



Facultad 5

Centro de Informática Industrial

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

Título: Componentes gráficos para la representación de accesorios de tuberías en el SCADA SAINUX.

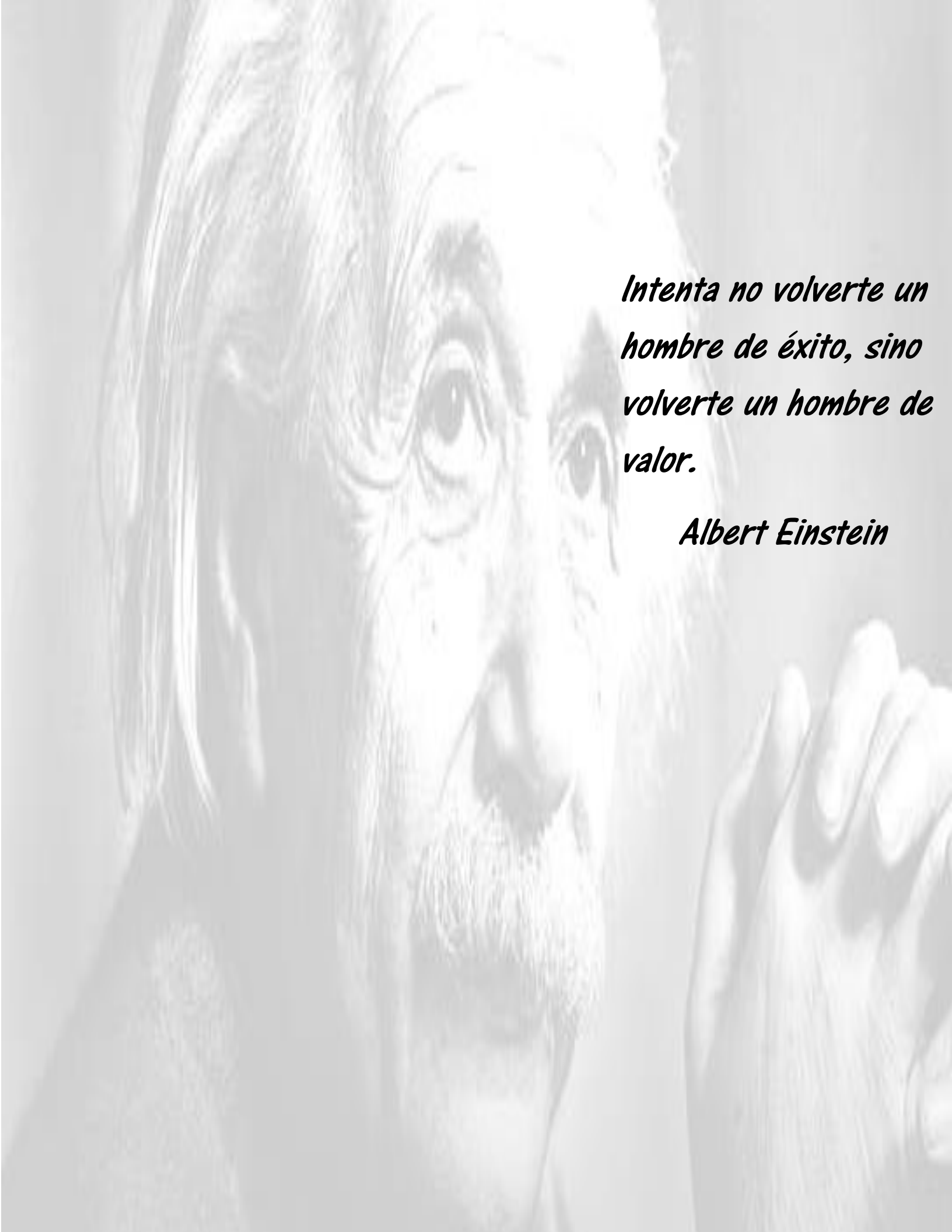
Autor: Rosmery Pedraza Ceballo.

Tutor: Ing. Luis Andrés Valido Fajardo.

Co-Tutor: Ing. Liuver Carrera Espinosa.

Ciudad de La Habana, 2016.

“Año 58 de la Revolución”



*Intenta no volverte un
hombre de éxito, sino
volverte un hombre de
valor.*

Albert Einstein

Declaración de Autoría

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

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

Firma del autor

Rosmery Pedraza Ceballo

Firma del tutor

Ing. Luis Andrés Validos Fajardo

Firma del co-tutor

Ing. Liuver Carrera Espinosa

Síntesis de los tutores

- **Ing. Luis Andrés Valido Fajardo:** graduado de Ingeniero en Ciencias Informáticas en el año 2013 en la Universidad de las Ciencias Informáticas. Posee 2 años de experiencia en el tema.
Correo Electrónico: lvalido@uci.cu
- **Ing. Liuver Carrera Espinosa:** graduado de Ingeniero en Ciencias Informáticas en el año 2015 en la Universidad de las Ciencias Informáticas. Posee 1 año de experiencia en el tema.
Correo Electrónico: liuver@uci.cu

Dedicatoria

Quiero dedicar este trabajo a mi papá que ha sido el motor impulsor de mi vida, la persona que jamás me ha dado la espalda, siempre ha confiado en mí, en lo que decido, gracias a ti papi por tus consejos, confianza, por darme todo a cambio de hacerme feliz, por luchar por mi superación profesional. Sé que en estos momentos la vida nos la ha puesto difícil, pero Dios está con nosotros y ayudará a que todo esté bien, en la lucha diaria por tu vida estás luchando aún más por la mía, te amo.

Agradecimientos

Muchas personas fueron participes de la realización de este trabajo, a todas ellas les ofrezco mis más sinceros agradecimientos en especial a:

- ✓ Mi mamá y mi papá que además de darme la vida, supieron hacer de mí una persona de bien, me enseñaron a luchar por las cosas que verdaderamente quiero, a ser fuerte ante las situaciones de la vida, a no rendirme nunca a pesar de las circunstancias. Gracias por ofrecerme su apoyo, la verdad sin ustedes no habría logrado convertirme en lo que ahora soy, le agradezco a dios por haberme dado unos padres como ustedes, gracias por existir, yo los amo.
- ✓ Mi hermanito lindo gracias por cuidarme tanto, por siempre pensar en mí, por la ayuda incondicional que me brindas, tati eres muy importante en mi vida ¿lo sabias?, quiero siempre tenerte cerca.
- ✓ Mis sobrinitos bellos, esos dos diablitos que no me dan un momento de suspiro, que me dejan sin aliento y sin paciencia, pero que son la alegría de la casa, que aunque para poder estudiar tenía que esperar a que durmieran. Mis cuquis ustedes son extremadamente importantes en mi vida.
- ✓ A mi tío Alfre, que a pesar de estar lejos, su ayuda nunca me ha faltado, gracias por todo lo que haces por mí, estaré eternamente agradecida, te quiero mucho.
- ✓ Mi amigo, hermano de Universidad, mi aliento para que cada día sea mejor. Li eres lo más puro y lindo que me ha pasado en la Universidad, has estado a mi lado en estos largos años sin esperar nada a cambio, has estado siempre para mí en los buenos y malos momentos, sin ti mi vida estaría incompleta, te necesito.
- ✓ Una persona muy especial que llevo a mi vida para quedarse y que es mi segunda madre, Mary gracias por llegar a mi vida te ganaste mi amor, mi respeto, mi confianza, y te estaré eternamente agradecida, porque tú te robaste mi corazón, a pesar de todo, nada ha cambiado entre nosotras nunca olvides a tu bebi.
- ✓ Mi compañera de cuarto durante estos largos años, por su amistad, su paciencia, su ayuda, su comprensión, nunca olvidaré todo lo que has hecho por mí, Lety eres una excelente persona y lograste convertirte en la hermana que nunca tuve, te quiero mucho.

- ✓ El Team Roice: Roi, Rodni, Futiel, Ortiz, Rubén, Adry y Di chicos ustedes son lo máximo, hemos sido el mejor equipo de la historia, siempre juntitos y cuidándonos el uno al otro. Nunca me había divertido tanto.
- ✓ Mis tutores, Valido por confiar siempre en mí, por su ayuda, entrega y apoyo incondicional, paciencia, por dedicarme tanto tiempo y Liuver por su ayuda no solo como tutor sino también durante la carrera. Sin ustedes no hubiera sido posible convertir este sueño en realidad.
- ✓ A todos los amigos por cada momento que me dedicaron Reitel, Yanet, Claudiña, Xiuny, Leduán, Yaser, mis compañeros de grupo. A todos muchas gracias, los quiero mucho.
- ✓ Todos los profesores que ayudaron a mi formación como profesional durante este largo camino en especial: Zaida, Yirka, Sailyn, Karina, Millé y Mariela.
- ✓ A todos los que de una forma u otra lograron que mi sueño de ser profesional se convirtiera en realidad, gracias.

Resumen

Un acueducto y un oleoducto son instalaciones responsables de la gestión integral de agua y petróleo respectivamente mediante el uso de las redes de tubería, las cuales permiten el transporte de estos líquidos en forma de fluido continuo, con el fin de distribuirlo desde un lugar en el que esta accesible en la naturaleza hasta un punto de consumo distante, generalmente una ciudad.

Los sistemas de supervisión, control y adquisición de datos (SCADA, por sus siglas en inglés) se encargan de supervisar y controlar procesos industriales tales como los que ocurren en los acueductos, oleoductos y en refinerías de petróleo.

El siguiente trabajo tiene como objetivo el desarrollo de componentes gráficos que permitan incrementar los grados de representación de los accesorios que conforman las redes de tubería en la Interfaz Hombre Máquina del sistema SCADA SAINUX, con el uso de tecnologías y herramientas como: QtCreator, Visual Paradigm y lenguaje de programación C++. Como resultados se obtuvieron componentes gráficos que permiten representar los accesorios que conforman las redes de tubería, siendo una solución con mayores prestaciones y calidad en la representación de la red. Además se desarrolló una animación para representar el transporte de fluido cuando este circula por los accesorios.

Palabras claves: Red de tubería, Fluido, SCADA.

Índice

Introducción	1
Capítulo 1. Fundamentación teórica.....	6
1.1 Introducción.....	6
1.2 Conceptos asociados.....	6
1.2.1 Componente Gráfico	6
1.2.2 Despliegue.....	6
1.3 Definición de SCADA	7
1.3.1 Funciones principales que debe cumplir un SCADA	7
1.3.2 Módulos del SCADA SAINUX	7
1.3.3 Módulo Interfaz Hombre – Máquina	9
1.4 Componentes Gráficos del HMI SAINUX	9
1.5 Red de tubería.....	10
1.5.1 Sistemas de representación de redes de tubería	10
1.5.2 Componentes de una red de tubería	11
1.6 Metodología de desarrollo de software.....	13
1.6.1 Metodología AUP-UCI.....	13
1.6.2 Fases de AUP-UCI.....	14
1.6.3 Disciplinas de AUP-UCI.	15
1.7 Herramientas y tecnologías	16
1.7.1 Gráficos Vectoriales Escalables (SVG)	16
1.7.2 Editor Gráfico	17
1.7.3 Lenguaje de programación.....	17

1.7.4 Marco de trabajo	17
1.7.5 Entorno de Desarrollo Integrado (IDE)	18
1.7.6 Lenguaje Unificado de Modelado (UML)	18
1.7.7 Visual Paradigm	19
1.7.8 Sistema Operativo.....	19
1.7.9 Herramienta de control de versiones.....	19
1.8 Conclusiones parciales.....	20
Capítulo 2. Análisis y diseño de la solución.	21
2.1 Introducción.....	21
2.2 Fase de inicio.....	21
2.3 Modelo de dominio	21
2.4 Fase de ejecución.....	23
2.5 Captura de requisitos	24
2.4.1 Requisitos funcionales (RF)	24
2.4.2 Requisitos no funcionales (RNF).....	26
2.4.3 Historias de Usuario.....	26
2.5 Planificación del desarrollo.	27
2.5.1 Estimación de esfuerzos e iteraciones por historia de usuario.	27
2.5.2 Plan de desarrollo	30
2.6 Diagrama de paquetes	31
2.7 Diagrama de secuencia.....	32
2.8 Diagrama de clases	33
2.8.1 Descripción del diagrama de clases.....	34
2.9 Arquitectura del sistema.....	35
2.9.1 Patrón de arquitectura de software	36
2.10 Patrones de diseño.....	37

2.10.1 Patrones GRASP	37
2.10.2 Patrones GOF	38
2.11 Conclusiones Parciales	39
Capítulo 3. Implementación y Pruebas.....	40
3.1 Introducción.....	40
3.2 Modelo de implementación.....	40
3.3 Estándar de codificación	41
3.4 Diagrama de despliegue	42
3.4.1 Descripción del diagrama de despliegue.....	43
3.5 Pruebas de software.....	43
3.5.1 Tipos de pruebas de software	44
3.5.2 Pruebas de caja negra.....	45
3.5.2.1 Diseño de casos de prueba.....	45
3.6 Conclusiones Parciales	47
Conclusiones generales.....	49
Recomendaciones	50
Referencias	51

Índice de Tablas

Tabla 1. Fases de AUP-UCI (17).....	14
Tabla 2. Disciplinas de AUP-UCI (17).	15
Tabla 3 Historia de Usuario 8.....	26
Tabla 4 Estimación de esfuerzos.	27
Tabla 5 Caso de Prueba 12	46
Tabla 6 Resultado de las iteraciones.	46

Índice de Figuras

Fig. 1 Módulos del SCADA SAINUX.	9
Fig. 2. Componentes de una red de tubería.	12
Fig. 3 Modelo de Dominio para la red de tuberías.	23
Fig. 4 Diagrama de Paquetes de Componentes Gráficos.	31
Fig. 5 Diagrama de secuencia- Mostrar/Ocultar bridas pertenecientes al componente.	33
Fig. 6 Diagrama de Clases de Componentes Gráficos.	34
Fig. 7 Patrón de Arquitectura de Software Modelo-Vista.	36
Fig. 8 Diagrama de componentes.	41
Fig. 9 Diagrama de despliegue	43
Fig. 10 Incidencias detectadas en el sistema	47

Introducción

La humanidad ha transitado por varias etapas en el desarrollo tecnológico, lo cual ha exigido una constante investigación científica que impulse esta evolución. En el extenso campo de la informática tiene lugar la automatización y supervisión de procesos industriales, que no es más que el uso de sistemas computarizados para controlarlos. La automatización de estos procesos se ha convertido en un gran avance para el sector empresarial, con su uso, se han revolucionado las producciones a gran escala, pues dicha automatización tiene el objetivo de mejorar la calidad de los productos.

