FACULTAD 5

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

DESARROLLO DE COMPONENTES GRÁFICOS DE ALMACENAMIENTO EN EL SCADA SAINUX

Autor: Luis Orlando Batista Cejas

Tutores: Ing. Henry Marcelo Cabrera Robles

MSc. Manuel Villanueva Betancourt

Ing. Andy Hernández Paez

Co-Tutor: Ing. Luis Andrés Valido Fajardo

La Habana, Cuba

2015-2016

DECLARACIÓN DE AUTORÍA

Por este medio declaro ser autor de este trabajo y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ___ días del mes de _____ del año _____.

_____

Firma del autor: Luis Orlando Batista Cejas.

_____

Firma del tutor: Ing. Henry Marcelo Cabrera Robles.

_____

Firma del tutor: MSc. Manuel Villanueva Betancourt.

_____

Firma del tutor: Ing. Andy Hernández Paez.

_____

Firma del co-tutor: Ing. Luis Andrés Valido Fajardo.

DATOS DE CONTACTO

Tutor: Ing. Henry Marcelo Cabrera Robles.
Edad: 25
Ciudadanía: cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informáticas
Síntesis del Tutor: Graduado de Ingeniería en Ciencias Informáticas en julio del 2013, en la Universidad de Ciencias Informáticas. Posee 3 años de experiencia en el centro de informática industrial CEDIN, el cual ejerce el cargo de jefe de proyecto del SCADA SAINUX.
E-mail: hmcabrera@uci.cu

Tutor: Ing. Andy Hernández Paez.
Edad: 27.
Ciudadanía: cubana.
Institución: Universidad de las Ciencias Informáticas (UCI).
Títulos: Ingeniero en Ciencias Informáticas.
Síntesis del Tutor: Graduado de Ingeniería en Ciencias Informáticas en junio del 2012, en la Universidad de Ciencias Informáticas. Profesor Instructor de la disciplina Ingeniería y Gestión de Software. Actualmente profesor del centro Vertex Entornos Interactivos 3D. Posee 5 años de experiencia como Analista de sistema en proyectos informáticos de Informática Industrial y Realidad Virtual. Se desempeña como Asesor de Calidad de software del Centro Vertex y dirige el grupo de Ingeniería y Calidad de Software de dicho centro. Especialista en el marco de trabajo de la Ingeniería de Requisitos y Arquitectura de la Información. Presenta varias publicaciones y participación en eventos.
E-mail: andyhp@uci.cu

DATOS DE CONTACTO

Tutor: MSc. Manuel Villanueva Betancourt

Edad: 64.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Licenciado en Educación.

Síntesis del Tutor: Licenciado en Educación en la especialidad Física en 1980, en la Universidad de Ciencias Pedagógicas de La Habana. Profesor Auxiliar de la disciplina Formación Pedagógica y Metodología de la Investigación. Posee 45 años de experiencia como docente, asesor metodológico e investigador. Tiene un estudio de postgrado, dos diplomados y una maestría. Presenta varias publicaciones y participación en eventos.

AGRADECIMIENTOS

Son muchas las personas a las cuales tengo que agradecer y me siento feliz de que así sea, de que existan tantas personas importantes en mi vida.

A mi familia principalmente, que con su experiencia y conocimiento en mi crianza son los que me han nutrido de valores como guía para enfrentar el largo camino de la vida y superar cada posible obstáculo. Ellos son la razón de quien soy, y no solo le dedico este título sino todos los logros que pueda alcanzar.

A las personas que compartieron estos últimos cinco años conmigo, dígase compañeros de aula o apartamento. A todos les tengo mucha estima.

A ese pequeño círculo cercano con el que casi siempre ando, que solo porque no llevemos la misma sangre no significa que no los considere más que amigos.

A mis tutores que son los que me han apoyado y ayudado de manera directa con el presente trabajo.

A mi facultad, que me ha dado ese orgullo de decir "soy de la 5", en donde he vivido un ambiente de armonía y he conocido a verdaderos amigos.

RESUMEN

La informatización de la sociedad es una prioridad para Cuba, demostrando la voluntad política del país por acercar cada vez más las nuevas tecnologías a la población, expresado en los Lineamientos de la Política Económica y Social del Partido y la Revolución, que rigen las transformaciones en curso, y parten de que no es posible una sociedad próspera y sostenible sin subordinar a tales objetivos las herramientas que garanticen el acceso al conocimiento, la eficiencia, la productividad y la excelencia.

La informatización ha alcanzado un elevado desarrollo en diversas esferas a nivel mundial, entre ellas: la industria con la automatización de sus procesos mejorando la calidad, el rendimiento y la eficiencia. La industria cubana del software presenta un papel protagónico en este proceso de informatización, destacándose la Universidad de las Ciencias Informáticas con el centro de informática industrial (CEDIN). En dicho centro se tiene como principal línea de desarrollo a los sistemas de supervisión, control y adquisición de datos (SCADA), como el SCADA SAINUX.

La presente investigación surge a partir de la necesidad de agregar nuevos componentes gráficos al sistema SCADA SAINUX, brindándole los recursos necesarios para la representación gráfica del proceso de almacenamiento, para que disponga de los mecanismos gráficos para la supervisión y el control del sistema, y así minimizar el trabajo manual con las limitaciones en tiempo y recursos que implica en la industria desplegada.

Para cumplir con el objetivo se realizó un estudio de los principales componentes gráficos de almacenamiento empleado en las industrias. Se investigó las principales tecnologías y herramientas existentes en el mundo y se hace una propuesta fundamentada de cuales utilizar. Se presenta además la descripción de los requerimientos del sistema y se realiza la implementación de los mismos. Posteriormente las diferentes pruebas realizadas permitieron la aceptación del producto.

Palabras clave: SCADA, SACADA SAINUX, gráficos, componentes, HMI, sistema.

CONTENIDO

ÍNDICE

INTRODUCCIÓN

La informatización de la sociedad cubana es una voluntad política del gobierno que está expresada en los Lineamientos de la Política Económica y Social, aprobados en el Sexto Congreso del Partido Comunista de Cuba (lineamiento 131) (1). En varios sectores de la industria y la economía cubana se necesita un sistema capaz de informatizar la supervisión y control de sus procesos debido a la gran cantidad de operaciones que deben ser controladas a tiempo completo. Con el desarrollo de los avances tecnológicos y de la informática se ha hecho posible desarrollar software capaz de automatizar estos procesos, mejorando la eficacia del proceso de monitoreo y control, proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas. La informática industrial es la rama de esta ciencia encargada del tratamiento automático de la información proveniente de los procesos industriales, utilizando para ello computadoras.

SAINUX (Sistema de Automatización Industrial basado en GNU/LiNUX), es un sistema de supervisión, control y adquisición de datos (SCADA) distribuido, orientado a componentes y en proceso de desarrollo por especialistas del Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas (UCI).

Los primeros SCADA eran simplemente sistemas de telemetría que proporcionaban reportes periódicos de las condiciones de campo, vigilando las señales que representaban medidas o condiciones de estado en ubicaciones de campo remotas. Estos sistemas ofrecían capacidades muy simples de monitoreo y control, sin proveer funciones de aplicación alguna. La visión del operador en el proceso estaba basada en los contadores y las lámparas detrás de paneles llenos de indicadores. Mientras la tecnología se desarrollaba, los ordenadores asumieron el papel de manejar la recolección de datos, disponiendo comandos de control y una nueva función de presentación de la información sobre una pantalla. Los ordenadores agregaron la capacidad de programar el sistema para realizar funciones de control más complejas. Las tendencias (gráficos) de valores analógicos en un cierto plazo son muy comunes.

Recoger los datos y resumirlos en informes para los operadores y la gerencia son características normales de un sistema SCADA, representando en un ordenador los procesos que ocurren.

El módulo HMI (Interfaz Hombre Maquina) en los SCADA es el encargado de representar en un ordenador los procesos que ocurren en el campo, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación. Este módulo es el que permite al operador estar en contacto directo con el sistema, realizar la supervisión y el control del proceso en general. Los sistemas HMI se pueden pensar como una “ventana” de un proceso, donde las señales son conducidas por medio de dispositivos como tarjetas de entrada/salida en la computadora. Esta ventana puede estar en dispositivos especiales como paneles de operador o en una computadora, también conocido como software HMI o de monitoreo y control de supervisión.

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

Tanto para el EC como para el EV el HMI provee una biblioteca de componentes gráficos (CG) que permiten recrear de una manera lo más real posible los procesos de campo. Estos gráficos pueden ser simples figuras geométricas como: línea, rectángulo, elipse, texto, polígono, polilínea, curvas; pero también muy complejas como los componentes de hardware utilizados en la industria que se desean supervisar como son: válvulas, generadores, motores de combustión, entre otros.

Los gráficos sinópticos más usados en la representación del proceso donde se supervisa y controla el almacenamiento, son los tanques, los cuales no se encuentran en la biblioteca de componentes gráficos del HMI SAINUX.

Por estos motivos surge la necesidad de desarrollar nuevas paletas de componentes gráficos para los entornos de visualización y configuración, los cuales no cuentan con los componentes gráficos necesarios que permitan configurar los despliegues; siendo necesaria la rapidez del desarrollo de los mismos.

Por tanto, **la situación problemática** es la siguiente:

La carencia de componentes gráficos, de tipo tanques sobre la biblioteca de componentes gráficos del HMI SAINUX, ocasiona varias desventajas, como son:

- Ausencia de recursos para la representación gráfica del proceso de almacenamiento.
- No se dispone de mecanismos gráficos para la supervisión y el control del sistema.
- Elevado trabajo manual, con las limitaciones en tiempo y recursos que ello implica.

Dada la situación problemática planteada se formula el siguiente **problema de investigación**: ¿Cómo visualizar y configurar los componentes gráficos de almacenamiento que permita el proceso de supervisión y control automatizado del sistema?

Por tanto, se plantea como **objeto de estudio**: Proceso de visualización y configuración de componentes gráficos para procesos industriales.

Para darle solución al problema de investigación planteado se define como **objetivo general**: Desarrollar componentes gráficos de almacenamiento para el módulo de configuración y visualización del SCADA SAINUX, que permita la supervisión y control automatizada del sistema.

Todo lo cual delimita como **campo de acción**: Proceso de visualización de componentes gráficos de almacenamiento para procesos industriales en el módulo de configuración y visualización del SCADA SAINUX.

Tareas de investigación a llevar cabo:

1) Revisión de la bibliografía referente al módulo de visualización de los SCADA, para la adquisición de información sobre el desarrollo de componentes.
2) Identificación de los principales elementos presentes en estructuras de almacenamientos utilizadas en los procesos industriales.
3) Identificación de las herramientas y tecnologías utilizadas en el proceso de desarrollo de los componentes gráficos.
4) Análisis de las herramientas y tecnologías a utilizar enfocadas a aplicaciones de sistemas SCADA en uso de seleccionar las más adecuadas en nuestro software.
5) Definición de los componentes gráficos a desarrollar y sus características.
6) Implementación de los componentes gráficos.
7) Integración con la herramienta de configuración, para probar las funcionalidades de los componentes.
8) Realización de pruebas para validar el cumplimiento de los requerimientos.
9) Valoración de los resultados obtenidos, para saber hasta qué punto se alcanzó el objetivo de este trabajo.

Métodos Científicos:

- Métodos Teóricos:

✓ Análisis histórico-lógico: Se utiliza para analizar la evolución histórica de soluciones similares en los Sistemas SCADA, así como los componentes gráficos de almacenamiento empleados en el módulo HMI. También es utilizado para evidenciar la evolución de las principales herramientas y técnicas del desarrollo del software.

✓ Analítico-Sintético: para el análisis de la documentación existente relacionada con el tema, extrayendo los elementos más importantes y necesarios para dar solución

al problema existente, de manera que permita sintetizar todo lo obtenido relacionado con el módulo HMI y sus componentes gráficos de almacenamiento.

✓ Modelación: Se emplea para representar gráficamente la solución que se propone.

➢ Métodos Empíricos:

✓ Consulta Bibliográfica: Empleada para consultar las fuentes de información relacionados con la elaboración del marco teórico y los componentes gráficos visualizados en el módulo HMI de los sistemas SCADAs.

✓ Observación: Se puso en práctica para conocer el funcionamiento existente en los despliegues del SCADA SAINUX mediante el comportamiento de los dispositivos de campo y sus parámetros de configuración, para la toma de decisiones de los operadores.

Se tiene la siguiente **idea a defender**: Si se aplica el desarrollo de los componentes gráficos de almacenamiento se podría visualizar y configurar los componentes gráficos permitiendo la supervisión y control automatizado del sistema en los procesos de almacenamiento.

Posibles resultados:

- Elaboración de los Componentes gráficos para la representación del proceso de almacenamiento.
- Supervisión y Control automatizado del proceso de almacenamiento.

El presente documento está estructurado en tres capítulos:

Capítulo 1: Marco teórico de la investigación relacionado con el desarrollo de componentes gráficos para el sistema SCADA SAINUX. Además, se abordará que es un SCADA, sus módulos, los componentes de almacenamientos, las tecnologías y herramientas actuales enmarcadas en el proceso, así como la metodología de software seleccionada para el cumplimiento del proyecto.

Capítulo 2: Describe la solución propuesta, los requerimientos funcionales, requisitos no funcionales y los elementos de documentación más significativos tras la aplicación de la metodología de desarrollo seleccionada.

