostrar y ocultar la escala.
Programador: Javier Bravo Calzado	Iteración Asignada: 1
Prioridad en Negocio: Alta	Tiempo estimado: 0.8 semana
Riesgo en Desarrollo: Riesgo definido en el GESPRO.	Tiempo real: 0.8 semana
Descripción: Su objetivo es asignar una medición al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde le da la opción de Si o No mostrar la escala, dando clic izquierdo.	

Tabla 15: Historia de Usuario 13

Anexo 2

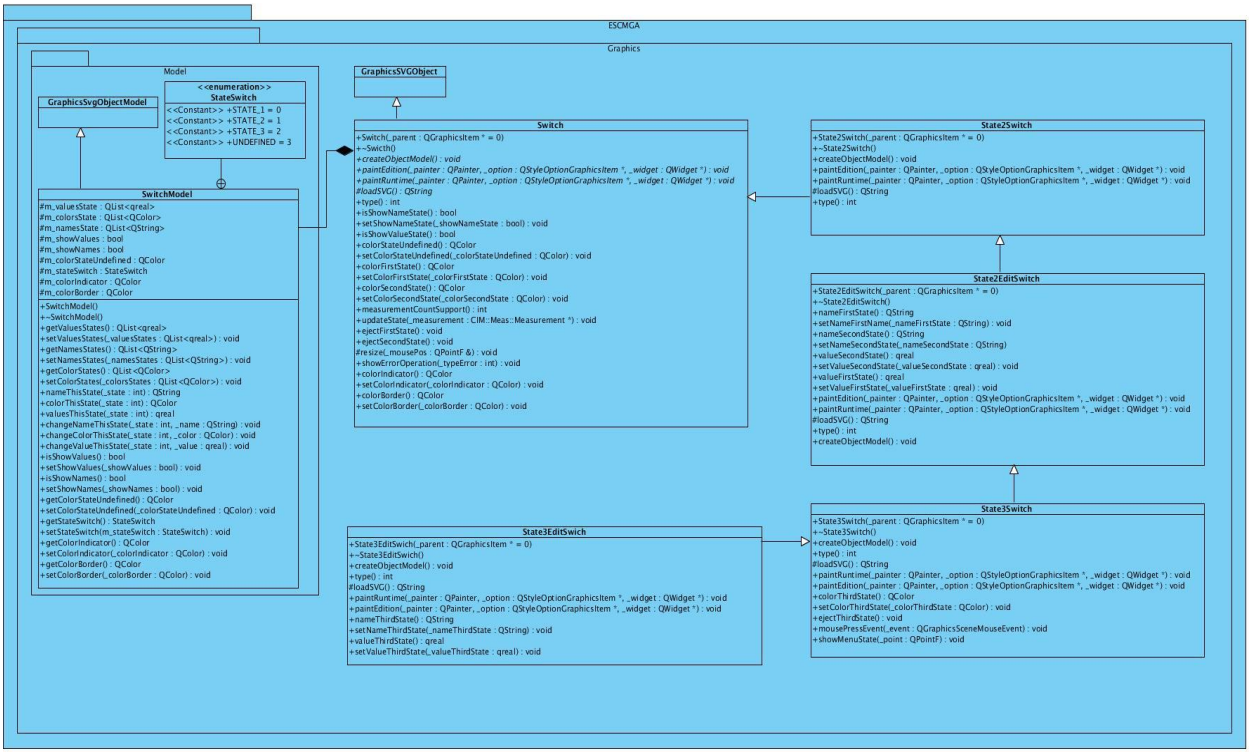


Ilustración 13: Diseño de clase

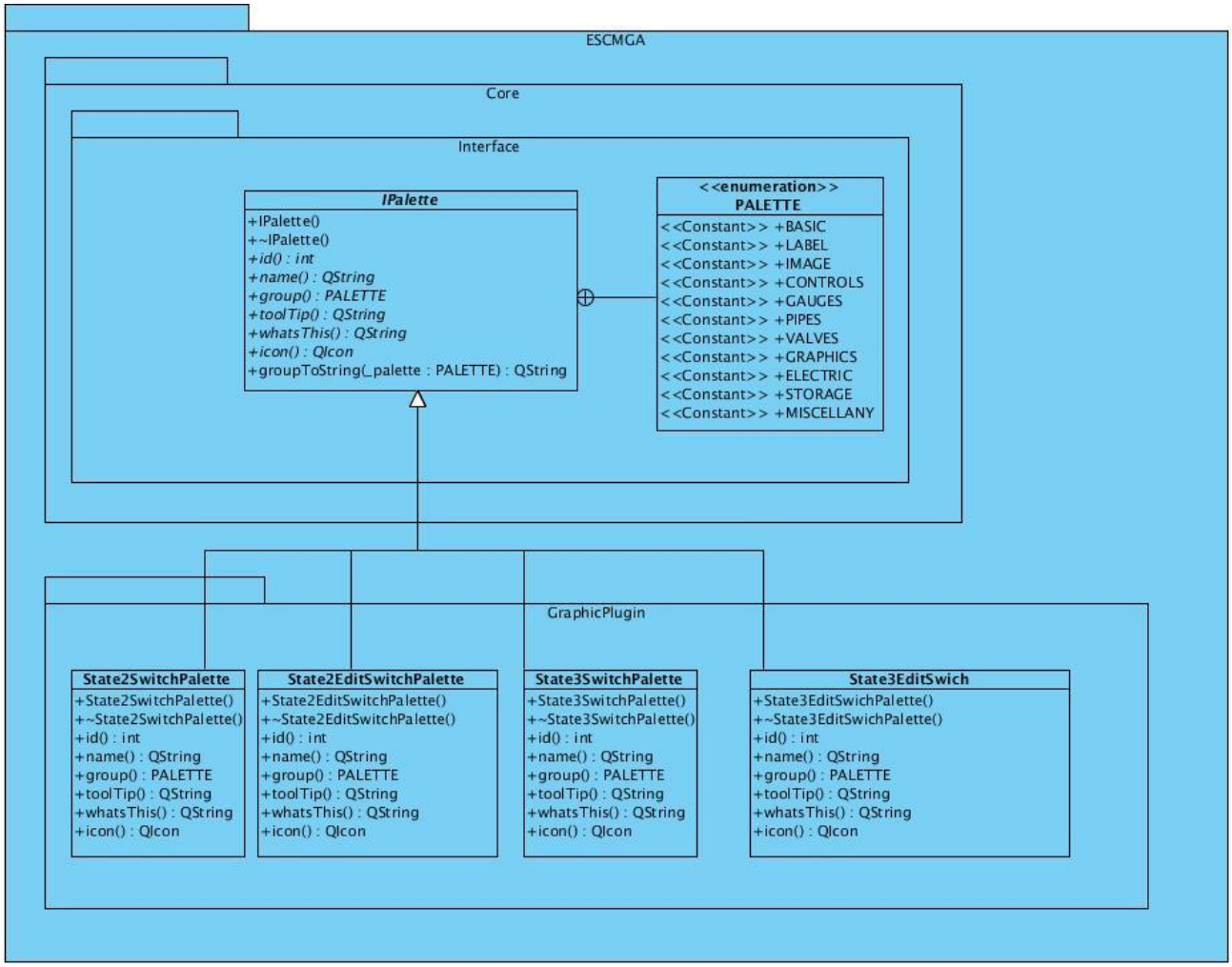


Ilustración 14: Diseño de clases de paletas

Anexo 3

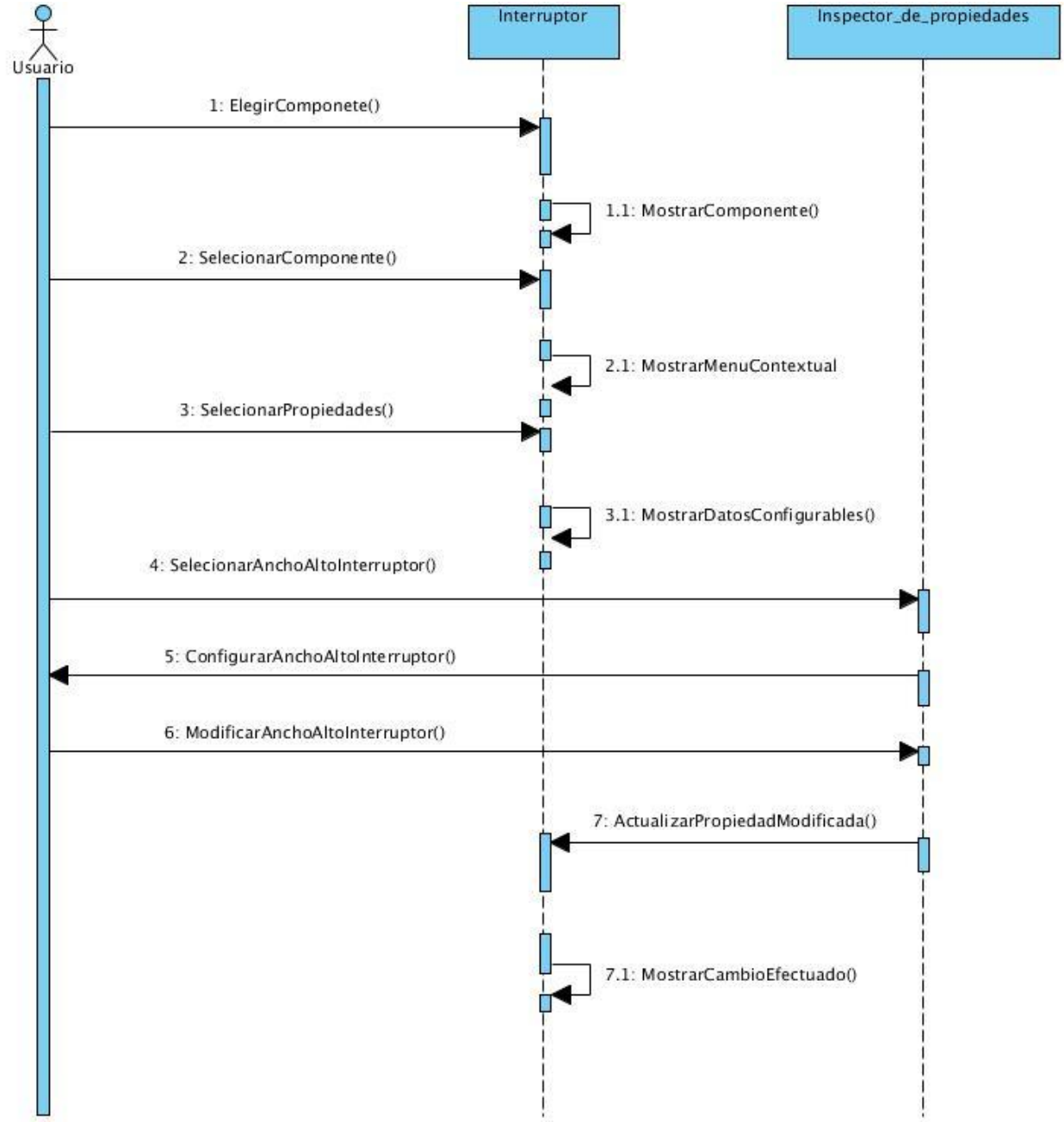


Ilustración 15: Diagrama de secuencia AnchoAlto

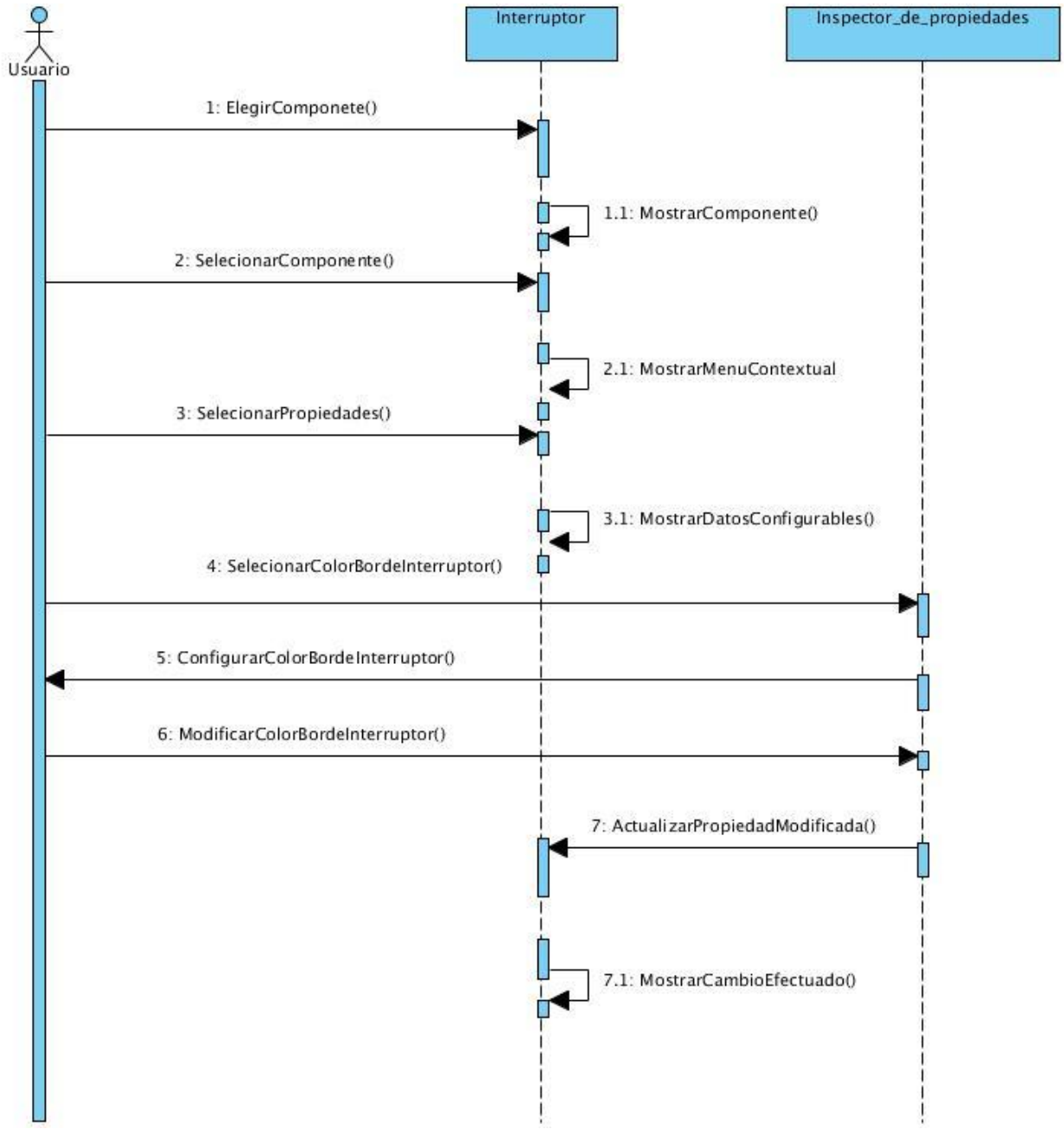


Ilustración 16: Diagrama de secuencia Color borde

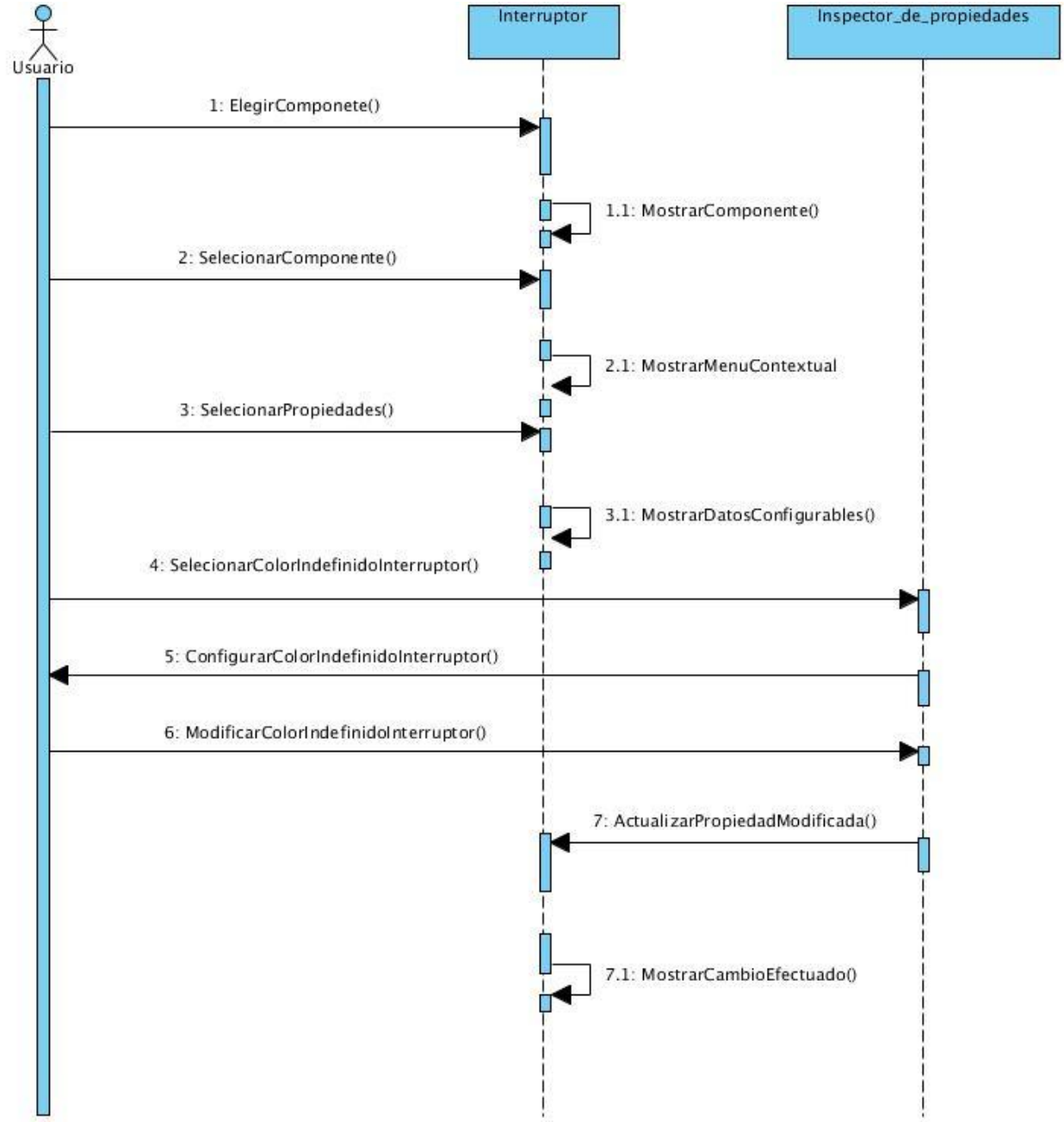


