Universidad de las Ciencias Informáticas

Facultad 5

Título: Desarrollo del flujo de requisitos para el subsistema de Gráficos Vectoriales del producto SCADA Nacional.

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

Autor: Iliana Pérez Pupo
Irina Elena Argota Vega

Tutores: Ing. Amado Espinosa Hidalgo
MSc. Yaneisy Hernández Hernández

Consultante: Lic. Juan Antonio Fung Goizueta

Ciudad de la Habana, Mayo 2007

DECLARACIÓN DE AUTORÍA

Por este medio declaramos que somos las únicas autoras de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) para que haga el uso que estime pertinente con este trabajo.

Para que así conste firmamos la presente a los 18 días del mes de mayo del 2007.

Firma del Autor	Firma del Autor
(Irina Elena Argota Vega)	(Iliana Pérez Pupo)
Firma del Tutor	Firma del Tutor
(MSc. Yaneisy Hernández Hernández)	(Ing. Amado Espinosa Hidalgo)

DATOS DE CONTACTO

MSc. Yaneisy Hernández Hernández

Graduada de Ingeniero en automática en el 2000 y Master en automática mención en Sistemas Computacionales en el año 2004. Especialista Superior de la Empresa de Automatización Integral (CEDAI). Tiene siete años de experiencia en la especialidad.

Ing. Amado Espinosa Hidalgo

Graduado de Ingeniero Informático en el 2004 y profesor instructor con tres años de experiencia docente en la Universidad de las Ciencias Informáticas (UCI) y cuatro en el desarrollo de software.

Lic. Juan A. Fung Goizueta.

Licenciado en Física en 1986, Profesor Asistente, Especialista en Microelectrónica de la Universidad de Sofía. Experiencia Docente de 17 años. Experiencia en la coordinación y dirección de proyectos informáticos de 3 años.

AGRADECIMIENTOS

Gracias en primer lugar a mi familia toda… en especial a mi mamá que me ha dado el beso de amor que nunca me ha faltado, a mi papá por enseñarme que las "cosas hay que ganárselas y que la palabra del triunfo es alcanzable mientras se estudia" gracias por tanto amor y por ese "irina1" que me hizo sentir grande y afortunada de tenerlo como padre, a mi hermanito por quererme tanto y ojalá la presente investigación, le sirva de inspiración a coger una buena carrera universitaria, a mi abuela Ramona por estar pendiente siempre a mis cositas, por ese abrazo que reconforta y respira pureza, a mi abuela María por tenerme siempre presente en sus oraciones y por combatir la distancia. A mi padrino Nene por su cariño y preocupación, y mi madrina por estar y por sus comidas ricas. A Julián por ser padre y amigo y Marlen por tanta "energía positiva". En fin, a todos los que han "diluido su arena" en mi…

A mis amigas y amigos: Diani, Ivo, Lenna, Raule, Pedry, Leonel, Mr. B, Manuel, Ani…por soportar a esta "petulante e inicua" que los quiere con la vida, por compartir, confiar…¡Iliana ya puedes reír en japonés!

Irina

Aprovechando esta última oportunidad de agradecimientos en mi carrera universitaria, quisiera decir "gracias" a todos los que han mostrado interés y preocupación por mi bien estar, felicidad y nivel profesional.

A mis padres por sus consejos, ánimos, esperanzas y profunda confianza en mí; por sus constantes llamadas en espera de alguna noticia, sus predicciones de advertencia, su infinito cariño a pesar de los pormenores y sus esfuerzos incesantes y sosegados por hacer de mí, una persona madura.

A Irina por su paciencia ante mis actitudes, a Anita por su gran ejemplo de amistad.

A Millet por su ayuda como profesor, padre y amigo, su gran aporte a mi preparación profesional y defensa personal. A Piñero por las primeras revisiones del presente documento de tesis y preparación inicial a esta última tarea, por ayudarme con la impresión de las lecciones del curso de japonés.

Iliana

A nuestro tutor Amado por soportar nuestras dudas, madrugadas de revisión y por ser tan valiente al tutoriar la tesis y nuestra tutora Yaneisy, por su amistad, cariño y respuestas inmediatas a las interrogantes. A Fung por su gran ejemplo de optimismo, profesionalidad y espiritualidad, amigo, líder e ídolo. A Rene por su ayuda y aporte al nivel cognitivo y formación como ingenieras. A nuestros compañeros de grupo.

Un agradecimiento especial a nuestro Comandante en Jefe y a la obra de la Revolución por permitirnos dar hoy un "clic" al mundo de la informática.

DEDICATORIA

A mis dos mamás y dos papás.
Especialmente a mi mamá Maga, que aunque no esté, siempre estuvo en mi vida, en mis primeros pasos, al igual que mi papá Antonio…
A mis padres…por tanto amor y entrega, por permitirme crecer…

Irina

A Yamila y Jorge Luis, mamá y papá respectivamente, que con tanta paciencia y sutileza se han esmerado en mi formación.

Iliana

RESUMEN

Con el objetivo de “desarrollar un Sistema que supervise, controle, optimice y gestione los procesos industriales de la Corporación Estatal de Petróleos de Venezuela S.A. (PDVSA), sirviendo de base para la implantación en otras Industrias y Organismos de Venezuela”, este país estableció un convenio con Cuba, específicamente con la Universidad de Ciencias Informáticas (UCI) para la creación de un Sistema de Adquisición de Datos y Control Supervisorio (SCADA) en inglés, “Supervisory Control And Data Acquisition”.

El diseño del SCADA desde su origen debe tener aplicaciones de supervisión y control en tiempo real con la suficiente flexibilidad y potencialidad para asegurar la cobertura de las necesidades de la empresa y que garantice todo lo planteado en el estado del arte actual de un SCADA. Dada la complejidad y el amplio alcance de este software se tiende a desarrollar en módulos, por funcionalidades, que posteriormente se comunicarán entre ellos. Entre estos módulos se encuentra el de visualización, por lo que se creó la línea de trabajo de Gráficos Vectoriales dentro del proyecto, cuya función es representar todo este proceso mediante sinópticos gráficos que se almacenan en la computadora y se generan en el editor gráfico del SCADA que se desarrollará, mostrándose en funcionamiento mediante el Ambiente Ejecución.

El equipo de trabajo del proyecto no poseía un análisis y especificación de las funcionalidades a incluir en la Interfaz de usuario de Gráficos Vectoriales; tarea que se le asignó a los analistas del proyecto, recopilándose dicho trabajo en la presente investigación.

PALABRAS CLAVES

SCADA, Ingeniería, requisitos, software, analista, gráficos, SVG, HMI.

TABLA DE CONTENTIDOS

INTRODUCCIÓN

El desarrollo de la industria informática en Cuba, la necesidad de la inserción del país en el mercado internacional de la informática y el fortalecimiento de la amistad con los pueblos americanos, son firmes y sólidas causas de desarrollo e investigación del sector informático en Cuba.

En estos momentos se está dedicando gran parte de esta fuerza para el fomento del progreso de la economía nacional y venezolana. Esto se ha encausado debido a los fuertes lazos de amistad entre ambos gobiernos, donde se le dio carácter oficial desde que ambos presidentes: Hugo Chávez Frías, en Venezuela y Fidel Castro Ruz, de Cuba, firmaron un acuerdo de cooperación integral en el Salón Ayacucho del Palacio de Miraflores el 30 de octubre de 2000. **(PDVSA)**

La "Ley Orgánica de Hidrocarburos", elaborado por el Ejecutivo Nacional de la República Bolivariana de Venezuela, el cual hace pertenecer a la República Bolivariana todos los yacimientos mineros que estén sobre o bajo su territorio, además de convertirlos en bienes del dominio público, por lo tanto inalienables e imprescriptibles, reafirmándose la propiedad del Estado venezolano sobre sus recursos, fue publicada en la Gaceta Oficial N° 36.793 el jueves 23 de septiembre de 1999.

Esto conllevó a inconformidades y provocaciones por parte de los derechistas de PDVSA, quienes realizaron un sabotaje petrolero durante diciembre 2002 y enero del 2003, ocasionando pérdidas al país.

Dichos trabajadores fueron expulsados de la industria , surgiendo la “Nueva PDVSA”, donde se construye un cuerpo interno con mecanismos de evaluación de mejoras, normativas y reglamentos, usada para operar y explotar los recursos bajo la conducción del Estado venezolano que desarrolla una política internacional de defensa de su principal recurso: el petróleo.

Buscando la independencia tecnológica, liberándose de la alta demanda de los productos en el mercado, surge en el 2004 la idea de desarrollar el proyecto “SCADA Nacional”, cuyo objetivo es: “Desarrollar un Sistema que supervise, controle, optimice y gestione los procesos industriales de PDVSA sirviendo de base para la implantación en otras Industrias y Organismos del país.”

En ese mismo año, se declaró la migración de software propietario a software libre, estableciendo leyes que promulgaban su utilización en las distintas ramas sociales y económicas del territorio venezolano**. (CHAVEZ 2004)**

Producto a las relaciones Cuba-Venezuela, y a los proyectos existentes con la UCI y a los precedentes existentes en empresas y universidades del país con respecto al desarrollo de SCADAs se asume el proyecto “SCADA Nacional”, conformando un equipo multidisciplinario de estudiantes, profesores y

especialistas automáticos de todo el país para el desempeño de esta importante tarea; que exigía sus componentes desarrollados en software libre, lo cual era un mundo nuevo por conocer.

Un sistema SCADA es una aplicación software especialmente diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador. Además, provee a diversos usuarios, toda la información que se genera en el proceso productivo, control de calidad, supervisión, mantenimiento. **(SCADA 2005)**

Entre los módulos o bloques de software que desarrolla un SCADA se encuentran: Configuración, Módulo de Procesos, Gestión de Archivos y de Datos, comunicación e Interfaz Gráfica del Operador, este último proporciona al operador las funciones de control y supervisión de la planta. Mediante sinópticos gráficos se representa el proceso. Estos sinópticos son almacenados en el computador consola de operación y generados desde un editor gráfico que puede estar incorporado al SCADA o importados desde otra aplicación durante la configuración del sistema.

El equipo de desarrollo del proyecto SCADA Nacional, no posee un análisis y especificación de las funcionalidades a incluir en la interfaz de Gráficos, lo que imposibilita el diseño de este módulo e integración con el sistema.

El Analista del Sistema se encarga de darle cumplimento a esta situación problemática, el cual "... es responsable del conjunto de requisitos que están modelados en los casos de usos, lo que incluye todos los requisitos funcionales y no funcionales que son casos de uso específico...de delimitar el sistema, encontrando los actores y los casos de uso y asegurando que el modelo de casos de uso es completo y consistente (...)" **(JACOBSON IVAR)**

Para dar solución a la situación problemática existente se tiene que responder la presente interrogante del **Problema Científico**: ¿Cómo lograr desarrollar el flujo de trabajo de requerimientos que propone la metodología RUP (Rational Unified Process) para el módulo Gráficos Vectoriales del SCADA integrando varios equipos de trabajo?

El **Objeto de Estudio** de este trabajo es el Proceso de Captura de Requisitos, y el campo de acción, el desarrollo de flujo de requisitos para el módulo de Gráficos Vectoriales del producto SCADA.

De acuerdo con el problema científico planteado la **Idea a Defender** definida es la siguiente:

Si se especifican los requerimientos básicos para visualizar los procesos supervisados y controlados de un sistema SCADA, detallando cada funcionalidad en forma de caso de uso e implementando sus

respectivos prototipos de interfaces entonces el módulo de Gráficos Vectoriales dispondría de una especificación de requisitos que guíe el proceso de desarrollo de software.

El **Objetivo General** que se desea alcanzar para darle cumplimiento a este trabajo es:

Desarrollar el flujo de requisitos para el subsistema Gráficos Vectoriales del producto SCADA Nacional.

Definiéndose los siguientes **Objetivos Específicos** para lograr un mayor nivel de detalles en la investigación:

- Definir qué tecnologías se utilizarán en el desarrollo del módulo de Gráficos.
- Especificar los requerimientos.
- Diseñar el prototipo de interfaz gráfica del SCADA.

