Universidad de las Ciencias Informáticas

Facultad 5

Desarrollo del componente gráfico de barras para el HMI del SCADA SAINUX

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

Autor: Maiyara Guerra Cardet

Tutor: Ing. Frank Rufino Nápoles

Ciudad de la Habana, 2015.

DECLARACIÓN DE AUTORÍA:

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

Para que así conste firmo la presente a los __ días del mes _____ del año 2015.

_____	_____
Maiyara Guerra Cardet	Ing. Frank Rufino Nápoles
Autora	Tutor

DATOS DEL CONTACTO

Tutor: Frank Rufino Nápoles
Categoría Científica: Ingeniero
Especialidad: Ingeniero en Ciencias Informáticas
Correo Electrónico: frufino@uci.cu
Categoría docente:
Años de experiencia: 4
Años de graduado: 4

Agradecimientos:

Muchas son las personas que a lo largo de estos cinco años han colaborado con lo que ayer era una meta y hoy se ha convertido en realidad. La vida alejó a algunos, aproximó a otros, e hizo permanente la presencia de los mejores.

A mis padres, Madeley y Rey, gracias por quererme, alentarme, ayudarme, por estar siempre ahí para mí cuando lo he necesitado, y por todo lo que han hecho para darme todos los gustos posibles, gracias por existir, los quiero muchísimo, para ustedes este regalo.

A mi hermano Randy, gracias por existir, te quiero muchísimo.

A mi abuela y mis padrinos por preocuparse por mí.

A toda mi familia, gracias por el apoyo que siempre me han brindado.

A Esmerida, gracias por ser la mejor amiga que he tenido en mi vida, por todos los momentos que hemos pasado juntas, por los consejos, por ser siempre sincera conmigo y por apoyarme cuando lo he necesitado, ten presente que más que mi amiga eres mi hermana.

A mis amistades durante estos cinco años, Yalena, Hayron, Nino, Handy, gracias por todo lo que hemos vivido, por las risas, por el apoyo, y por cuidarme.

A quien haya formado parte de mi vida, gracias por los momentos y los recuerdos que siempre tendré.

A Rufino, nunca hubiera encontrado un tutor como tú, gracias por toda la ayuda y la paciencia y, por sobre todo, gracias por la amistad

A mis compañeros de aula, los que se fueron y los que aún continúan; a todos aquellos que me ayudaron con la tesis, Ricardo, Sayli, mi oponente Yosvany, gracias a todos.

Dedicatoria:

A mi mamá y a mi papá por su amor, por su preocupación, por su paciencia, su confianza en mí, por cada granito de arena que pusieron en mi vida y en mi futuro, para ustedes es este regalo. Un besote bien grande.

A mi hermano Randy, a mi abuela, a toda la familia que ha estado para apoyarme en todo momento.

A mis amigos, en especial a Esmerida que más que mi amiga es casi mi hermana, por todos los momentos buenos y malos que compartimos juntos.

A mi tutor Rufino por ser un gran amigo y ayudarme siempre.

A la Revolución por permitirme la educación y formación profesional, por hacerme una persona digna de estos tiempos.

Resumen

Los sistemas de Supervisión, Control y Adquisición de Datos (SCADA, según sus siglas en inglés) se utilizan principalmente en el entorno industrial. Una de las principales prestaciones dentro del módulo HMI es el entorno de configuración, este cuenta con una biblioteca gráfica en la cual se encuentran los diferentes componentes gráficos encargados de mostrar los valores recogidos del campo mediante los dispositivos, los cuales son utilizados para realizar los diferentes despliegues operacionales.

Este trabajo tiene como objetivo el desarrollo del componente gráfico de barras, para incorporarlo a los gráficos ya existentes: tendencia y XY, y así enriquecer la calidad visual de la representación de la información y lograr que esta sea lo más aproximada a la correlación estadística de los datos.

El desarrollo de este componente gráfico se basa en el uso de la metodología AUP, utilizando las técnicas de modelación establecidas en el lenguaje unificado de modelado (UML, según sus siglas en inglés). Al finalizar el desarrollo de este proyecto, la aplicación fue sometida a distintas pruebas, de las que se obtuvieron resultados satisfactorios.

PALABRAS CLAVES:

SCADA, HMI, gráfico de barras, correlación estadística.

Abstract

Systems Supervisory Control and Data Acquisition (SCADA) are used primarily in industrial environments. One of the main features in the HMI module is the configuration environment, this have a graphical library where are different graphical components in charge of show the values collected from the field by the devices, which are used for different operational deployments.

This work aims at development component: bar graph, to incorporate into already existing graphics: trend and XY, and so enhance the visual quality of the representation of information and make it more approximated the statistical correlation the data.

The development of this chart component is based on the use of UPA methodology, using modeling techniques established in the *Unified Modeling Language* (UML). To complete the development of this project, the application was subjected to various tests, for which satisfactory results were obtained.

Índice

Índice de tablas

Índice de ilustraciones

Introducción

La humanidad ha transitado por múltiples etapas de desarrollo, y en las últimas décadas se ha experimentado un incremento considerable en el desarrollo tecnológico, propiciando así cambios significativos en las esferas sociales y económicas. Con los constantes cambios se ha exigido una profunda investigación científica que impulse la creación de nuevos métodos y técnicas que favorezcan el equilibrio de estas esferas a nivel mundial.

El desarrollo industrial tiene su base en la automatización de los procesos industriales, regido por el uso de sistemas para controlar maquinarias y/o procesos industriales, sustituyendo a operadores humanos, lo que permite al hombre controlarlos y supervisarlos de forma más eficiente. En los inicios de la automatización las herramientas utilizadas eran bastante sencillas, al paso del tiempo han ido ganando complejidad y funcionalidad en forma exponencial. En la actualidad los procesos industriales se monitorean y supervisan mediante los sistemas de supervisión, control y adquisición de datos (SCADA, por sus siglas en inglés).

Un SCADA es una aplicación de software diseñada para el control y la supervisión de los procesos industriales a distancia, facilitando la retroalimentación en tiempo real de los dispositivos de campo y controlar automáticamente los procesos. Provee de toda la información que se genera en un proceso productivo, convirtiéndose en un instrumento decisivo para la toma de decisiones.

Debido a las diversas funciones que presentan los sistemas SCADA, algunos de estos deben de funcionar las 24 horas del día, los 7 días de la semana, ofreciendo una alta disponibilidad, como los utilizados en centrales nucleares o centrales eléctricas.

Cuba, como país en vía de desarrollo, no está exento de dicho desarrollo tecnológico y desde hace varios años realiza programas integrales en aras de fortalecer la automatización con la participación de centros universitarios e instituciones científicas.

La Universidad de las Ciencias Informáticas (UCI) cuenta con el Centro de Informática Industrial (CEDIN), el cual se encarga del desarrollo de proyectos para elevar los procesos de automatización de las empresas de todo el país y aportar productos informáticos de alto nivel para la exportación. Dentro de estos proyectos, uno de los más destacados es el SCADA SAINUX.

SAINUX cuenta con un módulo Interfaz Hombre Máquina (HMI) dividido en dos entornos: Entorno de Configuración (EC) y Entorno de Visualización (EV). El EC permite editar el entorno de trabajo, configurar la seguridad del sistema y definir los parámetros de las variables a supervisar, mientras que el EV es el encargado de visualizar la configuración antes realizada en el EC y permitirle al operador interactuar con el sistema, brindado una mejor calidad en los procesos.

Uno de los elementos más importante dentro del EC son sus componentes gráficos ya que son los que permiten la visualización de estado del proceso de despliegue y así posibilitan la interacción del operador con el sistema, estos componentes pueden ser objetos animados, gráficos, textos, listados, ventanas múltiples, entre otras.

Dentro de los gráficos con los que hoy cuenta SAINUX se destacan las tendencias y los gráficos XY por su alto grado de muestreo y cantidad de información que brinda a los operadores. Con la utilización de estos se puede tener una mejor monitorización de las actualizaciones y cambios de las variables del sistema según la recolección de los datos en el campo. Como componentes de visualización y muestreo, estos componentes resultan muy útiles, pero aún SAINUX presenta una **situación problemática**: no es posible resaltar una medición para lograr una representación gráfica más aproximada a la correlación estadística de los datos. De ahí nuestro **problema a resolver**: ¿Cómo contribuir a la representación gráfica de la correlación estadística de los datos en HMI SAINUX?

Teniéndose como **objeto de estudio**: los tipos de representación de la información, enfocado al **campo de acción**: la representación gráfica de la información en HMI. Con el **objetivo general** de desarrollar un componente gráfico de tipo barras para la interfaz HMI del SCADA SAINUX.

Se tiene como **idea a defender** que: con la incorporación del componente gráfico de barras al HMI SAINUX, se permitirá la representación gráfica de una variable en forma de barras, donde se ofrezca una visión en grupo permitiendo observar una preferencia, o tendencia, de las mediciones por ubicarse hacia una determinada región de valores dentro del espectro de valores posibles.

Para dar cumplimiento al objetivo trazado, se proponen las siguientes **tareas de investigación**:

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 las funcionalidades necesarias del componente gráfico de barras de la interfaz hombre-máquina, a partir del estudio de la interfaz hombre-máquina del ambiente nativo.

3. Definición de la arquitectura candidata para el componente gráfico de barras de la interfaz hombre máquina en el SCADA SAINUX.

4. Diseño e implementación de la solución según las características y requisitos identificados.

5. Realización de pruebas para comprobar el cumplimiento de los objetivos propuestos en la presente investigación.

Apoyadas en los siguientes **métodos investigativos**:

Métodos Teóricos.

Método histórico-lógico: Para la comprensión de los antecedentes y las tendencias actuales de los sistemas SCADA.

Método analítico-sintético: Para la definición de las funcionalidades necesarias del componente gráfico de barras HMI del SCADA SAINUX.

Métodos Empíricos.

Análisis documental: Para la elaboración del marco teórico de la investigación.

Experimento: Para la elaboración de un prototipo funcional del componente gráfico de barras del HMI del SCADA SAINUX.

La investigación tendrá la siguiente estructura:

Capítulo 1: Fundamentación teórica.

En este capítulo 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 el componente gráfico de barras en el SCADA SAINUX.

Capítulo 2: Análisis y diseño de la solución.

En este capítulo se da a conocer la propuesta del sistema y los diagramas para apoyar la comprensión del funcionamiento del mismo. Tomándose como base la metodología AUP para guiar el proceso de desarrollo.

Capítulo 3: Implementación y pruebas

En este capítulo se detallan algunos aspectos de la etapa de implementación del componente gráfico y los diagramas correspondientes al modelo de implementación, al igual que se detallan las pruebas realizadas al sistema y los resultados obtenidos de estas.

Capítulo 1: Fundamentación teórica.

1.1. Introducción

El capítulo aborda temas relacionados con los conceptos más importantes durante la investigación, así como las tecnologías y herramientas utilizadas para su diseño e implementación.

1.2. Automatización industrial

La automatización industrial se define como la aplicación de distintas tecnologías para controlar y monitorizar un proceso, máquina, aparato o dispositivo que por lo regular cumple tareas o funciones repetitivas, haciendo que operen automáticamente, reduciendo al mínimo la intervención humana.

La automatización industrial la encontramos en muchos sectores de la economía, como en la fabricación de alimentos, productos farmacéuticos, productos químicos, en la industria gráfica, petrolera, automotriz, plásticos, telecomunicaciones, entre otros sectores en los cuales generan grandes beneficios. No solo se aplica a máquinas o fabricación de productos, también se aplica a la gestión de procesos, de servicios, al manejo de la información, a mejorar cualquier proceso que conlleven a un desempeño más eficiente, desde la instalación, mantenimiento, diseño, contratación e incluso la comercialización. (1)

1.3. Sistemas SCADA

Los sistemas de Supervisión, Control y Adquisición de Datos (SCADA, según sus siglas en inglés) consisten en una aplicación software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etc.) y controlando el proceso de forma automática desde la pantalla del ordenador. Además, provee de toda la información que se genera en el proceso productivo a diversos usuarios, tanto del mismo nivel como de otros supervisores dentro de la empresa: control de calidad, supervisión, mantenimiento, etc. (2)

Los sistemas SCADA en la actualidad tienen un incontable número de aplicaciones dentro de las que se pueden mencionar: el control de oleoductos, sistemas de transmisión de energía eléctrica, yacimientos de gas y petróleo, redes de distribución de

gas natural, subterráneos, generación energética, sistemas de distribución de agua, entre otros. (3)

1.3.1. Funcionalidades de un sistema SCADA

Las funcionalidades principales de un Sistema SCADA son: (4)

- Adquisición y almacenamiento de datos: Para la recolección, procesamiento y almacenamiento de la información recibida, en forma continua y confiable.
- Supervisión de procesos: Para el monitoreo del funcionamiento del proceso, procesamiento estadístico de los datos y confección de reportes. Además se brindan notificaciones al operador del sistema sobre cambios detectados en la instalación. Estas notificaciones se clasifican principalmente en alarmas y eventos. Las alarmas se basan en la vigilancia de los parámetros de las variables del sistema. Son los sucesos no deseables, porque su aparición puede dar lugar a problemas de funcionamiento. Este tipo de sucesos requieren de la atención de un operario para su solución antes de que se llegue a una situación crítica que detenga el proceso. El resto de las situaciones normales, tales como puesta en marcha, paro, cambios de consignas de funcionamiento, consultas de datos, entre otras, serán los denominados eventos del sistema o sucesos. Los eventos no requieren de la atención del operador del sistema, registran de forma automática todo lo que ocurre en el sistema. También será posible guardar estos datos para su posterior consulta.
- Control de procesos: Para el control de los procesos de la fábrica, planta o industria, actuando sobre los reguladores autónomos básicos como eventos y alarmas, o directamente sobre el proceso mediante las salidas conectadas.
- Transmisión de datos: Para la transmisión de información entre dispositivos de campo y computadoras de control o entre dispositivos ubicados a un mismo nivel.
- Presentación de la información: Para la representación gráfica de los datos.

1.3.2. Módulos de SCADA

Un SCADA se compone de bloques o módulos independientes que intervienen en su funcionamiento y permiten las actividades de supervisión, control y adquisición.

Entre los módulos que componen un SCADA están: (5)

- Bases de Datos Históricos: Es el encargado de almacenar la información recibida desde el campo, así como la sucesión de alarmas y eventos. Esta

información es de vital importancia para realizar cualquier tipo de análisis posteriores como diagnósticos o reportes. Debe permitir a los usuarios y ejecutivos el análisis rápido de información histórica, la que debe estar organizada según patrones de comportamiento de las plantas y dar datos relevantes de todo el proceso industrial.