Capítulo 3: En este capítulo se describe la fase de implementación y prueba del sistema, según la metodología propuesta.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

En el presente capítulo se precisan un conjunto de elementos para conformar el marco teórico relacionado con los aspectos definidos en el objeto de estudio. Se presentan los conceptos y definiciones que apoyan la comprensión del trabajo, se define el conjunto de tecnologías y herramientas para el desarrollo del sistema y la metodología de software seleccionada.

1.2 SISTEMAS SCADA

1.2.1 DEFINICIÓN

SCADA, (Supervisión, Control y Adquisición de Datos) es un software para ordenadores que permite controlar y supervisar procesos industriales a distancia. Facilita retroalimentación en tiempo real con los dispositivos de campo (sensores y actuadores) controlando el proceso automáticamente. Provee de toda la información que se genera en el proceso productivo (supervisión, control calidad, control de producción, almacenamiento de datos) y permite su gestión e intervención (2).

Se trata de una aplicación de software diseñada para funcionar sobre ordenadores en controles de producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, etc.). Se refiere a un sistema central que monitoriza una gran aérea (kilómetros / millas). Todo esto se ejecuta normalmente en tiempo real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar los procesos. La instalación de un sistema SCADA necesita un hardware de señal de entrada y salida, sensores y actuadores, controladores, HMI, redes, comunicaciones, base de datos entre otros (2).

Las principales funciones de un SCADA se pueden resumir en (2):

- **Adquisición de datos:** significa que el sistema tiene la capacidad de obtener información desde el sistema de control, por ejemplo, sobre valores de temperatura presión y a diferencia de la “supervisión” los datos son registrados o almacenados para su posterior explotación.
- **Supervisión**: significa poder observar o monitorear aquello que sucede en el proceso industrial, el equipo o la maquinaria, por ejemplo, conocer (generalmente de una forma gráfica) si un motor se encuentra encendido o apagado.
- **Control:** significa tener la posibilidad de ejecutar comandos; en otras palabras, poder enviar instrucciones hacia el sistema de control, por ejemplo, ordenar al Controlador Lógico Programable (PLC siglas en inglés) (3).

1.2.2 PRINCIPALES MÓDULOS DE UN SCADA

Los módulos o bloques de software que permiten las actividades de adquisición, supervisión y control son (4):

- **Configuración:** permite al usuario definir el entorno de trabajo de su SCADA, adaptándolo a la aplicación particular que se desea desarrollar.
- **Interfaz gráfico del operador:** proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos almacenados en el ordenador de proceso y generados desde el editor incorporado en el SCADA o importados desde otra aplicación durante la configuración del paquete.
- **Módulo de proceso:** ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas.
- **Gestión y archivo de datos:** se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

- **Comunicaciones:** se encarga de la transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y entre ésta y el resto de elementos informáticos de gestión.

1.2.3 MÓDULO HMI DE UN SCADA

La Interfaz Hombre-Máquina o HMI (Human Machine Interface) es el sistema que presenta los datos a un operador (humano) y a través de la cual éste controla el proceso (3). Son interfaces gráficas que muestran información del proceso en tiempo real, utilizando diagramas esquemáticos, algunos contornos y hasta animaciones en pantalla o paneles.

Al HMI son conducidas las señales de procesos por medios de dispositivos como tarjetas de entrada/salida en la computadora PLC's (Controladores Lógicos Programables), RTU (Unidades Remotas) o DRIVERS's (Variadores de velocidad de motores). Todos los dispositivos deben tener una comunicación que entienda el HMI.

1.2.3.1 ENTORNO DE CONFIGURACIÓN

El editor gráfico o ambiente de configuración es el que permite a un operador definir el ambiente de trabajo en un SCADA, adaptándolo a sus necesidades de desarrollo. El usuario define dentro del módulo las pantallas gráficas o de texto que va a utilizar. En este entorno las pantallas o procesos definen sus relaciones entre ellas, determinando el orden de aparición, los drivers, los comandos, las alarmas y las variables a visualizar que después se procesarán y controlarán en forma de listas o tablas para una mejor gestión. El entorno tiene como finalidad administrar toda información que sirva como base de configuración al resto de los módulos del SCADA, garantizando la persistencia de dicha información (5).

Los archivos de configuración estarán bajo el formato XML, separando radicalmente la información o contenido de su presentación o formato.

Algunas de las características que debe tener un editor gráfico, son las siguientes:

- Asociación de señales a sus gráficos.
- Utilizar colores que señalen situaciones con claridad: preferentemente verde para indicar normalidad, amarillos para señalar atención por parte del operador y rojos para situaciones o estados de peligro.
- Añadir texto y audio a las alarmas e intermitencias.
- Poder hacer zooms de parte del sinóptico del proceso para ampliar la información según sub-sinópticos, ya sea a voluntad del usuario o por indicación del propio SCADA.

1.2.3.2 ENTORNO DE VISUALIZACIÓN

Es el encargado de visualizar las animaciones y los objetos definidos en el entorno de configuración, se encarga de mostrar en tiempo real lo que está ocurriendo en el campo, utilizando para ello interfaces gráficas que modelan o simulan los procesos de forma simple, recibe los valores de las variables y envía los comandos a las estaciones remotas. Este entorno visualiza los datos históricos en forma de gráficos de tendencia y hace manejo de alarmas y eventos etc. Es el empleado de supervisar de manera directa el proceso puesto que interactúa con los operadores (6).

1.2.3.3 COMPONENTES GRÁFICOS

Los componentes gráficos son un conjunto de elementos que presenta como objetivo la comunicación visual a partir de aplicaciones que tienen su origen en el diseño bidimensional. Un buen empleo de los gráficos determina el comportamiento y el seguimiento del objeto en profundidad.

A través de los gráficos se incorporan los elementos comunicacionales necesarios para generar una determinada imagen del espacio y del contenido de la muestra, además de transmitir información textual.

Los gráficos se incorporan en todo tipo de soportes y materiales facilitando la comunicación global y la comprensión.

1.3 COMPONENTES DE ALMACENAMIENTO

1.3.1 DEFINICIÓN

Se denomina almacenamiento al proceso y la consecuencia de almacenar, se utiliza para hacer referencia a un acto mediante el cual se guarda algún objeto o elemento específico con el fin de poder luego recurrir a él en el caso que sea necesario (7).

1.3.2 COMPONENTES

Una gran parte de las sustancias utilizadas en la industria para la obtención de productos que permitan el desarrollo de las actividades humanas, se encuentran en estado líquido. El principal problema que se plantea con respecto a los líquidos es el almacenamiento como paso previo o posterior a un proceso de producción. Normalmente el almacenamiento de estos líquidos se realiza en los denominados tanques de almacenamiento.

Tipos de tanques de almacenaje:

- Ferro-tanque: Tanque de almacenamiento que es transportado en rieles por locomotoras.
- Carro-tanque: Tanque de almacenamiento que es transportado por un vehículo motorizado en carreteras o vías de tránsito.
- Tanques de almacenamiento de hidrocarburos: Tanque que se suele alojar en las refinerías para depositar los productos o subproductos del proceso de refinado de petróleo.
- Tanques de Almacenamiento de Proceso: Tanques utilizados en las industrias de procesos alimenticias, automotrices, electrónicas, para almacenar

sustancias liquidas o en forma granuladas según sea el caso, para su uso en el proceso de la manufactura de un producto terminado.

Las principales tecnologías del inventariado o componentes en tanques son:

- Sistemas tipo servo: Minimizan los componentes electrónicos por medio de micro-controladores. Los medidores de nivel tipo servo avanzados (ATG) tienen menos de 5 componentes móviles.
- Sistemas tipo radar: Fácil instalación y alta precisión. Mide elementos tanto en estado líquido como solido mediante señales de radar transmitidas.
- Medición de temperaturas: Son utilizados en casos de estratificación de la temperatura (MTT, MRT).
- Sistema tipo columna hidrostática: Dispone de transmisores de presión provistos de microprocesador (HTG).
- Sistema híbrido: Uso de equipos avanzados, servo o radar, para la medición del nivel. Mide todos los parámetros del tanque con un único sistema (HIMS).

1.4 SELECCIÓN DE HERRAMIENTAS Y TECNOLOGÍAS

En este epígrafe se expondrán las tecnologías y herramientas de desarrollo que pueden contribuirá la solución de la aplicación. Se hace una breve descripción del ambiente de trabajo como es el sistema operativo y las herramientas y tecnologías seleccionadas. La selección de varias herramientas y tecnologías está fundamentada por su utilización en el desarrollo del SCADA SAINUX.

1.4.1 GRÁFICOS VECTORIALES ESCALABLES (SVG)

Gráficos Vectoriales Escalables. Es una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de SMIL), en formato de lenguaje de marcas extensible (XML siglas en inglés) (8). Corresponde con las siglas Scalable Vector Graphics, que viene a significar Gráficos Vectoriales Escalables (9).

La palabra vector tiene distintos significados dependiendo el contexto en el que se maneje: matemáticas, geometría, física, en el caso del diseño y la ilustración por computador define la forma en que un gráfico o imagen puede ser producido y representado. Los gráficos vectoriales son en su formato completamente distintos a los gráficos de mapas de bits o también llamados matriciales, los cuales están constituidos por pixeles.

Los vectores tienen la ventaja sobre los pixeles de ser escalables, es decir, se puede aumentar su tamaño conservando su calidad original. Lo que no ocurre con los gráficos a base de pixeles que presentan degradación de la calidad al aumentar el tamaño. Otra diferencia importante entre estos dos tipos de imágenes digitales es el peso, los vectoriales por ser solamente parámetros matemáticos, suelen ser mucho menos pesados que una imagen rasterizada o de pixeles. Los gráficos vectoriales pueden ser transformados (estirar, rotar, mover, distorsionar) de una manera más sencilla y con menos requerimientos de memoria en el equipo (9).

Sin embargo, todos los gráficos vectoriales tienen que ser transformados a pixeles cuando se presentan en una pantalla o cuando se requiere su impresión.

1.4.2 INKSCAPE

Editor de gráficos vectoriales de código abierto, con capacidades similares a Illustrator, Freehand, CorelDraw o Xara X, usando el estándar de la W3C: el formato de archivo Scalable Vector Graphics (SVG) (10).

Es un sistema de desarrollo abierto disponibles en las distribuciones GNU/Linux para los usuarios de la comunidad de software libre. Ha visto algunas mejoras en los últimos lanzamientos que han sido capaces de preparar mejor a los artistas con las herramientas necesarias para crear trabajo vectorial foto-realistas. El objetivo principal de Inkscape es crear una herramienta de dibujo potente y cómoda, logrado un progreso significativo en la utilidad desde su surgimiento.

1.4.3 SISTEMA OPERATIVO

Dentro del mundo del software libre se encuentra la tecnología conocida como Linux. Linux es un Sistema Operativo (SO) tipo Unix diseñado para aprovechar al máximo las capacidades de las computadoras PC basadas en el microprocesador i386 y posteriores. Es un SO con capacidades de multiprocesamiento, multitarea y multiusuario (11).

El software libre es un modelo económico de producción de recursos software que se caracteriza por copiar, usar, estudiar y modificar el software.

En los sistemas operativos libres el software se distribuye en forma de paquetes modulares que suelen incluir completa información sobre interdependencias lo que facilita la gestión de la instalación del software, su actualización y su integridad. Con frecuencia estos sistemas son mucho más avanzados que las alternativas ofrecidas por el software propietario. Este ha sido desarrollado por miles de usuarios de computadores a través del mundo y su licencia tiene por objeto asegurar que Linux siga siendo gratuito y a la vez estándar (12).

La distribución seleccionada para el cumplimiento de este proyecto es Debian en su versión 7.0 Wheezy. Debian es especialmente popular entre los usuarios avanzados debido a su excelencia técnica y compromiso con las necesidades y expectativas de la comunidad Linux. Posee miles de paquetes pre-compilados estables y su sistema de gestor de paquetes se distingue de otras distribuciones GNU/Linux, otorgando un control total sobre los paquetes instalados (13). Además, se encuentra muy relacionado con el SCADA SAINUX, dado que este producto está siendo desarrollado sobre esta plataforma dependiendo de algunas de sus bibliotecas.

1.4.4 LENGUAJE DE PROGRAMACIÓN C++

El lenguaje de programación seleccionado es C++, por ser el lenguaje en el cual se está desarrollando el proyecto SCADA SAINUX y así lograr un alto grado de

compatibilidad. Es considerado por muchos como el lenguaje de programación más potente debido que recoge tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Permite la programación de alto y bajo nivel, logrando gran eficiencia en los tiempos de ejecución y bajo consumo de memoria en los programas desarrollados (14). El lenguaje de programación C++ presenta gran cantidad de bibliotecas implementadas que permiten la reutilización del código. La utilización de plantillas también permite la reutilización del código y la programación genérica.

Además se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones inline, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería (15).

