Universidad de las Ciencias Informáticas

Facultad 5

Procesador de estado de red para adecuaciones del SCADA-GALBA al sector eléctrico.

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

Autores: Carlos Manuel Castillo Chacón.
Jose Manuel Acevedo Medina.

Tutores: Ing. Marisniulkis Lescaille Cos.
Yordan Gallardo Avilés.

Co-tutores: Ing. Frank Rufino Nápoles.
Luis Andrés Valido Fajardo.

La Habana
Junio 2015

Declaración de autoría

Declaración de autoría

Declaramos ser los únicos autores del presente trabajo “Procesador de estado de red para adecuaciones del SCADA-GALBA al sector eléctrico y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Firma del Autor Firma del Autor

Carlos Manuel Castillo Chacón Jose Manuel Acevedo Medina

_____ _____

Firma del Tutor Firma del tutor Ing

Marisniulkis Lescaille Cos. *Ing. Yordan Gallardo Avilés.*

Pensamiento

"El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres."

Ernesto Che Guevara.

Agradecimientos

Agradezco primeramente a mis padres, a mi papá Juan Carlos y mi mamá Caridad, ellos han sido los principales responsables de que yo hoy este aquí discutiendo este trabajo de diploma, me apoyaron en todo lo que me hizo falta, hicieron lo posible y hasta lo imposible para que terminara la carrera y hoy me pudiera graduar.

Agradezco a mi abuela Edicia, mami como la conocemos todos, por haberme brindado todo su apoyo, siempre tratando de darme todo lo que pudiera con el fin de que me sienta bien.

Agradezco a mi tía Petronila que me brindó su apoyo a mí y a mis padres siempre que hizo falta.

En general agradezco a toda mi familia que siempre estuvo presente cada vez que los necesité.

Agradezco a mi novia Melissa que apareció en mi vida solo para hacer cosas buenas, en este último año me apoyó en todo lo que me hizo falta dándome muchas alegrías.

Agradezco a mis compañeros de cuarto Yaikel, Rigo que se han convertido en mis hermanos, siempre estuvieron presentes en cada situación o fiesta que hizo falta.

Agradezco a mis amigos o también mis hermanos, Alfonso, Raúl, Costa, Yasser, Mederos, Yoan, Luis Javier y todos los demás que son muchos por todo lo que vivimos en la universidad que de una manera u otra fueron parte importante de que hoy este aquí.

Agradezco a mis tutores Marisniulkis y Yordan por apoyarme en todo, gracias a sus actitudes siempre salimos bien, hasta llegar hoy aquí.

Agradezco a Rufino y Valído por siempre estar disponibles cada vez que me hizo falta aclarar una duda.

Agradezco a mi socio, hermano y compañero de Tesis Jose o más conocido como KOKO por todo lo que tuvimos que hacer para poder llegar hoy aquí, las malas noches, las fiestas, y todo con un solo objetivo, graduarnos.

Agradezco a todos los que de una forma u otra pusieron un granito de arena en este largo camino que hoy llega a su fin.

Agradecimientos

Agradezco a mis padres Jose Acevedo Fernández y Aristalia Medina Victoria los que siempre me apoyaron y creyeron en mí, en las buenas y en las malas los que siempre me aconsejaron para bien, a los que quiero con todo mi corazón.

Agradezco a mi familia por el apoyo brindado en todo momento por creer en mí, por darme su bendición cuando lo necesitaba en especial dos viejitas que siempre estuvieron conmigo Olga Victoria y Rosa Luz Fernández.

Agradezco a todos los que me ayudaron en el trascurso de la carrera a los hermanos que he hecho aquí a los inolvidables del 87.

Agradezco a mis compañeros de cuarto tanto del antiguo 87 como del actual 88 como Costa, Lara, Yacer, Alfonso, Yaikel, Rioger, Kokito, Rigo, Ana y todos los que de una forma u otra me ayudaron a sobrevivir en esta escuela.

Agradezco a mi compañero de tesis Carlos Manuel el cual compartió conmigo tantos momentos de entrega en cinco años tanto en buenos como en malos momentos, convirtiéndose en un hermano para mí y el que he llegado a respetar, estimar y admirar.

Agradezco en general a todo el que puso su poquito de ayuda no solo en el desarrollo de este trabajo si no en el desarrollo de la persona que hoy soy.

Agradezco a los tutores Marisnuilkis y Yordan los que siempre dieron su mejor actitud y estuvieron junto a nosotros en todo momento.

Dedicatoria

Dedico este trabajo de diploma a las dos personas más importantes de mi vida: Mis padres, los cuales han dedicado toda su vida al bienestar de sus hijos, siempre tratando de educarnos de la mejor manera para poder ser buenas personas en el futuro, gracias a ellos yo estoy aquí, sin ellos no hubiese sido posible.

Carlos Manuel Castillo Chacón.

Quisiera dedicar este triunfo a los seres que más quiero en este mundo los que me han dado la oportunidad de crecer y crecer para bien los que siempre confiaron hasta el final, a mis padres del alma.

Jose Manuel Acevedo Medina.

Resumen

El presente documento aborda la investigación para desarrollar una herramienta que permita determinar el estado de una red en las subestaciones para el proyecto Adecuaciones del SCADA Galba para el sector eléctrico del Centro CEDIN perteneciente a la Universidad de las Ciencias Informáticas. Para lograr este propósito se analizaron e implementaron los paquetes necesarios del estándar IEC 61970, parte del Modelo de Información Común, CIM. Así como las funciones matemáticas necesarias para realizar el procesamiento de estado de una red de energía. Además se guió el desarrollo de la herramienta por la metodología AUP, el marco de trabajo de programación Qt en su versión 4.8 y el lenguaje de programación C++.

Índice

Índice de figuras

Introducción

El desarrollo de las tecnologías de la información y las comunicaciones ha revolucionado la sociedad en general y específicamente el sector empresarial. La industria ha tomado parte en este desafío incorporando sistemas que dan solución a diferentes problemáticas y aportan a la toma de decisiones de los operadores en sus diferentes ramas. La necesidad de automatizar los procesos de la industria generó la implementación de sistemas para supervisar procesos industriales. Con la aparición de éstos se ha potenciado enormemente el desarrollo de la industria, aumentando los niveles de eficiencia, disminución de costos y optimización de los procesos.

Los más comunes se denominan Sistemas de Supervisión, Control y Adquisición de datos SCADA (*por sus siglas en inglés*) son software para ordenadores que permiten controlar y supervisar procesos industriales a distancia. Facilitan la retroalimentación en tiempo real con los dispositivos de campo, sensores y actuadores, siendo el primero un dispositivo diseñado para recibir información de una magnitud del exterior y transformarla en otra magnitud, normalmente eléctrica, los actuadores son dispositivos capaces de transformar energía en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado, ambos proporcionan el control de los procesos automáticamente. Siendo los SCADA los encargados de proveer de toda la información que se genera en el proceso productivo (supervisión, control de calidad, control de producción, almacenamiento de datos, etc.) y permite su gestión e intervención. Estos sistemas constan de funcionalidades básicas como, supervisión remota de instalaciones y equipos, control remoto de instalaciones y equipos, procesamiento de datos, visualización gráfica dinámica, almacenamiento de información histórica y programación de eventos (2).

Las soluciones de automatización de procesos son altamente aceptadas para los sistemas de energía. Están basados mayormente en un número grande de especificaciones propietarias o estándares de facto. Los sistemas de energía actuales y futuros enfrentan una creciente demanda de configuración de información que describen los datos del proceso, los dispositivos de automatización y posiblemente el equipamiento primario. Por lo que es cada vez más necesario que las aplicaciones tengan que manejar dentro de sí mismas o interactuar con otros sistemas a través de esta información de configuración.

En el Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas se desarrollan, despliegan y mantienen sistemas de este tipo teniendo contratos con la empresa mixta Guardián del ALBA de la República Bolivariana de Venezuela, comercializando sus productos a través de la empresa

ALBET S.A, la cual es una empresa cubana, cuyo origen y desarrollo se vincula estrechamente a la Universidad de Ciencias Informáticas (UCI), modelo de universidad productiva que agrupa más de doce mil profesionales, técnicos y estudiantes. Albet posee los derechos comerciales de todos los productos y servicios que desarrolla la UCI y mediante la alianza con otras prestigiosas entidades ofrece soluciones integrales en la esfera de las tecnologías de la información y las comunicaciones. La empresa Guardián del ALBA responde a los intereses de informatización a la empresa Petróleos de Venezuela, SA (PDVSA). Siendo el ejemplo más fehaciente el SCADA “Guardián del ALBA”, este sistema permite la supervisión y control de los procesos industriales en el sector del petróleo.

Debido a la creciente demanda de energía eléctrica tanto en el sector doméstico como industrial y a su vez el crecimiento de la empresa de Petróleos de Venezuela, SA (PDVSA), le fue dada la tarea de asumir la supervisión y control de la energía en sus subestaciones eléctricas. La empresa PDVSA tiene sistemas implantados en las subestaciones como el RANGER, es un SCADA propietario, desarrollado por ABB Network Management, para la supervisión de sistemas de distribución, transmisión y generación de energía eléctrica, y el Sinaut Spectrum. Estos incorporan componentes que realizan funciones de análisis de red y balance de carga eléctrica pero con un alto costo monetario, ya que están representados por altos intereses comerciales. Por lo tanto, es necesario y de prioridad para PDVSA sustituir estos sistemas por herramientas de software libre que satisfagan las necesidades inmediatas para la supervisión en subestaciones eléctricas. Debido a esto surge entonces la necesidad de implementar un Sistema de Manejo de Energía EMS (por sus siglas en inglés) que permita obtener información fiable sobre los equipos y valores que estos generan en las subestaciones eléctricas.

Un EMS es un sistema de herramientas asistidas por ordenador e integrada sobre un SCADA, utilizados por los operadores de redes de servicios eléctricos para supervisar, controlar y optimizar el rendimiento del sistema de generación /o transmisión, además de reducir costos considerablemente, siendo algunas de estas funcionalidades análisis de la red, tolerancia a fallos, procesador de topología de la red y procesador de estado de red, entre otros. Estos sistemas tienen sus raíces en la década de 1920 cuando ABB suministró su primer sistema de control remoto para una planta de energía, pero no fue hasta la década del 1960 con la aparición de los sistemas computarizados que surgieron los primeros EMS como se les conoce en la actualidad. En sus inicios estos sistemas eran creados de una manera diferente por distintas empresas por lo que el Instituto de Investigación de Energía Eléctrica (EPRI por sus siglas en inglés) se dio la tarea de crear el Modelo de Información Común (CIM, por sus siglas en inglés) que más tarde fue adoptado por la Comisión Electrónica Internacional (IEC, por sus siglas en inglés) como estándar internacional para sistemas eléctricos, que corresponde a la norma IEC 61897, este está centrado en la automatización de equipos de subestaciones eléctricas de diversos fabricantes; fue diseñado como el único

protocolo que ofrece una completa solución de comunicación y el principal beneficio que ofrece es la interoperabilidad entre equipos, y la norma IEC 61970, se refiere a las interfaces de programas de aplicación de sistemas de gestión energética (EMS) (6).

El sistema de supervisión y control SCADA GALBA aunque se ha desplegado con éxito para la supervisión de procesos petroleros, en el caso de la supervisión de subestaciones eléctricas y de energía en general, carece de funcionalidades que son básicas en estos procesos y limitan el buen desempeño y gestión de los mismos, la información y los datos. No contiene un modelo eléctrico para representar los equipos de energía, por lo que no permite al operador visualizar el estado de transmisión de energía entre los dispositivos supervisados y de la red en general. Carece de la representación de un modelo en tiempo real de la red de energía, que se traduce en un componente para procesar el estado de la red de energía, funcionalidad clave en un EMS, que comprende la creación de la topología de isla de la red y la definición de la conectividad de la red. Debido a la ausencia de lo anteriormente mencionado el operador no tiene información confiable en caso de fallos del sistema.

Por lo planteado anteriormente surge como **problema de investigación**, ¿Cómo proporcionar al SCADA GALBA un modelo en tiempo real de la red de energía así como un modelo eléctrico para representar equipos de energía? Se plantea como **objeto de estudio** el proceso de representación y modelado de objeto gráfico y coloreado de la red en sistemas EMS. El **objetivo general** que se persigue en la investigación es desarrollar una herramienta que permita procesar el estado de la red de energía basado en el Modelo de Información Común que se integre al SCADA GALBA enmarcando el **campo de acción** en el procesador de estado de la red y modelo de representación de objetos gráficos en el sistema SCADA GALBA.

Para lograr el cumplimiento de los objetivos se plantean las siguientes **tareas de investigación**:

- Revisión bibliográfica para generar el marco teórico conceptual de la investigación que represente las tendencias actuales en los sistemas manejadores de energía.
  - ✓ Análisis de los sistemas manejadores de energía para generar un software que incorpore las mejores prácticas de estos sistemas.
  - ✓ Análisis de implementaciones del modelo CIM.
  - ✓ Análisis de las tecnologías y herramientas existentes para la identificación de las que se usarán en el desarrollo del sistema.
- Selección de la metodología de desarrollo y las tecnologías a utilizar.
- Definición de la arquitectura del sistema a desarrollar y modelar la propuesta.
- Implementación de una herramienta de modelado.
- Validación de la herramienta implementada.

- Desarrollo de una documentación que recoja todo el proceso de investigación y desarrollo del sistema.

Se tiene como **posibles resultados** los siguientes:

- Artefactos de Diseño e Implementación de las funcionalidades a incorporar en la herramienta a desarrollar.

- Herramienta para procesar el estado de la red basada en el modelo CIM e integrada en el SCADA GALBA.

- Documentación como guía para futuros desarrollos de otras herramientas que son afines a sistemas manejadores de energía.

Para el desarrollo teórico científico del tema se proponen los siguientes **Métodos científicos.**

Métodos teóricos:

- Analítico Sintético: Admite el análisis de todo el contenido de características específicas del EMS, de manera que permita sintetizar todo lo obtenido relacionado con los estándares pertinentes y trabajos ya realizados anteriormente por expertos en el tema.

- Análisis histórico-lógico: Es utilizado para analizar la evolución histórica de soluciones similares, las tendencias más recientes de los EMS y basándose en esos datos, complementar las características necesarias y deseables para la solución que se propone.

Métodos empíricos:

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

- Generalización: Permite sistematizar en cada capítulo de la investigación las pautas más significativas de la misma, así como llegar a conclusiones más objetivas y explícitas para cada caso.

Capítulo 1
Introducción

En este capítulo se abordará el estudio del estado del arte con el objetivo de que se obtenga un punto de partida en la investigación que posteriormente se llevará a cabo en el presente trabajo, dando una panorámica del desarrollo de esta tecnología tanto en Cuba como en el mundo.

1.2 Sistemas de supervisión, control y adquisición de datos (SCADA)

1.2.1 Introducción a los sistemas SCADA

En la actualidad los sistemas SCADA desempeñan un papel fundamental en la automatización industrial, estos comprenden las soluciones de aplicación que necesitan de la captura de información de un proceso o planta industrial, la cual es empleada para realizar análisis con los que se pueden obtener importantes indicadores que permiten una realimentación del proceso. De forma general se consideran como funciones básicas de un sistema SCADA la supervisión y control remoto de instalaciones, procesamiento de información, generación de reportes, gestión de alarmas, almacenamiento de información histórica, presentación de gráficos de tendencias y programación de eventos entre otros (1).

