UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

cedin Informática Industrial

"DESARROLLO DEL COMPONENTE GRÁFICO DE TIPO SECTOR PARA EL HMI DEL SCADA SAINUX"

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

Autor: Rolando Rafael Futiel Gutiérrez
Tutor: Ing. Yailín González Cruz
Co-tutor: Ing. José Ernesto Carreño Bueno

La Habana, Junio de 2016
"Año 58 de la Revolución"

"El respeto por el medio ambiente exige el reconocimiento de una ley moral de la naturaleza humana"

(Discurso en la Asamblea General de la ONU, New York, Septiembre 25, 2015)

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas (UCI) 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 2016.

_____

Rolando Rafael Futiel Gutiérrez

Autor

_____

Ing. Yailín González Cruz

Tutor

_____

Ing. José Ernesto Carreño Bueno

Co-tutor

DATOS DE CONTACTO

Generales del tutor:

Tutor: Yailín González Cruz.

Categoría docente: Especialista superior

Categoría científica: Ingeniero.

Síntesis: Ha tenido responsabilidades como desarrollador del sistema AnPer, jefa de la línea de seguridad y de HMI pertenecientes al Centro de Informática Industrial.

Correo: yailingc@uci.cu

Generales del co-tutor:

Co-tutor: José Ernesto Carreño Bueno.

Categoría docente: Especialista

Categoría científica: Ingeniero.

Síntesis: Ha tenido responsabilidades como desarrollador de las líneas Adquisición y HMI, desarrollador del sistema AnPer y desarrollador de SIPP fase II pertenecientes al Centro de Informática Industrial.

Correo: jecarrenob@uci.cu

AGRADECIMIENTOS

A mí mamá y mi papá por haberme dado la vida, por siempre empujarme a ser un profesional, a tener metas y cumplirlas; a mi mamá por estar siempre ahí para mí y apoyarme con amor incondicional en cada decisión que tomé, eres la primera en mi corazón.

A Noel, por actuar incondicionalmente como mi padre, por guiarme, ayudarme, por estar al lado de mi madre en los momentos difíciles y brindarnos su apoyo. ¡Cómo te lo agradezco!

A Norma y Ruben por ser mi segunda familia, por velar por mí y hacer cosas que ni mi familia ha podido hacer en tiempos difíciles, por malcriarme, ayudarme, apoyarme, cuidarme, entre otras muchas cosas que no alcanzaría tiempo para mencionar; les estoy eternamente agradecido, ustedes han sido muy importantes en los últimos 5 años.

A mis hermanos; Raiza por ser la cascarrabias, pero que en el fondo me quiere mucho, le agradezco también por darme dos sobrinos lindos, por los cuales me siento más lleno todos los días.

A mis abuelos, los que están todavía cuidándome y apoyándome y a los que ya no están, pero están mirándome con orgullo desde arriba.

A mis amigos incansables, los más divertidos, los más psiquiátricos, lo más dedicados, los que más se dejan querer: Arletis por actuar como la madre de todos, por regañarnos siempre; Roisbel por sacarnos de paso todos los días con sus locuras y boberías; Rosmery por darnos tantos momentos divertidos y sensibles; Rodnier por dar siempre todo y quedarse sin nada; Claudia, Diomne y Yaimara por seguirnos en las locuras y ayudarnos a formar una amistad linda.

A los tutores por ayudarme en este proceso difícil y complicado.

A mis profesores, que me formaron y exigieron mucho durante todo el tiempo de estudiante, y a todos los demás que de una forma u otra contribuyeron con este trabajo.

DEDICATORIA

A Nancy Futiel, porque sé que en cualquier lugar que esté, se está sintiendo orgullosa de mí.

RESUMEN

Los Sistemas de Supervisión, Control y Adquisición de Datos (SCADA[1]) se utilizan principalmente en el entorno industrial. En el Centro de Informática Industrial (CEDIN), se desarrolla el SCADA SAINUX, el cual cuenta con un módulo Interfaz Hombre Máquina (HMI[2]), que permite configurar, supervisar y controlar los procesos industriales. El módulo cuenta con una biblioteca de componentes gráficos la cual permite recrear los procesos que ocurren en una determinada planta lo más real posible. La presente investigación tiene como objetivo el desarrollo de un componente gráfico de tipo sector, para incorporarlo a los gráficos ya existentes en el HMI del SCADA SAINUX: barras, tendencia y el gráfico XY, y así enriquecer la calidad visual de la representación de la información, así como su interpretación y la de la relación entre un valor y el total, o entre dos valores y la suma de ambos de una variable cualitativa o discreta de los datos. El desarrollo de este componente gráfico se basa en el uso de la metodología AUP[3]-UCI, utilizando las técnicas de modelación establecidas en el lenguaje unificado de modelado (UML[4]) y el marco de trabajo Qt.

Palabras clave: SCADA; SAINUX; gráfico de tipo sector; HMI; representación de la información

[1] Según sus siglas en inglés (Supervisory Control and Data Acquisition)
[2] Según sus siglas en inglés (Human-Machine Interface)
[3] Según sus siglas en inglés (Agile Unified Process)
[4] Según sus siglas en inglés (Unified Modeling Language)

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

Introducción

El surgimiento de las computadoras fue uno de los pasos más importantes que dio la humanidad para dejar atrás los métodos convencionales con los que lentamente se desarrollaba. Desde ese momento, se ha apreciado el desmedido avance de la tecnología y la ciencia, el cual se ha acelerado considerablemente en los últimos cincuenta años. Con tal progreso, se ha hecho necesaria la creación e investigación de novedosas técnicas y procedimientos, que caminen de la mano de dicho desarrollo.

La automatización de los procesos en las industrias, es la cúspide y la meta de toda la investigación científico-técnica; para alcanzarla, se han desarrollado sistemas y maquinarias que reemplazan la mano de obra humana, perfeccionando y supervisando de una forma más eficaz el resultado final. Cuando se comenzó a implementar este proceso, las herramientas que iban surgiendo carecían de complejidad y muchas no satisfacían las necesidades, por lo que todo el arsenal industrial no se podía acoger a la nueva forma de trabajo. Con el paso de los años y las investigaciones, estas herramientas tomaron una complejidad superior y fueron agilizando el trabajo de tal forma que la industria se hacía aún más dependiente de ellas. Hoy día, existen procesos industriales que se monitorean y se supervisan a través de los Sistemas de Control, Supervisión y Adquisición de Datos (SCADA).

Los primeros sistemas SCADA que incorporaron Interfaz Hombre Máquina (HMI), salieron al mercado a finales de la década de 1970, como una respuesta a la demanda de supervisión de los procesos (1). Los sistemas SCADA actuales, posibilitan al operador acceder a las variables del proceso con mucha rapidez; sin importar a la distancia a la que se encuentran. Un SCADA es una herramienta muy valiosa en el control de procesos, pues permite una panorámica general del mismo, facilitando así la toma de decisiones. (2)

La Universidad de las Ciencias Informáticas (UCI), cuenta con el Centro de Desarrollo de Informática Industrial (CEDIN) cuyo objetivo principal es desarrollar sistemas informáticos que contribuyan a la automatización de los procesos industriales, tanto de las empresas cubanas como extranjeras y brindarles productos eficientes y de alta calidad y durabilidad para su exportación. Uno de estos productos es el SCADA SAINUX.

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

Uno de los elementos con marcada importancia en el EC, son los componentes gráficos, pues permiten la visualización del estado del proceso en el despliegue y de esa manera se posibilita la interacción del operador con el sistema. Entre estos componentes se pueden encontrar: objetos animados, gráficos, ventanas múltiples.

SAINUX cuenta actualmente con tres tipos de gráficos entre sus componentes: gráfico XY, gráfico de tendencia y gráfico de barras. Con el uso de estos se puede obtener una mejor monitorización de las actualizaciones y cambio de las variables del sistema según la recolección de los datos en el campo; debido al alto grado de muestreo y cantidad de información que estos brindan a los operadores.

Como componentes de visualización y muestreo, estos resultan muy útiles, pero aún no son capaces de solventar dificultades como la representación de la relación entre un valor y el total, o entre dos valores y la suma de ambos, representar variables cualitativas o categóricas, de preferencia nominales que permitan mostrar la proporción que le corresponde a cada categoría, por lo que se plantea como **situación problémica**: no es posible resaltar una o varias mediciones para lograr una interpretación más aproximada a la relación entre un valor y el total, o entre dos valores y la suma de ambos de una variable cualitativa o discreta de los datos.

A raíz de la situación anterior surge el siguiente **problema de la investigación**: ¿Cómo contribuir a la interpretación de la relación entre un valor y el total, o entre dos valores y la suma de ambos de una variable cualitativa o discreta para los datos en HMI SAINUX?

Enmarcándose como **objeto de estudio** el siguiente: los tipos de representación de la información.

Por lo que la investigación abarca el siguiente **campo de acción**: la representación gráfica de la información en el HMI del SCADA SAINUX.

Teniendo en cuenta lo anterior, se define como **objetivo general**: desarrollar un componente gráfico de tipo sectores para la interfaz HMI del SCADA SAINUX.

Para dar cumplimiento al objetivo propuesto se trazan las siguientes **tareas de la investigación:**

1. Investigación de sistemas SCADA y gráficas de sectores.
2. Análisis de las tecnologías necesarias para el desarrollo de la investigación.
3. Definición de la arquitectura candidata para el componente gráfico de sector de la interfaz hombre máquina en el SCADA SAINUX.
4. Análisis y Diseño del componente gráfico de sectores según las características y requisitos identificados.

5. Implementación del componente gráfico de sectores no funcional.
6. Implementación del componente gráfico de sectores funcional.
7. Integración del componente con el HMI del SCADA SAINUX.
8. Realización de pruebas para comprobar el cumplimiento de los objetivos propuestos en la presente investigación.

Para el desarrollo de las tareas mencionadas anteriormente es necesario apoyarse 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 sectores para el 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 sectores del HMI del SCADA SAINUX.

La investigación estará estructurada de la siguiente manera:

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 por qué de la necesidad de tener el componente gráfico de sectores en el HMI del 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 Proceso Unificado Ágil (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

En este capítulo se abordan los temas vinculados a los principales conceptos durante la investigación, además de las principales herramientas y tecnologías empleadas para el diseño y la implementación del componente gráfico de sectores para el HMI del SCADA SAINUX.

1.2. Principales conceptos

1.2.1. Automatización

La automatización es la facultad que poseen algunos procesos físicos para desarrollar las actividades de operaciones y funcionamiento en forma autónoma, es decir por cuenta propia. (3)

1.2.2. Automatización industrial

En síntesis, se puede entender como la facultad de autonomía o acción de operar por sí solo que poseen los procesos industriales y donde las actividades de producción son realizadas a través de acciones autónomas, y la participación de fuerza física humana es mínima y la inteligencia artificial, máxima. (3)

1.2.3. Sistemas de control

La automatización industrial se hace posible a través de los Sistemas de Control, que son organizaciones de equipos e instrumentos (lo físico), que combinados con procedimientos mentales o algorítmicos (lo inteligente), trabajan en torno a propósitos previamente establecidos (lo deseado).

Entre sus funciones principales se encuentran: la observación del proceso y sus variables a automatizar; el acondicionamiento de las variables y parámetros observados; el procesamiento de esta información y su comparación con lo deseado; y posteriormente, la acción de corrección de los elementos terminales para conseguir lo deseado. (3)

1.2.4. Variable cualitativa

Característica que recoge una cualidad de los individuos de la muestra. Una variable cualitativa no puede medirse con un instrumento ni lleva asociada una unidad de medida. (5)

1.2.5. Variable discreta

Una variable discreta es una variable cuantitativa que toma valores aislados, es decir no admite valores intermedios entre dos valores específicos. (6)

1.3. Sistemas SCADA

Un sistema SCADA es una aplicación de software, diseñada con la finalidad de supervisar y controlar procesos a distancia.

Se basa en la adquisición de datos de los procesos remotos y está diseñado fundamentalmente 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 una computadora. Además envía la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel, como a otros supervisores dentro de la empresa. (7)

1.3.1. Funciones de un sistema SCADA

Las funciones básicas son las que se describen a continuación: (7)

- Supervisión remota de instalaciones.
- Procesamiento de información.
- Presentación de gráficos dinámicos.
- Presentación de alarmas.
- Almacenamiento de información histórica.
- Programación de eventos.

1.3.2. Características principales del sistema SCADA SAINUX

Las características son las siguientes: (8)

- Sistema en tiempo real.
- Arquitectura distribuida.
- Arquitectura flexible, extensible, capaz de crecer o adaptarse según las necesidades cambiantes de la industria.
- Comunicación con total facilidad y de forma transparente al usuario con el equipamiento de planta y con el resto de la industria (redes locales y de gestión).
- Adquisición de datos para recolectar, procesar y almacenar, en tiempo real, información relevante sobre la evolución del proceso productivo.
- Supervisión, para observar desde el monitor la evolución del proceso.
- Control, que permite modificar la evolución del proceso (consignas, alarmas, recetas, menús, etc.) a través de comandos.
- Visualización de los datos de campo y de sistema a través de Interfaces Hombre-Máquina, que representen los mímicos del proceso y permitan a los operadores actuar sobre la planta.

- Interoperabilidad.
- Implementado en C++.
- Uso de componentes bajo la licencia GPL.
- Componentes desarrollados con alto nivel de reusabilidad.

1.3.3. Módulos y servicios del SCADA SAINUX

Los módulos y servicios del SCADA SAINUX son los que se describen a continuación:

Middleware o capa de comunicaciones:

Capa de software, que se encarga de la comunicación entre los diferentes módulos que forman parte del sistema.

Configuración:

Es el módulo encargado de almacenar, persistir y suministrar la información base para el funcionamiento de los demás módulos del sistema.

Seguridad:

El módulo Seguridad es el encargado de implementar el control de acceso a los recursos basados en privilegios asociados a usuarios y grupos de usuarios y provee un mecanismo de autenticación.

Adquisición:

Este servicio se encarga de la adquisición de datos para recolectar, procesar y almacenar, en tiempo real, información relevante sobre la evolución del proceso productivo.

Cuenta con un grupo de comandos (que son utilizados para que el operador ejecute una acción de control, obteniendo una respuesta que indica si el resultado es exitoso o no) entre ellos:

- Acciones variables (Modificar, forzar o desforzar el valor).
- Acciones sobre alarmas (Reconocer, inhibir, desinhibir, habilitar, deshabilitar y reiniciar).

El sistema actualmente cuenta con un esquema de redundancia para los servicios de comunicación y adquisición.

Este servicio además cuenta con un conjunto de manejadores (drivers) que soportan los siguientes protocolos de comunicación:

- Modbus (TCP/IP, RTU, ASCII, OMNI, Slave, ELAM).
- EthernetIP, ABEthernet (AB y Interchange).
- OPC DA.
- BSAP (IP y Serial).
- DNP3 (IP y Serial).

- Gefanuc.
- DF1 (Half-Duplex y Full Duplex).
- Virtual Manual.