- Comunicaciones o Middleware: Este módulo representa la capa de software encargada de la comunicación entre los diferentes procesos distribuidos, de mediano y alto nivel. Enlaza todos los componentes del sistema y tiene relación directa con el canal de comunicación. Su arquitectura es compleja en la mayoría de los casos, pero presenta pequeñas interfaces que garantizan rapidez y robustez en la transmisión de mensajes. Es uno de los módulos horizontales al sistema y sus cambios suelen afectar el funcionamiento general del mismo. (6)
- Seguridad: Permite a los usuarios autenticarse en el sistema, y de esta forma acceder solo a los recursos que tiene asignado su rol. Posee herramientas para la protección ante ataques piratas, fallos eléctricos, problemas de red, entre otros. Su objetivo es garantizar la accesibilidad, integridad y confidencialidad de la información manejada por el sistema, lo cruza horizontalmente, y se asocia de forma discreta, a todos los módulos del sistema.
- Procesamiento de Datos: Este módulo representa el "centro neurológico" del sistema, el cual supervisa y recoge la información del resto de las subestaciones, bien sean otros ordenadores conectados (en sistemas complejos) a los instrumentos de campo o directamente sobre dichos instrumentos,
- Ambiente de configuración: Permite configurar varios procesos o parte de ellos. En este módulo se gestionan las variables, y se realiza la configuración de manejadores y comandos internos del sistema. Sirve como capa abstracta que garantiza la optimización y coherencia entre los componentes, permite la direccionalidad mutua entre elementos internos y les ofrece una interfaz sencilla para su comunicación.
- Interfaz Hombre-Máquina o HMI: Interfaz que comunica al sistema con sus operadores

1.4. Módulo HMI

En un SCADA el módulo HMI es el responsable de representar, en el ordenador, los procesos que ocurren en el campo en tiempo real. Muestra las estaciones remotas, los sensores, los componentes y el sistema de comunicación implicado en el proceso de producción garantizando un control total. Las señales de los procesos son trasladadas al HMI por medio de dispositivos como tarjetas de entrada/salida en la computadora, controladores lógicos programables (PLC, por sus siglas en inglés), unidades terminales remotas de entrada y salida o manejadores. Todos estos dispositivos deben tener una comunicación que entienda el HMI. (5)

1.4.1. Funcionalidades del HMI

Para gestionar un proceso industrial de forma efectiva se debe poseer una interfaz gráfica que garantice una visión real del proceso y un conjunto de funcionalidades que permitan su control y supervisión. Dentro de las funcionalidades que debe brindar un HMI se encuentran: (5)

- **Monitoreo:** es la habilidad de mostrar los datos del proceso en tiempo real.
- **Supervisión:** es la funcionalidad que junto al monitoreo permite ajustar las condiciones de trabajo del proceso directamente desde la computadora.
- **Alarmas:** es la capacidad de reconocer eventos excepcionales dentro del proceso y reportarlos. Las alarmas son reportadas a partir de límites de control preestablecidos.
- **Control:** es la capacidad de aplicar algoritmos que ajustan los valores del proceso y mantenerlos dentro de ciertos límites.
- **Históricos:** es la capacidad de muestrear y almacenar en un archivo datos del proceso a una determinada frecuencia. Es una poderosa herramienta para la optimización y corrección de procesos.

1.4.2. Entorno de configuración del HMI.

Es el ambiente que permite configurar los procesos, gestionar las variables, editar los controladores, los comandos, las alarmas y demás opciones para la supervisión y control. El editor de configuración permite definir el ambiente de trabajo del SCADA, adaptando mejor la aplicación al proceso que se desea supervisar. (7)

Funcionalidades del entorno de configuración.

Los HMI en sistemas SCADA cuentan con funcionalidades para realizar la configuración, permitiendo a los operadores definir el entorno de trabajo para adaptarlo a sus necesidades.

Entre las funcionalidades que brinda un editor de configuración se encuentran: (7)

- Administración de los usuarios que accederán al sistema, clasificándolos según sus niveles de importancia.
- Creación, modificación y organización de los recursos necesarios para la configuración de la aplicación definida. Gestionar los módulos del SCADA, permitiendo definir la ubicación en nodos físicos.
- Configuración de las comunicaciones, dando la facilidad para especificar los dispositivos que se utilizarán para recolectar los valores a monitorizar, además de los parámetros que estos necesitarán para su correcto funcionamiento.
- Creación de pantallas con imágenes y texto que simulen el proceso a controlar, brindando a los usuarios una mejor comprensión del proceso.

Edición Gráfica

La edición gráfica es el mecanismo que permitirá al operador del SCADA crear sinópticos de procesos industriales. Los sinópticos se representan a partir de objetos y primitivas básicas predefinidas, que se pueden agrupar, combinar, transformar, importar o exportar. El editor gráfico es el componente que permite la creación y edición de los objetos y despliegues presentes en los escenarios productivos. Para ello emplea un conjunto de herramientas entre las que se pueden encontrar:

- **Paleta de objetos:** Es la herramienta que contiene los objetos gráficos. Un objeto gráfico es un componente de control que presenta un conjunto de propiedades que pueden ser editadas y asociadas a variables del SCADA. Por medio de los objetos gráficos se crean animaciones en función de los valores de las variables asociadas. (8)
- **Área de edición:** Es el área de la aplicación donde se crean los despliegues y los reportes, seleccionando los elementos gráficos que se brindan en la paleta de objetos. Permite acceder a las propiedades de los elementos gráficos para construir las pantallas que se muestran en tiempo de ejecución del sistema. (9)
- **Barra de menú:** Contiene todas las acciones para el diseño gráfico.

- **Inspector de propiedades:** Los elementos gráficos y recursos en el sistema tienen propiedades que se deben modificar y explorar por los operadores de la aplicación. El inspector de propiedades es la vista que permite ver las propiedades. (9)
- **Paleta de Colores:** Permite asignar colores a los objetos seleccionados. Además de los 16 colores estándar, también puede utilizarse los colores personalizados que se hayan definido. (10)
- **Barra de herramientas:** Contiene los botones que permiten realizar rápidamente las acciones más frecuentes en el editor gráfico, entre las que se encuentran copiar, cortar, pegar, deshacer, rehacer , enviar al fondo , traer al frente, acercar y alejar un objeto gráfico. (10)

1.4.3. Ambiente de ejecución del HMI.

El ambiente de ejecución se encarga de visualizar las animaciones y los objetos definidos en el ambiente de configuración. Muestra lo que está ocurriendo en el proceso productivo en tiempo real, es el que envía los comandos a las estaciones remotas, quien recibe los valores de las variables e interactúa con la mayoría de los operadores pues se emplea para supervisar el proceso de manera directa. Al ser el HMI el módulo que se encarga de brindar el control total sobre el proceso de producción, debe facilitar un conjunto de funcionalidades primarias, entre ellas la generación de reportes, impresión, análisis de variables, visualización de la tendencia de indicadores, configuración de los manejadores para la comunicación y acceso a las alarmas. (11)

1.5. Tipos de representación de la información histórica

Los sistemas SCADA cuentan con dos tipos de representación histórica que permiten visualizar el estado de sus variables y facilitar la toma de decisiones, estos son la representación tabular y la representación gráfica, los que se describen a continuación.

1.5.1. Representación tabular

Un modelo tabular es una representación lógica de tablas y relaciones con fines analíticos; el modelo también incluye otras características, como jerarquías de atributos, para proporcionar una experiencia de resumen y exploración en profundidad más completa, como perspectivas, para simplificar o centrar el modelo en parte menor de él, como los indicadores clave de rendimiento y muchas características incluidas. (12)

Es un ordenamiento de la información en filas y columnas. Una buena representación tabular debe tener:

- Títulos y encabezamientos claros y completamente definidos.
- Incluir las unidades en que se expresa la medición.
- Incluir la suficiente información que permita chequear la validez de los cálculos o argumentos.
- Incluir fuente de datos cuando corresponda.

La información de los datos históricos se representa de forma tabular y gráfica. La siguiente ilustración muestra una representación tabular de un sumario de histórico de eventos:

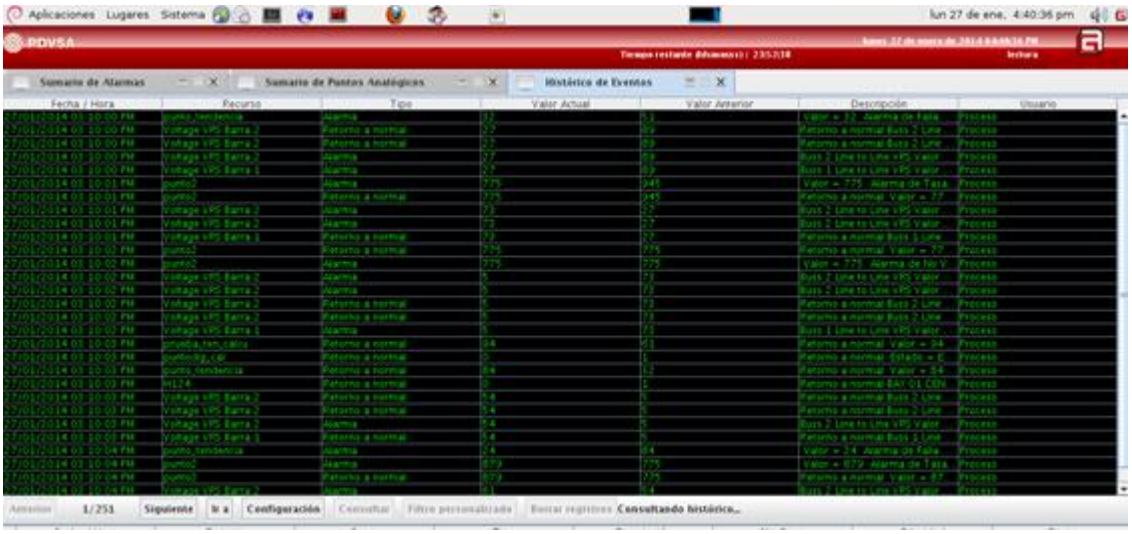


Ilustración 1 Representación tabular

1.5.2. Representación gráfica

Las gráficas o gráficos son las denominaciones de la representación de datos, generalmente numéricos, mediante recursos gráficos (líneas, vectores, superficies o símbolos), para que se manifieste visualmente la relación matemática o correlación estadística que guardan entre sí. También es el nombre de un conjunto de puntos que se plasman en coordenadas cartesianas y sirven para analizar el comportamiento de un proceso o un conjunto de elementos o signos que permiten la interpretación de un fenómeno. La representación gráfica permite establecer valores que no se han obtenido experimentalmente sino mediante la interpolación (lectura entre puntos) y la extrapolación (valores fuera del intervalo experimental). (13)

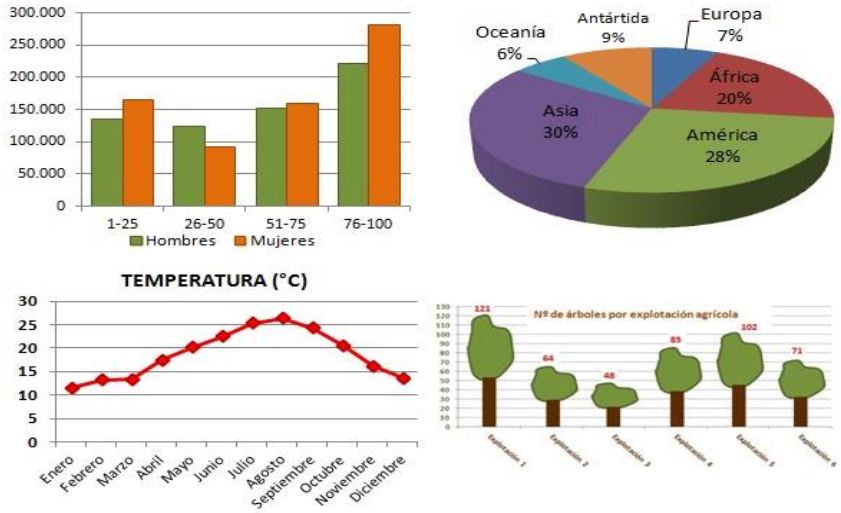


Ilustración 2 Representación gráfica

Tipos de representación gráfica

En un SCADA existen distintas formas de representación gráfica, los más usados son:

Gráficos de tendencia:

Esta herramienta permite analizar el comportamiento de los puntos según su variación en el tiempo.

El concepto de tendencia es simple en su modo más elemental: es la corriente o preferencia hacia determinados fines, o de forma más científica, es un patrón de comportamiento de ciertos elementos de un entorno particular durante un período determinado. Ahora, desde diversos campos, la tendencia asume definiciones especiales, como el provisto desde el punto de vista de análisis técnico en marketing: que lo acota como la dirección o rumbo del mercado. Dentro de la computación técnica y análisis de datos la tendencia es la dirección o comportamiento preferible de ciertos datos en un rango de tiempo dado. (14)

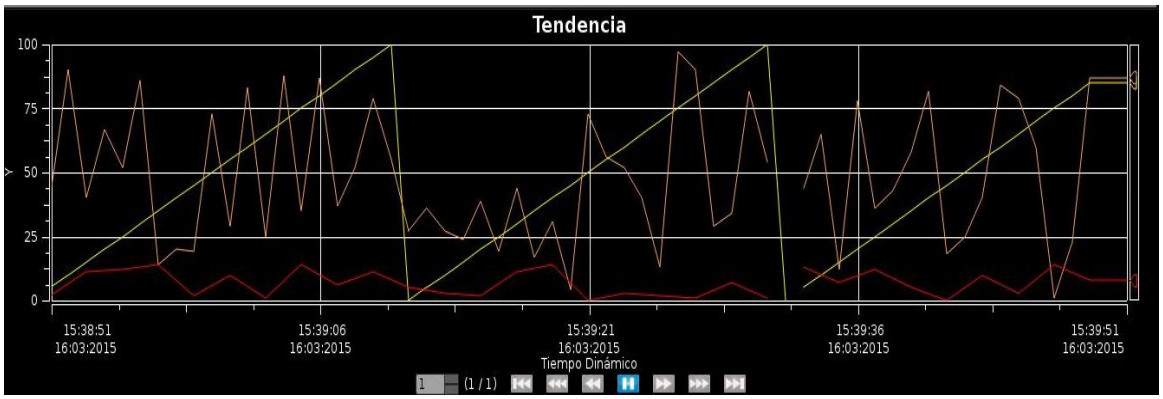


Ilustración 3 Gráfico de tendencia

A pesar de los diferentes conceptos, las tendencias son similares y determinan una línea creciente o decreciente con respecto a algo, delimitando que cierto conjunto de datos o comportamientos se alejen o acerquen a un punto de interés para quien analiza, estableciendo que el concepto, parta del principio matemático de límite, agregando patrones aleatorios a la curva que varía su comportamiento y a la vez representa el conjunto analizado. (15)

Gráficos XY:

Los gráficos XY o de dispersión muestran las series como un conjunto de puntos. Los valores se representan mediante la posición de los puntos en el espacio del gráfico. Las categorías, por su parte, mediante diferentes puntos del gráfico. Los gráficos de dispersión suelen utilizarse para comparar valores distintos de las categorías. (16)

Este se caracteriza por su impacto visual, ya que muestra la posibilidad de la existencia de correlación estadística entre dos variables de un vistazo, este proporciona mayor información que un simple análisis matemático, sugiriendo posibilidades y alternativas al estudio.