Actualmente estos procesos industriales se monitorean y controlan mediante los sistemas de supervisión, control y adquisición de datos (SCADA, por sus siglas en inglés). Un sistema SCADA es el encargado de monitorear y controlar un sitio completo o un sistema que se extiende sobre una gran distancia. La instalación de un SCADA necesita un hardware de señal de entrada y salida, sensores y actuadores, controladores, interfaz hombre máquina (HMI, por sus siglas en inglés), redes y base de datos (1). Los sistemas SCADA se componen por módulos como: manejadores, núcleo de procesamiento de datos, base de datos históricos y HMI.

La Universidad de las Ciencias Informáticas (UCI), tiene un peso importante dentro del sistema científico cubano lo que impone una destacada participación en las investigaciones. Dichas investigaciones están dirigidas a buscar soluciones a problemas que se presentan en el proceso de automatización. El Centro de Informática Industrial (CEDIN) perteneciente a la facultad 5, desarrolla proyectos de este tipo entre los cuales se encuentra el SCADA SAINUX (Sistema de Automatización Industrial basado en GNU/LINUX) el cual es un sistema SCADA distribuido, orientado a componentes y en proceso de desarrollo por especialistas del centro.

Situación problemática:

El módulo HMI dentro del SCADA se encarga de representar, en un ordenador todos los procesos que ocurren en el campo. Muestra los componentes implicados, los sensores, las estaciones remotas y el sistema de comunicación. Este módulo le permite al operador¹ tener control total

¹ Operador: Técnico encargado de manejar y hacer que funcionen ciertos aparatos.

sobre el sistema, estar en contacto directo con el mismo, realizar la supervisión y el control del proceso en general.

El HMI consta de dos entornos: El entorno de configuración (EC), donde el mantenedor² configura la información específica del área que se desea supervisar y diseña los despliegues, los cuales haciendo uso de los componentes gráficos permiten simular los procesos de campo; y el entorno de visualización (EV), donde el operador puede supervisar y controlar la configuración realizada en el EC, interactuando con los componentes gráficos para 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 biblioteca³ de componentes gráficos (CG) que permite recrear, de la manera más real posible, los procesos de campo. Estos gráficos pueden ser figuras geométricas como: líneas, rectángulos, elipses, textos, polígonos, polilíneas y curvas; o complejas como los componentes de hardware utilizados en la industria que se desea supervisar.

La biblioteca de componentes gráficos del HMI del SCADA SAINUX no le provee al mantenedor componentes para la representación de las redes de tubería, siendo estas usadas para la representación de procesos donde se supervisa y controla el flujo de líquidos y gases. Para dar solución a dicha situación se utiliza una imagen de una red de tuberías o se combinan varias figuras geométricas simples, lo cual tiene como desventajas que:

- ✓ La imagen en la mayoría de los casos no es la representación fiel de la red que se desea supervisar.
- ✓ Cuando se redimensiona, la imagen pierde calidad, por lo que se afecta la representación gráfica.
- ✓ El mantenedor no puede personalizar ni configurar la red según su interés.
- ✓ La compleja configuración del proceso a supervisar, ocasionada por la representación de la red con figuras simples.

A partir de la **situación problemática** planteada anteriormente, se propone como **problema de**

² Mantenedor: Persona ocupada del mantenimiento de aparatos o servicios.

³ Biblioteca: Conjunto de subprogramas que contienen código y datos, que proporcionan servicios a programas.

la investigación: ¿Cómo contribuir a incrementar los grados de configuración y visualización de los accesorios que componen las redes de tubería en la Interfaz Hombre Máquina de sistemas SCADA SAINUX?

En correspondencia con el problema planteado, se formula como **objeto de estudio:** La representación de los accesorios que conforman las redes de tubería en la Interfaz Hombre Máquina del sistema SCADA.

Siendo el **campo de acción:** La representación de los accesorios que conforman las redes de tubería en la Interfaz Hombre Máquina del sistema SCADA SAINUX.

De modo que se define como **objetivo general:** Desarrollar componentes gráficos que permitan incrementar los grados de representación de los accesorios que conforman las redes de tubería en el proceso de visualización de la Interfaz Hombre Máquina del sistema SCADA SAINUX.

De acuerdo al objetivo expuesto se definen las siguientes **tareas investigativas:**

1. Elaborar el marco teórico de la investigación a través del estudio del estado del arte que existe actualmente sobre el tema.
2. Seleccionar los principales elementos que componen los accesorios de una red de tubería y su representación en sistemas SCADA.
3. Caracterizar los principales elementos que componen los accesorios de una red de tubería y su representación en sistemas SCADA.
4. Realizar el levantamiento de requisitos funcionales y no funcionales.
5. Diseñar los componentes gráficos que brinden la solución al problema planteado.
6. Implementar los componentes gráficos que brinden la solución al problema planteado.
7. Realizar las pruebas necesarias para validar el cumplimiento de los requerimientos.

Luego de obtener toda la información relacionada con el tema de la investigación se plantean como **posibles resultados:**

- ✓ Componentes gráficos para la representación de dispositivos de monitoreo en las redes de tubería.
- ✓ Componentes gráficos para la representación de accesorios de las redes de tubería.
- ✓ Paleta de componentes gráficos especializada en la representación de redes de tubería.
- ✓ Una representación gráfica de la red, más cercana a la realidad.

Con el propósito de desarrollar las tareas planteadas se emplearon diferentes **métodos de investigación científica**, los cuales se explican con más detalles a continuación:

Métodos teóricos

Histórico-Lógico: Permitió entender y explicar los antecedentes de los accesorios de una red de tubería en los sistemas SCADA, sus tendencias actuales e importancia.

Modelación: Posibilitó la elaboración de múltiples diagramas para un mejor entendimiento y solución del problema.

Analítico-Sintético: Logró procesar teorías e información obtenida por vía empírica permitiendo la extracción de los elementos más importantes para el desarrollo del trabajo.

Métodos empíricos:

Revisión y consulta de la documentación: Permitió el estudio de diferentes documentos y tipos de bibliografías, posteriormente se seleccionó la información necesaria para la investigación.

Observación: Método aplicado para conocer el funcionamiento existente en los despliegues del SCADA SAINUX, mediante el comportamiento de los dispositivos de campo en las propiedades de los objetos gráficos y sumarios, empleados para la toma de decisiones de los operadores. Permite la recogida de información de cada uno de los conceptos y variables definidas.

Entrevista: Encuentro planificado para obtener información referente al producto que será desarrollado, es importante una buena comunicación entre los participantes.

El presente Trabajo de Diploma está estructurado de la siguiente forma:

Capítulo 1: “Fundamentación Teórica”, en este capítulo se muestra la elaboración del marco teórico de la investigación. Este se desarrolla mediante la definición de conceptos y características fundamentales de los componentes gráficos para la representación de redes de tuberías. Conjuntamente se definen las tecnologías y herramientas que se utilizarán en el diseño y posterior implementación de la solución propuesta.

Capítulo 2: “Análisis y Diseño de la solución”, este capítulo se centra fundamentalmente en el análisis y diseño de la aplicación. Basando sus principales objetivos en presentar los resultados de la fase de ejecución de la solución propuesta. Se obtienen los requisitos funcionales y no

funcionales que el sistema debe cumplir para satisfacer las necesidades del cliente, definiéndose además la arquitectura y los patrones de diseño que se utilizan.

Capítulo 3: “Implementación y Pruebas”, en este capítulo se describe la fase de implementación del sistema, según la metodología propuesta. Se realiza la fase de pruebas, a la cual se somete el producto para validar su correcto funcionamiento, permitiendo comprobar que el mismo cumple con todos los requerimientos del cliente.

Capítulo 1. Fundamentación teórica

1.1 Introducción

El presente capítulo tiene como objetivos definir y elaborar un marco teórico donde se expongan temas fundamentales que sustenten la investigación. Para ello se describe la concepción general del SCADA, las funciones que este debe cumplir y los módulos que posee el SCADA SAINUX, del mismo modo se introduce el sistema de representación de una red de tubería, así como los diferentes componentes para su representación. Además, se mencionan las principales herramientas, tecnologías y la metodología de desarrollo de software que serán empleadas en la construcción de la solución.

1.2 Conceptos asociados

1.2.1 Componente Gráfico

Componente informático: Elemento de un sistema que ofrece un conjunto de servicios, o funcionalidades, a través de interfaces definidas (2).

Gráfico en informática: Se denomina gráfico a una representación visual de una serie de datos en un ordenador, que tiene como finalidad dar una rápida noción de volúmenes y cantidades estableciendo comparaciones (3).

Por lo antes expuesto se puede afirmar que un **Componente Gráfico es:**

Elemento de un sistema que ofrece un conjunto de funcionalidades que son representadas en un ordenador.

1.2.2 Despliegue

Despliegue es un resumen esquematizado con la ventaja de permitir visualizar la estructura y organización de los elementos que componen el proceso, así como el comportamiento de los mismos, los cuales pueden estar ubicados en un área geográficamente extensa para que el operador monitoree constantemente cada uno de los componentes que conforman el proceso (4).

1.3 Definición de SCADA

SCADA, (Supervisión, Control y Adquisición de Datos) son aplicaciones de software de control de producción, que se comunican con los dispositivos de campo y controlan el proceso de forma automática desde la pantalla de un ordenador u otra tecnología de comunicación (5).

Los sistemas SCADA son partes integrales de la mayoría de los ambientes industriales complejos o muy geográficamente dispersos porque pueden recoger la información de una gran cantidad de fuentes rápidamente, y la presentan a un operador en una forma amigable. Además, mejoran la eficacia del proceso de monitoreo y control, proporcionando la información oportuna para poder tomar decisiones operacionales apropiadas (5).

1.3.1 Funciones principales que debe cumplir un SCADA

Adquisición de datos: Significa que el sistema tiene la capacidad de obtener información desde el módulo de control, por ejemplo, sobre valores de temperatura o presión, donde los datos son registrados para su posterior explotación (6).

Supervisión: Expresa el poder de observar o monitorear aquello que sucede en el proceso industrial, el equipo o la maquinaria, por ejemplo, conocer si un motor se encuentra encendido o apagado (7).

Control: Brinda la posibilidad de ejecutar comandos, es decir, poder enviar instrucciones hacia el sistema (8).

Mediante las herramientas de visualización y control el usuario tiene acceso al procesador digital, generalmente en un ordenador donde reside la aplicación de supervisión y control. La comunicación entre estos dos sistemas se suele realizar a través de redes de comunicación corporativas. El procesador digital capta el estado del sistema e informa al usuario a través de la herramienta HMI, basándose en los comandos ejecutados por el mismo, de esta manera el procesador digital inicia las acciones pertinentes para mantener control sobre el sistema (5).

1.3.2 Módulos del SCADA SAINUX

En el CEDIN, con el objetivo de buscar soluciones a problemas que se presentan en el proceso de automatización industrial, se encuentra en proceso de desarrollo el proyecto SCADA SAINUX el cual está formado por módulos como:

- ✓ **Comunicación:** Este módulo representa la capa de software encargada de la comunicación entre los diferentes procesos distribuidos, de mediano y alto nivel (9).
- ✓ **Configuración:** Módulo encargado de almacenar, persistir y suministrar, la información base para el funcionamiento de los demás módulos (10).
- ✓ **Seguridad:** Permite a los usuarios autenticarse ⁴ en el sistema, y de esta forma poder acceder sólo a los recursos que tiene asignado su rol⁵. Posee herramientas para la protección ante ataques piratas, fallos eléctricos y problemas de red (11).
- ✓ **Bases de Datos Históricas:** Es el que permite almacenar la sucesión de alarmas y eventos así como la información recibida desde los dispositivos que se encuentran en el campo. Esta información es de vital importancia para realizar cualquier tipo de análisis posterior como diagnósticos o reportes (12).
- ✓ **Adquisición o Procesamiento de Datos:** Supervisa y almacena la información del resto de las subestaciones, las cuales pueden ser ordenadores conectados a los dispositivos de campo o directamente sobre dichos dispositivos, algunas de las funciones que realiza son (13):
 - Actuar como interfaz al operador, incluyendo la presentación de información de variables en tiempo real, la administración de alarmas y la recolección y presentación de la información presente (13).
 - Puede ejecutar software especializado que cumple funciones específicas asociadas al proceso supervisado por el SCADA (13).
- ✓ **HMI:** Representa gráficamente los procesos con los que interactúa el usuario. A través de este módulo se pueden visualizar condiciones, valores de variables y alarmas.

⁴ Autenticarse: Acción que permite verificar la identidad digital de un usuario.

⁵ Rol: Responsabilidad asumida por una persona en el equipo de trabajo.