Los primeros sistemas SCADA se caracterizaban por ser monolíticos es decir ejecutaban todas las operaciones en una sola computadora central. Se ejercía poco control y la mayoría de las funciones de estos SCADA se limitaban a los sensores de control y en marcar las operaciones que superaban los niveles de alarma programados. Estos sistemas eran todos programas de propiedad del proveedor y por lo general se limitaban a una sola planta o instalación. Al igual que los programas, los equipos SCADA de un proveedor rara vez se podían utilizar en el sistema SCADA de otro proveedor. Actualmente estos tienden a ser distribuidos que no es más que un conjunto de elementos, tanto de procesamiento como de almacenamiento que están conectados a través de una red y que para un usuario del sistema se comporta como un único computador. Existen cinco características que son responsables de la utilidad de los sistemas distribuidos: compartir recursos, apertura, concurrencia, tolerancia a fallas y transparencia. Se hace notar que estas características no son consecuencia automática de la distribución, el software debe ser cuidadosamente diseñado para asegurar que ellas sean conseguidas (1).

Descripción general de un SCADA

Los SCADAs son aplicaciones de software, diseñadas con la finalidad de controlar y supervisar procesos a distancia. Se basan en la adquisición de datos de los

procesos remotos. Es una aplicación de software diseñada para funcionar sobre ordenadores en el control de la producción, proporcionando comunicación con los dispositivos de campo (controladores autónomos, autómatas programables, entre otros.) y controlando el proceso de forma automática desde la pantalla del ordenador. Además, envía la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa, es decir, que permite la participación de otras áreas como por ejemplo: control de calidad, supervisión, mantenimiento, entre otros (2).

En los SCADA, el operador puede visualizar en la pantalla del computador de cada una de las estaciones remotas que conforman el sistema, los estados de esta, las situaciones de alarma y tomar acciones físicas sobre algún equipo lejano, la comunicación se realiza mediante buses especiales o redes LAN. Todo esto se ejecuta normalmente en tiempo real, y están diseñados para dar al operador de planta la posibilidad de supervisar y controlar dichos procesos. Estos sistemas actúan sobre los dispositivos instalados en la planta, como son los controladores, autómatas, sensores, actuadores, registradores, entre otros. Además permiten controlar el proceso desde una estación remota, para ello el software brinda una interfaz gráfica que muestra el comportamiento del proceso en tiempo real (2).

El sistema SCADA en general está distribuido en módulos que trabajan de manera conjunta posibilitando el funcionamiento del sistema como un todo. Estos módulos se encuentran interconectados a través de un software para la distribución de los servicios en la red, conocido como "*middleware*" o software de comunicación entre aplicaciones (3). La distribución de los mismos en el SCADA permite obtener configuraciones escalables en dependencia de los requisitos que presente cada aplicación. Es un sistema en tiempo real y presenta una arquitectura distribuida. Está compuesto por varios módulos entre los que se encuentran (1):

- **Middleware**: Es la capa de software, que se encarga de la comunicación entre los diferentes módulos que forman parte del sistema. Este módulo tiene como finalidad proporcionar la capa de comunicación de alto nivel, tanto sincrónica, como asincrónica, para la comunicación de todos los módulos que conforman el sistema SCADA (1).
- **Adquisición:** Es el encargado de la adquisición, recepción, procesamiento y distribución de los datos provenientes del campo (1).
- **Configuración:** El servicio de configuración está formado por un grupo de componentes cada uno con una tarea específica y una base de datos que contendrá las configuraciones de cada uno de los módulos que conformen el proyecto activo. Es el encargado de almacenar, persistir y suministrar la información base para el funcionamiento de los demás módulos del SCADA (1).

- **Almacenamiento de datos históricos:** Es el encargado de almacenar la información del sistema para que posteriormente pueda ser empleada, por ejemplo, en generación de reportes, tendencias o en gestión de producción. La base de datos histórica (BDH) contendrá la información persistente de los datos recolectados de los dispositivos (1).
- **Seguridad:** Proporciona las funcionalidades necesarias para garantizar el trabajo autorizado a usuarios y módulos, además brinda las herramientas necesarias para la protección contra ataques maliciosos o involuntarios al sistema por parte de personas o recursos, tales como fallas de energía, problemas de red o servidores (1).
- **Visualización o HMI:** Se encarga de representar en un ordenador, los procesos que ocurren en el campo en tiempo real, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador diferentes niveles de control en dependencia de sus niveles de privilegios. Este módulo permite al operador el contacto directo con el sistema y realizar la supervisión y el control del proceso en general (1).

1.2.2 Funcionalidades y aplicaciones

Funcionalidades

- **Supervisión remota de instalaciones y equipos:** Permite al operador conocer el estado de desempeño de las instalaciones y los equipos alojados en la planta, lo que permite dirigir las tareas de mantenimiento y estadística de fallas.
- **Control remoto de instalaciones y equipos:** Mediante el sistema se puede activar o desactivar los equipos remotamente (por ejemplo abrir válvulas, activar interruptores, prender motores, entre otros.), de manera automática y también manual. Además es posible ajustar parámetros, valores de referencia, algoritmos de control, entre otros.

Procesamiento de datos: El conjunto de datos adquiridos conforman la información que alimenta el sistema, esta información es procesada, analizada y comparada con datos anteriores y con datos de otros puntos de referencia, dando como resultado una información confiable y veraz.

- **Visualización gráfica dinámica:** El sistema es capaz de brindar imágenes en movimiento que representen el comportamiento del proceso, dándole al operador la impresión de estar presente dentro de una planta real. Estos gráficos también pueden corresponder a curvas de las señales analizadas en el tiempo.
- **Generación de reportes:** El sistema permite generar informes con datos estadísticos del proceso en un tiempo determinado por el operador.
- **Representación de señales de alarma:** A través de las señales de alarma se logra alertar al operador frente a una falla o la presencia de una

condición perjudicial o fuera de lo aceptable. Estas señales pueden ser tanto visuales como sonoras.

- **Almacenamiento de información histórica:** Se cuenta con la opción de almacenar los datos adquiridos, esta información puede analizarse posteriormente, el tiempo de almacenamiento dependerá del operador o del autor del programa.
- **Programación de eventos:** Está referido a la posibilidad de programar subprogramas que brinden automáticamente reportes, estadísticas, gráfica de curvas, activación de tareas automáticas, entre otros (3).

Aplicaciones

- Supervisión, control, calidad, mantenimiento, control de producción, almacenamiento de datos durante los procesos de fabricación de sólidos y líquidos con una cierta complejidad.
- Términos de rentabilidad, calidad y seguridad en su sentido amplio, los nuevos requisitos de control ambiental y ahorro energético, o la necesidad de gestión flexible de fórmulas y recetas.
- SCADA para IPhone, Ipad, IPod, entre otros, dispositivos que son capaces de controlar y modificar los datos de forma inalámbrica, con la función de código de barras, son capaces de escanear la información de cualquier tipo de productos y permite varios enlaces con diferentes controladores lógicos programables, más conocido por sus siglas en inglés *PLC*(*Programmable Logic Controller*) y para cada enlace definir los puntos de diferentes tipos de datos como *Boolean*, *byte*, un entero de palabras, y de punto flotante, y leer los valores reales (7).

1.2.3 Caracterización de SCADA actuales

SCADA	Fabricantes	Tipo de licencia
Movicom	**Progea**	**Privativo**
RANGER	**ABB network management**	**Privativo**
SINAUT Spectrum	**SIEMENS**	**Privativo**
Intouch	**Wonderware**	**Privativo**

1.2.4 Especificaciones del SCADA GALBA

SCADA-GALBA

El SCADA-GALBA fue producido en el Centro de Desarrollo Industrial (CEDIN), de la Universidad de las Ciencias Informáticas (UCI).Es un software que integra las funcionalidades de alto nivel que permiten la solución de aplicaciones de supervisión y control de procesos, utilizando para ello una arquitectura distribuida de módulos que permite escalar a aplicaciones de gran envergadura. El SCADA "Guardián del ALBA" es el resultado de la colaboración de universidades, empresas venezolanas y cubanas coordinadas por el distrito tecnológico de AIT PDVSA con sede en Mérida y La Universidad de las Ciencias Informáticas integrando las soluciones a ambos lados del Caribe.

Características más significativas del SCADA-GALBA:

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

1.2.5 Sistemas similares

SCADA SINAUT Spectrum

Sistemas SINAUT Spectrum SCADA se construyen en la misma arquitectura de sistema abierto potente y flexible como SINAUT S**i**stema Spectrum Gestión de la Energía (EMS) y el Sistema de Gestión de la Distribución (DMS), que ofrece oportunidades de crecimiento para el más allá de la capacidad de los sistemas basados en PC. Puede funcionar como un sistema centralizado o como sistemas regionales o distritales remotas. Las configuraciones SINAUT Spectrum están diseñadas para cumplir con el requisito de las empresas eléctricas que varían en tamaño, desde las cooperativas eléctricas rurales y sistemas municipales a los sistemas de propiedad de inversionistas grandes y utilidades incluso en todo el país.

La arquitectura del SINAUT Spectrum es escalable. Por lo tanto, este sistema es ideal para la configuración de los sistemas de control de la red de cualquier magnitud, con todas las combinaciones posibles de los programas de aplicación.
En SINAUT Spectrum, todas las aplicaciones de EMS son totalmente integradas y modulares en la ejecución. Poseen interfaces definidas con precisión y utilizan un modelo de datos común con las aplicaciones SCADA, que incluyen todas las funciones necesarias para la alarma, medición, cálculo, archivo de eventos, control y sistemas de monitoreo. Personalización, modificaciones y adaptación a una situación específica según el requisito del cliente y la especificación es una característica clave de SINAUT Spectrum. Independientemente de si se trata de los requisitos de tamaño o funcionalidad, un SINAUT Spectrum EMS es expandible en cualquier momento y se adapta fácilmente a los cambios. SINAUT Spectrum ofrece tecnología de vanguardia para el futuro.

SCADA RANGER:

Es un SCADA/EMS de la compañía ABB. Posee las funciones estándar que provee un sistema SCADA: procesamiento de alarmas, recolección de datos e Interfaz Hombre Máquina. Además permite tomar acciones de control. Posee herramientas fundamentales de la gestión de energía como Control de Generación, análisis de red de energía, optimización de redes y planeación de producción.

Ventajas y Desventajas de estos Sistemas SCADAs

Ventajas

- La computadora puede registrar y almacenar una gran cantidad de datos.
- Los datos pueden mostrarse de la manera requerida por el usuario.
- Se pueden conectar al sistema miles de sensores distribuidos sobre una gran área.
- El operador puede incorporar simulaciones de datos reales al sistema.
- Se pueden recolectar muchos y diversos tipos de datos desde los dispositivos distribuidos en la red.
- Los datos pueden visualizarse desde cualquier lugar y no solamente en el sitio de instalación de los dispositivos de adquisición y control distribuido.

Desventajas

- Programación compleja.
- Inexistencia de reloj global (en ocasiones).
- Fallos independientes; (aunque el sistema sea más robusto).
- Inseguridad al momento de operar.

1.3 Modelo de Información Común (CIM)

1.3.1 ¿Qué es CIM?

El modelo de información común (CIM) es un modelo abstracto que representa todos los objetos en una empresa de servicios eléctricos que suelen participar en operaciones de servicios públicos. Al proporcionar una forma estándar de representar los recursos del sistema de energía como las clases de objetos y atributos, junto con sus relaciones, el estándar facilita la integración del sistema de gestión de energía a aplicaciones desarrolladas por diferentes proveedores independientemente de quienes fueron. El CIM facilita la integración mediante un lenguaje común, basado en el estándar para habilitar estas aplicaciones o sistemas de acceso a datos públicos y el intercambio de información independientemente de cómo están estructurados internamente. Esta norma debe ser entendida como una herramienta para permitir la integración en cualquier dominio donde se necesita un modelo de sistema de alimentación común para facilitar la interoperabilidad y compatibilidad del enlace entre las aplicaciones y los sistemas independientes de cualquier aplicación particular. Debido al tamaño del modelo CIM completo, las clases de objetos contenidos en el modelo se agrupan en un número de paquetes lógicos, cada uno de los cuales representa una parte determinada del sistema de potencia global que se está modelando. Colecciones de estos paquetes son tratados como Normas Internacionales separadas. Esta particular norma internacional especifica un conjunto base de paquetes que

proporcionan una vista lógica de los aspectos funcionales del Sistema de Gestión de la Energía (EMS) la información dentro de la empresa de servicios eléctricos que se comparte entre todas las aplicaciones (6).

1.3.2 Características y estructuras asociadas al CIM

El modelo CIM se define dentro de una series de normas de la IEC, dentro de las que se encuentran el IEC 61850 Communication Networks and Systems in Substations, especializado en la comunicación del hardware y el software en las subestaciones y el IEC 61970 centrado en los sistemas de gestión de redes de transportes, también conocidos como EMS.

Características IEC 61850

- Comunicación cerca de los equipos de potencia.
  - ✓ Capacidades de comunicación, adquisición de datos, y control, deben ser incluidas directamente en los equipos primarios.
- Reducción del cableado convencional.
  - ✓ LAN en lugar de múltiples cables de cobre.
- A prueba de futuro.
  - ✓ Los servicios y las inversiones serán duraderos a pesar de los rápidos cambios tecnológicos.
  - ✓ El estándar está diseñado para seguir tanto el progreso en las tecnologías de comunicación, como los requerimientos que envuelven a estos sistemas.

Estructura del estándar IEC 61850 dividida por capas (8)

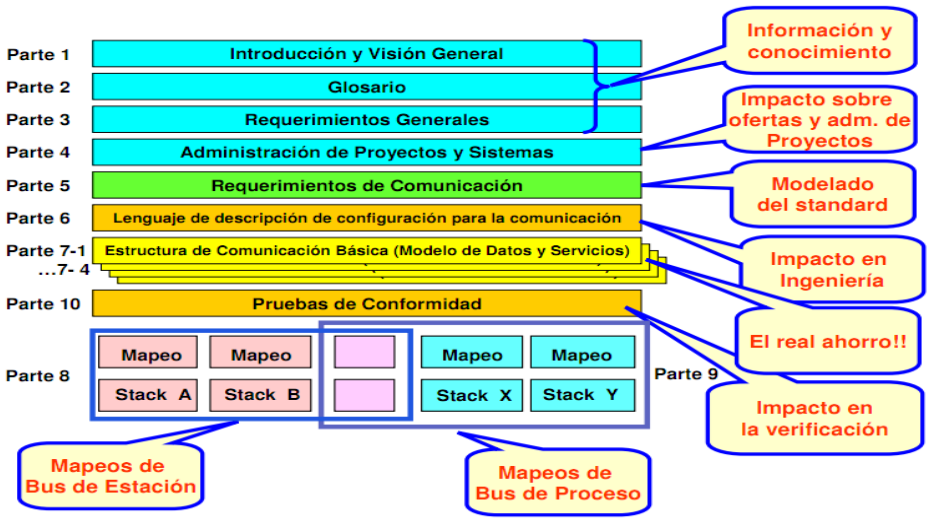


Ilustración 1 Partes del estándar IEC 61850 (8)

Parte 1 de la norma IEC 61850 define las formas de comunicación entre IEDs en la subestación eléctrica y los requisitos relacionados al sistema.

Parte 2 contiene el glosario de terminología específica y de definiciones que se utilizan en el contexto de los sistemas automáticos en subestaciones eléctricas, así como en las normas complementarias a esta.

Parte 3 especifica los requisitos generales de las redes de comunicación, con especial énfasis en los requisitos de calidad. Igualmente trata los requisitos relativos a las condiciones ambientales y de servicios auxiliares así como las recomendaciones correspondientes a requisitos específicos provenientes de otras normas y especificaciones.

Parte 4 describe los requisitos del proceso de gestión de sistema y proyecto y de las herramientas de soporte especial para ingeniería y pruebas.

Parte 5 describe los requisitos para la comunicación de funciones y modelos de los equipos eléctricos en las subestaciones.

Parte 6 especifica una descripción del lenguaje para la configuración de los IEDs en las subestaciones eléctricas.

Parte 7 se definen las estructuras básicas de comunicación para los equipos eléctricos en subestaciones y alimentadores de distribución.

Parte 8 define el mapeo específico de servicio de comunicación de los objetos y servicios de la Interfaz de Servicios de Comunicación Abstracta a especificación de mensajes de fabricación.