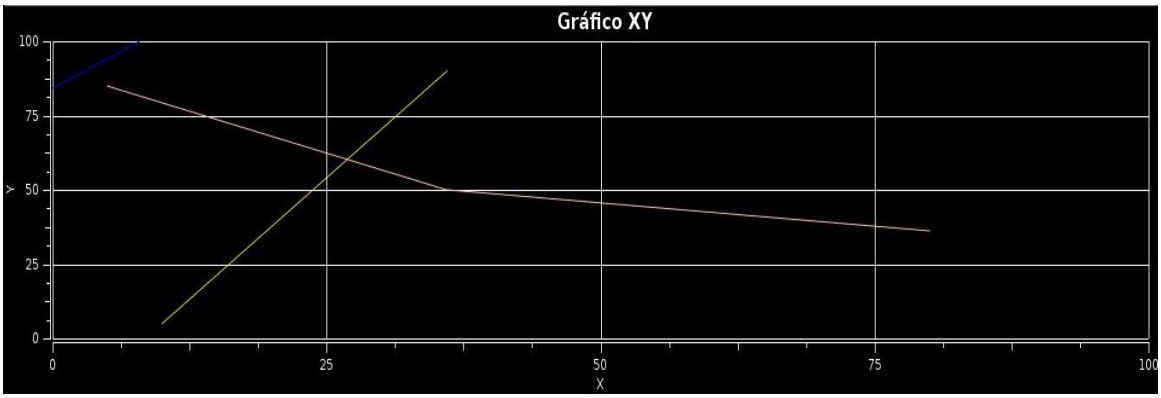


Ilustración 4 Gráfico XY

Gráficos de barras

Un gráfico de barras es aquella representación gráfica bidimensional en que los objetos gráficos elementales son un conjunto de rectángulos dispuestos paralelamente de manera que la extensión de los mismos es proporcional a la magnitud que se quiere representar. Los rectángulos o barras pueden estar colocados horizontal o verticalmente. En éste último caso reciben también el nombre de gráficos de columnas. Se emplea para representar de manera gráfica la información que se ha recolectado. El tipo de datos que se representa en una gráfica de barras es el número de eventos que son medidos en distintas categorías de datos. Una gráfica de barras usualmente se utiliza para representar datos que se han organizado en una tabla de datos. Se recomienda utilizar este tipo de gráfica cuando la información corresponda a una serie de eventos (escala nominal) y cuando quiera comparar dos o más grupos entre sí. (17)

Los gráficos de barras tienen como ventaja:

- **Representación sencilla de los datos:** Un gráfico de barras representa datos, o un conjunto de ellos, de una manera esquemática. Los gráficos de barras son útiles en la representación de datos con unidades distintas, tales como los años y meses. También son valiosos para mostrar diferencias o hacer comparaciones entre diferentes variables. Los gráficos de barras son útiles cuando es necesario demostrar variables contables, hechos y datos. Los gráficos de barras muestran estos y otros valores comparativos de una manera distintiva y completa, dando una imagen más clara y comprensible de la distribución de los datos.
- **Representa la frecuencia:** Las gráficas de barras son el método más popular de visualización de datos, analizando los datos en gráficos y mostrando la

frecuencia relativa o la frecuencia para cada grupo por separado (que suele ser representado por la altura de una barra). (18)

- **Fácil de hacer:** Los gráficos de barras son fáciles de elaborar, siempre y cuando se hayan recolectado y preparado los datos apropiados. Un gráfico de barras completo requiere un título, etiquetas y una escala. Se necesitan tres pasos para hacer un gráfico de barras: determinar las categorías o los valores que se van a colocar a lo largo de los ejes horizontal y vertical, hacer la escala que se va a utilizar para determinar los datos numéricos y establecer el tipo, el estilo y la longitud de cada barra. (19)
- **Versátil y ampliamente utilizado:** Hay muchos tipos de gráficos de barras que pueden utilizarse para diversos fines. Los gráficos de barras se utilizan en diversos entornos educativos, industriales, comerciales y de negocios para mostrar comparaciones entre valores, tales como el número de estudiantes con una calificación de aprobado o el número de ventas realizadas durante un período de tiempo determinado, etc.

En la actualidad, el SCADA SAINUX cuenta con los gráficos de tendencia y XY, por lo que este trabajo se centra en el desarrollo del componente gráfico de barras.

1.6. Metodología de desarrollo de software

La UCI cuenta con 14 centros productivos en su gran mayoría pertenecientes a las facultades que conforman a la Universidad. Cada uno de estos centros se dedica al desarrollo de software y/o servicios asociados a un dominio de aplicación bien definido. Esta diversidad de centros y proyectos hace que la actividad productiva en la UCI sea cada vez más amplia, y trae consigo la heterogeneidad en el proceso de desarrollo de software. En estas actividades productivas se utilizan una amplia variedad de metodologías, tanto ágiles como robustas y se ha comprobado que muchos proyectos no lo usan en su totalidad. (20)

Para erradicar los errores encontrados se decidió converger a una metodología que cubra las particularidades de cada una de las ya aplicadas. Esta metodología escogida se adapta 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. Esta metodología propuesta es una variación al Proceso Unificado Ágil.

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este 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. El AUP aplica técnicas ágiles incluyendo:

- Desarrollo Dirigido por Pruebas (test driven development - TDD en inglés)
- Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad.

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva. Estas fases son:

- 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.
- Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- 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.

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

I. 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.

II. 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.

III. 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.

IV. Despliegue. El objetivo de esta disciplina es la prestación y ejecución del sistema y que el mismo este a disposición de los usuarios finales.

V. 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.

VI. 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.

VII. 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. Herramientas y tecnologías

El desarrollo de software no fuera posible sin las herramientas y tecnologías que le dan soporte, pues de no existir estas el trabajo sería engorroso, tedioso y muy lento para la obtención de los resultados. Las tecnologías y herramientas a utilizar para llevar a cabo la propuesta de solución son las mismas que para el desarrollo del software al que el componente fue integrado, éstas se describen a continuación:

1.7.1. Lenguaje de modelado (UML)

Se trata de un estándar adoptado por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software. Son una serie de normas y estándares que dicen cómo se debe representar algo. UML incluye aspectos conceptuales como los procesos de negocio y funciones del sistema, y aspectos concretos como son expresiones de lenguajes de programación, componentes reutilizables y esquemas de bases de datos (21). Para la solución propuesta se utilizará UML 2.0.

1.7.2. Herramienta de modelado (Visual Paradigm)

Las herramientas de modelado se emplean para la creación de modelos de sistemas que ya existen o que se desarrollarán. Permiten crear un simulacro del sistema.

Visual Paradigm es una herramienta CASE: Ingeniería de software asistida por computación. Está concebida para soportar el ciclo de vida del desarrollo del software a través de la representación de todo tipo de diagramas. Se caracteriza por: disponibilidad, uso de un lenguaje estándar, varios idiomas, fácil de instalar y desinstalar, soporta aplicaciones web, importación y exportación de ficheros y editor de figuras. (22)

1.7.3. Lenguaje de programación (C++)

Los leguajes de programación permiten comunicarnos e instruir a la computadora para que realice una tarea específica. Cada lenguaje de programación posee una sintaxis y un léxico particular, es decir, forma de escribirse que es diferente en cada uno por la forma que fue creado y por la forma que trabaja su compilador para revisar, acomodar y reservar el mismo programa en memoria.

C ++ es un lenguaje de programación orientado a objetos y fue creado en la década de los 80 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos y en ese sentido es considerado como un lenguaje híbrido. Es rico en operadores y expresiones y presenta programación modular, además es flexible, conciso, eficiente, portable y breve. (23)

1.7.4. IDE de desarrollo (QtCreator)

Qt Creator se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y comenzar a desarrollar rápidamente, QTCreator cuenta con las siguientes características: (24)

- Editor de código con soporte para C++
- Herramientas para la rápida navegación del código
- Resaltado de sintaxis y auto-completado de código
- Control estático de código y estilo a medida que se escribe
- Ayuda sensitiva al contexto
- Plegado de código (code folding)
- Paréntesis coincidentes y modos de selección.

1.8. Conclusiones parciales

La elaboración del marco teórico brindó un acercamiento a los principales conceptos, técnicas y métodos a utilizar durante el resto de la investigación, lo que permitió profundizar los aspectos teóricos de sistemas SCADA, sus interfaces hombre-máquina y el análisis de las formas de representación de la información, propiciando una base de conocimientos útil para el desarrollo de la investigación. Además, la indagación sobre las técnicas y herramientas consolidaron las propuestas tecnológicas necesarias para la realización del componente, lográndose una fundamentación técnica que justifica la solución escogida.

Capítulo 2: Análisis y diseño de la solución

2.1. Introducción

En este capítulo se describen las actividades desarrolladas durante todo el proceso de análisis y diseño de la solución. Se define una lista de requisitos funcionales y no funcionales que se deben tener en cuenta para la elaboración del componente. Se analizan los conceptos fundamentales del modelo de dominio necesarios para comprender el problema planteado y se describen las principales clases.

2.2. Modelo de dominio

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

2.2.1. Diagrama de modelo de dominio

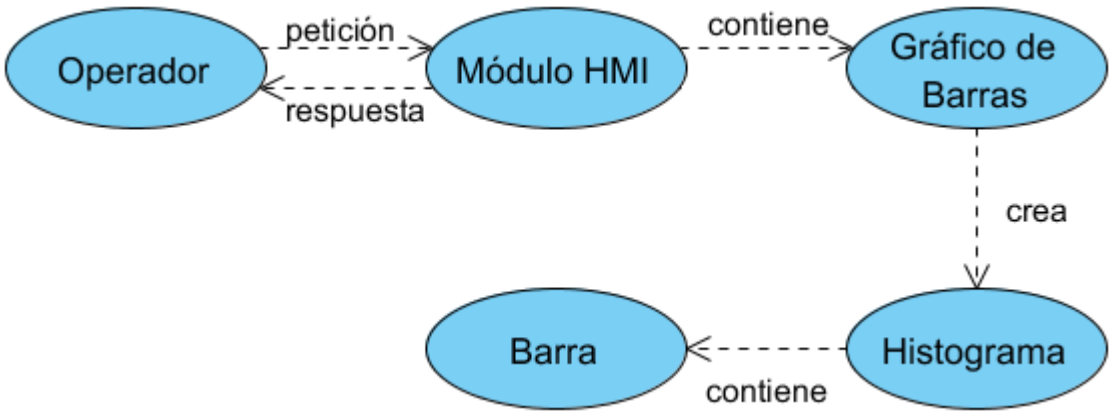


Ilustración 5 Diagrama de modelo de dominio

2.2.2. Descripción del modelo de dominio

A continuación se describen el flujo del sistema para lograr un mejor entendimiento de la lógica de negocio:

El **operador** comienza el proceso haciéndole una petición a la interfaz hombre-máquina en espera de la respuesta deseada. **HMI** contiene a su vez el **gráfico de barras** que se desea visualizar, este en conjunto con las configuraciones del usuario crea un **histograma** que contiene uno o más barras, estas **barras** responden a la representación gráfica de los valores de las mediciones del sistema, anteriormente seleccionadas por el usuario.

2.3. Captura de requisitos.

Un requisito es una "condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado". También se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación.

2.3.1. Requisitos funcionales.

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir.

Tabla 1 Requisitos funcionales

Requisitos funcionales	Descripción
RF 1: Visualizar componente	Permite visualizar el componente al hacerle *drop*[1] sobre el despliegue.
RF 2: Adicionar barra	Permite adicionar una barra al componente.
RF 3: Eliminar barra	Permite eliminar una barra seleccionada del componente.
RF 4: Modificar color de la barra	Permite modificar el color asignado a la barra seleccionada.
RF 5: Mostrar propiedades	Permite visualizar las propiedades a configurar para visualizar el componente

[1] Acción de soltar

	en el despliegue.
RF 6: Configurar el área de pintado	Permite configurar el área de pintado del componente
RF 7: Configurar rejillas	Permite configurar las rejillas del componente.
RF 8: Configurar eje X	Permite configurar el eje X del componente
RF 9: Configurar eje Y	Permite configurar el eje Y del componente
RF 10: Configurar propiedades generales	Permite configurar todas las propiedades comunes de los gráficos de HMI: alto, ancho, posición, visible, entre otras.

2.3.2. Requisitos no funcionales

Un requisito no funcional (RNF) especifica los criterios que se deben usar para juzgar el funcionamiento de un sistema, en lugar de un comportamiento específico. En general, los requisitos no funcionales verifican cómo un sistema debe de ser. Se definen los siguientes requisitos no funcionales:

Requisito de usabilidad

RNF 1: Forma de interacción de la interfaz: Basada en ventanas.

RNF 2: Interfaz de visualización de operaciones y sus atributos: La interfaz debe permitir efectuar todas las actividades de supervisión.

Requisitos de confiabilidad

RNF 3: Debe existir una comunicación eficiente y fiable, evitando que no existan pérdidas de información de variables y eventos.

RNF 4: Integridad de las configuraciones o parámetros: El sistema debe ser capaz de mantener la integridad de las configuraciones o parámetros de las tareas, así como, los scripts asociados a ellas.

RNF 5: Manejo de datos en memoria: Ante una solicitud de fin de ejecución del sistema, se deben procesar todas las tareas que cumplieron la condición para su procesamiento.

Requisitos de desempeño

RNF 6: Debe garantizar procesos y transacción masiva de datos: Hacia y desde todos los nodos y sistemas asociados, sin efectos negativos sobre el rendimiento del sistema.

RNF 7: Tiempo de respuesta en las consolas máximo de dos (2) segundos: Toda solicitud del operador debe presentar al menos una respuesta parcial en las consolas en un tiempo inferior a 2 segundos. En particular el tiempo para cambiar dos ventanas debe ser inferior a 1 segundo.

RNF 8: Tiempo de respuesta para adquisición y procesamiento de los datos: (asociado a la velocidad de respuesta del dispositivo de control) 500 milisegundos, referido al periodo de encuesta mínimo o frecuencia máxima de muestreo.

RNF 9: Tiempo para el refrescamiento de datos en el componente gráfico de 500 milisegundos.

RNF 10: Durante el intercambio de variables, debe existir una comunicación eficiente, garantizando una velocidad en la comunicación adecuada.

2.4. Historia de usuario

Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de requisitos, son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos, también permiten responder rápidamente a los requisitos cambiantes.