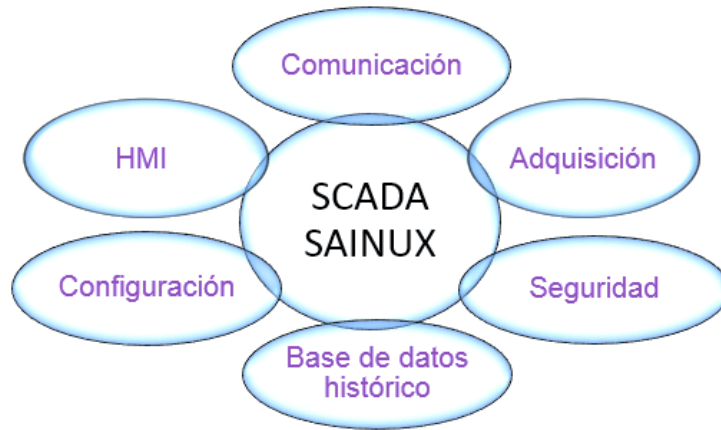


Fig. 1 Módulos del SCADA SAINUX.

1.3.3 Módulo Interfaz Hombre – Máquina

El módulo HMI es el encargado de representar, en un ordenador, los procesos que ocurren en el campo. A su vez muestra los componentes implicados, los sensores, las estaciones remotas y el sistema de comunicación, permitiendo que el operador posea el control sobre los procesos que intervienen en la industria. Este módulo logra que el operario esté en contacto directo con el sistema para así realizar la supervisión del proceso en general. Está compuesto por dos partes fundamentales:

- ✓ **EC o Editor:** Permite al mantenedor configurar la información específica del área que se desea supervisar y diseñar los despliegues, los cuales, haciendo uso de los componentes gráficos, permiten simular todos los procesos del campo.
- ✓ **EV o Visualizador:** Es donde el operador puede supervisar y controlar toda la información que fue configurada en el EC, interactuando con los componentes gráficos para asegurar el control sobre el sistema, monitorear los cambios de estado de las variables, gestionar alarmas y generar reportes.

1.4 Componentes Gráficos del HMI SAINUX

El SCADA SAINUX consta de una biblioteca de componentes gráficos (CG) la cual se encarga de recrear todos los procesos que ocurren en el campo del modo más real posible. Estos componentes se agrupan en:

- ✓ **Básicos:** Formado por figuras simples como líneas, flechas, elipses, rectángulos y polígonos.

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

1.5 Red de tubería

El hombre con el paso de los años ha perfeccionado sus conocimientos y técnicas logrando así el desarrollo de componentes que tienen como objetivo satisfacer sus necesidades. Entre los componentes desarrollados se encuentran las redes de tuberías, las cuales constituyen un sistema para el transporte de líquidos y gases.

Los sistemas de tuberías están formados por tramos de tuberías y aditamentos que se alimentan fluido arriba por un depósito o una bomba y descargan fluido abajo libremente a la atmósfera o a otro depósito (14).

1.5.1 Sistemas de representación de redes de tubería

Los procesos industriales encargados del transporte de líquidos, gases o cables para el transporte de fluido eléctrico, generalmente se representan en los planos técnicos por medio de trazos que forman las líneas de tubería y por símbolos que pueden representar toda clase de accesorios, como motobombas, compresores y válvulas (15).

El diseño de una red de tubería puede ser en proyección isométrica⁶ u ortogonal⁷, en caso de vistas ortogonales estas se deben presentar múltiples para determinar las dimensiones de los

⁶ Isométrica: Es una de las formas de proyección utilizadas en un dibujo técnico que permite la representación a escala.

⁷ Ortogonal: Generalización de la noción geométrica de perpendicularidad.

tramos de tubería y la ubicación de los accesorios o componentes del sistema. Lo anterior implica que sea más empleada en muchos casos la proyección isométrica (15).

En el plano de una red de tubería se pueden emplear **dos sistemas de representación:**

- ✓ **Sistema de trazado a escala (trazo a línea doble o real):** Empleado principalmente para tubos grandes, generalmente con bridas, como las obras de calderas y de centrales o plantas eléctricas, en que las longitudes son críticas y especialmente cuando el tubo no se corta y ajusta a la obra (15).
- ✓ **Sistema esquemático (trazo de línea simple o simplificada):** Se emplean en los dibujos que se hacen a escala pequeña, como los planos arquitectónicos y los de distribución en planta. Cuando los detalles no son relevantes se suelen simplificar los planos con símbolos a trazo simple, pero que representan de igual forma los accesorios y componentes. Los accesorios se indican por medio de símbolos y los tramos de tubería se muestran por una sola línea (15).

1.5.2 Componentes de una red de tubería

Para la representación de una red de tubería son empleados símbolos que persiguen el objetivo de representar de forma real todos los componentes o accesorios de dicho sistema, lo cual en algunos casos resulta muy complejo. Para lograr una mejor organización de estos símbolos se decide agruparlos de la siguiente manera:

- ✓ **Componentes ramificadores:** Formado por elementos que permiten representar una tubería compuesta por más de dos direcciones como: bridas, tubería de tipo TE, cruz y Y.
- ✓ **Componentes de control:** Conformado por las válvulas y bombas, las cuales brindan la posibilidad de controlar el flujo de líquidos y gases en la red.
- ✓ **Componentes de dirección:** Compuesto por símbolos como tubería de tipo codo de 90 grados, tubería de tipo codo de 45 grados y tubería de tipo línea, dichos símbolos permiten al operador representar tuberías con dos direcciones en la red.

- ✓ **Componentes de monitoreo:** Incluye los elementos encargados de medir e indicar el paso de flujo durante el proceso, llamado flujómetro⁸ los cuales se agrupan en analógicos y digitales.

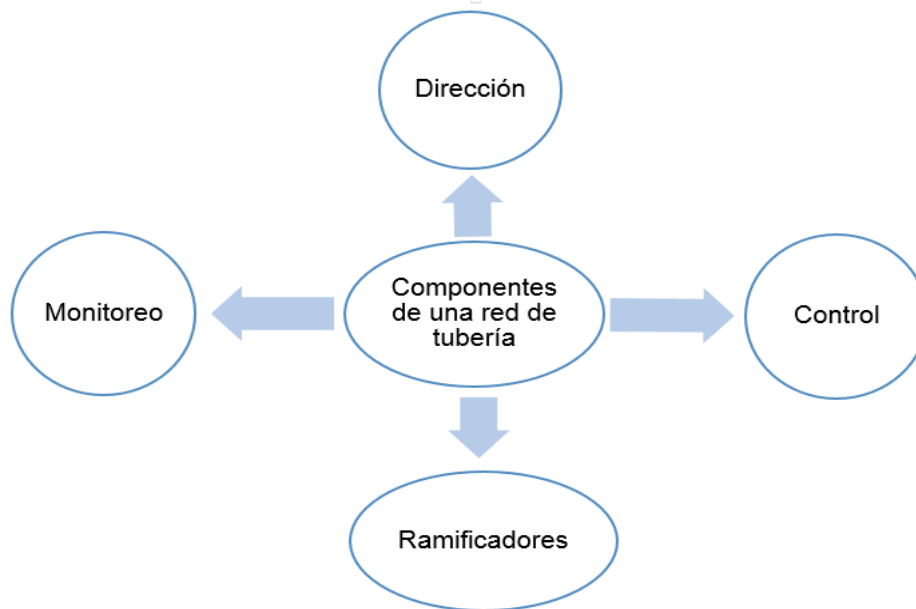


Fig. 2. Componentes de una red de tubería.

1.5.2.1 Accesorios de una red de tubería.

- ✓ **Bridas:** Representan los accesorios que permiten conectar tuberías con otros equipos. La unión se realiza por medio de dos bridas, en la cual una de ellas pertenece a la tubería y la otra al equipo a ser conectado. Este proceso permite el rápido montaje y desmontaje cuando se desea realizar reparaciones o mantenimiento.
- ✓ **Codos:** Son accesorios en forma de curva que se utilizan para cambiar la dirección del flujo que pasa por la tubería.
 - **Codo45:** Representa un codo con un ángulo de 45 grados.
 - **Codo90:** Simboliza un codo con un ángulo de 90 grados.
- ✓ **Tubería T:** Componentes que se fabrican de diferentes tipos de materiales, aleaciones y diámetros, utilizados para efectuar fabricación en líneas de tubería.

⁸ Flujómetro: Instrumento de medición que indica el flujo que pasa por la tubería.

- ✓ **Tubería Cruz:** Encargado de representar una tubería en forma de cruz, permitiendo distribuir el flujo en cuatro direcciones distintas.
- ✓ **Tubería U:** Son componentes que representan una tubería en forma de U, consiguiendo que el flujo que pasa por la tubería tome dos orientaciones diferentes.
- ✓ **Tubería Y:** Simboliza una tubería de tipo Y, la cual es capaz de lograr que el fluido circule en tres trayectorias disímiles.
- ✓ **Tubería Reducida:** Elementos de forma cónica que permiten disminuir el volumen del fluido a través de las líneas de tubería.
- ✓ **Flujómetro:** Accesorios encargados de medir e indicar el paso de flujo por la tubería durante el proceso.
 - **Flujómetro Analógico:** Representa un reloj analógico.
 - **Flujómetro Digital:** Simboliza un reloj digital.

1.6 Metodología de desarrollo de software

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de procesos genéricos como cascada, evolutivo, incremental y espiral. Adicionalmente una metodología debería definir con precisión las actividades, artefactos, roles involucrados, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, prácticas y técnicas recomendadas. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a las actividades del proceso de desarrollo, en ocasiones suele hablarse de métodos de análisis y diseño (16).

1.6.1 Metodología AUP-UCI.

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

Para eliminar los errores encontrados se decidió converger a una metodología que cubra las particularidades de cada una de las ya aplicadas. Esta metodología escogida se adapta a lo que

ya la Universidad ha estado proponiendo como ciclo de vida de los proyectos, sin alejarse de lo que hasta el momento se ha trabajado e introducir la menor cantidad de cambios posibles. Esta metodología propuesta es una variación al Proceso Unificado Ágil (17).

El Proceso Unificado Ágil (AUP, por sus siglas en inglés) es una versión simplificada del Proceso Racional Unificado (RUP, por sus iniciales en inglés). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (17).

Al igual que en RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva. Estas fases son: (17)

- ✓ **Inicio:** El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
- ✓ **Elaboración:** El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
- ✓ **Construcción:** Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
- ✓ **Transición:** El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

La variación de la metodología AUP, denominada AUP-UCI se adapta al ciclo de vida definido para la actividad productiva de la UCI y es utilizada por el CEDIN en sus productos de software. El uso de esta técnica de modelado ágil permite encapsular los requisitos funcionales en Historias de Usuario (HU) o descripción de requisitos por procesos (17).

1.6.2 Fases de AUP-UCI.

Tabla 1. Fases de AUP-UCI (17).

Fases	Objetivos
-------	-----------

Inicio	<ul style="list-style-type: none"> ✓ Planeación del proyecto. ✓ Estudio inicial de la organización cliente. ✓ Alcance del proyecto. ✓ Estimaciones de tiempo, esfuerzo y costo.
Ejecución	<ul style="list-style-type: none"> ✓ Ejecución de las actividades requeridas para desarrollar el software. ✓ Ajuste de los planes del proyecto. ✓ Modelado del negocio. ✓ Obtención de requisitos. ✓ Se elaboran la arquitectura y el diseño. ✓ Se implementa y se libera el producto.
Cierre	<ul style="list-style-type: none"> ✓ Resultados obtenidos. ✓ Cierre del proyecto.

1.6.3 Disciplinas de AUP-UCI.

Tabla 2. Disciplinas de AUP-UCI (17).

Disciplinas	Objetivos
Modelado de negocio (opcional)	Comprender los procesos de negocio.
Requisitos	Administración y gestión de los requisitos funcionales y no funcionales.
Análisis y diseño	Modelar el sistema.
Implementación	Construcción del sistema.
Pruebas internas	Verificar el resultado de la implementación.

Pruebas de liberación	Diseñar y ejecutar pruebas por una entidad externa certificadora de la calidad.
Pruebas de aceptación	Verificar que el software está listo.
Despliegue (Opcional)	Instalación, configuración, adecuación, y puesta en marcha.
Gestión y soporte	Conjunto de operaciones que se realizan para dirigir y administrar una empresa con el objetivo de sostener o mantener el negocio.

1.7 Herramientas y tecnologías

Para dar solución al problema planteado se utilizaron herramientas y tecnologías, que serán definidas a continuación, destacando las características fundamentales que las hacen relevantes ante otras de su tipo. En ocasiones la selección de dichas herramientas y tecnologías está fundamentada por su utilización en el proyecto SCADA SAINUX.

1.7.1 Gráficos Vectoriales Escalables (SVG)

Los gráficos vectoriales escalables (SVG, por sus siglas en inglés) son un lenguaje abierto que permite crear gráficos vectoriales basados en XML tanto estáticos como animados. A pesar de ser un lenguaje vectorial permite crear imágenes complejas, lo cual garantiza su selección para el diseño de componentes (18).

SVG es un formato vectorial capaz de ser interpretado por los navegadores web, lo cual permitirá tomar aquellos gráficos SVG que cuenten con licencia de uso libre presentes en el internet y usarlos en los diseños (18).

La utilización del SVG en su versión 1.1 se debe a que es una especificación para generar imágenes en Lenguaje de Marcas Extensible (XML, por sus siglas en inglés) y por lo tanto es fácilmente editable con un editor de texto como Notepad y Gedit sin necesidad de una herramienta profesional (18).

1.7.2 Editor Gráfico

El uso de Inkscape en su versión 0.48 se debe a que su objetivo principal es crear una herramienta de dibujo potente y cómoda, totalmente compatible con los estándares XML, SVG y hojas de estilo en cascada (CSS, por sus siglas en inglés). Las características soportadas incluyen: formas, trazos, texto, marcadores, transformaciones y gradientes (19).