Base de Datos Histórico (BDH):

Brinda servicio histórico de variables (puntos), alarmas, eventos, comunicaciones, con el objetivo de que esta pueda ser empleada posteriormente en la generación de reportes, tendencias, sumarios, entre otras. Siendo valioso para evaluar el comportamiento de los procesos en el tiempo.

Integración con terceros:

Permite a los sistemas gerenciales y a otros sistemas SCADA acceder a la información manipulada por el SAINUX, mediante servicios web.

Subsistema de Visualización:

La visualización de la información en el SCADA SAINUX se divide en:

- **Configuración**
  - ✓ Configuración de los recursos del sistema, módulos, puntos, alarmas, recetas, entre otros.
  - ✓ Edición de las interfaces gráficas de operación (Despliegues).
- **Ejecución**
  - ✓ Permite el monitoreo de la información del proceso mediante interfaces gráficas que modelen o simulen de forma simple y accesible el proceso.
  - ✓ Permite a los operadores el envío de comandos, visualización de datos históricos y en tiempo real en forma de gráficos de tendencia, XY y barras y el filtrado y manejo de alarmas y eventos.

Recolector gráfico:

Posibilita la recolección de puntos de supervisión de acuerdo a sus períodos de encuesta, mostrando en despliegues tabulares los valores obtenidos, así como sus marcas de tiempo y calidad.

Planificador ejecutor de tareas:

Se encarga de la configuración y ejecución de tareas y recetas en el sistema. Posibilita la creación y planificación de tareas de tipo síncronas, asíncronas, periódicas y aperiódicas.

Monitor de servicios:

Permite la configuración avanzada de manera visual de cada uno de los módulos del sistema, la visualización y auditoría de registros, detención y arranque de cada uno de los servicios del sistema.

1.4. Módulo Interfaz Hombre-Máquina

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[1]), unidades terminales remotas de entrada y salida o manejadoras. Todos estos dispositivos deben tener una comunicación que entienda el HMI. (9)

1.4.1. Funcionalidades del módulo 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: (9)

- **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. (10)

1.4.2.1. Funcionalidades del entorno de configuración.

El HMI en el SCADA SAINUX cuenta 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: (10)

[1] Por sus siglas en inglés (Programmable Logical Controllers).

- 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.

1.4.2.2. Edición gráfica

La edición gráfica es el mecanismo que permite 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 componentes gráficos:** Es el componente que contiene los objetos gráficos. Un objeto gráfico es un componente 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. (11)
- **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. (12)
- **Barra de menú:** Contiene todas las acciones para el diseño gráfico, salvar la configuración, abrir la configuración, entre otras.
- **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. (12)
- **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. (13)

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. (14)

1.5. Tipos de representación de la información histórica

Los sistemas SCADA, cuentan con dos tipos de representación de la información histórica que permiten visualizar el estado de sus variables y facilitar la toma de decisiones. Estos son la **representación tabular** y **la representación gráfica**, de los que se hará una breve descripción 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. (15)

Es un ordenamiento de la información en filas y columnas. Una buena representación tabular no debe prescindir de los siguientes elementos:

- Títulos y encabezamientos claros y completamente definidos.
- Incluir las unidades en las 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 se representa de forma tabular y gráfica. La siguiente ilustración muestra una representación tabular de un sumario de eventos:

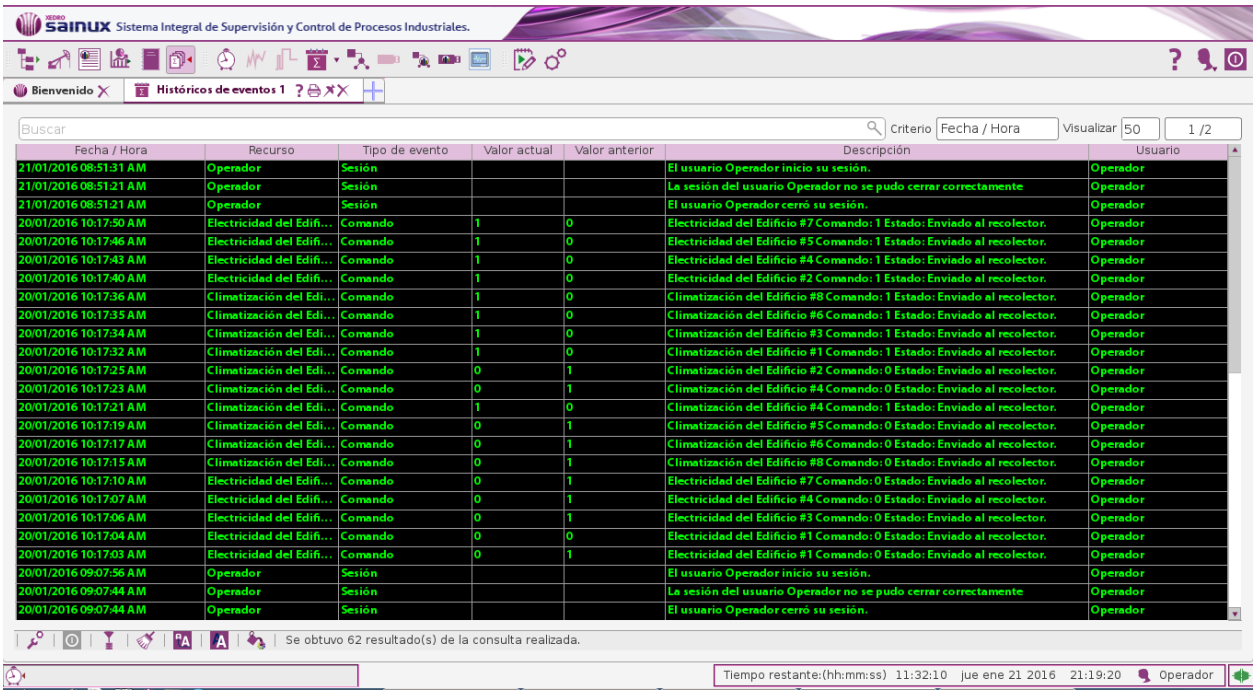


Ilustración 1 Representación tabular

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). (16)

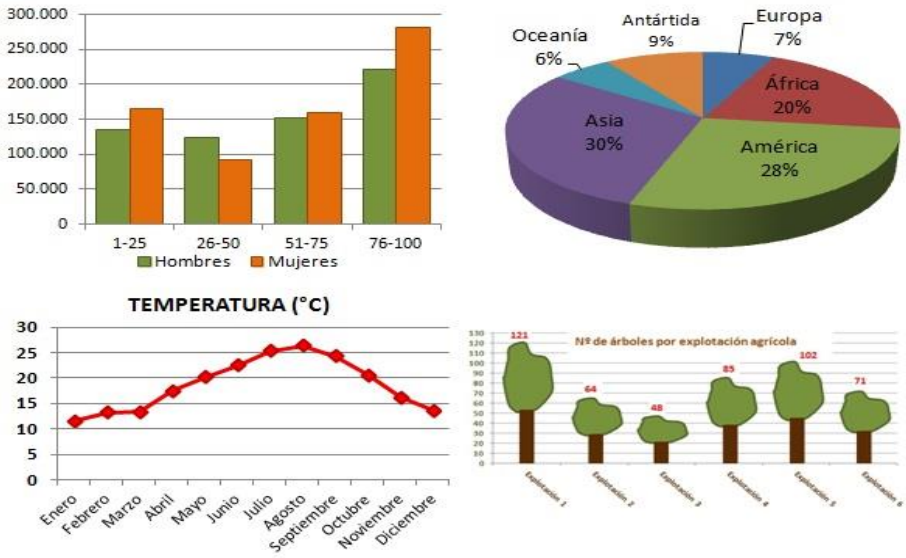


Ilustración 2 Representación gráfica

1.5.2.1. Tipos de representación gráfica

Los sistemas SCADA utilizan diversas formas de representar la información histórica de manera gráfica. El SCADA SAINUX cuenta con el gráfico de tendencia, el gráfico XY y el gráfico de barras; el gráfico de tipo sectores o de pastel no se encuentra actualmente en el sistema, por lo que este trabajo se centra en su desarrollo. A continuación se describirán de forma breve cada uno de estos tipos de gráficos.

Gráfico de tendencia

Este tipo de gráfico 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. (17)

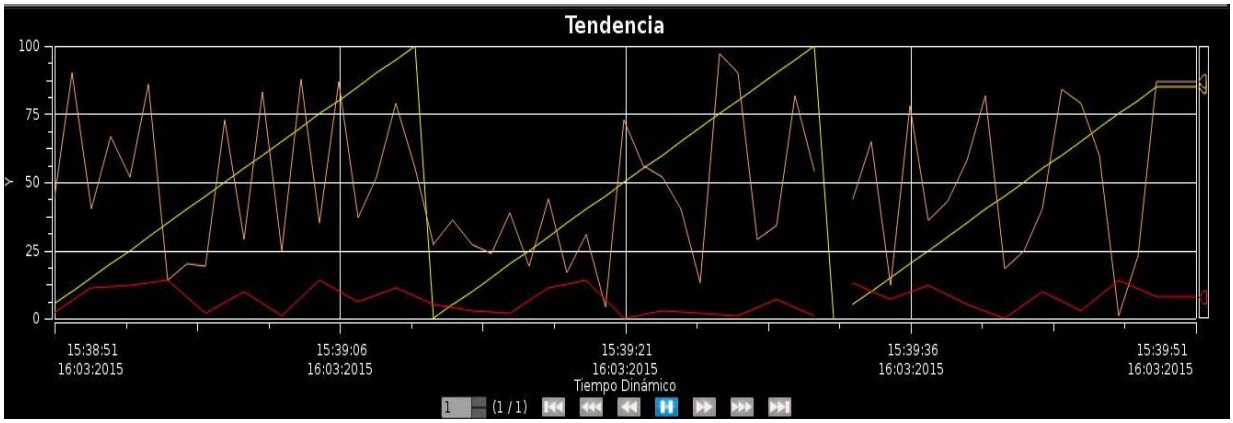


Ilustración 3 Gráfico de tendencia

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. (18)

Gráfico 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. (19)

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.

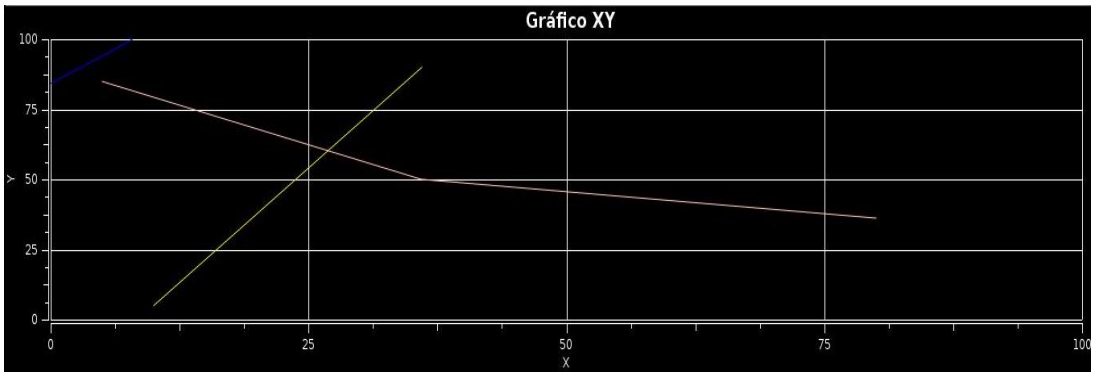


Ilustración 4 Gráfico XY

Gráfico 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 este 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 se quiera comparar dos o más grupos entre sí. (20)

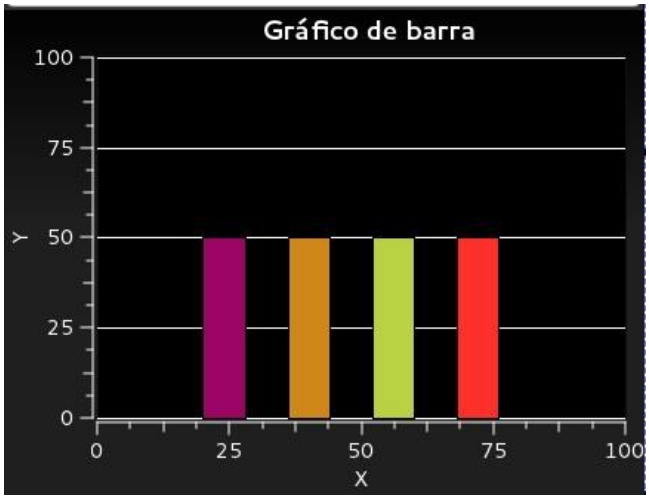


Ilustración 5 Gráfico de barras

Gráfico de tipo sectores o pastel

Una gráfica de tipo sectores, también llamada gráfico de pastel, es un recurso estadístico que se utiliza para representar porcentajes y proporciones. El número de elementos comparados dentro de una gráfica de pastel puede ser más de 4.

Al igual que en la gráfica de barras, el empleo de tonalidades o colores facilita la diferenciación de los porcentajes o proporciones. A diferencia de otros tipos de gráficos, el de sectores no tiene ejes X o Y.

Se utilizan en aquellos casos donde interesa no sólo mostrar el número de veces que se da una característica o atributo de manera tabular, sino más bien de manera gráfica, de tal manera que se pueda visualizar mejor la proporción en que aparece esa característica respecto del total y así obtener una mejor interpretación del comportamiento de las variables cualitativas o discretas.

Los gráficos de pastel consisten en subgrupos que son combinados para formar una unidad entera. Por ejemplo, en el caso de una extracción de petróleo, se pueden establecer porcientos con las mediciones de cuánto sedimento se extrajo, cuánto crudo limpio, cuánta agua y cuánto de residuos sólidos que estén en la pared interna del tanque, con el fin de interpretar la cantidad que representan estas sustancias excedentes del total y qué impacto puede tener.

Para construir un gráfico de sectores se deben tener en cuenta los siguientes elementos: (21)

- Se debe identificar el todo, así como sus partes.
- Cada elemento estudiado debe pertenecer solo a una categoría.
- Se deben representar las proporciones para cada categoría de la variable.
- La suma de las proporciones no debe exceder al 100%.

Debido a que en la gráfica de pastel lo importante es mostrar el porcentaje o proporción que le corresponde a cada categoría y no el orden, son más adecuadas para representar variables nominales en lugar de ordinales. (21)

Actualmente y debido al uso de software de aplicación general es muy común que se elaboren gráficas en perspectivas simulando tres dimensiones cuando solo se desea representar una o dos, lo que produce distorsión y una mala comunicación. (21)

Los gráficos de sectores tienen como ventajas:

- Funcionan como un apoyo visual para ayudar a la audiencia a examinar e interpretar la información que presentas. La información a menudo es difícil de entender cuando se presenta en grupos de estadísticas y cálculos en blanco y negro. Esta gráfica crea un modelo visual, que la gente puede usar cuando compara diferentes series de informaciones. Al usar colores diferentes, las gráficas circulares dividen la información en secciones que parecen rebanadas de pastel. Cada sección normalmente está acompañada de un porcentaje, y su tamaño cambia de acuerdo con ello.
- Usan porcentajes para ilustrar cantidades. Esta es una ventaja clara cuando se presentan estadísticas, como ganancias y gastos. La efectividad de estas gráficas para examinar porcentajes se encuentra en el entendimiento inmediato por parte de la audiencia de lo que intentas comunicar. La presentación sencilla de información la hace accesible a audiencias de todas las edades y niveles educativos.

