Universidad de las Ciencias Informáticas
Facultad 5

Título: Mecanismo de gestión de Gráficos Unifilares

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autor(es): Reinier Consuegra Peniche

Tutor(es):

Ing. Bernardo Zaragoza Hijuelos

Co-tutor(es):

Ing. Roberto Cardenas Isla

Abril, 2011

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Reinier Consuegra Peniche

_____
Firma del Autor

Ing. Bernardo Zaragoza Hijuelos

_____
Firma del Tutor

Ing. Roberto Cardenas Isla

_____
Firma del Tutor

AGRADECIMIENTOS

Le agradezco a ese hombre que tuvo entre sus grande y geniales ideas la creación de este centro univerisitario, nuestro eterno Comandante Fidel; logrando con esto darme la oportunidad de formarme como profesional. A mi familia por la educación recibida, por fomentar el amor y enseñame que la entrega constante y sacrificada da buenos frutos. Además a mis tutores Roberto Cárdenas Isla y Bernardo Zaragoza Hijuelos por ser elementos fundamentales para el desarrollo de este trabajo; por su disposción en cada momento de ayudarme. También agradecer al colectivo de profesores y estudiantes del proyecto HMI(Human Machine Interface) del Centro de Desarrollo de Informática Industrial de la UCI por su apoyo incondicional cada vez que lo necesité. A mis compañeros de estudio por apoyarme cuando lo necesite. A todas aquellas personas que me apoyaron y guiaron en todo momento.

DEDICATORIA

Este tarbajo está dedicado a mis seres más queridos a mi madre por ser mi sustento emocional por traerme a esta vida, a mi padre por ser mi guía tanto en la vida personal como en la vida profesional, a mi hermano Ale por aguantarme todos estos años, a mis abuelas Victoria y Zoila por ser las voces de la experiencia dentro de mi seno familiar,a mi abuelo Moises por ser uno de los simientos fundamentales de mi familia, a mi abuelo Luisito que aunque ya no este entre nosotros fue mi segundo padre;que en gloria este, a todos mis familiares; no menciono todos los nombre porque son un buen número, asi que si en algún momento alguien de mi familia lee esto no se sienta mal porque a todos los llevo bien dentro de mi ser. Agredecer además a mi piquetazo de la vida Ale, Hector, Danyer, Iriobe, Frank, Duniesky, Richard, Adis, Lisandra, Adonis, Leonel, Randy por estar en mi vida desde que tengo uso de razón y por apoyarme todo el tiempo en mis decisiones. Además quiero dedicarle este trabajo a ese grupo de personas que la vida me dio la dicha de conocer cuando comencé mis estudios universitarios Andy, Manuel, Yasmina, Anna, Cecy, Fumero, Nidelso, el viejo Ramiro, a mi flaqui de la vida Amaya, Tony, Rufino, Leitniz, Carlos, Ale, Dayana, Aylin. Este trabajo está dedicado a todas aquellas personas que de una manera u otra han aportado algo a mi vida en todas sus esferas.

RESUMEN

Los sistemas de SCADA, acrónimo de Supervisory Control and Data Acquisition (Control, Supervisión y Adquisición de Datos) se han convertido en pilar fundamental dentro del marco industrial. Una de las industrias que más se ha visto beneficiada por estos sistemas es la industria eléctrica debido a la complejidad de sus procesos industriales.

Los SCADA Eléctricos tienen entre sus principales prestaciones la visualización y configuración de todo el proceso industrial a partir de componentes gráficos conocidos como gráficos unifilares. Este tipo de componentes son una representación de un dispositivo eléctrico a partir de primitivas gráficas básicas: círculos, cuadrados, rectángulos, líneas, etc.

Actualmente se está desarrollando uno de estos sistemas para la industria eléctrica cubana. Este nuevo sistema deberá proveer un mecanismo para la configuración de los componentes gráficos de esta rama industrial. Dicho mecanismo estará respaldado por un conjunto de normas cubanas definidas por la Unión Nacional Eléctrica y desarrollado usando la tecnología Qt como framework de visualización.

Como resultado de este trabajo se pretende obtener una herramienta de edición de gráficos unifilares para la industria eléctrica cubana. Esta herramienta facilitará el proceso de diseño de dichos gráficos; permitirá salvar y cargar los componentes diseñados, obtener nuevos componentes a partir de otros ya concebidos con anterioridad; lo que facilita la extensibilidad de una biblioteca gráfica con un costo bajo de tiempo.

Palabras Claves:

SCADA, tecnología Qt, gráficos unifilares, diagrama eléctrico, herramienta de edición

TABLA DE CONTENIDOS

INTRODUCCION

La tecnología está en constante avance y junto a ella los procesos y sistemas industriales son cada vez más diversos y complejos. La informática ha logrado desarrollar software con la capacidad de automatizar los procesos industriales, mejorando así los resultados de las industrias en gran medida en términos de eficiencia y resultados de las mismas. La informatización industrial es la rama de la informática encargada del tratamiento automático de la información proveniente de los procesos industriales, apoyándose fundamentalmente en la computadora. La tecnología que trata la automatización de procesos en la industria es conocida como SCADA (Supervisión, Control y Adquisición de Datos) en sistemas, que como sus siglas intuyen, adquieren, controlan y supervisan los datos de campos. En el mundo existen varios tipos de SCADA pues estos presentan características diferentes para cada uno de los entornos o ramas de la economía que se quiere controlar o automatizar, ejemplo de ellos son los procesos meteorológicos, hidráulicos, petrolíferos, eléctricos.

La Universidad de las Ciencias Informáticas (UCI), creada con el objetivo de lograr el desarrollo de software nacional, ya ha tenido avances en la creación de software SCADA para la industria petrolífera en colaboración con la empresa venezolana GALBA (Guardián de la Alianza Bolivariana para las América), y ha implantado otros sistemas similares en Cuba en las ramas de la hidráulica y la meteorología. Estos sistemas dan respuesta a soluciones rápidas y no muy complejos, no siendo así para los SCADA en la industria eléctrica.

En conjunto con Venezuela, nuestra universidad desea desarrollar un SCADA eléctrico, tarea que esta siendo supervisada por el CEDIN (Centro de Desarrollo de Informática Industria). Este centro consta con varias líneas de trabajo o de componentes las cuales se dividen por especialidades o ramas dentro de la informática, en nuestro caso la línea de componentes es Interfaz Hombre-Máquina (Human Machine Interfaces) dentro de los SCADA. El módulo de HMI está compuesto por dos sub-módulos: Ambiente de configuración y de edición. El ambiente de configuración brinda la posibilidad de editar de manera sinóptica de todo el proceso industrial, además presenta un conjunto de funcionalidades que permiten gestionar cuestiones de diseño de los despliegues, configurar los diferentes módulos, editar las variables, etc. El sub-módulo de edición permite al operador la interacción con el sistema y las variables de campo desde el ordenador.

El HMI del SCADA petrolífero con el que se cuenta ahora no es totalmente compatible con la filosofía de un SCADA eléctrico por lo que se hace necesario

reutilizar todo el conocimiento acumulado además de las tecnologías ya desarrolladas e incluir nuevos conceptos que anteriormente no se tenían identificados en los sistemas de petrolíferos. Por ello se ha venido trabajando en una arquitectura nueva así como en el uso de la tecnología de extensiones, lo cual ha dado como resultado una nueva herramienta de edición “Editor Phoenix” que permita englobar de una manera, lo más genérica posible, los conceptos que ayuden a editar cualquier tipo de SCADA. Es por ello que este editor no cuenta con los nuevos conceptos necesarios para realizar o diseñar un sistema eléctrico.

Para lograr que Phoenix edite la configuración de un SCADA Eléctrico se hace necesario analizar e incluir nuevos conceptos que se usan en el diseño de circuitos los cuales se hacen puramente a partir de primitivas geométricas: líneas, elipses (círculos), rectángulos (cuadrados), curvas, etc. Este tipo de diseño se conoce como gráficos unificar (mono-lineal) que se usan frecuentemente en diseño de diagramas circuitales en la electrónica. Otro cambios a realizar es en cuanto al cúmulo de elementos gráficos o símbolos utilizados por la ingeniería de la electrónica lo que fuerza a tener una biblioteca gráfica extensa y por ende debe ofrecer más flexibilidad de creación al operador o diseñador en el diseño de los circuitos; además la información a mostrar incluye nuevos conceptos en la dinámica del diseño de circuitos que antes no se realizaban en los sistemas de petróleo.

En estos momentos el CEDIN se encuentra desarrollando un sistema de control, adquisición y supervisión de datos para la industria eléctrica. Para concepción de este proyecto se tomo como decisión la utilización de algunos mecanismo ya elaborados a raíz de soluciones anteriormente obtenidas por dicho centro algunas como el SCADA Guardián del ALBA. Uno de los mecanismos que el centro, específicamente la línea HMI, prevé reutilizar es el sistema de edición de componentes gráficos del sistema SCADA Guardián del ALBA, el mecanismo de edición lleva por nombre editor Phoenix. Este mecanismo de edición carece de la capacidad de gestionar los componentes gráficos de un SCADA Eléctrico, dichos componentes en términos de electricidad se denominan gráficos unifilares.

Ante la situación problemática planteada se define el siguiente **problema científico:** ¿Como proveer al editor Phoenix de un mecanismo para la gestión de componentes gráficos de la industria eléctrica?

Se plantea como **objeto de la investigación:** Herramientas y tecnologías para la edición de gráficos unifilares en sistemas SCADA.

El objeto de estudio delimita el **campo de acción**: modulo de visualización del SCADA Eléctrico.

Por tanto se define como **objetivo general:** Desarrollar un mecanismo para la gestión de gráficos unifilares en el editor Phoenix del SCADA Eléctrico.

Para cumplir el objetivo general se definen los siguientes objetivos específicos**:**

- Analizar la documentación existente acerca de la implementación de un sistema de edición de gráficos unifilares en sistemas SCADA eléctricos.
- Diseñar las clases que cumplan con los requerimientos encontrados en el análisis.
- Implementar el mecanismo de gestión de gráficos unifilares.
- Valorar la solución obtenida.

Se tiene la siguiente **idea a defender:** Si se desarrolla un editor para de gráficos unifilares se logrará un sistema de mayor funcionalidad, calidad y eficacia.

Para llevar cabo este trabajo y cumplir con los objetivos específicos propuestos se concibieron las siguientes **tareas:**

- Elaboración del marco teórico a través de estudio del estado del arte de los modelos y sistemas para la representación de gráficos unifilares en SCADA eléctricos.

- Selección de metodología de desarrollo de software.

- Selección de herramientas y tecnologías para el desarrollo de aplicaciones de gestión de gráficos unifilares.

- Definición de funcionalidades para la gestión de gráficos unifilares en SCADA eléctricos.

- Análisis y diseño del mecanismo para la gestión de gráficos unifilares en SCADA eléctricos.

- Implementación de un mecanismo para la representación de gráficos unifilares para el SCADA Eléctrico.

Se plantean los siguientes métodos teóricos:

Analítico-Sintético: Para la consulta de la documentación existente acerca de los mecanismo de gestión de gráficos unifilares en editores de configuración, y el estudio de los conceptos empleados en el tema en cuestión.

Análisis Histórico-Lógico: Para tener información sobre las tendencias actuales de los SCADA Eléctricos y de sus editores de configuración.

El presente documento está estructurado en tres capítulos:

En el **Capítulo 1** se realiza un esbozo teórico centrado en temáticas como sistemas SCADA y SCADA Eléctricos, módulo de visualización, esquemas unifilares y graficos unifilares, las principales tendencias y tecnologías actuales que sirven de apoyo a todo el trabajo desarrollado.

En el **Capítulo 2** se describe la solución propuesta, mostrando la tecnología seleccionada, los requerimientos funcionales y la descripción de las principales clases.

En el **Capítulo 3** se describe el diseño arquitectónico de la solución así como alguno de los principales diagramas de clases y de paquetes del sistema.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se presentan las definiciones y conceptos del tema de desarrollo y gestión de gráficos unifilares así como el análisis y descripción sobre la concepción general de un SCADA eléctrico. También se hace un resumen de las principales tecnologías y de las principales tendencias de desarrollo que se utilizan para el desarrollo de gráficos, así como las bibliotecas graficas que se podrán utilizar para el desarrollo de los mismos.