InkScape es un editor de gráficos vectoriales de código abierto que usa el formato de archivo SVG. Este editor es una herramienta multiplataforma que se encuentra desarrollada principalmente para el sistema operativo GNU/Linux (19).

1.7.3 Lenguaje de programación

Todo producto informático necesita un lenguaje de programación para su desarrollo, el cual se comunica con la computadora y orienta a la misma para que realice una tarea específica.

El lenguaje de programación seleccionado es C++ versión 98, con el objetivo de lograr una mayor compatibilidad con el SCADA SAINUX pues es el lenguaje en el que está desarrollado. C++ está considerado por muchos como un lenguaje de programación potente, debido a que permite el trabajo tanto a alto como a bajo nivel, logrando gran eficiencia en tiempo de ejecución y bajo consumo de memoria en los programas desarrollados, algo que requieren la mayoría de las aplicaciones (20).

1.7.4 Marco de trabajo

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

El framework utilizado es Qt en su versión 4.8.2, el cual es multiplataforma para el desarrollo de aplicaciones, las cuales pueden ser con o sin interfaz gráfica (21). Para el desarrollo del presente trabajo se hace uso de este framework debido a que utiliza el lenguaje de programación C++ de forma nativa y es precisa para realizar todas las interfaces gráficas necesarias.

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

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

1.7.5 Entorno de Desarrollo Integrado (IDE)

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

El IDE seleccionado para el desarrollo de la aplicación es Qt Creator en su versión 2.5.0. Qt Creator es un IDE multiplataforma para el desarrollo de aplicaciones que pueden o no tener interfaz gráfica. Este se centra en proporcionar características que ayudan a los nuevos usuarios del IDE a aprender y comenzar a desarrollar rápidamente (22).

Existen otras características importantes que presenta este IDE como la disponibilidad de código fuente, la excelente documentación organizada que provee en el QtAssistant y un editor para el diseño de formularios denominado QtDesigner (22).

Qt Creator cuenta con:

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

1.7.6 Lenguaje Unificado de Modelado (UML)

Para el desarrollo del presente trabajo se necesita de un lenguaje unificado de modelado (UML, por sus siglas en inglés), el cual es un lenguaje de modelado de sistemas de software muy conocido y manejado en la actualidad, utilizado para especificar o describir métodos y procesos.

UML se utiliza en su versión 2.0, para visualizar, especificar, construir y documentar el sistema. Ofrece un estándar para describir un modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de

lenguajes de programación, esquemas de bases de datos y componentes reutilizables. El lenguaje de modelado es una notación gráfica que indica los pasos que se deben seguir para confeccionar un diseño (23).

1.7.7 Visual Paradigm

La herramienta seleccionada para realizar el modelado de la solución es el Visual Paradigm en su versión 8.0. Visual Paradigm es una herramienta de Ingeniería de Software Asistida por Computación (CASE, por sus siglas en inglés). La misma propicia un conjunto de funcionalidades para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. (24).

1.7.8 Sistema Operativo

La distribución seleccionada es Debian, específicamente la versión 7 o Wheezy. Debian o Proyecto Debian es una comunidad conformada por desarrolladores y usuarios, que mantiene un sistema operativo GNU basado en software libre. El sistema se encuentra precompilado, empaquetado y en un formato deb para múltiples arquitecturas de computadoras y para varios núcleos. El modelo de desarrollo del proyecto es ajeno a motivos empresariales o comerciales, por lo que es llevado adelante por los usuarios, aunque cuenta con el apoyo de varias empresas en forma de infraestructuras. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuirlo comercialmente mientras se respete su licencia (25).

Una distribución de Linux es una variante de ese sistema operativo que incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios. Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU (GNU No es Unix); de ahí el nombre: GNU/Linux.

1.7.9 Herramienta de control de versiones

Se utiliza Apache Subversion (SVN) en su versión 1.6. SVN es una herramienta de control de versiones basada en un repositorio cuyo funcionamiento se asemeja al de un sistema de ficheros (26).

Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones sólo guarda el conjunto de modificaciones, optimizando así al máximo el uso de espacio en disco. SVN permite al usuario crear, copiar y borrar carpetas con la misma facilidad

con la que lo haría si estuviese en su disco duro local. Dada su flexibilidad, es necesaria la aplicación de buenas prácticas para llevar a cabo una correcta gestión de las versiones del software generado (26).

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Debido a que la investigación se encuentra bajo el control de versiones, no existe razón para que la calidad de la misma sea afectada. Esta herramienta permite recuperar versiones antiguas y ver el historial de cambios de un sistema de archivos (26).

1.8 Conclusiones parciales

Como resultado de la fundamentación teórica de la presente investigación, luego de realizar un análisis de las principales características de los sistemas SCADA y de los componentes gráficos del sistema SCADA SAINUX, se puede afirmar que el estudio de las redes de tubería existentes proporcionó la obtención de un profundo conocimiento acerca de los accesorios que la componen, lo cual es de vital importancia para el desarrollo del tema de la investigación. De esta manera la selección de las principales herramientas, tecnologías y metodología permitió sentar las bases para el desarrollo de la solución, entre ellas se encuentran el SVG y el InkScape utilizados para el diseño de los componentes gráficos, C++ como lenguaje de programación empleado en el desarrollo del producto, Qt, marco de trabajo utilizado para la construcción de las interfaces gráficas, QtCreator, entorno de desarrollo integrado usado para la implementación de los componentes, lenguaje de modelado UML, el cual fue de vital importancia para modelar los diagramas del sistema, Visual Paradigm utilizado para facilitar el análisis y diseño de los diagramas empleados en la solución y la metodología de desarrollo de software seleccionada, AUP-UCI es una técnica ágil que permitió estructurar, planificar y controlar el proceso de desarrollo en el sistema.

Capítulo 2. Análisis y diseño de la solución.

2.1 Introducción

El presente capítulo tiene como objetivo describir las actividades realizadas durante el proceso de análisis y diseño, realizadas para modelar y dar respuesta al problema planteado. Se especifican los requisitos funcionales y no funcionales que debe cumplir el sistema para satisfacer las necesidades del cliente. Al mismo tiempo se muestra la definición de las historias de usuario que regirán el desarrollo de la solución propuesta, se presenta el diseño de la arquitectura y se realiza la descripción del diseño.

2.2 Fase de inicio

Durante esta fase inicial se efectuó un análisis y estudio de la organización del negocio, donde se identificaron las necesidades del proyecto y se obtuvo toda la información necesaria acerca del alcance del mismo, para decidir si se ejecuta o no, siendo positivo el resultado final.

2.3 Modelo de dominio

El modelo de dominio permite capturar y expresar el conocimiento adquirido durante el análisis del área que se encuentra bajo investigación, como paso previo al diseño del sistema. Dicho modelo, está formado por el diagrama de las entidades que conforman el negocio así como las desarrolladas para los componentes gráficos que serán usados en la representación de las redes de tuberías, así como la relación existente entre ellas (27).

El modelo está formado por entidades como:

- ✓ **HMI:** Interfaz encargada de mostrar en un ordenador los procesos que ocurren en el campo.
- ✓ **Componente gráfico:** Representa de manera visual una serie de datos por medio de figuras o signos
- ✓ **Medición:** Entidad encargada de medir el fluido que pasa por la red de tubería.
- ✓ **Medición digital:** Su objetivo es representar mediante un reloj digital la medición de flujo presente en la red.
- ✓ **Medición analógica:** Encargada de representar mediante un reloj Analógico la medición de flujo presente en la red.

- ✓ **Medición calculada:** Entidad que representa las diferentes variables que maneja el SCADA que se clasifican según el valor que toma la función que le fue asignada.
- ✓ **Medición de entrada y salida:** Representa las variables que maneja el SCADA y toma los valores directo de los dispositivos.
- ✓ **Tubería modelo:** Entidad Modelo donde se guarda la configuración que tendrá la tubería y brinda la información necesaria para que un componente gráfico pueda visualizarse.
- ✓ **Tubería:** Entidad base de la cual heredan cada uno de los componentes que serán visualizados en la red de tubería.
- ✓ **Tubería lineal:** Encargada de representar una tubería en forma de línea recta dentro de la red.
- ✓ **Brida:** Permite simbolizar las bridas en una red de tuberías.
- ✓ **Tubería de tipo T:** Entidad encargada de representar la tubería en forma de T dentro de una red de tuberías.
- ✓ **Tubería de tipo cruz:** Permite representar la tubería en forma de cruz dentro de una red de tuberías.
- ✓ **Tubería lateral:** Simboliza una tubería en forma de Y en la red de tuberías.
- ✓ **Tubería Genérica:** Representa un componente que se puede comportar como cualquier otro definido.
- ✓ **Tubería reductora excéntrica:** Componente gráfico que representa un accesorio de tubería que permite reducir gradualmente el diámetro de la red.
- ✓ **Tubería de tipo codo de 90 grados:** Entidad encargada de representar la tubería en forma de codo de 90 grados dentro de una red de tuberías.
- ✓ **Tubería de tipo codo de 45 grados:** Clase que permite representar la tubería en forma de codo de 45 grados dentro de una red de tuberías.
- ✓ **Tubería de tipo U:** Consigue agregar la tubería en forma de U dentro de la red de tuberías.
- ✓ **Tubería Reductora:** Simboliza una tubería reductora para ser usada en la representación de la red.
- ✓ **Modelo de flujómetro:** Entidad modelo que almacena toda la configuración de los medidores de flujo y ofrece la información necesaria para que estos se visualicen.
- ✓ **Flujómetro:** Entidad de la cual heredan los componentes medidores de flujo que serán visualizados en la red.

- ✓ **Flujómetro digital:** Representa un componente encargado de medir en forma digital, el fluido que pasa por la red.
- ✓ **Flujómetro Analógico:** Incorpora un componente que permite medir el flujo que pasa por la red, en forma analógica.

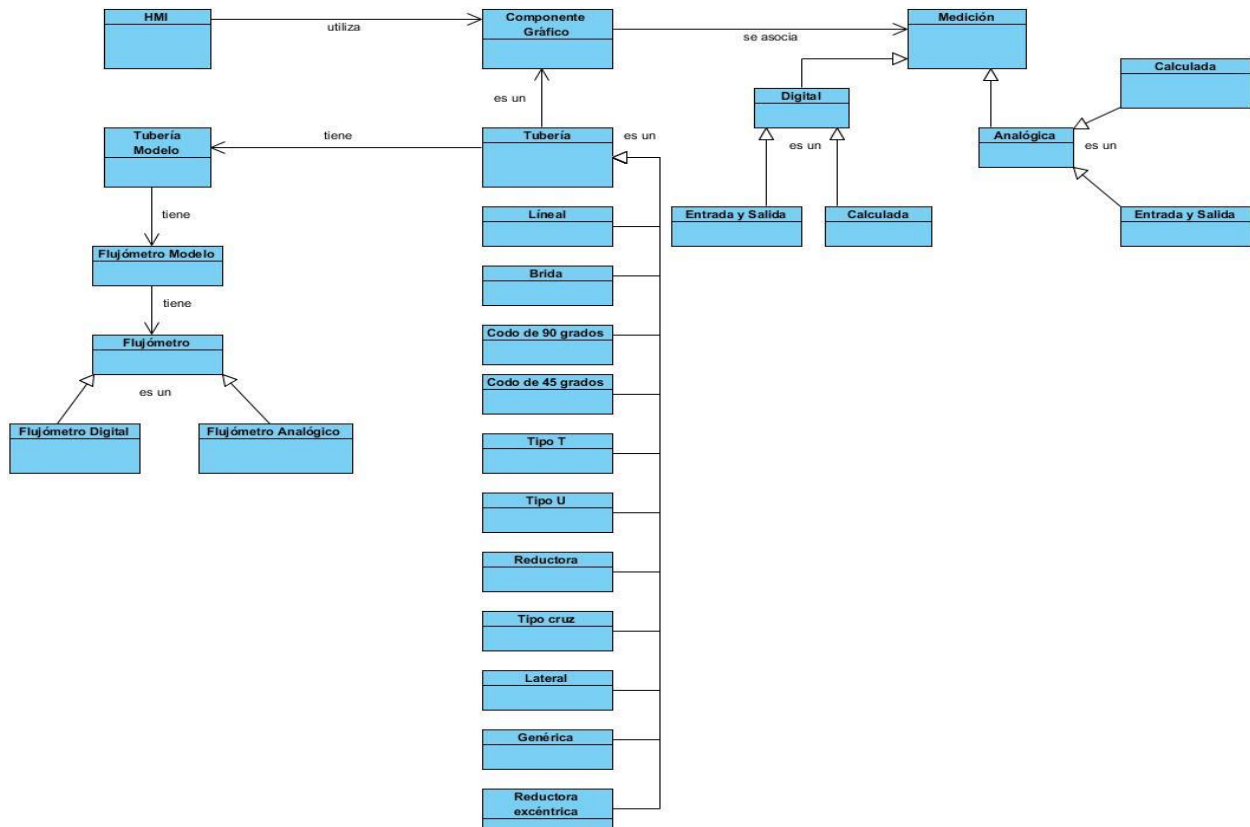


Fig. 3 Modelo de Dominio para la red de tuberías.

2.4 Fase de ejecución

Esta fase es la encargada de ejecutar las actividades requeridas para el desarrollo de la investigación, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el sistema, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto, el cual es entregado al usuario final, siendo este capacitado para la utilización del producto.

2.5 Captura de requisitos

Un requisito es una condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado. También se aplica a las condiciones que debe cumplir o poseer un sistema o uno de sus componentes para satisfacer un contrato, una norma o una especificación (28). Pueden ser funcionales o no funcionales.

2.4.1 Requisitos funcionales (RF)

Los requisitos funcionales son una definición de los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar antes situaciones particulares (28).

RF 1- Definir la propiedad nombre del accesorio de la tubería.