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. (22)

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. (22)

El Proceso Unificado Ágil (AUP) de Scott Ambler es una versión simplificada del Proceso Unificado Racional (RUP[1]). 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: (22)

- 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: (22)

- 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.

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

- ❖ 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: (22)

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 llevan 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. Herramientas y tecnologías

Las herramientas y tecnologías constituyen el soporte fundamental a la hora de desarrollar un software, 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 pueden o no ser las mismas utilizadas para el desarrollo del software al que el componente será integrado, éstas se describen a continuación.

1.7.1 Lenguaje de modelado

UML 2.0

Es un lenguaje de modelado visual de propósito general orientado a objetos. Agrupa notaciones y conceptos provenientes de distintos tipos de métodos orientados a objetos. Su vocabulario y reglas se centran en la representación conceptual y física de un sistema y se ofrecen para crear y leer modelos bien formados que constituyen los planos de un sistema software. UML es independiente del proceso de desarrollo, esto quiere decir que un uso óptimo se consigue en procesos dirigidos por casos de usos, centrados en la arquitectura, iterativos e incrementales. Cubre las diferentes vistas de la arquitectura de un sistema mientras evoluciona a través del ciclo de vida del desarrollo del software. (23)

Entre sus ventajas se encuentran: (23)

- Es estándar, por lo que facilita la comunicación.
- Está basado en metamodelo con una semántica bien definida.
- Se basa en una notación gráfica concisa y fácil de aprender y utilizar.
- Se puede emplear para modelar sistemas software en diversos dominios.
- Es fácilmente extensible.

Se escogió este lenguaje, porque es estándar a la hora de conformar los planos de un software. Para comprender la estructura del componente gráfico se necesitan varios modelos y diagramas que estén conectados entre sí. La utilización de UML permite definir concretamente el proceso de desarrollo, porque es un lenguaje para visualizar, especificar, construir, documentar; y esto se logra a través de la creación de artefactos como: requisitos, arquitectura, diseño, prototipos.

1.7.2 Herramienta de modelado

Visual Paradigm 8.0

Visual Paradigm for UML (VP-UML) es una herramienta de diseño UML y herramienta CASE UML diseñado para ayudar al desarrollo de software. VP-UML soporta los principales estándares de modelado como Lenguaje de Modelado Unificado (UML), SysML, ERD, DFD, BPMN 2.0, ArchiMate 2,0. Es compatible con los equipos de desarrollo de software en la captura de requisitos, software de planificación (use el análisis de casos), el código de ingeniería, el modelado de clases, modelado de datos, entre otros. (24)

Entre sus principales características se encuentran: (24)

- Disponibilidad en múltiples plataformas (Windows, Linux).
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.

- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Disponibilidad de múltiples versiones, para cada necesidad.
- Licencia: gratuita.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Varios idiomas.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas de flujo de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Generador de informes.
- Editor de figuras.

Se escogió esta herramienta porque ya se ha empleado en el desarrollo del SAINUX para generar los artefactos correspondientes, por lo que es de vital importancia generar los del componente a desarrollar con la misma.

1.7.3 Lenguaje de programación

C++

El lenguaje está formando por instrucciones explícitas, cortas, cuya duración de ejecución puede preverse con antelación en el momento de escribir el programa. Como el número de instrucciones y notaciones ha sido limitado de forma voluntaria, las interpretaciones de las construcciones semánticas son múltiples. C++ está enriquecido con clases que describen tipos de datos adaptados a las diferentes necesidades del programador. La visibilidad del lenguaje, combinada con la abstracción de las clases, proporciona programas de alto nivel. (25)

En resumen, se puede decir que C++ es un lenguaje de alto nivel que, al mismo tiempo, se basa en instrucciones cercanas a la máquina. Un sutil equilibro que los desarrolladores aprecian.

Se ha seleccionado este, porque todo el sistema SAINUX está desarrollado con C++ y por motivos de compatibilidad con el código fuente, el componente debe implementarse en este lenguaje.

1.7.4 Marco de trabajo (framework)

Qt 4.8

Qt es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores. Contiene al framework Qt con clases de C++ modulares y multiplataforma, biblioteca y herramientas de desarrollo y productividad. (26)

Desde su fase inicial, se han lanzado varias versiones del framework, la versión 4.8, que es la que se utilizará en el desarrollo de esta investigación, incluye las siguientes mejoras: Qt Quick 1.1, el cual viene con nuevas propiedades y mejor rendimiento; Qt Platform Abstraction (QPA) – Lighthouse, que permite portar Qt a diferentes sistemas de ventanas y de dispositivos más fácil; Qt WebKit 2.2; Threaded OpenGL, cuyas funciones son libres de amenazas ahora; adiciones al Qt API; además de nuevas clases, funciones y macros. (27)

Se seleccionó este framework para desarrollar el componente para que sea compatible con el código fuente del SAINUX que fue ejecutado y elaborado en esta versión de Qt.

1.7.5 Entorno de desarrollo integrado

Qt Creator 2.4.1

Qt Creator es un entorno completo de desarrollo integrado (IDE) para la creación de aplicaciones con el framework Qt. Qt está diseñado para desarrollar aplicaciones e interfaces de usuario y desplegarlas a múltiples sistemas operativos tanto móviles como de escritorio. (28)

Las características claves de Qt Creator permiten a los programadores realizar las siguientes tareas: (28)

- Empezar a desarrollar aplicaciones con Qt de una manera rápida y fácil con el asistente de proyectos, y acceder rápidamente a proyectos recientes y sesiones.
- Diseñar aplicaciones de interfaz de usuario basadas en widgets Qt con el editor integrado, Qt Designer.
- Desarrollar aplicaciones con el editor de código avanzado de C++ que provee nuevas y poderosas características para completar recortes de código, remanufactura de código, y viendo el esquema de archivos (que es, la jerarquía de símbolos de un archivo).
- Compilar, correr y desarrollar proyectos Qt que se dirigen a múltiples plataformas de escritorio y móviles, como Microsoft Windows, Mac OS X, Linux, Symbian, MeeGo, y Maemo.

- Depurar con el depurador GNU y el CDB[1] usando una interfaz gráfica de usuario con una mayor conciencia de las estructuras de clases Qt.
- Usar herramientas de análisis de código para verificar la administración de memoria en sus aplicaciones.
- Desarrollar aplicaciones para dispositivos móviles y crear paquetes de instalación para dispositivos Symbian, MeeGo, y Maemo que pueden ser publicadas en la tienda Ovi y otros canales.
- Acceder fácilmente a información con el módulo del sistema de ayuda contextual de Qt.

Se escoge este IDE, porque de los compatibles con Qt 4.8, es el más apropiado; pues todo el código que genera es completamente basado en Qt 4.8; mientras que otras versiones superiores del Qt Creator, generan código de Qt 5 y esto puede ocasionar problemas a la hora de integrar el componente al código fuente.

1.7.6 Tecnologías para la implementación del componente

Debido a la variedad de tecnologías disponibles para la implementación del componente gráfico de sector para el HMI del SCADA SAINUX, se establecerá una comparación entre algunas de ellas, con el fin de seleccionar la más amigable y confiable para su desarrollo. Las tecnologías seleccionadas son la biblioteca Qwt 6.1.0, la Qt Charts 2.0, el módulo gráfico del Qt 5 QWidgets/Chart y QSint 0.2.2. Los aspectos a comparar son: características, rendimiento y tipo de licencia empleada.

Tabla 1 Comparación entre tecnologías

Tecnologías	Características	Rendimiento	Tipo de Licencia
Qwt 6.1.0	La biblioteca Qwt contiene componentes GUI y clases de utilidad que son realmente útiles para programas con base técnica e industrial. Además de ser una biblioteca para trabajar con diagramas 2D; provee escalas, compases, termómetros, entre otros;	Alto rendimiento y bajo uso de memoria para procesar el pintado de los gráficos.	LGPL[2] de GNU[3].

[1] Console Debugger (depurador de consola).
[2] Licencia Pública General Reducida (por sus siglas en inglés Lesser General Public License).
[3] GNU no es Unix (por sus siglas en inglés GNU's Not Unix) (Sistema operativo libre).

	para controlar los valores de salida, arreglos o rangos de tipo double. (29)		
Qt Charts 2.0	El módulo Qt Charts provee una serie de componentes fáciles de utilizar para la implementación de los gráficos. Además utiliza el Qt Graphics View Framework, por lo que los gráficos se pueden adaptar fácilmente a interfaces de usuarios modernas. Qt Charts se puede usar como un tipo de QWidgets, QGraphicsWidget o QML[1]. (30)	Alto rendimiento con funcionalidades para optimizarlo.	LGPL de GNU.
QtWidgets/ Chart del Qt 5	Es una forma de representar en un gráfico de pastel, los datos registrados en una tabla, para una mejor interpretación; apoyada en la flexibilidad de la arquitectura modelo/vista. Solo se necesita crear una nueva clase vista si los datos necesitan una forma de representación especializada. Este módulo viene incluido con la instalación del Qt Creator. (31)	Alto rendimiento.	LGPL v3.
QSint 0.2.2	Es una colección libre de QtWidgets que se puede emplear en el desarrollo de aplicaciones basadas en Qt. La biblioteca está diseñada para usarse en la versión de Qt 4.x y la 5.x. Entre los módulos más importantes se encuentran el QSint::Core y el QSint::Charts. (32)	Alta rendimiento y bajo uso de memoria.	LGPL de GNU.

[1] Lenguaje de Meta-objeto de Qt (por sus siglas en inglés Qt Meta-object Language).

1.7.7 Conclusiones de la comparación

Qt Charts 2.0 posee funcionalidades variadas y concretas de fácil empleo a la hora de implementar el gráfico de pastel; posee ejemplos ya implementados y probados en aplicaciones reales; utiliza una licencia libre, que es la misma que se ha empleado para el sistema al que será integrado, pero no es una biblioteca; es un módulo de Qt 5.x, lo cual no concuerda con la versión que se debe usar del Qt, que es la 4.x.

En cambio la biblioteca Qwt 6.1.0, ha sido utilizada para el desarrollo de los componentes gráficos ya existentes en el SCADA SAINUX y también emplea un tipo de licencia libre, pero no incluye funcionalidades que permitan la implementación del gráfico de pastel, por lo que queda descartada.

Por otra parte, el módulo del Qt Creator, QtWidgets/Chart, tiene funcionalidades disponibles para diseñar y desarrollar un gráfico de pastel, pero no son tan amplias y variadas como para suplir las funcionalidades que debe tener el componente.

La biblioteca QSint en su versión 0.2.2, se adecúa y se adapta a la arquitectura a emplear para el desarrollo del componente visual. Las clases intrínsecas permiten el pintado de un gráfico de pastel dinámico y eficaz a la hora de mostrar e interpretar los datos y es compatible con la versión de Qt framework a utilizar y el tipo de licencia adecuado, pero hay algunas funcionalidades del componente que la biblioteca no permite implementar

Por todo lo anteriormente expuesto, se decidió no emplear ninguna de estas tecnologías, pero sí tomar y reimplementar métodos de alguna de ellas para facilitar el pintado el pastel, empleando para la implementación del gráfico, las clases modelos del mismo y sus facilidades y funcionalidades.

1.8. Conclusiones parciales del capítulo

Al finalizar el capítulo 1 se arribó a las siguientes conclusiones parciales:

- El estudio de los principales conceptos brindó un acercamiento a los sistemas de control automatizados, sus funciones en general y a las características del SCADA SAINUX, sistema al que será integrado el componente.
- La descripción detallada de los módulos y servicios del SCADA SAINUX, propició un mejor conocimiento de las funcionalidades concretas de este sistema y los servicios que brinda.
- Se realizó un análisis de los tipos de representación de la información del SAINUX, los tipos de gráficos ya existentes y se llegó a la conclusión que se necesita un gráfico que permita la interpretación de esta información.

- Posteriormente se hizo una caracterización de las herramientas y tecnologías necesarias para el desarrollo de la investigación, tanto para el modelado como para el desarrollo del componente, con esto se logró trazar un camino a seguir para una mejor organización de la investigación.
- Por último, la comparación de las tecnologías para implementar el gráfico de pastel, condujo a la elección de los métodos necesarios de algunas bibliotecas para su reimplementación y otras funcionalidades del Qt; los cuales brindan las herramientas necesarias para lograr el objetivo propuesto.

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 analizan los conceptos fundamentales del modelo de dominio, necesarios para comprender el problema planteado. También se define una lista de requisitos funcionales y no funcionales que se deben tener en cuenta para la elaboración del componente y se describen las principales clases.

2.2. Descripción de la solución

El componente propuesto será incorporado a los componentes de visualización gráfica existentes actualmente en el HMI del SCADA SAINUX, permitiendo realizar un análisis e interpretación estadísticos a partir de la visualización de un gráfico de tipo sectores y ha de cumplir los requisitos descritos en el epígrafe 2.4.

2.3. Modelo de dominio

Un modelo del dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos software. Es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. También se les denomina modelos conceptuales, modelo de objetos del dominio y modelos de objetos de análisis. (33)

2.3.1. Diagrama del modelo de dominio

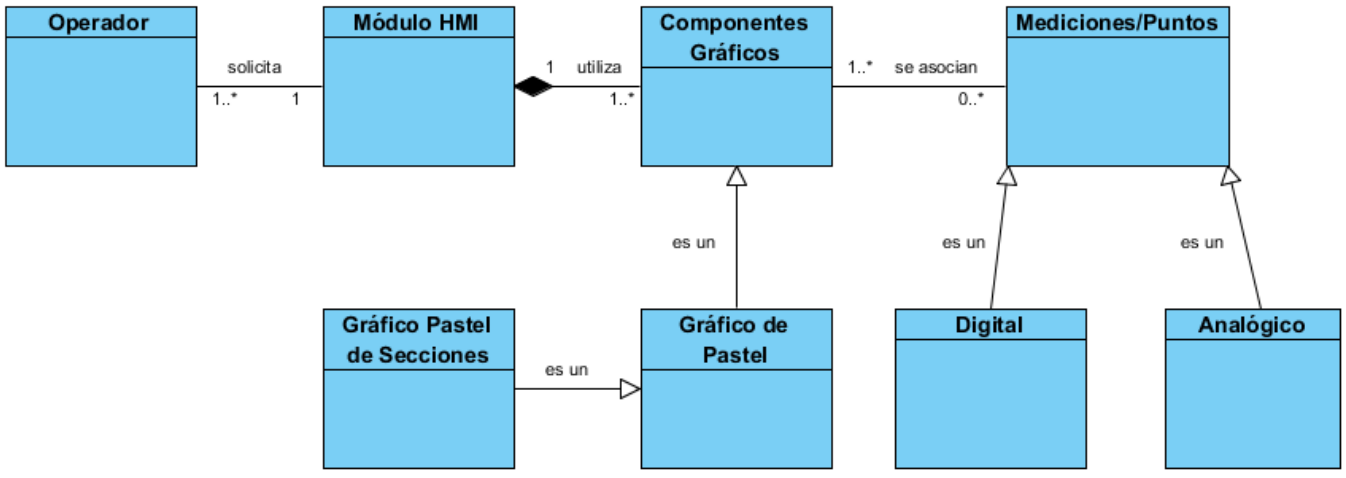


Ilustración 6 Diagrama del modelo de dominio Gráfico de Pastel

2.3.2. Descripción de los conceptos del dominio

Tabla 2 Descripción de los conceptos del dominio

Conceptos del dominio	Descripción
Operador	Comienza el proceso, haciendo una petición a la Interfaz Hombre-Máquina del SAINUX en espera de la respuesta deseada.
Módulo HMI	Contiene el gráfico de pastel que se desea visualizar el operador.
Componentes gráficos	Entre ellos se encuentra, el gráfico XY, el gráfico de tendencia, el gráfico de barras y el gráfico de pastel.
Mediciones o Puntos	Valores asociados a un despliegue previamente creado para ser analizados, los cuales pueden ser digitales o analógicos.
Gráfico Pastel	Componente gráfico del despliegue al que se le pueden asociar mediciones con valores fijos o valores en tiempo real del sistema.
Gráfico Pastel de Secciones	En conjunto con las configuraciones hechas por el operador contiene de uno (1) a cinco (5) secciones con valores fijos (rangos desde el valor mínimo hasta el valor máximo que puede tomar una variable específica del sistema) de las mediciones del sistema. Además contienen las mediciones que representan gráficamente los valores reales asociados a las variables del sistema con valores fijos especificadas por el operador.