1.2 Definición de SCADA

En términos de gestión y control de procesos de automatización industrial uno de los sistemas más utilizado es SCADA acrónimo de Supervisory Control And Data Acquisition (Supervisión, Control y Adquisición de Datos). Estos sistemas se han convertido en nuestros días parte imprescindible en casi todas las industrias a nivel mundial.

Este tipo de sistemas utilizan un grupo de tecnologías de comunicación para el control y monitoreo de los procesos industriales. Además controlan no solo la información dentro de la industria sino con la utilización de dispositivos de campo pueden capturar y recopilar información de manera tal que las conclusiones conjuntas dentro del sistema SCADA se hacen más eficientes y proporcionan un mayor cúmulo de datos al operador del sistema.

1.3 Definición de un SCADA eléctrico

La Industria Eléctrica no queda fuera del grupo de industria que disponen casi de manera imprescindible de este tipo de sistema de control (SCADA), ya que cuenta con un grupo de subestaciones eléctricas. Las subestaciones eléctricas no son mas unidades de distribución de energía eléctrica en alta o media tensión cuyas instalaciones normalmente son desatendidas, es decir no tienen personal dedicado a su mantención el 100% del tiempo; es por eso el porqué de la incorporación de un término conocido por SCADA Eléctrico, que no es más que un sistema de monitoreo y control de información como el de los sistemas de adquisición, supervisión y control de datos

comunes pero se le incorporan funcionalidades especificas para el sector eléctrico. Los SCADA eléctricos tiene como misión principal servir de medios para el monitoreo el control remoto manual de subestaciones eléctricas.

1.4 Componentes básicos de un SCADA eléctrico

Los SCADA Eléctricos están compuestos por un grupo de componentes básico encabezados por las estaciones de operación, RTU (Unidad de Terminal Remota) (1) con de entradas-salidas y puertas de comunicaciones, redes de comunicación, equipos de protección y medida, sincronismo horario y estampa de tiempo.

1.4.1 Estaciones de operación

El componente de estaciones de operaciones corresponde a un grupo de computadoras donde se cargan software de tipo industrial tales que permiten la visualización de las variables y estados de los equipos de las subestaciones tales como corrientes, potencias, voltajes, estados abierto y cerrado de equipos como desconectadores e interruptores y señalización de estados de trips.

1.4.2 RTU (Unidad de Terminal Remota) con de entradas-salidas y puertas de comunicaciones

En este componente se establece el nivel de confianza en las redes de comunicaciones que es un capítulo en los sistemas de control que tiene un largo camino que recorrer antes que sea adoptada como un estándar en la industria. Tal es así que es de uso normal hoy en día CPU con módulos de entradas salidas para el monitoreo de estados discretos y comandos de paños en alta tensión en rangos de 220 Kv, 110 Kv y 69 Kv. Es decir en estos casos se estima conveniente, por razones de confiabilidad, entregar las funciones antes indicadas a una RTU con señales alambradas antes que obtener dichas señales desde los IED vía comunicaciones. La captura de variables eléctricas por medio de tarjetas análogas de 4 a 20 mA ya no se utiliza prácticamente debido a obsolescencia tecnológica. Actualmente esta función radica en módulos de comunicaciones los que extraen tales señales mediante la puerta de comunicaciones de los IED (1).

1.4.3 Redes de comunicación

Las redes de comunicación siempre han marcado una gran pauta en los sistemas industriales y más en los sistemas SCADA. Todo este proceso comenzó por la década de los 70 donde las comunicaciones electrónicas eran principalmente punto a punto, es decir desde un equipo electrónico a otro. Ya en los años 80 las redes de comunicación fueron implementadas en una gran cantidad de marcas de sistemas de control industriales. La principal característica de dichas redes era su robustez frente al ruido electromagnético, la sencillez de construcciones basadas en cables apantallados y su carácter propietario, es decir sólo equipos y software del mismo fabricante podían comunicarse entre sí. Aún así el alcance en distancia de dichas redes era considerable: hasta dos o tres kilómetros. En la época de los años 90 se consolidó el uso de las redes ethernet en el mundo de las oficinas. Aún así el mundo de las redes industriales no aceptó dichas redes por no poseer las características de robustez requeridas tales como la inmunidad al ruido electromagnético y el hecho de no ser determinística. El cambio lo originó la fibra óptica y su gran disminución de precios tal que destruyó el concepto de poca robustez por cuanto la fibra está menos afecta al ruido eléctrico que los cables apantallados, posee un alcance en distancia que puede llegar al centenar de kilómetros sin repetidor, y su velocidad de comunicaciones tal que no requiere del concepto determinístico para lograr que una señal llegue a su destino en fracciones de segundo. A pesar de lo anterior hoy en día existe una variedad de redes de comunicaciones de diversas generaciones en las subestaciones. Se pueden encontrar desde redes seriales hasta redes ethernet sobre fibra óptica (1).

1.4.4 Equipos de protección y medida

Uno de los pilares del desarrollo de los SCADA eléctricos proviene de los fabricantes de equipos de protección y medida. Fabricantes como PML, Siemens, General Electric, Schweitzer y otros han impulsado el desarrollo de sus equipos desde simples puertas de comunicaciones tipo DB9, que permiten que un usuario se conecte al equipo mediante un computador portátil, con el fin de configurarlo o de extraer información de consumos, estados del equipos, oscilografías o reportes de fallas, hasta puertas ethernet en formatos RJ45 o conectores de fibra multi-modo o mono-modo. Este último desarrollo permite un salto significativo en la interacción con el equipo de protección y medida, ya que la red ethernet posibilita no sólo el monitoreo y control a distancia sino que la obtención de data en línea de oscilografía o la configuración de la protección a distancia (1).

1.4.5 Sincronismo horario y estampa de tiempo

Hoy en día ya existen exigencias de sincronización horaria de equipos de medida y protección, con el fin de recoger la información desde estos equipos de la misma forma, es decir información sincronizada en la Estación de Operación. Esto permite un estudio más acucioso de los diversos eventos que se ocasionan en un sistema eléctrico. Los equipos utilizados para este efecto son los denominados GPS, que son CPU con una antena tal que se sincroniza a relojes satelitales. A esto se suma la capacidad de adaptar el sincronismo satelital a la hora local, con los debidos cambios de horarios de invierno y verano en forma automática (1).

1.5 Módulos

Los SCADA Eléctricos tienen al igual que los demás tipos de SCADA un conjunto de módulos que comienzan por el modulo de configuración y termina en el modulo de gestión de archivos y datos:

Configuración: El modulo de configuración permite al usuario o grupo de usuarios definir el entorno de trabajo de su SCADA, adaptándolo a la aplicación particular que se desea desarrollar.

Interfaz gráfico del operador: El modulo interfaz grafico del operador es el encargado de proporcionarle al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos almacenados en el ordenador de proceso y generados desde el editor incorporado en el SCADA o importados desde otra aplicación durante la configuración del paquete.

Módulo de proceso: Este modulo ejecuta las acciones de mando pre-programadas a partir de los valores actuales de variables leídas.

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

Gestión y archivo de datos: El modulo de gestión y archivos de datos es el encargado del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.

Estos módulos se manejan o definen de la misma manera que los de un SCADA común pero no tienen lo mismo objetivo ya que lo requisitos no se obtienen de la misma manera y lo resultados que se esperan de los mismo no son iguales que los de un SCADA petrolífero o hidráulico. El modulo que más cambios sufre con respecto a los de los otros tipos de SCADA es el modulo HMI ya que la concepción de las interfaces graficas, los componentes que se manejan dentro del mismo tienen ciertas especificación debido a que la rama eléctrica tiene un manejo particular de estos términos y elementos. Todo esto se debe a que en la conformación de un SCADA Eléctrico una de las terminologías que mas resalta es la de esquemas unifilares que son una de las herramientas fundamentales dentro de este tipo de sistema de supervisión, control y adquisición de datos.

1.6 Definición de esquemas o diagramas unifilares

Los esquemas o diagramas unifilares no son más que una representación gráfica de una instalación eléctrica o de parte de ella. El esquema unifilar se distingue de otros tipos de esquemas eléctricos en que el conjunto de conductores de un circuito se representa mediante una única línea, independientemente de la cantidad de dichos conductores. Típicamente el esquema unifilar tiene una estructura de árbol.

1.6.1 Tipos de representación de circuitos eléctricos e instalaciones eléctricas

Dentro de las representaciones de circuitos eléctricos e instalaciones eléctricas existen dos grupos fundamentales la representación de multifilar y la representación unifilar. La representación multifilar está definida por la representación de todos los conductores con sus conexiones a los distintos elementos que intervienen en un circuito o en una instalación eléctrica. Son los que se utilizan normalmente en los montajes de los circuitos eléctricos. Para clarificar estos esquemas, siempre hay que separar el esquema de fuerza (circuito de alimentación a distintos receptores con sus elementos o sistemas de protección, por ejemplo motores, relés, térmicos, etc.); del

esquema de control, maniobra o mando (circuito al que van conectados los elementos a accionar a través de pulsadores, contactos auxiliares de contactores y relés, etc.). En la figura 1 se representa un esquema eléctrico multifilar, donde se han representado por tanto todos los hilos o conductores. Además y como es normal se ha separado el esquema de fuerza (o de alumbrado o de otros usos) del de mando, maniobra y control. El esquema de fuerza se corresponde con la alimentación a un motor trifásico conectado en estrella (en su placa de bornas) con posibilidad de cambio de sentido de giro (contactos principales de los contactores C1 y C2). El esquema de mando, de corriente alterna monofásica (fase T y neutro) alimenta a las bobinas de los contactores a través de pulsadores y de los contactos auxiliares, de mando o de maniobra de los contactores. Además en el esquema también podemos observar elementos de protección, regulación y medida.

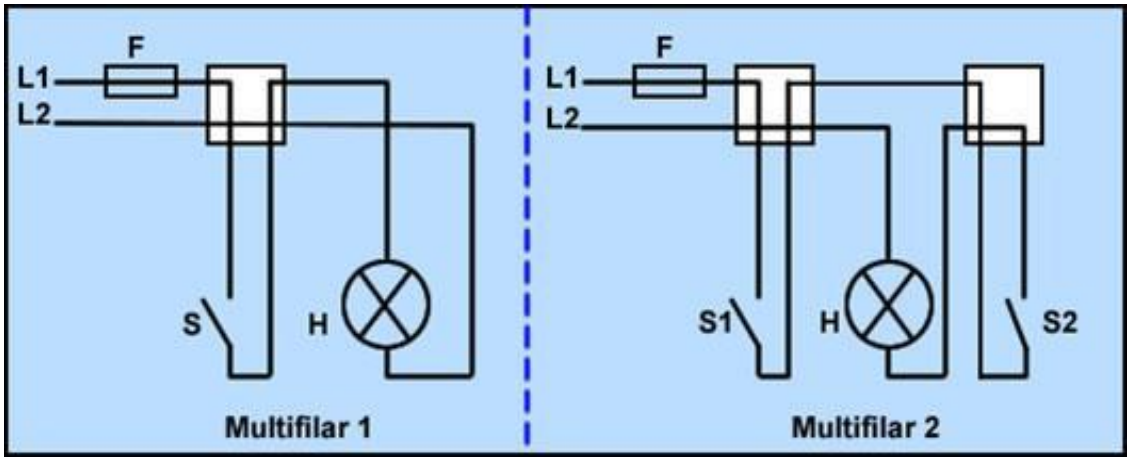

Fig.1 Esquema eléctrico representado de manera multifilar.

1.6.2 Representación de unifilar de circuitos eléctricos e instalaciones eléctricas

El termino de representación unifilar es el tema que más nos interesa abordar ya que el marco de trabajo de nuestro trabajo está basado fundamental en la obtención y gestión de este tipo de sistemas de visualización. La representación unifilar está definida por la esquematización visual de un circuito o de una instalación con un sólo hilo, utilizando la simbología y notaciones apropiadas para entender dicha representación. Es la que se suele utilizar en los “esquemas unifilares” que aparecen en los proyectos técnicos. Mientras con la multifilar se ve claramente un circuito con la unifilar se ve toda la instalación, por compleja que ésta sea.