RF 2- Definir descripción del componente.

RF 3- Asociar medición al componente.

RF 4- Configurar opacidad del componente.

RF 5- Configurar la dimensión del componente.

RF 6- Configurar la propiedad rotación del accesorio de la tubería.

RF 7- Configurar la propiedad posición del accesorio de la tubería.

RF 8- Asociar animaciones a los accesorios de la tubería.

RF 9- Cambiar el color de las bridas pertenecientes al componente.

RF 10- Cambiar el color de la tubería perteneciente al componente.

RF 11- Representar en tiempo real el estado de la medición del componente.

RF 12- Mostrar / ocultar bridas pertenecientes al componente.

Requisitos Funcionales específicos para los medidores de flujo.

RF 13- Cambiar el color de la aguja perteneciente al medidor de flujo analógico.

RF 14- Cambiar el color del valor que representa la parte entera, perteneciente al medidor de flujo digital.

RF 15- Cambiar el color del valor que representa la parte decimal, perteneciente al medidor de flujo digital.

RF 16- Cambiar el color del anillo perteneciente al componente.

RF 17- Cambiar el color del plato perteneciente al componente.

RF 18- Mostrar / ocultar escala del medidor analógico.

RF 19- Configurar la posición del valor del flujómetro.

RF 20- Configurar el tamaño de fuente del valor del flujómetro.

RF 21- Configurar el color de la unidad de medida.

RF 22- Configurar la posición de la unidad de medida.

RF 23- Configurar el tamaño de fuente de la unidad de ingeniería.

RF 24- Configurar dimensiones del flujómetro emergente.

RF 25- Configurar el color del valor del flujómetro.

RF 26- Configurar el tamaño de fuente de la escala.

RF 27- Mostrar /ocultar la escala numérica.

RF 28- Mostrar /ocultar el valor del flujómetro.

RF 29- Cambiar el color de la escala perteneciente al medidor analógico.

RF 30- Cambiar el color del texto que indica el valor de la medición.

RF 31- Cambiar el color del texto que indica la unidad de ingeniería.

RF 32- Mostrar / ocultar el texto con el valor de la medición.

RF 33- Mostrar / ocultar el texto con el valor que indica la unidad de ingeniería.

2.4.2 Requisitos no funcionales (RNF)

Son restricciones que afectan a los servicios o funciones del sistema, tales como restricciones de tiempo, sobre el proceso de desarrollo, estándares. Definen propiedades emergentes del sistema, tales como el tiempo de respuesta, la fiabilidad, entre otras (28).

Requerimientos de software

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

Requerimientos de Hardware

Debe ser ejecutado en computadoras que tengan como requerimientos los siguientes:

- ✓ Microprocesador: dualcore a 1.5 GHz.
- ✓ RAM: 2 GB

Restricciones en el diseño y la implementación

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

Requerimientos de usabilidad

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

Rendimiento

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

2.4.3 Historias de Usuario

Los modelos ágiles utilizan historias de usuario para captar las necesidades de los clientes en un proyecto de software. Una historia de usuario, es una descripción en primera persona y de alto nivel de una acción que usuario efectúa en un sistema (29).

Tabla 3 Historia de Usuario 8

Historia de Usuario	
Número: 8	Nombre del requisito: Permite asociar animaciones a los accesorios de la tubería.
Programador: Rosmery Pedraza Ceballo.	Iteración Asignada: 2.
Prioridad: Alta.	Tiempo Estimado: 1 semana.

Riesgo en Desarrollo: Riesgo definido en el GESPRO. **Tiempo Real:** 1 semana.

Descripción: Posibilita al operador el cambio de algunas de las propiedades configurables del componente con el uso de animaciones.

El operador debe hacer clic derecho sobre el componente, seleccionar la opción Propiedades, de esta forma se muestra el inspector de propiedades donde elige la opción animaciones y asocia la animación al componente.

Las restantes historias de usuario se encuentran en el Anexo 1.

2.5 Planificación del desarrollo.

Una vez definidas las historias de usuario (HU) se puntualiza la prioridad de cada una de ellas y la estimación de esfuerzo necesario para realizarlas para lo que se llevó a cabo un plan de ocho iteraciones las cuales son mostradas a continuación.

2.5.1 Estimación de esfuerzos e iteraciones por historia de usuario.

La siguiente tabla muestra estimación de esfuerzo para cada una de las historias de usuarios definidas en el desarrollo de la solución propuesta, el plan de iteraciones realizado y la duración total de las mismas.

Tabla 4 Estimación de esfuerzos.

Historias de Usuario	Puntos de Estimación	Iteración	Duración total de las iteraciones (semanas)
Definir la propiedad nombre del accesorio de la tubería.	0.2 semanas	1	3
Definir descripción del componente.	0.4 semanas		
Asociar medición al componente.	0.5 semanas		
Configurar opacidad del componente.	0.6 semanas		

Configurar la dimensión del componente.	0.8 semanas		
Configurar la propiedad rotación del accesorio de la tubería.	0.5 semanas		
Configurar la propiedad posición del accesorio de la tubería.	0.3 semanas	2	3
Asociar animaciones a los accesorios de la tubería.	1 semana		
Cambiar el color de las bridas pertenecientes al componente.	0.2 semanas		
Cambiar el color de la tubería perteneciente al componente.	0.3 semanas		
Representar en tiempo real el estado de la medición del componente.	1 semana		
Mostrar / ocultar bridas pertenecientes al componente.	0.2 semanas		
Cambiar el color de la aguja perteneciente al medidor de flujo analógico.	0.6 semanas	3	3
Cambiar el color del valor que representa la parte entera, perteneciente al medidor de flujo digital.	0.6 semanas		

Cambiar el color del valor que representa la parte decimal, perteneciente al medidor de flujo digital.	0.6 semanas		
Cambiar el color del anillo perteneciente al componente.	0.6 semanas		
Cambiar el color del plato perteneciente al componente.	0.6 semanas		
Mostrar / ocultar escala del medidor analógico.	1 semana	4	3
Configurar la posición del valor del flujómetro.	1 semana		
Configurar el tamaño de fuente del valor del flujómetro.	1 semana		
Configurar el color de la unidad de medida.	0.6 semanas	5	3
Configurar la posición de la unidad de medida.	1 semana		
Configurar el tamaño de fuente de la unidad de ingeniería.	1 semana		
Configurar dimensiones del flujómetro emergente.	1 semana		
Configurar el color del valor del flujómetro.	1 semana		

Configurar el tamaño de fuente de la escala.	1 semana	6	3
Mostrar /ocultar la escala numérica.	1 semana		
Mostrar /ocultar el valor del flujómetro.	1 semana	7	3
Cambiar el color de la escala perteneciente al medidor analógico.	0.6 semanas		
Cambiar el color del texto que indica el valor de la medición.	0.8 semana		
Cambiar el color del texto que indica la unidad de ingeniería.	0.6 semanas		
Mostrar / ocultar el texto con el valor de la medición.	1 semana	8	2
Mostrar / ocultar el texto con el valor que indica la unidad de ingeniería.	1 semana		

2.5.2 Plan de desarrollo

Luego de definidas las Historias de Usuarios (HU) y estimado el esfuerzo propuesto para su realización, se realizó el sistema en ocho iteraciones, las cuales se describen detalladamente a continuación:

Iteración I: Iteración que tiene como objetivo realizar las HU 1, 2, 3, 4, 5 y 6 las cuales se encargan de configurar las propiedades del accesorio seleccionado.

Iteración II: Esta iteración tiene como objetivo realizar las HU 7, 8, 9, 10, 11 y 12 siendo estas las encargadas de añadir animaciones al accesorio de la red de tubería.

Iteración III: Iteración tiene como objetivo realizar las HU 13, 14, 15, 16 y 17 las cuales son las referentes a cambiar el color de los elementos que componen el medidor de flujo.

Iteración IV: La presente iteración tiene como objetivo realizar las HU 18, 19 y 20 siendo estas las encargadas de configurar la posición y el tamaño de los elementos que componen el medidor de flujo.

Iteración V, VI, VII y VIII: Son las iteraciones conformadas por las HU 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32 y 33 las cuales tienen como objetivo configurar las propiedades de los medidores de flujo digital y analógico.

2.6 Diagrama de paquetes

Este diagrama es el encargado de representar las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre ellas (29).

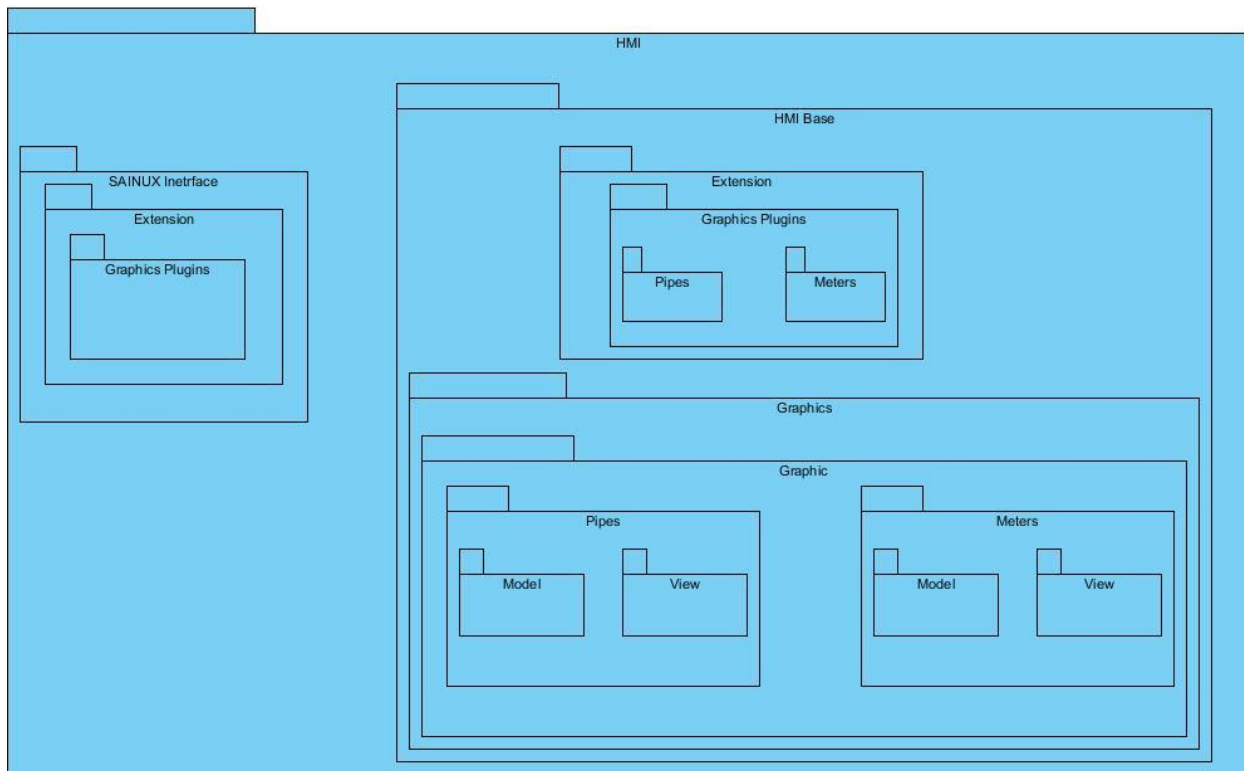


Fig. 4 Diagrama de Paquetes de Componentes Gráficos.

La solución propuesta esta agrupada en Componentes y Paletas de Componentes. La primera está formada por la carpeta SAINUX Interface, la cual encierra a la subcarpeta Graphics Plugins, que contiene las entidades referentes a las paletas de los componentes pertenecientes a la interfaz SAINUX. En la segunda parte están las Paletas de Componentes divididas en el paquete HMI donde se almacenan las propiedades comunes de cualquier módulo de un sistema SCADA, este módulo contiene los paquetes Extensión y dentro de él el subpaquete Graphics Plugins donde están situadas las entidades relacionadas con las paletas Pipe y Meters. Seguidamente el paquete está conformado por la carpeta Graphics, donde se guardan las entidades de la biblioteca de componentes gráficos y se representan los componentes Pipe y Meters, para lograr representar la Vista y el Modelo.

2.7 Diagrama de secuencia

Un diagrama de secuencia muestra una interacción, que representa la secuencia de mensajes entre instancias de clases, componentes, subsistemas o actores. El tiempo fluye por el diagrama y muestra el flujo de control de un participante a otro (30).

Los diagramas de secuencia en UML, son los encargados de representar la forma en que los objetos se comunican entre sí al transcurrir el tiempo. Esta representación está formada por: objetos con sus líneas de vida, mensajes intercambiados entre objetos en una secuencia ordenada y línea de vida activa (opcional). Además muestra los objetos que participan en la interacción y la secuencia de mensajes intercambiados (31).