Parte 9 define el mapeo específico de servicio de comunicación para la comunicación entre el nivel campo (bahía) y el nivel proceso en enlaces punto a punto.

Parte 10 define los métodos de pruebas de conformidad del equipo usado en las subestaciones eléctricas.

Características IEC 61970

El estándar incluye un diagrama UML (Unified Modeling Language), una notación de modelado orientada a objeto el cual representa los paquetes y las clases que el CIM contiene.

Estos paquetes representan la estructura de una subestación eléctrica desde su diseño de redes hasta cada elemento que contiene los cuales se relacionan y

comparten información entre ellos para conformar el modelo de información común los cuales son:

- Core
- Domain
- OperationalLimits
- Topology
- Wires
- Generation
- GenerationDynamics
- Generation.Production
- LoadModel
- Outage
- Protection
- Equivalents
- Means
- SCADA
- ControlArea
- Contingency
- StateVariables

1.3.3 Descripción de los paquetes más importantes del modelo CIM

Domain: Este paquete contiene el dominio de datos utilizados en todo el CIM como datos primitivos y datos específicos del estándar el cual plantea en la documentación del mismo, para obtener compatibilidad entre los demás paquetes (6).

Core: Contiene el núcleo y entidades, compartida por las aplicaciones más comunes de las colecciones de estas entidades. No todas las aplicaciones requieren de todas las entidades del Núcleo .Este paquete no depende de ningún otro paquete, excepto el paquete de dominio, pero la mayoría de los otros paquetes tienen asociaciones y generalizaciones que dependen de ella (6).

OperationalLimits: Los modelos de paquetes OperationalLimits son una especificación de límites asociados a los equipos y otras entidades operacionales (6).

Topology: Una extensión para el paquete Core que, en asociación con los modelos de clase Terminal Conectividad, que es la definición física de cómo los equipos están conectados entre ellos. Dentro adiciona este modelo topología, que es la definición lógica de cómo el equipo está conectado a través de interruptores cerrados dándole el aspecto físico de la subestación eléctrica (6).

Wires: Una extensión para el paquete Core y Topología que modela la información sobre las características eléctricas de las redes de transmisión y distribución de energía. Este paquete es utilizado por aplicación de redes como Estimación de Estado, Flujo de carga y flujo óptimo de potencia (6).

Generation: Este paquete contiene información de las unidades de generación de energía tanto hidroeléctricas como estaciones convencionales como por combustibles fósiles, siendo algunas de estas unidades, Despacho Económico de Hydro y unidades generadoras térmicas, previsión de cargas, Control Automático de Generación y Modelado Unidad de Entrenamiento Dinámico Simulator (6).

LoadModel: Este paquete es responsable de la modelización de los consumidores de energía y la carga del sistema en forma de curvas y datos de la curva asociados. Las circunstancias especial es que puedan afectar a la carga, tales como estaciones y daytypes, también se incluyen aquí (6).

Protection: Una extensión de los paquetes básicos y Alambres de que modela la información para los equipos de protección, tales como relés. Estas entidades se utilizan dentro de los simuladores de entrenamiento y de la red de distribución de aplicaciones de localización de fallas (6).

Estos serían algunos de los paquetes más importantes dentro de este estándar los cuales son necesarios pero no imprescindibles para el correcto funcionamiento del EMS y lograr así los objetivos del mismo, los cuales son la reducción de costos y tiempo cuando se debe agregar tanto un dispositivo de hardware como una nueva funcionalidad al software sobre el cual corre el estándar además de proteger la inversión de aplicaciones o sistemas existentes que están trabajando con eficacia con un EMS (6).

1.3.4 Conclusión del estándar EMS-CIM

En la actualidad la creciente demanda de tecnología ha conllevado a una gran demanda de manejo de información así como comunicación y almacenamiento de datos por lo que la competencia de industrias desarrolladoras de software y hardware para estaciones eléctricas crece cada vez más y se ve la necesidad de crear un modelo común para todos estas tecnologías en busca de integración, flexibilidad y apertura hacia el futuro, logrado por el estándar IEC61970. Este es considerado el estándar para la automatización de equipos de subestaciones eléctricas de diversos fabricantes; fue diseñado como el único protocolo que ofrece una completa solución de comunicación. La norma IEC-61970 surge con el objetivo de garantizar la interoperabilidad entre distintos equipos electrónicos inteligentes (IED, Intelligent Electronic Device) que componen un sistema de automatización de una subestación eléctrica. Para lograr este objetivo, la norma

desarrolla un modelo de datos que recoge toda la información que puede ser necesaria en un sistema de automatización de una instalación eléctrica, de modo que todos los IEDs que cumplen con la norma organicen su información según el mismo modelo de datos. Por las ventajas anteriormente planteadas esta tecnología de modelado resuelve gran parte de los problemas fundamentales existentes en una industria de servicios eléctricos por lo que su uso es una tendencia mundial actual muy aceptada dentro de este campo siendo indispensable para el desarrollo de cualquier proyecto de este tipo, en nuestro caso el SCADA GALBA constará con este estándar ya probado y utilizado en varios software para su aceptación y desempeño en el ámbito eléctrico.

1.4 Procesador de estado de red

EL procesador de estado de red es un esquema de procesamiento digital que proporciona una base de datos en tiempo real para muchas de las funciones de control y despacho central en un sistema de energía, el mismo es un programa informático que calcula magnitudes relativas de los buses de una red eléctrica. (13) Dentro de los sistemas de gestión de la energía, la estimación del estado o procesador de estado es una función clave para la construcción de un modelo en tiempo real. Un modelo en tiempo real es una representación matemática cuasi-estática de las condiciones actuales en una red (14).
Por lo que podemos decir que un procesador de estado es la solución a nivel de software que garantiza la extracción y verificación de los datos obtenidos en tiempo real para la posterior modelación gráfica de una red eléctrica.

1.4.1 Descripción

Clasificación

Los procesadores de estado se clasifican en tres tipos de formas dependiendo de las variantes de tiempo o la naturaleza invariante de las mediciones o los modelos estáticos dinámicos de los sistemas de poder:

- Estimador estático.
- Estimador por seguimiento de estado.
- Estimador dinámico.

Estimador estático es definido como el algoritmo de procesamiento de datos para convertir las mejores lecturas redundantes e información disponible en una estimación del vector, las mediciones de datos son tomadas en tiempo invariantes y el modelo de energía es considerado (23).

Estimador por seguimiento de estado este algoritmo es basado en una simple extensión del estimador de estado estático. Estos utilizan los valores recientes de

los estados del sistema para actualizar sus estimaciones no iterativamente dentro del periodo de muestreo (23).

Estimador dinámico estos utilizan además de los estados presentes los previos a los actuales. Estos estimadores presentan la capacidad de predicción de los vectores de estado (23).

1.4.2 Funcionalidades de un procesador de estado

Análisis de observabilidad

Este análisis consiste en determinar si el número de mediciones obtenidos de los componentes estiman el vector de estado que se traduce en si son suficientes los valores recibidos para calcular el vector de estado.

Clasificación de errores en los datos

El estimador de estado computa los valores de magnitudes de voltaje y ángulo de la energía que generalmente son reales y reactivas, todo en tiempo real, por lo que se generan errores en la obtención de estas mediciones los cuales se clasifican en tres grupos:

- **Errores severos o extremos:** Son aquellos donde la diferencia absoluta entre el valor de la medición y el valor real excede los 20δ donde δ es la desviación estándar de las mediciones.
- **Error bruto:** Son aquellos donde la diferencia absoluta entre el valor de la medición y el valor real están entre 5δ y 20δ.
- **Error normal:** Son aquellos donde la diferencia absoluta entre el valor de la medición y el valor real no excede de 5δ.

1.4.3 Estructura del procesador de estado de red

1.4.3.1 Procesador de topología

Este proceso es el encargado de construir un modelo matemático a partir de las mediciones y conexiones existentes en una red de energía basándose en los componentes y sus relaciones entre sí. Esta funcionalidad es clave dentro del procesador en general ya que guarda la relación entre objetos eléctricos existentes para su posterior modelación en tiempo real de la red. Para su implementación es necesario construir una estructura de datos robusta y consistente que sea capaz de manejar eficientemente un gran cumulo de datos teniendo así acceso a cada uno de los componentes eléctricos de la red en con un rendimiento asequible.

1.4.3.2 Estimador de estado

Estimación de estado es el proceso de asignar un valor a una variable de estado del sistema, basado en mediciones sobre ese sistema y algún criterio determinado

(15). La estimación de un estado de cada componente resuelve el problema de los errores en las mediciones obtenidas, esto se realiza con métodos de optimización como mínimos cuadrados ponderados el cual devuelve el error en los datos pasados a una función determinada, este es el paso más importante del procesador de topología ya que evita la obtención de datos falsos o erróneos obtenidos de las mediciones analógicas de los dispositivos eléctricos.

El estado de un sistema queda definido por el conjunto de valores que adquieren aquellas propiedades del sistema que pueden variar (25). Por otra parte, las variables de estado de un sistema son el conjunto mínimo de variables internas del sistema que es necesario conocer para determinar el estado del sistema frente a cualquier estimulo de entrada en todo tiempo posterior (26). En el contexto de los SEP (Sistema eléctrico de potencia) esto corresponde al conjunto de tensiones complejas de los nudos del sistema.

La estimación de estado en SEP entonces significa estimar las tensiones complejas en todos los nudos eléctricos de un sistema dado, mediante el procesamiento de las medidas disponibles e información sobre la topología de la red: sus líneas, transformadores y elementos constitutivos en general.

Los tipos de mediciones más comúnmente utilizados son los siguientes:

- Flujos de potencia activa y reactiva a través de líneas y transformadores.
- Inyecciones de potencia activa y reactiva en los nodos de generación y consumo.
- Módulos de tensiones en las barras del sistema.

Aunque por lo general estos valores se obtienen a través de aparatos de medida, en la práctica, pueden utilizarse valores basados en datos históricos o predicciones, llamados Pseudo-medidas, los que si bien cuentan con una precisión inferior a las medidas ordinarias, permiten mejorar la redundancia de datos en aquellas zonas de la red pobremente monitorizadas.

Dado el modelo de red, cualquier medición sobre el sistema puede ser expresada como una función, por lo general no lineal, de las variables de estado, más un término que representa el error asociado a dicha medida.

Para un sistema de N barras, se tienen n=2N-1 variables de estado, correspondientes a los módulos de la tensión en las N barras y los N-1 ángulos medidos con respecto al voltaje en una barra de referencia llamada slack. Considerando además m medidas, lo anterior se expresa de la siguiente forma:

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \begin{bmatrix} h_1(x_1, x_2, \ldots, x_n) \\ h_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ h_m(x_1, x_2, \ldots, x_n) \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} = h(x) + e$$

Ilustración 2: Sistema vectorial

Donde:
z es el vector de medidas del sistema.
hi es la función no lineal que relaciona la medida i con el vector de estado x.
x=(x1,…,xn) es el vector de estado.
e=(e1,…,em) es vector de errores de las medidas.
El estimador de mínimos cuadrados obtiene el vector x tal que minimiza la suma del cuadrado de las componentes ei. Lo anterior puede formularse como sigue:

$$\text{Min}_x \|e\|^2 = e^T \cdot e = \sum_{i=1}^{m} [z_i - h_i(x)]^2$$

Ilustración 3: Ecuación para mínimos cuadrados

Como se observa de la ecuación todas las mediciones son tratadas en la misma manera en la formulación del estimador LS (mínimos cuadrados), luego, las mediciones más inexactas afectarán la estimación tan significativamente como las medidas más precisas, lo cual puede llevar a resultados poco satisfactorios.

Es posible enfatizar la influencia de las medidas más precisas en el resultado de la estimación, mediante la introducción de la matriz W, la cual es una matriz diagonal de dimensiones m x m, en la que cada término de la diagonal se interpreta como

el peso relativo de cada medida en el resultado del estimador. Esta formulación se conoce como estimación WLS (mínimos cuadrados ponderados).

$$\mathrm{Min}_x J(x) = e^T W e = \sum_{i=1}^{m} w_{ii}\left[z_i - h_i(x)\right]^2$$

Ilustración 4: Ecuación para mínimos cuadrados ponderados

La función de residuos J (ˆ x), donde ˆ x es el resultado de la estimación, puede interpretarse como un índice de cuanto se ajustan las medidas disponibles al estado del sistema calculado. Intuitivamente se puede observar que valores muy grandes de J (ˆ x) indican que las medidas no son consistentes entre sí.

Bajo el supuesto que el error de cada medida *ei* es una variable aleatoria que sigue una distribución normal de media cero y varianza igual al inverso del peso asociado wii, se puede demostrar que la función J (ˆ x) sigue una distribución chi-cuadrado de m-n grados de libertad, lo cual se denota ?2(n-m). Es posible entonces estimar la probabilidad con que un determinado valor de J (ˆ x) puede ser obtenido. Si esta probabilidad es más baja que una determinada tolerancia, puede considerarse que al menos uno de los valores *ei* no sigue una distribución normal, y por lo tanto el conjunto de medidas es erróneo.

1.4.4 Métodos utilizados por el estimador de estado

Para el funcionamiento del estimador de estado es necesario realizar métodos de estimación capaces de optimizar un volumen de valores para obtener las mejores aproximaciones de cada estado de la red, en el presente trabajo se tuvieron en cuenta tres de estos:

Mínimos cuadrados ponderados la solución de este método proporciona el estado estimado del (estimado de máxima verosimilitud) que satisface la siguiente condición de optimización:

$$\frac{\partial J(x)}{\partial x} = 0 \;\Rightarrow\; g(\hat{x}) = H^T(\hat{x})W[z - h(\hat{x})] = 0$$

donde

$$H(x) = \frac{\partial h(x)}{\partial x}$$

Es la matriz jacobino.

La ecuación es un sistema de n ecuaciones no-lineales, cuya solución no puede ser obtenida en forma directa. Entre los métodos iterativos para la solución de este tipo de sistemas, el más eficaz es el método de Newton-Raphson, el cual se aproxima cuadráticamente a la solución a partir de aproximaciones de primer orden de las funciones no lineales involucradas.

El desarrollo en serie de primer orden de una función genérica f(x) se muestra en la siguiente ecuación:

$$f(x) = 0$$

$$f(x) \cong f(x^k) + \frac{\partial f(x^k)}{\partial x}(x^{k+1} - x^k) = 0$$

Aplicando el desarrollo en serie a la ecuación, se obtiene lo siguiente:

$$H^T(x)W\left[z - h(x)\right] \cong H^T(x^k)W\left[z - h(x^k)\right] - H^T(x^k)WH(x^k)(x^{k+1} - x^k) = 0$$

Reordenando términos se obtiene el sistema lineal del tipo Ax=b, que debe ser resuelto en cada iteración:

$$G(x^k)\Delta x^k = H^T(x^k)W\Delta z^k$$

En donde:

$$x^{k+1} = x^k + \Delta x^k$$

denota el valor del vector de estados en la iteración k-ésima, actualizado en cada iteración,

$$\Delta z^k = z - h(x^k)$$

representa el vector de residuos del sistema en la iteración k-ésima, y

$$G(x^k) = H^T(x^k)WH(x^k)$$

es denominada matriz de ganancia. Si la matriz H es de rango completo, es decir al menos n de las m ecuaciones que la componen son linealmente independientes, esta matriz es definida positiva, y el sistema tiene solución única.