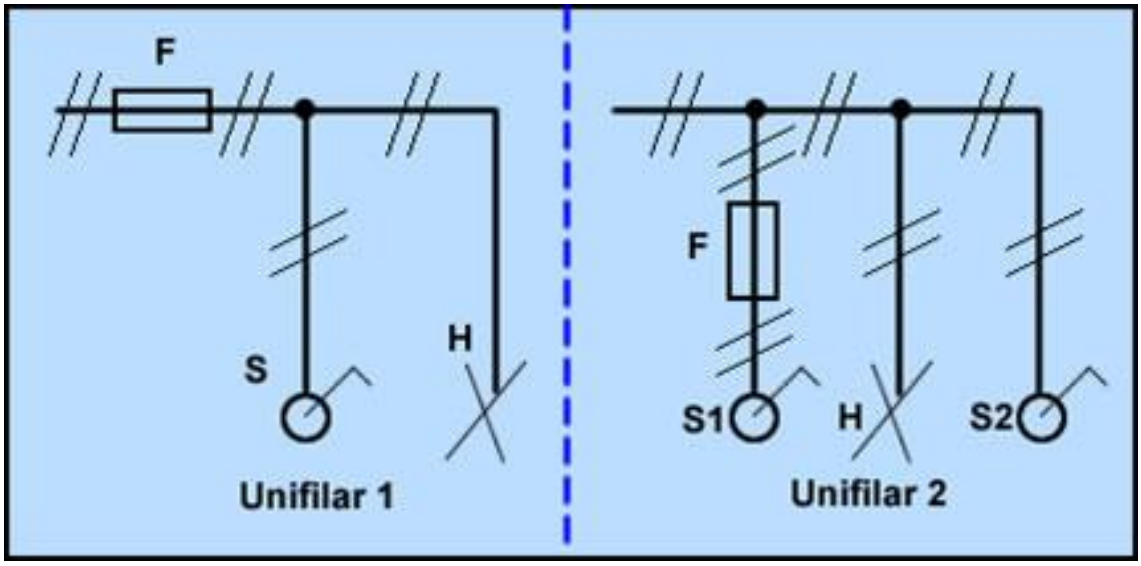

Fig. 2 Esquema eléctrico representado de manera unifilar.

1.7 Norma para la representación de esquemas unifilares

Los sistemas de esquemas unifilares se rigen por normas de representación tal es así que esto varía tanto como las áreas geográficas del mundo. Las normas se definen según las necesidades de las regiones y sectores donde se quieran implantar la misma. Un norma está definida como regla o conjunto de reglas que hay que seguir para llevar a cabo una acción, porque está establecido o ha sido ordenado de ese modo. Es por eso que podemos establecer como norma de representación de esquemas unifilares como el conjunto de reglas para representar un elemento eléctrico determinado que esa regla va desde su diseño hasta la dimensión del mismo.

1.7.1 Norma de la UNE (Unión Nacional Eléctrica).

En Cuba la UNE ha establecido como normativa para la representación de esquemas unifilares la norma UNE-EN 60617(ICE 60617) (2). La norma UNE-EN 60617(ICE 60617) está basada en la EN 60617(ICE 60617) (4) aprobada por el CENELEC (Comité Europeo de Normalización Electrotécnica). La normativa eléctrica para la representación de esquemas unifilares eléctricos de la UNE fue restablecida en el año 1996 a raíz de una reestructuración dentro del sistema eléctrico cubano. Esta normativa esta divida en trece partes fundamentales la primera es la introducción a la misma ya la restantes esta divididas entre los grupos de componentes eléctricos y

su representación las dos últimas partes fueron aprobadas en el año 1997 y 1998 respectivamente por él, esta división se refleja en la siguiente figura.

Parte	Descripción
UNE-EN 60617-2	Elementos de símbolos, símbolos distintivos y otros símbolos de aplicación general
UNE-EN 60617-3	Conductores y dispositivos de conexión
UNE-EN 60617-4	Componentes pasivos básicos
UNE-EN 60617-5	Semiconductores y tubos electrónicos
UNE-EN 60617-6	Producción, transformación y conversión de la energía eléctrica
UNE-EN 60617-7	Aparamenta y dispositivos de control y protección
UNE-EN 60617-8	Instrumentos de medida, lámparas y dispositivos de señalización
UNE-EN 60617-9	Telecomunicaciones : Conmutación y equipos periféricos
UNE-EN 60617-10	Telecomunicaciones : Transmisión
UNE-EN 60617-11	Esquemas y planos de instalación, arquitectónicos y topográficos.
UNE-EN 60617-12	Operadores lógicos binarios
UNE-EN 60617-13	Operadores analógicos

Fig.3 Partes de la Norma UNE-EN (ICE 60617).

1.8 Definición de gráficos unifilares

Los gráficos unifilares no son más que un compuesto grafico conformado por una primitiva gráfica básica o grupo de estas, dígase primitivas gráficas básicas circulo, rectángulo, cuadrado, línea entre otras; con el fin de representar un esquema unifilar que no solo puede ser eléctrico sino que también pueden ser utilizados en otros esquemas como esquemas unifilares fotográficos y en otros tan simples como la conformación de un figura geométrica compuesta.

1.9 Editor de gráficos unifilares

Un editor de gráficos unifilares es un herramienta con el fin de configurar, crear, “editar”, diseñar los nuevos elementos que por su definición pasaran a ser esquemas o diagramas unifilares. También es el marco de trabajo de un operador o usuario capacitado para la edición de estos elementos gráficos. El editor de gráficos unifilares de un sistemas SCADA eléctrico se encuentra en el modulo HMI del mismo

sistema lo que lo convierte en una herramienta fundamental y casi imprescindible dentro de dicho modulo.

1.10 Ejemplo de editor de gráficos unifilares

Existen un gran número de editores de gráficos unifilares, cada uno con las características específicas para lo cual fueron creados, a continuación se muestran algunos ejemplos de dichos editores.

1.10.1 Editor de gráficos unifilares de QElectrotech

Dentro del mundo de los editores de esquemas unifilares que existen, resalta QElectrotech. Este software fue creado a mediados del año 2009 por un grupo de desarrolladores franceses. Está actualmente disponible en versión 0.8. QElectrotech está bajo licencia GNU/GPL, se encuentra disponible para Windows y Linux. PEC le permite poner un elemento en un esquema de enlaces, los conductores. El esquema tiene una ventana de edición donde puede editar los datos. Puede mover y rotar los elementos. Los conductores que enlazan los elementos se colocan con el cursor. La ruta se puede modificar. Puede cambiar los parámetros de los conductores: pueden ser simples, con un cable o con varios cables esto puede ser de forma automática con los puntos de unión entre los conductores. También puede agregar campos independientes de texto en los esquemas.

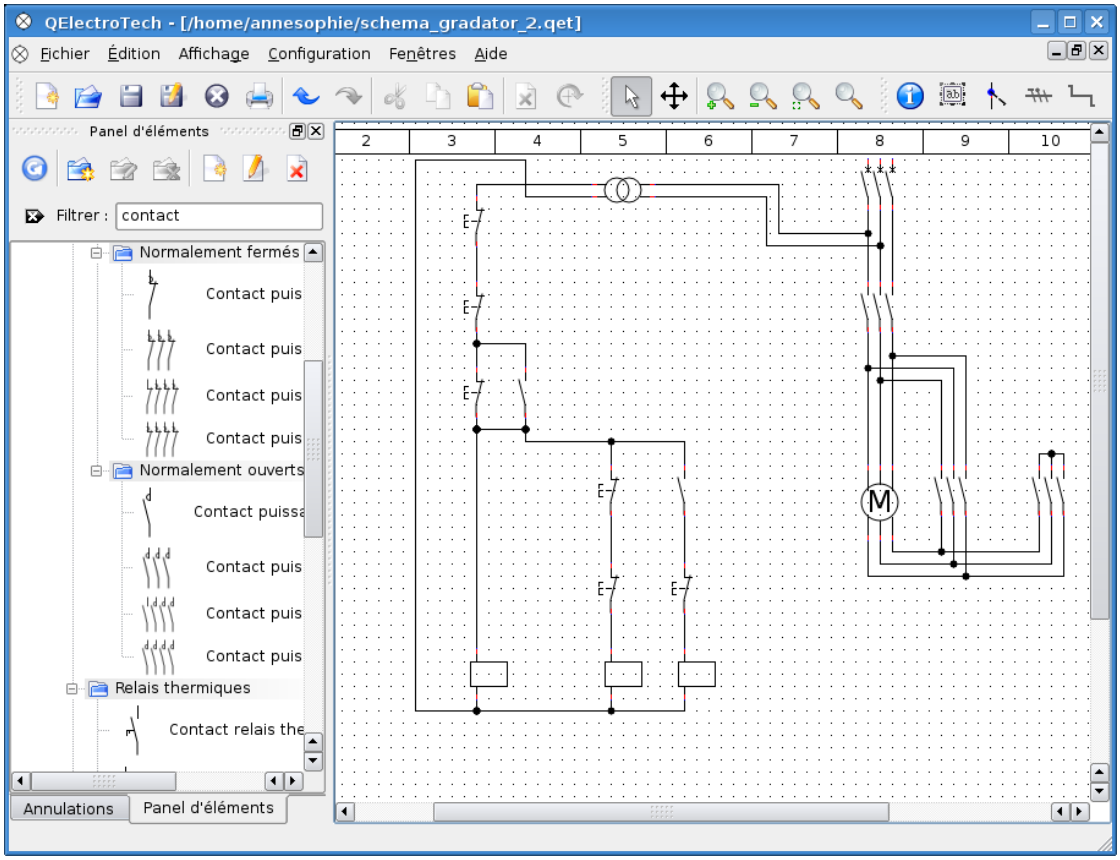

Fig.2 Editor QElectrotech

1.11 Tendencias y tecnologías

En la actualidad la tendencia de muchas empresas, instituciones y personas civiles, es el uso del software libre por su capacidad de proveer al usuario de todos los derechos sobre lo que produzca o cree haciendo uso del mismo. Es por ello que cada día crece el uso de Software Libre a nivel mundial, pues en la medida de que nos damos cuenta que los software realizados sobre plataformas propietarias, se encuentran frenadas por la licencias, además de no mostrar su código fuente, y el proceso de mantenimiento o cambio de versiones es realizado puramente por la compañía propietaria y en todos estos procesos se invierten grandes saldos de dinero para poder adquirir y dar mantenimiento a un producto determinado, por lo que la mejor opción es hacer uso de la tecnología antes mencionada, dada las características de las posibilidades económicas de nuestro país, y las posibilidades que brindan el uso Software Libre.

El desarrollo de los gráficos unifilares o componentes gráficos, se hará sobre plataforma libre, esto es primeramente garantía de que el producto final podrá ser

modificado por quien lo desee y de esta manera se le podrán hacer las adaptaciones o mejoras que se estimen convenientes, para ello es necesario utilizar herramientas, lenguajes de programación, bibliotecas gráficas, etc.

1.12 Biblioteca

En ciencias de la computación, una biblioteca (del inglés library) es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables. Ejecutables y bibliotecas hacen referencias (llamadas enlaces) entre sí a través de un proceso conocido como enlace, que por lo general es realizado por un software denominado enlazador (18).

La mayoría de los sistemas operativos modernos proporcionan bibliotecas que implementan la mayoría de los servicios del sistema. De esta manera, estos servicios se convierten en una "materia prima" que cualquier aplicación moderna espera que el sistema operativo ofrezca. Como tal, la mayor parte del código utilizado por las aplicaciones modernas se ofrece en estas bibliotecas.