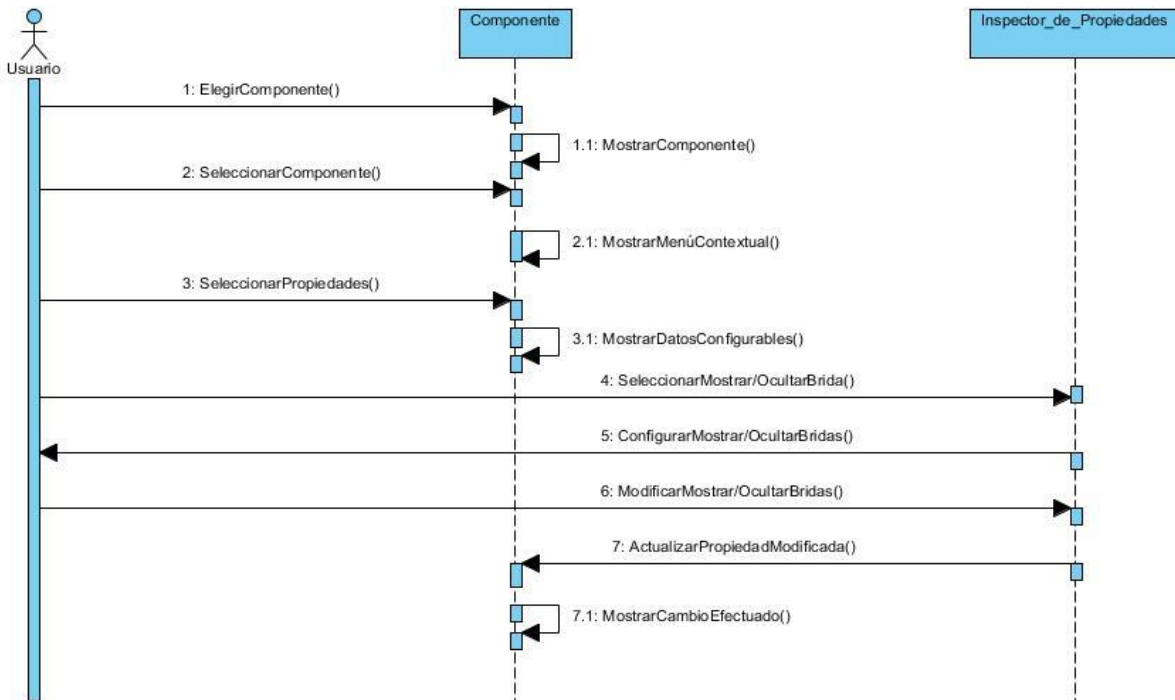


Fig. 5 Diagrama de secuencia- Mostrar/Ocultar bridas pertenecientes al componente.

Los restantes diagramas de secuencia se encuentran en el Anexo 3.

2.8 Diagrama de clases

El modelo de diseño juega un papel importante en el desarrollo de software lo cual permite producir varios modelos del sistema o producto que se va a construir, dicho modelo forma una especie de plan para la solución que será generada (32).

Los diagramas de clases representan un conjunto de interfaces, colaboraciones y sus relaciones. Gráficamente son una colección de nodos y arcos (33). Muestran una serie de clases, elementos y contenidos, representados a través de las relaciones entre ellos, conformando de esta manera la estructura el sistema.

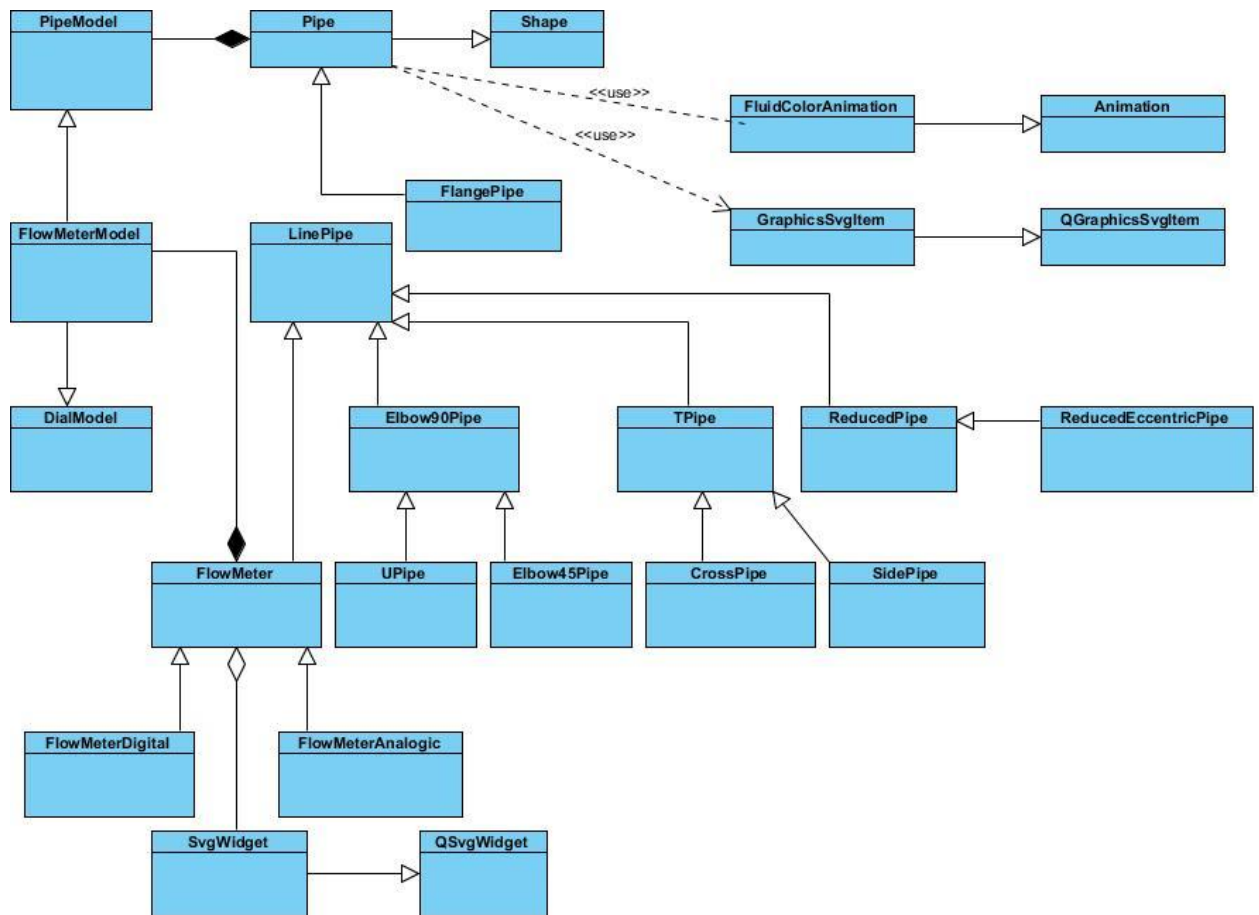


Fig. 6 Diagrama de Clases de Componentes Gráficos

En el Anexo 4 se encuentra una detallada explicación de este diagrama.

2.8.1 Descripción del diagrama de clases

- ✓ PipeModel: Entidad Modelo donde se guarda la configuración que tendrá la tubería.
- ✓ Pipe: Brinda la información necesaria para que un componente gráfico pueda visualizarse.
- ✓ GraphicsSVGObject: Entidad base para todos los componentes gráficos que utilizan en su representación SVG.
- ✓ FluidColorAnimation: Entidad encargada de realizar la animación de fluido.
- ✓ Animation: Entidad base de todas las clases encargadas de realizar alguna animación.
- ✓ GraphicsSvgItem: Clase que permite representar archivos en SVG.
- ✓ QGraphicsSvgItem: Entidad que provee el marco de trabajo para representar archivos en SVG.
- ✓ FlangePipe: Clase que permite simbolizar las bridas en una red de tuberías.

- ✓ FlowMeterModel: Entidad modelo que almacena toda la información de los medidores de flujo.
- ✓ LinePipe: Encargada de representar una tubería en forma de línea recta dentro de la red.
- ✓ ReducePipe: Simboliza una tubería reductora para ser usada en la representación de la red.
- ✓ ReduceEccentricPipe: Entidad encargada de representar un accesorio de tubería que permite reducir gradualmente el diámetro de la red
- ✓ TPipe: Entidad encargada de representar la tubería en forma de T dentro de una red de tuberías.
- ✓ SidePipe: Simboliza una tubería en forma de Y en la red de tuberías.
- ✓ CrossPipe: Permite representar la tubería en forma de cruz dentro de una red de tuberías.
- ✓ FlowMeter: Entidad que ofrece la información necesaria para que los medidores de flujo se visualicen
- ✓ Elbow90Pipe: Entidad encargada de representar la tubería en forma de codo de 90 grados dentro de una red de tuberías.
- ✓ Elbow45Pipe: Clase que permite representar la tubería en forma de codo de 45 grados dentro de una red de tuberías.
- ✓ UPipe: Permite representar una tubería en forma de U dentro de la red de tuberías.
- ✓ SvgWidget: Entidad encargada de representar un archivo SVG dentro de una forma visual.
- ✓ QSvgWidget: Clase que propone el marco de trabajo para representar un archivo SVG dentro de una forma visual.
- ✓ FlowMeterAnalogic: Entidad encargada de incorporar un componente que indica el paso de flujo por la red, en forma analógica.
- ✓ FlowMeterDigital: Entidad encargada de representar un componente encargado de indicar el de forma digital el paso de fluido por la red.
- ✓ DialModel: Clase modelo encargada de almacenar los datos referentes al dial.

2.9 Arquitectura del sistema

La arquitectura del software determina su estructura general y las formas en que proporciona una integridad conceptual para un sistema. En su forma más simple, la arquitectura es la organización de los componentes del programa, la manera en que estos interactúan y la estructura de datos que utilizan. En un sentido más amplio, los componentes pueden generalizarse para representar elementos importantes del sistema y sus interacciones (34).

2.9.1 Patrón de arquitectura de software

Para el desarrollo de la aplicación se propone utilizar el patrón de arquitectura de software Modelo-Vista de Qt.

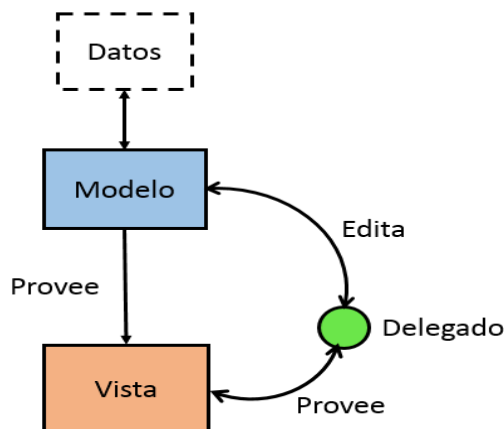


Fig. 7 Patrón de Arquitectura de Software Modelo-Vista.

El modelo se comunica con una fuente de datos, proporcionando una interfaz para los otros componentes en la arquitectura. La naturaleza de la comunicación depende del tipo de fuente de datos, y la forma en que se implementa el modelo. La vista obtiene índices de modelo; estos son referencias a objetos de datos. Mediante el suministro de los índices del modelo, la vista puede recuperar los elementos de datos del origen de datos (35).

En las vistas estándares, un delegado transforma los datos de manera que puedan ser interpretados de una mejor forma por los usuarios; cuando este objeto se edita, el delegado se comunica directamente con el modelo utilizando sus índices (35).

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 destinadas a ser una subclase con el fin de proporcionar el conjunto completo de funcionalidad esperada por otros componentes (35).

Entre las clases modelo utilizadas en la solución se encuentran PipeModel y FlowMeterModel,

del mismo modo las entidades UPipe y TPipe son entidades que representan la vista.

2.10 Patrones de diseño

Los patrones de diseño son un conjunto de prácticas de óptimo diseño que se utilizan para abordar problemas en la programación orientada a objetos. Un patrón de diseño proporciona un esquema para refinar sus subsistemas o componentes, o las relaciones entre ellos. Describe la estructura de la solución de un problema que aparece repetidamente; de componentes que se comunican entre ellos (36).

2.10.1 Patrones GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns, por sus siglas en inglés) o Patrones Generales de Software para Asignación de Responsabilidades, indican cual es la manera de asignar responsabilidades a objetos software (37). Estos patrones describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (38).

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

- **Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos (39). En el diseño del sistema se encuentra la clase Pipe, la cual representa gráficamente las tuberías, esta se encuentra compuesta por la clase PipeModel encargada de guardar todos los datos referentes a las tuberías como lo son: los colores del fluido y el valor de la medición. En esta relación de composición se evidencia el patrón Creador, pues es la clase Pipe responsable por la creación de la clase PipeModel.
- **Controlador:** Plantea asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Donde un evento del sistema es un evento de alto nivel generado por un actor externo (39). Este patrón se encuentra evidenciado en la clase Pipe y la clase PipeModel.
- **Experto:** Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (39). De forma general el diseño del sistema se basa en asignar a cada clase la responsabilidad que solo ellas pueden realizar, pues cada una cuenta con la información necesaria para llevarlas a cabo. Por lo que todas las clases del sistema son expertas.
- **Bajo acoplamiento:** El uso de los patrones Experto, Creador y Controlador contribuyen al bajo acoplamiento entre las clases del sistema. Este patrón se tuvo presente debido a la

importancia que se le atribuye a realizar un diseño de clases independientes que puedan soportar los cambios de una manera fácil y que a su vez permitan la reutilización.

- **Alta cohesión:** Se diseñaron las clases de forma tal que contengan las mínimas responsabilidades necesarias y colaboren con otras para llevar a cabo una tarea. Este patrón permitirá tener clases fáciles de mantener, entender y reutilizar.

2.10.2 Patrones GOF

Los patrones Gang-of-Four (pandilla de los cuatro, por sus cuatro autores) o comúnmente llamados Patrones GOF, descritos en el libro Design Patterns (39) definen un catálogo con 23 patrones básicos.

Se clasifican según su propósito en:

- **Creacionales:** Abstracción del proceso de creación de instancias.
- **Estructurales:** Muestran cómo las clases y los objetos son utilizados para componer estructuras de mayor tamaño.
- **Comportamiento:** Corresponde a los algoritmos y a la asignación de responsabilidades entre objetos (40).

Para la construcción de la solución, con el objetivo de garantizar una buena práctica de programación, se emplearon los **patrones GOF de comportamiento:** observador y el método plantilla. A continuación, se presenta una breve descripción de cómo se definen estos patrones y su comportamiento.