2.4.1. Descripción de las historias de usuario

Tabla 2 HU Visualizar componente

Historia de Usuario	
Número: HU1	**Nombre del requisito:** Visualizar componente
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 10

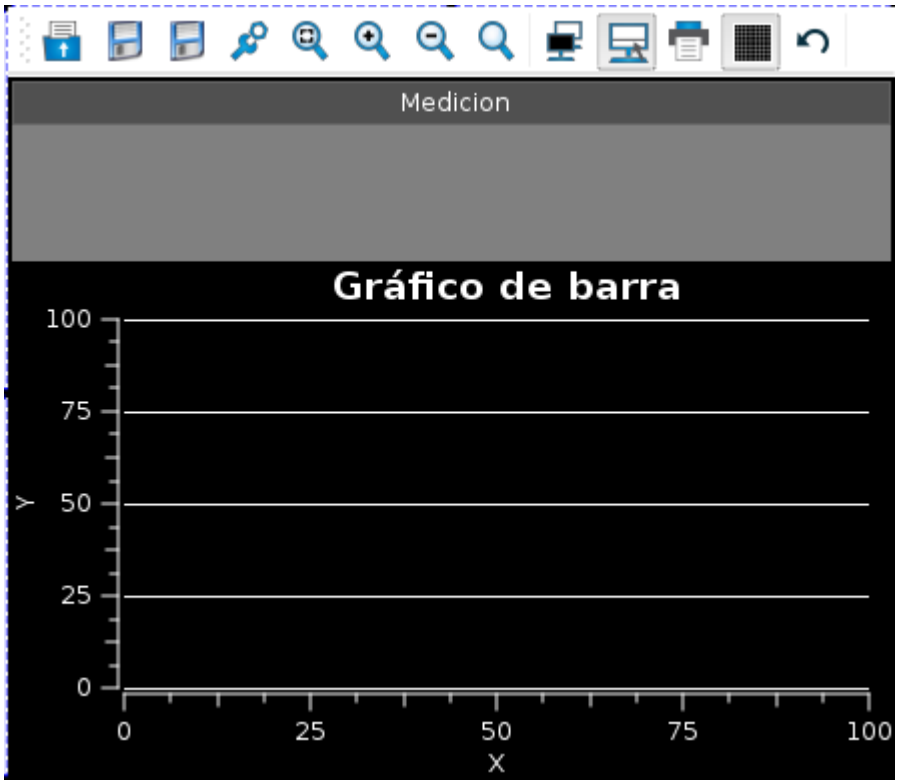
Riesgo en desarrollo: ○ Problemas eléctricos. ○ Problemas técnicos.	**Tiempo real:** 2 semanas.
Descripción: El operador será capaz de adicionar el componente gráfico de barras a un despliegue anteriormente. Para esto el operador debe seleccionar el Gráfico de Barras de la paleta de componentes y hacerle *drop* dentro del despliegue.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 3 HU Adicionar barra

Historia de usuario	
Número: HU2	**Nombre del requisito:** Adicionar barra
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 5 semanas.
Riesgo en desarrollo: ○ Problemas eléctricos. ○ Problemas técnicos.	**Tiempo real:** 1 semanas.

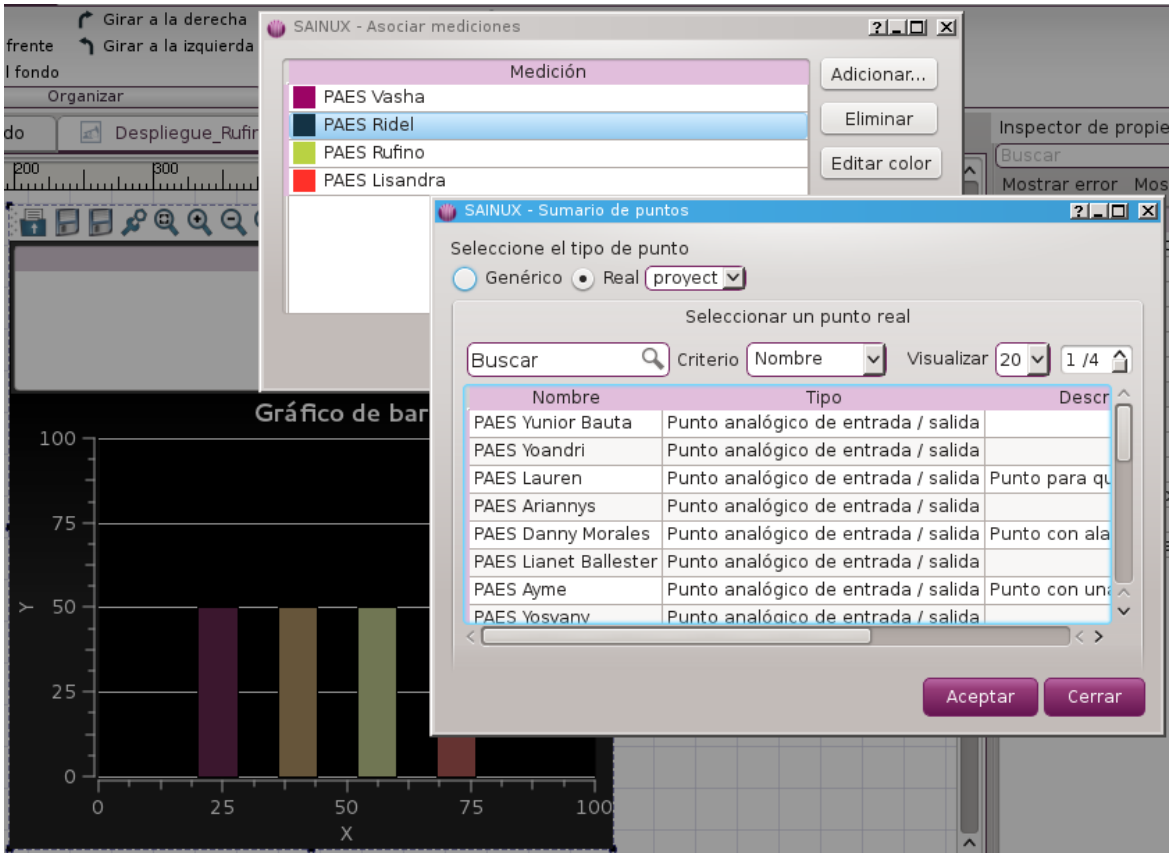
Descripción: El operador será capaz de agregar una medición a la gráfica ya existente en el despliegue, cada medición se representa a través de una barra. Para adicionar una barra se accede a la propiedad Histogram del inspector de propiedades, luego acceder a Mediciones. Se visualizará una ventana que presentará la opción de agregar una medición, dando clic izquierdo en el botón correspondiente, aparece otra ventana que dará la opción de agregar una medición ya existente o crear una medición genérica. Ya agregada la medición, se le asigna automáticamente un color a la barra que le corresponde.
Observaciones:
Prototipo de interfaz: 

Tabla 4 HU Eliminar barra

Historia de usuario	
Número: HU3	**Nombre del requisito:** Eliminar barra
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 5 días.
Riesgo en Desarrollo:	**Tiempo real:** 1 semana.

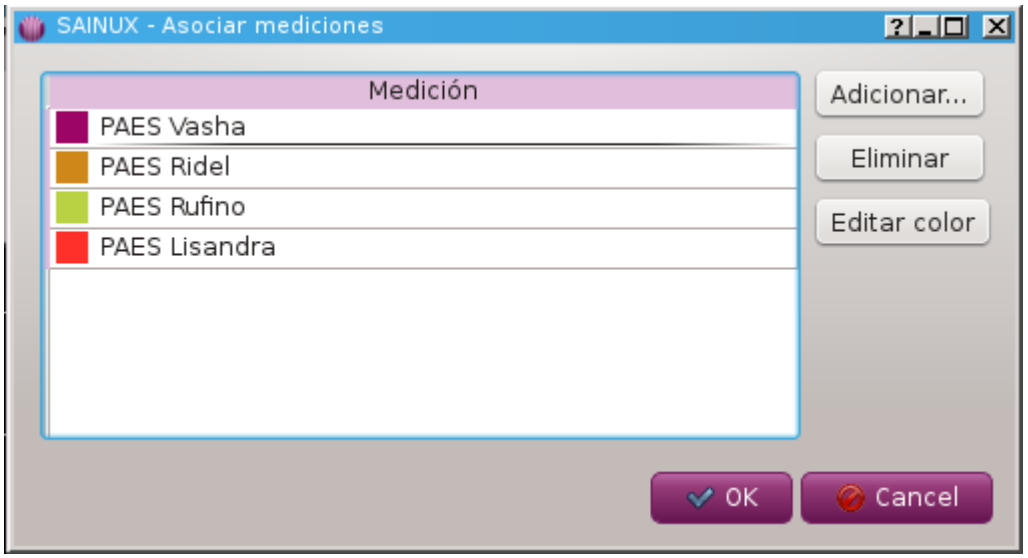
○ Problemas eléctricos. ○ Problemas técnicos.	
Descripción: El operador será capaz de eliminar una medición existente de la gráfica en el despliegue. Para eliminar una barra se accede a la propiedad Histograma del inspector de propiedades, luego acceder a Mediciones. Se visualizará una ventana que presentará la opción de eliminar una medición, se selecciona la medición que se quiere eliminar del gráfico y dando clic izquierdo en el botón correspondiente esta se elimina.	
Observaciones:	
Prototipo de interfaz: 	

Tabla 5 HU Modificar color de la barra

Historia de usuario	
Número: HU4	**Nombre del requisito:** Modificar color de la barra
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 2 semanas.
Riesgo en desarrollo: ○ Problemas eléctricos. ○ Problemas técnicos.	**Tiempo real:** 2 semanas.
Descripción: El operador será capaz de modificar el color de una barra representada en el componente. Para modificar una barra se accede a la propiedad Histograma del inspector de propiedades, luego acceder a Mediciones. Se visualizará una ventana que presentará la opción de editar el color, se da clic	

izquierdo en el botón correspondiente y se visualizará una ventana en la que se seleccionará el color deseado.

Observaciones:

Prototipo de interfaz:

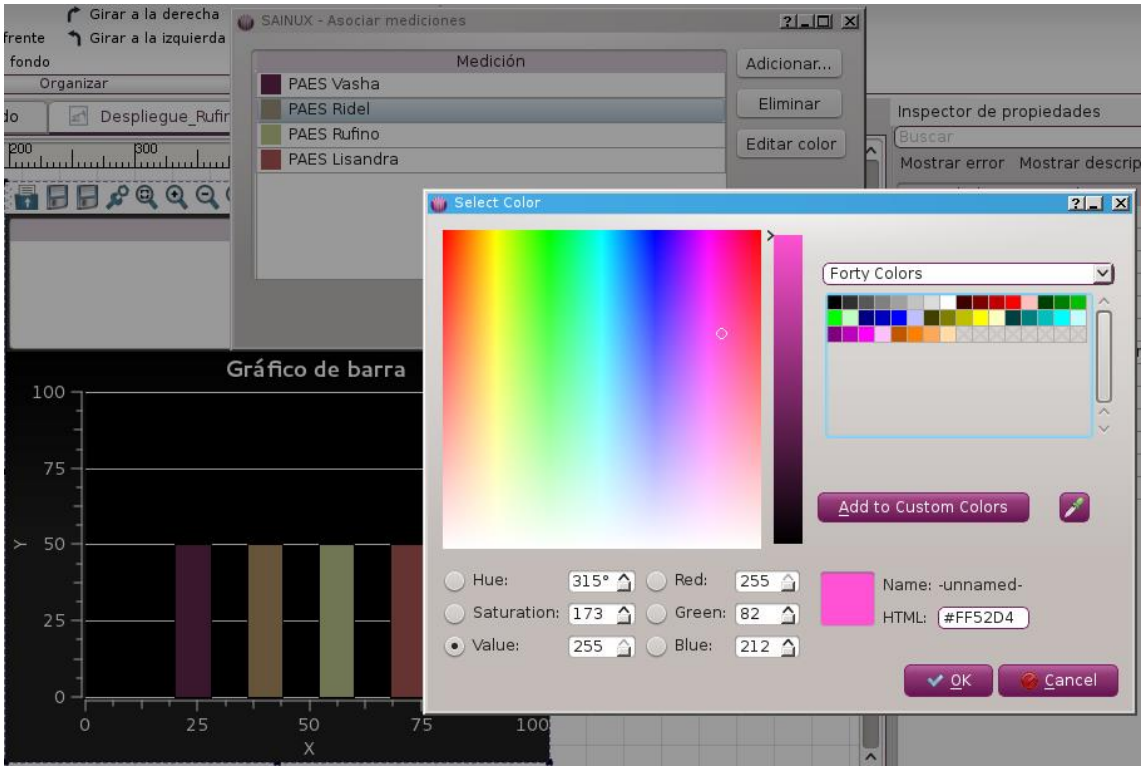


Tabla 6 HU Mostrar propiedades

Historia de usuario	
Número: HU5	**Nombre del requisito:** Mostrar propiedades
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 3 días.
Riesgo en desarrollo: o Problemas eléctricos. o Problemas técnicos.	**Tiempo real:** 1 semanas.
Descripción: El operador será capaz de visualizar las propiedades del componente. Para ello debe dar clic sobre el componente y se mostrará el Inspector de propiedades de este.	
Observaciones:	
Prototipo de interfaz:	

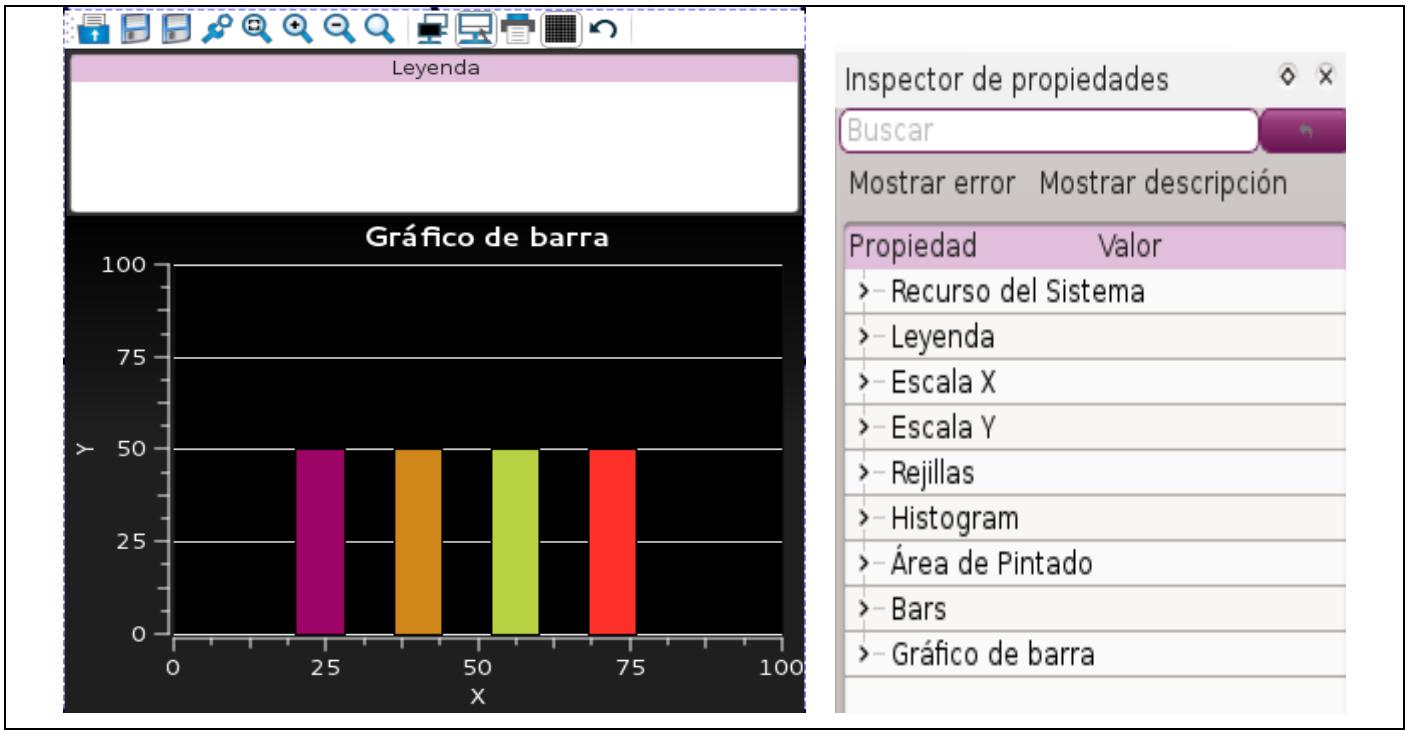


Tabla 7 HU Configurar el área de pintado

Historia de usuario	
Número: HU6	**Nombre del requisito:** Configurar el área de pintado
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 4 días.
Riesgo en desarrollo: ○ Problemas eléctricos. ○ Problemas técnicos.	**Tiempo real:** 1 semanas.
Descripción: El operador será capaz de editar el área de pintado del componente dependiendo de sus necesidades o especificaciones. Para esto debe acceder en el Inspector de Propiedades a la propiedad correspondiente a dicha área, este presenta la posibilidad de editar el ancho y el relleno.	
Observaciones:	
Prototipo de interfaz:	