1.12.1 Biblioteca GTK+

GTK+ es la abreviatura de *GIMP toolkit* (conjunto de rutinas para GIMP). GTK+ es un grupo de bibliotecas para desarrollar interfaces gráficas de usuario. Esta sustentada bajo la licencia LGPL, licencia de distribución de software libre. Además este conjunto de rutinas es multiplataforma y parte importante del proyecto GNU (4). Actualmente es muy usada por muchos programas en los sistemas GNU/Linux. GTK+ se ha diseñado para permitir programar con lenguajes como C, C++ (gtkmm), C#, Java, Perl, PHP o Python (5).

GTK+ se basa en varias bibliotecas del equipo de GTK+ y de GNOME (GTK):

- **GTK.** Biblioteca la cual realmente contiene los objetos y funciones para crear la interfaz de usuario. Maneja *widgets* como ventanas, botones, menús, etiquetas,

deslizadores, pestañas, etc.

- **GDK.** Biblioteca que actúa como intermediario entre gráficos de bajo nivel y gráficos de alto nivel.

- **ATK**. Biblioteca para crear interfaces con características de una gran accesibilidad muy importante para personas discapacitadas o minusválidas. Pueden usarse utilerías como lupas de aumento, lectores de pantalla, o entradas de datos alternativas al clásico teclado y mouse.

- **Pango**. Biblioteca para el diseño y renderizado de texto, hace hincapié especialmente en la internacionalización. Es el núcleo para manejar las fuentes y el texto de GTK+2.

- **Cairo**. Biblioteca para gráficos en 2D con soporte para múltiples dispositivos de salida (incluyendo el sistema X Windows, Win32) mientras aprovecha la aceleración del hardware si está disponible.

- **GLib.** Biblioteca de bajo nivel, estructura básica de GTK+ y GNOME. Proporciona manejo de estructura de datos para C, portabilidad, interfaces para funcionalidades de tiempo de ejecución como ciclos, hilos, carga dinámica o un sistema de objetos.

1.12.2 Biblioteca Qt

Es un producto de la empresa noruega de software Trolltech AS. Esta biblioteca comenzó a distribuirse comercialmente en 1996 y desde entonces ha sido la base para numerosas aplicaciones incluyendo la popular interfaz gráfica para GNU/Linux llamada KDE.

Características:

- QT es una biblioteca para la creación de interfaces gráficas. Se distribuye bajo una licencia libre GPL.

- Se encuentra disponible para una gran número de plataformas: GNU/Linux, MacOs X, Solaris, HP-UX, Windows. Además, existe también una versión para sistemas empotrados.

- Es orientado a objetos, lo que facilita el desarrollo de software. El lenguaje para el que se encuentra disponible es C++ aunque han aparecido versiones (conocidas como bindings) para C, Python (PyQt), Java (Qt Jambi), Perl (PelrQt), entre otros.

- Provee un mecanismo para la comunicación entre objetos basado en señales y ranuras (signals-slots).

- Provee herramientas para el pintado de objeto gráficos 2D.

- Ofrece interfaces para el trabajo con tecnología XML.

- Brinda la posibilidad del trabajo a partir de extensiones (Plugins), bibliotecas dinámicas (Imágenes, formatos, ...) .

1.13 Lenguaje de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación (15).

1.13.1 Lenguaje de programación Python

Python es un lenguaje de programación de alto nivel cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje de programación multi-paradigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un interpretado, usa tipado dinámico, es fuertemente tipado y es multi-plataforma. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores (16).

1.13.1.1 Características de Python

- Resolución dinámica de nombres; es decir, lo que enlaza un método y un nombre de variable durante la ejecución del programa (también llamado ligadura dinámica de métodos).
- Facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.
- Los tipos de datos en Python permiten expresar operaciones más complejas en sentencias cortas.
- El agrupamiento de sentencias se realiza por identación en lugar de llaves de principio y fin
- No se necesita de la declaración de variables.
- Python es extensible por lo que puede ser utilizado su lenguaje en extensiones de C (18).

1.13.2 Lenguaje de programación C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. Su creación se basó sobre la necesidad extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. Desde el punto de vista de los lenguajes orientados a objetos, es un lenguaje híbrido.

C++ ofrece un control de tipos y análisis en tiempo de compilación. Declara funciones de modo que el compilador pueda controlar su uso. La necesidad del preprocesador ha sido prácticamente eliminada para sustitución de valores y macros, que eliminan muchas dificultades para encontrar errores. Las referencias permiten un manejo más conveniente de direcciones para argumentos de funciones y retorno de valores. El manejo de nombres se mejora a través de la sobrecarga de funciones, que le permite usar el mismo nombre para diferentes funciones. La mayoría de las aplicaciones requieren algún grado de eficiencia, de modo que C++ siempre será una buena opción cuando se quiere mayor eficiencia. Algunas características en C++ facilitan el rendimiento y eficiencia. El programador para favorecer esto debe adoptar algunas sencillas precauciones:

1.13.2.1 Características de C++

- Usar enteros (**int**) con preferencia sobre cualquier otro tipo de variable numérica. En especial en los contadores de bucles. Las operaciones con enteros son del orden de 10 a 20 veces más rápidas que las de números en coma flotante.
- Usar operadores incremento y decremento ++/--.
- Usar variables de registro, en especial en los bucles críticos, sobre todo si son anidados.
- Usar aritmética de punteros frente a subíndices de matrices.
- En problemas de computación numérica recordar que el cálculo de funciones trascendentes es por lo general muy lento.
- Usar referencias para argumentos y valores devueltos en funciones, antes que objetos "por valor".
- Al definir clases utilizar al mínimo las funciones virtuales así como los punteros a funciones-miembro.
- Sustituir **inline** en funciones definidas por el usuario.
- Prestar atención al modo de uso de aquellas funciones de librería que se presentan en dos versiones (13).

1.14 Metodologías de desarrollo

Las metodologías de desarrollo surgen por la necesidad de evitar problemas que se presentan en la producción de un software por no seguir normas específicas. Son un conjunto de procedimientos, técnicas y ayudas para el desarrollo de productos software. Cubren todo el ciclo de desarrollo del producto, estableciendo etapas y controles a aplicar en cada momento. Estas recopilan un conjunto de técnicas y procedimientos en cada una de las fases que las componen (3).

1.14.1 Metodología de desarrollo de Software OpenUP

El OpenUP es un Proceso Unificado que aplica propuestas de gestión ágil como son desarrollo iterativo e incremental dentro del ciclo de vida, tratando de ser manejable en relación con el RUP, ya que mantiene sus características esenciales. OpenUP es un proceso de desarrollo de software completo debido a que puede ser manifestado como todo el proceso para construir un sistema. Es extensible ya que en el proceso se puede agregar o adaptar según lo vayan requiriendo los sistemas.

OpenUP es ligero y proporciona una comprensión detallada del proyecto, beneficiando a clientes y desarrolladores sobre productos a entregar y su formalidad.

Su desarrollo es dirigido por casos de uso. OpenUP como metodología de desarrollo es conducida por el principio de colaboración para alinear intereses y para compartir su comprensión. Es un proceso para pequeños equipos de desarrollo que valoran los beneficios de la colaboración y de los involucrados con el resultado del proyecto, por encima de formalidades innecesarias (6).

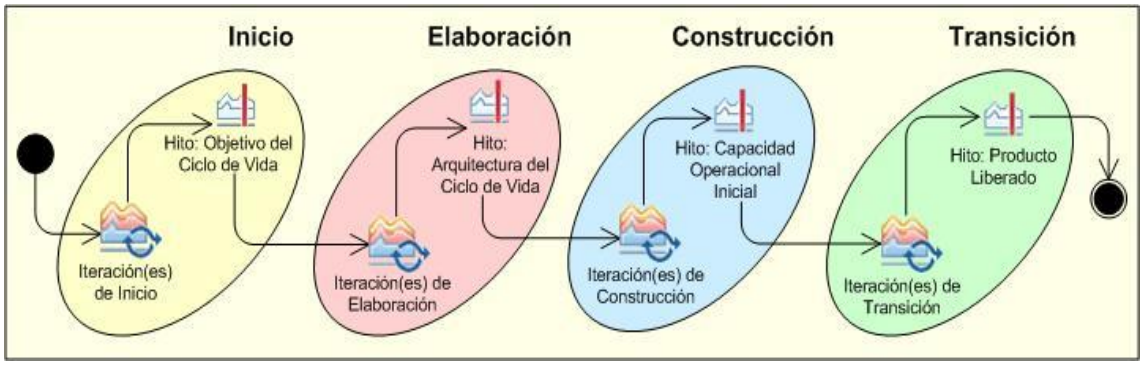

Fig.4 Ciclo de vida OpenUP

1.14.2 Metodología de desarrollo de Software RUP

El Proceso Unificado de Software o RUP, acrónimo de Rational Unified Process, es una metodología tradicional para el desarrollo de productos de software que permite la realización del análisis y el diseño orientado a objetos. Esta además de ser un proceso, es un marco de trabajo extensible, ya que puede ser adaptado a proyectos y organizaciones específicas. A través de una serie de actividades transforma los requerimientos de los usuarios de forma tal que se obtiene un sistema de software que satisfaga los mismos (7).

Características principales de RUP:

- RUP está dirigido y guiado por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

El ciclo de vida de RUP cuenta con 4 fases fundamentales: inicio, elaboración, construcción y transición.

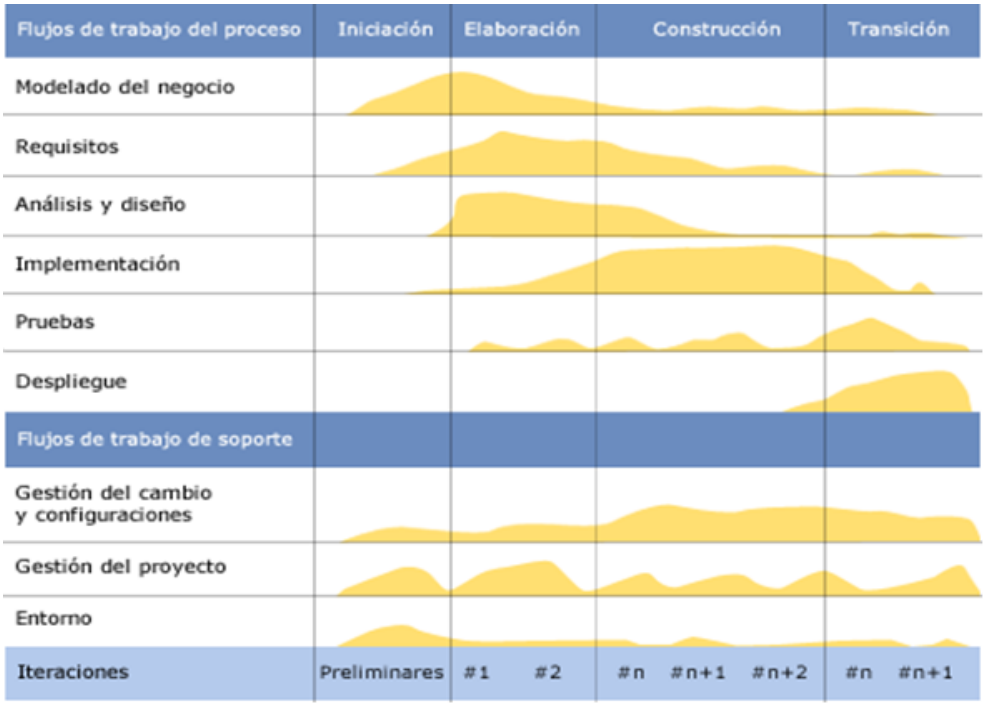

Fig.5 Ciclo de vida de RUP

1.15 Herramientas ingeniería asistida por computadora

Las herramientas CASE, acrónimo de Computer Aided Software Engineering, son utilizadas hoy en día con el objetivo de desarrollar software, ya que estas están diseñadas y definidas con el fin de automatizar los aspectos claves de todo el proceso de desarrollo del sistema. CASE proporciona un conjunto de herramientas automatizadas que están desarrollando una cultura de ingeniería. Uno de los objetivos más importante a largo plazo es conseguir la generación automática de programas desde una especificación a nivel de diseño (8).