- **Observador:** Es un patrón de comportamiento que define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos (41). El uso de este patrón se ve reflejado en la funcionalidad `updateState ()`, la cual es utilizada para notificarle al componente que la medición asociada a él ha sufrido cambios.
- **Método Plantilla:** Este sencillo patrón resulta útil en casos en los que podamos implementar en una clase abstracta el código común que será usado por las clases que heredan de ella, permitiéndoles que implementen el comportamiento que varía mediante la reescritura (total o parcial) de determinados métodos (41). Este patrón se ve reflejado en los métodos `paintRuntime ()`, `loadSVG ()` y `paintEdition ()`.

2.11 Conclusiones Parciales

Como resultado del desarrollo del presente capítulo se obtuvo el diseño de la aplicación a desarrollar. Donde se logró generar el diagrama de dominio, el modelo del negocio donde se desea insertar la solución, lo que permitió identificar los requerimientos funcionales a implementar, los cuales, según la metodología seleccionada se describen como historias de usuarios. Así mismo se identificaron los requisitos no funcionales que permiten que el sistema funcione correctamente. La arquitectura del software permitió estructurar y organizar los componentes del programa, generar los diagramas de clases con sus respectivas relaciones y diseñar la solución mediante el uso de los patrones de diseño. Todo este proceso llevado a cabo durante el concluyente capítulo permitió obtener el modelado del negocio, listo para ser implementado.

Capítulo 3. Implementación y Pruebas

3.1 Introducción

Una vez diseñados los componentes gráficos para la representación de accesorios de una red de tubería, es necesario el desarrollo de esta solución. Posteriormente serán realizadas las pruebas, pues a pesar del correcto funcionamiento, pueden presentar fallas. Es por ello que este capítulo está centrado en la implementación y pruebas, donde será desarrollada la aplicación y validadas de manera eficiente cada una de las soluciones, para finalmente ser desplegadas en los sistemas de producción.

3.2 Modelo de implementación

El diagrama de componentes es otro de los artefactos importantes que incluye el Modelo de Implementación. El mismo muestra elementos del modelo, tales como, los componentes y sus relaciones. Se utiliza para modelar la vista estática del sistema y muestra la organización y las dependencias lógicas entre los componentes de software, sean estos de código fuente, binarios o ejecutables (42).

El diagrama de componente que se presenta a continuación está compuesto por bibliotecas (Library) y ejecutables (Executable):

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

- Resource: Biblioteca donde se encuentran agrupadas cada una de las entidades encargadas de brindar las imágenes, los iconos y los diseños de los componentes en SVG.

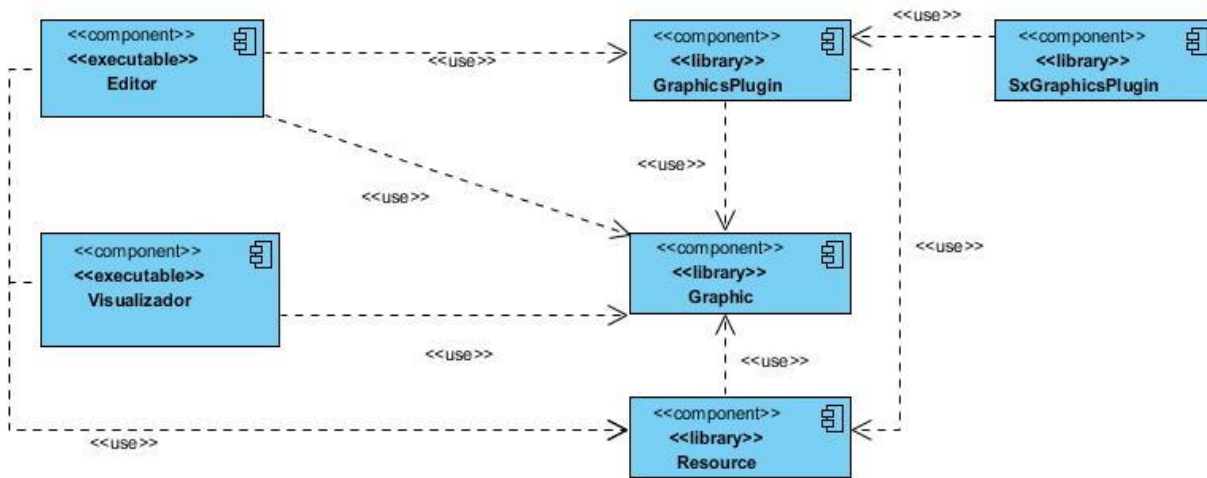


Fig. 8 Diagrama de componentes.

3.3 Estándar de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez.

Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada.

Estos estándares facilitan el mantenimiento del código, sirven como punto de referencia para los programadores, mantienen un estilo de programación y ayudan a mejorar el proceso de codificación haciéndolo más eficiente (43).

Como la solución propuesta en este trabajo es parte del sistema SCADA SAINUX el estándar de codificación utilizado fue definido por el proyecto:

- Los atributos de las clases deben comenzar con la letra m seguido de guion bajo y a continuación el nombre del atributo, si existen atributos compuestos la segunda palabra debe comenzar con mayúscula.

Ejemplo: m_isVisibleFlange

- Ninguna función debe tener más de 200 líneas de código.
- Las secciones public, protected y private serán declaradas en este orden.
- El código será escrito en inglés y la documentación en español.
- Para especificar el nombre del autor, en la documentación, se utilizan los comandos @autor.

Ejemplo: @author Rosmery Pedraza Ceballo rceballo@estudiantes.uci.cu.

- Las variables y funciones comienzan con letra minúscula, cada palabra consecutiva en el nombre comienza con letra mayúscula.

Ejemplo: tPipeModel.

- Los valores de los numerativos deben ser con letras mayúsculas.

Ejemplo: Type.

- Las funcionalidades y atributos para documentar deben cumplir el siguiente formato @brief Nombre del método.

Ejemplo: @brief isVisibleFlange1

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

Ejemplo: void setIsVisibleFlange1 (bool _isVisibleFlange)

3.4 Diagrama de despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria (44).

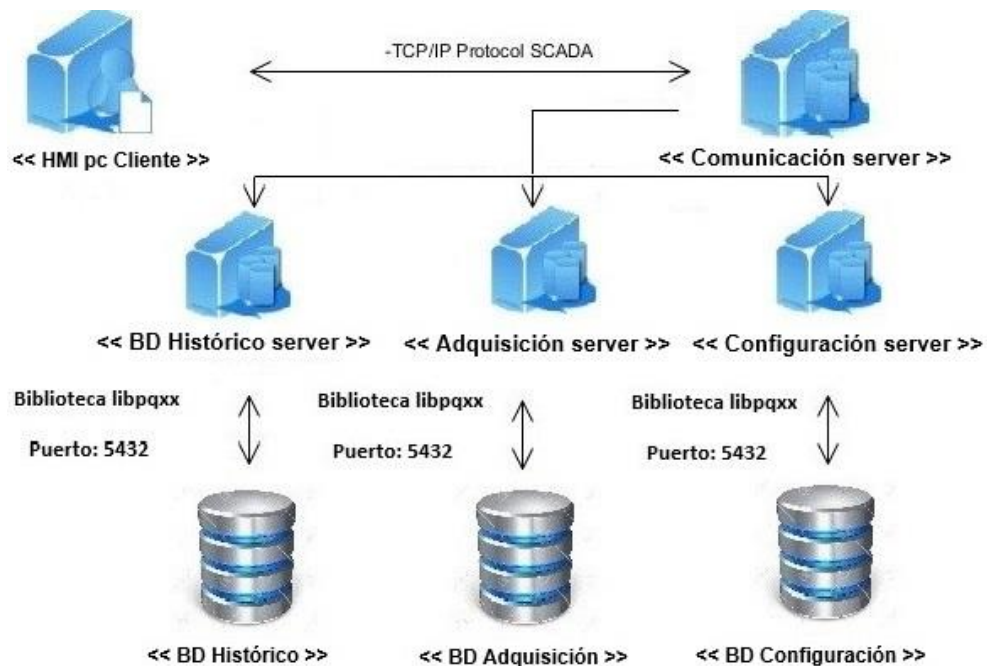


Fig. 9 Diagrama de despliegue

3.4.1 Descripción del diagrama de despliegue

La PC-Cliente representa el ordenador donde se instala el módulo HMI, en el cual pueden estar instalados los dos entornos o simplemente uno de ellos, es en este módulo específicamente donde se encuentra la solución desarrollada. Los módulos adquisición, configuración y base de datos históricos se ejecutan en una PC, que puede ser un servidor para cada uno por separado poniendo en evidencia la arquitectura distribuida del SAINUX, la conexión entre estos módulos y la PC-Cliente se realiza utilizando comunicación TCP/IP con el protocolo SCADA desarrollado por el centro. El módulo de configuración y base de datos históricos están conectados a sus respectivas bases de datos por la biblioteca libpqxx usando el puerto 5432.

3.5 Pruebas de software

El único instrumento adecuado para determinar el grado de calidad de un producto de software, es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el nivel en que el software cumple con los requerimientos (45).

En las pruebas se usan casos de prueba, especificados de forma estructurada mediante Técnicas de prueba. El proceso de pruebas, sus objetivos y los métodos y técnicas usados se describen en el plan de prueba (45).

3.5.1 Tipos de pruebas de software

Pruebas	Objetivo
Unitarias	<ul style="list-style-type: none"> - Ejecutar cada módulo. - Particionar y definir los casos de prueba. - Comparar el resultado (46).
Regresión	<ul style="list-style-type: none"> - Identificar errores introducidos por la combinación de programas probados unitariamente. - Determina como la base de datos de prueba será cargada (46).
Integridad	<ul style="list-style-type: none"> - Asegurar que los métodos de acceso y proceso funcionen adecuadamente y sin provocar corrupción de datos. - Determinar cómo la base de datos de prueba será cargada. - Asegurar que la implementación ha sido exitosa (46).
Aceptación	<ul style="list-style-type: none"> - Verificar por parte de cliente que el software cumpla con sus expectativas. - Evaluar las entradas que se realizan sobre el software y las salidas que produce, sin preocuparse por su comportamiento interno (47).

Seguridad y control de acceso

- Verificar que un actor solo pueda acceder a las funciones y datos que su usuario tiene permitido.
- Verificar que solo los actores con acceso al sistema y a la aplicación están habilitados para accederla (48).

De los tipos de pruebas de software existentes presentados con anterioridad se selecciona para comprobar la calidad de la solución desarrollada las pruebas de aceptación, ya que este es el tipo de pruebas recomendado por la metodología de desarrollo empleada. Es válido aclarar que estas pruebas las realiza el propio cliente acompañado del equipo de desarrollo y se orientan a las funcionalidades del sistema. Este tipo de pruebas genera como artefacto los Casos de Pruebas (CP).

3.5.2 Pruebas de caja negra

Las pruebas de caja negra también conocidas como pruebas de aceptación son realizadas por el cliente a partir de las historias de usuario. Básicamente aplicadas al sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, son aplicadas al producto terminado e integrado (49).

Una prueba de aceptación puede ir desde un informal caso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas. Pueden tener lugar a lo largo de semanas o meses, descubriendo así errores latentes o escondidos que pueden ir degradando el funcionamiento del sistema. Estas pruebas son muy importantes, ya que definen el paso nuevas fases del proyecto como el despliegue y mantenimiento (49).

Para la realización de las pruebas de caja negra se empleó la técnica Partición de Equivalencia. Esta permite examinar los valores válidos e inválidos de las entradas existentes en el software (50).

3.5.2.1 Diseño de casos de prueba

Un caso de prueba cubre el software más a fondo y con más detalle que un caso de uso. Los mismos incluyen todas las funciones que el programa es capaz de realizar. Estos deben tener

en cuenta el uso de todo tipo de datos de entrada/salida, cada comportamiento esperado, todos los elementos de diseño, y cada clase de defecto. Todos los requisitos deberán ser cubiertos por los casos de prueba (51).

A continuación, se muestra un CP correspondiente a las pruebas de aceptación realizadas, los restantes se encuentran en el Anexo 4.

Tabla 5 Caso de Prueba 12

Prueba de Aceptación	
Número: 12	Historia de usuario: 12
Nombre: Mostrar/Ocultar bridas pertenecientes al componente.	
Descripción: Encargado de comprobar que la opción mostrar/ocultar brida funciona correctamente.	
Condiciones de Ejecución: El usuario debe comprobar que puede mostrar u ocultar las bridas pertenecientes al componente.	
Entradas/ Pasos de Ejecución:	
<ul style="list-style-type: none"> • Seleccionar componente. • Clic derecho con el ratón y abrir inspector de propiedades. • Marcar o no la opción mostrar/ocultar bridas. 	
Resultado esperado: Las bridas se pueden mostrar u ocultar.	
Evaluación de la prueba: Se realiza la primera iteración donde fueron identificadas varias no conformidades, se solucionan estas y se realiza otra iteración que arrojó una evaluación satisfactoria.	

Siendo realizadas 3 iteraciones de pruebas, los resultados obtenidos son los siguientes:

Tabla 6 Resultado de las iteraciones.

Sistema	HU	Iteración	NC	Cerrada	No Procede
Accesorios de una red de tubería.	33	1ra	10	8	2

		2da	8	8	0
		3ra	0	0	0

Durante el plan de iteraciones realizado, para llevar a cabo la fase de pruebas con el objetivo de validar el correcto funcionamiento de la aplicación desarrollada, se detectaron 18 no conformidades las cuales fueron resueltas una vez realizadas 3 iteraciones siendo dos de ellas no procesadas. Las no conformidades encontradas pueden clasificarse según el tipo de falta:

De presentación: No se visualiza toda la información que se desea mostrar y fueron localizados varios errores ortográficos siendo estos generalmente de acentuación.

De ejecución: Durante el despliegue del sistema fueron detectadas diferentes fallas, siendo estas ocasionadas porque el sistema no realizaba todos los procedimientos definidos.