De acuerdo a los objetivos mencionados se plantearon **Tareas de la Investigación** por las cuales se regirá esta investigación, son las sucedidas a continuación:

- Estudiar sobre el módulo Gráficos de un SCADA.
- Realizar el levantamiento de requerimientos del software.
- Realizar un análisis de las técnicas y herramientas de la ingeniería para la captura de requisitos.
- Identificar los casos de uso.
- Describir detalladamente los casos de uso arquitectónicamente significativos.
- Realizar el modelo de casos de uso.
- Estudiar sobre la herramienta Rational Software Arquitect (RSA).
- Adquirir y aplicar el lenguaje de modelado estándar 2.0 (UML 2.0).
- Diseñar el prototipo de interfaz de usuario.

Para el desarrollo de las tareas científicas se combinan diferentes **Métodos y Técnicas en la búsqueda y procesamiento de la información**, los fundamentales son:

A nivel teórico

Métodos de análisis-síntesis e inducción-deducción: Para el estudio de las concepciones y conceptos empleados en el campo; para analizar las teorías y documentos generados por desarrolladores de otros SCADAS como el Movicon, el OASyS, el Eros y otros, permitiendo la extracción de los elementos más importantes que se relacionen con el desarrollo de un SCADA.

Análisis histórico-lógico: Para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a los SCADAS desarrollados, sobre todo en lo referente al módulo de Gráficos; además de conceptos, términos y vocabularios propios del campo como variables, Widgets (controles), Interfaz Hombre - Máquina o HMI (Human Machine Interface), dispositivos de control, puntos digitales y analógicos y los diferentes tipos de alarmas.

Método de modelación: Para la caracterización de las nuevas funcionalidades que tendrá el módulo Gráficos cuando se estaban definiendo los requisitos funcionales.

A nivel empírico

Entrevistas: En el diagnóstico de necesidades, fueron entrevistados no sólo un grupo de profesionales, ingenieros y especialistas que desarrollan un determinado rol dentro del proyecto y con experiencias de trabajos con sistemas SCADAS, sino también a veteranos trabajadores de la industria de PDVSA para conocer sobre el antiguo funcionamiento del sistema industrial petrolero.

Experimentos: En la elaboración de prototipos funcionales, con el objetivo de comprobar los Casos de Uso que se fueron analizando.

El presente documento está estructurado en tres capítulos:

En el **Capítulo 1** se describe la concepción general del sistema, se hace referencia a las características de un SCADA, y de los SCADAs existentes en el mercado y en Cuba. Se tratan los principales conceptos de la Ingeniería de Software que tienen que ver con la investigación.

En el **Capítulo 2** se hace referencia a las tendencias y tecnologías existentes en la actualidad que se deben considerar para hacer la selección de aquellas que se van a utilizar en el proyecto.

En el **Capítulo 3** se describe la solución propuesta, realizándose la modelación del sistema para la línea de Gráficos Vectoriales. Se muestran los requerimientos, casos de usos, diagramas de actividades y el prototipo de interfaz de usuario.

FUNDAMENTACIÓN TEÓRICA

CAPITULO

1

1.1 Introducción

En el presente capítulo se hace alusión a los módulos fundamentales de un SCADA para la gestión y control de datos, se mencionan características de los SCADAs y conceptos de la Ingeniería de Software que se vinculan con el proceso a desarrollar en la presente investigación.

1.2 Concepción general del sistema.

El SCADA es una aplicación software de control de producción que se comunica con los dispositivos de campo y controla el proceso de forma automática desde la pantalla del ordenador, proporcionando de esta forma información del proceso a diversos usuarios entre los que se encuentran los operadores, supervisores de control de calidad, supervisión, mantenimiento y otros.

Los sistemas de interfaz entre usuario y planta, que están basados en paneles de control y repletos de indicadores luminosos, instrumentos de medidas y pulsadores, están siendo sustituidos por sistemas digitales que implementan el panel sobre la pantalla de un ordenador.

El control directo lo realizan los Controladores Lógicos Programables (PLC). Éstos están conectados a un ordenador que realiza las funciones de diálogo con el operador, tratamiento de la información y control de la producción utilizando el SCADA. **(LUGONES 2005)**

Un SCADA debe cumplir tres **funciones principales**:

- **Adquisición de datos** para recoger, procesar y almacenar la información recibida.
- **Supervisión** para observar desde un monitor la evolución de las variables de control.
- **Control** para modificar la evolución del proceso, actuando bien sobre los reguladores autónomos básicos (consignas, alarmas, menús, etc.) bien directamente sobre el proceso mediante las salidas conectadas.

1.2.1 Requisitos de un sistema SCADA

Un SCADA debe cumplir varios objetivos para que su instalación sea perfectamente aprovechada:

- Debe ser sistemas de arquitectura abierta, capaces de crecer o adaptarse según las necesidades cambiantes de la empresa.
- Deben comunicarse con toda facilidad y de forma transparente al usuario con el equipo de planta y con el resto de la empresa (redes locales y de gestión).
- Deben ser programas sencillo de instalar, sin excesivas exigencias de hardware, y fáciles de utilizar, con interfaces amigables de usuario. **(SCADA 2005)**

1.2.2 Elementos básicos de los sistemas SCADAs

Interfaz Hombre – Máquina o HMI (Human Machine Interface)

Es el aparato que presenta los datos a un operador (humano) y a través del cual éste controla el proceso.

La industria de HMI nació esencialmente de la necesidad de estandarizar la manera de monitorear y de controlar múltiples sistemas remotos, PLCs y otros mecanismos de control. Aunque un PLC realiza automáticamente un control pre-programado sobre un proceso, normalmente se distribuyen a lo largo de toda la planta, haciendo difícil recoger los datos de manera manual, los sistemas SCADA lo hacen de manera automática.

Históricamente los PLC no tienen una manera estándar de presentar la información al operador. La obtención de los datos por el sistema SCADA parte desde el PLC o desde otros controladores y se realiza por medio de algún tipo de red, posteriormente esta información es combinada y formateada. Un HMI puede tener también vínculos con una base de datos para proporcionar las tendencias, los datos de diagnóstico y manejo de la información así como un cronograma de procedimientos de mantenimiento, información logística, esquemas detallados para un sensor o máquina en particular, incluso sistemas expertos con guía de resolución de problemas.

Desde cerca de 1998, virtualmente todos los productores principales de PLC ofrecen integración con sistemas HMI/SCADA, muchos de ellos usan protocolos de comunicaciones abiertos y no propietarios. Numerosos paquetes de HMI/SCADA de terceros ofrecen compatibilidad incorporada con la mayoría de PLCs, incluyendo la entrada al mercado de ingenieros mecánicos, eléctricos y técnicos para configurar estas interfaces por sí mismos, sin la necesidad de un programa hecho a medida escrito por un desarrollador de software.

SCADA es popular debido a esta compatibilidad y seguridad. Ésta se usa desde aplicaciones pequeñas, como controladores de temperatura en un espacio, hasta aplicaciones muy grandes como el control de plantas nucleares. **(WIKIPEDIA 2007)**

Unidad Maestra (MTU)

El Ordenador Central o Unidad Maestra (MTU), en inglés Master Terminal Unit, es el elemento central de control de un sistema de adquisición de datos y control supervisorio. Este término ha sido comúnmente utilizado para designar al sistema electrónico de computación que adquiere toda la información procedente de las Unidades Terminales Remotas (RTUs) y que la presenta de una forma a una RTU para ejecutar una acción de control remoto. La capacidad funcional de una estación maestra incluye todas las tareas de recolección de datos y envío de comandos remotos. Adicionalmente las funciones de la MTU incluyen el almacenamiento de información histórica, programación, despacho y ejecución de tareas específicas tales como reportes y contabilidad de producción. **(LUGONES 2005)**

Unidades Remotas (RTU):

Las Unidades Terminales Remotas (RTUs), en inglés Remote Terminal Units, son dispositivos de adquisición de datos y control en campo, cuya función principal es hacer de interfaz entre los equipos de instrumentación y control local y el SCADA. **(LUGONES 2005)**

Canales o Medios de Comunicación:

La efectividad y confiabilidad operacional de un sistema SCADA depende en gran medida de la transmisión de datos entre la estación maestra y las unidades terminales remotas, por lo tanto, debe ser provisto de un medio a través del cual se establezca el intercambio de datos entre éstas unidades de una forma coordinada, confiable y segura.

El sistema de comunicación que forma parte de un sistema SCADA es el conjunto de elementos, dispositivos y equipos de transmisión de datos a través de los cuales se realiza el intercambio efectivo de mensajes entre las RTUs y la MTU.

Entre los componentes del sistema de comunicación de un SCADA típico, se encuentran las interfaces de comunicación digital, módems, medios de transmisión de datos, el computador frontal de comunicaciones. (CFE, Communication Front End) y el protocolo de comunicación. **(LUGONES 2005)**

Instrumentación de Campo:

Constituyen uno de los elementos de mayor relevancia para el sistema SCADA.

La instrumentación es un área específica de la ingeniería de automatización y control de procesos la cual comprende la medición de cantidades físicas asociadas a los mismos.

La instrumentación de campo comprende sensores y actuadores. Un sensor es un dispositivo que detecta, o mide manifestaciones de cualidades o fenómenos físicos, como la energía, velocidad, aceleración, tamaño, cantidad, nivel, flujo, temperatura presión etc. Es un dispositivo que transforma la magnitud que se quiere medir, en otra, que facilita obtener una señal eléctrica en un rango manejable por el próximo elemento de la cadena de adquisición de la información. Por otra parte los actuadores son dispositivos que transforman una señal de entrada (eléctrica) en una acción física.

A continuación se presenta un Esquema de un sistema de Adquisición, supervisión y control de datos.

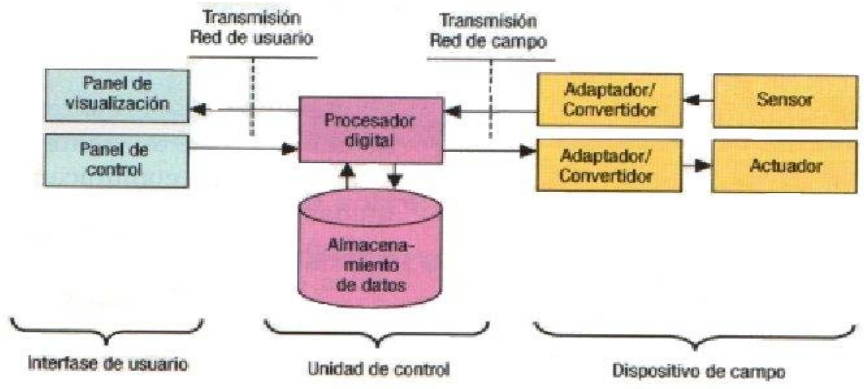


Figura 1. Esquema de un Sistema de Adquisición, Supervisión y Control de Datos.

1.2.3 Facilidades que brinda un SCADA

Los SCADAs ofrecen los servicios más demandados por los sectores industriales. Debido a la importancia que están tomando las comunicaciones en las soluciones ofrecidas por las aplicaciones de control y supervisión, se quiere garantizar mediante un buen diseño de interfaz gráfica la comunicación entre los diferentes sistemas y equipos físicos conectados, redes de área local, redes de procesos, redes o buses de campos. **(LUGONES 2005)**

Con este sistema se pueden evaluar procesos continuos o discretos, los que se presentan en toda la naturaleza, ofreciendo ventajas como:

- Reporte en tiempo real de las variables físicas del sistema.
- Control de Procesos.
- Almacenamiento de datos históricos.
- Planeamiento de la operación.
- Análisis de instrumentos.
- Eficiencia y seguridad de la operación.
- Confiabilidad en la medición.
- Análisis predictivo y de tiempo de supervivencia.
- Facilidad de mantenimiento.

1.3 Caracterización de sistemas SCADAs disponibles en el mercado y en Cuba.

Durante la investigación realizada se tuvieron en cuenta las características principales de algunos SCADAs utilizados tanto en Cuba, como el EROS, Movicon, WINCC y OASyS, este último utilizado por la empresa petrolera de Venezuela.

1.3.1 SCADA EROS (versión 3.0).

Eros es un sistema de supervisión y control de procesos industriales, en él se concentran múltiples facilidades para operar y dirigir cualquier proceso productivo. Desarrollado en 1991 por especialistas de la "Unión del níquel" de la provincia de Holguín.

Realiza variadas funciones dentro del entorno de la dirección de los procesos industriales.

Eros puede trabajar acoplado con diversos sistemas de colección de datos como elemento único o formando parte de una red industrial.

Eros se configura de forma fácil y tiene en cuenta todas las características de las variables medidas. Posee un potente tratamiento estadístico y determinístico.