1.15.1 Umbrello

Una de las herramientas más utilizadas para el análisis de la arquitectura del software y el diseño de la misma es el nombrado Umbrello UML Modeller que no es más que una herramienta de diagramas. Umbrello facilita la creación de un producto de alta calidad, también puede usarse para documentar diseños de software ayudando así a los desarrolladores. Originalmente llamado UML Modeller, en septiembre de 2002, el proyecto cambió el nombre de UML Modeller a Umbrello, por ser un nombre muy genérico. En Febrero del 2004 el proyecto se incorpora a la suite de KDE, permitiendo la inclusión de más desarrolladores en el mismo. Actualmente Umbrello

permite instalarse en diferentes plataformas y posee más de 30 idiomas diferentes, gracias a su licencia original GPL.

Cuenta con una Interfaz Gráfica que posee tres áreas, estas áreas reciben el nombre de: vista en árbol, área de trabajo y ventana de documentación.

Umbrello UML Modeller incluye soporte para los siguientes lenguajes: C++, Java, C#, PHP, JavaScript, SQL, Pascal, Python, etc. Soporta un gran número de diagramas como: diagrama de casos de uso, diagrama de componentes, diagrama de despliegue, diagrama de clases, entre otros (9).

1.15.2 Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Cuenta con una licencia gratuita y comercial, es un producto de calidad (10).

1.16 Conclusiones del Capítulo

En este capítulo se realizo un resumen de lo que es un SCADA y más en profundidad lo que es un SCADA Eléctrico, de su estructura así como sus principales componentes y de los módulos que lo componen, haciendo énfasis en el modulo HMI (Interfaz Hombre Maquina).También se hicieron referencias de las principales tendencias y tecnologías que se usan hoy en día. Además se realizo un resumen de las principales herramientas de desarrollo y de gestión de la ingeniería de software; todas enfocadas a ser las posibles tecnologías y herramientas a utilizar en el desarrollo de la propuesta de solución de nuestro trabajo.

Capítulo 2: Descripción de la solución propuesta.

2.1 Introducción

En este capítulo se abordarán las características de las tecnologías seleccionadas para el desarrollo de los de los gráficos unifilares del módulo de Visualización del SCADA Eléctrico. Además tendremos los requisitos generales para la gestión de los gráficos unifilares. Se describen las clases y métodos más importantes.

2.2 Selección de la tecnología a utilizar

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

2.2.1 Software Libre. GNU/Linux

El Software libre brinda libertad a los usuarios sobre el producto adquirido y por tanto, el mismo puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software; de modo más preciso, se refiere a cuatro libertades de los usuarios del software: la libertad de usar el programa, con cualquier propósito; de estudiar el funcionamiento del programa, y adaptarlo a las necesidades; de distribuir copias, con lo que puede ayudar a otros; de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie (para la segunda y última libertad mencionadas, el acceso al código fuente es un requisito previo).

Una distribución de Linux es una variante de ese sistema operativo (SO) que incorpora determinados paquetes de software para satisfacer las necesidades de un grupo específico de usuarios, dando así origen a ediciones hogareñas, empresariales y para servidores. Pueden ser exclusivamente de software libre, o también incorporar aplicaciones o controladores propietarios.

Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU (acrónimo que significa GNU No es Unix); de ahí el nombre: GNU/Linux (11).

La distribución de Linux seleccionada es Ubuntu, específicamente la versión 9.10 Karmic Koala. Ubuntu es una distribución Linux basada en Debian GNU/Linux que proporciona un sistema operativo actualizado y estable para el usuario promedio, con un fuerte enfoque en la facilidad de uso y de instalación del sistema. Se compone de múltiples paquetes de software normalmente distribuidos bajo una licencia libre o de código abierto.

2.2.2 Eclipse

Eclipse fue el ambiente de desarrollo integrado seleccionado. Ha sido utilizado para desarrollar todo tipo de aplicaciones, y altamente probado en cualquier ambiente, ya sea en la construcción de Servicios Web, aplicaciones de escritorio, juegos, etc. El Eclipse es una gran estructura formada por un núcleo y muchos plugins que van conformando la funcionalidad final. La forma en que los plugins interactúan es mediante interfaces o puntos de extensión; así, las nuevas aportaciones se integran sin dificultad y sin conflictos. Como se refleja en lo anterior el Eclipse tiene una gran capacidad y versatilidad, lo cual es permitido por su arquitectura. La web oficial de Eclipse define al mismo como “una plataforma (IDE), abierta para todo y para nada en particular”. Eclipse es una plataforma porque no se encuentra acabada en su totalidad, pero está diseñado para que sea extensible indefinidamente con la adecuada implementación de plugins. La característica clave de Eclipse es la extensibilidad (12).

Quizás lo más interesante de Eclipse es ser completamente neutral a la plataforma y al lenguaje. Qt ha lanzado un módulo con el que es posible integrar Qt en Eclipse. Incluye opciones integradas de depuración, integración con Qt Designer, editor de archivos de proyectos e incluso un editor de recursos. La integración de Qt con este IDE funciona muy bien, sin embargo el gran consumo de recursos necesarios para su correcto funcionamiento es su mayor punto negativo.

2.2.3 Biblioteca Qt

Además de las características expuestas en el capítulo 1, la selección de esta biblioteca gráfica se fundamenta en

- **El Framework Graphics View.** Este framework proporciona mecanismos para la visualización de gráficos en 2D. Incluye además una arquitectura de propagación de eventos que permite la interacción precisa con los objetos. Los objetos se pueden manejar por eventos del teclado y del cursor.

Esta biblioteca usa una estructura BSP, acrónimo de Binary Space Partitioning; con el objetivo de proveer un rápido repintado de los objetos que se están visualizando.

- **Escena gráfica:** Para la visualización y manejo de los componentes gráficos esta biblioteca provee un mecanismo encabezado por el modelo QGraphicsScene. Este modelo se encarga fundamentalmente de manejar los eventos sobre los componentes gráficos brindando un marco de configuración de estos elementos. Este mecanismo esta relacionado con el framework Graphics View, este último se encarga de visualizar los elementos que se encuentran dentro de la escena gráfica asi como la misma escena.

- **Pintado de los objetos gráficos:** Provee un mecanismo de pintado de los objetos gráficos denominado QPainter que permite el dibujado de estos componentes, facilitando la concepción de los mismos. Además este proceso permite predefinirle algunas propiedades gráficas a los elementos como el fondo, el borde entre otros.

- **El sistema de propiedades:** Proporciona un sofisticado sistema de propiedades gráficas, asociadas fundamentalemente el control de la meta información, término que se maneja de manera enficiente dentro de la biblioteca. Está basado en el sistema Meta-Object que brinda un grupo de funcionalidades y herramientas para el manejo la información de los componentes gráficos.

2.2.4 Lenguaje de programación C++

C++ fue diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje multi-paradigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen

también algunos intérpretes, tales como ROOT. Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales. C++ permite trabajar tanto a alto como a bajo nivel (13).

2.2.5 Metodología OpenUp

Es un proceso de desarrollo iterativo del software que es mínimo porque solo incluye el contenido del proceso fundamental el cual no deja de ser completo debido a que el mismo puede ser manifestado como proceso entero para construir un sistema. Además es extensible debido a que puede ser utilizado como base para agregar o para adaptar más procesos (14). OpenUp es utilizada por desarrolladores de alto nivel en casi todo el mundo por sus altas cualidades administrativas. Entre sus beneficios se encuentran: permitir disminuir las probabilidades de fracaso, incrementar las probabilidades de éxito y permitir detectar errores tempranos a través de un ciclo iterativo (6).

2.2.6 Visual Paradigm

Es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Algunas de las funcionalidades que ofrece son:

- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación. Además brinda un modelo y código que permanecen sincronizado en todo el ciclo de desarrollo Lo que permite que el grupo de desarrolladores se rijan bajo una misma estructura lo que hace eficiente el trabajo de la línea.

- Capacidades de ingeniería directa, estableciendo la manera de elaborar el marco ingenieril del proceso de desarrollo del software. Brinda la funcionalidad de crear una estructura del software de manera que se puede hasta generar hasta el código del sistema en general. Además brinda un mecanismo para la generación de una documentación con toda la información del proceso.

- Disponibilidad de integrarse en los principales IDE. Esta herramienta posee la capicidad de integración con los principales IDE que existen algunos como Visual Studio y muy fundamental con el selecionado por nuestro grupo de trabajo Eclipse lo que permite la generación de la documentación para el desarrollo de manera eficiente.

- Disponibilidad en las plataformas Windows y UNIX (10). Está disponible para varios sistemas operativos lo que facilita el trabajo y manejo de esta herramienta desde varios entornos.

2.3 Selección de modelo de representación de los gráficos

El modelo de representación que se utilizará es la representación unifilar que no es más que un modelo que establece una representación mono-lineal de los conectores de los componentes gráficos de un diagrama o esquema eléctrico, partiendo fundamentalmente de primitivas básicas graficas, díganse primitivas básicas círculo, recta entre otras.

2.4 Requerimientos funcionales para la gestión de los gráficos unifilares.

Los requisitos funcionales son capacidades o condiciones que los gráficos unifilares y el editor de gráficos unifilares deben cumplir, los requerimientos serán vistos de manera general, esto se muestra en la siguiente tabla:

Tabla Requisitos funcionales

Requisitos funcionales	**Descripción**
General	Los requisitos generales son aquellos que deben cumplir tanto todos los gráficos unifilares, sean básicos ó avanzados, así como el editor de gráficos unifilares.
RF 1 Gestionar el gráfico unifilar	El sistema debe permitir gestionar el gráfico unifilar que no es más que el proceso de

	crear, modificar o eliminar el gráfico dentro del marco de edición del mismo.
RF 2 Definir propiedades del gráfico unifilar.	El sistema debe permitir definir las propiedades para un gráfico unifilar.
RF 3 Adicionar propiedades dinámicas al gráfico unifilar	El sistema debe permitir adicionar propiedades dinámicas al gráfico unifilar de manera que el mismo quede asociado a un variable técnica del sistema.
RF 4 Modificar propiedades dinámicas al gráfico unifilar	El sistema debe permitir modificar las propiedades dinámicas del gráfico unifilar.
RF 5 Salvar el gráfico unifilar	El sistema debe permitir salvar el gráfico unifilar en el formato predeterminado en el proceso de concepción del editor de gráficos unifilares.
RF 6 Cargar el gráfico unifilar	El sistema debe permitir cargar el gráfico unifilar desde un directorio determinado dentro del sistema.

2.5 Requerimientos no funcionales para la gestión de gráficos unifilares.

Requerimientos de Usabilidad

RNF 1: El mecanismo debe estar basado en un modelo de ventana o interfaces visuales que le permitan al usuario u operador interactuar de manera sencilla con el sistema.

RNF 2: Los colores de los componentes gráficos deben ser *negro, rojo o azul* o la combinación de estos.

RNF 3: El marco eléctrico debe mostrarse todo el tiempo en el centro de la interfaz visual.

2.6 Diagrama de casos

Los diagramas de casos de uso son una especie de diagrama de comportamiento estos últimos no son más que un elemento que se emplean para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema. Los aspectos dinámicos de un sistema de software involucran cosas tales como el flujo de mensajes a lo largo del tiempo y el movimiento físico de componentes en una red. Un diagrama de casos de uso también representa un conjunto de casos de uso y actores y además las relaciones de dichos actores con los casos de uso (17).