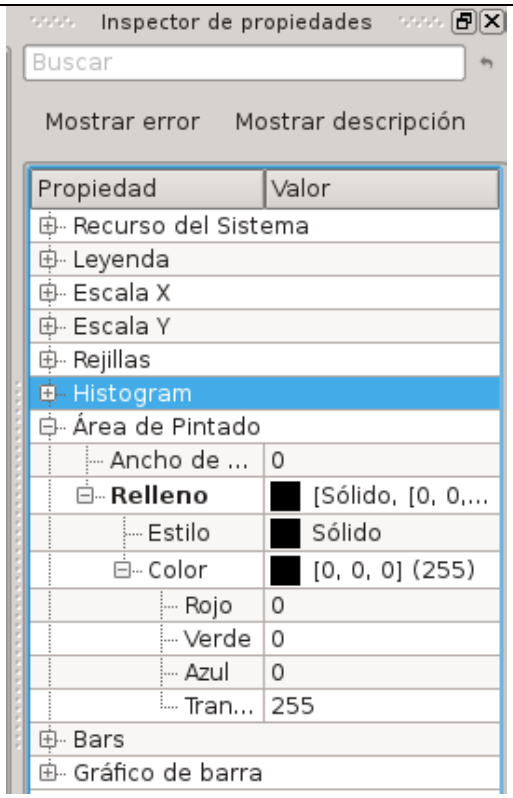


Tabla 8 HU Configurar rejillas

Historia de usuario	
Número: HU7	**Nombre del requisito:** Configurar rejillas
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 4 días.
Riesgo en desarrollo: ○ Problemas eléctricos. ○ Problemas técnicos.	**Tiempo real:** 1 semanas.
Descripción: El operador será capaz de editar la forma de visualización de las rejillas del componente. Para esto debe acceder en el Inspector de Propiedades a la propiedad correspondiente a las rejillas, en esta se brindan las opciones de si se desea o no mostrar dimensiones y las propiedades del lápiz con que se dibujarán las líneas.	
Observaciones:	
Prototipo de interfaz:	

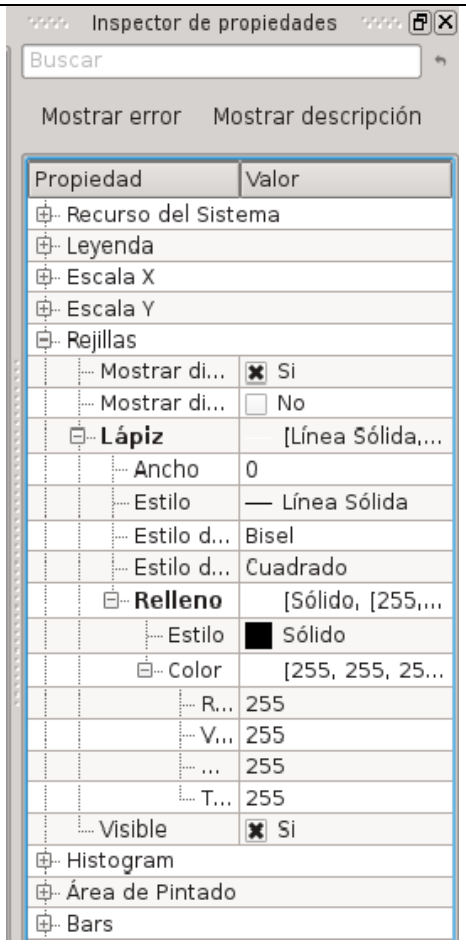


Tabla 9 HU Configurar eje X

Historia de usuario	
Número: HU8	**Nombre del requisito:** Configurar eje X
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 4 días.
Riesgo en desarrollo: o Problemas eléctricos. o Problemas técnicos.	**Tiempo real:** 1 semanas.
Descripción: El operador será capaz de editar la forma de visualización del eje X del componente. Para esto se accede dentro del Inspector de Propiedades a la propiedad correspondiente a editar dicho eje, esta da la posibilidad de editar el nombre, título del eje al igual que el color de la letra, la fuente y la forma que se pintará el eje.	

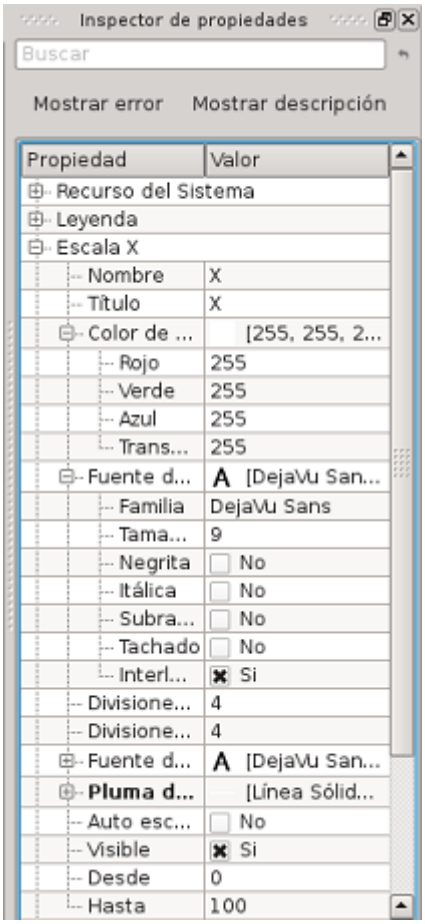
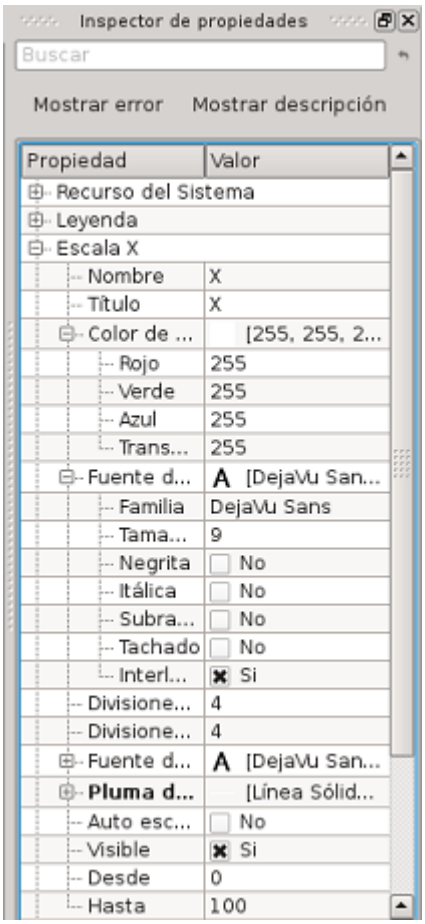
Observaciones:
Prototipo de interfaz: 

Tabla 10 HU Configurar eje Y

Historia de usuario	
Número: HU9	**Nombre del requisito:** Configurar eje Y
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 4 días.
Riesgo en desarrollo: ○ Problemas eléctricos. ○ Problemas técnicos.	**Tiempo real:** 1 semanas.
Descripción: El operador será capaz de editar la forma de visualización del eje Y del componente. Para esto se accede dentro del Inspector de Propiedades a la propiedad correspondiente a editar dicho eje, esta da la posibilidad de editar el nombre, título del eje al igual que el color de la letra, la fuente y la	

forma que se pintará el eje.
Observaciones:
Prototipo de interfaz: 

Historia de usuario	
Número: H10	**Nombre del requisito:** Configurar propiedades generales
Programador: Maiyara Guerra Cardet	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 8 días.
Riesgo en desarrollo: o Problemas eléctricos. o Problemas técnicos.	**Tiempo real:** 2 semanas.
Descripción: El operador podrá editar las propiedades comunes con las que cuentan todos los gráficos, dicho sea el caso de: alto, ancho, posición, visible, entre otras.	
Observaciones:	

2.5. Descripción de la solución

El componente propuesto será incorporado a los componentes de visualización gráfica existentes actualmente al SCADA SAINUX, permitiendo realizar un análisis estadístico a partir de la visualización de un gráfico de barras y ha de cumplir los requisitos descritos en el epígrafe anterior.

2.6. Diagrama de clases

Un diagrama de clases es utilizado para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones. Cuando se crea un diagrama de clases, se está modelando una parte de los elementos y relaciones que configuran la vista de diseño del sistema. A continuación se muestra el diagrama de clases del diseño teniendo en cuenta las entidades, sus atributos y relaciones.

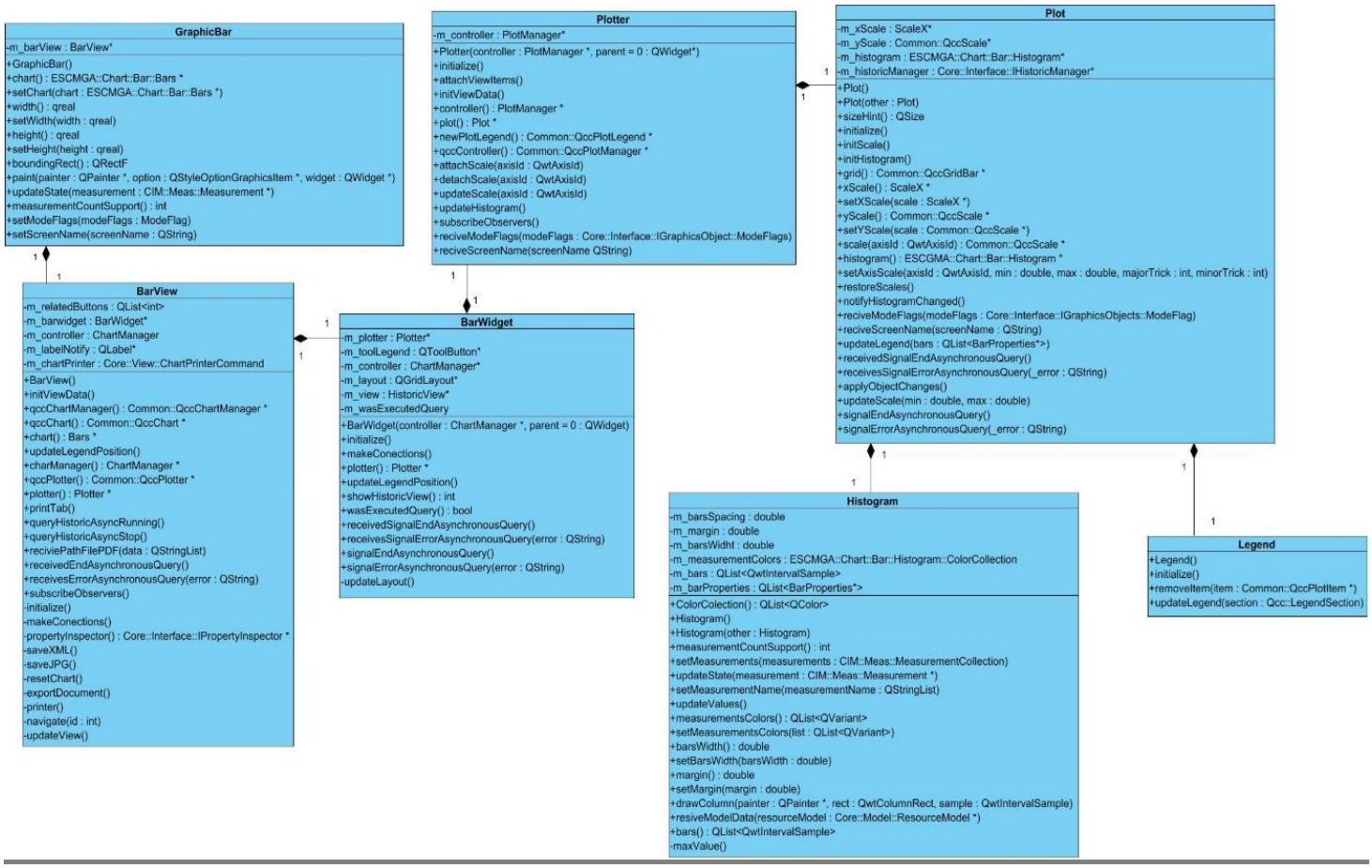


Ilustración 6 Diagrama de clases

2.6. Arquitectura del sistema

Una arquitectura de software consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. (25)

La arquitectura usada para el desarrollo de la solución propuesta, es el Modelo Vista Presentador, debido a que es la arquitectura base del marco de trabajo QT, utilizado en el desarrollo de la misma.

2.6.1. Modelo Vista Presentador (MVP)

La arquitectura Modelo-Vista-Presentador (MVP) surge para ayudar a realizar pruebas automáticas de la interfaz gráfica, para ello la idea es codificar la interfaz de usuario lo más simple posible, teniendo el menor código posible, de forma que no merezca la pena probarla. En su lugar, toda la lógica de la interfaz de usuario, se hace en una clase

separada (que se conoce como Presentador), que no dependa en absoluto de los componentes de la interfaz gráfica y que, por tanto, es más fácil de realizar pruebas. (26)

El Presentador hará de enlace entre el modelo y la vista, y dotará de inteligencia a la vista. Como el objetivo es poder probarlo fácilmente, el Presentador recibe las interfaces que deben implementar el modelo y la vista, con los métodos públicos a los que el Presentador debe llamar.