Prestaciones del EROS:

- Esta soportado en ambiente Windows95 lo cual permite utilizar todas las facilidades del mismo.
- Se incorporan el mando a distancia y el control automático desde el sistema que son herramientas que potencian el automatismo del mismo.
- Se incorpora un nuevo manejador de la red que brinda mayor conectividad entre estaciones y un procesamiento más distribuido.
- La configuración de las variables del sistema se realizan de una manera más sencilla y accesible además permite configurar estaciones remotas.
- La frecuencia de medición de las variables es más versátil y amplia por lo que se puede medir con mayor precisión pudiendo alcanzar desde 136 mediciones por segundo (frecuencia del reloj de Windows 95) hasta 2 mediciones por minuto por lo que a cada variable se le puede adecuar con mayor especificad su tiempo, este tratamiento diferenciado garantiza que el tiempo del procesador se emplee de manera óptima, por supuesto éstos tiempos están condicionados a la velocidad de trabajo de los dispositivos de medición con que se cuente.
- Los mímicos son configurables por el usuario y usan la herramienta OLE. (OLE es una tecnología de integración de aplicaciones que pueden utilizarse para compartir información gráfica entre aplicaciones).
- Correo interno del sistema para enviar mensajes y/o documentos informáticos a los usuarios de la red que estén trabajando dentro del Eros. **(LUGONES 2005)**

1.3.2 SCADA Movicon

Movicon X representa la tercera generación innovadora de las plataformas industriales del software de supervisión y control (Scada/HMI), desarrollado por el grupo italiano Progea.

Ofrece la posibilidad de realizar potentes compactos sistemas de visualización HMI. El panel operador se convierte en una pequeña estación SCADA, ofreciendo independencia del hardware, conectividad con los sistemas superiores de información e incrementando la potencia de la máquina localmente.

Las ventajas de usar Movicon X son:

Sistema abierto: se puede integrar el mismo proyecto en diferentes terminales hardware. La ventaja es que el mismo software puede permanecer a pesar de que el panel de operador cambie, permitiendo escoger el producto que mejor se adapte a las necesidades.

Flexibilidad: puede integrar la información de la máquina con la planta o con un sistema de nivel superior de la fábrica. La ventaja es que se obtiene la libre circulación de la información, mediante el puerto Ethernet a las tecnologías OPC (OLE for Process Control).

Potente: Incrementa la calidad gráfica del HMI. Pudiéndose considerar un "pequeño SCADA", con el potencial integrado de una plataforma SCADA de alto nivel. Potentes gráficos vectoriales, manejo de las alarmas y registros con bases de datos relacionales que pueden integrarse en red, tendencias, recetas, scripts, envío de servicio de mensajes cortos o SMS (Short Message Service) y correos electrónicos.

Reducción de costes: Movicon X hace posible el uso de un solo software de supervisión (SCADA) tanto para PC como para paneles táctiles (HMI), con considerables ahorros en términos de aprendizaje, formación del personal y de mantenimiento.

Potente gestión de alarmas: gestión de alarmas con pantalla y registros totalmente configurables. Incluye un sistema de notificación de alarmas vía SMS y correo electrónico.

Registro de históricos, DataLoggers y recetas en Bases de Datos: Movicon X convierte automáticamente las conexiones ODBC del proyecto a conexiones ActiveX para Microsoft Windows CE (ADOCE)**,** en destino. Esto permite tener la posibilidad de registrar en bases de datos relacionales e integrar estas en los sistemas de información de la planta.

Control Remoto y Tele-servicio: Movicon X permite acceso remoto vía módem, para acceder remotamente a dispositivos así como para permitir acceso transparente al PLC conectado al dispositivo, para asegurar el mantenimiento remoto de la instalación.

Cliente Web Integrado: Movicon X incluye la función de cliente Web para permitir el acceso al equipo también vía Web por medio de un navegador gracias a la tecnología Web Services (o Servicios de la Web). **(PROGEA 2007)**

1.3.3 SCADA WINCC

SIMATIC WinCC es un sistema de supervisión sobre PC ejecutable bajo Microsoft Windows 95 y Windows NT, desarrollado por la empresa Siemens.

WinCC está concebido para la visualización y manejo de procesos, líneas de fabricación, máquinas e instalaciones. El volúmen de funciones de sistema incluye la emisión de avisos de eventos forma adecuada para la aplicación industrial, el archivo de valores medios, creación de recetas y el listado de los mismos. **(INDUSTRIALES 2005)**

Posee una interfaz de configuración compatible con el resto de software SIMATIC y accede a la base de datos compartida por todas las herramientas. El usuario dispone de una biblioteca de objetos gráficos y ventanas de mando estándar. **(MÁRQUEZ and BERMÚDEZ 2007)**

Se pueden realizar una multitud de posibles soluciones de automatización:

- Estructuras Servidor-Cliente con instalación sencilla.
- Seguridad en el servicio de proceso e integridad de los datos mediante redundancia.
- Ampliación de función sin límites por implementación de elementos ActiveX, lo que permite la interacción y personalización de los sitios Web.
- Posibilidades de comunicación abierta mediante OPC.
- Comunicación sencilla mediante drivers (código que implementa el protocolo de comunicaciones con un determinado equipo inteligente) implementados, o Programación en línea: no es necesaria detener el tiempo de ejecución del desarrollo para poder actualizar las modificaciones en la misma.

1.3.4 SCADA OASyS

OASyS es la sigla Open Architecture SyStem, (Sistema de Arquitectura Abierta) del sistema SCADA diseñado por Valmet para la adquisición de datos en tiempo real, control de dispositivos interactivos, anunciación y respuesta de alarmas, y reporte automatizado. Originalmente desarrollado para oleoductos

y gasoductos, OASyS se usa ahora también en las industrias eléctrica y de control y distribución de agua. **(SANGUINETTI)**

Incorpora tres subsistemas principales: un paquete de proceso y base de datos en tiempo real llamado Control & Measurement eXecutive (CMX), es español, Ejecutivo de Medición y Control; una base de datos relacional para datos históricos, llamado XIS (eXtended Information System) o Sistema de Información Extendida; y una interfaz de usuario gráfica, llamada XOS (eXtended Operator Station), o Estación de Operador Extendida, la cual presenta visualmente las condiciones del sistema y proporciona funciones de control de tipo operacional. La operación de la interfaz de usuario es el enfoque de este manual.

CMX colecciona y escala los datos, revisa las condiciones de alarma almacena la información de tiempo real, se comunica con unidades terminales remotas (RTUs), y le permite al usuario enviar comandos de control a los dispositivos de campo. XIS proporciona el espacio de almacenaje de datos históricos (típicamente datos de edad de más de cinco minutos) obtenidos desde el CMX, junto con funciones que le permiten al usuario crear reportes históricos y tendencias. XOS le permite a los usuarios interactuar recíprocamente con el sistema; incluye resúmenes de datos, mapas dinámicos, Pop-ups de control de dispositivo, y una interfase de comandos controlada a través del ratón.

Existen tres grupos principales de usuarios en OASyS: personal operacional, administrativo, y técnico.

El propósito de la interfase operacional OASyS es permitir al personal monitorear el funcionamiento de una instalación, tal como un gasoducto, y controlar elementos de esa instalación para lograr algún resultado deseado o resolver algún problema determinado.

1.4 Módulo de Gráficos Vectoriales.

En el Proyecto SCADA Nacional, se crearon líneas de trabajos e investigativas de acuerdo con los módulos y requerimientos necesarios para el desarrollo del SCADA entre ellas el Módulo de Gráficos Vectoriales, el cual tiene como objetivo principal: “Desarrollar una tecnología que permita la gestión de gráficos y animaciones vectoriales sobre Linux, necesarios para la visualización de los sinópticos durante su diseño y ejecución , inicialmente en PDVSA”. Para lo cual se esta desarrollando dos herramientas importantes del SCADA: el ambiente de edición y configuración, en el que se realiza el diseño de los despliegues que representan los procesos que se encuentran en el campo y el ambiente de ejecución, que es el encargado de visualizar los despliegues creados en el Editor, esta herramienta permite mostrar el proceso en tiempo real y controlarlo desde la misma.

Esta línea aportará al proyecto una interfaz de usuario que será la más cercana a la supervisión y control, y se convertirá en una vía palpable entre el usuario y el sistema.

1.5 Conceptos de la Ingeniería de Software.

Para lograr el desarrollo de los requisitos del subsistema de Gráficos Vectoriales se analizaron metodologías, flujos de trabajos y se asignaron roles que interactuarían con el sistema a desarrollar.

1.5.1 Análisis de Sistema.

Es la etapa donde se deberán utilizar las herramientas para la recolección de datos como Cuestionarios, Entrevistas, Revisión de Documentos, Observación (a sistemas de mayor, menor o igual complejidad), y representar la información recabada en diagramas de procesos como Yourdon, Hipo, Tablas, Árboles, etc.**(SANGUINETTI)**

Un Análisis de Sistema se lleva a cabo teniendo en cuenta los siguientes objetivos:

- **Identificación de Necesidades**

  Es el primer paso del análisis del sistema. En este proceso el Analista se reúne con el cliente y/o usuario (un representante institucional, departamental o cliente particular), e identifican las metas globales, se analizan las perspectivas del cliente, sus necesidades y requerimientos, sobre la planificación temporal y presupuestal, líneas de mercado y otros puntos que puedan ayudar a la identificación y desarrollo del proyecto.

  Este paso es llamado también por otros autores como "Análisis de Requisitos" y lo dividen en cinco partes:

  - Reconocimiento del problema.
  - Evaluación y Síntesis.
  - Modelado.
  - Especificación.
  - Revisión.

Antes de su reunión con el analista, el cliente prepara un documento conceptual del proyecto, aunque es recomendable que este se elabore durante la comunicación Cliente – Analista, ya que de hacerlo el cliente sólo de todas maneras tendría que ser modificado, durante la identificación de las necesidades.

- **Estudio de Viabilidad**

  La viabilidad y el análisis de riesgos están relacionados de muchas maneras, si el riesgo del proyecto es alto, la viabilidad de producir software de calidad se reduce, sin embargo se deben tomar en cuenta cuatro áreas principales de interés:

  - **Viabilidad económica**

    Es la evaluación de los costos de desarrollo, comparados con los ingresos netos o beneficios obtenidos del producto o Sistema desarrollado.

  - **Viabilidad Técnica**

    Es el estudio de funciones, rendimiento y restricciones que puedan afectar la realización de un sistema aceptable.

  - **Viabilidad Legal**

    Es determinar cualquier posibilidad de infracción, violación o responsabilidad legal en que se podría incurrir al desarrollar el Sistema.

  - **Alternativas**

    Una evaluación de los enfoques alternativos del desarrollo del producto o Sistema.

Otros objetivos:

- Identificar las necesidades del cliente.
- Evaluar los conceptos que tiene el cliente del sistema para establecer su viabilidad.
- Realizar un análisis técnico y económico.
- Asignar funciones al hardware, software, personal, base de datos y otros elementos del Sistema.
- Establecer restricciones de presupuestos y planificación temporal.

1.5.2 El Analista de Sistema.

A raíz del surgimiento y explotación del uso de las nuevas tecnologías informáticas, la necesidad de organizar aplicaciones, modificar procedimientos, crear nuevos métodos para dirigir los negocios y buscar y aplicar metodologías adecuadas para la creación y desarrollo del producto en tiempo de mercado, nace el Analista de Sistemas.

En este epígrafe se harán referencias y citas a conceptos dados por especialistas a cerca del rol "Analista de Sistema".

Jacobson, Booch y Rumbauch, definen al Analista de Sistema como:

"el responsable del conjunto de requisitos que están modelados en los casos de usos, lo que incluye todos los requisitos funcionales y no funcionales que son casos de uso específicos. El analista de sistemas es responsable de delimitar el sistema, encontrando los actores y los casos de uso y asegurando que el modelo de casos de uso es completo y consistente. Para la consistencia, el analista de sistemas puede utilizar un glosario para conseguir un acuerdo en los términos comunes, nociones, y conceptos durante la captura de los requisitos. "**(RUMBAUGH et al. 2000)**

Aunque el analista de sistemas es responsable del modelo de casos de uso y de los actores que contiene, no es responsable de cada caso de uso en particular. Esto es una responsabilidad aparte, que pertenece al trabajador Especificador de casos de uso. El analista de sistemas es también el que dirige el modelado y coordina la captura de requisitos.