Matriz aumentada de Hachtel para la solución de este problema se utiliza la matriz aumentada de Hachtel en la cual las ecuaciones se aumentan con el vector de residuos y resuelve en cada iteración las siguientes ecuaciones:

$$\begin{bmatrix} 0 & 0 & C \\ 0 & \alpha W^{-1} & H \\ C^T & H^T & 0 \end{bmatrix} \begin{bmatrix} -\alpha^{-1}\lambda \\ \alpha^{-1}W\Delta r \\ \Delta x \end{bmatrix} = \begin{bmatrix} \Delta c \\ \Delta z \\ 0 \end{bmatrix}$$

siendo:

$$\Delta r = \Delta z - H\Delta x$$

ʎ es el multiplicador de LaGrange y α es un parámetro utilizado para controlar la estabilidad numérica del problema obteniéndose

$$\begin{aligned} \lambda' &= -\alpha^{-1}\lambda \\ \Delta r' &= \alpha^{-1}W\Delta r \end{aligned}$$

se define

$$K(x) = \begin{bmatrix} 0 & 0 & C \\ 0 & \alpha W^{-1} & H \\ C^T & H^T & 0 \end{bmatrix}$$

realizándose la factorización triangular y mediante eliminación hacia adelante y sustitución hacia atrás se obtienen las correcciones.

Este método tiene el inconveniente de destruir la simetría de la matriz debido a los pivotamientos.

Mínima mediana de cuadrados este método es desarrollado para modelos no lineales y utilizando técnicas de matrices dispersas. Siendo N el número de nudos de la red y n=2N-1, en caso de unidimensional y de regresión simple n=1 o 2, el estimador no minimiza la suma sino la medina de los cuadrados de los residuos obteniéndose:

$$J(x) = r^2_{W(k)}$$

$$k = \left[\frac{m}{2}\right] + \left[\frac{n+1}{2}\right]$$

donde r es el residuo de las medidas.

1.4.5 Objetivos principales de un procesador de estado

- Proporcionar una visión en tiempo real de las condiciones del sistema eléctrico. (14)
- Proporcionar una representación coherente para el análisis de la seguridad del sistema eléctrico. (14)
- Proporcionar diagnósticos para el modelado y mantenimiento. (14)

1.4.6 Modelado de la red

Los modelos en tiempo real de la red se construyen a partir de una combinación de instantáneas de mediciones en tiempo real y datos de la red estática. Las mediciones en tiempo real consisten en mediciones y los estados de los dispositivos de conmutación analógicos, mientras que los datos estáticos de la red corresponden a los parámetros y configuraciones básicas de la subestación. El modelo en tiempo real es una representación matemática de las condiciones actuales en una red de energía extraídos a intervalos de resultados de la estimación del estado (14).

Ejemplo de un modelo

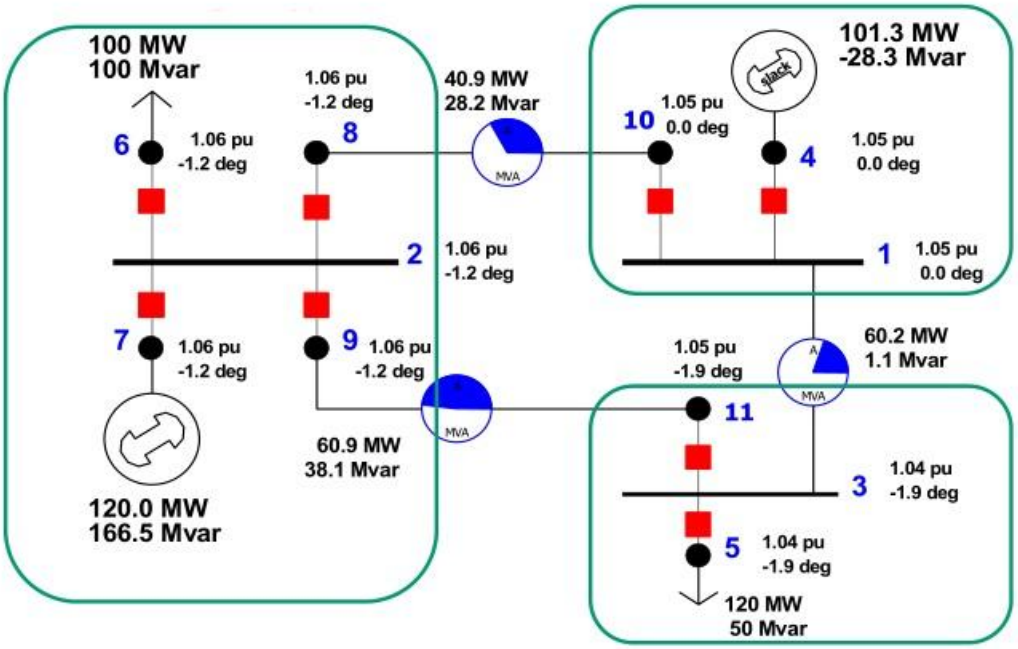


Ilustración 5: Modelo topológico de una subestación eléctrica

Conclusiones del procesador de estados

Las nuevas necesidades de modelado relacionados con la introducción de nuevos dispositivos de control y los cambios inducidos por los mercados emergentes de la energía hacen que la estimación del estado y sus funciones relacionadas sean más importante que nunca.

1.5 Metodología de desarrollo de software

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Racional (RUP).Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP (12).

Al igual que otras metodologías de desarrollo esta consta con fases de desarrollo que originalmente eran inicio, construcción, elaboración, transición, la universidad

hizo adaptaciones para un mejor acoplamiento en la producción quedando de la siguiente forma.

Fases de AUP para la UCI

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

- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elabora la arquitectura, el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.

- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.6 Herramientas y tecnología

El desarrollo de software no fuera posible sin las herramientas y tecnologías que le dan soporte, pues de no existir estas el trabajo sería engorroso, tedioso y muy lento para la obtención de los resultados. Las metodologías y herramientas a utilizar para llevar a cabo la propuesta de solución son las siguientes.

1.6.1 Lenguaje UML

Este lenguaje es crucial para el desarrollo de un sistema ya que permite la representación del software para su posterior programación por lo que brinda un mapa de desarrollo para los programadores y resto del equipo.

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML) es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software (17).

El Lenguaje Unificado de Modelado es el lenguaje que permite la modelación de sistemas de software con tecnología orientada a objetos más conocido y utilizado en la actualidad. Es un lenguaje gráfico que cuenta con un grupo de diagramas,

los cuales son utilizados para visualizar, especificar, construir y documentar un sistema de software en cada una de las etapas por las que tiene que pasar. Indica que es lo que supuestamente hará el sistema, pero no como lo hará. Para la solución propuesta se utilizará UML 2.0 con el objetivo de representar los diagramas de clases para su posterior implementación.

1.6.2 Herramienta Visual Paradigm

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son las aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas contribuyen de manera directa en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Visual Paradigm para UML es una herramienta profesional fácil de utilizar, que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a la rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, la realización de ingeniería tanto directa como inversa, generar el código desde diagramas y generar la documentación automáticamente en varios formatos como Web o Formato de Documento Portable (pdf) y el control de versiones. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto. Proporciona también abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. La única desventaja es que es una herramienta propietaria, pero se distribuyen copias gratuitas para la educación. Para la solución propuesta se utilizará CASE Visual Paradigm 8.0 con el objetivo de crear los diagramas de clases para su posterior implementación.

1.6.3 Lenguaje de programación utilizado C++

C++ es la evolución de C adaptada a la programación orientada a objetos. Se encuentra entre los más usados para desarrollar aplicaciones gráficas en 3D, por ser los que con más velocidad ejecutan el código, en general puede llegar a ser un lenguaje tan rápido como C (el más rápido después del Lenguaje Ensamblador). Además, tiene algunas cuestiones más pulidas como un control más estricto en el manejo de tipos de datos, y otras características que ayudan a la programación libre de errores (18).

Es considerado como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel. Este lenguaje de programación posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)

- Identificación de tipos en tiempo de ejecución.
- Embeber código ensamblador.

Entre las principales ventajas de C++ como lenguaje de programación orientado a objetos se encuentran:

- Es un lenguaje de propósito general, o sea, que con él se puede programar cualquier cosa, desde sistemas operativos y compiladores hasta aplicaciones de bases de datos y procesadores de texto.
- Los programas escritos en C++ tienen la ventaja de que podrán ejecutarse en cualquier máquina y bajo cualquier sistema operativo.
- Es un lenguaje multi-nivel, es decir, se puede usar para programar directamente el hardware o para crear aplicaciones tipo Windows.
- Es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

1.6.4 Técnica de programación: Programación Orientada a Objetos

Con el uso de esta técnica de programación se pretende simplificar la modificación y extensión del software para lograr una mayor reutilización del mismo, permite una fácil comprensión debido a que las estructuras del software y del problema a resolver están relacionadas directamente. Aplicando diseño orientado a objetos, se crea software resistente al cambio y escrito con economía de expresión. Esta técnica es la más usada actualmente por programadores. (18)

1.6.5 Entorno y marco de trabajo QTcreator

Los Entornos Integrados de Desarrollo, IDE (del inglés: Integrated Development Environment) constituyen programas compuestos por un conjunto de herramientas para los programadores que facilitan la escritura de programas. Pueden dedicarse en exclusivo a un sólo lenguaje de programación o bien, pueden utilizarse para varios. Los IDE facilitan un ambiente de trabajo amigable. Para el desarrollo de la extensión se decidió usar QtCreator como IDE. QtCreator ofrece un ambiente de desarrollo completo para la creación de aplicaciones Qt. Es una herramienta ligera y multiplataforma, con un enfoque estricto hacia las necesidades de los desarrolladores de aplicaciones Qt. Las principales características son:

- El editor avanzado de C++, para escribir, editar y navegar por el código fuente sin utilizar el ratón (mouse).
- Interfaz gráfica para la depuración, que incrementa la percepción de la estructura de clases de Qt.
- Integración con QtDesigner para editar los archivos de interfaz gráfica de usuario.
- La herramienta Localizador (Locator) para la navegación rápida por archivos de proyecto, funciones, clases e informaciones de ayuda.

- Integración con QtAssistant, facilitando el acceso a la ayuda de la API Qt (tipos de datos, funciones) de Qt a través de una ayuda sensible al contexto.
- Uso de diferentes sistemas para el control de versiones como Git, Subversion, CVS and Perforce.
- Construir y ejecutar proyectos Qt con la herramienta multiplataforma qmake.

Qt es un marco de trabajo (framework) escrito en C++ para el desarrollo de aplicaciones de Interfaz Gráfica de Usuario, el cual sigue la filosofía de software "escriba una vez, compile donde quiera" (write once, compile anywhere).

Además, amplía las posibilidades del lenguaje C++ con una extensa biblioteca de clases, de uso intuitivo y dividida en módulos, en los cuales se puede encontrar desde programación de interfaces gráficas de usuarios hasta programación de redes, procesamiento de textos enriquecidos, acceso a datos almacenados en varios de los gestores de bases de datos más populares. Incorpora herramientas para escribir aplicaciones robustas y en el menor tiempo posible como el Diseñador Qt (QtDesigner). A esto se le suma su extensa base de datos de ejemplos listos para usar. Qt está disponible bajo licencia LGPL, permitiendo el desarrollo de Software Libre, y si se desea, software comercial no libre, en ambos casos sin vernos obligados al pago de licencias o derechos.

1.6.6 Sistema operativo Linux

Software libre es la denominación del software que brinda libertad a los usuarios sobre un producto adquirido y por tanto, una vez obtenido, 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 (19).

Este proyecto será desarrollado a partir de las herramientas que brinda el software libre. Buscando la independencia y soberanía tecnológica que este brinda al posibilitar la libertad de uso y distribución de los programas sin incurrir en discusiones de licenciamiento o asuntos legales.

1.6.7 GNU Scientific Library

GNU Scientific Library (GSL) es una biblioteca escrita en C, destinada a cálculos numéricos en matemática y ciencia, distribuida bajo la licencia GNU GPL.

Incorpora, entre otras, rutinas para el manejo de números complejos, funciones elementales y funciones espaciales, combinatoria, algebra lineal, integración y derivación numéricas, transformadas rápidas de Fourier, transformadas wavelet discreta, generación de números aleatorios y estadística (24).

Características

La biblioteca proporciona herramientas para:(24)

- Funciones matemáticas básicas
- Números complejos
- Polinomios
- Funciones especiales
- Vectores y matrices
- Permutaciones
- Combinaciones
- Multiset
- Ordenación
- BLAS
- Álgebra lineal
- Eigenvectores
- Transformada rápida de Fourier
- Integración numérica
- Generación aleatoria de números
- Secuencias Quasi-aleatorias
- Distribuciones de números aleatorios
- Estadísticas
- Histogramas
- N-tuplas
- Integración de Monte Carlo
- Simulated annealing
- Ecuaciones diferenciales ordinarias
- Interpolación
- Derivación numérica
- Aproximaciones de Chebyshev
- Aceleración de Series
- Transformada discreta de Hankel
- Búsqueda de raíces en una y varias dimensiones
- Minimización en una y varias dimensiones
- Medida de mínimos cuadrados

- Medida de mínimos cuadrados no lineales
- Constantes Físicas
- Aritmética de punto flotante IEEE

1.7 Estructura de datos

Las estructuras de datos pueden en gran medida hacer la diferencia entre una aplicación en tiempo real rápida y una lenta en dependencia de la correcta utilización de las mismas (10).

Octree

Un octree es una estructura jerárquica que comienza con un cubo contenedor del conjunto total de los datos. Este cubo se subdivide en 8 cubos más pequeños, con los que se comprueba la intercepción de los elementos de la malla (vértices, aristas o triángulos). Aquellos cubos con los que intercepte algún elemento son nuevamente subdivididos en 8 cubos más pequeños (ver Figura 6). Este proceso se repite hasta que se satisfaga una cierta condición de parada. Esta función de parada puede venir en función de los objetivos finales de la aplicación final. Ejemplos de estas condiciones de paradas podría ser el hecho de que el tamaño del cubo este por debajo de un determinado umbral, que el número de elementos que interceptan con el cubo sea lo suficientemente pequeño o que el nivel de profundidad alcanzado llegue a un máximo predeterminado (10).

DCEL

El DCEL es una estructura de datos espacial para representar subdivisiones planares que contiene una referencia por cada cara, arista y vértice de una subdivisión. Para cada referencia se almacenan diferentes tipos de informaciones como:

- Información Geométrica: que seria las coordenadas de la geometría.
- Información Topológica: Que no es más que la relación entre vértices y aristas.
- Información Adicional: Que serían las etiquetas asignadas a cada cara.

Grafo Dual

Su estructura de relación de aristas y nodos es que cada arista conoce sus nodos adyacentes y cada nodo conoce sus aristas por lo que se recorre por las aristas reduciendo considerablemente el tiempo de búsqueda. La topología se refiere únicamente a la relación geométrica entre los elementos de una red, no con el tipo de los propios elementos. El corazón de una representación topológica de una red es la gráfica de la red. Los elementos de conexión (líneas eléctricas) se representan como los bordes de la gráfica. Un borde se dibuja como una línea,

que termina en puntos o círculos pequeños de la que otros bordes pueden emanar. En el análisis de circuitos, los bordes de la gráfica se llaman ramas. Los puntos se denominan vértices de la gráfica y representan los nodos de la red. Nodo y el vértice son términos que se pueden utilizar de manera intercambiable cuando se habla de gráficos de redes. Los gráficos utilizados en el análisis de redes son generalmente dirigidos, para capturar la dirección del flujo y voltaje de corriente, y los gráficos etiquetados, para capturar la singularidad de las ramas y nodos. Debemos puntualizar su robusta complejidad de desarrollo pero sus buenos resultados obtenidos en trabajos anteriores en esta área de desarrollo por lo que consideramos sea la más viable para resolver el objetivo principal de este trabajo (11)**.**

La estructura de datos Grafo Dual fue creada a partir de diversos criterios de diferentes estructuras de datos y la misma está basada sobre la representación de listas entrelazadas guardando la topología de la malla con una peculiaridad muy importante, almacena los índices de los vértices para la creación de cada triángulo, formando una relación única de posiciones que evita recorridos innecesarios, por estas razones la estructura de datos usada para la representación de nuestra topología de red es un grafo dual ya que es muy utilizada en este tipo de trabajo y brinda facilidades en cuanto a rendimiento y tiempo de ejecución.