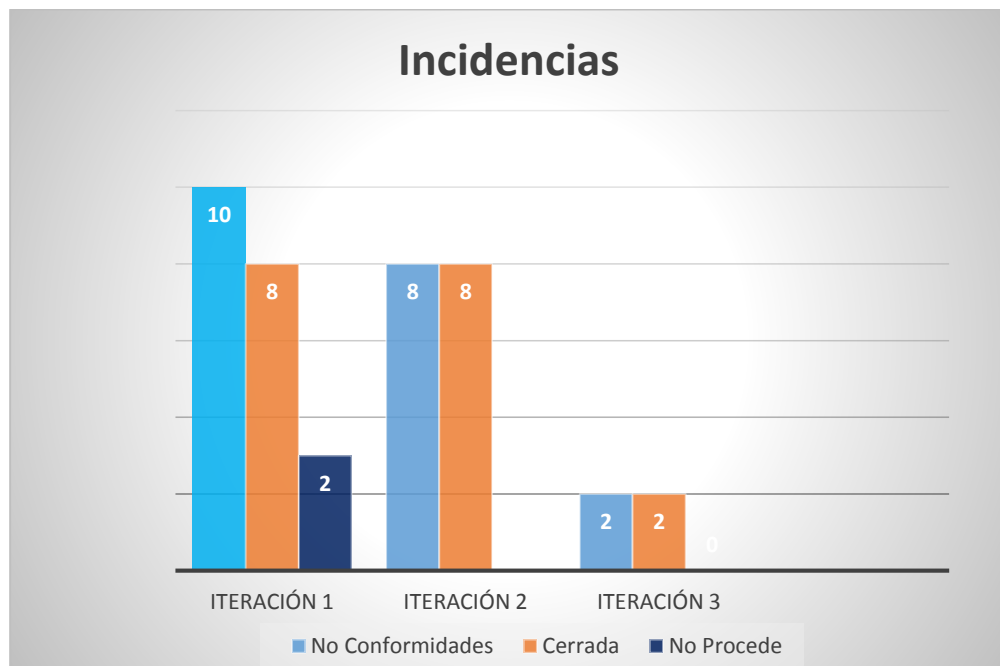


Fig. 10 Incidencias detectadas en el sistema

3.6 Conclusiones Parciales

En el desarrollo del capítulo se especificaron los resultados obtenidos luego de desarrollado el sistema, el cual cumple con los requisitos funcionales encapsulados en las Historias de Usuario

definidas anteriormente; todo esto permitió un mejor entendimiento del mismo. De igual forma, en este capítulo, se ejecutaron las pruebas de aceptación de manera satisfactoria, donde a partir de los casos de prueba se pudo verificar que el software desarrollado está listo y que puede ser usado por los usuarios finales, para ejecutar aquellas funciones y tareas para las cuales fue diseñado y construido.

Conclusiones generales

Una vez concluida la presente investigación se logró:

- ✓ Definir el marco teórico de la investigación a través del estudio del estado del arte que existe sobre los accesorios utilizados para la representación de las redes de tubería en el HMI de sistemas SCADA.
- ✓ Desarrollar componentes gráficos que permitieron incrementar los grados de representación de los accesorios que conforman las redes de tubería en el proceso de visualización de la Interfaz Hombre Máquina del sistema SCADA SAINUX.
- ✓ Además de los objetivos propuestos se logró como resultado adicional el desarrollo de una animación para representar el fluido que circula por los accesorios.

Cabe destacar que para la realización de la presente investigación se utilizaron herramientas y tecnologías libres.

Recomendaciones

Teniendo en cuenta los resultados obtenidos en esta investigación y basado en la experiencia adquirida, se recomienda:

- ✓ Ampliar la gama de accesorios utilizados para la representación de una red de tubería.
- ✓ Mejorar la animación que representa el paso del fluido por estos accesorios.
- ✓ Incorporar los componentes gráficos desarrollados en el visor web existente en el centro.
- ✓ Visualización de los accesorios utilizando la simbología estándar definida para ellos.

Referencias

1. Penin, A.R. *Sistemas SCADA*. s.l. : Marcombo, 2012. ISBN 9788426716477.
2. **OMG Unified Modeling Language™ (OMG UML), Superstructure.** [En línea] 2011. [Citado el: 14 de enero de 2016.] <http://www.omg.org/spec/UML/2.4/Superstructure>.
3. Definición. [En línea] [Citado el: 14 de enero de 2016.] <http://definicion.mx/grafico/>.
4. Ingeniería de Software. [En línea] [Citado el: 15 de marzo de 2016.] <http://clases3ggingsof.wikifoundry.com/page/Proceso+de+Despliegue+de+RUP>.
5. Dagoberto Monteros, David B. Barrantes y José M. Quirós. *Introducción a los sistemas de control, supervisión y adquisición de datos (SCADA)*. 2004.
6. Instruments, National. National Instruments. [En línea] [Citado el: 2 de diciembre de 2015.] <http://www.ni.com/data-acquisition/what-is/esa>.
7. Definición.de. [En línea] [Citado el: 2 de diciembre de 2015.] <http://definición.de/supervisión>.
8. Funciones del Sistema SCADA. [En línea] [Citado el: 4 de octubre de 2015.] http://www.oocities.org/gabrielordonez_ve/FUNCIONES_DEL_SISTEMA_SCADA.htm.
9. Emersonprocess. Módulo de Comunicación (serie ROC 800). [En línea] octubre de 2003. [Citado el: 20 de septiembre de 2015.] http://www.documentation.emersonprocess.com/groups/public/documents/specification_sheets/d301583x012.pdf.
10. B.O.D. Servicio de Biodinternet, guía de acceso al módulo de configuración. [En línea] enero de 2013. [Citado el: 25 de octubre de 2015.] http://www.bod.com.ve/media/26317/Guía_del_Módulo_Configuración.pdf.
11. Group, ABB. Módulo de Seguridad. [En línea] [Citado el: 25 de octubre de 2015.] <http://www.abb.com/product/es/9AAC170130.aspx>.
12. Yepes, José López. Redalyc.org, Las Bases de Datos Históricas. [En línea] septiembre de 2001. [Citado el: 25 de octubre de 2015.] <http://www.redalyc.org/articulo.oa?id=16100905>.
13. Morales., Carlos de Castro Lozano y Cristóbal Romero. *Introducción a SCADA*.
14. Prezi. [En línea] [Citado el: 21 de abril de 2016.] <https://prezi.com/yzgen1zgciz5/sistemas-de-redes-de-tuberias/>.

15. Puentes, Carlos A. Interpretación de planos. Simbología de Tuberías. [En línea] [Citado el: 28 de septiembre de 2015.] <http://es.slideshare.net/carlosapuentes/simbologiadetuberias> .
16. Eumed. Eumed.net. [En línea] [Citado el: 2 de diciembre de 2015.] <http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm>.
17. MEJORA, PROGRAMA DE. Metodología de Desarrollo para la Actividad Peroductiva de la UCI. [En línea] Tamara Rodríguez Sánchez. [Citado el: 2 de diciembre de 2015.] <http://excriba.prod.uci.cu/page/context/shared/sharedfiles/Metodologiauci.pdf>.
18. ALEGSA.com.ar. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. Definición de SVG. [En línea] [Citado el: 17 de enero de 2016.] <http://www.alegsa.com.ar/Dic/svg.php>.
19. Mad, Freepress S. Coop. Freepress S. Coop. Inkscape-software-libre-para-diseño-vectorial. [En línea] [Citado el: 2 de diciembre de 2015.] <http://www.freepress.coop/recursos/inkscape-software-libre-para-diseno-vectorial/>.
20. Oliag, Sergio Talens. Curso de Programación en C++. [En línea] 1995. [Citado el: 30 de octubre de 2015.] <http://www.uv.es/~sto/cursos/c++/curso95.pdf>.
21. CORPORATION, NOKIA. Qt. [En línea] 2008. [Citado el: 2 de diciembre de 2015.] <http://qt.nokia.com/products/developer-tools/>.
22. Raydel Raúl Viñolo Sosa, Alexander Roquero Figueroa. Sistema Gestor de Proceso de Media v2. [En línea] julio de 2012. [Citado el: 2 de diciembre de 2015.] <http://publicaciones.uci.cu/index.php/SC | seriecientifica@uci.cu>.
23. Cornejo, José Enrique González. DOCIRS. [En línea] enero de 2008. [Citado el: 2 de diciembre de 2015.] <http://www.docirs.com/uml.htm>.
24. CMM. CMM.net. [En línea] [Citado el: 2 de diciembre de 2015.] <http://es.ccm.net/download/descargar-28127-visual-paradigm-for-uml-enterprise-edition>.
25. Debian. The Universal Operating System. [En línea] [Citado el: 2 de diciembre de 2015.] <http://www.debian.org>.
26. Subversion. Apache Subversion. [En línea] [Citado el: 2 de diciembre de 2015.] <https://subversion.apache.org/>.
27. UML y Patrones. [aut. libro] Craig Larman. *UML y Patrones. Modelo de Dominio*. . Prentice Hall : s.n., 2003.
28. Pearson. *Sommerville, Ingeniería del Software. Vol. 7*. 2005.
29. Cardozzo., Daniel Ramos. *Desarrollo de Sotware. Campus Academy*. 2014.

30. Microsoft. [En línea] [Citado el: 28 de marzo de 2016.] <https://msdn.microsoft.com/es-es/library/dd409377.aspx>.
31. Microsoft. Diagramas de secuencia UML: Referencia. [En línea] [Citado el: 29 de marzo de 2016.] <https://msdn.microsoft.com/es-es/library/dd409377.aspx>.
32. —. Diagramas de clases de UML: Instrucciones. [En línea] [Citado el: 14 de marzo de 2016.] <https://msdn.microsoft.com/es-es/library/dd409416.aspx>.
33. Relaciones entre clases: Diagramas de clases UML. [En línea] [Citado el: 10 de marzo de 2016.] <http://elvex.ugr.es/decsai/java/pdf/3C-Relaciones.pdf>.
34. Pressman, Roger S. Ingeniería del Software. Un enfoque Práctico. [En línea] [Citado el: 3 de febrero de 2016.] <http://bibliodoc.uci.cu/pdf/8448111869.pdf>.
35. Qt Documentation. [En línea] [Citado el: 16 de marzo de 2016.] <http://doc.qt.io/qt-4.8/model-view-programming.html>.
36. CCM. *Patrones de Diseño*. [En línea] [Citado el: 17 de marzo de 2016.] <http://es.ccm.net/contents/224-patrones-de-diseno..>
37. Hern., Marcello Visconti y. *Fundamentos de Ingeniería de Software*.
38. Marcello Visconti, Hernán Astudillo. Fundamentos de Ingeniería de Software. [En línea] [Citado el: 10 de abril de 2016.] <https://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf>.
39. Gamma, E., y otros. *Design Patterns. Elements of Reusable Object-Oriented Software*.
40. geektheplanet. [En línea] [Citado el: 8 de abril de 2016.] <http://geektheplanet.net/5462/patrones-gof.xhtml>.
41. Timothy G. Mattson, Beverly A. Sanders, Berna L. Massingill. *Patterns for Parallel Programming*. s.l. : Addison-Wesley, 2008.
42. Arquitectura de software. Diagrama de componentes. [En línea] [Citado el: 29 de marzo de 2016.] <http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-yDespliegue>.
43. Estándares de Codificación .Net v.1.0.0.0. [En línea] [Citado el: 15 de abril de 2016.] <http://serk.kualtus.com/codigo.htm>.
44. Diagrama de Despliegue. ANALISIS Y DISEÑO DE SISTEMAS II. [En línea] [Citado el: 26 de mayo de 2016.] virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc.
45. pruebasdesoftware. [En línea] [Citado el: 18 de abril de 2016.] <http://www.pruebasdesoftware.com/laspruebasdesoftware.htm>.

46. EliuMM. [En línea] [Citado el: 19 de abril de 2016.]
<http://javablog.eliumontoya.com/home/tipodepruebasparadesarrollodesoftware>.
47. 4R Blog. [En línea] [Citado el: 20 de abril de 2016.] <http://www.4rsoluciones.com/test-de-aceptacion-el-ultimo-paso-para-el-aseguramiento-de-calidad-en-software/>.
48. Javier Zapata S. Ingeniería de Software con énfasis en las pruebas. Pruebas de Software. *Niveles de Pruebas del Software*. [En línea] [Citado el: 20 de abril de 2016.]
<https://pruebasdelsoftware.wordpress.com/>.
49. 4RBLOG. [En línea] [Citado el: 21 de abril de 2016.] <http://www.4rsoluciones.com/test-de-aceptacion-el-ultimo-paso-para-el-aseguramiento-de-calidad-en-software/>.
50. ¿Qué es un Patrón de Diseño?. [En línea] [Citado el: 2 de junio de 2016.]
<https://msdn.microsoft.com/es-es/library/bb972240.aspx>.
51. Definición de caso de prueba. Testeando el Software. [En línea] [Citado el: 13 de mayo de 2016.] <http://testeandosoftware.com/casos-de-uso-vs-casos-de-prueba..>
52. Juan Quijano. GENBETA:dev. *Historias de usuario, una forma natural de análisis funcional*. [En línea] [Citado el: 16 de marzo de 2016.]
<http://www.genbetadev.com/metodologias-de-programacion/historias-de-usuario-una-forma-natural-de-analisis-funcional>.
53. Definición de caso de prueba. *Testeando el Software*. [En línea] [Citado el: 15 de mayo de 2015.] <http://testeandosoftware.com/casos-de-uso-vs-casos-de-prueba/>.
54. Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la actividad productiva de la UCI*. Universidad de las Ciencias Informáticas. La Habana : s.n.
55. Sommerville, I. *Ingeniería del Software*. s.l. : Pearson, 2005. Vol. 7.