Ilustración 17: Diagrama de secuencia Color Indefinido

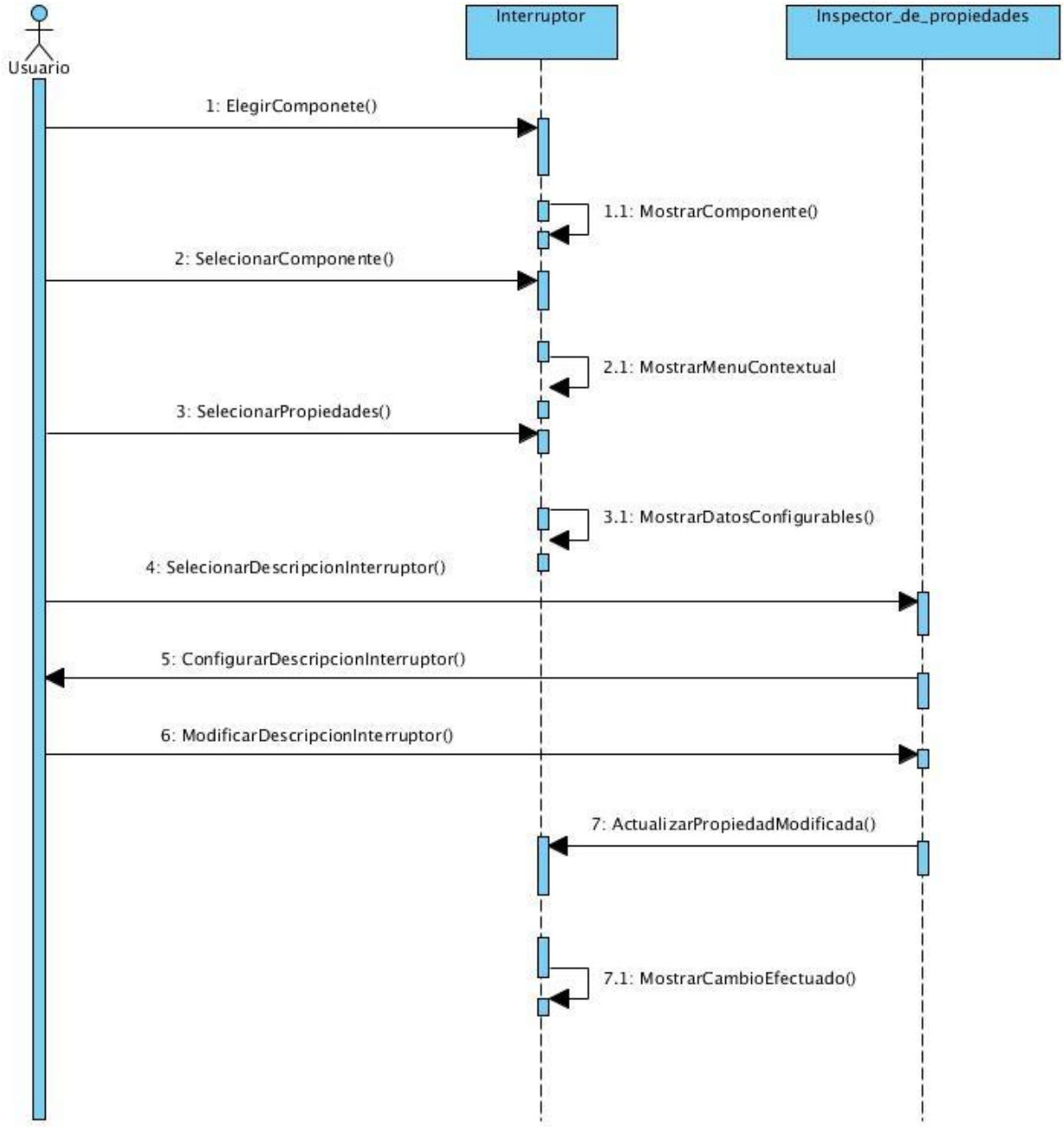


Ilustración 18: Diagrama de secuencia Descripción

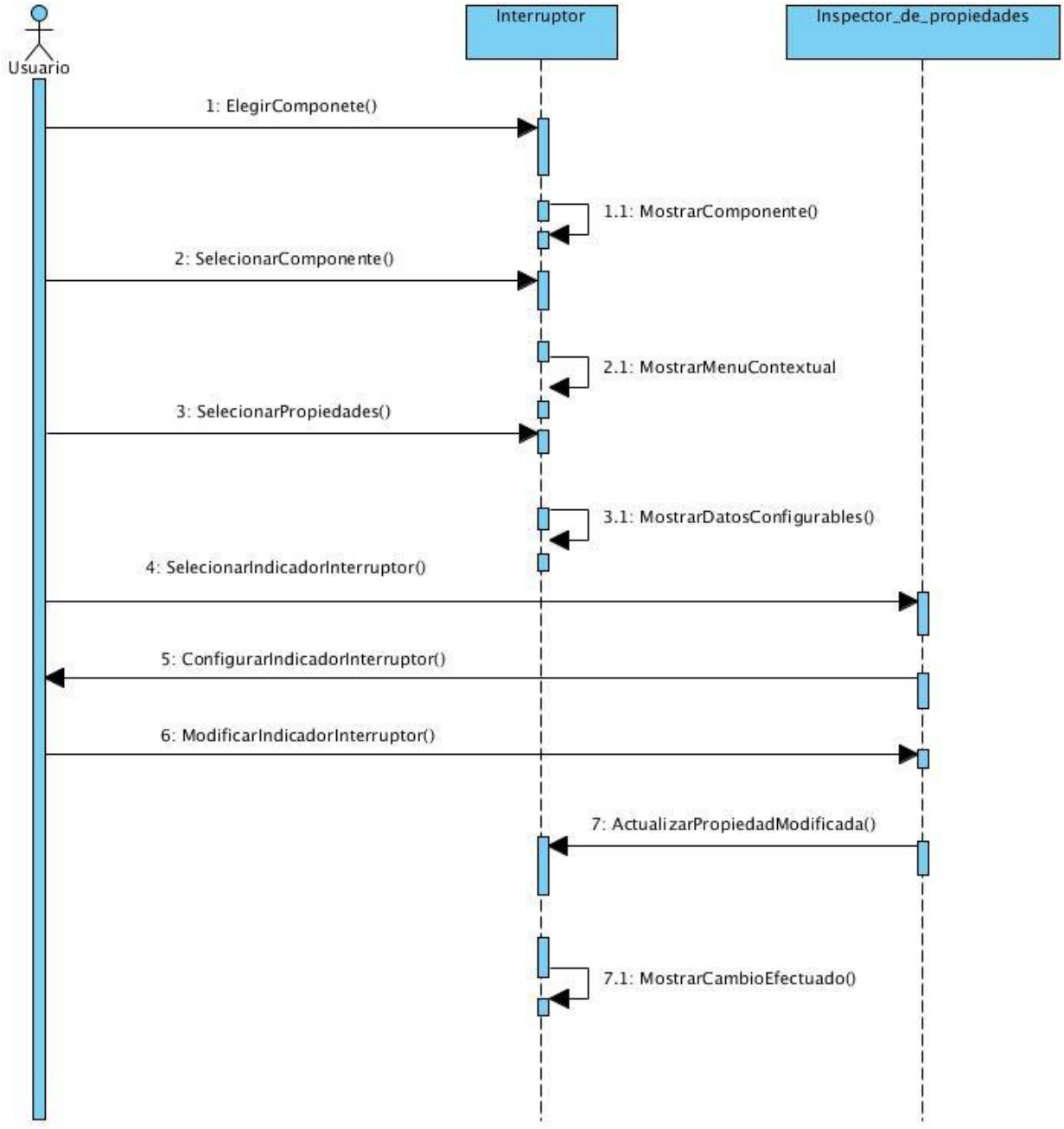


Ilustración 19: Diagrama de secuencia Color Indicador

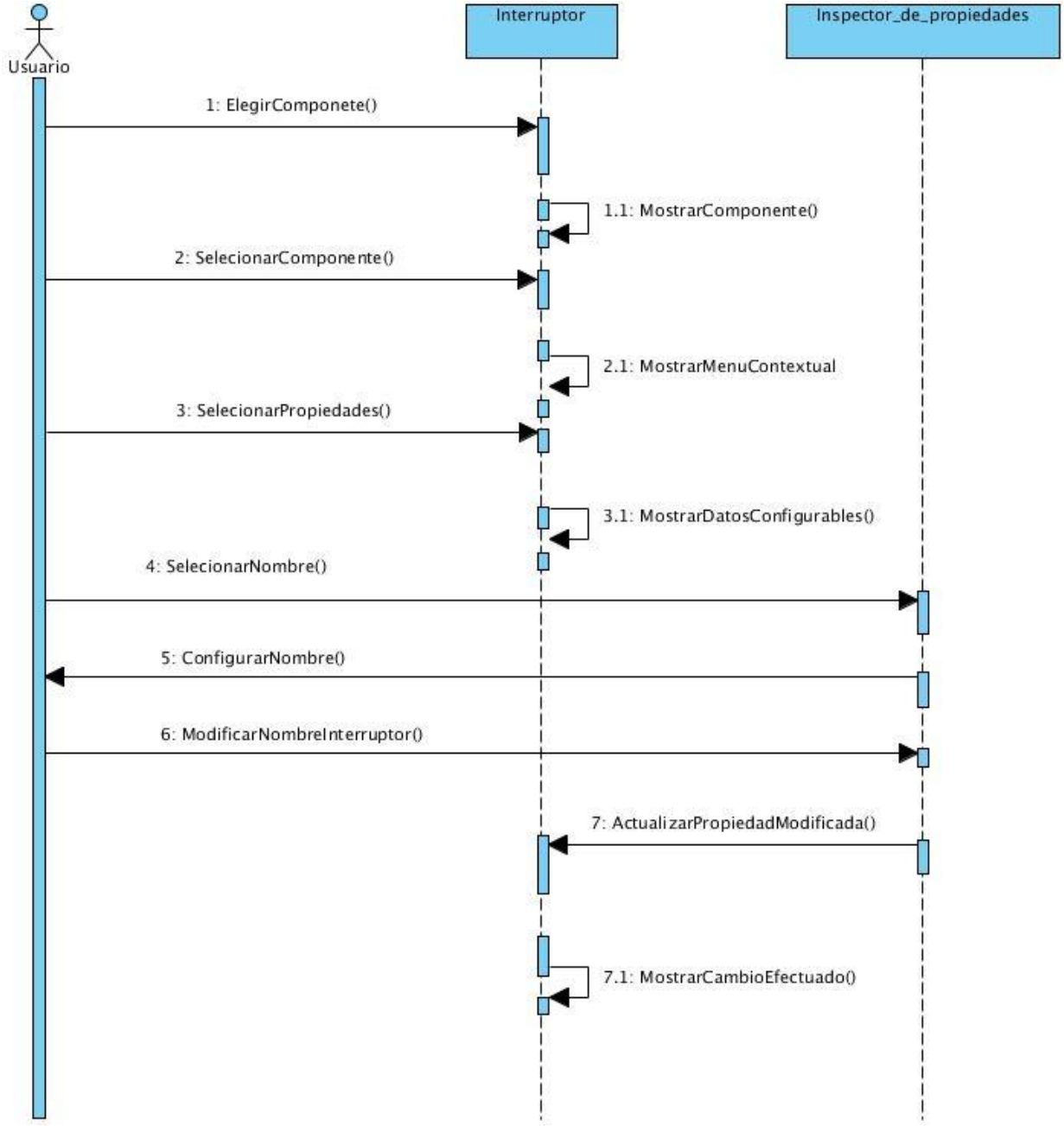


Ilustración 20: Diagrama de secuencia Nombre

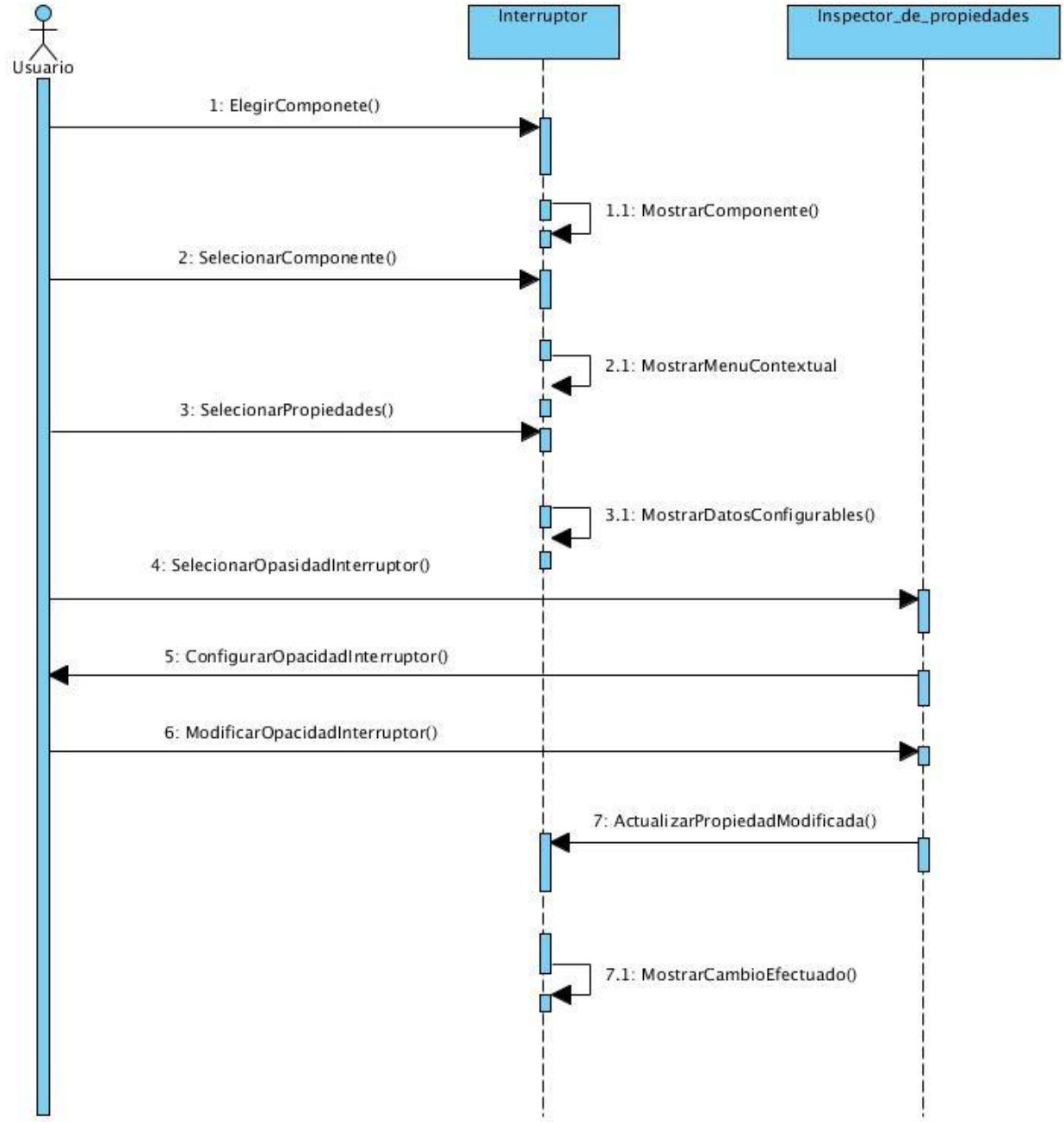


Ilustración 21: Diagrama de secuencia Opacidad