2.4. 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. (34) Pueden ser funcionales o no funcionales.

2.4.1. Requisitos funcionales (RF)

Definición de los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar antes situaciones particulares. (34)

Tabla 3 Requisitos funcionales

Requisitos funcionales	Descripción
RF 1: Visualizar componente	El sistema debe permitir visualizar el componente al hacer *drop*[1] sobre el despliegue.
RF 2: Adicionar sección	El sistema debe permitir adicionar una sección con un valor fijo de una medición del sistema.
RF 3: Eliminar sección	El sistema debe permitir eliminar una sección con un valor fijo de una medición del sistema.
RF 4: Modificar color de la sección	El sistema debe permitir modificar el color asignado a la sección seleccionada.
RF 5: Adicionar subsección	El sistema debe permitir adicionar una subsección con un valor real dentro de una sección con un valor fijo de una medición del sistema.
RF 6: Generar gráfico de pastel real	El sistema debe permitir generar un gráfico de pastel a partir de los valores de todas las subsecciones con valores reales de una medición del sistema.
RF 7: Mostrar propiedades	El sistema debe permitir mostrar las propiedades a configurar para visualizar el componente.
RF 8: Guardar como imagen	El sistema debe permitir guardar la información gráfica contenida en el componente como una imagen.
RF 9: Exportar a XML	El sistema debe permitir exportar la información gráfica contenida en el componente como XML.
RF 10: Imprimir	El sistema debe permitir imprimir la información gráfica contenida en el componente.
RF 11: Configurar propiedades generales	El sistema debe permitir configurar todas las propiedades comunes de los gráficos de HMI: alto, ancho, posición, leyenda visible, entre otras.

[1] Acción de soltar.

2.4.2. Requisitos no funcionales (RNF)

Son restricciones que afectan a los servicios o funciones del sistema, tales como restricciones de tiempo, sobre el proceso de desarrollo, estándares. Definen propiedades emergentes del sistema, tales como el tiempo de respuesta, la fiabilidad, entre otras. (34)

2.4.2.1. Requisitos de usabilidad

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

2.4.2.2. Requisitos de confiabilidad

RNF 2: Debe existir una comunicación eficiente y fiable, evitando que existan menos pérdidas de información de variables y eventos.

RNF 3: Integridad de las configuraciones o parámetros: el sistema debe ser capaz de mantener la integridad de las configuraciones o parámetros de las tareas.

RNF 4: 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.

2.4.2.3. Requisitos de desempeño

RNF 5: 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 1 segundo. En particular el tiempo para cambiar dos ventanas debe ser inferior a 1 segundo.

RNF 6: 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 7: Tiempo para el refrescamiento de datos en el componente gráfico de 500 milisegundos.

RNF 8: Durante el intercambio de variables, debe existir una comunicación eficiente, garantizando una velocidad en la comunicación adecuada.

2.4.2.4. Restricciones de implementación

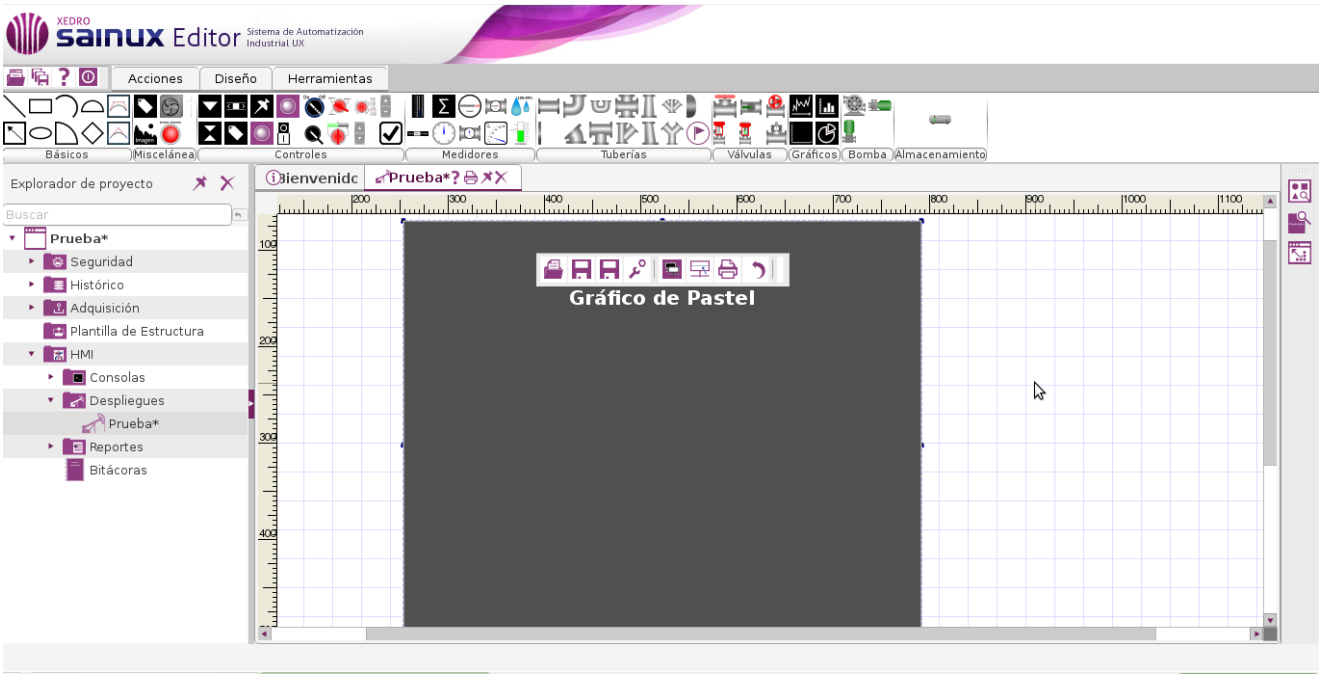
RNF 9: El gráfico sólo admitirá como máximo dieciséis (16) secciones, ya que al añadir una decimoséptima, el color puede ser muy similar a los 16 anteriormente definidos.

2.5. Historias de usuario

Los modelos ágiles utilizan historias de usuario para captar las necesidades de los clientes en un proyecto de software. Una historia de usuario, es una descripción en primera persona y de alto nivel de una acción que usuario efectúa en un sistema. (35)

2.5.1. Descripción de las historias de usuario

Tabla 4 HU1 Visualizar componente

Historia de Usuario	
Número: HU1	**Nombre del requisito:** Visualizar componente
Programador: Rolando Futiel Gutiérrez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: ❖ Problemas eléctricos. ❖ Problemas técnicos.	**Tiempo real:** 2 semanas.
Descripción: El operador será capaz de adicionar el componente gráfico de pastel a un despliegue previamente creado. Para esto, el operador debe seleccionar el Gráfico de Pastel de la paleta de componentes y hacerle *drop* dentro del despliegue.	
Observaciones:	
Prototipo de interfaz: 	

Nota: Para ver el resto de las historias de usuario, ver desde el anexo 1 hasta el anexo 7.

2.6. 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. (36) A continuación se muestran algunos de los diagramas diseñados para la solución propuesta.

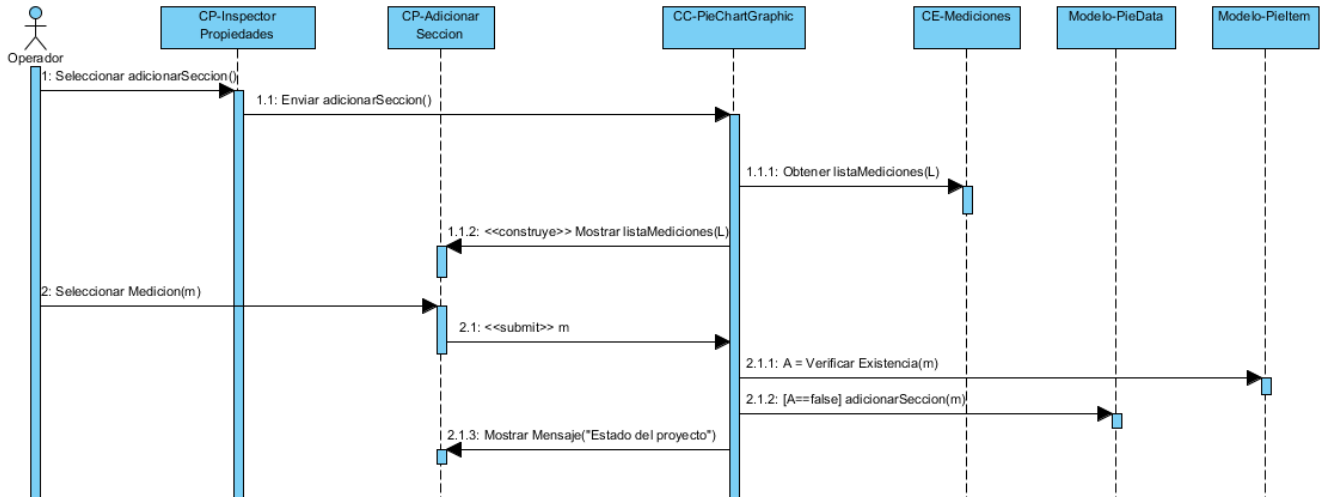


Ilustración 7 Diagrama de secuencia Adicionar Sección

Nota: Para ver el resto de los diagramas de secuencia, ver los anexos 8 y 9.

2.7. Diagrama de clases

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de composición. Está compuesto por varios elementos como las clases, que son; una descripción de conjunto de objetos que comparten los mismos atributos, objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica. (37) A continuación se muestra el diagrama de clases del diseño teniendo en cuenta las entidades, sus atributos y relaciones.

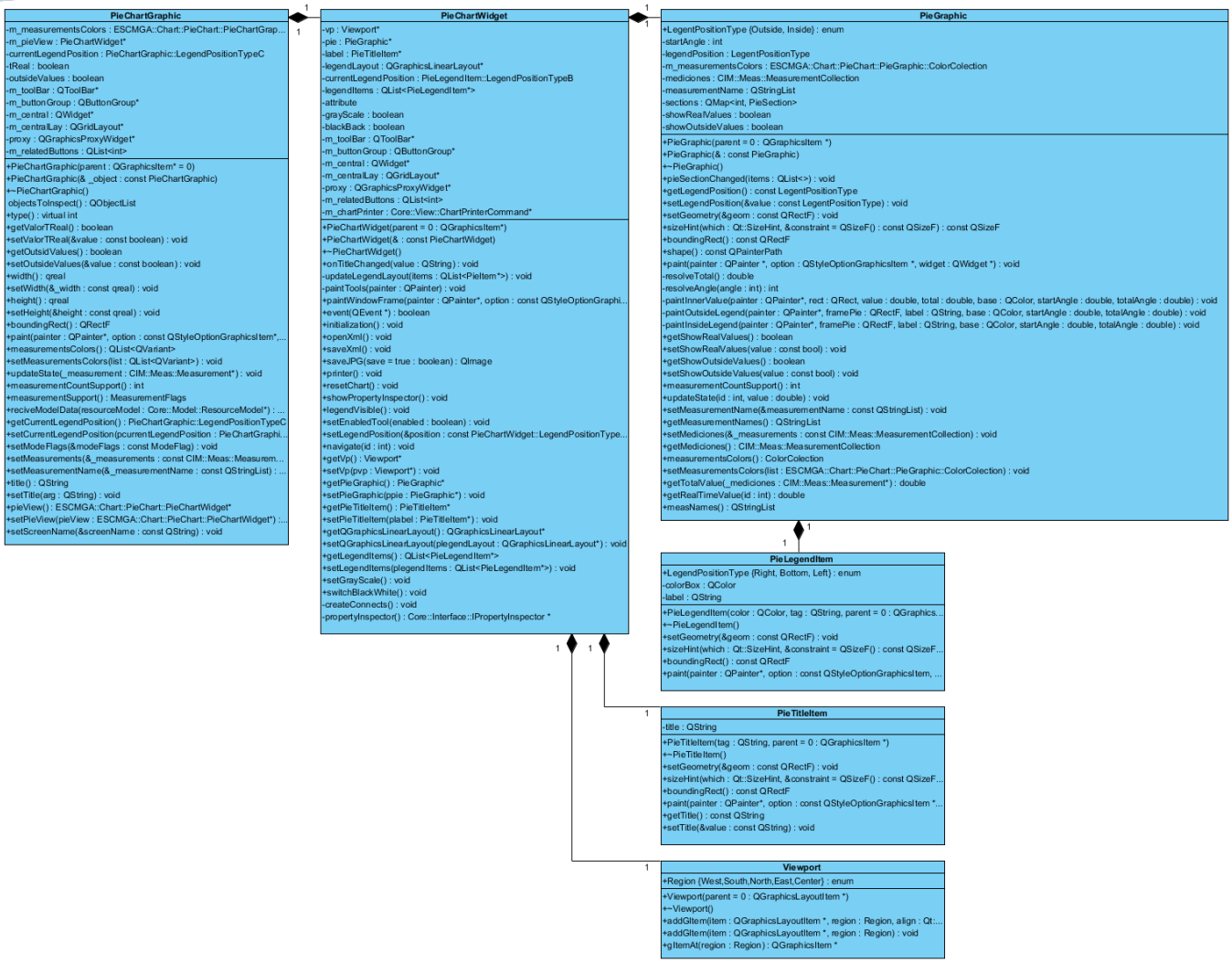


Ilustración 8 Diagrama de clases Gráfico de Pastel

2.8. 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. (38)

La arquitectura escogida para el desarrollo de la solución propuesta es la Modelo/Vista, debido a que es la arquitectura base del marco de trabajo Qt y satisface mejor las necesidades del componente.