1.4.5 MARCO DE TRABAJO

Qt es una biblioteca para desarrollar aplicaciones utilizando el lenguaje C++ de forma nativa, las cuales pueden ser con o sin interfaz gráfica (16). Se encuentra disponible para sistemas tipo Unix. El principal motivo por el que Qt se ha hecho tan popularidad es su propiedad de ser multiplataforma. Este proporciona una serie de herramientas y documentación imprescindibles para el desarrollador como (15):

- Qt Creator: Entorno de desarrollo integrado multiplataforma diseñado a la medida de los desarrolladores de Qt.
- Qt Designer: Diseñador de interfaces gráficas de usuario multiplataforma. Permite la creación rápida de formularios con la tecnología de los layouts.
- Qt Linguist: Herramienta para crear aplicaciones multilingües.
- Qt Assistant: Generador de documentación.

1.4.6 ENTORNO INTEGRADO DE DESARROLLO

Qt Creator es un Entorno Integrado de Desarrollo (IDE) creado por Trolltech, multiplataforma, diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil (17). Posee diversas características dentro de las que se destacan su avanzado editor de código C++, herramientas para proyectos y administración, depurador visual, resaltado y auto-completado de código, soporta los lenguajes: C#/.NET Languages (Mono), Python: PyQt y PySide, Ada, Pascal, Perl, PHP y Ruby y mucho más (18).

Qt Creator se centra en proporcionar características que ayudan a los nuevos usuarios de Qt a aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt.

Dadas las características descritas y algunas de sus ventajas decidimos elegir a Qt Creator para el desarrollo del proyecto, además de estar familiarizado con la herramienta.

1.4.7 HERRAMIENTA CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas Aplicaciones informáticas destinadas a aumentar la productividad en el Desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero (19). Su objetivo es acelerar el proceso para el que han sido diseñadas, automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas. Estas brindan toda una gama de componentes que incluyen todas o la mayoría de los requisitos necesarios para el desarrollo de los sistemas, han sido creados con una gran exactitud en torno a las necesidades de los desarrolladores de sistemas para la automatización de procesos incluyendo el análisis, diseño e implantación.

Se selecciona para el modelado del sistema el Visual Paradigm 8.0, dado que es una herramienta CASE para UML (Unified Modeling Language, Lenguaje de Modelado

Unificado), de fácil uso y completa, con soporte multiplataforma, y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Además, se puede encontrar en varios idiomas, es fácil de instalar, actualizar y posee compatibilidad entre ediciones.

1.5 METODOLOGÍA DE DESARROLLO

Desarrollar un buen software depende de un sinnúmero de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo, en un determinado proyecto es trascendental para el éxito del producto. El papel predominante de las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas.

No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, tiempo, recursos, etc.) exigiéndose así que el proceso sea configurable. Las metodologías de desarrollo se pueden dividir en dos grupos de acuerdo con sus características y los objetivos que persiguen: ágiles y robustas (20).

El éxito del producto depende en gran parte de la metodología escogida por el equipo, ya sea tradicional o ágil, donde los equipos maximicen su potencial, aumenten la calidad del producto con los recursos y tiempos establecidos.

Las metodologías ágiles se caracterizan por darle un mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Con cortos documentos centrados en lo esencial. (Amaro Calderón 2007).

Las metodologías tradicionales en cambio, se centran especialmente en el control del proceso, mediante una exhaustiva documentación; definiendo roles, actividades, artefactos, herramientas y notaciones para el modelado y una documentación detallada. Estas metodologías son muy efectivas y necesarias en proyectos grandes (21).

Se presenta una tabla que ilustra las diferencias entre las metodologías tradicionales y ágiles.

Tabla 1: "Diferencias entre metodologías Tradicionales y Ágiles"

Metodologías Tradicionales	Metodologías Ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.	Basadas en heurísticas provenientes de prácticas de producción de código.
Resistencia a los cambios.	Preparadas para cambios durante el proyecto.
Impuestas externamente.	Impuestas internamente (por el equipo).
Proceso más controlado, con numerosas políticas/normas.	Proceso menos controlado, con pocos principios.
Existe un contrato prefijado.	No existe contrato tradicional (bastante flexible)
El cliente se comunica con el equipo de desarrollo mediante reuniones.	El cliente es parte del equipo de desarrollo.
Grupos grandes y distribuidos.	Grupos pequeños (menos de 10 integrantes) trabajando en el mismo sitio.
Más artefactos.	Pocos artefactos.

Más roles.	Pocos roles.
La arquitectura de software es esencial y se expresa mediante modelos.	Menos énfasis en la arquitectura del software.

Dado el escaso tiempo para la realización del proyecto, recurso, documentación necesaria y personal, se opta por escoger una metodología ágil. Dentro de las metodologías agiles existentes encontramos algunas como:

- XP.
- OPEN UP.
- RUP.
- BPM.
- DAC.
- KIMBALL.
- SXP.
- SCRUM.
- NOVA OPEN UP.
- AUP.

Existen otras metodologías llamadas híbridas, tal es el caso de la AUP (Agile Unified Process, Proceso Unificado Ágil). Las metodologías llamadas híbridas caen dentro de alguna de las dos clasificaciones mencionadas: ágiles o tradicionales.

AUP es una versión simplificada de RUP (Rational Unified Process), que describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP.

AUP aplica técnicas ágiles incluyendo (22):

- Desarrollo Dirigido por Pruebas.
- Modelado ágil.
- Gestión de Cambios ágil.

- Refactorización de Base de Datos para mejorar la productividad.

Cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos, debido a esto la Universidad de las Ciencias Informáticas (UCI) propone una adaptación de la metodología ágil AUP a utilizar en sus proyectos productivos. En él se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3 (22). AUP se propone para la actividad productiva de la UCI logrando hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos.

En la UCI se decide hacer cuatro variaciones (escenarios) de AUP-UCI para adaptarla a las características de cada proyecto. Teniendo en cuenta que el sistema a implementar no modela negocio, es escogido el escenario No. 4, donde solo se puede modelar el sistema con historias de usuario. Esta es aplicada a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una historia de usuario no debe poseer demasiada información.

Presenta las fases del ciclo de desarrollo siguientes (22):

Tabla 2: "Variación AUP-UCI".

Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto

Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto
Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.6 CONCLUSIONES PARCIALES

Con el desarrollo del marco teórico se organizó y encaminó el trabajo hacia el desarrollo del objetivo general. Se profundizó en el estudio de algunos conceptos fundamentales para el desarrollo del sistema, identificando el estándar SVG como posible línea base para el diseño de los componentes gráficos de almacenamiento. Se seleccionaron las herramientas y tecnologías libres a utilizar y la metodología de desarrollo de software propuesta para la solución del problema.

CAPÍTULO 2: CARACTERÍSTICAS Y DISEÑO DE LA SOLUCIÓN

2.1 INTRODUCCIÓN

En el siguiente capítulo se aborda la descripción de los pasos a seguir durante el análisis y el diseño de la solución. Contiene la especificación de los requisitos funcionales y los no funcionales que debe cumplir el sistema, con sus pertinentes descripciones**.** A partir de los requerimientos se determinan las historias de usuarios que regirán el desarrollo de la solución propuesta, además se muestran los diagramas que describen el funcionamiento del sistema.

2.2 PROPUESTA DE LA SOLUCIÓN

El presente trabajo está orientado al desarrollo de componentes gráficos de almacenamiento en el sistema SCADA SAINUX, el cual contará con una paleta de componentes gráficos para su funcionamiento y estructuración. Los componentes de almacenamientos deberán ser capaces de controlar parámetros de su fuente de almacenamiento como los de tipo líquidos, gaseosos y en algunos casos sólidos, soportar transformaciones en su dimensión sin que afecte la imagen, así como también deben ser editados por el usuario final para su funcionamiento en la industria.

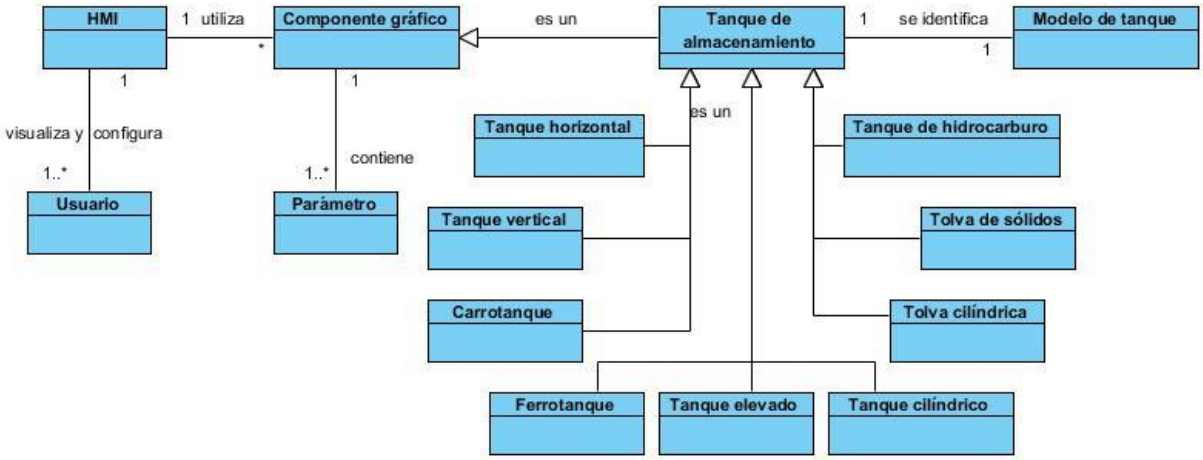


Ilustración 1: "Modelo conceptual".

La anterior imagen es utilizada como fuente de comprensión a la solución propuesta, para organizar y representar el conocimiento de manera que se facilite el entendimiento del tema, donde los usuarios visualizan y configura los componentes gráficos mediante el HMI del sistema; los tanques de almacenamiento son componentes gráficos que tienen parámetros asociados. Estos tanques de almacenamiento presentan un modelo heredado según el tipo de tanque correspondiente como: tanque horizontal, tanque vertical, tanque de hidrocarburo, tanque elevado, tolva cilíndrica entre otras.

2.3 ESPECIFICACIÓN DE REQUISITOS

La especificación de requisitos de software es una descripción completa del comportamiento del sistema que se va a desarrollar. También contiene requisitos no funcionales (o suplementarios). Los requisitos no funcionales son los requisitos que imponen restricciones al diseño o funcionamiento del sistema (23).

2.3.1 REQUISITOS FUNCIONALES

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer (24).

A continuación, se especifican los requisitos teniendo en cuenta las funcionalidades que el sistema debe proveer:

Tabla 3: "Requisitos Funcionales".

Número	Nombre	Descripción
1	Configurar unidad de ingeniería del componente.	Proporcionarle el tipo de unidad de ingeniería a medir del componente.
2	Mostrar u ocultar unidad de ingeniería del componente.	Mostrar la unidad de ingeniería en caso de que se desee u ocultarla del componente.
3	Configurar valor de unidad de ingeniería del componente.	Proporcionarle el valor de unidad de ingeniería a medir del componente.
4	Mostrar u ocultar valor de unidad de ingeniería del componente.	Mostrar el valor de la unidad de ingeniería en caso de que se desee u ocultarla del componente.
5	Configurar escala numérica de valores de unidad de ingeniería del componente.	Proporcionarle valores a la escala numérica de unidad de ingeniería del componente.
6	Mostrar u ocultar escala numérica de valores de unidad de ingeniería del componente.	Mostrar la escala numérica de valores de unidad de ingeniería en caso que se desee u ocultarla del componente.
7	Configurar color interior del componente.	Proporcionarle el color deseado por el usuario al interior del componente.

8	Configurar el color del nivel de medición del componente.	Proporcionarle el color deseado por el usuario del nivel de medición del componente.
9	Configurar la dimensión del ancho del componente.	Aumentar o reducir la anchura del componente al gusto del usuario.
10	Configurar la dimensión de altura del componente.	Aumentar o reducir la altura del componente al gusto del usuario.
11	Configurar la posición (X, Y) del componente.	Ubicar el componente en la posición (X,Y) deseada por el usuario.
12	Configurar la opacidad del componente.	Otorgarle mayor o menor opacidad al componente según lo deseado por el usuario.
13	Configurar el color del componente.	Cambiar el color del componente a gusto del usuario.

2.3.2 REQUISITOS NO FUNCIONALES

Un requisito no funcional o atributo de calidad es un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales, son todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento (25).

Con el objetivo de estandarizar la redacción de los requisitos no funcionales (RNF) de Atributos de calidad se propone una nueva Taxonomía partiendo de la ISO 25010. Las partes que conforman la taxonomía pueden o No aplicar en dependencia del RNF que se esté evaluando.

Tabla 4: "Requisito no Funcional #1".

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-característica	Operabilidad.
Objetivo	Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
Origen	Proveedor de requisitos.
Artefacto	Pantallas pertenecientes a los componentes gráficos.
Entorno	El sistema desplegado.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a. Número de pasos a ejecutar para llegar a la visualización y configuración de componentes gráficos	
Interactuar con las funcionalidades de visualización y configuración.	1. Se seleccionan los componentes de la paleta correspondiente. 2. Se arrastran los componentes seleccionados y se lanzan sobre el área de configuración. 3. Se seleccionan las propiedades a configurar de los componentes con el click derecho. 4. A partir de las propiedades seleccionadas se procede a la configuración de estas.
Medida de respuesta	
Navegar por el entorno de configuración.	

Tabla 5: "Requisito no Funcional #2".

Atributo de Calidad	Usabilidad.
Sub-atributos/Sub-característica	Agradabilidad.
Objetivo	Capacidad del producto que permite al usuario interactuar con el sistema mediante interfaces agradables e intuitivas.
Origen	Proveedor de requisitos.
Artefacto	Pantallas pertenecientes a los componentes gráficos.