2.6.1 Diagrama de casos del sistema

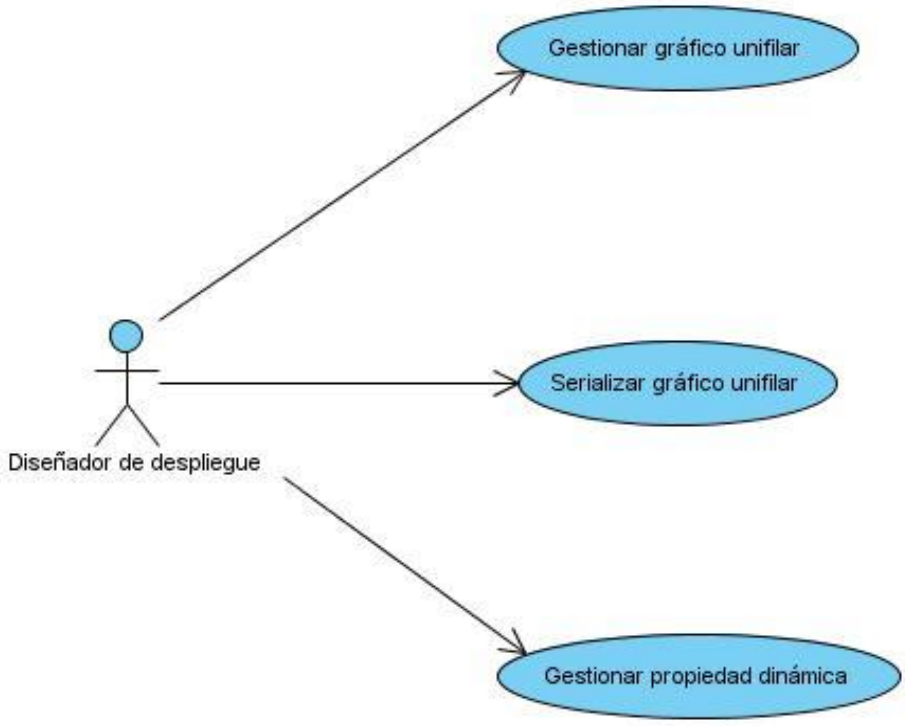

Fig.6 Diagrama de casos de uso del sistema.

2.7 Descripción de casos de uso del sistema

2.7.1 Definición de los actores del sistema

Actor	Descripción
Diseñador de despliegue	Individuo que tiene la tarea de supervisar, controlar y ejecutar las acciones sobre la aplicación o sistema de edición. Además tiene la tarea del proceso de edición de los componentes gráficos. El ambiente del Diseñador de Despliegue es la Sala de Edición, este es el responsable de los ambientes de edición, dentro del sistema.

2.7.2 CU Gestionar gráfico unifilar

Nombre del caso de uso	Gestionar gráfico unifilar.
Actores	Diseñador de despliegue.
Propósito	Proceso de crear, modificar y eliminar el gráfico unifilar.
Resumen: El caso de uso comienza cuando el diseñador de despliegue escoge la opción de crear, modificar o eliminar el gráfico unifilar.	
Referencias	RF 1, RF 2
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
a) Si el actor desea crear el gráfico unifilar ir a la sección “crear”. b) Si el actor desea modificar el gráfico unifilar ir a la sección “modificar”. d) Si el actor desea modificar el gráfico unifilar ir a la sección “eliminar”.	
Sección: “crear”	
Acción del actor	Respuesta del sistema
1- El diseñador de	2- El sistema muestra un editor de

despliegue escoge la opción de crear nuevo gráfico unifilar. 3 - El diseñador de despliegue comienza el proceso de creación del gráfico unifilar.	gráficos unifilares. El editor estará conformado por ventana, la misma poseerá una barra de herramientas , otra de componentes gráficos y una tercera de funcionalidades para el trabajo en términos de creación de los gráficos unifilares
Sección: “modificar”	
Acción del actor	Respuesta del sistema
1- El diseñador de despliegue escoge la opción de modificar el gráfico unifilar. 3- El diseñador de despliegue realiza modificaciones sobre el gráfico unifilar.	2- El sistema muestra la opción de modificar dándote la opción de modificar el gráfico unifilar tanto el que está en proceso de creación así como alguno que exista dentro del sistema.
Sección: “eliminar”	
Acción del actor	Respuesta del sistema
1- El diseñador de despliegue escoge la opción de eliminar el gráfico unifilar. 3- El diseñador de despliegue escoge elimina el gráfico unifilar.	2- El sistema muestra la opción de eliminar dándote la opción de eliminar el gráfico unifilar tanto el que está en proceso de creación o modificación así como alguno que exista dentro del sistema. 4- Muestra el mensaje de que ha sido eliminado el elemento.
Precondiciones	Tanto para los procesos de modificar y eliminar el gráfico unifilar debe existir dentro del sistema.
Poscondiciones	El gráfico unifilar es creado, modificado y eliminado de manera satisfactoria.

2.7.3 CU Gestionar propiedad dinámica.

<table>
<tr><td colspan="2">Nombre del caso de uso</td><td>Gestionar propiedad dinámica</td></tr>
<tr><td colspan="2">Actores</td><td>Diseñador de despliegue.</td></tr>
<tr><td colspan="2">Propósito</td><td>Proceso de adicionar o modificar propiedades dinámicas del gráfico unifilar.</td></tr>
<tr><td colspan="3">Resumen: El caso de uso comienza cuando el diseñador de despliegue escoge la opción de adicionar o modificar una nueva propiedad dinámica o varias nuevas propiedades dinámicas al gráfico unifilar.</td></tr>
<tr><td colspan="2">Referencias</td><td>RF 3, RF 4</td></tr>
<tr><td colspan="3">Curso normal de los eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">a) Si el actor desea adicionar una nueva propiedad dinámica al gráfico unifilar ir a la sección “adicionar propiedad”.
b) Si el actor desea modificar una nueva propiedad dinámica al gráfico unifilar ir a la sección “modificar propiedad”.</td><td></td></tr>
<tr><td colspan="3">Sección “adicionar propiedad”</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del sistema</td></tr>
<tr><td colspan="2">1- El diseñador de despliegue escoge la opción de adicionar propiedad dinámica al gráfico unifilar.
3- El diseñador de despliegue introduce el nombre de la nueva propiedad y el valor de la misma en lo campos pertinentes.</td><td>2- El sistema mostrará un ventana donde le dará la opción del diseñador de despliegue de introducir el nombre de la nueva propiedad y el valor de la misma, la ventana estará compuesta por dos campos:
Property Name: Campo para escribir el nombre de la nueva propiedad.
Value: Campo para introducir el valor de la nueva propiedad.
4- El sistema muestra verifica la valides de los datos introducidos por el diseñador de</td></tr>
</table>

	despliegue, en caso que lo datos no sean correctos el sistema muestra un mensaje de que los datos introducidos no son correctos e inicia el proceso automáticamente, en caso de que estos datos son correctos muestra un mensaje que la propiedad ha sido adicionada.
Sección “modificar propiedad”	
Acción del actor	Respuesta del sistema
1- El diseñador de despliegue escoge la opción de modificar una propiedad dinámica o varias propiedades dinámicas al gráfico unifilar. 3- El diseñador de despliegue realiza los cambios y modificaciones sobre la propiedad dinámica o las propiedades dinámicas del gráfico unifilar.	2- El sistema mostrara una ventana donde le dará la opción del diseñador de despliegue de modificar las propiedades dinámicas del elemento. 4- El sistema muestra un mensaje de que se realizaron las modificaciones al gráfico unifilar.
Precondiciones	Para el proceso de adición de la nueva propiedad el gráfico unifilar debe existir dentro del sistema. Para el proceso de modificación de la propiedad dinámica o propiedades dinámicas debe o deben estar dentro de las propiedades del elemento.
Poscondiciones	La propiedad dinámica es adicionada o modificada de manera satisfactoria.

2.7.4 CU Serializar gráfico unifilar

Nombre del caso de uso	Serializar gráfico unifilar.
Actores	Diseñador de despliegue.

Propósito	Proceso de guardar y cargar el gráfico unifilar
Resumen: El caso de uso comienza cuando el diseñador de despliegue escoge la opción de guardar o cargar el gráfico unifilar.	
Referencias	RF 5, RF 6
Curso normal de los eventos	
Acción del actor	Respuesta del sistema
a) Si el actor desea guardar el gráfico unifilar ir a la sección “guardar”. b) Si el actor desea cargar el gráfico unifilar ir a la sección “cargar”.	
Sección: “guardar”	
Acción del actor	Respuesta del sistema
1- El diseñador de despliegue escoge la opción de guardar el gráfico unifilar. 3 - El diseñador de despliegue escoge el directorio dentro del sistema donde va a ser guardado el gráfico unifilar.	2- El sistema muestra ventana, dándole la opción al diseñador de despliegue de guardar el gráfico unifilar en un directorio determinado dentro del sistema. 4- El sistema muestra un mensaje de que el gráfico unifilar has sido guardado.
Sección: “cargar”	
Acción del actor	Respuesta del sistema
1- El diseñador de despliegue escoge la opción de cargar el gráfico unifilar. 3- El diseñador de despliegue escoge el directorio dentro del sistema donde se encuentra almacenado el gráfico	2- El sistema muestra ventana, dándole la opción al diseñador de despliegue de cargar el gráfico unifilar desde un directorio determinado dentro del sistema. 4- El sistema carga el gráfico unifilar dentro de la herramienta de edición.

unifilar.	
Precondiciones	Para el proceso de cargar el elemento debe existir dentro algún directorio del sistema así como el directorio escogido debe ser válido.
Poscondiciones	El gráfico unifilar es guardado y cargado de manera satisfactoria.

2.8 Conclusiones del Capítulo

El Software libre brinda libertad a los usuarios sobre el producto adquirido y por tanto, el mismo puede ser usado, copiado, estudiado, modificado y redistribuido libremente. Este tipo de software nos brinda, a partir de sus características, la capacidad de generar un producto propio sin las travas que nos establecen las frenantes licencias y patentes de mantención que tiene el software privativo. Además con el desarrollo de un producto sobre el, la confiabilidad del mismo es mayor. El IDE Eclipse brinda la posibilidad de incorporarle extensiones para el trabajo en diferentes marcos. También esta bajo licencia libre lo que lo hace un herrmienta de trabajo muy factible para desarrolladores independientes. Qt es un framework potente y fundamentalmente para el trabajo con elementos gráficos; es por eso que es un herramienta muy eficiente para la creación y gestión de gráficos unifilares, brindando facilidad de concepción de los mismos y medios para hacerlos robustos. En este capítulo se presentó la tecnología seleccionada para el desarrollo de los gráficos unifilares. Se definieron los requerimientos funcionales para la gestión de estas componentes gráficas. Se hizo además una descripción de los casos de uso del sistema que facilitaran la gestión de los mismo.

Capítulo 3: Diseño de arquitectónico del sistema

3.1 Introducción

En este capítulo se hará una descripción de las principales características para la implementación de un mecanismo de gestión de gráficos unifilares dentro de un SCADA Eléctrico. Además se hace una descripción de la arquitectura del sistema y los patrones utilizados para el diseño e implementación del mecanismo de gestión de gráficos unifilares. También se describirán los paquetes y funcionalidades de las clases del sistema mediante diagramas.

3.2 Arquitectura de la solución

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos. Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea. La arquitectura del editor de gráficos unifilares es una solución basada en el modelo arquitectónico Modelo - Vista (Model - View). Este patrón describe una forma de desarrollar aplicaciones software separadas en dos capas, posibilitando un alto nivel de encapsulamiento de las responsabilidades y reduce al máximo el acoplamiento.

- **Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario así como el mecanismo para gestionar el paquete de modelo.

- **Modelo:** Es la representación específica de la información con la cual el sistema opera, la lógica de negocios y sus mecanismos de persistencia.

En el proceso de confección de la solución otro de los patrones de diseño que se utilizo fue el patrón Comando. Este patrón se basa en la creación de un grupo de operaciones genéricas o especificas. Estas operaciones estarán centralizadas es decir pueden ser utilizadas desde cualquier sector del sistema cuando las necesite. Además una de las ventajas fundamentales que tiene este patrón es la reutilización de código, ya que cada comando comprende toda la lógica para realizar una operación que el

mismo sabe hacer; lo que hace posible que cualquiera que esté interesado en realizar dicha operación solo tiene que invocar el comando adecuado.