Anexo 4

Prueba de Aceptación	
Número: 3	Historia de usuario: 3
Nombre: El sistema debe permitir definir nombre del componente.	
Descripción: Su objetivo es asignar un nombre al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde el operador puede asignarle el nombre deseado al componente, haciendo clic izquierdo encima del campo que está al lado de Nombre.	
Condiciones de Ejecución: El usuario debe comprobar que al asignar un nombre al componente, el mismo tiene que tomar dicho nombre asociado.	
Entradas/ Pasos de Ejecución: • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar la opción del campo que está al lado de Nombre. • Insertar el nombre deseado.	
Resultado esperado: EL componente toma el nombre definido.	

Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.

Ilustración 22: Prueba de aceptación 3

Prueba de Aceptación	
Número: 4	Historia de usuario: 4
Nombre: El sistema debe permitir configurar la dimensión del componente.	
Descripción: Su objetivo es asignar una dimensión al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde el operador puede seleccionar el alto y el ancho que desea que tenga el componente, definido en el rango (- 2147483647; 2147483647).	
Condiciones de Ejecución: El usuario debe comprobar que al asignar una dimensión, el componente, tome dicha dimensión asociada.	
Entradas/ Pasos de Ejecución: • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar la opción Dimensión. • Asociar la dimensión en el eje X y Y deseada en un intervalo de valor de (- 2147483647 ; 2147483647)	

Resultado esperado: EL componente toma la dimensión deseada.
Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.