Entorno	El sistema desplegado.
Estímulo **1. a. Ambiente gráfico de interacción con los componentes desarrollados**	**Respuesta: Flujo de eventos (Escenarios)**
Interactuar con los componentes gráficos mediante interfaces de comunicación.	1. Selecciona pantallas gráficas que permiten visualizar y configurar los componentes desarrollados. 2. Muestra las interfaces gráficas uniformes incluyendo pantallas que incluyen modelos planos a partir de archivos .svg, así como también menús y opciones interactivas.
Medida de respuesta	
Navegar por el entorno de configuración.	

Tabla 6: "Requisito no Funcional #3".

Atributo de Calidad	Fiabilidad.
Sub-atributos/Sub-característica	Disponibilidad.
Objetivo	Capacidad del producto de estar operativo y accesible para su uso cuando se requiere.
Origen	Proveedor de requisitos.
Artefacto	El sistema de configuración y visualización de componentes.
Entorno	El sistema desplegado, funcionando en el tiempo deseado.
Estímulo **1. a. El sistema desplegado en los escenarios especificados por el usuario durante el tiempo deseado**	**Respuesta: Flujo de eventos (Escenarios)**
El sistema no deberá tener limitante de tiempo y de inactividad. El sistema no guarda el estado antes una falla de energía eléctrica. Si el computador es apagado durante el juego la aplicación se verá interrumpida. El sistema al cambiar la hora del computador cliente	N/A

no se afectará.	
Medida de respuesta	
Navegar por el entorno de configuración.	

Tabla 7: "Requisito no Funcional #4".

Atributo de Calidad	Mantenibilidad.
Sub-atributos/Sub-característica	Analizabilidad.
Objetivo	Facilidad con la que se puede evaluar el impacto de un determinando cambio sobre el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar las partes a modificar.
Origen	Arquitecto de software.
Artefacto	El código fuente.
Entorno	El ambiente de desarrollo del sistema de configuración y visualización de componentes.
Estímulo **1. a. Impacto de un determinado cambio sobre el resto del sistema de configuración y visualización de componentes.**	**Respuesta: Flujo de eventos (Escenarios)**
Se evalúa el impacto de un cambio en el sistema de configuración y visualización de componentes.	1. El arquitecto de software identifica los paquetes del diseño y clases implicadas en el cambio, así como las funciones que se reutilizarán o se crearán para introducir el cambio. 2. Se evalúa el impacto partiendo de los resultados que arrojó el análisis anterior con respecto a la arquitectura del sistema de configuración y visualización de componentes.
1. b. Diagnosticar las deficiencias o causas de fallos en el sistema de configuración y visualización de componentes.	**Respuesta: Flujo de eventos (Escenarios)**

Se diagnostican las deficiencias o causas de fallos en el sistema de configuración y visualización de componentes.	1. El arquitecto de software identifica las posibles deficiencias o causas de fallos que se pueden originar en el sistema de configuración y visualización de componentes. 2. Se diagnostican las deficiencias o causas de fallos partiendo de los resultados que arrojó el análisis anterior.
1. c. Identificar las partes a modificar.	**Respuesta: Flujo de eventos (Escenarios)**
Se identifican las partes a modificar en el sistema de configuración y visualización de componentes.	1. El arquitecto de software identifica los paquetes y clases implicados en el cambio, así como las funciones que se reutilizarán o se crearán para introducir el cambio.
Medida de respuesta	
Analizar el entorno de configuración.	

Tabla 8: "Requisito no Funcional #5".

Atributo de Calidad	Mantenibilidad.
Sub-atributos/Sub-característica	Modificabilidad.
Objetivo	Capacidad del producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño.
Origen	Arquitecto de software.
Artefacto	El código fuente.
Entorno	El ambiente de desarrollo del sistema de configuración y visualización de componentes.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a. Capacidad de modificación del sistema de configuración y visualización de componentes.	
La arquitectura del sistema de configuración y visualización de componentes está diseñada para brindar facilidades a la hora de introducir modificaciones en el sistema de configuración y	N/A

visualización de componentes. Esto permite que se puedan introducir cambios o modificaciones, que no afecten el correcto desempeño del resto de las funcionalidades de la solución.	
Medida de respuesta	
Introducir una modificación al sistema de configuración y visualización de componentes.	

Tabla 9: "Requisito no Funcional #6".

Atributo de Calidad	Adecuación funcional.
Sub-atributos/Sub-característica	Integridad funcional.
Objetivo	Grado en el que el conjunto de funciones cubre todas las tareas y objetivos del usuario especificados.
Origen	Proveedor de requisitos.
Artefacto	El sistema de configuración y visualización de componentes.
Entorno	El sistema desplegado.
Estímulo **1. a. Grado en que el sistema de configuración y visualización de componentes cubre todas las tareas y objetivos especificados.**	**Respuesta: Flujo de eventos (Escenarios)**
El desarrollo del sistema de configuración y visualización de componentes está guiado por las necesidades expresadas por parte de los proveedores de requisitos, dándole total cumplimiento a sus especificaciones declaradas e implícitas.	N/A
Medida de respuesta	
Navegar por el entorno de configuración.	

Tabla 10: "Requisito no Funcional #7".

Atributo de Calidad	Eficiencia en el rendimiento.
Sub-atributos/Sub-característica	Comportamiento en el tiempo.

Objetivo	Grado en que los tiempos de respuesta, procesamiento y las tasas de rendimiento de un producto o sistema, al realizar sus funciones cumplen con los requisitos.
Origen	Arquitecto de software.
Artefacto	El sistema de configuración y visualización de componentes.
Entorno	El sistema desplegado.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a. Tiempo de respuesta ante alguna acción realizada por el usuario	
Se realizan las operaciones de configuración y visualización de componentes.	1. El sistema de configuración y visualización de componentes muestra las pantallas gráficas de visualización y configuración de componentes en 5 milisegundos.
Medida de respuesta	
Navegar por el entorno de configuración.	

Tabla 11: "Requisito no Funcional #8".

Atributo de Calidad	Eficiencia en el rendimiento.
Sub-atributos/Sub-característica	Utilización de recursos.
Objetivo	Grado en el que las cantidades y tipos de recursos utilizados por un producto o sistema, al realizar sus funciones cumplen con los requisitos.
Origen	Arquitecto de software.
Artefacto	El sistema de configuración y visualización de componentes.
Entorno	El sistema desplegado.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. a. Características de Hardware	
Cliente de aplicación:	1. El sistema de visualización y configuración

• Microprocesador: Intel(R) Core(Tm) 2 Duo. • Memoria RAM: 2 Gigabytes. • Capacidad del Disco Duro de 60 Giga. • Acelerador gráfico de 256 Mb. • Aceleración gráfica: Los controladores de aceleración gráfica deberán estar instalados dependiendo del hardware gráfico. • Para la interacción con la aplicación el computador debe tener los periféricos mouse *(ratón)* y teclado.	de componentes funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. b. Características de Software	
• Sistema operativo GNU/Linux en su distribución Debian, específicamente la versión Wheezy.	1. El sistema de visualización y configuración de componentes funcionando correctamente.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1. c. Características de diseño e implementación.	
• Qt en su versión 4.8. • Para exportar archivos .svg planos se emplea Inkscape como editor de gráficos vectorial. • Visual Paradigm For UML en su versión 8.0.	1. El sistema de visualización y configuración de componentes en proceso de desarrollo.
Medida de respuesta	
Desarrollo y funcionamiento del sistema de visualización y configuración de componentes.	

2.4 HISTORIAS DE USUARIOS

Una historia de usuario es una representación de un requisito escrito en una o dos frases utilizando el lenguaje común del usuario. Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad

de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes (26).

2.4.1 DESCRIPCIÓN DE LAS HISTORIAS DE USUARIO (HU).

El equipo de desarrollo realizó un análisis de las prioridades y la dependencia de las HU para definir el costo en semanas para el desarrollo de cada una. Estas nuevas historias pueden describirse en cualquier momento.

Las HU son representadas por tablas que contienen los siguientes campos o secciones:

- Número: Número que representa el identificador de la historia de usuario.
- Nombre de Historia: Nombre que identifica la HU.
- Prioridad: El valor de este campo es Baja, Media o Alta, definido por el cliente dependiendo de la importancia y el orden para el proceso de desarrollo.
- Iteración Asignada: Número de la iteración en la cual se desarrollará la HU.
- Actor: Personajes o entidades que participarán en un caso de uso.
- Tiempo Estimado: Tiempo estimado en horas que se le asignará.
- Tiempo Real: Tiempo en horas dedicado al cumplimiento del requisito.
- Descripción: Breve descripción del proceso que define la historia.
- Observaciones: Algunas aclaraciones que es importante señalar acerca de la historia.
- Prototipo elemental de interfaz gráfica de usuario: Descrito el requisito mediante una breve imagen.

Tabla 12: "Historia de Usuario #1".

Historia de Usuario

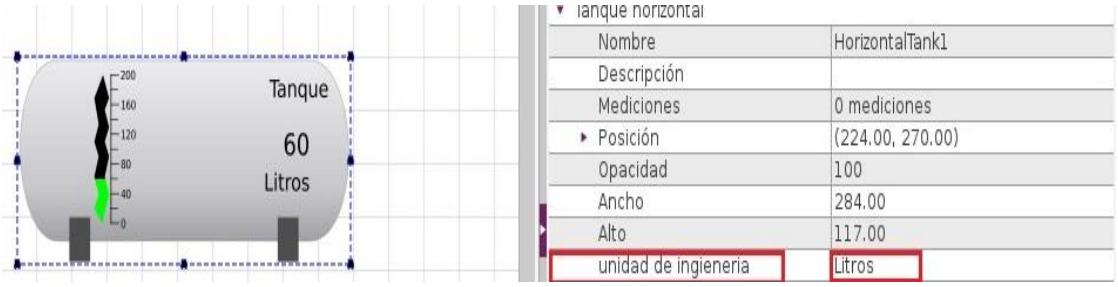
Número: 1	**Nombre del requisito:** Configurar unidad de ingeniería del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 42 (horas)
Descripción: Su objetivo es asignar la unidad de ingeniería del componente. El operador al dar clic derecho encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde el operador puede asignarle la unidad de ingeniería deseada al componente, haciendo clic izquierdo encima del campo que está al lado de Unidad de Ingeniería.	
Observaciones: No Aplica	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 13: "Historia de Usuario #2".

	Historia de Usuario
Número: 2	**Nombre del requisito:** Mostrar y ocultar unidad de

	ingeniería del componente.
Actor: Usuario	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Media	**Tiempo Real:** 42 (horas)
Descripción: Su objetivo es visualizar o no la unidad de ingeniería asociada al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde se encuentra un CheckBox que brinda la funcionalidad de marcarlo, dando clic izquierdo para mostrar la unidad de ingeniería del componente o desmarcarlo para no mostrarlo.	
Observaciones: No Aplica	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 14: "Historia de Usuario #3".

Historia de Usuario	
Número: 3	**Nombre del requisito:** Configurar valor de unidad de ingeniería del componente.

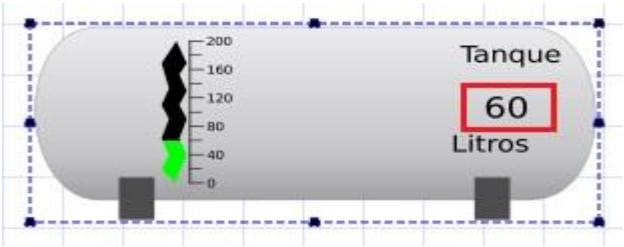
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 35 (horas)
Descripción: Su objetivo es asignarle un nuevo valor a la unidad de ingeniería del componente. El operador al seleccionar el componente y dar doble clic encima del valor de la unidad de ingeniería del componente, podrá cambiar el valor numérico correspondiente por otro deseado.	
Observaciones: No Aplica	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 15: "Historia de Usuario #4".

Historia de Usuario	
Número: 4	**Nombre del requisito:** Mostrar y ocultar valor de unidad de ingeniería del componente.
Actor: Usuario	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)

Nivel de Complejidad: Media	**Tiempo Real:** 35 (horas)
Descripción: Su objetivo es visualizar o no el valor de la unidad de ingeniería asociada al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde se encuentra un CheckBox que brinda la funcionalidad de marcarlo, dando clic izquierdo para mostrar el valor de la unidad de ingeniería del componente o desmarcarlo para no mostrarlo.	
Observaciones: No Aplica 	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 16: "Historia de Usuario #5".

Historia de Usuario	
Número: 5	**Nombre del requisito:** Configurar escala numérica de valores de unidad de ingeniería del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)

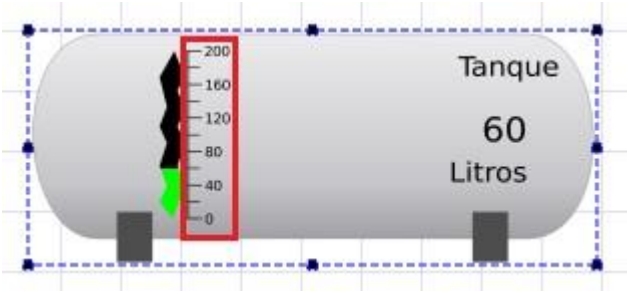
Nivel de Complejidad: Alta	**Tiempo Real:** 35 (horas)
Descripción: Su objetivo es asignarle nuevos valores a la escala numérica de valores unidad de ingeniería del componente. El operador al seleccionar el componente y dar doble clic encima de cada valor numérico de la escala, podrá cambiar el valor numérico correspondiente por otro deseado.	
Observaciones: No Aplica	
Prototipo elemental de interfaz gráfica de usuario: 	

Tabla 17: "Historia de Usuario #6".