En la siguiente figura se muestra la relación existente entre los patrones Modelo – Vista y el patrón Comando:

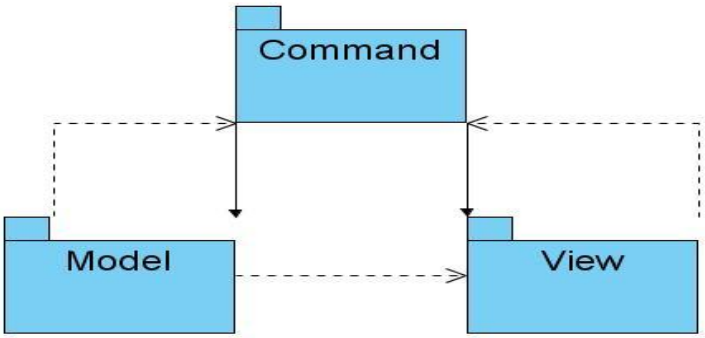

Fig. 7 Diagrama Modelo Vista Comando.

Otro de los patrones utilizados en el diseño de nuestro sistema es el patrón Composite. Este patrón se utiliza para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

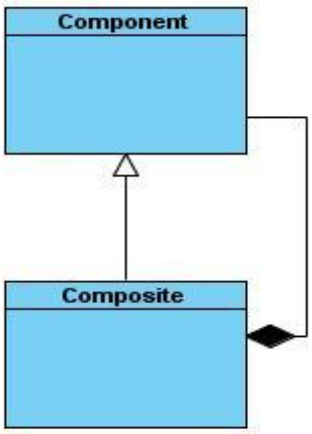

Fig.8 Diagrama patrón Composite.

3.3 Diagramas de clases

3.3.1 Diagrama de clases general del sistema

El diagrama general del sistema da una vista de cómo queda conformado el sistema de manera global. También ilustra las principales clases del sistema así como una agrupación de las clases, con el fin de que quede reflejado a que patrón están asociadas las mismas. Este diagrama está encabezado por la clase EditorView que no es más que la ventana de edición que se le mostrara al usuario, esta clase está compuesta entre otros elementos por objetos de las clases GraphicsScene y GraphicsView. La clase GraphicsScene forma parte del paquete Model, y maneja los eventos sobre los elemento dentro del marco de edición. La clase GraphicsView esta manipula está encargada de visualizar los elementos que se manejan dentro de la clase GraphicsScene, la clase GraphicsView está dentro del paquete View. Otras de los elementos que se encuentran dentro de este diagrama es Command que es un agrupamiento de todos los comandos que se manejan dentro del sistema. También en el diagrama se refleja la clase UnifilarGraphics que es uno de los principales resultados de la investigación e implementación del sistema ya que recoge todo lo que se maneja sobre los gráficos unifilares dentro del sistema.

En la siguiente figura se muestra el diagrama general del sistema, en el mismo se hace un agrupamiento de manera tal que quedan reflejados los patrones de diseño mencionados en anteriores epígrafes.

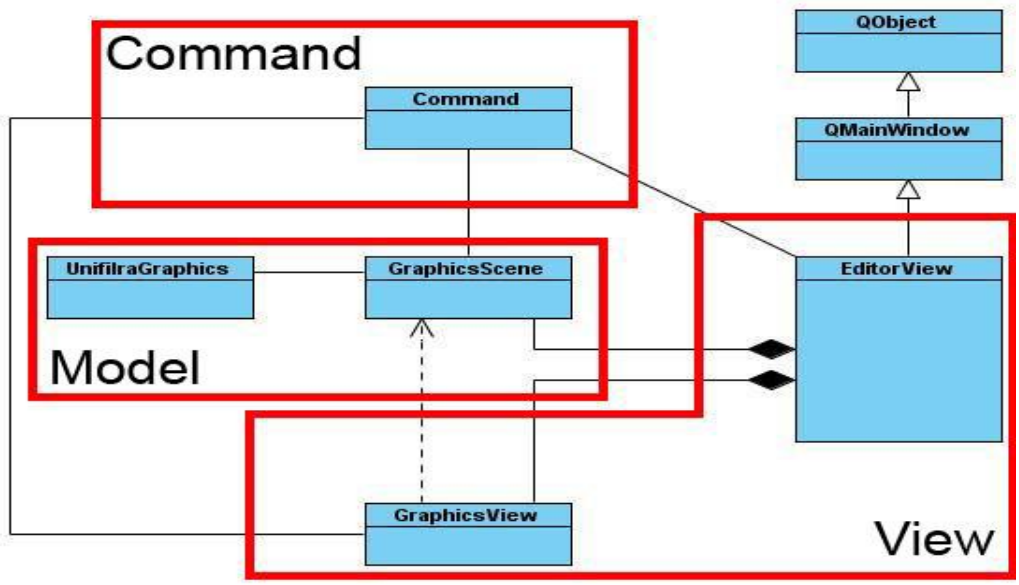

Fig. 9 Diagrama general del sistema.

3.3.2 Diagrama de clases Gráficos Unifilares

Este diagrama refleja el grupo de clase que se diseñaron para conformar de manera técnica el concepto de gráfico unifilar dentro de nuestro sistema. La clase GraphicsObject es la clase que recoge las funcionalidades generales de los gráficos dentro del sistema. Otras de las clases más importantes de este diagrama es GraphicsGroup esta clase un abstracción de la clase QGraphicsItemGroup de las paquete de clases de la librería Qt. La principal clase de este esquema es UnifilarGraphics ya que esta es el principal resultado de la investigación dentro de nuestro trabajo esta clase está basada en el patrón Composite, anteriormente mencionado, es en esta clase donde queda conformado el gráfico unifilar.

En la siguiente figura se representa el diagrama gráficos unifilares.

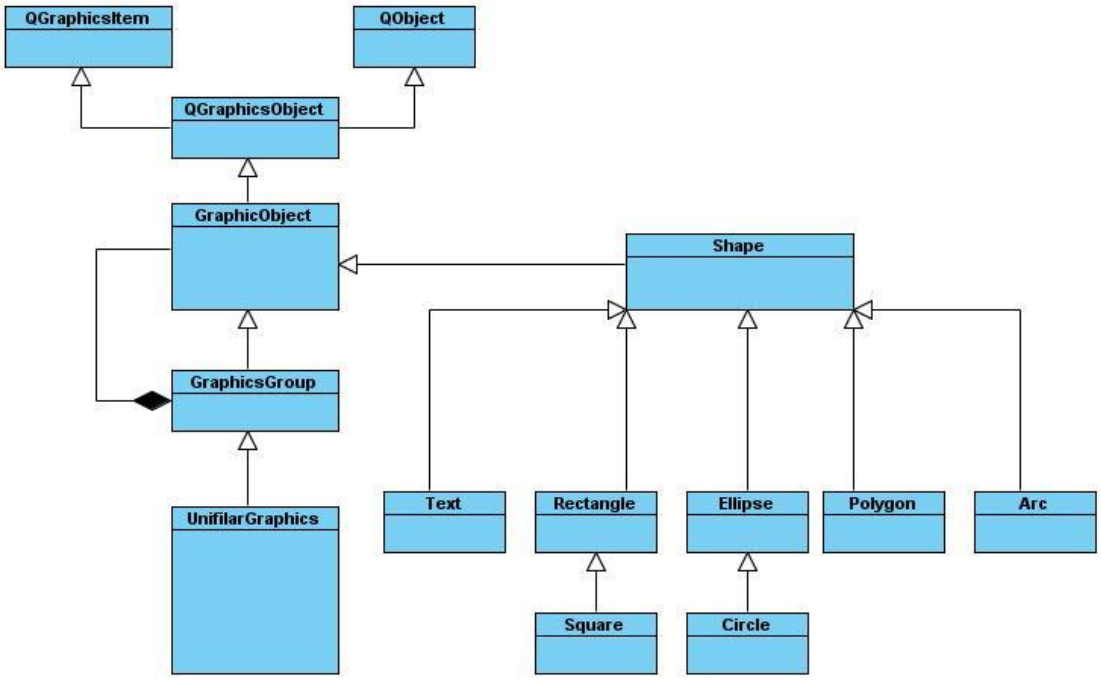

Fig.9 Diagrama de gráficos unifilares.

3.3.2.1 Descripción de la clase GraphicObject

Nombre: GraphicObject
Descripción: Esta clase es el punto más alto de la jerarquía de clases de los gráficos. En la misma se reúnen atributos generales de todos los objetos gráficos. También se recogerán las propiedades gráficas de manera general. Además tendrá implementado el método de salvado y cargado, desde un extensión XML predefinida en el sistema,

de los objetos gráficos que se manejan dentro de la aplicación.	
Tipo de clase:	
Atributo	Tipo
height	double
width	double
Para cada responsabilidad:	
Nombre	double getHeight()
Descripción	Retorna el valor de la altura o height del objeto gráfico.
Nombre	void seHeight(double pheight)
Descripción	Establece la manera de modificar el valor de la altura o height del objeto gráfico.
Nombre	double getWidth()
Descripción	Retorna el valor del ancho o width del objeto gráfico.
Nombre	void seWidth(double pwidth)
Descripción	Establece la manera de modificar el valor del ancho o width del objeto gráfico.
Nombre	void getX()
Descripción	Retorna el valor de la posición X, _ del objeto gráfico en la escena de visualización.
Nombre	void setX(double px)
Descripción	Establece la manera de modificar el valor de la posición X, _ del objeto gráfico en la escena de visualización.
Nombre	void getY()
Descripción	Retorna el valor de la posición _, Y del objeto gráfico en la escena de visualización.
Nombre	void setY(double py)
Descripción	Establece la manera de modificar el valor de la posición _,Y del objeto gráfico en la escena de visualización.
Nombre	void read(QXmlStreamReader &in)
Descripción	Establece la manera de cargado del objeto gráfico en un formato XML.
Nombre	void write(QXmlStreamReader &out)
Descripción	Establece la manera de salvado del objeto gráfico en un formato XML.

3.3.2.2 Descripción de la clase UnifilarGraphics

Nombre: UnifilarGraphics	
Descripción: Esta clase será el resultado final del marco de edición la misma está basada en el patrón Composite anteriormente mencionada. Esta clase estará compuesta por un grupo de atributos entre los que se encuentra en QGraphicsItemGroup que definirá un grupo de objetos gráficos.	
Tipo de clase:	
Atributo	Tipo
composite	QGraphicsItemGroup
Para cada responsabilidad:	
Nombre	void addToGroup(GraphicObject *object)
Descripción	Establece la manera de adicionar un nuevo elemento al grupo o más en específico al composite que este grupo de elementos gráficos.
Nombre	void removeToGroup(GraphicObject *object)
Descripción	Establece la manera de eliminar un objeto gráfico del grupo.

3.3.3 Diagrama de clase Escena

En este diagrama se reflejan los elementos del sistema que conforman el manejador de los elementos gráficos encapsulados en la clase GraphicsScene. Otras de las clases de este sistema es AbstractScene que como su nombre lo dice es una abstracción de la clase QGraphicsScene de la librería Qt.

En la siguiente figura se muestra el diagrama Escena.

Fig.10 Diagrama Escena.

3.3.3.1 Descripción de la clase GraphicsScene

Nombre: GraphicsScene	
Descripción: Esta clase será el marco de manejo de los objetos gráficos y donde los mismos estarán contenidos.	
Tipo de clase:	
Atributo	Tipo
myGridVisible	bool
myGridScale	int
myGrid	qreal
mode	SceneMode
myMenu	QMenu *
maxZ	qreal
xGrid	int
yGrid	int
m_size	QSize
m_rect	QRectF
Para cada responsabilidad:	
Nombre	void resize(const QSize & size)
Descripción	Establece la manera de redimencionar la escena gráfica en este

	caso GraphicsScene.
Nombre	void drawBackground(QPainter *p, const QRectF &r)
Descripción	Dibuja las guías de edición de la escena.
Nombre	void drawForeground(QPainter *p, const QRectF &rect)
Descripción	Dibuja el contorno(BoundingRect) de la escena y el hotspot de edición.

3.4.1 Clase Vista

En este diagrama se refleja el modelo vista del sistema que estará encargado de visualizar todos los elementos dentro de las escenas que tendrá el sistema así como los componentes gráficos que se manejaran en estas últimas, todo esto se encapsula en la clase GraphicsView.

En el siguiente diagrama se representa el diagrama vista.