Capítulo 2 Propuesta de solución

Introducción

En el presente capítulo se adquiere una visión práctica del sistema desarrollado. En el mismo se expondrán las reglas del negocio, así como los requisitos funcionales y no funcionales que regirán el desarrollo de la solución al problema, definiendo qué esperan los usuarios de la misma. Partiendo de esto se determinaron las historias de usuario. Además se describieron los procesos de las principales funcionalidades del subsistema.

2.0 Análisis y diseño

¿Qué es el modelo de dominio?

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

2.1 Representación del modelo del dominio

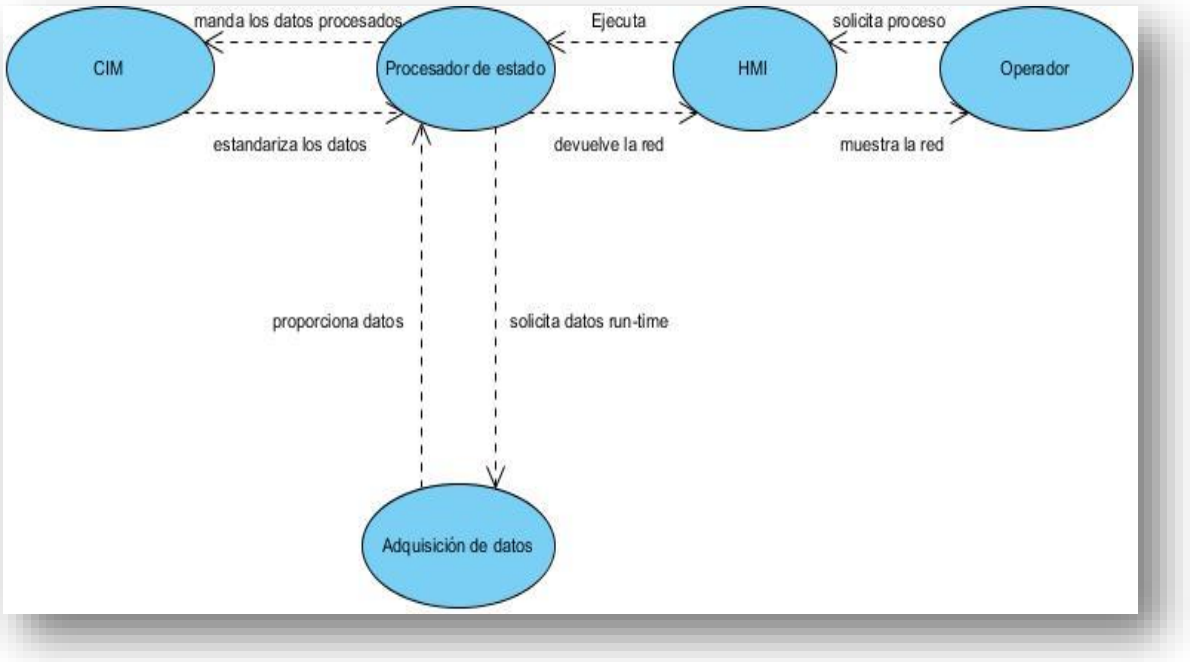


Ilustración 6: Diagrama del modelo de dominio

El **operador** es en este caso el usuario del sistema el cual solicita al sistema comenzar el proceso comunicándose con el sistema por medio de la interfaz

hombre máquina. El módulo HMI da comienzo al proceso de estado de red levantando el procesador de estado de red. EL procesador de estado solicita los datos en tiempo real del módulo de adquisición para realizar la estimación de la red. El módulo de adquisición proporciona los datos solicitados por el procesador de estado de la red. El procesador de estado luego de tener los datos necesarios los envía al CIM para su estandarización con los datos estandarizados se realiza la estimación de estado y el procesamiento topológico el que devuelve la topología de la red al HMI para que este último lo muestre al operador.

Soluciones técnicas

Para darle cumplimiento al objetivo de este trabajo, se pretende desarrollar un subsistema que se integrará al SCADA GALBA el cual tendrá como funciones la representación de una subestación eléctrica, así como la estimación de estados, además de su coloreado y edición de la misma.

2.2 Requisitos funcionales

RF1: Construir un modelo de la red eléctrica.
RF2: Graficar la topología de la subestación eléctrica.
RF3: Editar la topología de la red eléctrica.
RF4: Analizar la observabilidad del sistema eléctrico.
RF5: Estimar el estado de la red eléctrica.
RF6: Realizar el coloreado de la red eléctrica.
RF7: Mostrar los valores resultados del procesamiento de estado de red para cada nodo.

2.3 Requisitos no funcionales

Requerimientos de Restricciones en el diseño e implementación

- El sistema debe de ser desarrollado en el lenguaje de programación C++.
- Las la implementación debe ser desarrollada utilizando el marco de trabajo Qt.
- El sistema debe de cumplir con las especificaciones de las extensiones del ambiente de configuración del Guardián del ALBA.

Requerimientos de ayuda y documentación

- Cada una de las etapas del ciclo de vida del proyecto deberá contar con una documentación según la metodología establecida.
- Requerimientos de licencias y patentes.
    - ✓ Se deben utilizar herramientas libres.

Requerimientos de Portabilidad

- El sistema debe funcionar en sistemas de la familia GNU/Linux.

Requerimientos de Seguridad

- Cuando se intente realizar cualquier acción irreversible, existirá una opción de advertencia antes realizar dicha acción.

Requerimientos de Software

- Sistema operativo GNU/Linux, distribución Debian y derivados.

2.4 Historia de Usuario para los requisitos funcionales de la aplicación

Número: 1.	Nombre: Construir un modelo de la red eléctrica.
Programador: Carlos Manuel Castillo Chacón, Jose Manuel Acevedo Medina.	
Modificación de historia número:	Iteración asignada:
Prioridad en Negocio: (Alta / Media / Baja)	Puntos Estimados:
Riesgo en Desarrollo:	Puntos Reales:
Descripción: El sistema debe construir un modelo de la red eléctrica para el control, almacenamiento y accesibilidad a los componentes.	
Observaciones:	

Número: 2.	Nombre: Representar gráficamente la topología de la red.
Programador: Carlos Manuel Castillo Chacón, Jose Manuel Acevedo Medina.	
Modificación de historia número:-	Iteración asignada: 1.
Prioridad en Negocio: Alta. (Alta / Media / Baja)	Puntos Estimados: 3.
Riesgo en Desarrollo: Definido por el centro	Puntos Reales:
Descripción: El sistema debe ser capaz de representar gráficamente la estructura física de la red para un control en tiempo real de la subestación por parte del operador.	
Observaciones: Muestra cada componente que posee la subestación y sus respectivas conexiones.	

Número: 3.	Nombre: Editar la topología de la red.
Programador: Carlos Manuel Castillo Chacón, Jose Manuel Acevedo Medina.	

Modificación de historia número:-	Iteración asignada: 1.
Prioridad en Negocio: Alto. (Alta / Media / Baja)	Puntos Estimados: 2.
Riesgo en Desarrollo: Definido por el centro.	Puntos Reales:
Descripción: El sistema debe permitir al operador añadir y eliminar nodos de la representación topológica y las relaciones entre ellos.	
Observaciones: Este proceso está relacionado directamente con la estructura de datos grafo.	

Número: 4.	Nombre: Analizar la observabilidad de la red eléctrica.
Programador: Carlos Manuel Castillo Chacón, Jose Manuel Acevedo Medina	
Modificación de historia número:-	Iteración asignada: 1.
Prioridad en Negocio: Alto. (Alta / Media / Baja)	Puntos Estimados: 2.
Riesgo en Desarrollo: Definido por el centro.	Puntos Reales:

Descripción: El sistema debe realizar el análisis de observabilidad del sistema para poder realizar una estimación más exacta del estado del sistema.
Observaciones: Este proceso está relacionado directamente con la estimación de estados.

Número: 5.	Nombre: Estimación de estado.
Programador: Carlos Manuel Castillo Chacón, Jose Manuel Acevedo Medina	
Modificación de historia número:-	Iteración asignada:1.
Prioridad en Negocio: Alto (Alta / Media / Baja)	Puntos Estimados:2.
Riesgo en Desarrollo: Definido por el centro.	Puntos Reales:
Descripción: El sistema debe realizar la estimación de estado de cada nodo de la red correctamente.	
Observaciones: Este proceso permite hacer una estimación más exacta de las variables de estado del sistema eléctrico.	

Número: 6.	Nombre: Realizar el coloreo de la red eléctrica.
Programador: Carlos Manuel Castillo Chacón, Jose Manuel Acevedo Medina	
Modificación de historia número:-	Iteración asignada:1.
Prioridad en Negocio: Alto (Alta / Media / Baja)	Puntos Estimados:2.
Riesgo en Desarrollo: Definido por el centro.	Puntos Reales:
Descripción: El sistema debe representar el estado de la corriente en la red mediante un coloreado de la misma dependiendo de colores definidos por el cliente.	
Observaciones:	

Número: 7.	Nombre: Mostrar estado de los valores manejados por el estimador para cada nodo.
Programador: Carlos Manuel Castillo Chacón, Jose Manuel Acevedo Medina	

Modificación de historia número:-	Iteración asignada:1.
Prioridad en Negocio: Alto. (Alta / Media / Baja)	Puntos Estimados:2.
Riesgo en Desarrollo: Definido por el centro.	Puntos Reales:
Descripción: El sistema debe mostrar los valores manejados por el estimador para cada nodo en la representación topológica.	
Observaciones:	

2.5 Patrones de arquitectura de software

De acuerdo al Software Engineering Institute (SEI), la Arquitectura de Software se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos (27).

El término “elementos” dentro de la definición del SEI es vago a propósito, pues puede referirse a distintas entidades relacionadas con el sistema. Los elementos pueden ser entidades que existen en tiempo de ejecución (objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios). Por otro lado, las relaciones entre elementos dependen de propiedades visibles (o públicas) de los elementos, quedando ocultos los detalles de implementación. Finalmente, cada conjunto de elementos relacionados de un tipo particular corresponde a una estructura distinta, de ahí que la arquitectura está compuesta por distintas estructuras.

Dentro de los objetivos de los patrones se encuentran:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

2.5.1 Patrón utilizado

Modelo Vista Presentación

El patrón Modelo Vista Presentador (MVP) es un patrón derivado del patrón Modelo Vista Controlador (MVC) que nos ayuda a ofrecer una clara separación entre la vista, el modelo y el controlador.

La clave del patrón MVP es una estricta regulación de la interacción entre la vista y el controlador, aunque en el patrón MVP, al controlador se le conoce como presentador.

La vista y el modelo de datos son claramente separados a través de un contrato (interfaz) que expone la vista y al cual el presentador accede de modo polimórfico.

Resumiendo, podemos decir que el patrón MVP es una mejora del patrón MVC basado en tres características:

- La vista no conoce el modelo.
- El presentador es independiente de la tecnología de interfaz de usuario.
- La vista y el presentador es testeable puesto que está basada en un contrato.

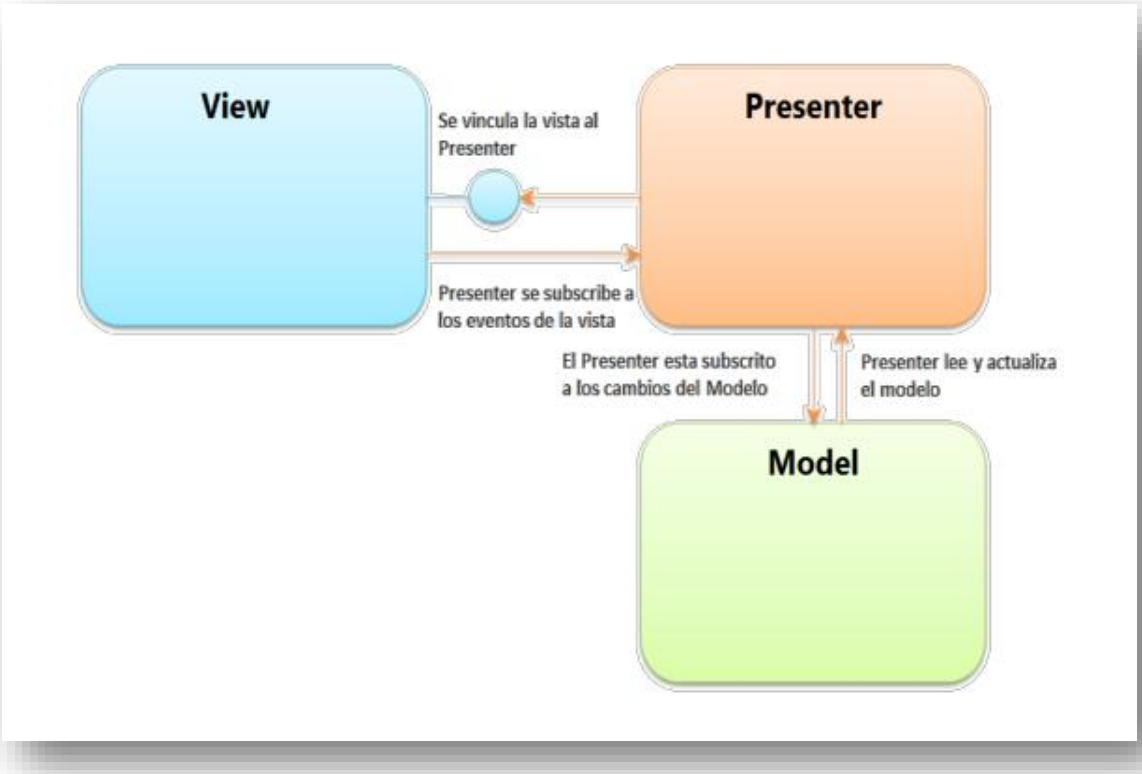


Ilustración 7: Arquitectura del sistema

Vista (Interfaz Gráfica de Usuario)

Es la que se encarga de mostrar los datos y comandos de rutas de usuarios al presentador para actuar sobre esos datos.

Presentador

Es el que se encarga de presentar las acciones del usuario al sistema y obtener la respuesta del mismo.

El presentador se ubica entre la vista y modelo y actúa de la siguiente manera:

1. Recibe los datos de la vista.
2. Convierte los datos de la vista en acciones que se ejecutan contra el sistema.
3. Con la respuesta del sistema actualiza los datos de la vista.

Modelo

Es donde residen las funciones que se ejecutan, se reciben las peticiones del presentador, se procesa la información y se envían las respuestas tras el proceso. Aquí radica la lógica del negocio.

2.7 Patrones de diseño

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

Los patrones de diseño pueden ser clasificados:

- **Creacional:** Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de que clase crear o cómo crearla. Según donde se tome dicha decisión se pueden clasificar los patrones de creación en: patrones de creación de clases (la decisión se toma en los constructores de las clases y usan la herencia para determinar la creación de las instancias) Los patrones de creación son (22).

- **Estructural:** Son los que plantean las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Lo fundamental son las relaciones de uso entre los objetos, y éstas están determinadas por las interfaces que soportan los objetos. Estudian cómo se relacionan los objetos en tiempo de ejecución. Sirven para diseñar las interconexiones entre los objetos. Los patrones estructurales son (22).

- **Comportamiento:** Plantea la interacción y cooperación entre las clases. Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal. Los patrones de comportamiento son (22).

2.8.1 Patrones utilizados

Composite: Patrón de diseño muy poderoso en sí mismo, consiste en crear objetos a partir de otros más pequeños. Permite combinar objetos en estructuras de árbol para representar jerarquías. Permite además, agrupar varios objetos como si fueran uno solo. El mismo fue utilizado en la creación de objetos gráficos.

Template Method: Es un patrón de diseño que define una estructura algorítmica en la súper clase, delegando la implementación a las subclases. Es decir, define una serie de pasos, en donde los pasos serán redefinidos en las subclases. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura. El mismo fue utilizado para pintar los objetos gráficos.