Historia de Usuario	
Número: 6	**Nombre del requisito:** Mostrar y ocultar escala numérica de valores de unidad de ingeniería del componente.
Actor: Usuario	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Media	**Tiempo Real:** 29 (horas)
Descripción: Su objetivo es visualizar o no los valores de la escala numérica	

asociada al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde se encuentra un CheckBox que brinda la funcionalidad de marcarlo, dando clic izquierdo para mostrar la escala numérica de valores de la unidad de ingeniería del componente o desmarcarlo para no mostrarlo.

Observaciones: No Aplica

Prototipo elemental de interfaz gráfica de usuario:

Tabla 18: "Historia de Usuario #7".

Historia de Usuario	
Número: 7	**Nombre del requisito:** Configurar color interior del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 30 (horas)
Descripción: El operador al dar clic derecho encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un	

inspector de propiedades, en donde puede seleccionar el color interior que quiere que tome el componente, al dar clic izquierdo sobre la opción se despliega una venta con diferentes gamas de colores a escoger según el gusto del operador.
Observaciones: No Aplica
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 19: "Historia de Usuario #8".

Historia de Usuario	
Número: 8	**Nombre del requisito:** Configurar el color del nivel de medición del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 35 (horas)
Descripción: El operador al dar clic derecho encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde puede seleccionar el color del nivel de medición que quiere que tome el componente, al dar clic izquierdo sobre la	

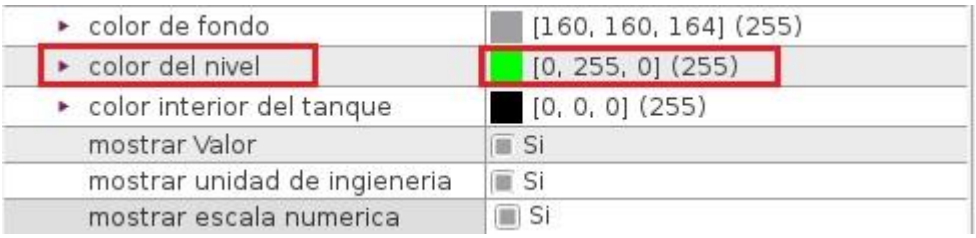
opción se despliega una venta con diferentes gamas de colores a escoger según el gusto del operador.
Observaciones: No Aplica
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 20: "Historia de Usuario #9".

Historia de Usuario	
Número: 9	**Nombre del requisito:** Configurar la dimensión del ancho del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 35 (horas)
Descripción: Su objetivo es asignar una dimensión al componente. El operador al dar clic derecho encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde el operador puede definirle el ancho que desea que tenga el componente. También el operador al marcar el componente con un clic, situar el puntero en el borde izquierdo o derecho del componente y presionar el clic izquierdo, puede	

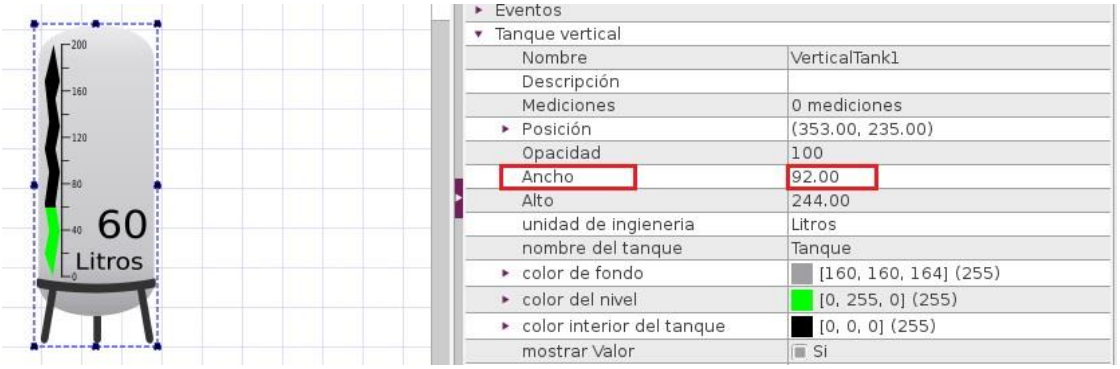
aumentar o disminuir el ancho del componente a su preferencia.
Observaciones: No Aplica
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 21: "Historia de Usuario #10".

Historia de Usuario	
Número: 10	**Nombre del requisito:** Configurar la dimensión de altura del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 35 (horas)
Descripción: Su objetivo es asignar una dimensión al componente. El operador al dar clic derecho encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde el operador puede definirle la altura que desea que tenga el componente. También el operador al marcar el componente con un clic, situar el puntero en el	

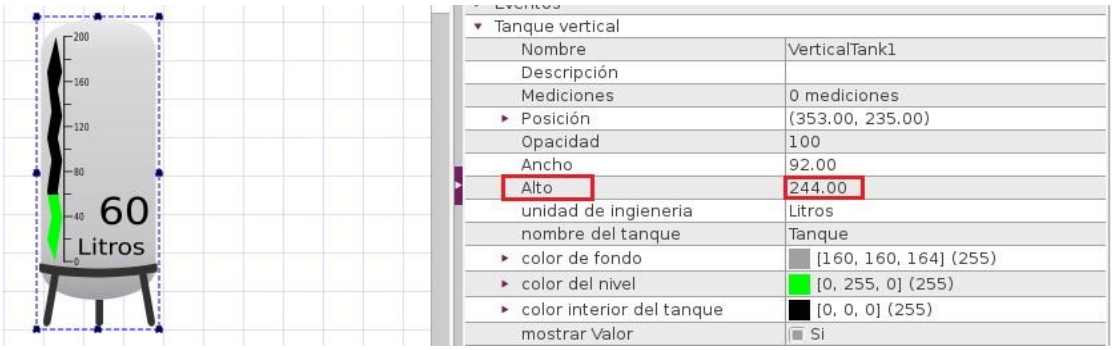
borde de arriba o debajo del componente y presionar el clic izquierdo, puede aumentar o disminuir la altura del componente a su preferencia.
Observaciones: No Aplica
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 22: "Historia de Usuario #11".

Historia de Usuario	
Número: 11	**Nombre del requisito:** Configurar la posición (X, Y) del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 35 (horas)
Descripción: Su objetivo es asignar una posición al componente. El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, donde puede configurar la posición en la que se traslada el objeto sobre el eje X y el	

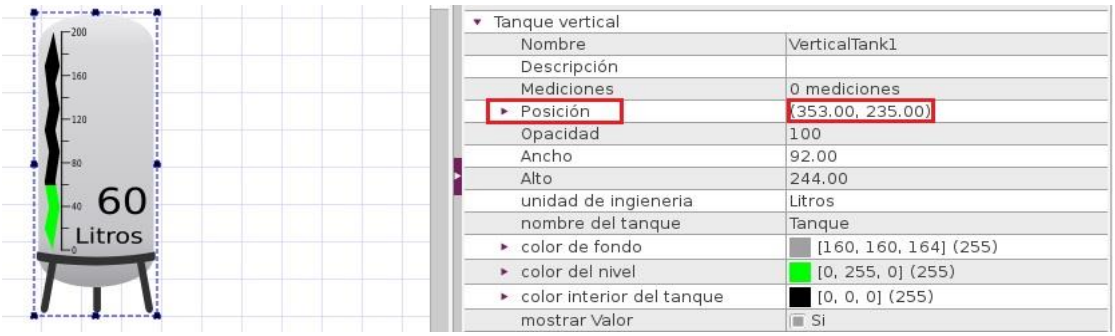
eje Y. También el operador puede seleccionar el componente dejando presionado el clic izquierdo sobre este y moverlo según su preferencia.
Observaciones: No Aplica
Prototipo elemental de interfaz gráfica de usuario: 

Tabla 23: "Historia de Usuario #12".

Historia de Usuario	
Número: 12	**Nombre del requisito:** Configurar la opacidad del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 42 (horas)
Descripción: El operador al dar clic izquierdo encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde el operador puede seleccionar la opacidad del componente que desee en un intervalo de 0 – 100 fijo.	

Observaciones: No Aplica

Prototipo elemental de interfaz gráfica de usuario:

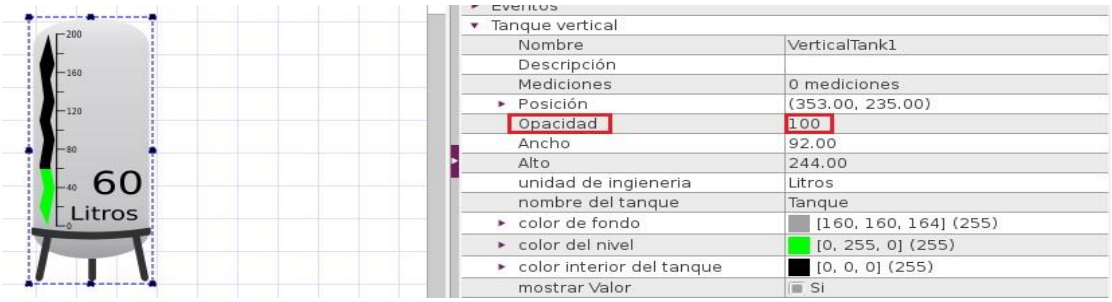
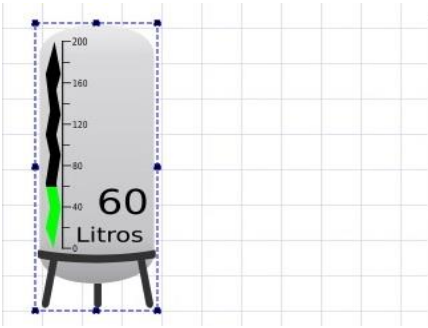


Tabla 24: "Historia de Usuario #13".

Historia de Usuario	
Número: 13	**Nombre del requisito:** Configurar el color del componente.
Actor: Usuario	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 35 (horas)
Nivel de Complejidad: Alta	**Tiempo Real:** 25 (horas)
Descripción: El operador al dar clic derecho encima del componente y seleccionar la opción propiedades, en la parte derecha del editor se muestra un inspector de propiedades, en donde puede seleccionar el color del componente deseado al dar clic izquierdo sobre la opción desplegándose una venta con diferentes gamas de colores a escoger según el gusto del operador.	
Observaciones: No Aplica	
Prototipo elemental de interfaz gráfica de usuario:	



Eventos	
Tanque vertical	
Nombre	VerticalTank1
Descripción	
Mediciones	0 mediciones
Posición	(353.00, 235.00)
Opacidad	100
Ancho	92.00
Alto	244.00
unidad de ingieneria	Litros
nombre del tanque	Tanque
color de fondo	[160, 160, 164] (255)
color del nivel	[0, 255, 0] (255)
color interior del tanque	[0, 0, 0] (255)
mostrar Valor	Si

2.4 PLAN DE ITERACIONES

Después de identificar y elaborar cada una de las HU, se confecciona el plan de iteraciones. Para cada iteración son seleccionas las HU de acuerdo al nivel de complejidad, donde en una primera iteración se escogieron las historias de usuario de prioridad alta y en una segunda iteración a las de prioridad media.

Se definieron dos iteraciones para el desarrollo de las HU de la aplicación. A continuación, se explican y justifican cada una de las iteraciones:

- Iteración 1: Para esta iteración se desarrollan las HU de configuración de los componentes correspondientes, garantizado la representación gráfica de los parámetros de cada componente.
- Iteración 2: En esta iteración se desarrollarán las HU asociadas a mostrar u ocultar los parámetros deseados de configuración. Para la implementación de los requisitos de esta iteración es tomado como apoyo la iteración anterior.

Para aproximar el tiempo de ejecución de las iteraciones se tomó como medida de tiempo las horas, constando que una semana consta de 5 días (lunes, martes, miércoles, jueves, viernes), de los cuales se trabaja aproximadamente 7 horas diarias. A continuación, se muestra en la tabla 6 el plan de iteraciones con el tiempo estimado.

Tabla 25: "Plan de Iteraciones".

Iteración	Historias de usuario	Puntos de estimación (horas)
1	Configurar unidad de ingeniería del componente.	350
	Configurar color interior del componente.	
	Configurar valor de unidad de ingeniería del componente.	
	Configurar el color del nivel de medición del componente.	
	Configurar escala numérica de valores de unidad de ingeniería del componente.	
	Configurar la dimensión del ancho del componente.	
	Configurar la dimensión de altura del componente.	
	Configurar la opacidad del componente.	
	Configurar el color del componente.	
	Configurar la posición (X, Y) del componente.	
2	Mostrar y ocultar escala numérica de valores de unidad de ingeniería del componente.	105

	Mostrar y ocultar unidad de ingeniería del componente.	
	Mostrar y ocultar valor de unidad de ingeniería del componente.	