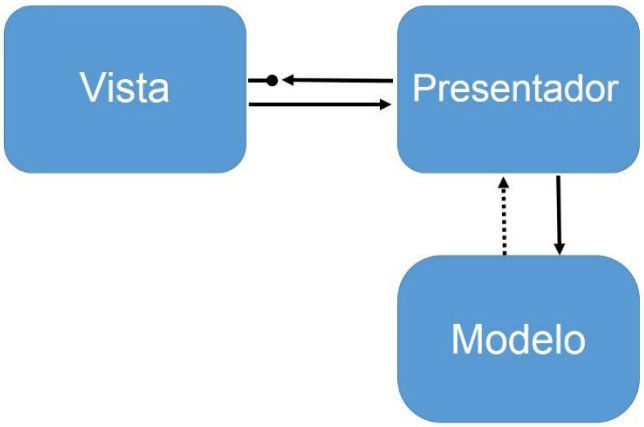


Ilustración 7 Modelo Vista Presentador

2.7. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.

En la construcción de la solución, para garantizar una buena práctica de programación, se utilizaron dos patrones de diseño, el patrón observador y el método plantilla. El primero fue utiliza ya que en el momento de actualizar las mediciones representadas en el componente, inmediatamente se actualizarían los observadores, siendo utilizado en la clase Plot, que sería observador de Legend e Histogram, y en la clase Plotter, observador de la clase Plot; y el método plantilla se utiliza ya que a la hora de pintar las barras fue necesario reimplementar el método de pintado ya existente en la clase Histogram de la biblioteca *Qwt*, siendo utilizado en la clase Histogram de la solución.

A continuación se presenta una breve descripción de cómo se definen estos patrones y su comportamiento.

Patrón Observador (Observer): Es un patrón de comportamiento que 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. (27)

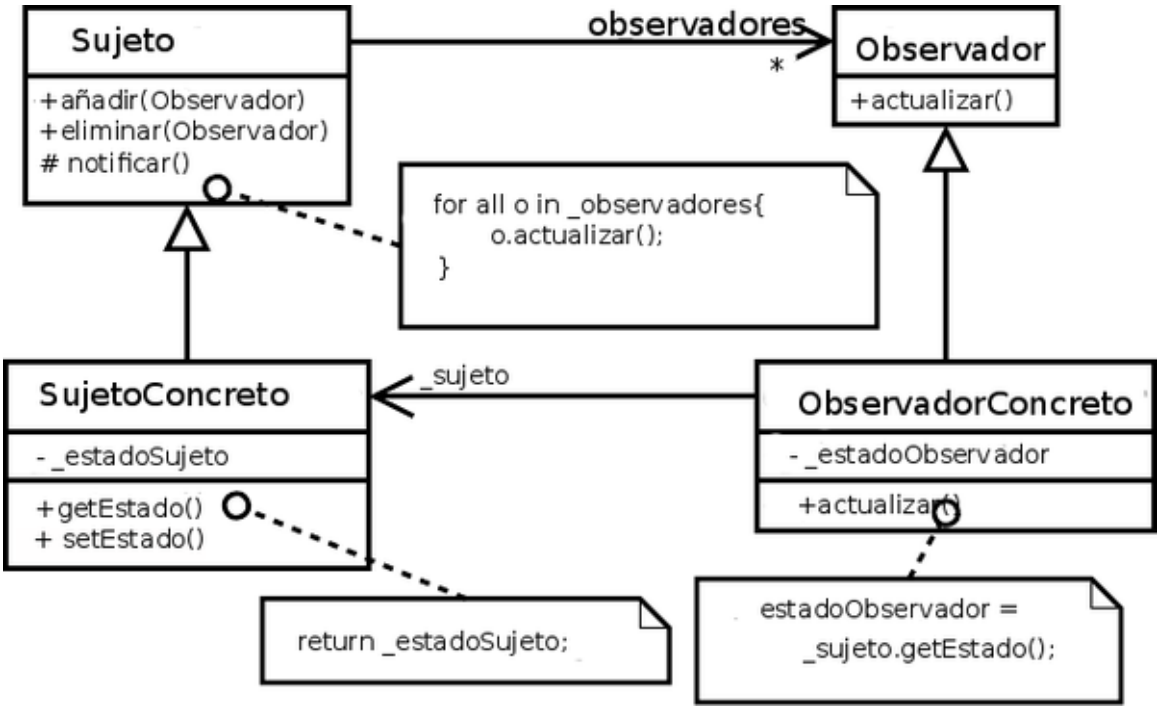


Ilustración 8 Patrón Observador

- **Sujeto (Subject):**

  El sujeto proporciona una interfaz para agregar (attach) y eliminar (detach) observadores. El Sujeto conoce a todos sus observadores.

- **Observador (Observer):**

  Define el método que usa el sujeto para notificar cambios en su estado (update/notify).

- **Sujeto Concreto (ConcreteSubject):**

  Mantiene el estado de interés para los observadores concretos y los notifica cuando cambia su estado. No tienen por qué ser elementos de la misma jerarquía.

- **Observador Concreto (ConcreteObserver):**

  Mantiene una referencia al sujeto concreto e implementa la interfaz de actualización, es decir, guardan la referencia del objeto que observan, así en caso de ser notificados de algún cambio, pueden preguntar sobre este cambio.

Patrón método planilla (method template): Define, en una operación, el esqueleto de un algoritmo delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura utilizado en el proceso de visualización de los objetos gráficos del SVG (siglas en inglés de *Gráficos Vectoriales Redimensionables*) los cuales tienen operaciones comunes en los momentos iníciales solo diferenciándose en los finales de este proceso. (28)

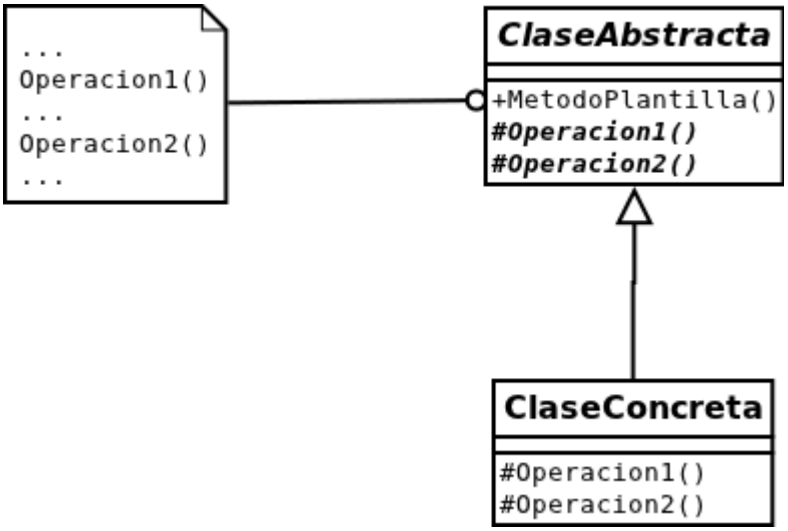


Ilustración 9 Patrón Método Planilla

- **Clase Abstracta**: proporciona la definición de una serie de operaciones primitivas (normalmente abstractas) que implementan los pasos de un algoritmo y que serán definidas en las subclases.

  Se encarga también de la implementación de un método desde el cual son invocadas, entre otras, las operaciones primitivas. Dicho método actúa a modo de plantilla, de ahí el nombre de este patrón, definiendo la secuencia de operaciones de un algoritmo.

- **Clase Concreta**: implementa las operaciones primitivas definidas en la clase abstracta de la cual hereda, quedando así determinado el comportamiento específico del algoritmo definido en el método plantilla, para cada subclase.

2.8. Conclusiones parciales

En el presente capítulo se realizó el proceso de análisis y diseño de la solución propuesta. Dentro de las actividades descritas se encuentran la captura de requisitos y sus respectivas historias de usuario, el diagrama de clases propuesto y la arquitectura de software utilizada, junto a los patrones de diseño para lograr una buena práctica de programación.

Capítulo 3: Implementación y pruebas

3.1. Introducción

En este capítulo se realiza la descripción de la solución propuesta, fundamentando la necesidad de desarrollar el componente gráfico de barras como parte del plan de evolución en desarrollo, para dotar al HMI del SCADA SAINUX con los diferentes tipos de gráficos que permitan evaluar el desempeño y los cambios sufridos por las variables durante su representación gráfica. Al igual que incorporará los diagramas correspondientes al modelo de implementación y las pruebas realizadas al sistema para comprobar su validez dependiendo de la aceptación del cliente.

3.2. 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.

3.2.1. 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. (29)

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

- Executable, componente que se puede ejecutar
- Library, biblioteca de objetos estática o dinámica
- Table, componente que representa una tabla de base de datos
- File, componente que representa un documento que contiene código fuente o datos.
- Document, componente que representa un documento.

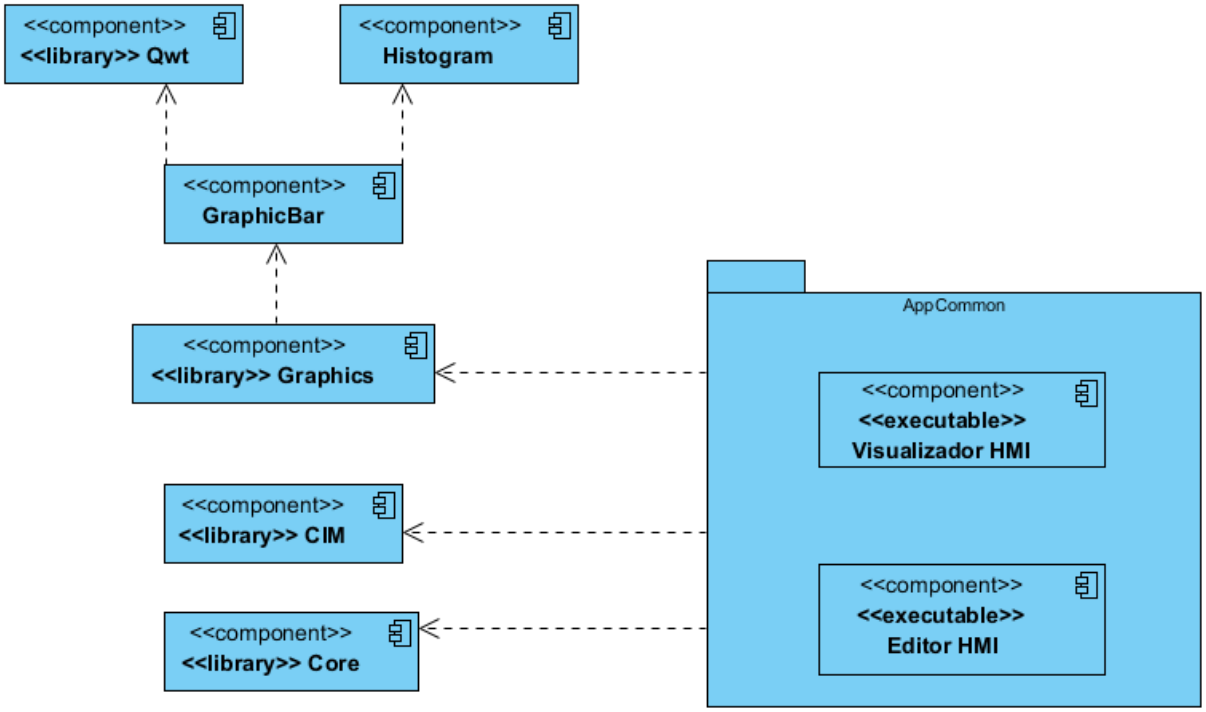


Ilustración 10 Diagrama de componentes

El sistema general cuenta con elementos comunes para los ambientes de ejecución, el Editor HMI y Visualizador HMI. Estos elementos comunes se dividen en varias bibliotecas especializadas como Core, CIM y Graphics, en el presente desarrollo se agrega a la biblioteca Graphics el componente GraphicBar que cuenta con un Histogram para la gestión de las distintas barras. De igual manera, Graphicbar incorpora la biblioteca Qwt para la gestión de pintado del componente gráfico a visualizar.

3.2.2. 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.

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

A continuación se tiene el diagrama de despliegue modelado para la aplicación a desarrollar:

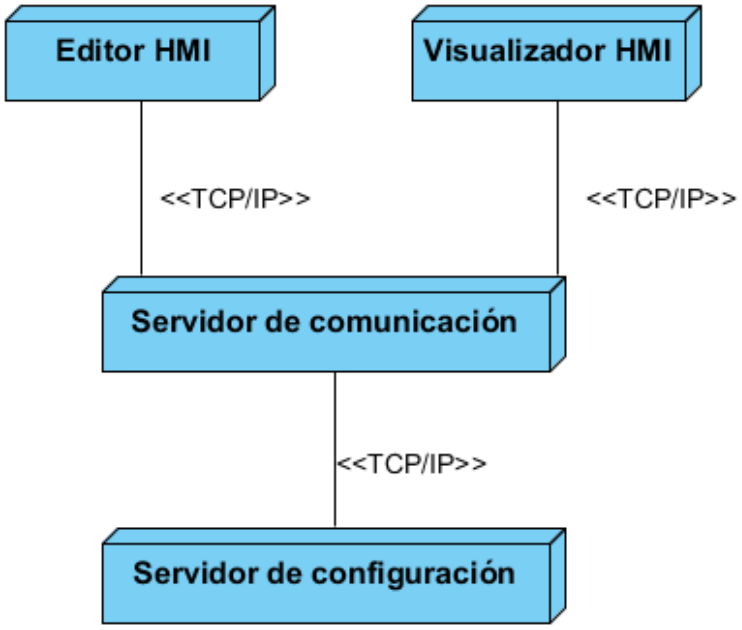


Ilustración 11 Diagrama de despliegue

3.3. Estándar de codificación

Los estándares de codificación, también conocidos como estilos de programación o convenciones de código, son convenios para escribir código fuente en ciertos lenguajes de programación. Permiten que el código en consecuencia sea mantenible y que todos los participantes lo puedan entender en un menor tiempo.

Para la implementación del componente gráfico de barras es necesario utilizar el estándar de codificación de C++ para el proyecto SCADA SAINUX.

Algunas de las pautas que define el estándar utilizado define:

- En los archivos cabecera debe incluir el copyright y la licencia, o una referencia de la misma, al estilo GNU GPL.
- Se adopta el estilo de bloques de documentación de JavaDoc, el cual consiste de un bloque de comentario de estilo C.
- 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.
- Para hacer referencia a otras clases utilizar el comando @see.
- El código será escrito en inglés y la documentación en español.
- Las variables y funciones comienzan con letra minúscula. Cada palabra consecutiva en el nombre comienza con letra mayúscula.

3.4. Creación e integración del módulo gráfico de barras dentro del código actual del HMI SAINUX como componente visual dentro de la paleta de componentes.

En el desarrollo actual del HMI SAINUX se deben seguir ciertas formalidades para lograr incorporar un nuevo componente a la barra de componentes y que este se logre visualizar correctamente.

A continuación se describirán los pasos a seguir durante la lograr incorporar el ícono del gráfico de barras dentro de la paleta de componentes.

Primero es necesario crear dentro del módulo **Extension** una clase que herede de la interfaz **Core::Interface::IPalette**, en nuestro caso se incluye dentro de la carpeta **ChartPlugin** debido a que el componente que se quiere crear pertenece a los de tipo **Gráficos** dentro de la paleta de componentes, para cumplir con las normas de organización del proyecto.