Visitor: Proporciona una forma fácil y sostenible de ejecutar acciones en una familia de clases. Este patrón centraliza los comportamientos y permite que sean modificados o ampliados sin cambiar las clases sobre las que actúan. El mismo fue utilizado para recorrer el grafo.

2.9 Diagramas de clases propuestos como solución

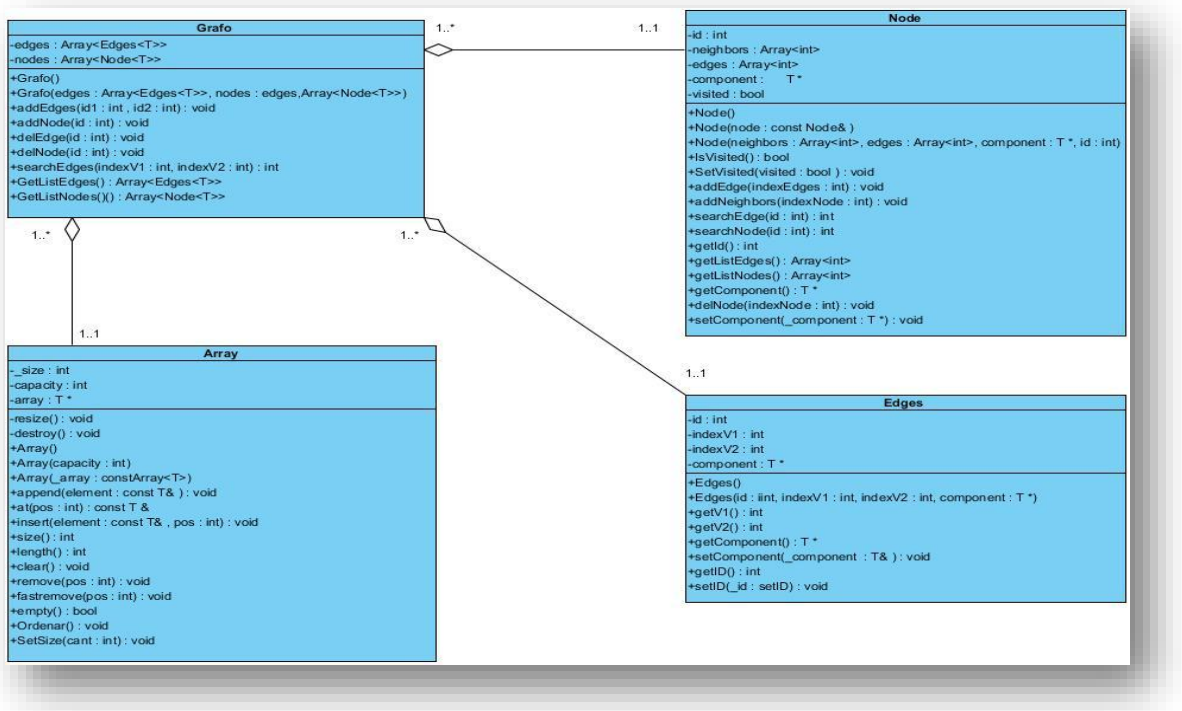


Ilustración 8: Diagrama de clase Grafo Dual

La estructura de datos propuesta es un grafo dual el cual consta de cuatro clases que lo conforman para el correcto funcionamiento de la misma además es totalmente genérica.

Las clases son identificadas como:

Array es la encargada de funcionar como un contenedor de objetos y manipular los mismos, a pesar que existen clases previamente diseñadas en C++ se implementó la siguiente para que surtiera solo los objetivos necesarios para el proyecto así se evita código innecesario a la hora de ejecutar el software además de facilitar el trabajo con esta clase.

Edges es la encargada de comportarse como una arista del grafo conociendo sus nodos y además comportándose como un componente de conexión esta clase contiene una de las particularidades del grafo dual ya que optimiza el tiempo de búsqueda al conocer los nodos a los que está conectada.

Node es la encargada de comportarse como un nodo del grafo los mismo representan cada componente de la red eléctrica además de conocer sus nodos vecinos y las aristas que llegan hasta su dominio.

Grafo es la clase más importante del grafo ya que construye y maneja tanto los nodos como las aristas del grafo dándole forma a la estructura de datos para obtener una representación de la red eléctrica a nivel de componentes para su posterior tratamiento en el procesador de estado de red.

2.10 Diagrama de clases del estimador de estado

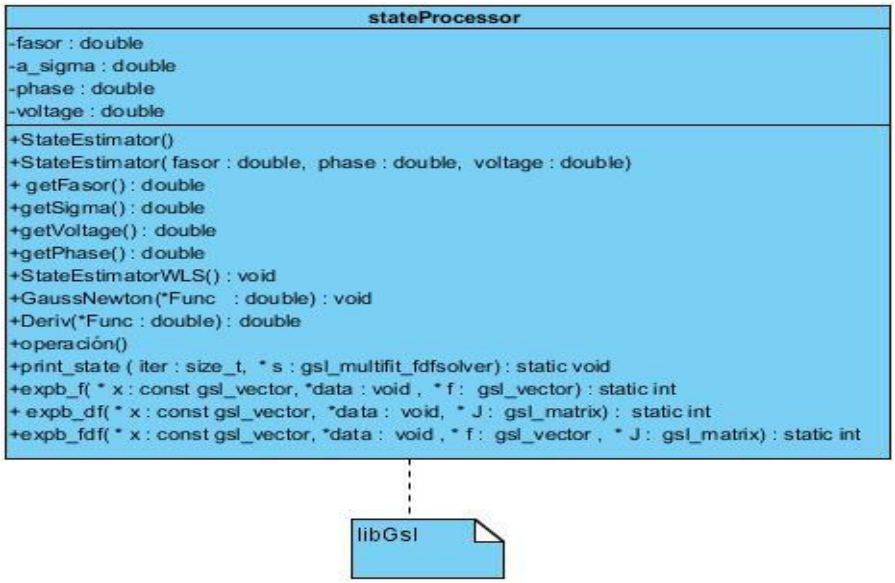


Ilustración 9: Diagrama del procesador de estado

2.11 Modelo de despliegue

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Este diagrama es útil para ilustrar la arquitectura física de un sistema. A continuación se muestra una representación gráfica donde se encuentra el HMI situado en la PC cliente con la funcionalidad adicionada.

Los artefactos representan elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo. En el caso propuesto tenemos el módulo HMI como código fuente del sistema y la funcionalidad desarrollada procesador de estado de red el cual es agregado al módulo HMI.

Destino de despliegue está generalmente representado por un nodo que es o bien de los dispositivos de hardware o bien algún entorno de ejecución de software. En la solución propuesta constamos como plataforma de ejecución una pc cliente.

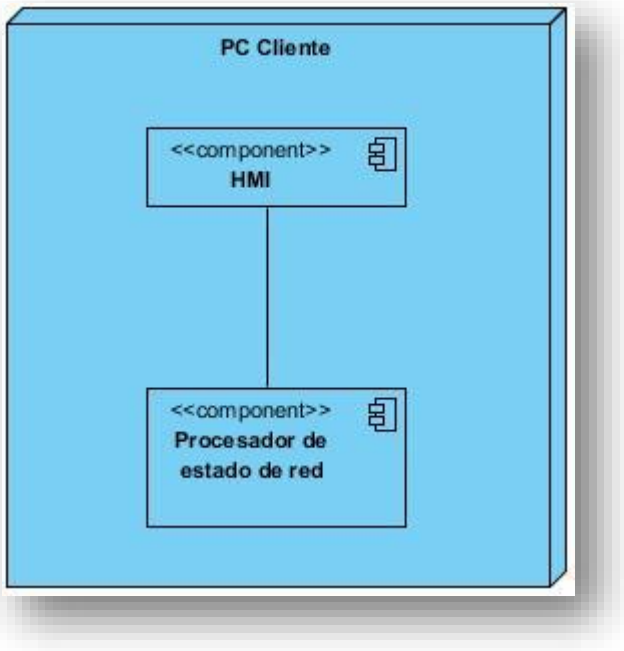


Ilustración 10: Diagrama de despliegue

2.12 Diagrama de componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de

componentes de software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente, se realizan por partes. Cada diagrama describe un apartado del sistema y se representa como un grafo de componentes software unidos por medio de relaciones de dependencia.

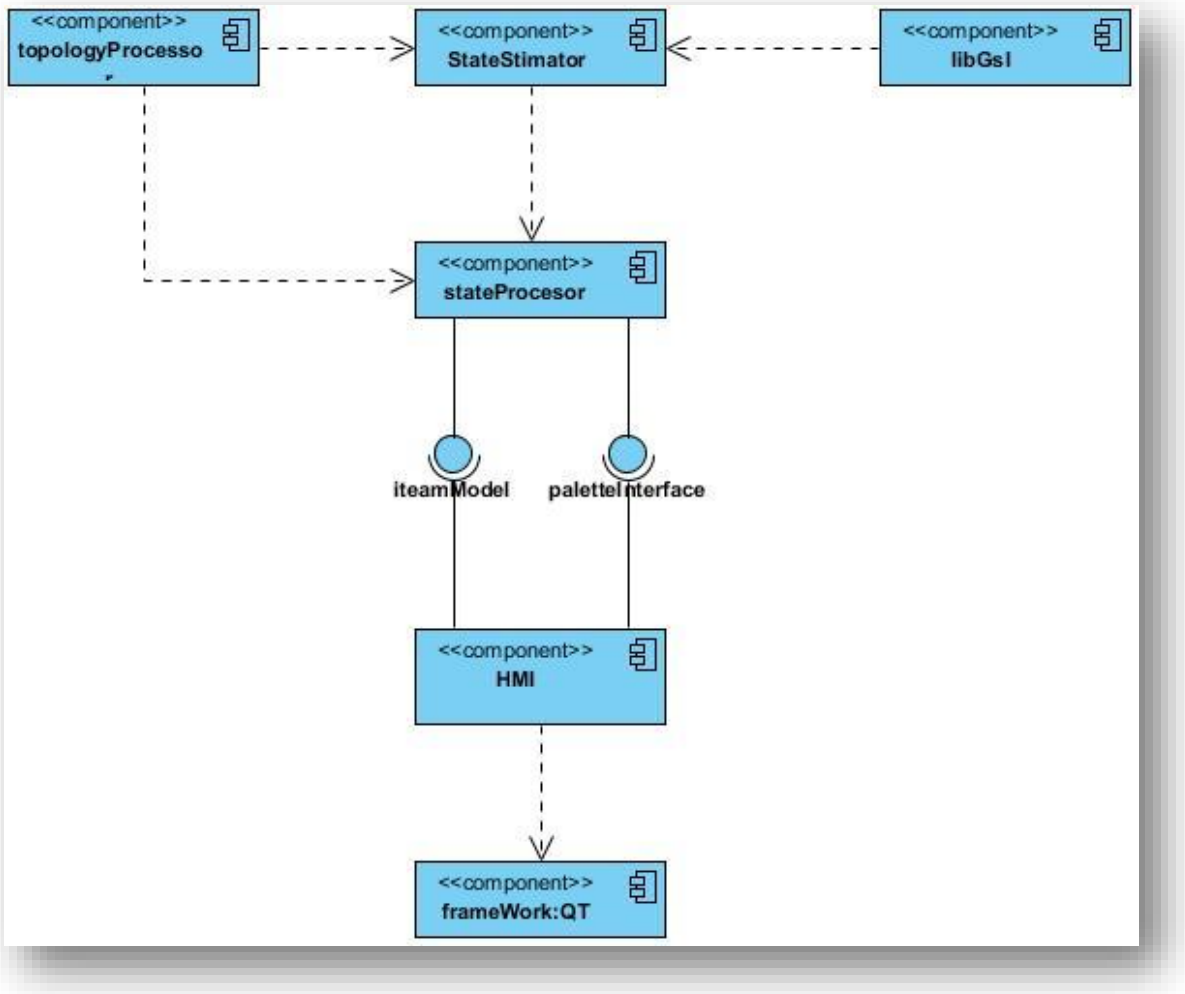


Ilustración 11: Diagrama de componente

El componente procesador de estado de red representa la funcionalidad agregada la cual es ejecutada por el usuario a través de la interfaz de usuario para mediante la paleta de componentes y los ítems necesarios construir una red eléctrica que es procesada por el procesador de topología y el estimador de estado con el uso de la biblioteca libgsl.

2.13 Solución

Como solución fue desarrollado un software capaz de cumplir con los requisitos que exige un procesador de estado de red, respondiendo a las funcionalidades siguientes:

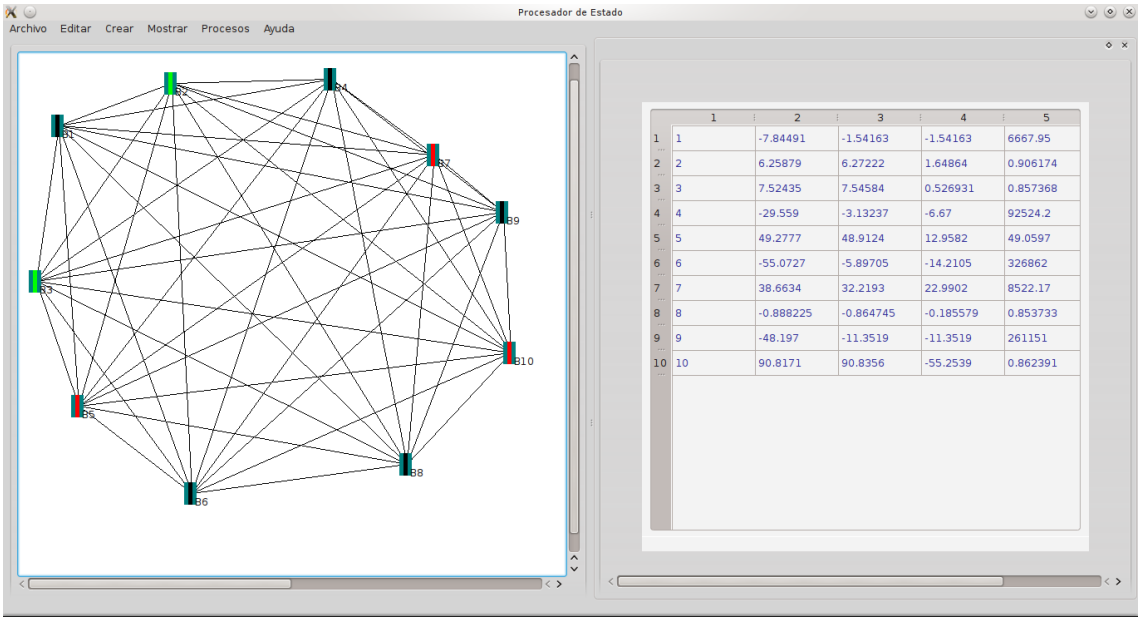
1. El software permite al operador confeccionar la topología de una subestación eléctrica para su monitoreo en tiempo real seleccionando el menú editar/subestación y confeccionando la red componente a componente llenando los campos necesarios.

2. La aplicación permite al operador realizar el análisis de observabilidad seleccionando el menú procesos/análisis de observabilidad.

3. La aplicación responde a la estimación de estado cuando el operador la solicita en menú procesos/estimación de estado luego de haber realizado el análisis de observabilidad.

4. El software luego de haber estimado el estado de la red muestra en la tabla inferior derecha los valores obtenidos.

Se implementó un software de prueba (demo) del procesador de estado completamente funcional dando como salida una herramienta que puede integrarse a las adecuaciones del SCADA GALBA para el sector eléctrico y además puede usarse como una herramienta independiente por operadores de otros sistemas.