Hay un analista de sistemas para cada sistema. No obstante, en la práctica, este trabajador está respaldado por un equipo (en talleres o eventos similares) que incluye otras personas que también trabajan como analistas.

Otra definición es la realizada por Ernesto Santos. (1980)

> "…el analista de problemas en computación deberá conocer procedimientos para indagar sobre lo existente y para saber proponer un verdadero sistema racionalizado, pero también deberá conocer sobre modernos sistemas de información, base del diseño, sobre todo en computación… Estos últimos factores son los que justifican tal especialidad, porque realmente debieron existir los analistas de sistemas, aunque no hubiera computadores, toda vez que siempre hubo sistemas para organizar, que posiblemente no se difundieron porque no existieron en importancia esos dos factores que hoy prevalecen: el computador y la información." **(TORRES)**

La definición de Analista de Sistemas de Senn (1992, p. 12), agrega:

"…Los analistas hacen mucho más que resolver problemas. Con frecuencia se solicita su ayuda para planificar la expansión de la organización…", es decir, el papel de los analistas sobrepasa los límites impuestos por la definición inicial, también cumplen el papel de asesores, ya sea en sistemas manuales o informatizados, o cualquier otro sistema donde la empresa tenga que invertir en información, después de todo esa es la razón de ser del analista." **(TORRES)**

1.5.3 Metodologías de desarrollo de software

Las metodologías para el desarrollo de software describen el conjunto de fases, etapas, actividades y tareas asociadas a la producción de software (aplicación) de calidad. La finalidad del uso de metodologías es lograr el desarrollo de un software de calidad.

Rumbaugh dio la siguiente definición:

> “Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo” (RUMBAUGH et al.

2000)

Existen diversas metodologías para el desarrollo de software donde cada una tiene ventajas y desventajas, y se utilizan en aquellas situaciones donde una metodología resulta más apropiada que otra. **(SCADA 2005)**

En el contexto de este proyecto se evaluaron las siguientes:

- **Modelo Espiral.**
- **Modelo Incremental.**
- **Metodología Orientada a objetos: Rational Unified Process (RUP).**

1.5.3.1 Modelo Espiral

Propuesto inicialmente por Boehm, es un modelo de proceso de software evolutivo, proporciona el potencial para el desarrollo rápido de versiones incrementales del software. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o en prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del diseño diseñado.

Este modelo utiliza la construcción de prototipos como mecanismo de reducción de riesgos, permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Mantiene el enfoque sistémico de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora al marco de trabajo iterativo que refleja de forma más realista el mundo real. **(PRESSMAN 2005)**

1.5.3.2 Modelo Incremental

Este modelo aplica secuencias lineales de forma escalonada mientras avanza el tiempo.

Se centra en la entrega de un producto operacional por cada incremento, por lo que los primeros incrementos son versiones incompletas del producto final; pero le proporciona al usuario la funcionalidad que precisa y una plataforma para la evaluación.

El desarrollo incremental es particularmente útil cuando la dotación del personal no está disponible para una implementación completa en la fecha límite establecida para el proyecto.**(PRESSMAN 2005)**

1.5.3.3 Metodología Orientada a Objetos: RUP

La metodología RUP, fue desarrollada en 1998 por Grady Booch, Ivar Jacobson y James Rumbaugh, prominentes metodologistas en la industria de la tecnología y sistemas de información. RUP se caracteriza por ser: dirigido por Casos de Uso, centrado en la arquitectura, iterativo e incremental.

Una metodología de desarrollo de software Orientada a Objeto (OO) consta de los siguientes elementos:

- Conceptos y diagramas (Modelo).
- Etapas y definición de entrega en cada una de ellas.
- Actividades y recomendaciones. (ISW 2006)

Las principales disciplinas de esta metodología son:

Disciplinas	Descripción
Modelado del negocio	Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización. Propone comprender los problemas de la organización e identificar posibles mejoras, evaluar el impacto del cambio introducido con la automatización, asegurar que todos los miembros tienen una misma visión de la organización.
Requerimientos	Esta disciplina se encarga de convertir las peticiones de los clientes en un conjunto de requerimientos de software, que definan el alcance y lo que debe hacer el producto a ser construido. Propone mantener un acuerdo a los interesados de lo que el sistema debe hacer, provee a los desarrolladores de una mejor visión de los requerimientos del sistema, define la frontera del sistema, crea las bases para la planificación y estimación.

Análisis y Diseño	Esta disciplina define como transformar artefactos resultantes del flujo de requerimientos de software en especificaciones del diseño del proyecto a desarrollar, en el se hace evolucionar la arquitectura de modo que se adapte al entorno del usuario final.
Implementación	Define como desarrollar, organizar realizar pruebas de unidad e integración de todos los componentes implementados basado en las especificaciones del sistema.
Pruebas	Este flujo se centra en encontrar y documentar los defectos en la calidad del software, validar y probar todos los supuestos hechos durante el diseño, verificar que el producto se desempeña como fue diseñado y cubre todos los requisitos pactados durante el flujo de "Captura de Requerimientos".
Instalación	Se encarga de garantizar que el producto esta disponible para lo usuarios en su ambiente, se realizan actividades como empaque, instalación, asistencia a usuarios.
Administración del proyecto	Se centra los aspectos esenciales del desarrollo iterativo como son la planificación, control de riesgos, seguimiento del proceso de desarrollo de software y aplicación de métricas. Las restantes serán utilizadas durante el proceso de administración.
Administración de configuración y cambios	Se encarga de controlar y sincronizar la evolución de todos los productos generados, introduce el uso de herramientas que faciliten el trabajo colaborativo en el equipo.
Ambiente	Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Tabla1. Principales etapas de RUP.

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo. **Ver Figura 2**.

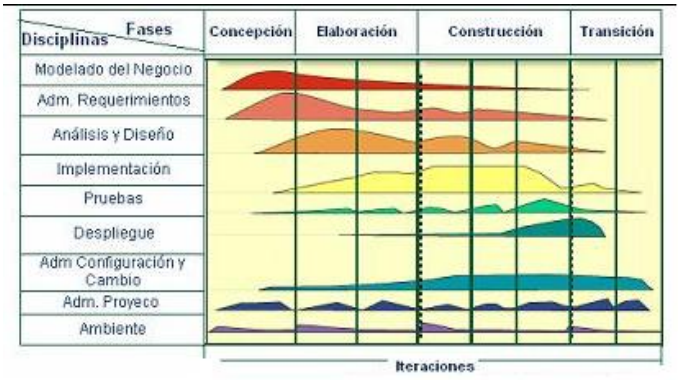


Figura 2. Metodología de desarrollo de software: RUP.

Cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.
- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene uno o varios entregables del producto que han pasado las pruebas. Se ponen estos entregables a consideración de un subconjunto de usuarios.
- **Transición:** El entregable ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Se decidió usar RUP como metodología de Desarrollo de Aplicaciones. El cliente sostuvo una conversación con IBM a fin de obtener la actualización de las versiones que existían en PDVSA debido a la obsolescencia de las mismas. Esta metodología posee herramientas que se ejecutan en Linux.

Las opciones de gestión de proyectos de desarrollo de software y de trabajo colaborativo de diferentes roles del desarrollo que ofrece la herramienta Rational no existe actualmente en herramientas de Software Libre. Además, herramientas basadas en RUP permiten la generación automática de la estructura del código para múltiples lenguajes de programación, y posee herramientas para el control de calidad y del proyecto.

1.6 Conclusiones

En este capítulo se ha realizado un análisis general del funcionamiento de un sistema SCADA y de su estado actual teniendo como premisa las características estudiadas de algunos de los SCADAs más reconocidos.

También se mostraron conceptos aplicables en la Ingeniería de Software que se llevó a cabo en el módulo de Gráficos, entre éstos se tuvo en cuenta algunas metodologías de desarrollo de software, concluyendo con la selección de RUP como la más adecuada dada las características del grupo de trabajo y la complejidad del sistema.

TENDENCIAS Y TECNOLOGIAS ACTUALES

CAPITULO

2

1.7 Introducción

En este capítulo, se hace un análisis del estado del arte de los métodos, técnicas y herramientas empleadas para la captura y especificación de requisitos así como de las tecnologías que se emplean en el desarrollo de los prototipos.

1.8 Sistema Operativo GNU/Linux.

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

1.8.1 Distribución de Linux: Debian GNU/Linux

El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo (SO) libre, bajo la licencia General Public Licence (LGPL). Basado en el conocido y distribuido núcleo de Linux la distribución es llamada Debian.

Una gran parte de las herramientas básicas que completan el sistema operativo, vienen del proyecto GNU; de ahí el nombre: GNU/Linux.

Debian es la única distribución importante de GNU/Linux mantenida solamente por voluntarios, es decir, sin un enfoque comercial, esto tiene ventajas y desventajas.

En primer lugar, las personas que se dedican a Debian tienen una alta motivación en participar en la misma, y se actualiza la distribución diariamente, apareciendo paquetes nuevos de software constantemente. Al mismo tiempo, existe un compromiso de calidad, no se desea distribuir software con errores.

En segundo lugar, dada su actitud abierta a la participación de todos, en el mismo espíritu original de Linux, constantemente hay personas que se unen a Debian para participar aportando su granito de arena, no solamente haciendo paquetes de programas, sino colaborando en el Servidor de Web, traduciendo

documentación de Debian, documentando fallos, o ayudando a los usuarios a través de las listas de correo que mantiene la comunidad.

Como desventajas tiene un mayor componente técnico que otras distribuciones. También, es posible que ciertos paquetes no estén tan actualizados como debieran, quizás porque sus desarrolladores han dejado de actualizarlos y nadie se ha hecho cargo. Sin embargo esto es algo que todos los desarrolladores tratan de evitar y, aunque cada desarrollador mantiene sus paquetes, no es raro que otro desarrollador (incluso un usuario) envíe una nueva versión del paquete para arreglar un problema o actualizarlo. **(DEBIAN 2007)**.

Se decidió usar la distribución Debian porque es una distro (distribución de GNU/Linux) de desarrollo muy estable, por lo que los paquetes que se desarrollen en él y quieran ejecutarse utilizando cualquier distribución siempre serán estables. A diferencia de otras distribuciones tiene un magnífico soporte de estabilidad en las aplicaciones (no requieren ser compiladas en la máquina que las esté usando). Los módulos del LDAP (Lighweight Directory Access Protocol) se pueden ejecutar sin problemas permitiendo que los usuarios usen sus sesiones en cualquier máquina dentro del área de trabajo, ahorrando recursos de hardware.

1.9 Ingeniería de Requisitos

La definición de las necesidades de un sistema es un proceso complejo, pues en él hay que identificar los requisitos que se deben cumplir para satisfacer las necesidades de los usuarios finales y de los clientes.

El proceso de especificación de requisitos se puede dividir en tres grandes actividades:

1- Captura de requisitos.

2- Definición de requisitos.

3- Validación de requisitos.

A continuación se explicarán brevemente algunas técnicas clásicas para realizar las actividades expuestas anteriormente.

1.9.1 Captura de requisitos

La captura de requisitos es la actividad mediante la cual el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema.

El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa.

Las técnicas siguientes, de forma clásica han sido utilizadas para esta actividad en el proceso de desarrollo de todo tipo de software.

- **Entrevistas:** resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización.

  Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista).

  A pesar de que las entrevistas son esenciales en el proceso de la captura requisitos y con su aplicación el equipo de desarrollo puede obtener una amplia visión del trabajo y las necesidades del usuario, es necesario destacar que no es una técnica sencilla de aplicar. Requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado. Aquí desempeña un papel fundamental la preparación de la entrevista.

- **JAD (Joint Application Development/ Desarrollo conjunto de aplicaciones):** esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación “lo que ve es lo que obtiene” (WYSIWYG, What You See Is What You Get) es decir, durante la aplicación de la técnica se trabajará sobre lo que se generará. Tras una fase de preparación del

JAD al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación.

Durante la sesión se discute en grupo sobre estos temas, llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema.

Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales, ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado, pero requiere un grupo de participantes bien integrados y organizados.

- **Brainstorming (Tormenta de ideas):** es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo diez personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador.

  Como técnica de captura de requisitos es sencilla de usar y de aplicar, contrariamente al JAD, puesto que no requiere tanto trabajo en grupo como éste. Además suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.