2.8.1. Arquitectura 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. (39)

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. (39)

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. (39)

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. (39)

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. (39)

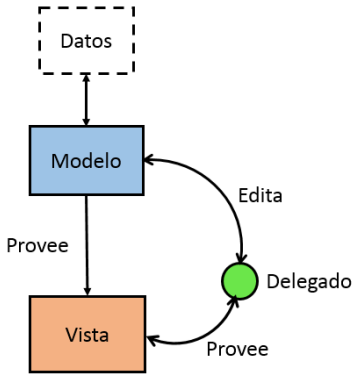


Ilustración 9 Arquitectura Modelo/Vista

- Las señales desde el modelo informan a la vista de cambios en los datos almacenados en la fuente. (39)
- Las señales desde la vista proveen información sobre la interacción del usuario con los objetos de datos que son mostrados. (39)
- Las señales desde el delegado se usan durante la edición para comunicarle al modelo y a la vista sobre el estado del editor. (39)

2.9. Patrones de diseño

Un patrón de diseño da un esquema para refinar sus subsistemas o componentes, o las relaciones entre ellos. Describe la estructura de una solución a un problema que aparece repetidamente; de componentes que se comunican entre ellos. (40)

En la construcción de la solución, para garantizar una buena práctica de programación, se emplearon los patrones de diseño GOF[1]: observador y el método plantilla. El primero se ha utilizado en la clase PieChartGraphic, que sería observador de PieGraphic, PieLegendItem, PieTitleItem y Viewport, pues en el momento de actualizar las mediciones representadas en el componente, inmediatamente se actualizan los observadores. El segundo se empleó en la reimplementación del método para pintar el gráfico de pastel del Qt, QGraphicsLayoutItem y QGraphicsObject, sin violar o cambiar su estructura principal.

También se emplearon patrones GRASP[2]; de los cuales se emplearon el creador que se evidencia en la clase PieChartGraphic para la creación de un objeto de tipo PieChartWidget y en la clase PieChartWidget para la creación de objetos de tipo PieGraphic, PieTitleItem, Viewport. El patrón controlador se evidencia en la clase PieChartWidget, pues esta controla las funcionalidades internas de sus clases subordinadas. El patrón experto se emplea en la clase PieGraphic, pues es la encargada de pintar el gráfico con todas sus propiedades. El bajo acoplamiento y la alta cohesión se emplearon en las clases para reducir a lo mínimo posible la cantidad de relaciones de una clase con las demás y asignar a cada una la cantidad mínima de responsabilidades.

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

2.9.1. Patrones GOF

2.9.1.1. 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. (41)

[1] Pandilla de los cuatro; por sus siglas en inglés (Gang of Four).
[2] Por sus siglas en inglés (General Responsability Assingment Software Patterns)

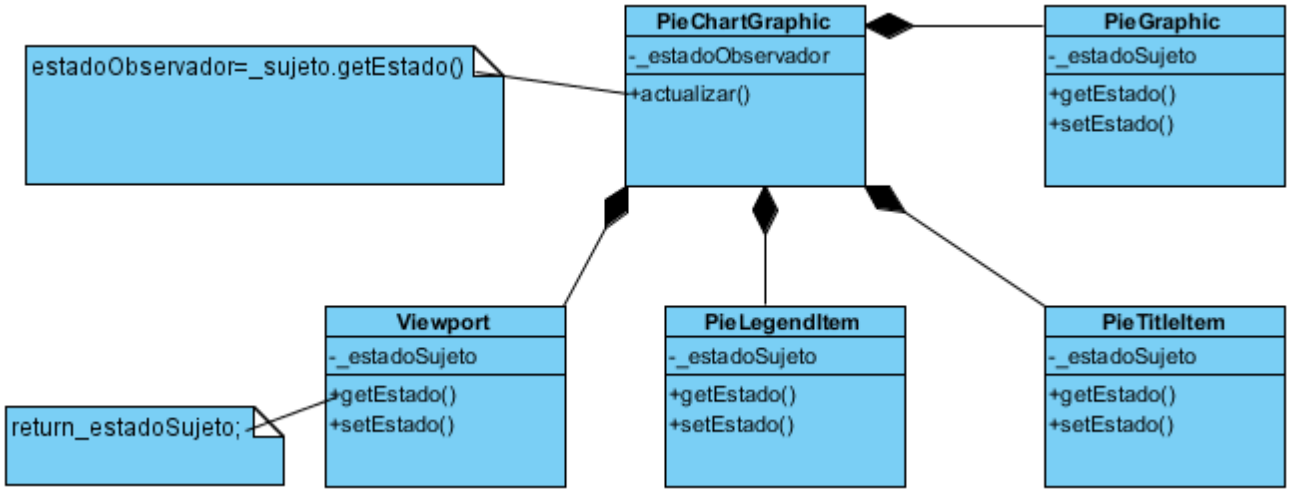


Ilustración 10 Patrón Observador

- **Sujeto (Subject):**

  El sujeto proporciona una interfaz para agregar (attach) y eliminar (detach) observadores. El Sujeto conoce a todos sus observadores. (41)

- **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. (41)

- **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. (41)

2.9.1.2. Patrón Método planilla (method template)

Define, en una operación, el esqueleto de un algoritmo delegando en las subclases algunos de sus pasos. (42)

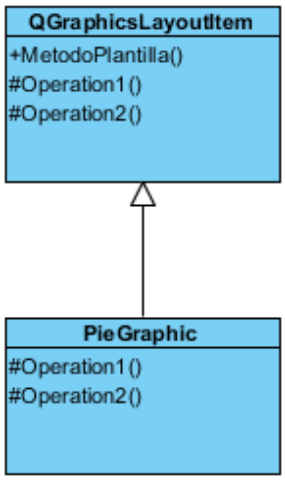


Ilustración 11 Patrón Método de Plantilla

- **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. (42)
- **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. (42)

2.9.2. Patrones GRASP

2.9.2.1. Patrón Creador

Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. (42)

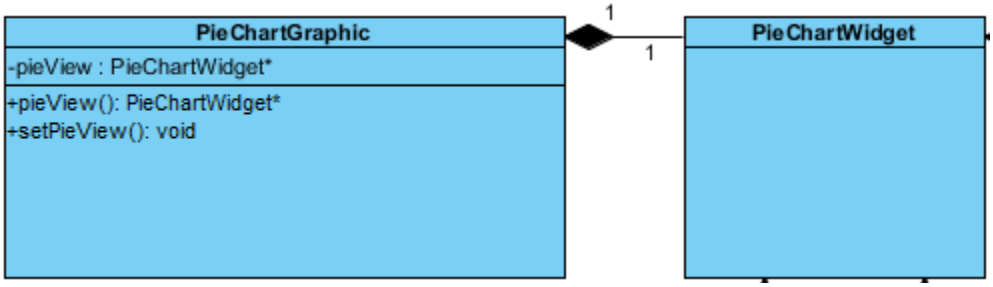


Ilustración 12 Patrón Creador

2.9.2.2. Patrón Experto

Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. (42)

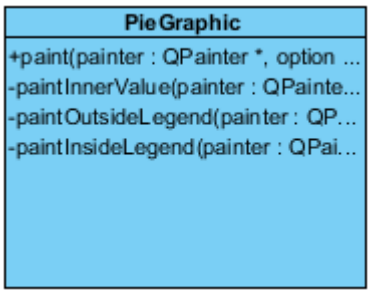


Ilustración 13 Patrón Experto

2.9.2.3. Patrón Controlador

Plantea asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Donde un evento del sistema es un evento de alto nivel generado por un actor externo. (42)

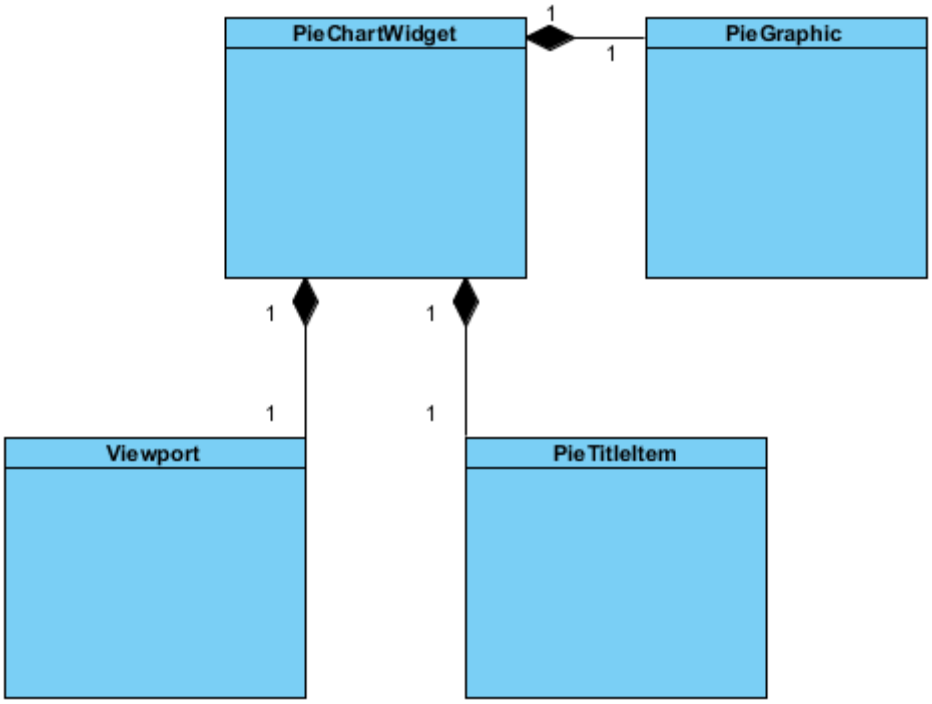


Ilustración 14 Patrón Controlador

2.9.2.4. Patrón 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. (42)

2.9.2.5. Patrón 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. (42)

2.10. Conclusiones parciales del capítulo

Al finalizar el capítulo 2 se arribó a las siguientes conclusiones:

- Debido al proceso de análisis y solución propuesta, se puedo diseñar un modelo de dominio, el cual describe el funcionamiento e interacción del componente con el operador.
- Se capturaron los requisitos funcionales y no funcionales, para dirigir el desarrollo del componente de acuerdo con las necesidades imperativas que debe resolver, así como las especificaciones para la usabilidad, rendimiento y otros que debe cumplir. Los primeros fueron descritos detalladamente en las historias de usuario.
- Para completar el proceso de diseño, se generó el diagrama de clases que estructura el componente y sus relaciones internas; se definió la arquitectura a emplear: modelo/vista y los patrones de diseño.

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 tipo sectores como parte del objetivo de evolucionar en desarrollar, al HMI del SCADA SAINUX con los diferentes tipos de gráficos que permitan evaluar e interpretar el desempeño y los cambios sufridos por las variables durante su representación gráfica. Al igual que incorporarán 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. (43)

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. (44)

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. (44)

A continuación se muestra el diagrama de componentes generado para el componente Gráfico de Pastel para el HMI del SCADA SAINUX.

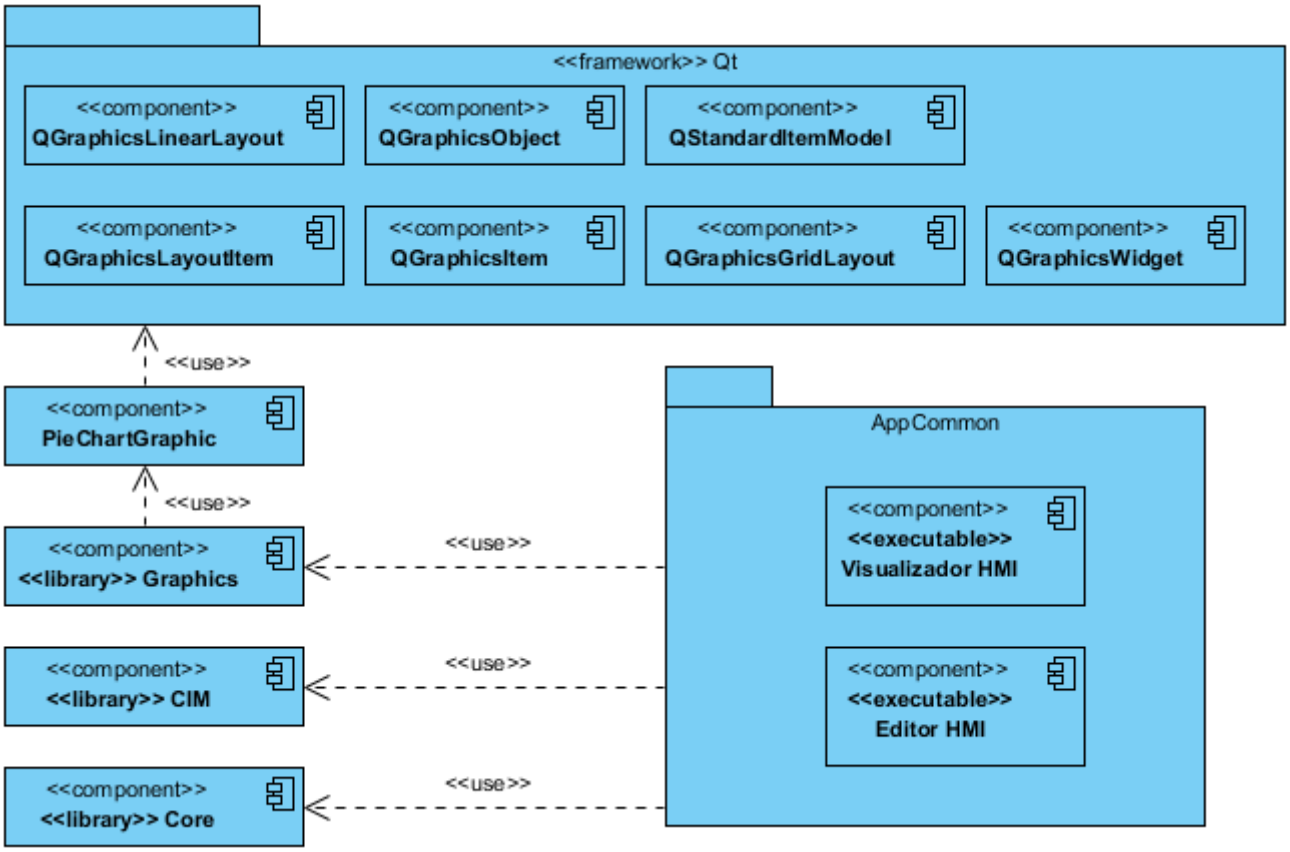


Ilustración 15 Diagrama de componentes Gráfico de Pastel

El SCADA SAINUX cuenta con elementos comunes para los ambientes de ejecución, el Editor HMI y el 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 PieChartGraphics que cuenta con varios componentes del marco de trabajo Qt para la gestión de las distintas secciones y subsecciones, sus datos y de igual manera, 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. (44)

- 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 muestra el diagrama de despliegue modelado para el componente gráfico a desarrollar.