Ilustración 23: Prueba de aceptación 4

Prueba de Aceptación	
Número: 5	Historia de usuario: 5
Nombre: El sistema debe permitir configurar la opacidad del objeto del componente.	
Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde el operador puede seleccionar la opacidad del componente que la misma está dada en un intervalo de 0 – 100 fijo.	
Condiciones de Ejecución: El usuario debe configurar la opacidad del componente.	
Entradas/ Pasos de Ejecución: • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar la opción Opacidad. • Modificar la opacidad del componente según el usuario desee en un rango de valores de 0 - 100.	

Resultado esperado: EL componente toma la opacidad deseada del usuario.
Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.

Ilustración 24: Prueba de aceptación 5

Prueba de Aceptación	
Número: 6	Historia de usuario: 6
Nombre: El sistema debe permitir definir la descripción del componente.	
Descripción: Su objetivo es asignar una descripción al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde el operador puede efectuar una breve descripción del componente seleccionado.	
Condiciones de Ejecución: El usuario debe comprobar que al asignar una descripción, el componente, tome dicha descripción asociada.	

Entradas/ Pasos de Ejecución:

- Seleccionar componente.
- Clic derecho con el ratón y abrir inspector de propiedades.
- Marcar la opción Descripción.

Insertar descripción deseada

Resultado esperado: EL componente toma la descripción deseada.

Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.

Ilustración 25: Prueba de aceptación 6

Prueba de Aceptación	
Número: 7	Historia de usuario: 7
Nombre: El sistema debe permitir configurar posición del componente.	
Descripción: Su objetivo es asignar una posición al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, configurar la posición en la que se traslada el objeto sobre el eje X y el eje Y, definido en el rango (- 2147483647; 2147483647) donde quiere el operador que se muestre el componente en el despliegue.	

Condiciones de Ejecución: El usuario debe comprobar que al asignar una posición, el componente, tome dicha posición asociada.
Entradas/ Pasos de Ejecución: • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar la opción posición. • Seleccionar la posible posición que desee que tome.
Resultado esperado: EL componente toma la posición deseada.
Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.

Ilustración 26: Prueba de aceptación 7

Prueba de Aceptación	
Número: 8	Historia de usuario: 8
Nombre: El sistema debe permitir configurar color que representa el estado indefinido.	
Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde puede seleccionar el color indefinido que quiere que tome el componente, al dar clic izquierdo sobre Color Indefinido se despliega tres combinaciones de colores, rojo, verde y	

azul el cual le podemos asignar valores de 0 – 255, o simplemente damos clic izquierdo al lado del campo de Color indefinido y se muestra una ventana que le da la opción de escoger el color básico que desee.
Condiciones de Ejecución: El usuario debe comprobar que al asignar un color indefinido al componente, tome dicho color indefinido.
Entradas/ Pasos de Ejecución: • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar la opción Color Indefinido. • Seleccionar el posible color indefinido que desee que tome.
Resultado esperado: EL componente toma el color indefinido deseado.
Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.

Ilustración 27: Prueba de aceptación 8

Prueba de Aceptación	
Número: 9	Historia de usuario: 9
Nombre: El sistema debe permitir configurar color del indicador.	

Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde puede seleccionar el color indicador que quiere que tome el componente, al dar clic izquierdo sobre Color Indicador se despliega cuatros combinaciones de colores, rojo, verde, azul y transparencia el cual le podemos asignar valores de 0 – 255, o simplemente damos clic izquierdo al lado del campo de Color Indicador y se muestra una ventana que le da la opción de escoger el color básico que desee.
Condiciones de Ejecución: El usuario debe comprobar que al asignar un color indicador al componente, tome dicho color indicador.
Entradas/ Pasos de Ejecución: • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar la opción Color Indicador. • Seleccionar el posible color de indicador que desee que tome.
Resultado esperado: EL componente toma el color indicador deseado.
Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.

Ilustración 28: Prueba de aceptación 9

Prueba de Aceptación	
Número: 10	Historia de usuario: 10
Nombre: El sistema debe permitir configurar color de borde.	
Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte izquierda del editor se muestra un inspector de propiedades, en donde puede seleccionar el color borde que quiere que tome el componente, al dar clic izquierdo sobre Color Borde se despliega cuatros combinaciones de colores, rojo, verde, azul y transparencia el cual le podemos asignar valores de 0 – 255, o simplemente damos clic izquierdo al lado del campo de Color borde y se muestra una ventana que le da la opción de escoger el color básico que desee.	
Condiciones de Ejecución: El usuario debe comprobar que al asignar un color borde al componente, tome dicho color borde.	
Entradas/ Pasos de Ejecución: • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar la opción Color borde. • Seleccionar el posible color borde que desee que tome.	
Resultado esperado: EL componente toma el color borde deseado.	
Evaluación de la prueba: Se realiza la primera iteración donde no fueron identificadas no conformidades ya que arrojó una evaluación satisfactoria.	

Ilustración 29: Prueba de aceptación 10