Para toda clase que herede de la interfaz **Core::Interface::IPalette** es necesario reimplementar los métodos que se describen a continuación.

```
1. int id() const;
```

Este método retornará el id del gráfico creado, que sería en su momento la visualización completa del *widget* que se crea al lanzar el componente sobre el despliegue. Quedaría de la siguiente manera:

```
int BarPalette::id() const
{
            return METATYPE_ID(ESCMGA::Chart::Bar::GraphicBar);
}
```

En este caso se retorna el id de la clase **ESCMGA::Chart::Bar::GraphicBar** que será la que se encargaría de la visualización del gráfico, esta se explicará en detalles más adelante.

```
2. QString name() const;
```

El método **name** retorna el nombre del componente dentro de la paleta de componentes, en este caso *"Gráfico de barra"*.

```
QString BarPalette::name() const
{
            return QObject::trUtf8("Gráfico de barra");
}
```

3. `QString group() const;`

El método **group** retorna el nombre del grupo dentro de la paleta de componentes que se encontraría en el componente desarrollado.

```
QString BarPalette::group() const
{
           return QObject::trUtf8("Gráficos");
}
```

4. `QString toolTip() const;`

El método **toolTip** retorna el texto a visualizar como ayuda cuando se detiene el cursor sobre el componente.

```
QString BarPalette::toolTip() const
{
            return QObject::trUtf8("Gráfico de Barra");
}
```

5. `QIcon icon() const;`

Como su nombre lo indica, este método retorna el ícono que llevaría el componente dentro de la paleta de componentes que lo identifique.

```
QIcon BarPalette::icon() const
{
    return RESOURCE_ICON("graph_bar.png");
}
```

3.5. Creación del componente gráfico que se visualizará al hacer *drop* al componente seleccionado: gráfico de barras, de la paleta de componentes, dentro del despliegue.

Para lograr que el componente seleccionado se cree correctamente este debe heredar de clases existentes dentro del HMI que hacen referencia a los componentes que se visualizan dentro de un despliegue, como la clase **Shape** y **Widget**. En este caso se hereda de la clase **Widget**.

Widget hereda de **Shape**, **Shape** de **GraphicObject**, **GraphicObject** hereda **IGraphicObject** y de **QGraphicItem**.

Es necesario en este punto reimplementar los métodos abstractos que imponen dicha jerarquía, como es el caso de **width(), setWidth(), height(), setHeight(), boundinghtRect(), paint() y updateState()**.

1. `qreal width() const; void setWidth(const qreal &_width); qreal height() const; void setHeight(const qreal &_height);`

Los métodos **width, setWidth, height, setHeight** se encargan de modificar el alto y ancho del componente, permitiendo que se le pueda hacer ***resize*** al mismo dentro del despliegue.

```
qreal GraphicBar::width() const
{
   return m_barView->width();
}
void GraphicBar::setWidth(const qreal &_width)
{
    m_barView->setGeometry(m_barView->x(), m_barView->y(), _width, height());
}
qreal GraphicBar::height() const
{
    return m_barView->height();
}
void GraphicBar::setHeight(const qreal &_height)
{
    m_barView->setGeometry(m_barView->x(),          m_barView->y(), width(), _height);
}
```

2. `QRectF boundingRect() const;`

El método **boundingRect** devuelve el área rectangular que ocupa el componente dentro del despliegue.

```
QRectF GraphicBar::boundingRect() const
```

```
{
  return m_proxy->geometry();
```

```
3. void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget);
```

El método **paint,** una vez implementado el **boundingRect,** este se encarga en su interior de la llamada al método **paintCorners,** que es el encargado de pintar las líneas discontinuas que marcan que el componente esta seleccionado y se pueda hacer el **resize** a partir de dicha rejilla.

```
void GraphicBar::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
{
    Q_UNUSED( widget );
    paintCorners( painter, option );
}
```

```
4. void updateState(CIM::Meas::Measurement *_measurement);
```

El método **updateState** se encarga de las actualizaciones gráficas sufridas en el componente si este depende de alguna medición, ya que una vez que dicha medición cambia en el proceso de ejecución se invoca a este método debido que el componente es un Observador de dicha medición.

3.6. Lógica de pintado y actualización del componente gráfico de barras

Para el pintado y actualización de un componente dentro del despliegue es necesario seguir cierta lógica de pintado dentro del modelo propuesto por HMI. En este caso la clase **GraphicBar** es invocada cuando se hace ***drop*** en el componente seleccionado en la paleta de componentes dentro del despliegue. Este sólo se encarga de controlar los ***resize*** del componente y recibir las actualizaciones del mismo en modo de ejecución a través del método **updateState**, por lo tanto, es necesario una nueva jerarquía de clase que se encargue de la lógica de pintado. Por lo que se propone la siguiente estructura:

El componente **GraphicBar** contiene una clase **BarView** que se encarga de la creación de las acciones sobre el área de pintado, dígase sea el caso de **ZoomIn**, **ZoomOut**, etc. También contiene una clase **BarWidget** que sería la encargada de la zona de pintado que

se modificará según las acciones del usuario sobre el componente como son las leyendas y el histograma. Para ello se crea la clase **Plotter** que es quien maneja dichas modificaciones.

3.6.1. Lógica de creación del Histograma

Para la representación de las barras dentro del gráfico se utilizó el componente **Histogram** de la librería **Qwt** para el pintado donde fue necesario implementar la lógica de cómo se pintarán las barras según las modificaciones del sistema con la variación de las mediciones asociadas a cada barra. Debido a que la configuración por defecto de visualización de la clase Histogram de la Qwt no responde a los requerimientos deseados para la representación de las barras, fue necesario crear una función que sería la encargada de la lógica para la representación visual de las barras dentro del componente.

La función realizada lleva por nombre updateValues(), la misma debe ser invocada cada vez que se detecte alguna modificación realizada por el usuario en el proceso de configuración. A continuación se muestra la implementación de la misma:

```
void Histogram::updateValues()
{
    if(!plot())
        return;
    if(m_measurements.count() == 0)
        return;
    if(m_modeFlags == IGraphicsObject::RUNTIME)
        dynamic_cast<Plotter*>(plot())->plot()->updateScale(0,
                                                maxValue()+10);
    m_bars.clear();
    m_barProperties.clear();
    double from, to;
    QVector< QwtIntervalSample > samples;
    for(int i=0; i<m_measurements.count(); i++)
    {
        if(i == 0)
        {
            from = 0 + m_margin;
```

```
            to = from + m_barsWidth;
        }
        else
        {
            from = (i*m_barsWidth) + (m_barsSpacing*i) + m_margin;
            to = from + m_barsWidth;
        }
        if(m_modeFlags == IGraphicsObject::RUNTIME)
        {
            samples << QwtIntervalSample(m_measurements.at(i)->
                                    value().toDouble(), from, to);
            m_bars.append( QwtIntervalSample(m_measurements.at(i)
                            -> value().toDouble(), from, to) );
        }
        else
        {
            samples << QwtIntervalSample(50, from, to);
            m_bars.append( QwtIntervalSample(50, from, to) );
        }
        BarProperties* barProperty = new BarProperties;
        barProperty->setName(m_measurements.at(i)->aliasName());
        barProperty->setColor(m_measurementsColors.at(i));
        m_barProperties.append(barProperty);
    }
    setSamples(samples);
    dynamic_cast<Plotter*>(plot())->plot()
                                        ->notifyHistogramChanged();
    dynamic_cast<Plotter*>(plot())->plot()
                                    ->updateLegend(m_barProperties);
}
```

El proceso de configuración por parte del operador se realizaría de la siguiente manera:

Paso 1: El usuario selecciona Mediciones dentro de la propiedad Histogram del Inspector de Propiedades.

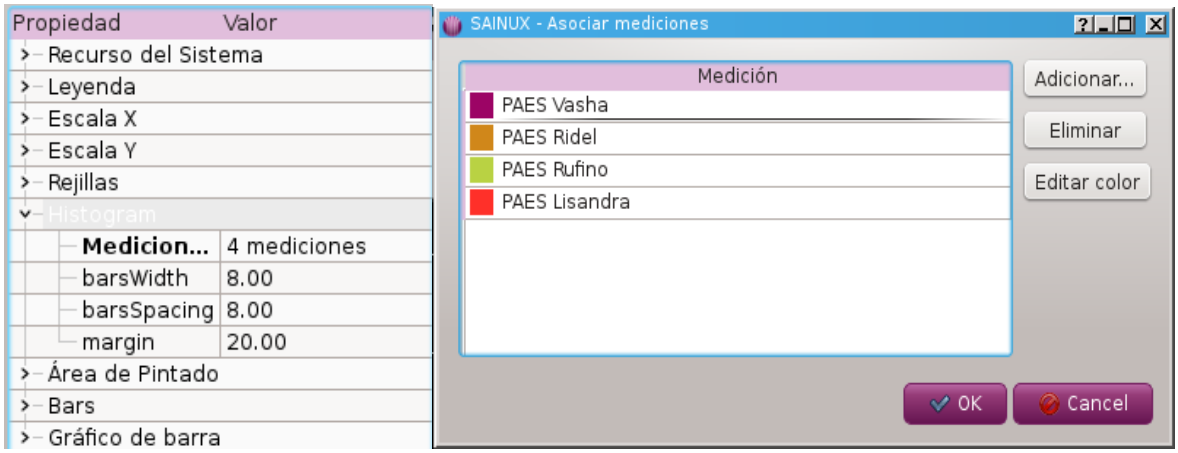


Ilustración 12 Paso 1

Paso 2: Se adicionan los puntos que se desea visualizar en el Gráfico de Barras.

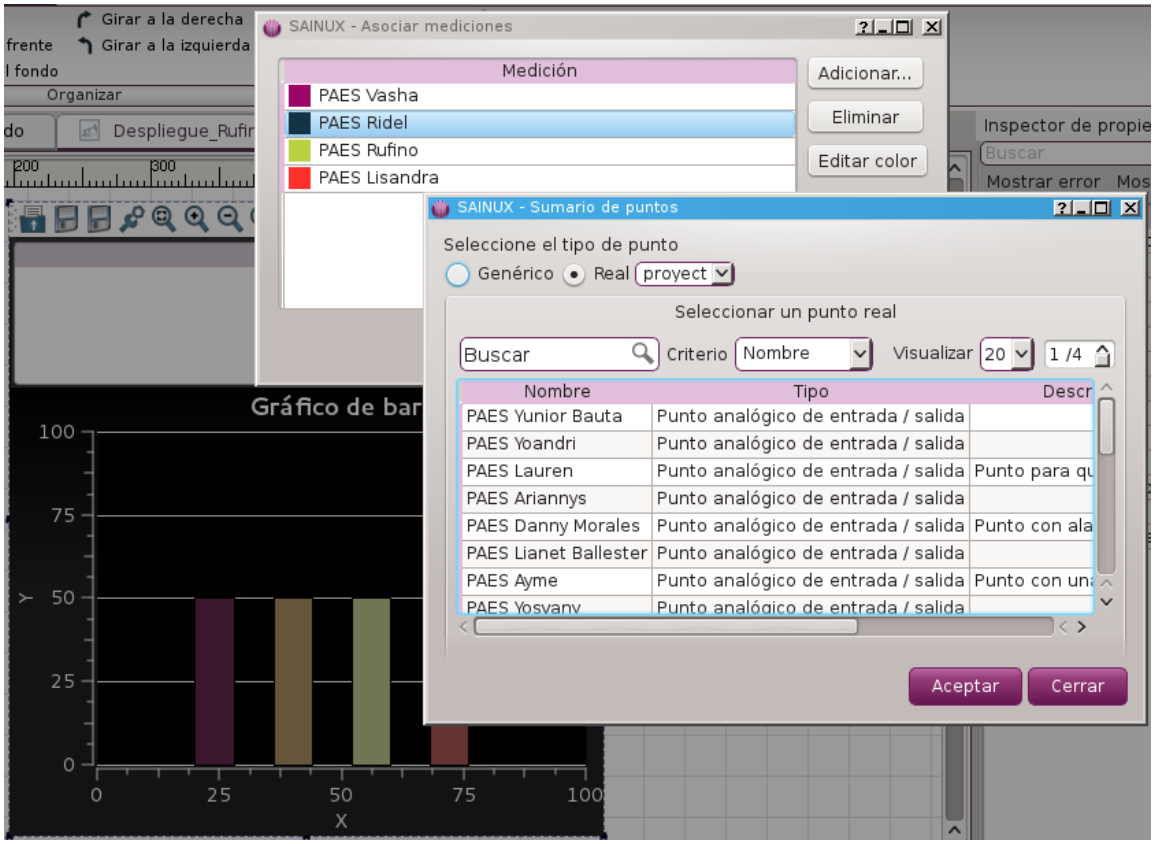


Ilustración 13 Paso 2

Paso 3 (Opcional): Se editar el color de la barra que representará la medición.

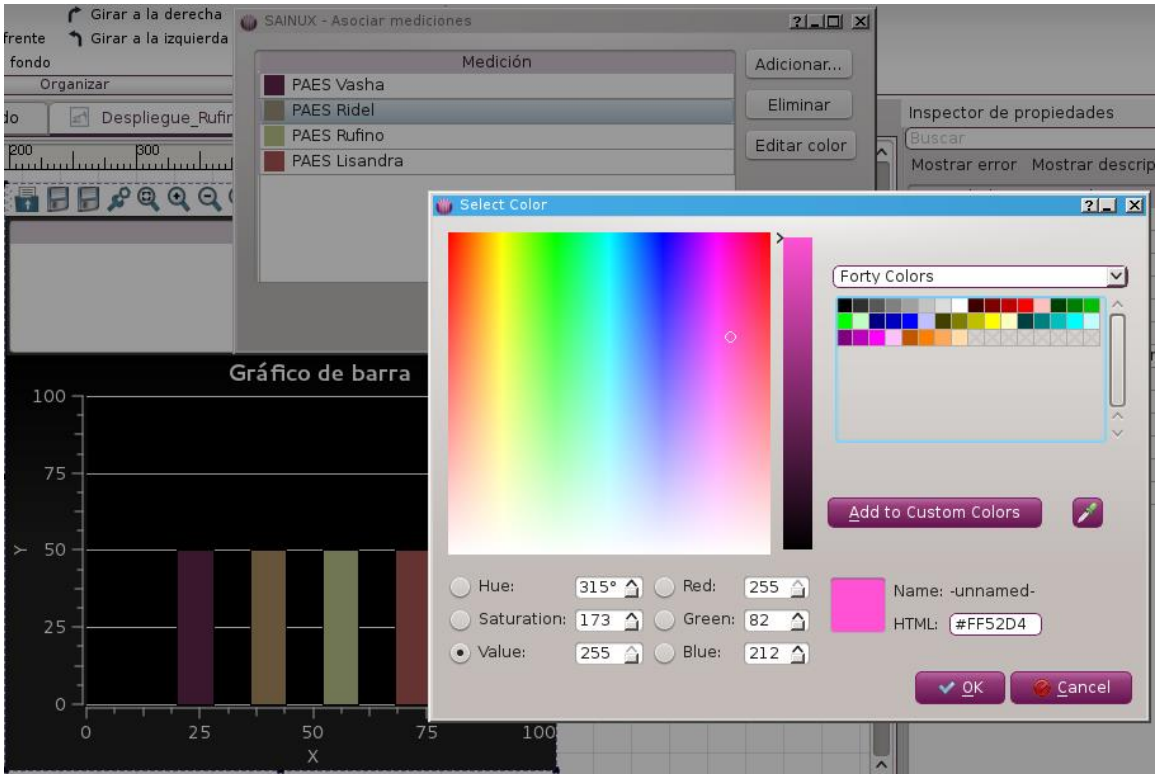


Ilustración 14 Paso 3

Paso 4: Se modifica las propiedades barsWidth, barsSpacing y margin. Estas son las encargadas de editar como se mostrara el gráfico, el ancho que tendrán las barras, el espacio entre ellas y el margen.