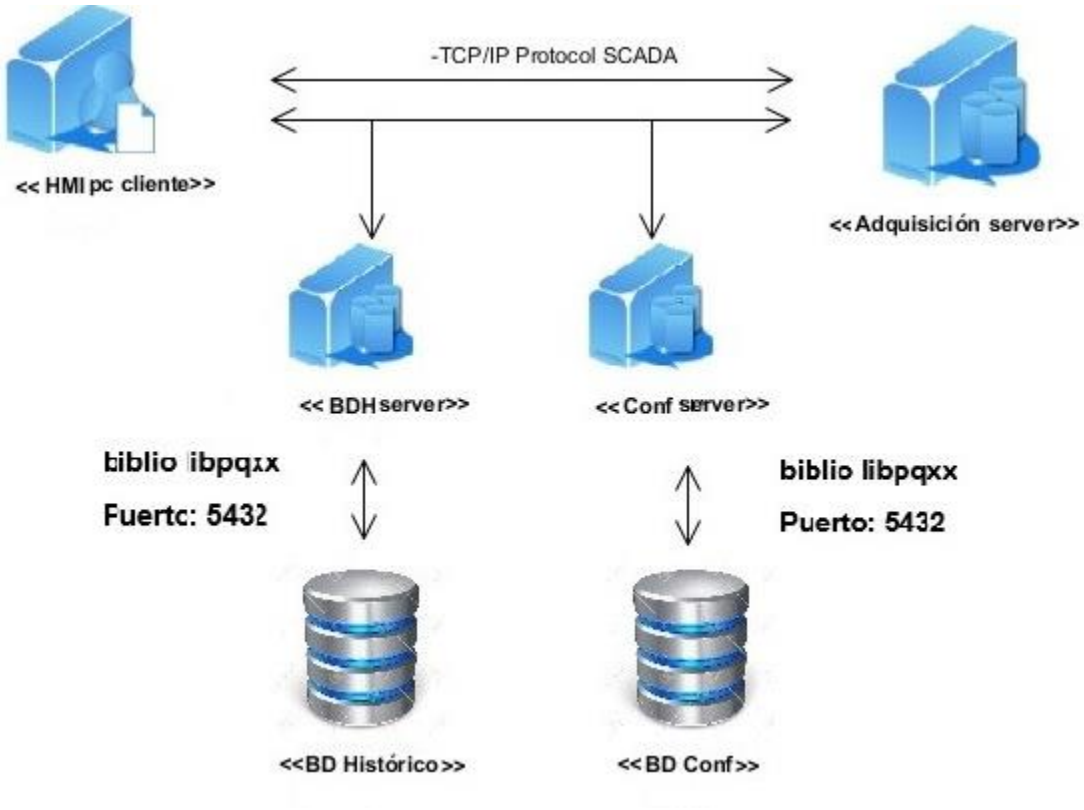


Ilustración 16 Diagrama de despliegue Gráfico de Pastel

3.3. Estándar de codificación

Es un complemento a la programación, su propósito fundamental es tener una arquitectura y un estilo consistente, independiente del autor, con lo cual el sistema resulte fácil de entender y mantener. Debe promover la intención del código. (45)

Para la implementación del componente Gráfico de Pastel es necesario utilizar el estándar de codificación de C++ establecido 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 Gráfico de Pastel dentro del código actual del HMI del SCADA SAINUX como componente visual dentro de la paleta de componentes

En el desarrollo actual del HMI del SCADA SAINUX, se deben seguir ciertas funcionalidades para lograr incorporar un nuevo componente gráfico a la paleta de componentes y que éste se logre visualizar correctamente.

A continuación se describirán los pasos a seguir para lograr la incorporación del ícono del Gráfico de Pastel 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 PieChartPalette::id() const
{
        return METATYPE_ID(ESCMGA::Chart::PieChart::PieChartGraphic);
}
```

En este caso se retorna el id de la clase **ESCMGA::Chart::PieChart::PieChartGraphic** 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 Pastel"*.

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

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 PieChartPalette::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 PieChartPalette::toolTip() const
{
             return QObject::trUtf8("Gráfico de Pastel");
}
```

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 PieChartPalette::icon() const
{
    return RESOURCE_ICON("PieIcon.png");
}
```

3.5. Creación del componente gráfico que se visualizará al hacer *drop* al componente seleccionado: Gráfico de Pastel, 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**. La cual hereda de **Shape**, que hereda de **GraphicObject** y esta, hereda de **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*** (redimensionar) al mismo, dentro del despliegue.

```
qreal PieChartGraphic::width() const
{
   return m_pieView->geometry()->width();
}
void PieChartGraphic::setWidth(const qreal &_width)
{
    m_pieView->setGeometry(_width, height());
}
qreal PieChartGraphic::height() const
{
    return m_pieView->geometry()->height();
}
void PieChartGraphic::setHeight(const qreal &_height)
{
    m_pieView->setGeometry(width(), _height);
}
```

2. `QRectF boundingRect() const;`

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

```
QRectF PieChartGraphic::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.