Después de realizar un análisis se pudo estimar que el costo estimado para el desarrollo es de 455 horas, equivalente a 13 semanas.

2.5 MODELO DE DISEÑO

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 agregación, ya que una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones. Un diagrama de clases está compuesto por los siguientes elementos: clase: atributos, métodos y visibilidad. relaciones: herencia, composición, agregación, asociación y uso (27).

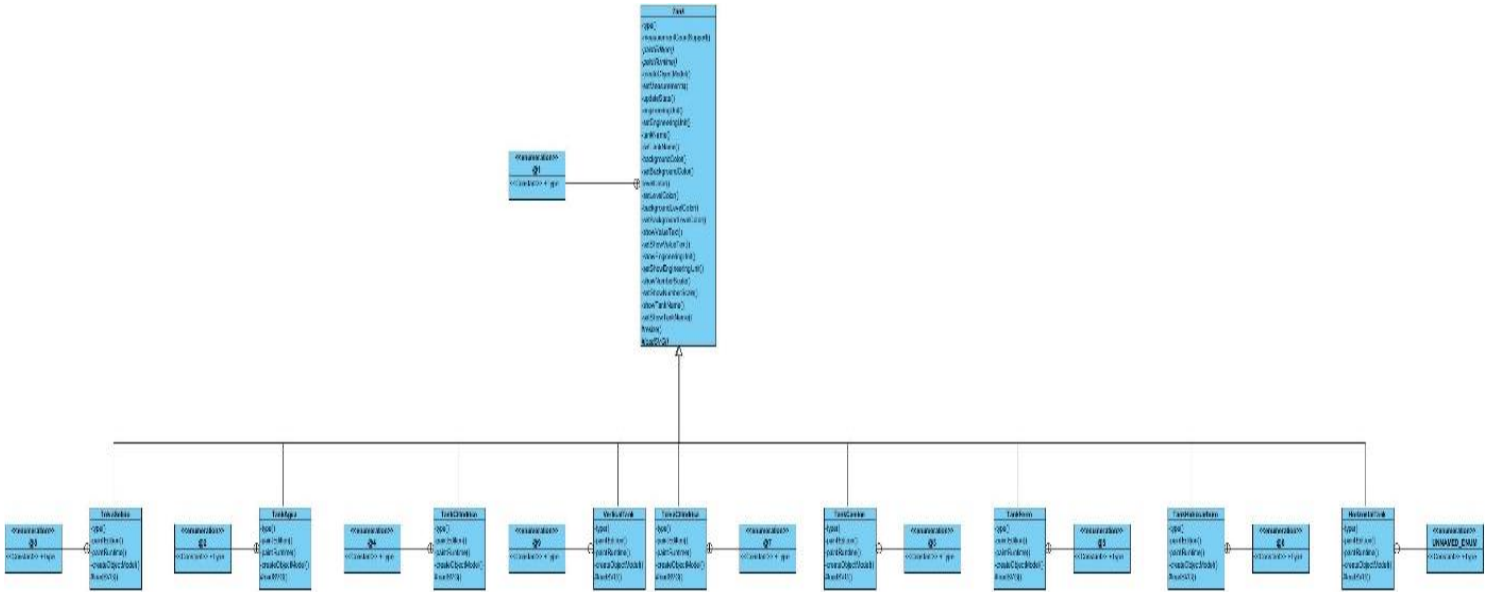


Ilustración 2: Diagrama de clase "Tank".

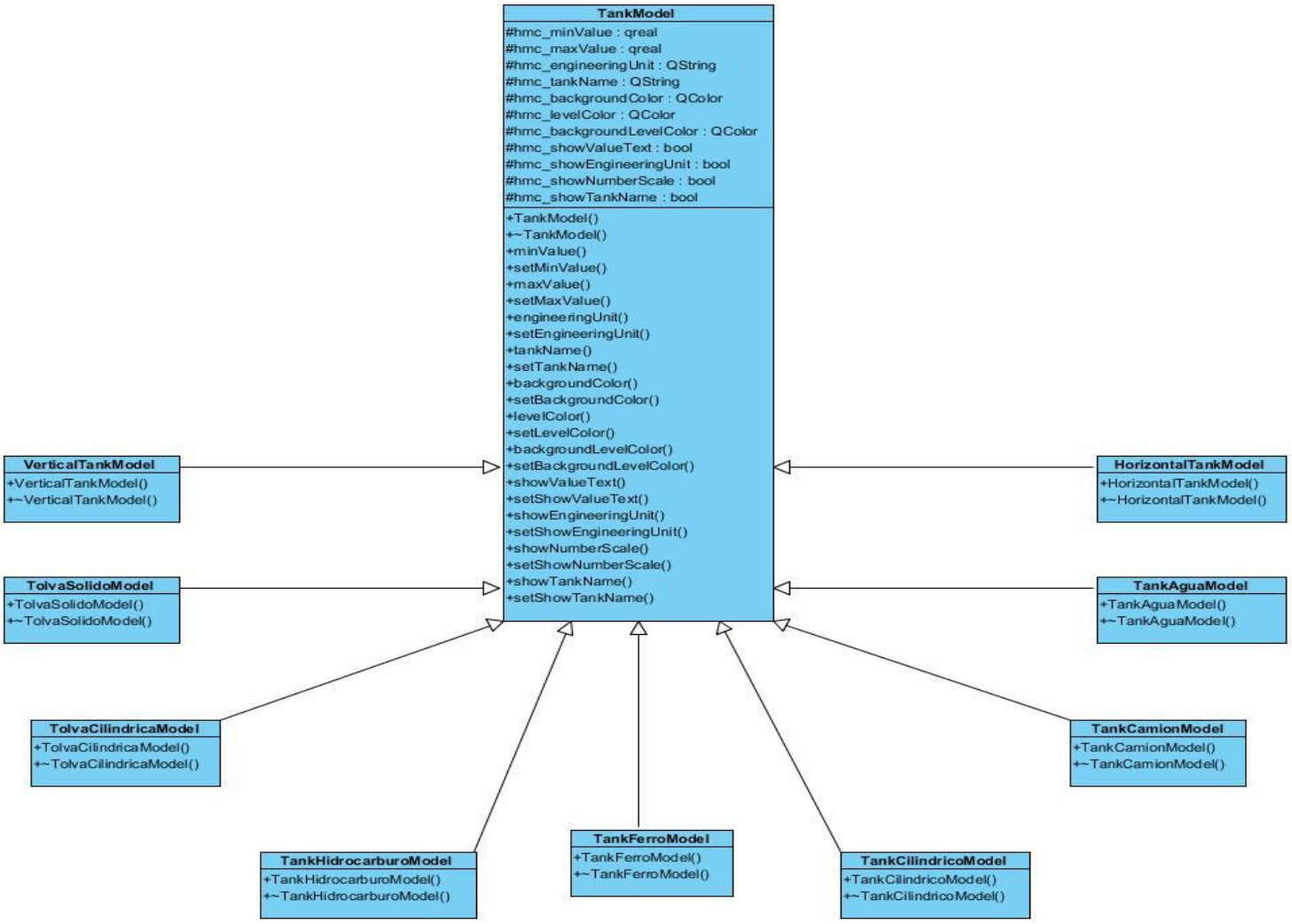


Ilustración 3: Diagrama de clase "TankModel".

2.6 PATRONES DE ARQUITECTURA

Los patrones de arquitectura ofrecen soluciones a problemas de arquitectura de software, dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. El patrón de arquitectura usado es el Modelo / Vista que propone Qt aplicado a la solución propuesta.

Qt 4 introdujo un nuevo conjunto de clases de vista del elemento que utilizan una arquitectura de modelo / vista para gestionar la relación entre los datos y la forma en que se presenta al usuario. La separación de la funcionalidad introducida por esta arquitectura ofrece a los desarrolladores una mayor flexibilidad para personalizar la presentación de artículos, y proporciona una interfaz modelo estándar para permitir una amplia gama de fuentes de datos para ser utilizado con las vistas de elementos existentes (28) .

Modelo / vista es el resultado de la combinación de la vista y el controlador de la arquitectura Modelo-Vista-Controlador. Esto todavía separa la forma en que los datos se almacenan de la forma en que se presenta al usuario, pero proporciona un marco sencillo basado en los mismos principios. Para permitir un manejo flexible de entrada del usuario, se introduce el concepto de delegado. La ventaja de tener un delegado es que permite que, la forma en que los elementos de datos se procesan y editan se personalice (28).

En general, las clases de modelo / vista se pueden separar en los tres grupos descritos anteriormente: modelos, vistas y delegados. Cada uno de estos componentes se define por las clases abstractas que proporcionan interfaces comunes y, en algunos casos, las implementaciones por defecto de características. Las clases abstractas están destinados a ser una subclase con el fin de proporcionar el conjunto completo de funcionalidad esperada por otros componentes; esto también permite que los componentes especializados puedan reescribirse (28).

Expresado en la solución propuesta el patrón de arquitectura modelo-vista se observa en la implementación de los componentes gráficos de tipo tanque, donde en cada uno de estos tipos se crearon las clases de sus modelos y vistas respectivamente y junto representan al componente. Para mostrar lo dicho anteriormente utilizamos al componente “HorizontalTank” el cual presenta la clase “HorizontalTankModel” como clase modelo y la clase “HorizontalTank” como su vista, formando ambas clases al componente gráfico.

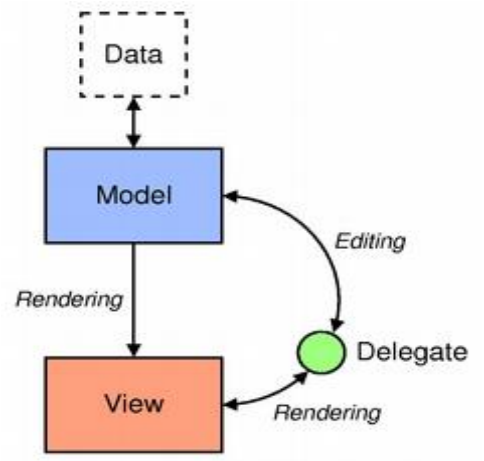


Ilustración 4: Patrón de Arquitectura Modelo/Vista.

2.7 PATRONES DE DISEÑO

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Estos facilitan la reutilización de arquitecturas y diseños de software exitosos (29).

Los patrones de diseño Banda de los Cuatro [Gang of Four, (GoF por sus siglas en inglés)] son clasificados principalmente por tres grupos como son (30):

- **Patrones creacionales:** utilizados para instanciar objetos, y así separar la implementación del cliente de la de los objetos que se utilizan. Con ellos intentamos separar la lógica de creación de objetos y encapsularla.
- **Patrones de comportamiento:** se utilizan a la hora de definir como las clases y objetos interaccionan entre ellos.
- **Patrones estructurales:** utilizados para crear clases u objetos que incluidos dentro de estructuras más complejas.

Los patrones GoF utilizados fueron Patrones de Comportamiento como:

- **Observador (Observer):** Define una dependencia de uno a muchos entre objetos de forma que, cuando un objeto cambia de estado, se notifica a los objetos dependientes para que se actualicen automáticamente. El uso de este patrón se ve reflejado en la funcionalidad ***updateState()*,** la cual es utilizada para notificarle al componente que la medición asociado a él ha sufrido cambios.
- **Plantilla (Template Method):** Define una operación, el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura. Es utilizado en clase abstracta donde el código común será usado por las clases que heredan de ella permitiendo la reescritura de determinados métodos. Reflejado en los métodos ***paintRuntime ()***, ***loadSVG ()*** y ***paintEdition ()***.

Los Patrones Generales de Asignación de Responsabilidades de Software (GRASP, por sus siglas en inglés) describen los principios fundamentales de la asignación de responsabilidades a objetos, de forma tal que se pueda diseñar software orientado a objetos. Utilizados en la aplicación encontramos algunos patrones como son:

- **Alta cohesión:** Facilita la solución al problema ¿Cómo mantener manejable la complejidad? mediante la asignación de responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. El uso de este patrón se ve reflejado en todo el diagrama debido a que cada clase almacena la información de ella misma de manera coherente.
- **Experto**: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. De forma general en el diseño del sistema se basa en asignar a cada clase la responsabilidad que solo ellas pueden realizar, pues cada una cuenta con la información necesaria para llevarlas a cabo. Por lo que en todas las clases del sistema es utilizado este patrón.

- **Bajo acoplamiento:** Facilita la solución al problema ¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización? Reflejada en todo el diagrama.
- **Polimorfismo:** Cuando por el tipo varían las alternativas o comportamientos afines, las responsabilidades del comportamiento se asignarán mediante operaciones polimórficas a los tipos en que el comportamiento presenta variantes, reflejado en la funcionalidad ***loadSVG ()***.

2.8 DIAGRAMA DE PAQUETES

Un diagrama de paquetes representa las dependencias entre los paquetes que componen un modelo, cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones. Los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema (31).