- **Concept Mapping (Mapas Conceptuales):** son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste. Sin embargo, deben ser usados con cautela porque en algunos casos pueden ser muy sugestivos y pueden llegar a ser ambiguos en casos complejos, si no se acompaña de una descripción textual.

- **Sketches y Storyboards (Esbozos y Guiones Gráficos):** Está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno Web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario (esbozos). Estos esbozos pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación (guiones gráficos).

- **Casos de Uso:** Aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores proponen casos de uso como técnica para la captura de requisitos. Los casos de

uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas, otros sistemas.) que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores.

La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades.

- **Cuestionarios y Checklists (Listas de Chequeo)**: Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (listas de chequeo). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.

- **Comparación de terminología:** Uno de los problemas que surge durante la licitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), una misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste). **(KOCH 2007).**

- **Ingeniería Inversa:** su objetivo es obtener información técnica a partir de un producto accesible al público, con el fin de determinar de qué está hecho, qué lo hace funcionar y cómo fue fabricado.

  Es denominada ingeniería inversa porque avanza en dirección opuesta a las tareas habituales de ingeniería, que consisten en utilizar datos técnicos para elaborar un producto determinado. En general si el producto u otro material que fue sometido a la ingeniería inversa fue obtenido en forma apropiada, entonces el proceso es legítimo y legal. **(WIKIPEDIA 2007)**

  Aplicar ingeniería inversa a algo supone profundizar en el estudio de su funcionamiento, hasta el punto de de llegar a entender, modificar, y mejorar dicho modo de funcionamiento.

  Áreas de la Ingeniería inversa:

  - Redocumentación: es la creación o revisión de una representación equivalente

semánticamente dentro del mismo nivel de abstracción relativo.

- Recuperación de diseño: es un subconjunto de la ingeniería inversa, en el cual, aparte de las observaciones del sistema, se añaden conocimientos sobre su dominio de aplicación, información externa, y procesos deductivos con el objeto de identificar abstracciones significativas a un mayor nivel.
- Rediseño: consiste en consolidar y modificar los modelos obtenidos, añadiendo nuevas funciones requeridas por los usuarios.
- Reingeniería de Software: es el examen y alteración de un sistema para reconstruirlo de una nueva forma y la subsiguiente implementación de esta nueva forma.**(ALARCOS)**

Para definir los requerimientos funcionales del módulo Gráfico, se usaron algunas técnicas de captura de requisitos favorables a las necesidades del equipo y que posibilitaran un dominio previo de los conceptos del negocio, permitiendo una familiarización con los mismos en un corto período de tiempo.

Una de las técnicas empleadas fue la tormenta de ideas con la participación de especialistas de empresas y universidades de nuestro país, experimentados tanto en el desarrollo como en la explotación de SCADAs. Esto permitió adquirir una visión inicial, común y general del nuevo sistema.

Se realizaron entrevistas a expertos y especialistas de PDVSA que tenían precedentes en la utilización de otros SCADAs. Éstas ayudaron a obtener información rápida y confiable de las necesidades del módulo Gráfico. En el **Anexo 1** se muestra una guía de entrevistas realizadas para la captura de requisitos.

Dada la existencia de un SCADA con reconocimiento y prestigio mundial y la posibilidad de que nuestros especialistas lo adquirieran se decidió emplear la técnica de ingeniería inversa al SCADA Movicon X. El análisis de esta aplicación que recoge todo el estado del arte existente en temas de SCADAs permitió la familiarización con los nuevos conceptos de este negocio, en particular, de la interfaz gráfica. Se obtuvo experiencia en aspectos de ergonomía de este tipo de aplicación, tanto en su ambiente de edición como de ejecución. Lo cual fue determinante para propiciarle al cliente una serie de requerimientos que satisficieran sus necesidades.

Como el dominio del sistema a desarrollar se enmarca en la automatización industrial, el equipo de analistas encargados de ejecutar las tareas relacionadas con el flujo de requerimientos, tuvo que hacer un estudio profundo del funcionamiento de sistemas similares destinados al control de procesos industriales, guiándose de la experiencia de los asesores en la especialidad y consultando documentación acreditada respecto al tema.

1.9.2 Definición de requisitos

Para la actividad de definición de requisitos en el proceso de ingeniería de requisitos existe un gran número de técnicas propuestas. Se describen a continuación las más relevantes:

- **Lenguaje natural**: Resulta una técnica muy ambigua para la definición de los requisitos. Consiste en definir los requisitos en lenguaje natural sin usar reglas para ello. A pesar de que son muchos los trabajos que critican su uso, es cierto que a nivel práctico se sigue utilizando.
- **Glosario y ontologías**: La diversidad de personas que forman parte de un proyecto de software hace que sea necesario establecer un marco de terminología común. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema.

  En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos.
- **Plantillas o patrones**: Esta técnica, recomendada por varios autores, tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta, menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado estructurado, el trabajo de rellenar las plantillas y mantenerlas, puede ser demasiado tedioso.
- **Escenarios:** La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia una representación gráfica en forma de diagramas de flujo. El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales del sistema.
- **Casos de uso**: Como técnica de definición de requisitos es como más ampliamente han sido aceptados los casos de uso. Actualmente se ha propuesto como técnica básica del proceso RUP. Sin embargo, son varios los autores que defienden que pueden resultar ambiguos a la hora de definir los requisitos, por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.
- **Lenguajes Formales:** Otro grupo de técnicas que merece la pena resaltar como extremo opuesto

> al lenguaje natural, es la utilización de lenguajes formales para describir los requisitos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la ingeniería de requisitos desde hace años. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la comunicación entre cliente y analista. Por el contrario, es la representación menos ambigua de los requisitos y la que más se presta a técnicas de verificación automatizadas. **(KOCH 2007)**

Mediante el análisis de las técnicas existentes para definir los requisitos, se decidió utilizar el glosario, las plantillas, los escenarios y casos de uso.

El Glosario, para conceptualizar las terminologías y lograr un entendimiento entre equipo de desarrollo y el cliente.

Las plantillas, para estructurar de una forma estándar la información, previendo que el nivel de detalles no fuera tan estructurado evitando así las ambigüedades. Las plantillas seleccionadas fueron las que propone la metodología RUP.

Los escenarios, para la modelación del sistema, utilizando los diagramas de actividad para una mejor comprensión de los casos de uso.

Los casos de uso, se tomaron por ser considerados una técnica básica del proceso de RUP utilizado en el desarrollo del sistema. Éstos fueron descritos en plantillas estándares de RUP.

1.9.3 Validación de requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias.

Aún así, existen algunas técnicas que pueden aplicarse para ello:

- **Reviews (Revisión):** Está técnica consiste en la lectura y corrección completa de la documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar consistencia de la documentación o información faltante.
- **Auditorías**: La revisión de la documentación con esta técnica consiste en un chequeo de los

resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.

- **Matrices de trazabilidad**: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.
- **Prototipos**: Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. **(KOCH 2007)**.

Para la validación de los requisitos se emplearon las técnicas: revisión y prototipos.

Posterior a la obtención de varios requisitos y del modelado de ellos se sostuvieron revisiones para cerciorarse que la interpretación por parte del analista fue correcta y que no faltó información.

Para obtener una idea de la interfaz de usuario se desarrolló un prototipo con parte de las funcionalidades críticas del sistema.

1.10 Herramienta Requisite Web

Requisite Web es un producto del Software “Rational” que tiene parentesco con la herramienta “Requisite Pro”.

Básicamente, Requisite Web es una aplicación de “IIS Active Server Pages(ASP)”, que es una tecnología de Microsoft para páginas Web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).

Esta herramienta es un anticipado de las bases de datos de requerimientos del proyecto Requisite Pro. Gran parte de su funcionamiento es el mismo que en el Requisite Pro, aunque particularmente es una versión nueva que permite que los usuarios agreguen requisitos así como la visión y que los corrijan.

Por las potencialidades que brinda esta herramienta, se decidió utilizarla en el módulo de Gráficos para la administración de los requerimientos, esto incluye las actividades de trazabilidad, control y seguimiento de de su elaboración.

En el **Anexo 2** se muestran imágenes del uso de la herramienta.

1.11 Lenguaje de modelado UML

Entre los lenguajes de modelado que define OMG (Object Management Group) el más conocido y usado es UML (Unified Modelling Language). UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. UML fue diseñado para ser un lenguaje de modelado de propósito general, por lo que puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación y plataformas de objetos.

Se estableció comparación entre UML 1.5 y UML 2.0; este último ofrece a los proveedores de herramientas CASE (Computer-Aided Software Engineering) "la producción automática de programas basado en especificaciones del software".

El UML 1.5 está constituido por 7 diagramas básicos y dos diagramas (**Ver Figura 3**) que constituyen variaciones de dos de los anteriores:

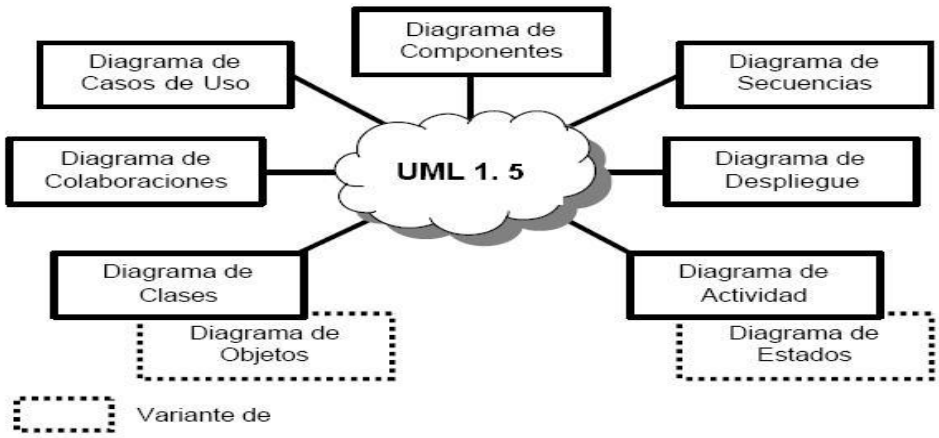


Figura 3. Diagramas de UML 1.5

En UML 2.0 se definen una serie de diagramas adicionales a los establecidos en UML 1.5. El conjunto de diagramas se encuentra organizado en torno a dos categorías: diagramas estructurales y diagramas dinámicos o de comportamiento. (**Ver Figura 4**)

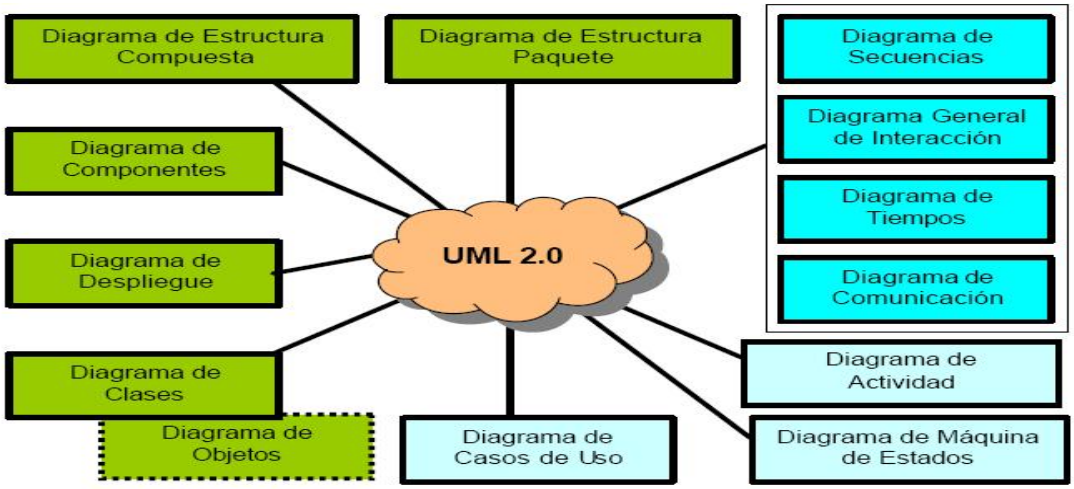


Figura 4. Diagramas de UML 2.0

UML 2.0 es la mayor revisión que se le ha hecho a UML desde la versión 1.0. El modelo conceptual en el 2.0 ha sido reestructurado completamente y nuevos diagramas han sido incorporados. Los diagramas tradicionales también han sido mejorados, por ejemplo, los diagramas de secuencia y de despliegue; en el primero es posible hacer referencias a otros diagramas de secuencia, además de incluir flujos alternos, opcionales, lazos, entre otros y en el segundo, sobre los diferentes nodos de la infraestructura de red se colocan, a modo de artefactos (elementos físicos simples), los elementos componentes del software.**(ANACHE and JOEL. 2005)**

En UML 2.0 los diagramas aparecen dentro de un marco (frame) que posee una etiqueta para indicar el tipo de diagrama, lo que permite al usuario una referencia rápida al diagrama invocado.