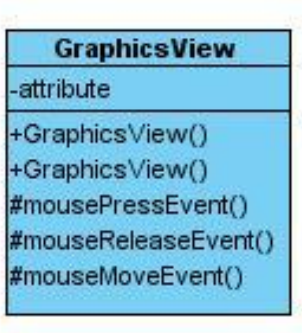

Fig. 11 Diagrama Vista.

3.4.1.1 Descripción de la clase GraphicsView

Nombre: GraphicsView	
Descripción: Esta clase estará encargada de visualizar la escena gráfica o GraphicsScene. Además manejara los eventos de adición y eliminación de los objetos gráficos que se manejaran en la escena.	
Tipo de clase:	
Atributo	Tipo
Para cada responsabilidad:	
Nombre	void mousePressEvent(QMouseEvent *event)

Descripción	Establece en manejo de la adición de los objetos gráficos a la escena así como el manejo de los modos en que se encuentra la escena.

3.5.1 Diagrama de clases Comando

El diagrama Comando representa la jerarquía de clases que conforman el paquete de clases que tienen que ver con la concepción los comandos que se manejaran dentro del sistema. Este diagrama está encabezado por la clase BaseCommand la misma es un abstracción de la clase QUndoCommand de la librería Qt. Otra de las clases que se representan en este diagrama es la clase CreateUnifilarCommand esta se comporta como el comando que está encargado de crear los gráficos unifilares dentro de nuestro sistema. También se representa en el esquema la clase RemoveUnifilarCommand que se comporta como el comando encargado de eliminar los gráficos unifilares dentro del ambiente de edición de nuestro sistema.

La siguiente figura representa el diagrama o esquema Comando.

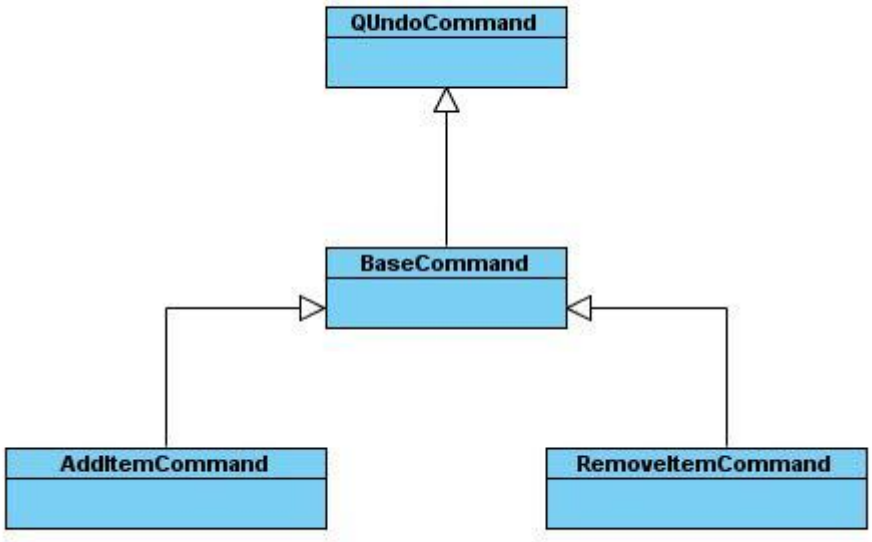

Fig. 12 Diagrama Comando.

3.5.1.1 Descripción de la clase BaseCommand

Nombre: BaseCommand	
Descripción: Esta clase abstracta de la cual heredaran todos los comandos que se diseñen en el sistema.	
Tipo de clase: Abstracta	
Atributo	Tipo
Para cada responsabilidad:	
Nombre	virtual void init()
Descripción	Establece el procedimiento de iniciación del comando.

3.5.1.2 Descripción de la clase AddItemCommand

Nombre: AddItemCommand	
Descripción: Esta clase está encargada de controlar y proveer las operaciones para el proceso de adición de los objetos gráficos en el marco de edición.	
Tipo de clase:	
Atributo	Tipo
scene	GraphicsScene *
item	GraphicObject *
Para cada responsabilidad:	
Nombre	void init()
Descripción	Establece el procedimiento de iniciación del comando AddItemCommand.
Nombre	void redo()
Descripción	Establece la operación de “hacer” del comando AddItemCommand; en este caso la de adicionar el objeto gráfico a la escena (GraphicsScene).
Nombre	void undo()
Descripción	Establece la operación de “deshacer” del comando AddItemCommand; en este caso la de eliminar el objeto gráfico de la escena (GraphicsScene).

3.5.1.3 Descripción de la clase RemoveItemCommand

Nombre: RemoveItemCommand

Descripción: Esta clase está encargada de controlar y proveer las operaciones para el proceso de eliminar los objetos gráficos de la escena (GraphicsScene).		
Tipo de clase:		
Atributo		Tipo
scene		GraphicsScene *
item		GraphicObject *
Para cada responsabilidad:		
Nombre	void init()	
Descripción	Establece el procedimiento de iniciación del comando RemoveItemCommand.	
Nombre	void redo()	
Descripción	Establece la operación de “deshacer” del comando RemoveItemCommand; en este caso la de eliminar el objeto gráfico de la escena (GraphicsScene).	
Nombre	void undo()	
Descripción	Establece la operación de “hacer” del comando RemoveItemCommand; en este caso la de adicionar el objeto gráfico a la escena (GraphicsScene).	

3.6 Conclusiones del Capítulo.

En este capítulo se hizo referencia sobre los principales patrones de diseño utilizados para la construcción de la solución final de nuestro trabajo, algunos como el patrón Composite y el patrón Comando. También se describieron los principales diagramas de clases del sistema así como una descripción de las tareas que cumplen algunas de las principales clases del sistema, algunas como UnifilarGraphics, GraphicsScene, EditorView y BaseCommand.

CONCLUSIONES

- La documentación estudiada durante el proceso de elaboración de este trabajo nos amplió nuestros conocimientos acerca de la rama industrial eléctrica.

- Se desarrollo una herramienta de gestión de gráficos unifilares logrando con esto crear, eliminar componentes gráficos eléctricos y además crear de estos componentes a partir de otros ya existente dentro del sistema.

- La incorporación del mecanismo de gestión de gráficos unifilares al editor Phoenix logró que dicho editor pueda ser utilizado en la concepción de un SCADA para la rama industrial eléctrica.

RECOMENDACIONES

- Adicionar la funcionalidad de salvar los componentes gráficos unifilares en otros formatos como binario, texto plano, lenguajes de base de datos entre otros.

- Adicionar un inspector de propiedas al editor de gráficos unifilares.

- Adicionar la funcionalidad de asociarle a los componentes gráficos variables reales de campo tanto en tiempo de ejecución como en tiempo de configuración.

REFERENCIAS BIBLIOGRAFICAS.

1. Componentes de un sistema SCADA [Online] Sistemas de SCADA para subestaciones eléctricas http://www.sertiic.cl/documentos/doc.pdf

2. Normativa UNE- EN 60617(ICE 60617) [Onlie]
http://amsfrancisco.planetaclix.pt/download/Outros/IEC_60617.pdf

3. Metodología de Desarrollo de Software. [Online]
http://www.scribd.com/doc/12983329/Metodologia-de-Desarrollo-de-Software .

4.Normativa EN 60617(ICE 60617) [Online]http://std.iec.ch/iec60617

5. **Gutierrez Gonzalez, Rocio y Sanchez Gonzalez , Raul.** *Comparativa entre las biblotecas gráficas GTK + y QT.*

6. OpenUP como alternativa metodológica para proyectos pequeños de software. [Online]
http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodolgica.html .

7. El Proceso Unificado de Desarrollo de Software (RUP). [Online]
http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm .

8. **CASE, Herramientas.** [Online]
http://www.google.com.cu/url?sa=t&source=web&ct=res&cd=1&ved=0CBcQFjAA&url=http%3A%2F%2Fwww.itescam.edu.mx%2Fprincipal%2Fsylabus%2Ffpdb%2Frecursos%2Fr19670.DOC&rct=j&q=Herramientas+ingenier%C3%ADa+asistida+por+computadora+%28case%29&ei=q9v9S7GgBcH88AaQ7aS.

9. Umbrello UML Modeller - Presentation Transcript. [Online]
http://www.slideshare.net/ovruni/umbrello-uml-modeller-presentation.

10. Paradigma visual para UML (Plataforma Java) (Visual Paradigm for UML [Java Platform]) 6.0. [Online]
http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

11. GNU Operating System. [Online] http://www.gnu.org/home.es.html .

12. Eclipse,herramienta universal -IDE abierto y extensible. [Online] julio 29, 2009.
http://zintegra.blogspot.com/2009/07/eclipse-herramienta-universal-ide.html.

13. **Stroustrup, Bjarne.** *The C++ Programming Language.* Addison-Wesley : Third Edition, 1997.

14. Análisis de la clase (RUP parte 2 y OPEN/UP). [Online]
http://eproano334.blogspot.es/tags/Vida/ .

15. Lenguaje de programación. [Online] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml

16. Lenguaje de programación python. [Online] http://www.ecualug.org/files/Flisol%20-%20Python.pdf

17. Diagrama de casos de uso. [Online] http://www.developer.com/design/article.php/2109801

18. Biblioteca (Informática) [Online]. http://www.utas.edu.au/infosys/info/documentation/C/CStdLib.html

BIBLIOGRAFIA

1. Design_Patterns_in_C++_with_Qt4.chm [Asistente].

2. Design_Patterns.chm [Asistente]

3. Introducción a las bibliotecas gráficas GTK. [Online]
http://gsyc.es/~lcanas/memo_camx/x70.html

4. SISTEMAS SCADA. [Online] http://www.automatas.org/redes/scadas.htm .

5. DISEÑO DE APLICACIONES SCADA CON LABVIEW. [Online] Universidad Autónoma de Barcelona. http://www.automatas.org/redes/scadas.htm .

6. About the Eclipse Foundation. [Online] http://www.eclipse.org/org/.

7. Metodología de Desarrollo de Software. [Online]
http://www.scribd.com/doc/12983329/Metodologia-de-Desarrollo-de-Software .

8. **Gutierrez Gonzalez, Rocio y Sanchez Gonzalez , Raul.** *Comparativa entre las biblotecas gráficas GTK + y QT.*

7. OpenUP como alternativa metodológica para proyectos pequeños de software. [Online]
http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodolgica.html .

8. El Proceso Unificado de Desarrollo de Software (RUP). [Online]
http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm .

9. **CASE, Herramientas.** [Online]
http://www.google.com.cu/url?sa=t&source=web&ct=res&cd=1&ved=0CBcQFjAA&url=http%3A%2F%2Fwww.itescam.edu.mx%2Fprincipal%2Fsylabus%2Ffpdb%2Frecursos%2Fr19670.DOC&rct=j&q=Herramientas+ingenier%C3%ADa+asistida+por+computadora+%28case%29&ei=q9v9S7GgBcH88AaQ7aS.

10. Umbrello UML Modeller - Presentation Transcript. [Online]
http://www.slideshare.net/ovruni/umbrello-uml-modeller-presentation.

11. Paradigma visual para UML (Plataforma Java) (Visual Paradigm for UML [Java Platform]) 6.0. [Online]
http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

13. GNU Operating System. [Online] http://www.gnu.org/home.es.html .

14. Eclipse,herramienta universal -IDE abierto y extensible. [Online] julio 29, 2009.
http://zintegra.blogspot.com/2009/07/eclipse-herramienta-universal-ide.html.

15. **Stroustrup, Bjarne.** *The C++ Programming Language.* Addison-Wesley : Third Edition, 1997.

16. Análisis de la clase (RUP parte 2 y OPEN/UP). [Online] http://eproano334.blogspot.es/tags/Vida/ .

17. Diagrama de casos de uso. [Online] http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php

18. Normativas de representación circuital.[Online] http://www.ual.es/Universidad/Depar/IngenRural/documentos/electrotecnia2.pdf

19. Normativas de representacion circuital.[Online] http://www.etitudela.com/profesores/rbv/04f7af9bf50f1820a/04f7af9bf50f22910/index.html

20. Patrón arquitectónico Modelo-Vista-Controlador. http://www.proactiva-calidad.com/java/patrones/mvc.html