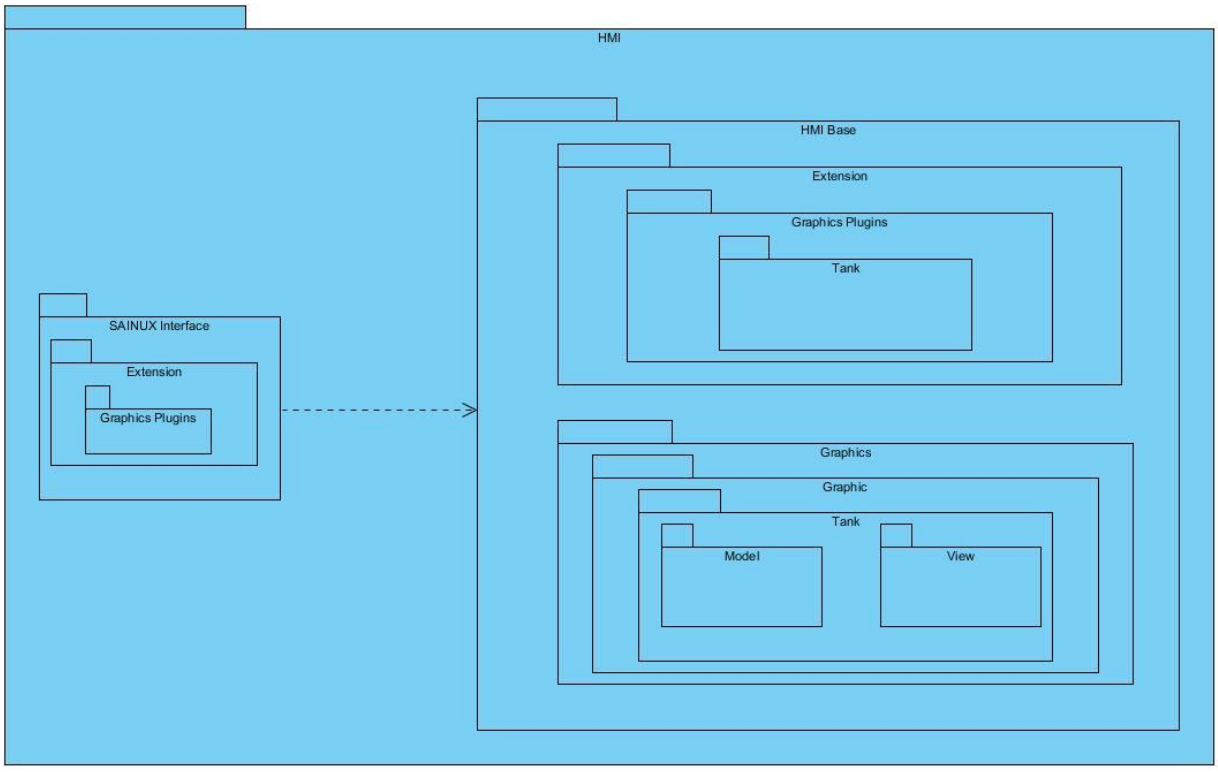


Ilustración 5: "Diagrama de Paquete".

En la solución propuesta el HMI representa el paquete genérico conteniendo a los sub paquetes de SAINUX Interface y HMI Base. La primera agrupa todas las entidades que son utilizadas para la interface del HMI en SAINUX, dentro del sub paquete Extension se encuentra el paquete Graphics Plugins donde se tienen a las entidades que representan las paletas de componentes para la interfaz de SAINUX. En el segundo sub paquete del HMI, es donde se encuentran todas las entidades y paquetes bases para el desarrollo del módulo HMI, independiente de su interfaz. Dentro del paquete Extension se encuentra el sub paquete Graphics Plugins el cual contiene las entidades que representa las paletas de componentes que están agrupada en sub paquetes de acuerdo a la clasificación del componente, en este caso Tank. En el paquete Graphics se encuentra el sub paquete Graphic donde se guardan las entidades de la biblioteca de componentes gráficos y se representan los componentes de tipo Tank compuesto por los sub paquetes View y Model respectivamente.

2.9 CONCLUSIONES

Con la realización del presente capítulo se define como punto de partida el modelo de la propuesta de solución, se especificaron además los requisitos funcionales y los no funcionales que debe cumplir el sistema, administrándola de forma rápida con las historias de usuario y sus pertinentes descripciones propuesta por la metodología seleccionada para el cumplimiento del sistema. Se realizó un plan de iteraciones y la estimación del desarrollo de la aplicación. Es definido el modelo de diseño del sistema y seleccionados los patrones arquitectónicos y de diseño a cumplir. Y se muestra el diagrama de paquetes del sistema dividido en sus agrupaciones lógicas y las dependencias entre esas agrupaciones y paquetes. De esta manera en el presente capítulo se modela el negocio lo más detallada posible para dar paso a la implementación del sistema.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

3.1 INTRODUCCIÓN

En el presente capítulo se describirá las disciplinas de implementación y prueba del sistema. Se expondrá el estándar de codificación utilizado para el desarrollo de la solución, así como su respectivo diagrama de componentes y de despliegue. Se realizarán las pruebas correspondientes para la validación del sistema como las pruebas internas y de aceptación, utilizando los métodos de caja blanca y caja negra.

3.2 ESTÁNDAR DE CODIFICACIÓN

La forma de escribir código es propia de cada programador y completamente diferente a la forma de cualquier otro. De la forma usada depende la facilidad para entender el código y retomar ciertas partes realizadas por otros integrantes, así como la depuración de las mismas. El estándar de codificación debería establecer cómo operar con la base de código existente, como si un único programador hubiera escrito todo el código de una sola vez.

Como la solución propuesta en este trabajo es parte del sistema SCADA SAINUX el estándar de codificación utilizado fue definido por el proyecto. Algunas de las pautas definidas son:

- 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.
- Se adopta el estilo de bloques de documentación de JavaDoc, el cual consiste de un bloque de comentario de estilo C.
- 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 una descripción breve se adopta el uso del comando @brief.
- Se deben, en lo posible, usar los tipos nativos del lenguaje. Es decir que en C++ se usará bool y no alguna extraña macro BOOL o BOOLEAN.
- Las funciones y variables globales deben tener nombres descriptivos, sin importar que sean largos. Ejemplo: cerrar_conexion().
- Las variables de corto alcance, por ejemplo un contador, deberían tener nombres cortos y simples.
- Cada función deberá estar documentada en el código en caso de ser requerido respondiendo a algunas interrogantes como: ¿Qué hace? ¿Cuáles son sus parámetros?

3.3 DIAGRAMA DE COMPONENTES

Un diagrama de componentes 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 (32).

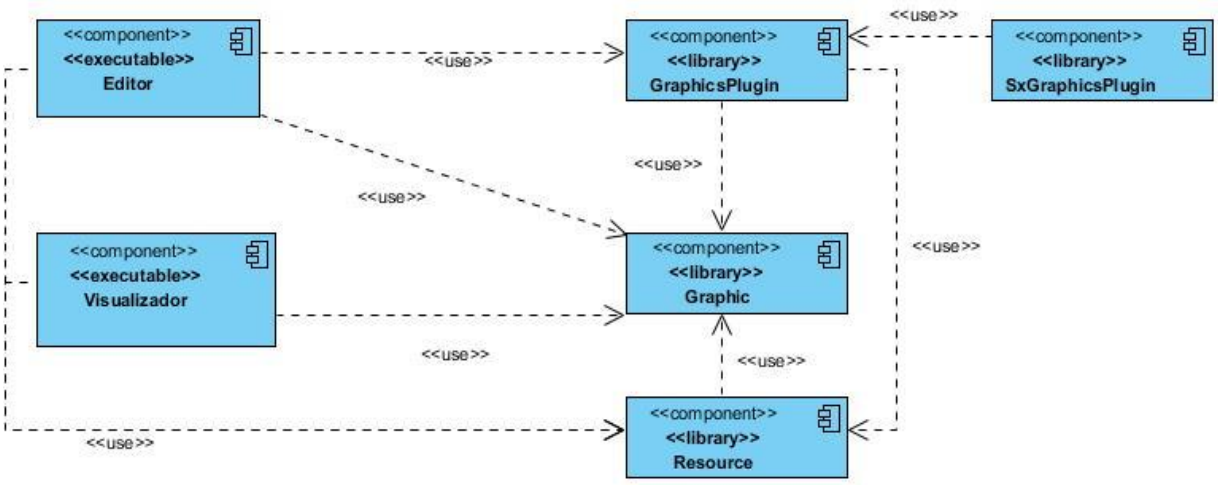


Ilustración 6: "Diagrama de Componentes".

3.4 DIAGRAMA DE DESPLIEGUE

Un diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos (33).Esto muestra la configuración de los elementos de hardware y cómo los elementos y artefactos del software se asocian.

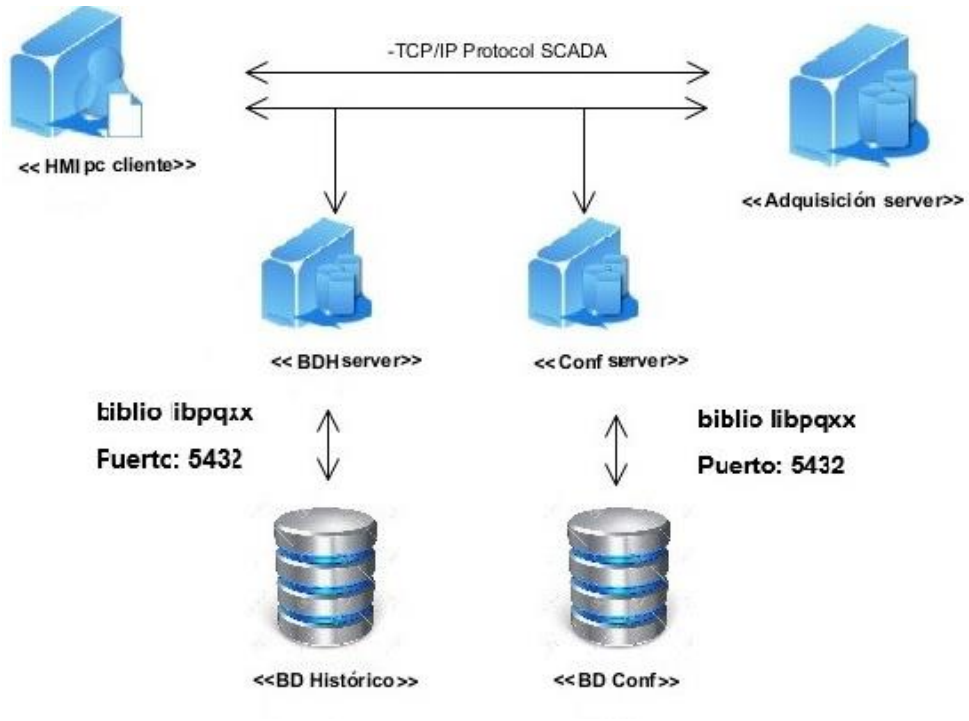


Ilustración 7: "Diagrama de Despliegue".

3.5 DISEÑO DE PRUEBAS

Las pruebas de software tienen un rol muy importante en el aseguramiento de la calidad ya que permiten detectar los errores introducidos en las fases previas del proyecto. Las pruebas realizadas a un producto de software se clasifican en dos categorías: de caja blanca y de caja negra.

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca se puede obtener casos de prueba que (34):

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Las pruebas de caja negra permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

3.5.1 PRUEBAS DE ACEPTACIÓN

Las pruebas de aceptación verifican que el sistema que recibe funciona y lo hace de acuerdo con las especificaciones. Las pruebas de aceptación se realizan de acuerdo con protocolos específicos del producto. Al participar en un proceso de prueba de aceptación, puede obtener un conocimiento más profundo acerca de cómo funciona el equipo. Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales (35).

En el proceso de desarrollo de las pruebas de aceptación se tomó como indicador 100% en caso de satisfactorias para pasar de iteración. Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo y el objetivo de estas pruebas no es tener un conjunto de casos escritos que cubran el 100% del código, sino poder realizarle pruebas al sistema desde el punto de vista del usuario.

Para representar las pruebas de aceptación se definieron los siguientes elementos:

- **Código:** Representa al caso de prueba, incluye el número de HU, de la prueba y si posee diferentes escenarios.
- **HU:** Número de la HU a la cual pertenece.
- **Nombre:** Junto al código conforma el identificador del caso de prueba.
- **Descripción:** Acción que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- **Entrada/Pasos de Ejecución:** Incluye las entradas necesarias para realizar el sistema, además de los pasos para realizar el caso de prueba.
- **Resultados Esperados:** Descripción de la respuesta del sistema ante el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

La siguiente tabla muestra las pruebas de aceptación de la HU2 perteneciente a la primera iteración de sistema. La cual fue escogida por ser una de los procesos más relevantes en el desarrollo del sistema.

Tabla 26: Caso de Prueba de Aceptación # 1.

Caso de Prueba de Aceptación	
Código: HU1_P1	**HU:** 1
Nombre: Configurar unidad de ingeniería del componente.	
Descripción: Configurar la unidad de ingeniería del componente pasada por parámetro.	
Condiciones de Ejecución: Debe existir un componente desplegado al cual se le asocie el parámetro de unidad de ingeniería correspondiente.	
Entrada/Pasos de Ejecución:	

Cargar fichero. Interpretación de los parámetros que define al objeto gráfico. Interpretación de la propiedad transformación. Conformación del objeto gráfico. Conformación del objeto transformación. Visualización del objeto gráfico. Aplicación de la transformación al objeto gráfico.
Resultado Esperado: Visualización de la modificación del parámetro unidad de ingeniería del componente .
Evaluación de la Prueba: Satisfactoria.

Tabla 27: Caso de Prueba de Aceptación # 2.