	1	2	3	4	5
1	1	-7.84491	-1.54163	-1.54163	6667.95
2	2	6.25879	6.27222	1.64864	0.906174
3	3	7.52435	7.54584	0.526931	0.857368
4	4	-29.559	-3.13237	-6.67	92524.2
5	5	49.2777	48.9124	12.9582	49.0597
6	6	-55.0727	-5.89705	-14.2105	326862
7	7	38.6634	32.2193	22.9902	8522.17
8	8	-0.888225	-0.864745	-0.185579	0.853733
9	9	-48.197	-11.3519	-11.3519	261151
10	10	90.8171	90.8356	-55.2539	0.862391

Ilustración 12: Procesador de Estados de Red

Capítulo 3: Pruebas y validaciones del sistema

3.1 Introducción

En el presente capítulo se abordarán las pruebas necesarias para verificar el correcto funcionamiento del software y para junto con el cliente probar y aceptar el sistema además de revisar las tareas realizadas y ver el estilo de implementación que se utilizó.

3.2 Estilo de código

Líneas: Se utiliza solo una instrucción por línea de código para una mejor legibilidad, comprensión y limpieza en el trabajo.

Las líneas de código no son superiores a 80 caracteres, que es el tamaño de línea habitual en editores de texto plano de sistemas operativos como Unix y Linux.

```
int searchEdges(int id1, int id2);
Array<Edges<T> > *GetListEdges();
Array<Node<T> >  *GetListNodes();
int searchNode(int id);
```

Ilustración 13: Ejemplo de estilo de código en cuanto a lianas

Indentación o sangrado: El código está indentado, de una forma adecuada y consistente. La indentación (o sangrado) ayuda a ver la estructura del programa y facilita la identificación de posibles errores.

```
for (int j = 0; j < nodes->size(); ++j) {
    if (nodes->at(j)->getId()==edges->at(i).getV1()) {
        nodes->at(j)->delNode(edges->at(i).getV2());
        nodes->at(j).delEdge(id);
    }
```

Ilustración 14: Ejemplo de estilo de código en cuanto a sangrado

Bloques: Se utiliza llaves para delimitar todos los bloques de sentencias de control y bucles. Las llaves están en la misma columna (solos, sin código ejecutable).

```
for (i = 0; i < n; i++)
{
    /* Model Yi = V * sin(lambda+b) */
    double t = i;
    double Yi=V* cos(lambda*t+b) ;
    gsl_vector_set (f, i, (Yi - y[i])/sigma[i]);
}
```

Ilustración 15: Ejemplo de estilo de código en cuanto a bloques

Paréntesis: Se hace uso de los paréntesis para hacer más claro el orden de evaluación de operadores en una expresión, incluso en situaciones en las que puede no ser obligatorio utilizarlos.

```
size_t n = ((struct data *)data)->n;
double *y = ((struct data *)data)->y;
double *sigma = ((struct data *) data)->sigma;
```

Ilustración 16: Ejemplo de estilo de código en cuanto a paréntesis

Espacios en blanco: Se hace uso generoso de líneas y espacios en blanco en pro de hacer más claro y comprensible el código. Las funciones y los bloques de código son aislados unos de otros, usando varias líneas en blanco para facilitar su lectura y poner de manifiesto su carácter de unidad de programación.

Así conseguiremos tener un código de fácil lectura (si fuera excesivamente compacto sería poco legible), y en el que sea sencillo localizar funciones y bloques de código.

```
int StateEstimator::expb_f (const gsl_vector * x, void *data, gsl_vector * f)
{


int StateEstimator::expb_df (const gsl_vector * x, void *data,gsl_matrix * J)
{

int StateEstimator::expb_fdf (const gsl_vector * x, void *data,gsl_vector * f, gsl_matrix * J)
{
```

.

Ilustración 17: Ejemplo de estilo de código en cuanto a espacios en blanco

Comentarios:

Los comentarios son minimizados (si se utilizan comentarios muy largos puede entorpecer la lectura), facilitan la comprensión del código y aportan información útil sobre las sentencias, bloques, variables, funciones que afectan.

```
/**
 * Funcionalidad que permite adicionar una arista al grafo dado el indice de su vertice
 *@brief addEdges
 * @param indexV1
 * @param indexV2
 */
void addEdges(int id1,int id2);
```

Ilustración 18: Ejemplo de estilo de código en cuanto a comentarios

3.3 Tareas de ingeniería

Las tareas de ingeniería son pequeñas actividades que los desarrolladores conocen que el sistema debe hacer, y son deducibles a partir de las historias de usuario. Estas deben ser estimables, su tiempo de implementación debe ser corto (aproximadamente entre uno y tres días) y responden directamente a la historia de usuario de la cual se deriva. Una historia de usuario puede tener una o varias tareas dependiendo de su complejidad. También pueden existir tareas de ingeniería técnica, que son aquellas que no responden a ninguna historia de usuario, pero son necesarias para que el sistema funcione bajo los requerimientos especificados (21).

Para el desarrollo de este acápite se propone el siguiente modelo para tareas de ingeniería:

Tareas de ingeniería	
Número de la tarea: N	**Número de HU:** N
Nombre de la tarea: Nombre	
Tipo de tarea: Tipo	**Días estimados:**
Programador(es) responsable(s): Responsable	

Descripción: Descripción

Argumentada en los siguientes campos:

Número de la tarea: Número consecutivo asignado a la tarea, ayuda a organización de dicho artefacto.

Número de la historia de usuario: Una tarea de ingeniería debe responder a una, o varias historias de usuario, en este campo se especifica la historia o historias a las que responde la tarea. En caso de ser una tarea técnica, este campo se deja en blanco.

Nombre de la tarea: Nombre asignado a la tarea de ingeniería, debe expresar explícitamente su objetivo.

Tipo de tarea: Una tarea puede variar su tipo en dependencia de su objetivo, puede ser, como se abordó anteriormente, una tarea técnica, pero también, respondiendo a una historia de usuario, una tarea de desarrollo, de diseño, o de una revisión y/o prueba específica.

Programador(es) responsable(s): Dispone quién o quiénes del equipo de desarrollo son responsables de la ejecución de la tarea.

Días estimados: Una tarea de ingeniería se puede expresar en horas o días, en dependencia del sistema de trabajo y calendario del equipo de desarrollo. La sumatoria de tiempo de las tareas de ingeniería de una historia de usuario no puede sobrepasar el valor convertido a puntos estimados de la propia historia. **Descripción:** Expresa una breve descripción sobre los objetivos de la tarea de ingeniería.

Tareas realizadas para la elaboración del software

No. HU	Tareas	No. Tarea
	Instalación y configuración del entorno de trabajo.	1
1, 2, 3	Implementación de la estructura de datos.	2
	Integración de la estructura de datos al sistema.	3
4	Integración de la librería libgsl al sistema.	4
	Obtención de los valores de cada nodo a	5

	estimar.	
	Implementación del estimador de estado de red.	6
2	Implementación del área de trabajo del operador.	7
5	Implementación del método de coloreado de la red.	8
	Implementación de la visualización del coloreado de la red en el modelo topológico.	9
6	Obtención de los valores reales del estimador de estado.	10
	Implementación del visual para mostrar los valores manejados en el en el sistema.	11
	Ejecución de las pruebas necesarias para validar el correcto funcionamiento software.	12

Tareas realizadas durante la primera iteración del sistema para su desarrollo:

Tareas de ingeniería	
Número de la tarea: 1	**Número de HU:**
Nombre de la tarea: Instalación y configuración del entorno de trabajo.	
Tipo de tarea: Técnica.	**Días estimados:** 2.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón.	
Descripción: Se configura adecuadamente el entorno de trabajo para el desarrollo de la aplicación.	

Tareas de ingeniería	
Número de la tarea: 2.	**Número de HU:** 1, 2, 3.
Nombre de la tarea: Implementación de la estructura de datos.	
Tipo de tarea: Desarrollo.	**Días estimados:** 56.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Se implementa una estructura de datos robusta que brinda cualidades ventajosas para el software desarrollado.	

Tareas de ingeniería	
Número de la tarea: 3.	**Número de HU:** 1, 2, 3.
Nombre de la tarea: Integración de la estructura de datos al sistema.	
Tipo de tarea: Desarrollo.	**Días estimados:** 7
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Se integra el grafo al sistema almacenándole los componentes para su manipulación.	

Tareas de ingeniería	
Número de la tarea: 4.	**Número de HU:** 4.
Nombre de la tarea: Integración de la lib-gsl.	
Tipo de tarea: Desarrollo.	**Días estimados:** 21.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Se resuelven los problemas de adaptación de la lib-gsl para el sistema.	

Tareas de ingeniería	
Número de la tarea: 5.	**Número de HU:** 4.
Nombre de la tarea: Obtención de los valores.	
Tipo de tarea: Desarrollo.	**Días estimados:** 14.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Se manejan los valores obtenidos de los componentes de la subestación eléctrica.	

Tareas de ingeniería	
Número de la tarea: 6.	**Número de HU:** 4.
Nombre de la tarea: Implementación del estimador de estado.	
Tipo de tarea: Técnica.	**Días estimados:** 28.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Se implementan las funcionalidades necesarias para el correcto funcionamiento del estimador.	

Tareas de ingeniería	
Número de la tarea: 7.	**Número de HU:** 2.
Nombre de la tarea: Implementación del área de trabajo.	
Tipo de tarea: Desarrollo.	**Días estimados:** 7
Programador(es) responsable(s): Carlos Manuel Castillo Chacón.	
Descripción: Se desarrolla el área donde el operador interactúa con el software.	

Tareas de ingeniería	
Número de la tarea: 8.	**Número de HU:** 5.
Nombre de la tarea: Implementación del método de coloreo de la red.	
Tipo de tarea: Desarrollo.	**Días estimados:** 7.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Se implementa la funcionalidad del coloreado de la red dependiendo de estándares.	

Tareas de ingeniería	
Número de la tarea: 9.	**Número de HU:** 5.
Nombre de la tarea: Implementación de la representación visual del coloreado de la red.	
Tipo de tarea: Desarrollo.	**Días estimados:** 5.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón.	
Descripción: Se representa visualmente el coloreado de la red.	

Tareas de ingeniería	
Número de la tarea: 10.	**Número de HU:** 6.
Nombre de la tarea: Obtención de los valores reales del estimador de estado.	
Tipo de tarea: Desarrollo.	**Días estimados:** 7.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Obtención de los datos devueltos por el estimador de estado.	

Tareas de ingeniería	
Número de la tarea: 11.	**Número de HU:** 6.
Nombre de la tarea: Implementación del visual para mostrar los valores manejados en el en el sistema.	
Tipo de tarea: Desarrollo.	**Días estimados:** 5.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Se implementa la representación visual de los valores obtenidos de la estimación.	

Tareas de ingeniería	
Número de la tarea: 12.	**Número de HU:**
Nombre de la tarea: Pruebas del sistema.	
Tipo de tarea: Técnicas.	**Días estimados:** 2.
Programador(es) responsable(s): Carlos Manuel Castillo Chacón y Jose Manuel Acevedo.	
Descripción: Se realizan las pruebas necesarias para validar el software.	

3.4 Pruebas

Los casos de prueba son pruebas unitarias o funcionales que se le realizan al sistema. Las pruebas unitarias son validaciones desde la perspectiva del desarrollador, mientras que una prueba funcional se realiza por un usuario externo a la aplicación, generalmente, usuario final o cliente. El objetivo general de una prueba es certificar que la historia de usuario está lista. Mientras un código no se ha probado no existe, por lo que los casos de prueba deben acompañar al sistema durante su ciclo de explotación (20).

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

Para las pruebas del software procesador de estado de red se decidió aplicar las pruebas de aceptación bajo la siguiente estructura:

Caso de prueba de Aceptación	
Número: N	**Historia de Usuario:** N
Nombre: Nombre	
Condiciones de ejecución: Condiciones	
Entradas/Pasos de ejecución: Pasos	
Resultado esperado: Resultado esperado.	
Evaluación de la prueba: Evaluación.	

Número: Representa el número del caso de prueba. Este debe ser consecutivo.

Historia de usuario: Número de la historia de usuario a la que responde el caso de prueba.

Condiciones de ejecución: Condiciones previas que deben cumplirse para la realización del caso de prueba.

Entradas/Pasos de ejecución: Secuencia de pasos o entradas que se realizan para validar la funcionalidad que se prueba.

Resultados esperados: Describe los objetivos concretos de la funcionalidad.

Evaluación de la prueba: Evaluación obtenida luego de realizada la prueba. Puede ser satisfactoria o no satisfactoria.

A continuación se muestran algunas de las pruebas de aceptación realizadas a las funcionalidades de la aplicación.

Caso de prueba de Aceptación	
Número: 1.	Historia de Usuario: 1.
Nombre: Construcción del modelo de la red eléctrica.	
Condiciones de ejecución: Debe construirse dependiendo de los valores obtenidos para cada nodo.	
Entradas/Pasos de ejecución: 1. El usuario levanta el procesador de estado. 2. El sistema debe crear automáticamente el modelo.	

Resultado esperado: Construcción del grafo.
Evaluación de la prueba: Satisfactoria.

Caso de prueba de Aceptación	
Número: 2.	**Historia de Usuario:** 2.
Nombre: Representación de la red eléctrica.	
Condiciones de ejecución: Debe de estar construido el modelo matemático.	
Entradas/Pasos de ejecución: 1. El usuario selecciona construir subestación. 2. El operador agrega cada componente a la red eléctrica. 3. El sistema debe aceptar los cambios hechos por el usuario.	
Resultado esperado: Obtención del modelo físico de la subestación eléctrica y almacenamiento de los nodos de la red adecuadamente a la estructura de datos.	
Evaluación de la prueba: Satisfactoria.	

Caso de prueba de Aceptación	
Número: 3.	**Historia de Usuario:** 3.
Nombre: Edición de la topología.	
Condiciones de ejecución: Existencia de un modelo físico de la red.	
Entradas/Pasos de ejecución: 1. El usuario selecciona agregar o eliminar nodo. 2. Completa los requisitos pedidos por la aplicación. 3. El sistema edita la subestación.	
Resultado esperado: La topología de la red correctamente editada tanto en la representación visual como en el modelo matemático o estructura de datos.	
Evaluación de la prueba: Satisfactoria.	

Caso de prueba de Aceptación	
Número: 4.	**Historia de Usuario:** 4.
Nombre: Análisis de observabilidad.	
Condiciones de ejecución: El estimador debe conocer los valores que necesita de cada nodo para realizar el análisis de observabilidad.	
Entradas/Pasos de ejecución: 1. El usuario selecciona la opción de analizar la observabilidad del sistema.	
Resultado esperado: El sistema debe realizar el análisis de observabilidad de la subestación eléctrica.	
Evaluación de la prueba: Satisfactoria.	

Caso de prueba de Aceptación	
Número: 5.	**Historia de Usuario:** 5.
Nombre: Estimación de estado.	
Condiciones de ejecución: El estimador debe conocer los valores que necesita de cada nodo a estimar.	
Entradas/Pasos de ejecución: 1. El usuario selecciona la opción de estimar estado.	
Resultado esperado: El sistema debe estimar el estado del nodo y devolver un vector de estado.	
Evaluación de la prueba: Satisfactoria.	

Caso de prueba de Aceptación	
Número: 6.	**Historia de Usuario:** 6.
Nombre: Coloreado de la red.	
Condiciones de ejecución: Se debe haber hecho la estimación de estado.	
Entradas/Pasos de ejecución: 1. El usuario selecciona la opción estimar estado.	

Resultado esperado: Resultado esperado.
Evaluación de la prueba: Evaluación.

Caso de prueba de Aceptación	
Número: 7.	**Historia de Usuario:** 7.
Nombre: Mostrar el estado de los componentes.	
Condiciones de ejecución: El sistema debe haber realizado la estimación de estado.	
Entradas/Pasos de ejecución: 1. El usuario selecciona la opción de estimar estado.	
Resultado esperado: Cada nodo debe mostrar sus valores correspondientes que maneja la aplicación.	
Evaluación de la prueba: Satisfactoria.	