4.
```
void PieChartGraphic::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)

{
    Q_UNUSED( widget );
```

```
    paintCorners( painter, option );
}
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 a que el componente es un observador de dicha medición.

3.6. Proceso de 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. (46)

Para realizar las pruebas al componente Gráfico de Pastel, se decidió el uso de las pruebas de aceptación, empleando el método de caja negra y la técnica de partición de equivalencia.

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. (47)

A partir de las historias de usuario definidas se realizaron casos de prueba; la descripción se muestra a continuación.

Tabla 5 CP Aceptación 1 Visualizar componente

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 pastel 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.	

Nota: Para ver el resto de los casos de prueba de aceptación, ver desde el anexo 10 hasta el anexo 16.

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

Tabla 6 Resultados del proceso de pruebas

Sistema	HU	Iteración	NC	Cerrada	No Procede
Componente Gráfico de Tipo Sector.	8	1ra	20	17	3
		2da	7	7	0
		3ra	0	0	0

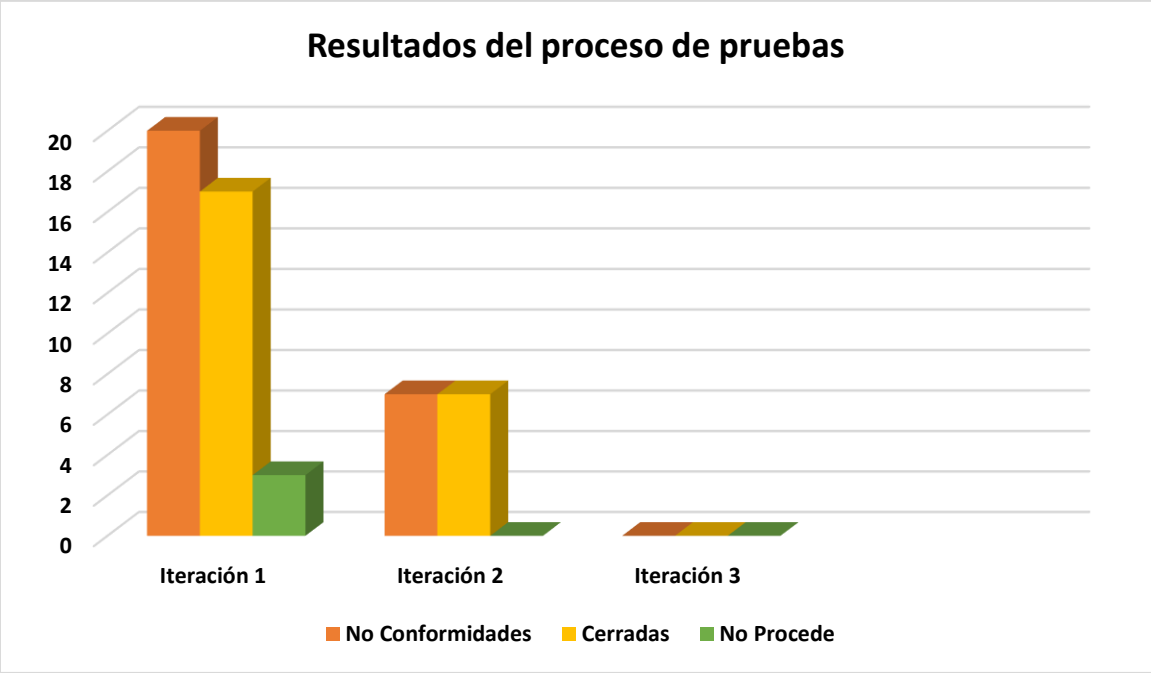


Gráfico 1 Resultados del proceso de pruebas

Al obtener resultados satisfactorios durante el proceso de pruebas, se prevé un alto grado de calidad de software.

3.7. Conclusiones parciales del capítulo

Al finalizar el capítulo 3, se arribó a las siguientes conclusiones:

- Se expusieron los principales artefactos del modelo de implementación, mostrando así los componentes del sistema y su relación a través del diagrama correspondiente.
- Así como los vínculos de comunicación que existen 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 Pastel se concluye que:

- Se logra una mejor interpretación de los datos que almacenan las mediciones, lo que permite evaluar el valor existente en tiempo real con respecto a su rango de ocurrencia.
- Con la utilización de los métodos de las tecnologías estudiadas para la implementación del componente, es posible lograr representaciones visuales con una mayor profesionalidad de representación gráfica e interpretación.
- Las diferentes formas de configurar el componente propuesto, hacen que el entorno de trabajo sea amigable para los operadores del sistema, logrando de esta forma la aceptación de los mismos.
- La incorporación del gráfico de pastel al HMI del SCADA SAINUX, junto con sus funcionalidades especializadas, enriquece la gama de componentes gráficos disponibles para el trabajo con las mediciones y las variables.

RECOMENDACIONES

Se recomienda que se continúe el desarrollo de otras funcionalidades dentro del Gráfico de Pastel que aporten mejoras al componente como:

- Guardar la información del gráfico en formato CSV[1].
- Cambiar el estilo del gráfico a 3D con secciones flotantes.
- Agregar el gráfico como componente externo en la paleta de gráficos en el visualizador.

[1] Según sus siglas en inglés (comma-separated values).

REFERENCIAS

1. **Strothman, J.** *Leaders of the Pack.* s.l. : In Tech 50, 2006. pp. 122-127. 1.

2. **Ramos, Marco Antonio Paz.** *Análisis, Síntesis y Contsrucción de un Controlador Adaptable Genérico con Supervisión Inteligente.* Morelos, México : s.n., 2006. Tesis Doctoral. 2.

3. **Quiroz, Juan Herrera.** Electro Industria. [Online] Marzo 2004. http://www.emb.clelectroindustriaarticulo.mvcxid=81.

4. **Divestadística.** Diccionario estadístico. [Online] 2015. [Cited: Octubre 14, 2015.] http://www.divestadistica.es/es/diccionario_estadistico.html.

5. **Ditutor.** Ditutor.com. [Online] 2015. [Cited: Noviembre 12, 2015.] http://www.ditutor.com/estadistica/variable_discreta.html.

6. **Ordonez, Gabriel.** OoCities. [Online] Mayo 2011. http://www.oocities.org/gabrielordonez_ve/SISTEMAS_SCADA.htm.

7. **Nieto Doce, Yanet, et al.** *Sistemas SCADA.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2009.

8. **Automatas.org.** Automatas.org. [Online] Marzo 2006. http://www.automatas.org/redes/scadas.htm.

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

10. **Bornot Rivero, María Mercedes y Rivera Acosta, Alexei.** *Desarrollo de componentes gráficos para el módulo de visualización del SCADA nacional.* La Habana : s.n., 2010.

11. **Sehara Driggs, Yosell Luis y Companioni Sardiña, Yusmary.** *Herramientas de configuración para sistemas SCADA.* La Habana : s.n., 2010.

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

13. **Acevedo Sánchez, José.** *Control Avanzado de Procesos.* 2002.

14. **Microsoft Corporation.** Microsoft Technet. [Online] 2014. [Cited: Octubre 14, 2015.] http://technet.microsoft.com/es-es/library/hh230886.aspx.

15. **Geisecke, Frederick.** *Dibujo y comunicación gráfica.* México D.F. : Person Education, 2006. ISBN 9702608112.

16. **Sánchez, Juan J.** *Manual de análisis estadístico de los datos.* s.l. : Alianza Editorial S.A., 2008. ISBN 9788420687162.

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

18. **Microsoft Corporation.** Microsoft. [Online] [Cited: Octubre 14, 2015.] https://msdn.microsoft.com/es-es/library/ms251748%28v=vs.90%29.aspx.

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

20. **Universidad Nacional Autónoma de México.** Gráfica de pastel. [Online] [Cited: Octubre 14, 2015.] http://www.cuautitlan.unam.mx.

21. **Sánchez, Tamara Rodríguez.** *Metodología de desarrollo para la actividad productiva de la UCI.* Universidad de las Ciencias Informáticas. La Habana : s.n.

22. **Patricia López, Francisco Ruiz.** Unican (Universidad de Cantabria). [Online] [Cited: Noviembre 11, 2015.] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/materiales-de-clase-1/is1-t02-trans.pdf.

23. **Ltd., Visual Paradigm International.** Visual Paradigm. [Online] [Cited: Noviembre 11, 2015.] http://www.visual-paradigm.com/.

24. **Guérin, Brice-Arnaud.** *Lenguaje C++.* s.l. : Ediciones ENI, 2005. ISBN 2746028492, 9782746028494.

25. **The Qt Company.** [Online] 2016. [Cited: Enero 2016, 18.] http://www.qt.io/qt-framework/.

26. —. Qt.io. [Online] 2016. [Cited: Mayo 03, 2016.] http://doc.qt.io/qt-4.8/qt4-8-intro.html.

27. —. Qt.io. [Online] 2015. [Cited: Noviembre 11, 2015.] https://wiki.qt.io/QtCreatorWhitepaper_Spanish.

28. **Sourceforge.net.** Qwt User's Guide 6.1.0. [Online] [Cited: Noviembre 11, 2015.] http://qwt.sourceforge.net/.

29. **The Qt Company.** Qt.io. [Online] 2015. [Cited: Noviembre 11, 2015.] http://doc.qt.io/QtCharts/.

30. —. Qt.io. [Online] 2015. [Cited: Noviembre 11, 2015.] http://doc.qt.io/qt-5/qtwidgets-itemviews-chart-example.html.

31. **Sintegral Technologies.** [Online] 2011-2014. [Cited: Enero 18, 2016.] http://www.sintegrial.com.

32. **Larman, Craig.** Modelo de dominio. *UML y Patrones.* s.l. : Prentice Hall, 2003, Vol. 2.

33. **Sommerville, I.** *Ingeniería del Software.* s.l. : Pearson, 2005. Vol. 7.

34. **Cardozzo, Daniel Ramos.** *Desarrollo de Sotware.* s.l. : IT Campus Academy, 2014. p. 125. ISBN 150325948X, 9781503259485.

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

36. **Riesco, Daniel.** Diagrama de clases y de objetos. [Online] [Cited: Marzo 21, 2016.] http://sel.unsl.edu.ar/licenciatura/ingsoft2/UML-DiagramaClaseObjeto.pdf.

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

38. **The Qt Company.** Qt.io. [Online] 2016. [Cited: Marzo 11, 2016.] http://doc.qt.io/qt-4.8/model-view-programming.html.

39. **Teniente López, Ernest, et al.** *Diseño de sistemas de software en UML.* s.l. : Universidad Politécnica de Catalunya, 2004. p. 216. ISBN 8498800757, 9788498800753.

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

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

42. **Hernández, Leovigilda.** [Online] Junio 1, 2013. [Cited: Marzo 30, 2016.] http://ithleovi.blogspot.com/2013/06/unidad-5-modelo-deimplementacion-el.html.

43. **Scribd.** Scribd.com. [Online] 2008. [Cited: Marzo 30, 2016.] http://www.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue.

44. **Vernal, Francisco.** Utfsm.cl. [Online] [Cited: Abril 02, 2016.] http://www.inf.utfsm.cl/~visconti/xp/Guia_Estandares_Codificacion_2.doc.

45. **Salas, Ing. Daniel Cafferata.** CalidadySoftware.com. [Online] http://www.calidadysoftware.com/testing/casos_de_prueba.php.

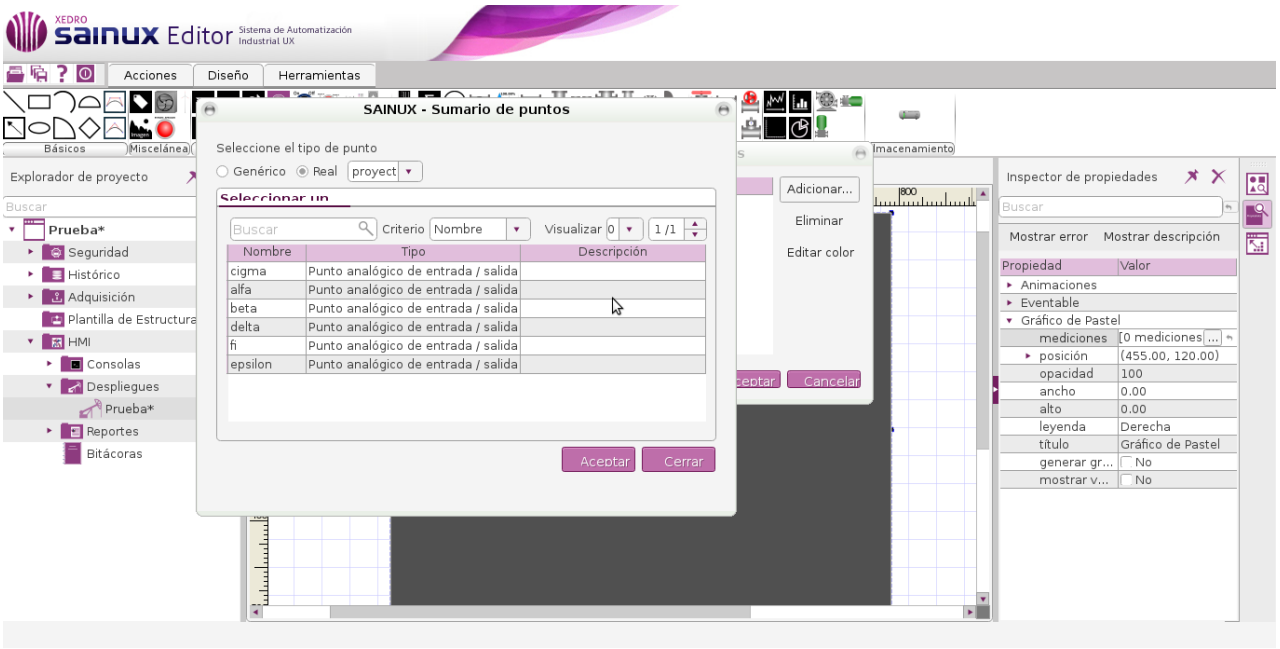
46. **pruebasdesoftware.cpm.** Pruebas de Software. [Online] http://pruebasdesoftware.cpm.