Esta versión proporciona a los analistas, arquitectos y desarrolladores; herramientas cada vez más potentes que les permite aprovechar mejor los modelos y como consecuencia generar una mayor cantidad de código reduciendo significativamente el ciclo de desarrollo de sus aplicaciones.

1.12 Plataformas de desarrollo de Software

Como parte de la necesidad de implantar una plataforma homogénea de diseño, modelado y desarrollo de software en función de la estrategia de la Gerencia de Automatización, Informática y Telecomunicaciones (AIT) y de la magnitud del desarrollo y criticidad del desarrollo del SCADA, se establecieron los siguientes requerimientos:

• Soporte de modelado UML v 2.0.

• Desarrollado bajo esquemas de Software Libre.

• Integración con la herramienta de desarrollo Eclipse 3.1.

• Generación automática de código en C++ a partir de los modelos UML.

• Generación automática de código para estructuras CORBA a partir de los modelos UML.

• Manejo de requerimientos y sus vínculos con los distintos modelos. (Trazabilidad entre el requerimiento y la implantación).

• Inclusión de código en el modelado.

• Facilidad de inclusión de parámetros (Entidad/Relación) de base de datos en el modelado.

• Manejo de repositorio de información y configuración centralizado.

• Capacidad de manejar perfiles de trabajo de acuerdo a los grupos de desarrollo.

• Manejo de la administración, planificación y gestión del desarrollo, así como la medición de los avances del proyecto.

• Capacidad de establecer métricas de desarrollo de software y auditoría del código.

• Pueda ser empleado en las siguientes fases del proyecto, e igualmente, que pueda ser empleado en futuros proyectos de desarrollo de software.

Se investigaron distintas aplicaciones y productos que cumplían los requerimientos mencionados, se evaluaron distintas tecnologías de licenciamiento libre (Software Libre) y propietario; sin embargo, las aplicaciones de este tipo, existentes en licenciamiento libre (Umbrello, ArgoUML y Dia), no cumple en su mayoría con los requerimientos mínimos necesarios, como es el manejo de requerimientos, trazabilidad de los requerimientos hasta la implantación, gestión del proyecto, perfiles de usuarios, métricas y auditoría de software, entre otros.

Además estas herramientas carecen de un entorno de desarrollo distribuido, es decir no permiten el modelado colaborativo.

Teniendo en cuenta la magnitud del proyecto y de la cantidad de personas que estarán trabajando de manera concurrente en el mismo, es de vital importancia que todos los desarrolladores estén sincronizados permanentemente y que los cambios que realicen en los modelos se vean reflejados de manera instantánea por los demás desarrolladores.

Por lo anteriormente expuesto se centró la selección en las herramientas propietarias nombradas a continuación:

- Suite de aplicaciones Rational
- Suite de aplicaciones Borland.

Como el cliente contaba con la licencia de la herramienta del Rational de IBM y las pruebas realizadas a la misma fueron satisfactorias, se determinó utilizarla como herramienta de modelado.

En dichas pruebas se determinaron ciertos elementos que deben permanecer siendo válidos a mediano plazo para garantizar la aplicabilidad de la herramienta en nuevos desarrollos de software en un futuro y que sean compatibles con la mayor cantidad de elementos en Software Libre, lo cual favorece a que los productos desarrollados puedan ser ejecutados en esta plataforma. Bajo esta premisa, la suite de herramientas de desarrollo de Rational presenta ventaja debido a que su desarrollo está basado en emplear el "framework" Eclipse, muy usado y conocido entre los desarrolladores en plataforma Linux.

Crea modelos UML, desde la toma de requerimientos, la definición de las clases y los diagramas de secuencia, hasta la generación del código base que permite iniciar el desarrollo de la aplicación. Rational ofrece además herramientas para la automatización de las pruebas de desempeño de la aplicación desarrollada, y someterla a pruebas severas que permitan descubrir fuga de recursos esenciales una vez que haya sido instalada.

Rational permite desplegar los cambios desde la toma de requerimientos hasta la implementación del código a través de repositorios de archivos. (EQUIPO 2005)

1.13 IDE a utilizar: Eclipse

Con el objetivo de realizar el prototipo de interfaz para el módulo de Gráficos Vectoriales se investigó sobre la herramienta Eclipse, versión 3.2.

El Eclipse es un Entorno Integrado de Desarrollo (IDE) multiplataforma libre para crear aplicaciones clientes de cualquier tipo. La primera y más importante aplicación que ha sido realizada con este entorno es el afamado IDE Java llamado Java Development Toolkit (JDT) y el compilador incluido en Eclipse, que se usaron para desarrollar el propio Eclipse.

Fue creado originalmente por International Business Machines Corporation (IBM), ahora lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El IDE de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java. Por ejemplo, existe un módulo para dar soporte a C/C++ +. Existen módulos para añadir un poco de todo, desde Telnet hasta soporte a bases de datos.

La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

Una de sus grandes ventajas es que basa su funcionamiento en plugins con lo que es ampliable para que haga prácticamente cualquier cosa, desde edición de XML a control del Tomcat (considerado un servidor de aplicaciones Web), pasando por plugins para otros lenguajes como Perl o Shell Script.

En cuanto a las aplicaciones clientes, provee al programador frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones Web, etc. Por ejemplo, GEF (Graphic Editing Framework para la edición gráfica) es un plugin de eclipse para el desarrollo de editores visuales que pueden ir desde procesadores de texto WYSIWYG hasta editores de diagramas UML, interfaces gráficas para el usuario (GUI). Dado que los editores realizados con GEF "viven" dentro de eclipse, además de poder ser usados conjuntamente con otros plugins, hacen uso de su interfaz gráfica personalizable y profesional. **(ECLIPSE 2007)**

1.14 Biblioteca Gráfica: GTK

Para el desarrollo de los prototipos de interfaces gráficas de usuario y de la posterior aplicación, se hace necesario utilizar alguna biblioteca que permita implementarlo, con este objetivo se seleccionó la biblioteca GTK+ (GIMP Toolkit), por una experiencia previa obtenida en otros proyectos por el equipo de desarrollo y por la premura de la entrega del sistema.
Se distribuye bajo la licencia LGPL, por lo que posibilita el desarrollo de software libre, e incluso software comercial no libre que use GTK sin necesidad de pagar licencias o derechos.

Ofrece todo lo necesario para el desarrollo de interfaces gráficas, pasando por los componentes más básicos (botones, cajas de texto, menús, ventanas.) hasta otros mucho más complejos y elaborados que serán de gran ayuda a la hora de programar aplicaciones gráficas.

GTK+ está a su vez separado en varias bibliotecas, algunas de ellas sólo disponibles para la versión 2.0:

- GDK implementa el nivel más bajo de la arquitectura, es decir, las primitivas gráficas. Es una biblioteca que forma una capa sobre la implementación gráfica real (X Window, MS Windows, Mac OS X), y es por tanto la única parte de GTK+ que tiene que ser reescrita para soportar otra plataforma/sistema operativo. Es por esta razón por la que GTK+ ya ha sido portada a varios entornos (X Window, MS Windows, QNX, BeOS).
- gdk-pixbuf es la biblioteca que permite el tratamiento de imágenes gráficas. Esta biblioteca permite el tratamiento (carga, visualización, grabación) de imágenes gráficas en distintos formatos (png, gif, jpeg, etc).
- Pango es un sistema que permite la representación de caracteres en distintos alfabetos (occidental, cirílico, árabe, chino, etc), y que supone uno de los pasos más importantes dentro del proyecto GNOME para la universalización del software libre.
- ATK es una biblioteca de clases abstractas cuyo objetivo es servir de base para el desarrollo de aplicaciones accesibles para personas con deficiencias físicas. Es un desarrollo de la empresa Sun, pues forma parte de su estrategia de inclusión de GNOME en entornos Solaris. **(MOYA 2004)**

1.15 Conclusiones

Una vez analizado las características más importantes de las tecnologías existentes y las tendencias actuales para el desarrollo de software se decide que el sistema que se propone se desarrolle haciendo uso de las herramientas de software libre, el cual dará la oportunidad de implementar el SCADA Nacional a bajos costos y con soberanía tecnológica, permitiendo desarrollar e investigar para lograr nuevos productos sin limitaciones.

La herramienta CASE seleccionada para el proceso de modelado acorde al lenguaje UML 2.0, es la suite Rational de IBM. Para el desarrollo de los prototipos de interfaz y la posterior aplicación se utilizó el IDE Eclipse, creando interfaces gráficas utilizando la biblioteca GTK. Entre las técnicas escogidas para la captura de requisitos se encuentran el uso de las entrevistas, tormenta de Ideas, ingeniería inversa y comparación de terminología. Para la definición de los requisitos se decidió utilizar el glosario, las plantillas, los escenarios y casos de uso, posibilitando su validación a través de las técnicas: revisión y prototipos.

DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

CAPITULO

3

1.16 Introducción

En este capítulo se realiza el proceso de modelación del sistema para la línea de Gráficos Vectoriales del SCADA. Para ello se identifican los actores y los casos de usos correspondientes para el funcionamiento y realización de la línea. Se describen los requerimientos funcionales y no funcionales del SCADA.

1.17 Modelación del sistema

En el presente epígrafe se exponen una selección del conjunto de requerimientos funcionales y no funcionales del módulo de manejo de gráficos del proyecto SCADA Nacional, para un análisis detallado se puede consultar el documento "Especificación de Requerimientos de Software de Interfaz de Configuración" **(PROYECTO 2007)** donde se representan todos los requerimientos y especificaciones de casos de uso del sistema.

En las especificaciones se incluyen las siguientes categorías: suplementarias, usabilidad, confiabilidad, desempeño, diseño, implementación, interfaces, infraestructura (físico); así como los requerimientos de documentación de usuario, requerimientos legales y de ambiente.

1.17.1 Actores del Sistema

Los actores representan personas o sistemas externos que interactúan con el sistema; actualmente para el SCADA se consideran cinco perfiles de usuarios:

Administrador: tendrá el dominio y control de todas las funcionalidades del sistema, las bases de datos y los usuarios.

Mantenedor: tendrá todas las facilidades funcionales del administrador exceptuando los permisos del control de usuarios.

Supervisor: sólo tendrá permiso de visualización del sistema.

Operador: se limitará a realizar operaciones de control y visualización en el sistema.

Auditor: sólo tendrán permiso de visualización del sistema.

A continuación se muestra una tabla que resume los permisos por perfiles definidos.

Perfil	**Sistema**		**Base de Datos**				**Usuarios**
	Visualizador	**Controlador**	**Modificar**	**Eliminar**	**Crear**	**Grupos**	**Administrar**
Administrador	**x**	**x**	**x**	**x**	**x**	Todos	Todos
Mantenedor	**x**	**x**	**x**	**x**	**x**	Todos	Su propio usuario
Supervisor	**x**					Configurable	
Operador	**x**	**x**				Configurable	
Auditor	**x**					Todos	

Tabla 2. Permisos por perfiles del Sistema SCADA.

Los perfiles describen las funcionalidades que los usuarios pueden realizar en el Sistema. Estos se especifican tomando en cuenta los permisos o niveles de acceso que tienen los Usuarios a los diferentes componentes que representan los procesos.

El Operador y Supervisor se asignan por área (la supervisión y control se limita por áreas; es decir, que los despliegues se asignarán por áreas. Los operadores y supervisores solo pueden acceder a los despliegues, gráficos y variables del área a las que se le da permiso en la configuración .Los demás actores tiene accesos a todas las áreas con sus particularidades, antes mencionadas.

El Operador es el encargado del control y supervisión de la planta mediante los sinópticos gráficos representados en la pantalla, donde se visualiza el proceso. Éstos sinópticos son almacenados en el computador consola del Operador y generados desde un editor gráfico que puede estar incorporado al SCADA, ya que este sistema no permite importar desde otra aplicación durante la configuración del sistema.