3.4.1 Sumario de evaluación de las pruebas

Para la ejecución de las pruebas de aceptación se contó con un total de 7 requisitos funcionales. Con la realización de las mismas, se cubrió el 100 % de dichos requisitos. Para la aplicación de estas pruebas se realizaron tres iteraciones. En una primera iteración se diseñaron 7 casos de pruebas de los cuales 3 fueron satisfactorios dando lugar a 4 no conformidades, las cuales fueron resueltas. En una segunda iteración se diseñaron 7 casos de pruebas de los cuales 6 fueron satisfactorios dando lugar a 1 no conformidad, la cual fue resuelta, dando lugar que en una tercera iteración de 7 casos de pruebas diseñados todos fueron satisfactorios, de esta manera quedó validada la aplicación con un 100% de aceptación.

Cobertura de las pruebas

Para obtener la cobertura de las pruebas de Aceptación que fueron ejecutadas en el sistema se recogieron los siguientes resultados:

1. **Primera iteración**
    - Casos de prueba diseñados: 7
    - Casos de prueba aplicados: 7
    - Cobertura de pruebas (ejecutadas) = 100%
    - Cobertura de pruebas (exitosas)= 3/7= 43%
2. **Segunda iteración**
    - Casos de prueba diseñados: 7
    - Casos de prueba aplicados: 7
    - Cobertura de pruebas (ejecutadas) = 100%
    - Cobertura de pruebas (exitosas)= 6/7= 86%
3. **Tercera iteración**
    - Casos de prueba diseñados: 7
    - Casos de prueba aplicados: 7
    - Cobertura de pruebas (ejecutadas) = 100%
    - Cobertura de pruebas (exitosas)= 7/7= 100%

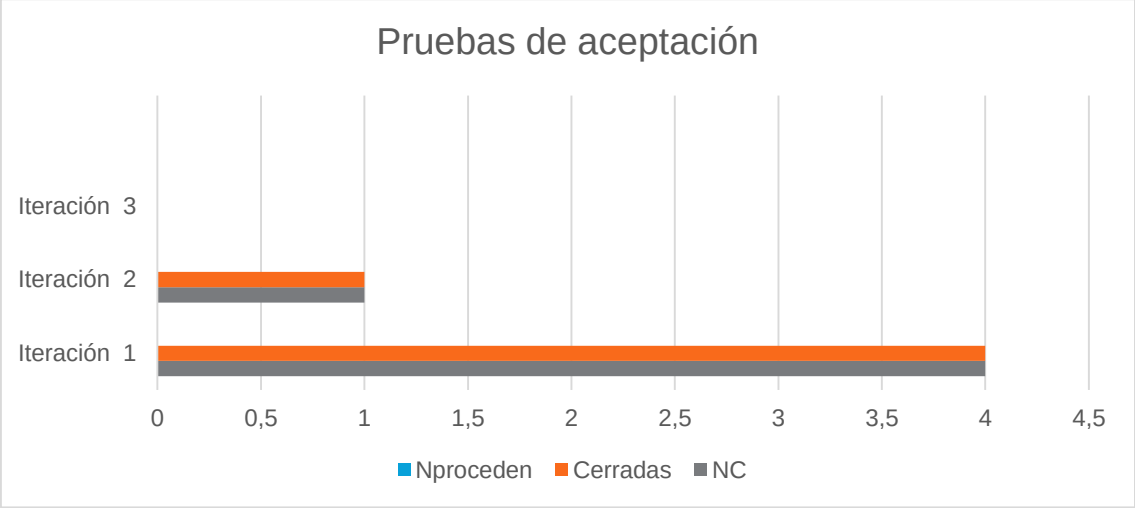


Ilustración 19: Gráfica de las pruebas realizadas

3.5 Consideraciones finales

El proceso de implementación satisfizo las necesidades de la plataforma, lo cual fue comprobado mediante las pruebas realizadas al sistema dando lugar a un correcto funcionamiento de la plataforma. Con el fin de este capítulo se da por terminada la propuesta que trae consigo este trabajo.

Conclusiones

Al término de la presente investigación sobre procesador de estado de red bajo el modelo CIM se arriba a las siguientes conclusiones:

- Se formalizó un estudio del arte de sistemas procesadores de red de energía y EMS similares lo que permitió realizar el análisis y diseño basado en la metodología AUP para el desarrollo del software.
- Se implementaron los paquetes Topología y Wires del modelo CIM, resolviendo problemas de compatibilidad y complejidad de desarrollo presentes en software anteriores de estas características.
- Se obtuvo un procesador de estado de red que permite la creación y edición del modelo topológico de una subestación eléctrica, realiza un análisis de observabilidad de los datos obtenidos, estima el estado de la red eléctrica y muestra el error obtenido en la estimación de estado.
- Se desarrolló una herramienta de prueba (demo) para verificar el correcto funcionamiento del procesador de estado de red.

Recomendación

Se recomienda completar el código necesario en el SCADA GALBA para la integración del procesador de estado de red.

Referencias bibliográficas

1. **Herrera Vázquez, Moisés**. Introducción a la Arquitectura del Guardián del ALBA. 2008.
2. **Meza, Luis Eduardo Chavarría**. SCADA System's & Telemetry. 2007.
3. **Lozano, Carlos de Castro**. Introducción SCADA.
4. **Industrial, Centro de Informática**. Sistemas SCADA desarrollados por el CEDIN.
5. **Díaz, Ing. Henry Mendiburu**. Sistemas SCADA.
6. **International Electrotechnical Commission.** Energy Management System Application Program Interface (EMS-API) –Part 301:Common Information Model (CIM) Base.
7. **InfoPLC**. Top 5 de aplicaciones SCADA para iPhone y iPad.
8. **"IEC 61850**, El Nuevo Estándar en Automatización de Subestaciones".
9. Interfaz Hombre Máquina Selección de Tecnologías Versión 0.1, 11/03/2002.
10. **Frank Rufino Nápoles** ,DECIMACIÓN DE GEOMETRÍAS OBTENIDAS POR MARCHING CUBE.
11. **Surech Kumar,** Electric Circuit and Network.
12. **Tamara Rodríguez Sánchez,** Metodología de desarrollo para la Actividad productiva de la UCI.
13. **F.C. Schweppe and J.Wildes**, "Power System Static State Estimation, Part-I: Exact Model".
14. **A.Monticelli**, "Electric Power System State Estimation"
15. A. Wood, B. Wollenberg, "Power Generation, Operation and Control", 2nd Edition, John Wiley and Sons Inc. New York, 1996, ISBN: 0-471-58699-4.Carlos Reynoso – Nicolás Kicillof, " Estilos y Patrones en la Estrategia de Arquitectura de Microsoft"
16. **Booch, Grady, Rumbaugh, James y Jacobson, Ivar**. "El Lenguaje Unificado de Modelado. s.l: Addison Wesley".
17. **Benítez Herrera, Alejandro y Saurín Ojeda, Gelson Rafael**. "Módulo para el comportamiento autónomo de autos en un Entorno Virtual urbano". Ciudad de la Habana: s.n., 2008.
18. **González Abad, Israeldis.** Diseño e Implementación del Sistema de Gestión de Información de los Recursos de la Facultad 3. Ciudad de la Habana: s.n., 2008.
19. **Sehara Driggs, Yosell Luis**. Implementación de un modelo para la configuración de un sistema SCADA. Ciudad de la Habana : s.n., 2008.
20. **Gómez Argüello, Wilson Javier**. Metodología de desarrollo de software un enfoque práctico y global versión 1.0.11.
21. **Penedés, Patricio Letelier y Mª Carmen**. Métodologías ágiles para el desarrollo de software.

22. **Pérez Mariñán, P.** “Patrones de Diseño (Design Patterns)”
23. **A. Abur y A. Gómez Expósito**. Power system state estimation, theory and implementation, Marcel Dekker.
24. Edition 1.16, for GSL Version 1.16 17 July 2013, GNU Scientific Library.
25. **R. Román**, “Apuntes del curso Termotecnia ME-43A", Departamento de Ingeniería Mecánica de la Facultad de Ingeniería de la Universidad de Chile, Santiago, 2002.
26. **F.E. Cellier**, “Continuous System Modeling”, Springer-Verlag Inc, New York, 1991, ISBN: 0-387-97502-0.
27. **L. Bass, P. Clements, R. Kazman**, Software Architecture in Practice, 2nd Edition, Addison Wesley, 2003.
28. Guia_Arquitectura_N-Capas_DDD_NET_4.pdf

Anexos

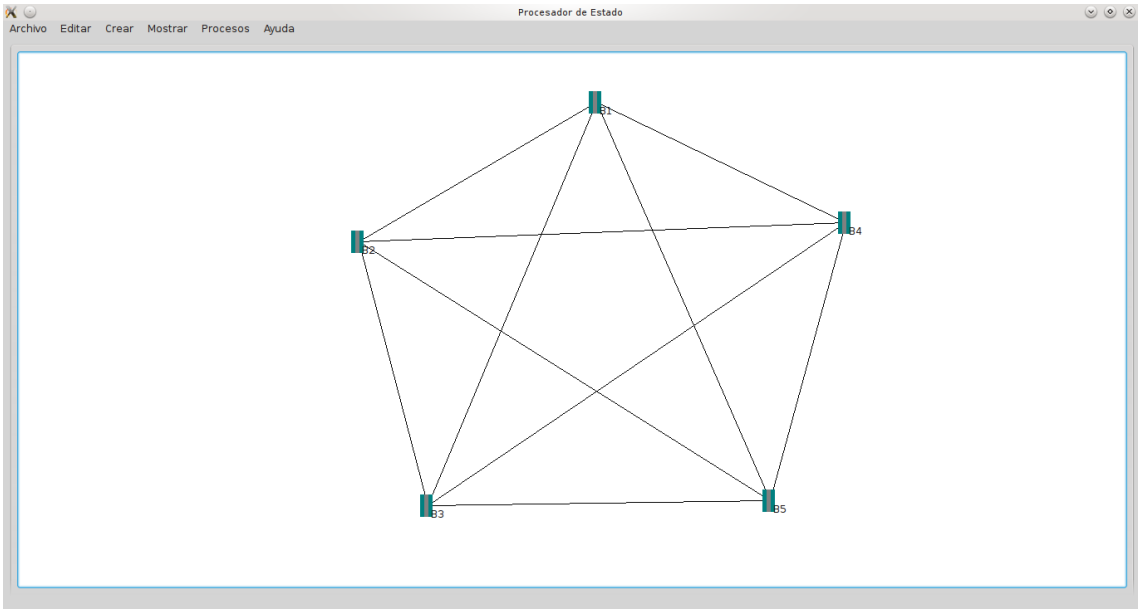


Ilustración 20: Edición de topología del Procesador de Estado de Red

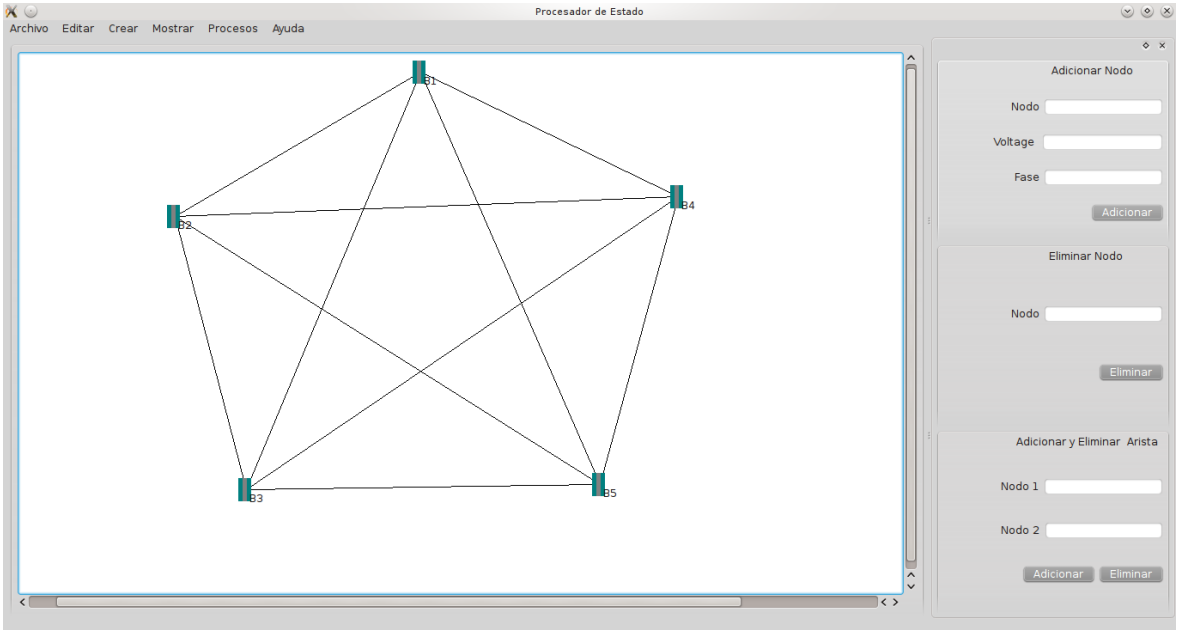
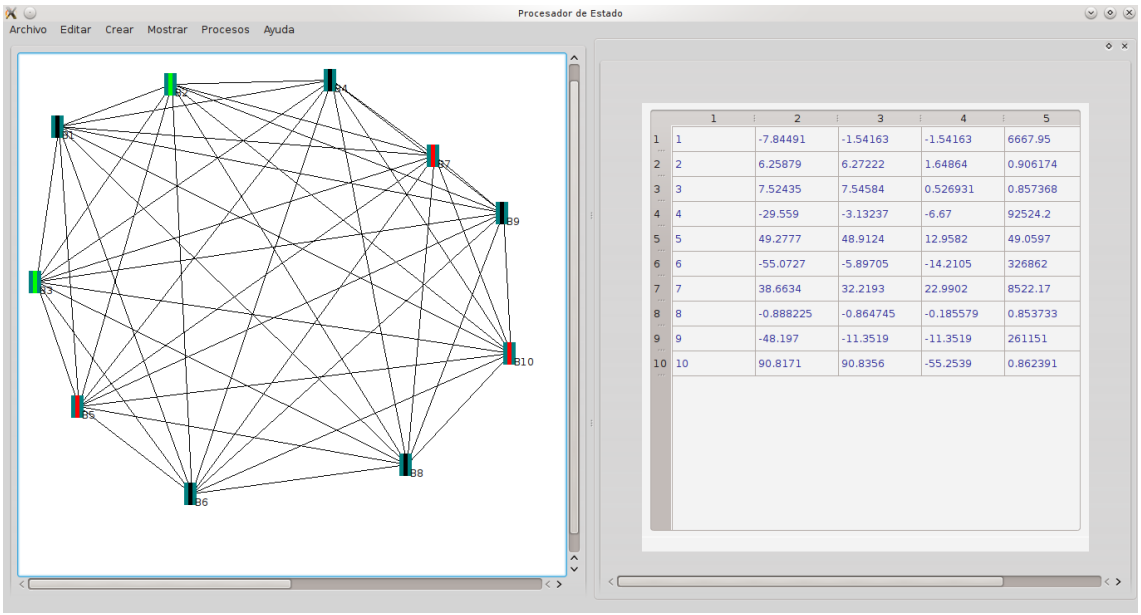


Ilustración 21: Edición de topología del Procesador de Estado de Red



	1	2	3	4	5
1	1	-7.84491	-1.54163	-1.54163	6667.95
2	2	6.25879	6.27222	1.64864	0.906174
3	3	7.52435	7.54584	0.526931	0.857368
4	4	-29.559	-3.13237	-6.67	92524.2
5	5	49.2777	48.9124	12.9582	49.0597
6	6	-55.0727	-5.89705	-14.2105	326862
7	7	38.6634	32.2193	22.9902	8522.17
8	8	-0.888225	-0.864745	-0.185579	0.853733
9	9	-48.197	-11.3519	-11.3519	261151
10	10	90.8171	90.8356	-55.2539	0.862391

Ilustración 22: Estimación de Estados del Procesador de Estados de Red