47. **Tanilkan, Sinan.** [Online] 2011. http://labs.qt.nokia.com/2011/12/15/qt-4-8-0-released/.

ANEXOS

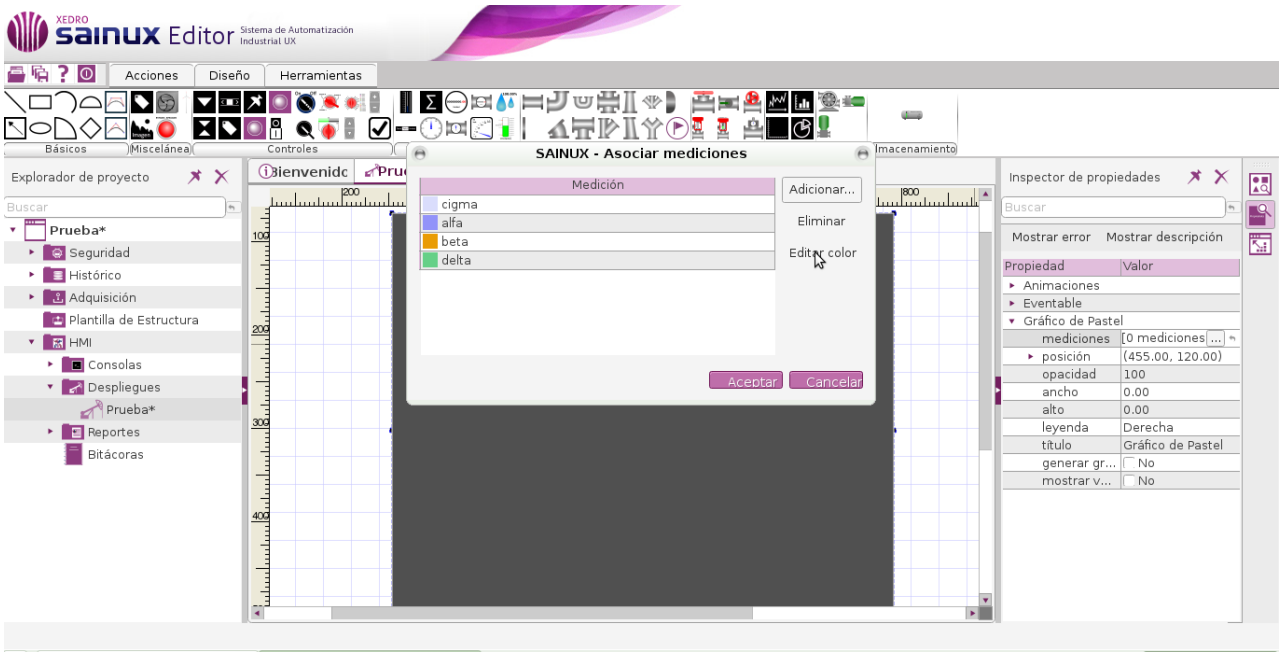
Anexo 1 HU 2 Adicionar sección

Tabla 7 HU 2 Adicionar sección

Historia de Usuario	
Número: HU2	**Nombre del requisito:** Adicionar sección
Programador: Rolando Futiel Gutiérrez	**Iteración asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 2
Riesgo en desarrollo: ❖ Problemas eléctricos. ❖ Problemas técnicos.	**Tiempo real:** 2 semanas.
Descripción: El operador será capaz de agregar una medición al gráfico ya existente en el despliegue; cada medición se representa a través de una sección. Para adicionar una sección se accede a la propiedad Gráfico de pastel 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 sección que le corresponde	
Observaciones:	
Prototipo de interfaz: 	

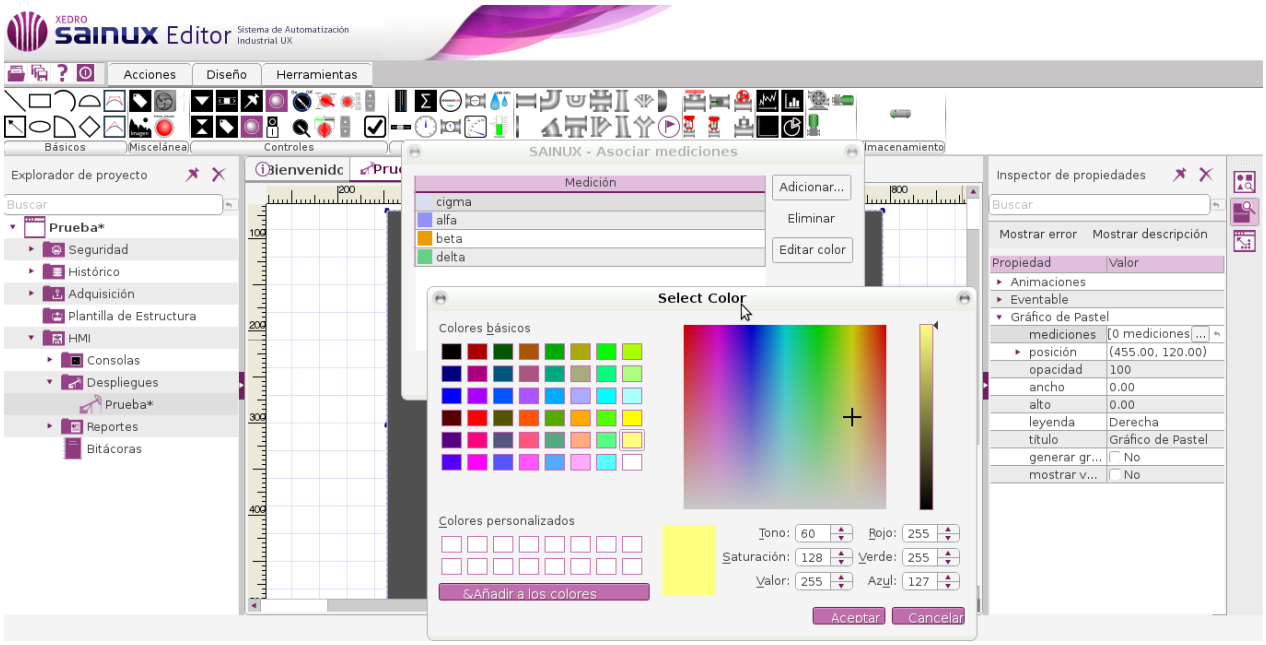
Anexo 2 HU3 Eliminar sección

Tabla 8 HU3 Eliminar sección

Historia de Usuario	
Número: HU3	**Nombre del requisito:** Eliminar sección
Programador: Rolando Futiel Gutiérrez	**Iteración asignada:** 2
Prioridad: Media	**Tiempo estimado:** 1
Riesgo en desarrollo: ❖ Problemas eléctricos. ❖ Problemas técnicos.	**Tiempo real:** 1 semanas.
Descripción: El operador será capaz de eliminar una medición existente del gráfico en el despliegue. Para eliminar una sección, se accede a la propiedad Gráfico de pastel 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: 	

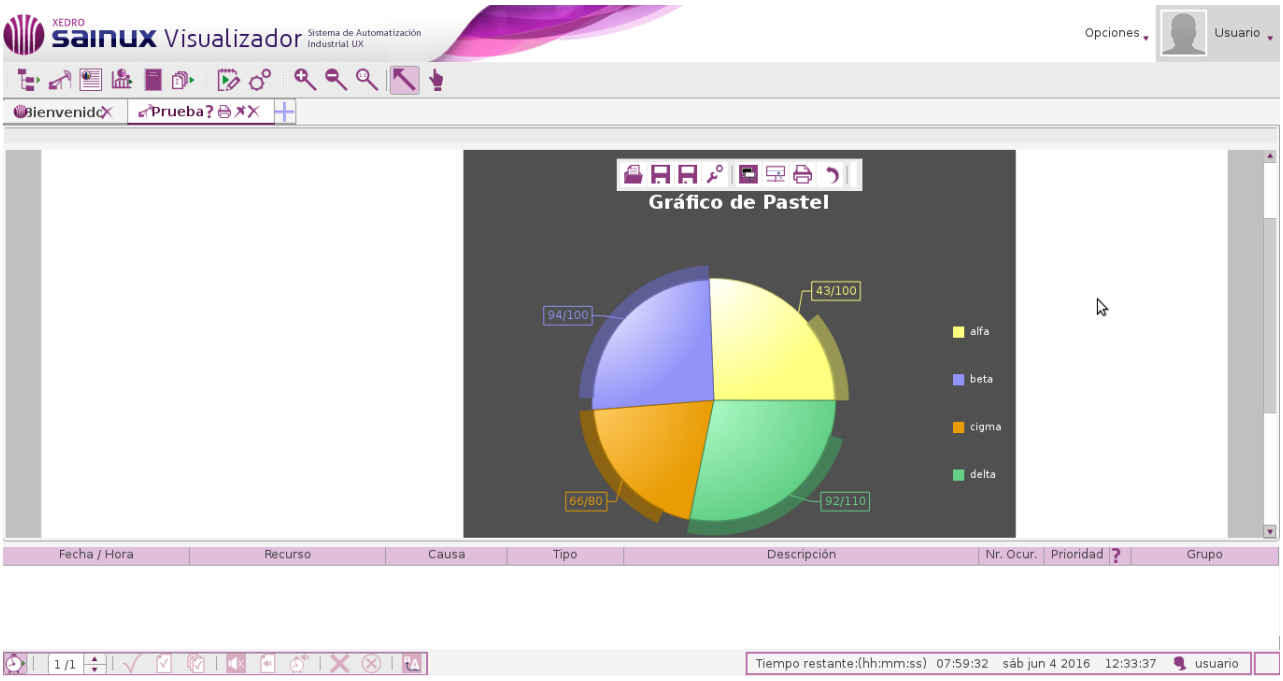
Anexo 3 HU4 Modificar color de la sección

Tabla 9 HU4 Modificar color de la sección

Historia de Usuario	
Número: HU4	**Nombre del requisito:** Modificar color de la sección
Programador: Rolando Futiel Gutiérrez	**Iteración asignada:** 3
Prioridad: Media	**Tiempo estimado:** 2
Riesgo en desarrollo: ❖ Problemas eléctricos. ❖ Problemas técnicos.	**Tiempo real:** 2 semanas.
Descripción: El operador será capaz de modificar el color de una sección representada en el componente. Para modificar una sección se accede a la propiedad Gráfico de pastel 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: 	

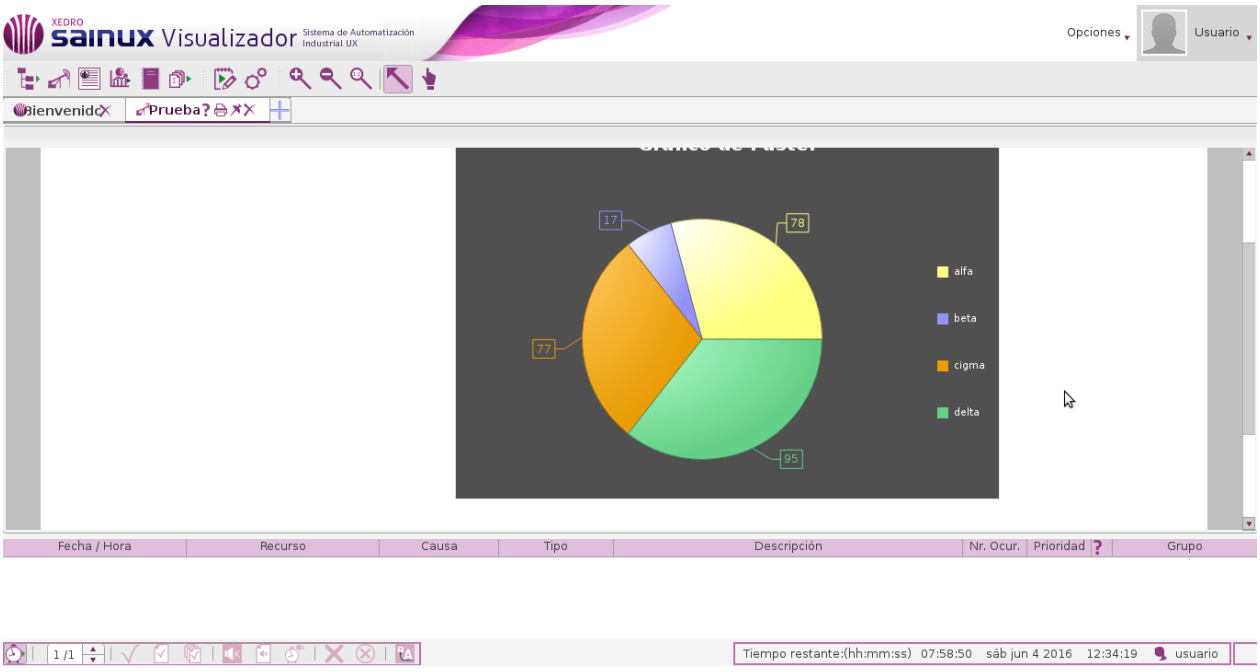
Anexo 4 HU5 Adicionar subsección

Tabla 10 HU5 Adicionar subsección

Historia de Usuario	
Número: HU5	**Nombre del requisito:** Adicionar subsección
Programador: Rolando Futiel Gutiérrez	**Iteración asignada:** 4
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: ❖ Problemas eléctricos. ❖ Problemas técnicos.	**Tiempo real:** 2 semanas.
Descripción: El operador será capaz de adicionar una subsección con un valor real asociado a una sección con un valor fijo de una medición del sistema. Para adicionar una subsección, se accede a la propiedad Gráfico de pastel del inspector de propiedades y marcar la opción Mostrar valores en tiempo real, esto mostrará en el entorno de visualización, las subsecciones asociadas a cada sección con valor fijo ya previamente agregada en el sistema.	
Observaciones:	
Prototipo de interfaz: 	

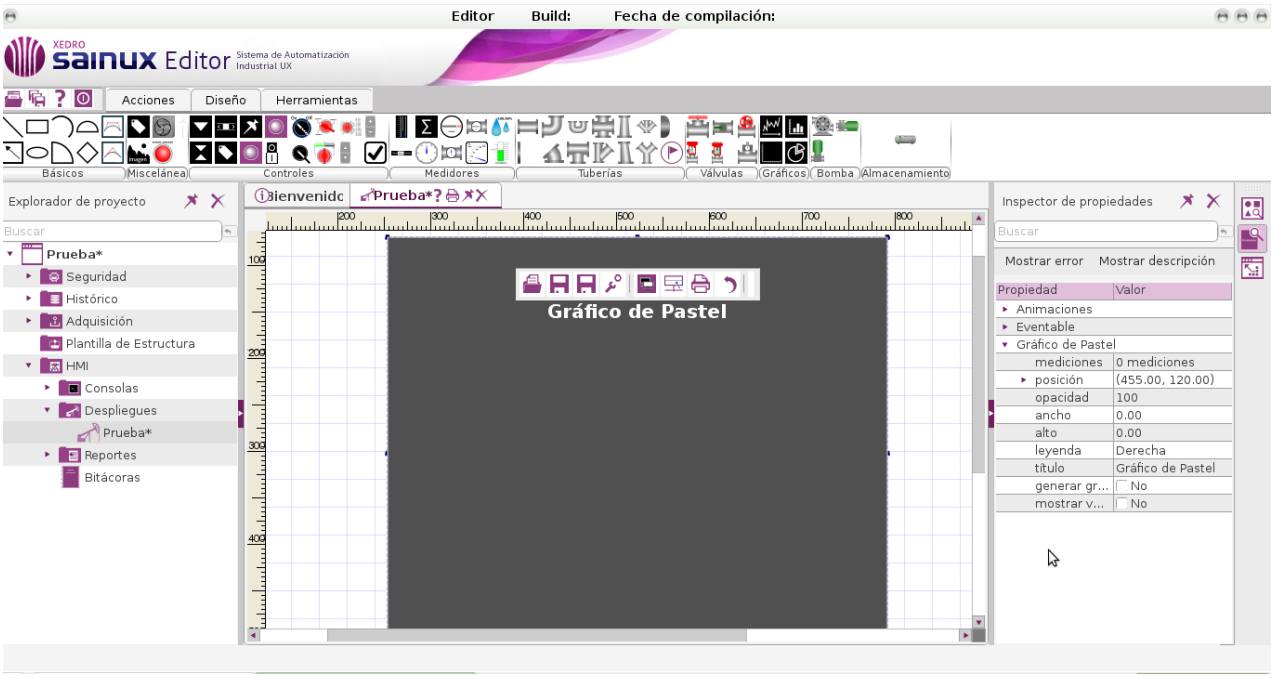
Anexo 5 HU6 Generar gráfico de pastel real

Tabla 11 HU6 Generar gráfico de pastel real

Historia de Usuario	
Número: HU6	**Nombre del requisito:** Generar gráfico de pastel real
Programador: Rolando Futiel Gutiérrez	**Iteración asignada:** 5
Prioridad: Alta	**Tiempo estimado:** 3
Riesgo en desarrollo: ❖ Problemas eléctricos. ❖ Problemas técnicos.	**Tiempo real:** 2 semanas.
Descripción: El operador será capaz de generar un gráfico de pastel a partir de los valores de todas las subsecciones con valores reales de una medición del sistema. Para generar el gráfico, se accede a la propiedad Gráfico de pastel del inspector de propiedades, luego marcar la opción Generar gráfico de tiempo real; lo que mostrará en el entorno de visualización, solo el gráfico con las mediciones en tiempo real previamente agregadas en el sistema.	
Observaciones:	
Prototipo de interfaz: 	

Anexo 6 HU7 Mostrar propiedades

Tabla 12 HU7 Mostrar propiedades

Historia de Usuario	
Número: HU7	**Nombre del requisito:** Mostrar propiedades
Programador: Rolando Futiel Gutiérrez	**Iteración asignada:** 6
Prioridad: Media	**Tiempo estimado:** 3
Riesgo en desarrollo: ❖ Problemas eléctricos. ❖ Problemas técnicos.	**Tiempo real:** 2 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: 	

Anexo 7 HU8 Configurar propiedades generales

Tabla 13 HU8 Configurar propiedades generales

Historia de Usuario	
Número: HU8	**Nombre del requisito:** Configurar propiedades generales

Programador: Rolando Futiel Gutiérrez	**Iteración asignada:** 7
Prioridad: Media	**Tiempo estimado:** 2
Riesgo en desarrollo: ❖ Problemas eléctricos. ❖ 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: ancho, alto, posición, título, entre otras.	
Observaciones:	
Prototipo de interfaz:	

Anexo 8 Diagrama de secuencia Eliminar sección

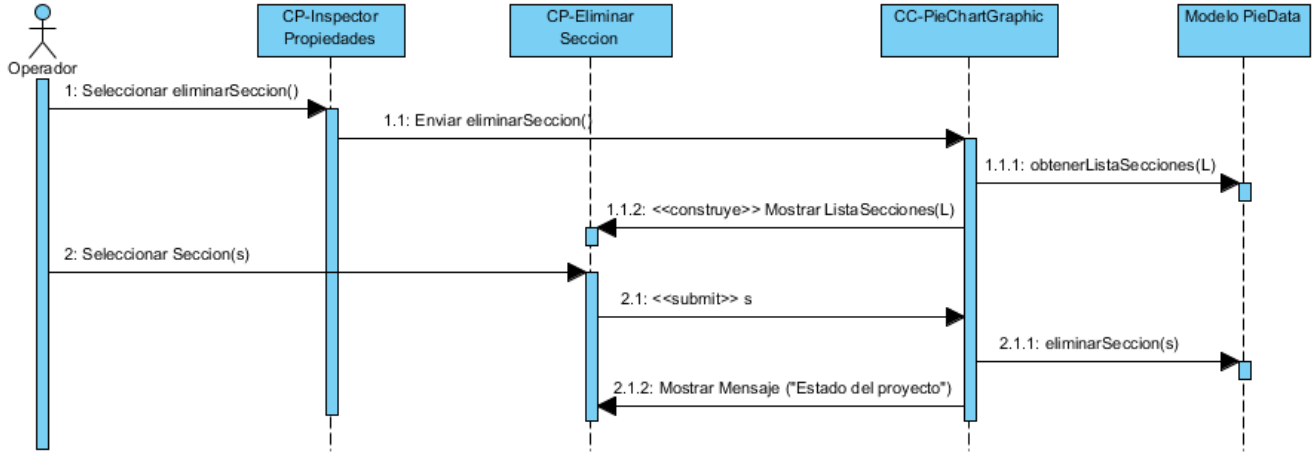


Ilustración 17 Diagrama de secuencia Eliminar sección

Anexo 9 Diagrama de secuencia Modificar color de la sección

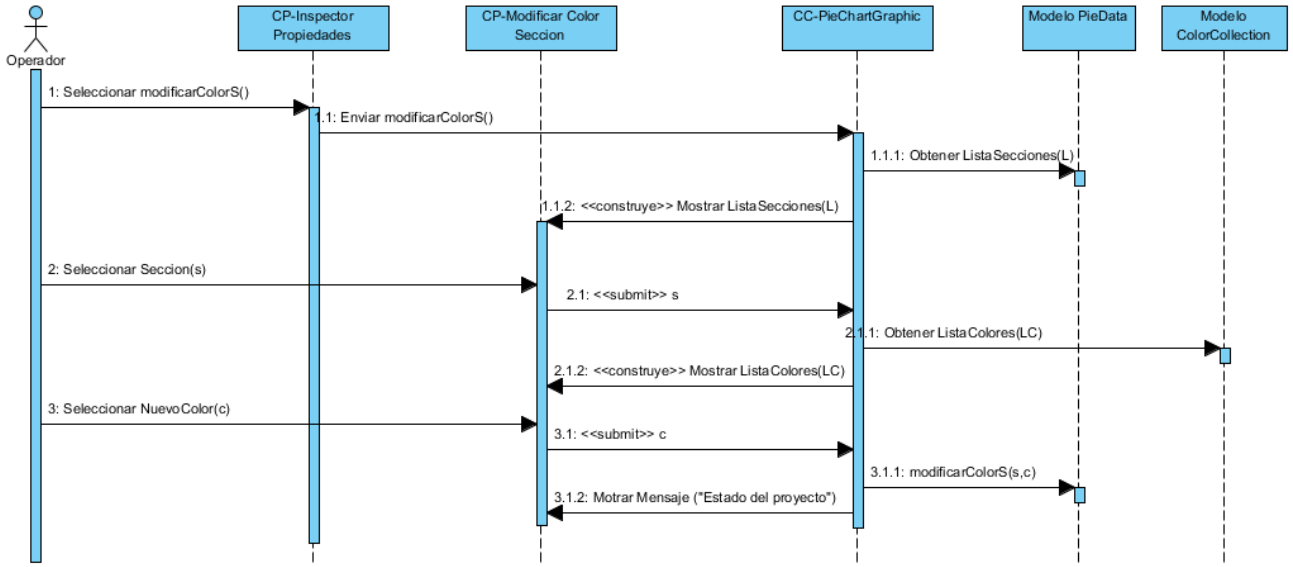


Ilustración 18 Diagrama de secuencia Modificar color de la sección

Anexo 10 CP Aceptación 2 Adicionar sección

Tabla 14 CP Aceptación 2 Adicionar sección

Casos de prueba de Aceptación	
Número: 2	**Historia de Usuario**: 2
Nombre: Adicionar sección.	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue.	
Entradas/Pasos de ejecución: ❖ El usuario accede a la propiedad Gráfico de pastel 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 sección con un color otorgado automáticamente.	
Resultado esperado: Visualizar la medición en forma de sección.	
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.	

Anexo 11 CP Aceptación 3 Eliminar sección

Tabla 15 CP Aceptación 3 Eliminar sección

Casos de prueba de Aceptación	
Número: 3	**Historia de Usuario**: 3
Nombre: Eliminar sección.	
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 Gráfico de pastel 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 sección correspondiente a la medición eliminada.	
Evaluación de la prueba: Satisfactorio.	

Anexo 12 CP Aceptación 4 Modificar color de la sección

Tabla 16 CP Aceptación 4 Modificar color de la sección

Casos de prueba de Aceptación	
Número: 4	**Historia de Usuario**: 4
Nombre: Modificar color de la sección.	
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 al Gráfico de pastel 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 sección.	
Resultado esperado: Se modifica el color de la sección.	
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.	

Anexo 13 CP Aceptación 5 Adicionar subsección

Tabla 17 CP Aceptación 5 Adicionar subsección

Casos de prueba de Aceptación	
Número: 5	**Historia de Usuario**: 5
Nombre: Adicionar subsección.	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue con al menos una medición asociada.	
Entradas/Pasos de ejecución: ❖ El usuario accede a la propiedad Gráfico de pastel dentro del Inspector de Propiedades. ❖ El usuario marca la opción Mostrar valores en tiempo real. ❖ El sistema visualiza las subsecciones asociadas a cada sección con valor fijo ya previamente agregada.	
Resultado esperado: Visualizar la medición en forma de subsección.	
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.	

Anexo 14 CP Aceptación 6 Generar gráfico de pastel real

Tabla 18 CP Aceptación 6 Generar gráfico de pastel real

Casos de prueba de Aceptación	
Número: 6	**Historia de Usuario**: 6
Nombre: Generar Gráfico de Pastel Real.	
Condiciones de ejecución: Debe estar visualizado el gráfico dentro del despliegue con al menos una medición y subsección asociadas.	
Entradas/Pasos de ejecución: ❖ El usuario accede a la propiedad Gráfico de pastel dentro del Inspector de Propiedades, y marca la opción Generar gráfico de tiempo real. ❖ El sistema visualiza solo el gráfico con las mediciones en tiempo real previamente agregadas.	
Resultado esperado: Visualizar solo el gráfico de pastel real.	
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.	

Anexo 15 CP Aceptación 7 Mostrar propiedades

Tabla 19 CP Aceptación 7 Mostrar propiedades

Casos de prueba de Aceptación	
Número: 7	**Historia de Usuario**: 7
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.	

Anexo 16 CP Aceptación 8 Configurar propiedades generales

Tabla 20 CP Aceptación 8 Configurar propiedades generales

Casos de prueba de Aceptación	
Número: 8	**Historia de Usuario**: 8
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 ancho y alto 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.	