1.17.2 Requerimientos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Para el módulo de gráficos que se está desarrollando, ellos son:

1. Administrar proyectos. (PROYECTO 2007)

1.1 Agregar un nuevo proyecto.

1.1.1 Introducir el nombre y la descripción del proyecto.

1.1.2 Actualizar el árbol de dependencias del proyecto.

1.1.3 Actualizar el inspector de propiedades.

1.2 Abrir proyecto existente.

1.3 Guardar proyecto.

1.4 Eliminar proyecto.

2. Administrar nodos. (PROYECTO 2007)

2.1 Agregar un nuevo nodo.

2.1.1 Introducir el nombre, la descripción y el grupo de privilegios del nodo.

2.1.2 Actualizar el árbol de dependencias del proyecto.

2.1.3 Actualizar el inspector de propiedades.

2.2 Modificar nodo.

2.2.1 Modificar las propiedades del nodo.

2.2.2 Actualizar el árbol de dependencias del proyecto.

2.2.3 Actualizar el inspector de propiedades.

2.3 Importar nodo.

2.4 Exportar nodo.

2.5 Eliminar nodo.

3. Administrar puntos digitales. (PROYECTO 2007)

3.1 Agregar un nuevo punto digital.

3.1.1 Definir la información general de puntos como el nombre, la descripción, el grupo de privilegios al que está asociado y la frecuencia de muestreo.

3.1.2 Definir los parámetros de control como el número de bits de entrada, la dirección de entrada y su bit asociado.

3.1.3 Definir los parámetros de salida como el tiempo de espera, el número de bit de salida, la dirección de salida y su bit asociado.

3.1.4 Actualizar el árbol de dependencias del proyecto.

3.1.5 Actualizar el inspector de propiedades.

3.2 Modificar punto digital.

3.2.1 Modificar las propiedades del punto.

3.2.2 Actualizar el árbol de dependencias del proyecto.

3.2.3 Actualizar el inspector de propiedades.

3.3 Importar punto digital.

3.4 Exportar punto digital.

3.5 Eliminar punto digital.

4. Administrar puntos analógicos. (PROYECTO 2007)

4.1 Agregar un nuevo punto analógico.

4.1.1 Definir la información general de puntos como el nombre, la descripción, el grupo de privilegios al que está asociado y la frecuencia de muestreo.

4.1.2 Definir los parámetros de control como el tipo de dato de entrada, la dirección de entrada, el valor de la banda muerta y las unidades de ingeniería y las de campo con sus límites inferior y superior.

4.1.3 Definir los parámetros de salida como el tiempo de espera y la dirección de salida.

4.1.4 Actualizar el árbol de dependencias del proyecto.

4.1.5 Actualizar el inspector de propiedades.

4.2 Modificar punto analógico.

4.2.1 Modificar las propiedades del punto.

4.2.2 Actualizar el árbol de dependencias del proyecto.

4.2.3 Actualizar el inspector de propiedades.

4.3 Importar punto analógico.

4.4 Exportar punto analógico.

4.5 Eliminar punto analógico.

5. Administrar canales. (PROYECTO 2007)

5.1 Agregar un nuevo canal.

5.1.1 Introducir el nombre, la descripción, el grupo de privilegios al que está asociado el canal, definir sus parámetros físicos y el modo de acceso disponible.

5.1.2 Actualizar el árbol de dependencias del proyecto.

5.1.3 Actualizar el inspector de propiedades.

5.2 Modificar canal.

5.2.1 Modificar las propiedades del canal.

5.2.2 Actualizar el árbol de dependencias del proyecto.

5.2.3 Actualizar el inspector de propiedades.

5.3 Importar canal.

5.4 Exportar canal.

5.5 Eliminar canal.

6. Administrar subcanales. (PROYECTO 2007)

6.1 Agregar un nuevo subcanal.

6.1.1 Introducir el nombre, la descripción, el grupo de privilegios y el protocolo de comunicación asociado al subcanal.

6.1.2 Actualizar el árbol de dependencias del proyecto.

6.1.3 Actualizar el inspector de propiedades.

6.2 Modificar subcanal.

6.2.1 Modificar las propiedades del subcanal.

6.2.2 Actualizar el árbol de dependencias del proyecto.

6.2.3 Actualizar el inspector de propiedades.

6.3 Importar subcanal.

6.4 Exportar subcanal.

6.5 Eliminar subcanal.

7. Administrar dispositivo de control. (PROYECTO 2007)

7.1 Agregar un nuevo dispositivo.

7.1.1 Introducir el nombre, la descripción, el grupo de privilegios, la dirección del controlador y los parámetros de adquisición de puntos del dispositivo.

7.1.2 Actualizar el árbol de dependencias del proyecto.

7.1.3 Actualizar el inspector de propiedades.

7.2 Modificar dispositivo.

7.2.1 Modificar las propiedades del dispositivo.

7.2.2 Actualizar el árbol de dependencias del proyecto.

7.2.3 Actualizar el inspector de propiedades.

7.3 Importar dispositivo.

7.4 Exportar dispositivo.

7.5 Eliminar dispositivo.

Se realizó la captura de otros requisitos funcionales, pero no se presentarán en la presente investigación, puesto que aún no se han llevado a prototipo funcional.

1.17.3 Requerimientos no Funcionales

En este epígrafe se especifican los requerimientos no funcionales que se tuvieron en cuenta para el diseño y desarrollo de los prototipos.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe cumplir. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

- **Requerimiento de Rendimiento y Disponibilidad del Sistema**

Se debe garantizar restablecimiento general del sistema durante falla de procesos, servidores en cluster, en menos de tres segundos.

Se debe garantizar el restablecimiento del sistema en condiciones normales desde un estado desenergizado a un estado de completo funcionamiento en un período de cinco minutos.

Se debe garantizar el apagado del sistema en condiciones normales desde su estado de operación normal a un estado de totalmente apagado en un período de cinco minutos.

Se debe garantizar el tiempo de acceso a las diferentes ventanas y despliegues del sistema en menos de dos segundos.

Se debe garantizar que las modificaciones realizadas sobre la configuración del sistema (datos, despliegues, entre otros) sean registradas en el registro de eventos y reflejadas en todos los nodos y sistemas asociados en un tiempo de un segundo.

Se debe garantizar el soportar a una configuración de redundancia de redes, de forma tal que pueda restablecer conexión a alta velocidad en todos los nodos asociados (consolas, servidores en cluster, interfaces con otros sistemas).

- **Requerimiento de Soporte**

Se debe confeccionar un manual de usuario.
Se debe confeccionar un curso o guía de adiestramiento para cada perfil de usuario en el sistema.
Se debe ofrecer servicios de mantenimiento y actualización anualmente.

- **Requerimiento de Portabilidad**

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

- **Requerimiento de Confiabilidad**

Se debe garantizar que el módulo esté disponible las 24 horas del día.

Se debe garantizar que las paradas de mantenimiento no deben interferir en el correcto desempeño del resto del sistema.

Las actividades de mantenimiento, supervisión y edición del sistema no deben interferir en el correcto desempeño del resto del sistema, ni afectar la ejecución de la aplicación.

- **Requerimiento de Documentación de Usuario y Sistema de Ayuda en Línea**

Se debe garantizar que el módulo posea una ayuda dinámica en línea que se le brinde al usuario en dependencia de la acción u objeto seleccionado.

- **Requerimiento asociados al Licenciamiento**

Se debe garantizar que el sistema sea software libre y, por tanto, cualquier componente software que se utilice también deberá ser libre.

- **Requerimiento de Apariencia o interfaz externa**

Se debe garantizar que los colores de la interfaz de la aplicación sean claros.

La interfaz debe ser de fácil comprensión en su funcionamiento permitiendo la utilización del sistema sin mucho entrenamiento

- **Requerimiento de Interfaces de Usuario**

La aplicación debe proveer al usuario una gran versatilidad en la presentación de la información en las pantallas, en cuanto a disponibilidad de los recursos y organización de los mismos.

Debe permitir el manejo de ambientes multi-ventanas o despliegues, visualizándose varios despliegues simultáneamente, máximo diez (configurable por el administrador), sin generar efectos negativos en el rendimiento del sistema.

- Posibilidad de ocultar y mostrar las ventanas de configuración del proyecto en un área determinada; o sea, que aunque se oculten éstas se mantengan activas. También debe ofrecerse la posibilidad de cerrarlas.
- Mostrar el nombre de la aplicación en la parte superior izquierda mediante una barra de título, el nombre del despliegue sobre el cual se trabaja y el camino de la carpeta que contiene el proyecto.
- La aplicación debe estar sectorizada en varias áreas funcionales:

- Para el Menú y las Barras de Herramientas, utilizar la parte superior.
- Para el editor de script y mensajes del sistema y errores utilizar la parte inferior.
- Para el panel de controles y dibujos, biblioteca de plantillas de símbolos gráficos, ayuda dinámica, propiedades de cada recurso y componentes un proyecto, utilizar el lateral derecho.
- Para el panel de dependencia de los recursos de proyectos, utilizar el lateral izquierdo.
- Para realizar el diseño de los despliegues, utilizar la parte central.

- Permitir mostrar opciones estándares (copiar, cortar, pegar…), pudiendo acceder mediante el clic derecho sobre un componente o por la combinación de teclas calientes.
- Permitir mostrar opciones funcionales (agregar, modificar, eliminar…), pudiendo acceder mediante el clic derecho sobre un componente o por la combinación de teclas calientes.
- Permitir tener un menú que posea una serie de opciones que garanticen realizar acciones para el manejo y control de los objetos gráficos.
  - Mostrar Información en los laterales de la ventana principal.
  - Mostrar una ventana que contenga todos los recursos del sistema localizados en forma de árbol, “explorador de la aplicación”.
  - Mostrar ventana de propiedades de cada recurso y componente.

- El explorador de aplicaciones debe mostrar los siguientes recursos del proyecto.
  - Proyecto, elemento que contiene todo lo definido por un proyecto SCADA específico.
    - Nodos físicos contienen todos los elementos configurables del proyecto, pueden contener además la base de datos de configuración o histórica, la base de datos de tiempo real, variables, despliegues, canales, etc.

1.18 Descripción de los casos de uso

El objetivo principal de detallar cada caso de uso es describir su flujo de sucesos en detalle, incluyendo cómo comienza, termina e interactúan con los actores **(RUMBAUGH et al. 2000).**

Los casos de usos del módulo de gráfico se encuentran descritos en los documentos “Casos de Usos de Gráficos Vectoriales”. **(PROYECTO 2007)**

A continuación se muestra la descripción del caso de uso “Administrar Proyecto”.

Desarrollo SCADA Nacional

Caso de Uso Administrar Proyecto

Versión 1.0

Histórico de revisión

Fecha	Versión	Descripción	Autor
11/04/2007	1.0	Creación del documento	Rene López Banacaldo, Yosell Sehara Driggs, Irina Argota, Iliana Pérez Pupo.

Caso de Uso Administrar Proyecto

1. Nombre: Administrar proyecto

2. Breve Descripción

CUS2 Este caso de uso le permite al Mantenedor crear un proyecto y definir sus parámetros de configuración a través de una interfaz gráfica.

3. Actores

3.1 Mantenedor “activa”

4. Flujo básico.

4.1 Agregar proyecto

El caso de uso comienza cuando el Mantenedor desde el ambiente de edición selecciona “Agregar proyecto”. El Sistema carga y muestra el espacio de trabajo al Mantenedor donde se realizará la construcción y configuración de toda la información que debe tener el proyecto.

4.2 Asignar nombre al proyecto

El Sistema solicita al Mantenedor el nombre del proyecto. El Mantenedor le asigna al proyecto el nombre con el cual será identificado.

4.3 Escribir la descripción del proyecto

El Sistema solicita al Mantenedor ingresar la descripción del proyecto. El Mantenedor suministra la descripción del proyecto. La longitud máxima de caracteres de esta descripción será definida de acuerdo a los estándares establecidos en el documento de especificaciones técnicas del SCADA Nacional.

4.4 Guardar los cambios

El Mantenedor decide guardar la configuración realizada sobre el proyecto. El Sistema almacena la configuración realizada sobre el proyecto y registra la acción en el registro de eventos.