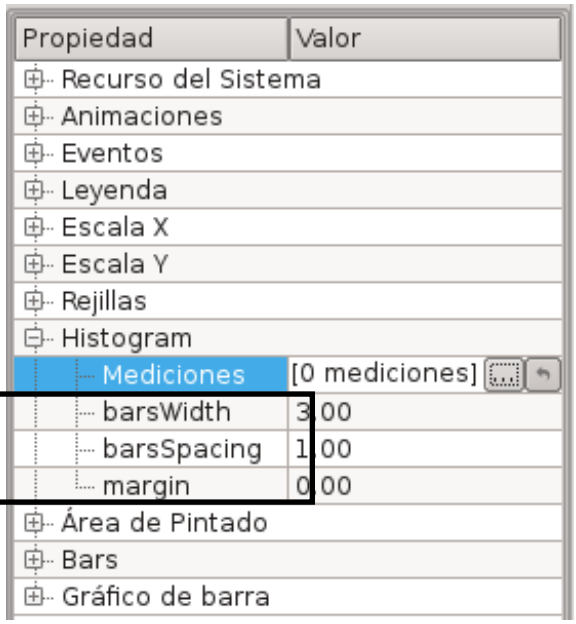


Ilustración 15 Paso 4

3.7. Pruebas

Los casos de prueba son pruebas unitarias o funcionales que se le realizan al sistema. Las pruebas unitarias son validaciones desde la perspectiva del desarrollador, mientras que una prueba funcional se realiza por un usuario externo a la aplicación, generalmente, usuario final o cliente. El objetivo general de una prueba es certificar que la historia de usuario está lista. Mientras un código no se ha probado no existe, por lo que los casos de prueba deben acompañar al sistema durante su ciclo de explotación. (31)

Generalmente se define una prueba por cada historia de usuario, aunque no existe un límite definido de la cantidad de pruebas que un sistema puede tener.

Para realizar las pruebas al componente gráfico de barras se decide el uso de las pruebas de aceptación.

Las pruebas de Aceptación permiten determinar si las ventajas que ofrece el software realmente justifican su uso. En esta prueba se evalúa el grado de calidad que tiene el software con relación a todos los aspectos relevantes para que el uso del producto se justifique. (32)

A partir de las historias de usuario definidas se realizaron casos de prueba, la descripción de estos.

Tabla 11 Caso de prueba de Aceptación 1

Casos de prueba de Aceptación	
Número: 1	**Historia de Usuario:** 1
Nombre: Visualizar Componente	
Condiciones de ejecución: Debe estar creado un despliegue	
Entradas/Pasos de ejecución: **1. El usuario selecciona el ícono gráfico de barras y le hace *drop* dentro del despliegue.** **2. El sistema crea el gráfico dentro del despliegue.**	
Resultado esperado: Visualización del gráfico dentro del despliegue.	
Evaluación de la prueba: Se realiza una primera iteración donde se encontraron varias no conformidades, se corrigen estas y se realiza otra en la que se obtuvo una evaluación de satisfactoria.	

Tabla 12 Caso de prueba de Aceptación 2

Casos de prueba de Aceptación

Número: 2	**Historia de Usuario**: 2
Nombre: Adicionar barra	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue	
Entradas/Pasos de ejecución: • El usuario accede a la propiedad Histograma dentro del Inspector de Propiedades, y selecciona mediciones. • El usuario selecciona la opción adicionar medición y escoge la medición que desea visualizar o si desea crear un punto genérico. • El sistema visualiza la medición en forma de una barra con un color otorgado automáticamente.	
Resultado esperado: Visualizar la medición en forma de barra	
Evaluación de la prueba: Se realiza una primera iteración donde se encontraron varias no conformidades, se corrigen estas, al realizar una segunda iteración se detectan otras no conformidades que son corregidas y en una tercera iteración se obtiene un resultado de satisfactorio.	

Tabla 13 Caso de prueba de Aceptación 3

Casos de prueba de Aceptación	
Número: 3	**Historia de Usuario**: 3
Nombre: Eliminar barra	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue y tener visibles al menos una medición.	
Entradas/Pasos de ejecución: • El usuario accede a la propiedad Histograma dentro del Inspector de Propiedades, y selecciona mediciones. • El usuario selecciona la medición que desea eliminar y da clic izquierdo en el botón Eliminar.	
Resultado esperado: Elimina la barra correspondiente a la medición eliminada	
Evaluación de la prueba: Satisfactorio	

Tabla 14 Caso de prueba de Aceptación 4

Casos de prueba de Aceptación	
Número: 4	**Historia de Usuario**: 4

Nombre: Modificar color de la barra
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue y tener visibles al menos una medición.
Entradas/Pasos de ejecución: • El usuario accede a la Histogram dentro del Inspector de Propiedades, y selecciona mediciones. • El usuario selecciona la medición a la cual desee modificarle el color y selecciona la opción Editar Color. • El usuario selecciona el color que desee darle a la barra
Resultado esperado: Se modifica el color de la barra
Evaluación de la prueba: Se realiza una primera iteración donde se encontraron varias no conformidades, se corrigen estas, al realizar una segunda iteración se detectan otras no conformidades que son corregidas y en una tercera iteración se obtiene un resultado de satisfactorio.

Tabla 15 Caso de prueba de Aceptación 5

Casos de prueba de Aceptación	
Número: 5	**Historia de Usuario**: 5
Nombre: Mostrar propiedades	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue	
Entradas/Pasos de ejecución: • El usuario da clic izquierdo en el componente. • El sistema muestra el Inspector de propiedades	
Resultado esperado: Se muestra las propiedades del gráfico.	
Evaluación de la prueba: Se realiza una primera iteración donde se encontraron varias no conformidades, se corrigen estas y se realiza otra en la que se obtuvo una evaluación de satisfactoria.	

Tabla 16 Caso de prueba de Aceptación 6

Casos de prueba de Aceptación	
Número: 6	**Historia de Usuario**: 6
Nombre: Configurar el área de pintado	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue	

Entradas/Pasos de ejecución: • El usuario accede a la propiedad Área de Pintado dentro del Inspector de Propiedades. • El usuario edita la opción que desee cambiar.
Resultado esperado: Se modifica el área de pintado del componente
Evaluación de la prueba: Satisfactorio

Tabla 17 Caso de prueba de Aceptación 7

Casos de prueba de Aceptación	
Número: 7	**Historia de Usuario**: 7
Nombre: Configurar rejillas	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue	
Entradas/Pasos de ejecución: • El usuario accede a la propiedad Rejillas dentro del Inspector de Propiedades. • El usuario edita la opción que desee cambiar.	
Resultado esperado: Se modifica las rejillas del componente	
Evaluación de la prueba: Satisfactorio	

Tabla 18 Casos de prueba de Aceptación 8

Casos de prueba de Aceptación	
Número: 8	**Historia de Usuario**: 8
Nombre: Configurar eje X	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue	
Entradas/Pasos de ejecución: • El usuario accede a la propiedad Eje X dentro del Inspector de Propiedades. • El usuario edita la opción que desee cambiar.	
Resultado esperado: Se modifica el eje x del componente	
Evaluación de la prueba: Satisfactorio	

Tabla 19 Caso de prueba de Aceptación 9

Casos de prueba de Aceptación	
Número: 9	**Historia de Usuario**: 9

Nombre: Configurar eje Y
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue
Entradas/Pasos de ejecución: • El usuario accede a la propiedad Eje Y dentro del Inspector de propiedades. • El usuario edita la opción que desee cambiar.
Resultado esperado: Se modifica el eje y del componente
Evaluación de la prueba: Se realiza una primera iteración donde se encontraron varias no conformidades, se corrigen estas y se realiza otra en la que se obtuvo una evaluación de satisfactoria.

Tabla 20 Casos de prueba de Aceptación 10

Casos de prueba de Aceptación	
Número: 10	**Historia de Usuario**: 10
Nombre: Configurar propiedades generales	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue	
Entradas/Pasos de ejecución: • El usuario accede al Inspector de propiedades. • El usuario edita la propiedad que desee cambiar.	
Resultado esperado: Se modifica el eje y del componente	
Evaluación de la prueba: Se realiza una primera iteración donde se encontraron varias no conformidades, se corrigen estas y se realiza otra en la que se obtuvo una evaluación de satisfactoria.	

Luego de realizar 3 iteraciones de pruebas se obtuvieron los siguientes resultados:

Sistema	HU	Iteración	NC	Cerrada	No Procede
Componente gráfico de barras	10	1ra	28	25	3
		2da	7	7	0
		3ra	0	0	0

Al obtener resultados satisfactorios en las pruebas realizadas se prevee un alto grado de calidad del software.

3.8. Conclusiones parciales

En este capítulo se expusieron los principales artefactos del modelo de implementación, mostrando así los componentes del sistema y su relación a través del diagrama de componentes y los vínculos de comunicación que existe entre los nodos del diagrama de despliegue, el cual modela la arquitectura en tiempo de ejecución. También se realizó una breve descripción de la implementación de la solución propuesta y de los casos de prueba que verifican el correcto funcionamiento del sistema a través de las pruebas de aceptación.

Conclusiones generales

A partir del desarrollo del componente gráfico de barras se concluye que:

- Se permite la representación gráfica de una o varias variables en forma de barras, asegurando de esta forma lograr una representación visual más aproximada a la correlación estadística de los datos para el operador.
- Se logra observar una preferencia, o tendencia, de las mediciones por ubicarse hacia una determinada región de valores dentro del espectro de valores posibles.
- Con la utilización de la librería Qwt es posible lograr representaciones visuales con una mayor profesionalidad de representación gráfica, teniendo en cuenta que si se desea lograr una representación personalizada de visualización es necesario recurrir a la reimplementación de la función de pintado.
- Las diferentes formas de configuración del componente propuesto hacen que el ambiente de trabajo sea lo más amigable posible para los operadores del sistema, logrando de esta forma la aceptación de los mismos.

Recomendaciones

Se recomienda que:

- Se continúe el desarrollo de las distintas formas de representación de los gráficos de barras:
  - Visualización horizontal
  - Barras en 3D
  - Barras Flotantes
  - Barras Apiladas

Referencias

1. **Automatizacion Industrial. [En línea] https://automatizacionindustrial.wordpress.com/2011/02/09/queeslaautomatizacionindustrial/.**

2. Lozano, Carlos de Castro. *Introducción SCADA.*

3. Díaz, Ing. Henry Mendiburu. *Sistemas SCADA.*

4. De Castro Lozano, Carlos y Romero Morales, Cristóbal. *Introducción a SCADA.*

5. http://www.automatas.org/redes/scadas.htm. [En línea] 2 de Marzo de 2006.

6. Ingeniería en automatización y control industrial. [En línea] http://iaci.unq.edu.ar/materias/laboratorio2/HMI\Introduccion HMI.pdf.

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

8. Bornot Rivero, María Mercedes y Rivera Acosta, Alexei. *Desarrollo de componentes gráficos para el módulo de visualización del SCADA Nacional.* Ciudad de la Habana : s.n., 2010.

9. Sehara Driggs, Yosell Luis y Companioni Sardiña, Yusmary. *Herramienta de configuración para sistemas SCADA.* Ciudad de la Habana : s.n., 2010.

10. Forcade Gómez, Natasha y Torres Lorenzo, William. *Manual de WinCC V 5.1.* Ciudad de la Habana : s.n., 2007.

11. Acevedo Sánchez, Jose. *Control Avanzado de Procesos.* 2002.

12. Microsoft Technet. [En línea] 2014. technet.microsoft.com/es-es/library/hh230886.aspx..

13. Geisecke, Frederick. *Dibujo y comunicación gráfica.* Cuidad México : PERSON EDUCATION, 2006. ISBN: 9702608112..

14. Sánchez, Juan J. *Manual de Analisis Estadistico de Los Datos.* s.l. : Alianza Editorial S.A, 2008. ISBN: 9788420687162.

15. Rodríguez Oromendía, Ainhoa. *Marketing, Estrategias y Tendencias.* s.l. : s.l.: Sanz y Torres, 2012. ISBN: 9788415550037.

16. Microsoft. [En línea] https://msdn.microsoft.com/es-es/library/ms251748%28v=vs.90%29.aspx.

17. Palmer, Alfonso Luis. *Análisis de datos, etapa exploratoria.* s.l. : Piramede. ISBN: 9788436813395.

18. Rumsey, Deborah. *Statistics Workbook for Dummies.*

19. Staszkow, Ronald y Robert, Bradshaw. *The Mathematical Palette.*

20. Rodriguez Sanchez, Tamara. *Metodología de desarrollo para la actividad productiva de la UCI.*

21. Perdita Stevens, Rob Pooley. *Utilización de UML en Ingeniería del Software con Objetos y Componentes.* s.l. : Addison Wesley, 2002.

22. Visual Paradigm. [En línea] http://www.visual-paradigm.com/aboutus/.

23. Hamilton, Naomi. *The A-Z of Programming Languages: C++.* s.l. : Computerworld, 2008.

24. blog.qt.io. *Qt Creator 3.4.0 released.* 2015.

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

26. Imaginanet. [En línea] 2 de Febrero de 2012. http://www.imaginanet.com/blog/patron-mvp.html.

27. Timothy G. Mattson, Beverly A. Sanders, Berna L. Massingill. *Patterns for Parallel Programming.* s.l. : Addison- Wesley, 2008.

28. E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns: elements of reusable object-oriented software.* s.l. : Addison-Wesley, 1994.

29. [En línea] (http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html).

30. Penedés, Patricio Letelier y Mª Carmen. *Métodologías ágiles para el desarrollo de software.*

31. Salas, Ing. Daniel Cafferata. CalidadySoftware.com. [En línea] http://www.calidadysoftware.com/testing/casos_de_prueba.php.

32. Pruebas de Software. [En línea] http://pruebasdesoftware.cpm.

33. Buenas Tareas. [En línea] 11 de Mayo de 2014. http://www.buenastareas.com/ensayos/Gr%C3%A1fico-De-Barras/51954917.html.