Caso de Prueba de Aceptación	
Código: HU2_P1	**HU:** 2
Nombre: Mostrar u ocultar unidad de ingeniería del componente.	
Descripción: Mostrar la unidad de ingeniería del componente en caso de que se desee u ocultarla.	
Condiciones de Ejecución: Debe existir un componente desplegado al cual se le asocie la configuración de mostrar u ocultar correspondiente.	
Entrada/Pasos de Ejecución: Cargar fichero. Interpretación de los parámetros que define al objeto gráfico. Interpretación de la propiedad transformación. Conformación del objeto gráfico.	

Conformación del objeto transformación. Visualización del objeto gráfico. Aplicación de la transformación al objeto gráfico.
Resultado Esperado: Visualización del mostrado u ocultado del parámetro unidad de ingeniería del componente .
Evaluación de la Prueba: Satisfactoria.

La descripción de las siguientes pruebas realizadas al sistema está plasmada en el anexo 1.

En la primera iteración se realizaron dos etapas de pruebas, en la primera etapa se realizaron 10 casos de prueba donde se obtuvieron 3 caso de pruebas no satisfactorios, las cuales representan un 30%, posteriormente se corrigieron y se realizó una segunda etapa de pruebas donde no se obtuvieron no conformidades, logrando así un 100% de resultados satisfactorios y cumpliendo el criterio de aceptación.

Las no conformidades encontradas en las etapas de pruebas realizadas en la primera iteración se clasifican en funcionalidades que no realizaban la acción prevista y errores visuales donde el sistema no mostraba lo requerido. En la ilustración 11 se muestra el comportamiento de las pruebas realizadas en la primera iteración.

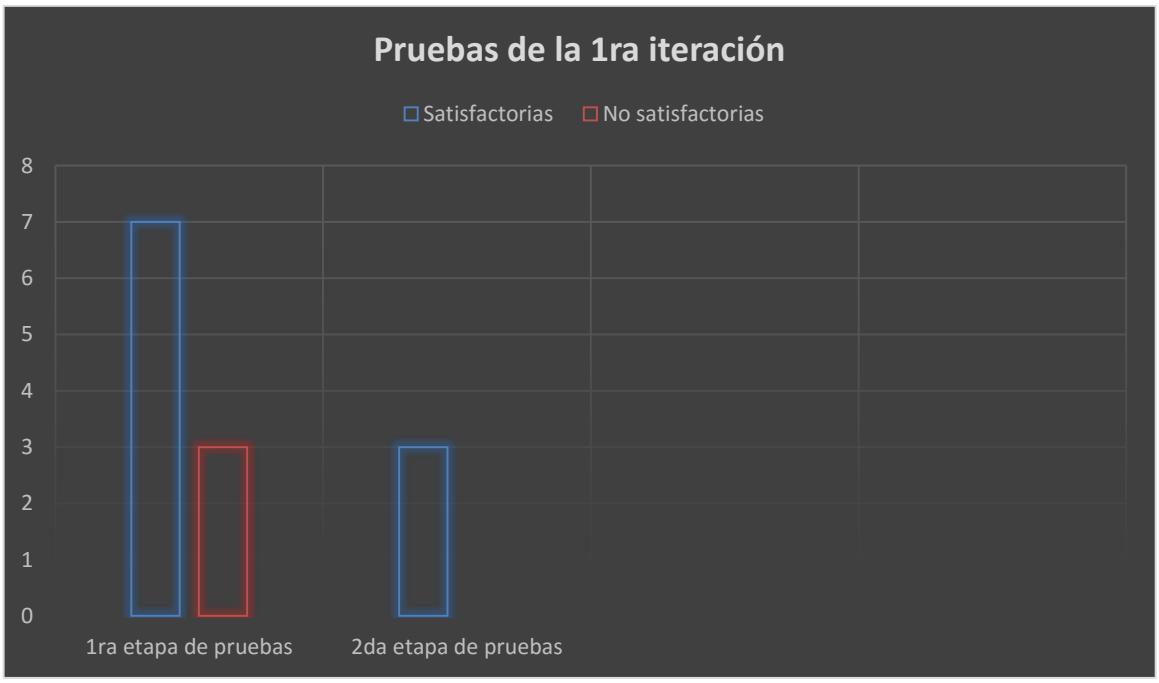


Ilustración 8: No conformidades de la 1ra iteración.

En la segunda iteración se realizaron 3 casos de pruebas, detectándose en la primera etapa un caso de prueba no satisfactorio representando el 33.3 %. En la segunda etapa de pruebas no se detectaron no conformidades logrando así el criterio de aceptación.

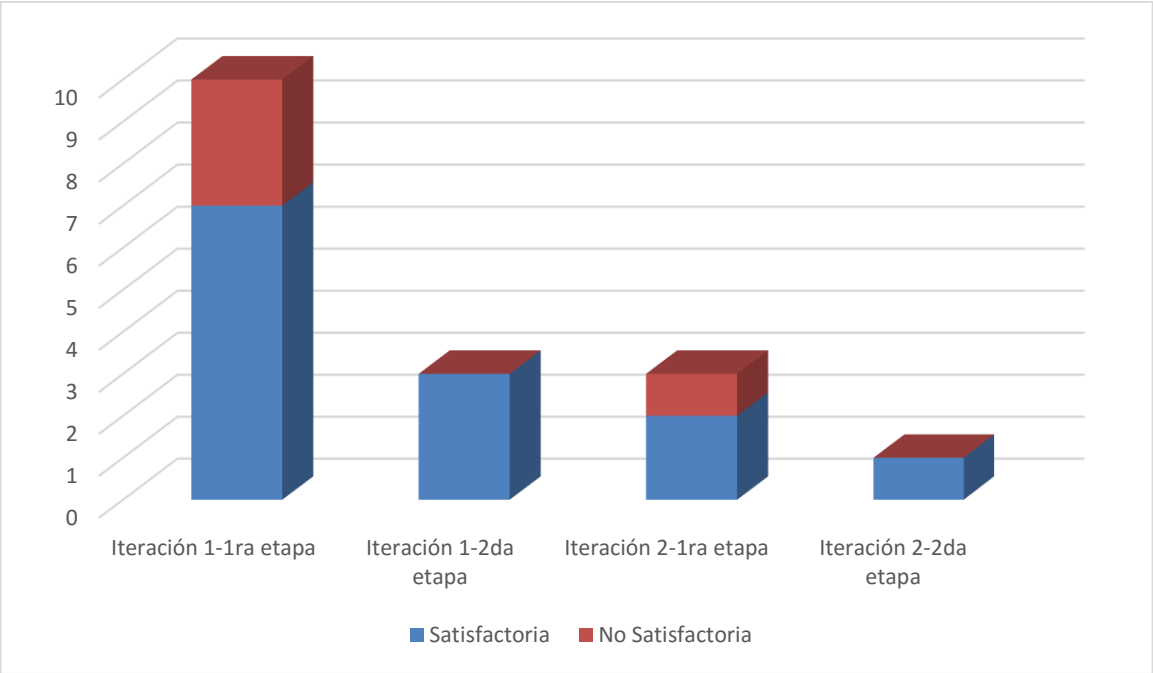


Ilustración 9: Pruebas de aceptación satisfactorias por iteración.

3.6 CONCLUSIONES PARCIALES

Durante el presente capítulo se expuso el estándar de codificación utilizado para el desarrollo de la solución en su fase de implementación como establece la metodología para el cumplimiento del sistema, así como su respectivo diagrama de componentes y de despliegue. Se desarrolló el proceso de implementación del producto, el cual fue validado a través de las pruebas de aceptación y se tomó como criterio de aprobación el 100% de los casos de prueba satisfactorios por iteración. Se corrigieron todas las no conformidades encontradas, además se determinó que la propuesta de solución desarrollada cumple con todos los requerimientos acordados con el cliente.

CONCLUSIONES GENERALES

Con la realización del presente trabajo se lograron desarrollar los componentes gráficos de almacenamientos a integrar al sistema SCADA SAINUX, para lo cual:

- ✓ El diagnóstico inicial apoyado en la aplicación de los métodos científicos permitió identificar las funcionalidades y las tecnologías libres a utilizar en el desarrollo de la propuesta de solución.
- ✓ Se obtuvo los componentes gráficos de almacenamiento para el sistema SCADA SAINUX, obteniendo los recursos para la representación gráfica de los procesos de almacenamiento, así como también los mecanismos gráficos para la supervisión y control de dichos procesos.
- ✓ Mediante la realización de las pruebas de aceptación se comprobó que la propuesta de solución cumple con los requerimientos pactados con el cliente y le proporciona conformidad. Además, permitió el cumplimiento del objetivo trazado inicialmente en la investigación.

RECOMENDACIONES

1. Seguir ampliando los diferentes tipos de componentes gráficos de almacenamiento en el SCADA SAINUX.
2. Incorporar los componentes gráficos desarrollados en el visor web que existe en el centro.

REFERENCIAS

1. ***Liniamientos de la Política Económica y Social del Partido y La Revolución.*** **2000.**

2. Valido Fajardo, Luis Andrés. ***Mecanismo de incorporación de gráficos SVG al HMI del SCADA GALBA.*** **La Habana : s.n., 2013.**

3. Pérez Feria, Jordenys. ***Análisis de la arquitectura del módulo Interfaz Hombre Máquina del SCADA Nacional.*** **La Habana : s.n., 2008.**

4. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 13 de enero de 2016.] http://www.ecured.cu/Sistema_SCADA.**

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

6. Abaffy, Carlos y Lárez, Jesús. ***Desarrollo de Scada en una Plataforma de Software Libre.*** **s.l. : CITEG, 2007.**

7. Qué es, Significado y Concepto. ***Qué es, Significado y Concepto.*** **[En línea] [Citado el: 15 de enero de 2016.] http://definicion.de/almacenamiento/.**

8. [En línea] [Citado el: 16 de enero de 2016.] http://www.xml.org/.

9. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 16 de enero de 2016.] http://www.ecured.cu/SVG.**

10. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 16 de enero de 2016.] http://www.monografias.com/trabajos6/sisop/sisop.shtml.**

11. [En línea] [Citado el: 16 de enero de 2016.] http://xml.ier.unam.mx/xml/Linux/glinux-2.html.

12. [En línea] [Citado el: 16 de enero de 2016.] http://www.monografias.com/trabajos6/sisop/sisop.shtml.

13. Debian. ***Debian.*** **[En línea] [Citado el: 16 de enero de 2016.] https://www.debian.org/releases/stable/armhf/ch01s03.html.es.**

14. Stroustrup, Bjarne. ***The C++ Programming Language.*** **s.l. : Addison-Wesley, 1997.**

15. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 16 de enero de 2016.] http://www.ecured.cu/C%2B%2B.**

16. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 16 de enero de 2016.] http://www.ecured.cu/Qt.**

17. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 16 de enero de 2016.] http://www.ecured.cu/Qt_Creator.**

18. [En línea] [Citado el: 17 de enero de 2016.] http://pixelcoblog.com/qt-creator-completo-entorno-de-desarrollomultiplataforma/.

19. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 17 de enero de 2016.] http://www.ecured.cu/Herramienta_CASE.**

20. JACOBSON, I., BOOCH y JAMES. ***El Proceso Unificado de Desarrollo de Software.*** **2000.**

21. Modelos Y Metodologías Para El Desarrollo De Software. ***Modelos Y Metodologías Para El Desarrollo De Software.*** **[En línea] [Citado el: 17 de enero de 2016.] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.**

22. Sánchez, Tamara Rodríguez. ***Metodología de desarrollo para la Actividad productiva de la UCI.*** **La Habana : s.n., 2015.**

23. ***Especificaciones de los requisitos del Software.*** **s.l. : IEEE STD 830-1998.**

24. Sommerville, Ian. ***Software Engineering.*** **2006.**

25. Pressman. ***Software Engineering*** **. 2001.**

26. Beck, K. ***Extreme Programming Explained.*** **s.l. : Pearson Education, 1999.**

27. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 10 de abril de 2016.] http://www.ecured.cu/Diagrama_de_Clase.**

28. QT Documentation. ***QT Documentation.*** **[En línea] [Citado el: 10 de mayo de 2016.] http://doc.qt.io/qt-4.8/model-view-programming.html.**

29. Harmes, Ross y Díaz, Dustin. ***Pro JavaScript.Design Patterns.*** **New York City : Chris Mills y Tom Welsh, 2008.**

30. [En línea] [Citado el: 20 de abril de 2016.] http://programacionsolida.com.ar/2012/07/patrones-de-diseno-decomportamiento_09.html.

31. W. Ambler , Scott. Introduction to UML 2 Package Diagrams.

32. W. Ambler , Scott . ***UML 2 Component Diagram Guidelines.***

33. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 26 de abril de 2016.] http://www.ecured.cu/Diagrama_de_despliegue.**

34. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 1 de mayo de 2016.] http://www.ecured.cu/Pruebas_de_caja_blanca.**

35. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 2 de mayo de 2016.] http://www.ecured.cu/Niveles_de_prueba_de_software.**

36. Blanco Bueno, Carlos. Construcción y Pruebas de Sotfware. Cantabria : s.n.

37. EcuRed. ***EcuRed.*** **[En línea] [Citado el: 2 de mayo de 2016.] http://www.ecured.cu/Pruebas_de_caja_blanca.**

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