5, Flujos alternos

5.1 Abrir proyecto

En el paso 4.1 del flujo básico el Mantenedor desde el ambiente de edición selecciona “Abrir proyecto”. El Sistema muestra la lista de proyectos configurados. El Mantenedor selecciona el proyecto que desea abrir. El Sistema muestra al Mantenedor el detalle de los parámetros configurados para el proyecto y los elementos que contiene en el árbol de dependencia.

5.2 Modificar proyecto

En el paso 4.1 del flujo básico el Mantenedor desde el ambiente de edición selecciona “Modificar proyecto”. El Sistema muestra la lista de proyectos disponibles. El Mantenedor selecciona el proyecto que desea modificar. El Sistema muestra al Mantenedor los parámetros del proyecto configurados y retorna al paso 4.2 del flujo básico.

5.3 Eliminar proyecto sin nodos asociados

En el paso 4.1 del flujo básico el Mantenedor desde el ambiente de edición selecciona “Eliminar proyecto”. El Sistema muestra la lista de proyectos. El Mantenedor selecciona el proyecto que desea eliminar. El Sistema verifica que el proyecto no tiene nodos asociados. Se solicita confirmación al Mantenedor para realizar la eliminación. El Mantenedor acepta la acción de eliminar y se registra en el registro de eventos la identificación del Mantenedor, la fecha y las acciones realizadas al eliminar el proyecto y el caso de uso termina.

5.4 Error al intentar eliminar al proyecto

En el paso 5.3 del flujo alterno el Sistema le muestra al Mantenedor un mensaje de error al intentar eliminar un proyecto que tiene al menos un nodo asociado. El Sistema le indica al Mantenedor que debe eliminar el (los) nodo (s) asociado (s) al proyecto y el caso de uso termina.

5.5 Descargar los cambios de la configuración al servidor

En el paso 4.5 del flujo básico el Mantenedor decide descargar la configuración del proyecto al servidor. El Sistema ofrece al Mantenedor las opciones para realizar la descarga al servidor (Ver Notas 10.1 las “*Opciones de configuración para la descarga de proyecto al servidor*”). El Sistema por omisión presenta la

opción de descargar la configuración del proyecto al servidor. El Mantenedor acepta la opción. El Sistema registra la fecha y la acción en el registro de eventos y el caso de uso termina.

5.6 Descargar al servidor proyectos seleccionados

En el paso 4.5 del flujo básico el Mantenedor decide descargar al servidor proyectos seleccionados. El Sistema muestra la lista de proyectos creados. El Mantenedor selecciona qué proyectos va descargar al servidor. El Sistema muestra un mensaje de confirmación. El Mantenedor acepta el mensaje y el caso de uso termina.

5.7 Importar proyecto duplicado

En el paso 4.1 del flujo básico el Mantenedor desde el ambiente de edición selecciona “Importar proyecto”. El Sistema muestra al Mantenedor un explorador donde ubicar los archivos bajo control de versiones. El Mantenedor selecciona un proyecto y el Sistema verifica la duplicidad con respecto a los existentes. El Sistema registra la existencia de duplicidad entre el proyecto a importar y los existentes. El Sistema muestra un mensaje de error al Mantenedor y la posibilidad de cancelar la acción o de cambiar el nombre del proyecto a importar. Si el Mantenedor cancela, se termina el caso de uso. Si no, el Mantenedor cambia el nombre duplicado y el Sistema carga el proyecto en el ambiente de edición y el caso de uso retorna al paso 4.2 del flujo básico.

5.8 Importar proyecto sin duplicado

En el paso 5.7 el Mantenedor selecciona un proyecto y el Sistema verifica que no existe duplicidad, el Sistema carga el proyecto en el ambiente de edición y el caso de uso retorna al paso 4.2 del flujo básico.

5.9 Exportar proyecto

En el paso 4.1 del flujo básico el Mantenedor desde el ambiente de edición selecciona “Exportar proyecto”. El Sistema le muestra una lista al Mantenedor de los proyectos existentes y el Mantenedor selecciona el que va a exportar. El Sistema le pregunta al Mantenedor el nombre y directorio donde almacenar el proyecto. El proyecto es almacenado en el directorio indicado con toda la configuración realizada y el caso de uso termina.

6. Requerimientos especiales

7. Precondiciones

7.1 El Mantenedor está registrado en el ambiente de edición.

7.2 El Sistema debe validar que el Mantenedor esté autorizado para agregar y realizar el resto de las funcionalidades sobre el proyecto.

8. Puntos de extensión

9. Puntos de inclusión

10. Notas

10.1 Opciones de configuración para la descarga de proyectos al servidor

Las opciones que ofrece el Sistema al Mantenedor al momento de descargar un proyecto al servidor son las siguientes:

No descargar el proyecto al servidor

Descargar al servidor los proyectos configurados y seleccionados

Descargar al servidor solamente el proyecto configurado

1.19 Diagramas de caso de uso

Luego de haber definido los actores y requerimientos funcionales del sistema, se modela la relación que existe entre el actor Mantenedor con los casos de usos críticos definidos a partir de la captura de los requerimientos.

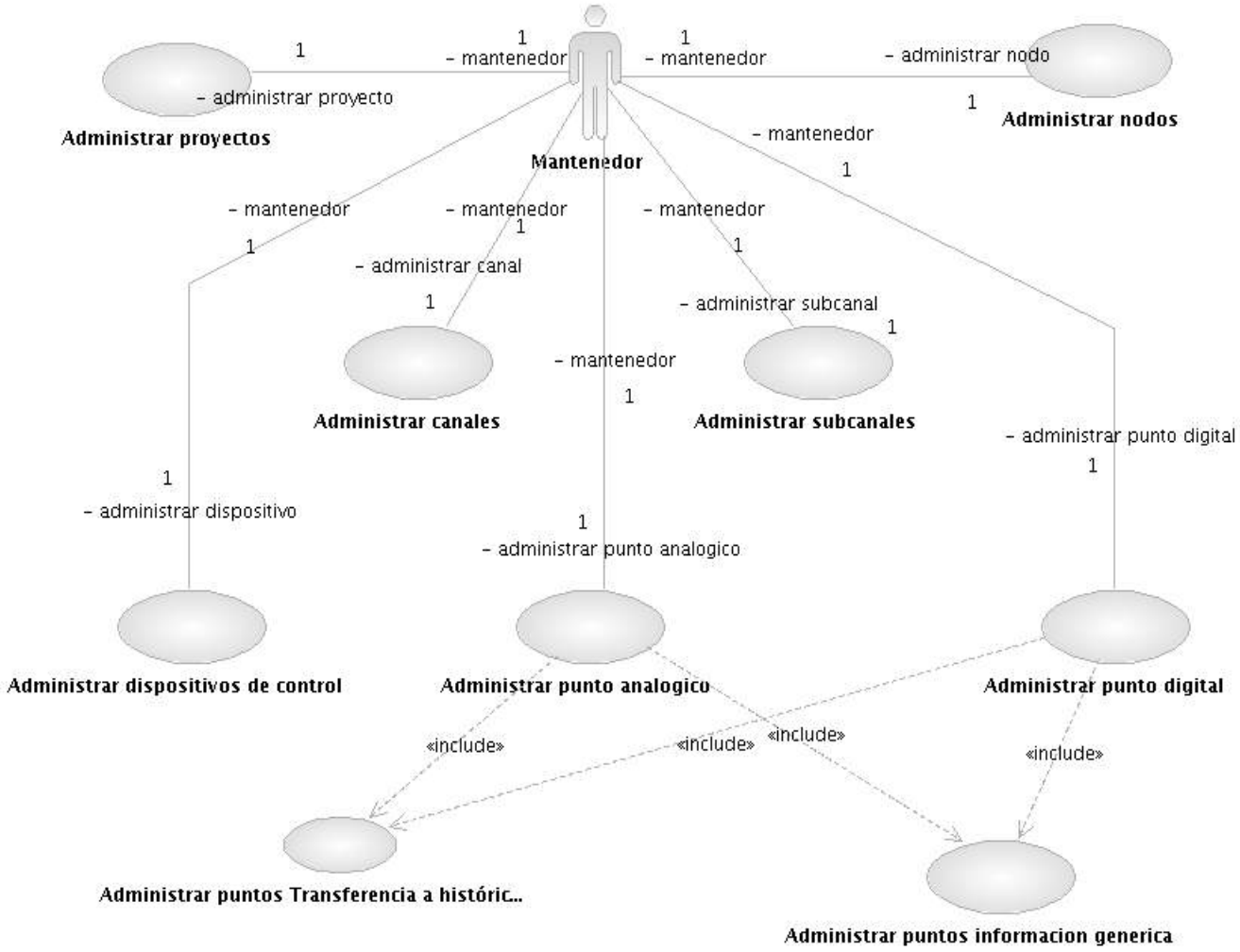


Figura 5. Diagrama de casos de uso más significativos

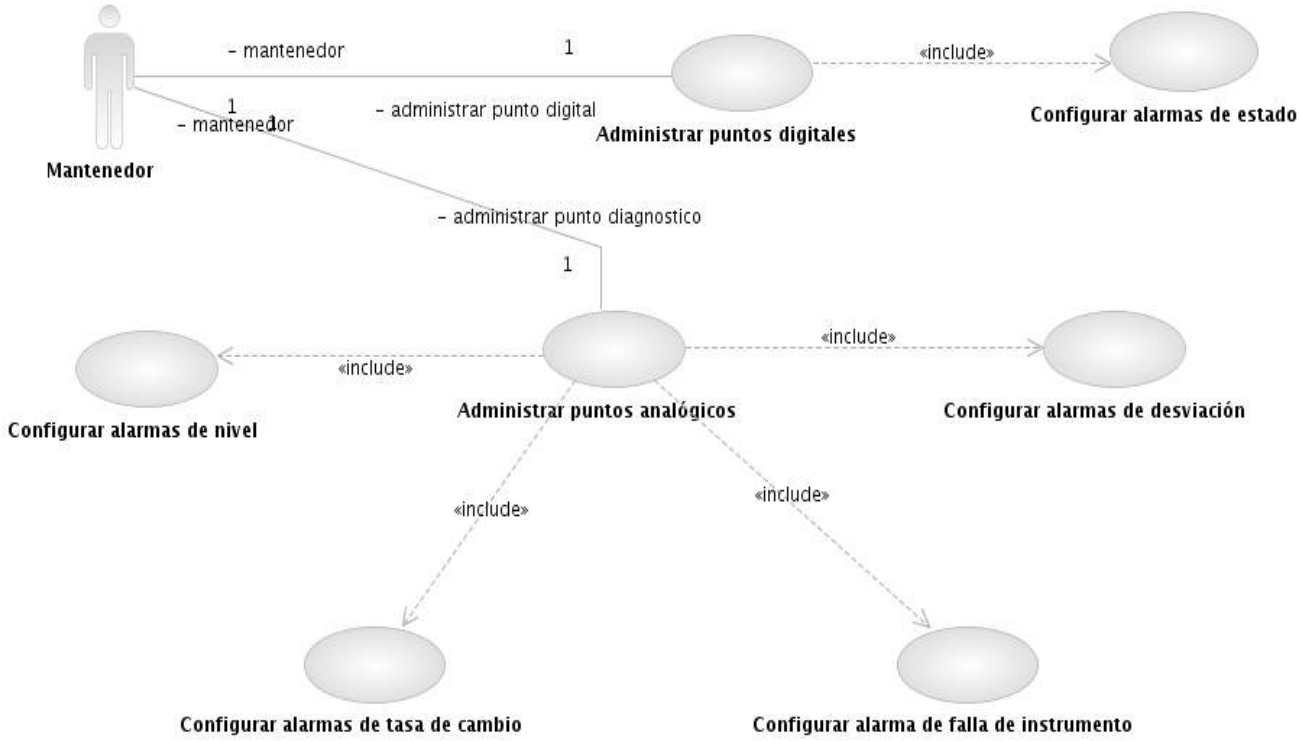


Figura 6. Diagrama de los casos de uso "Administrar puntos analógicos" y "Administrar puntos digitales" con sus casos de uso de inclusión referentes a las alarmas

1.20 Diagramas de actividad

El Diagrama de Actividad es un diagrama de flujo del proceso multi-propósito que se utiliza para modelar el comportamiento del sistema. Los diagramas de actividad se pueden usar para modelar un caso de uso, una clase, o un método complicado. A continuación se muestran algunos diagramas de actividad para determinados casos de usos del módulo de Gráficos.

1.20.1 Diagramas de actividad del caso de uso "Administrar Proyecto" (PROYECTO 2007)

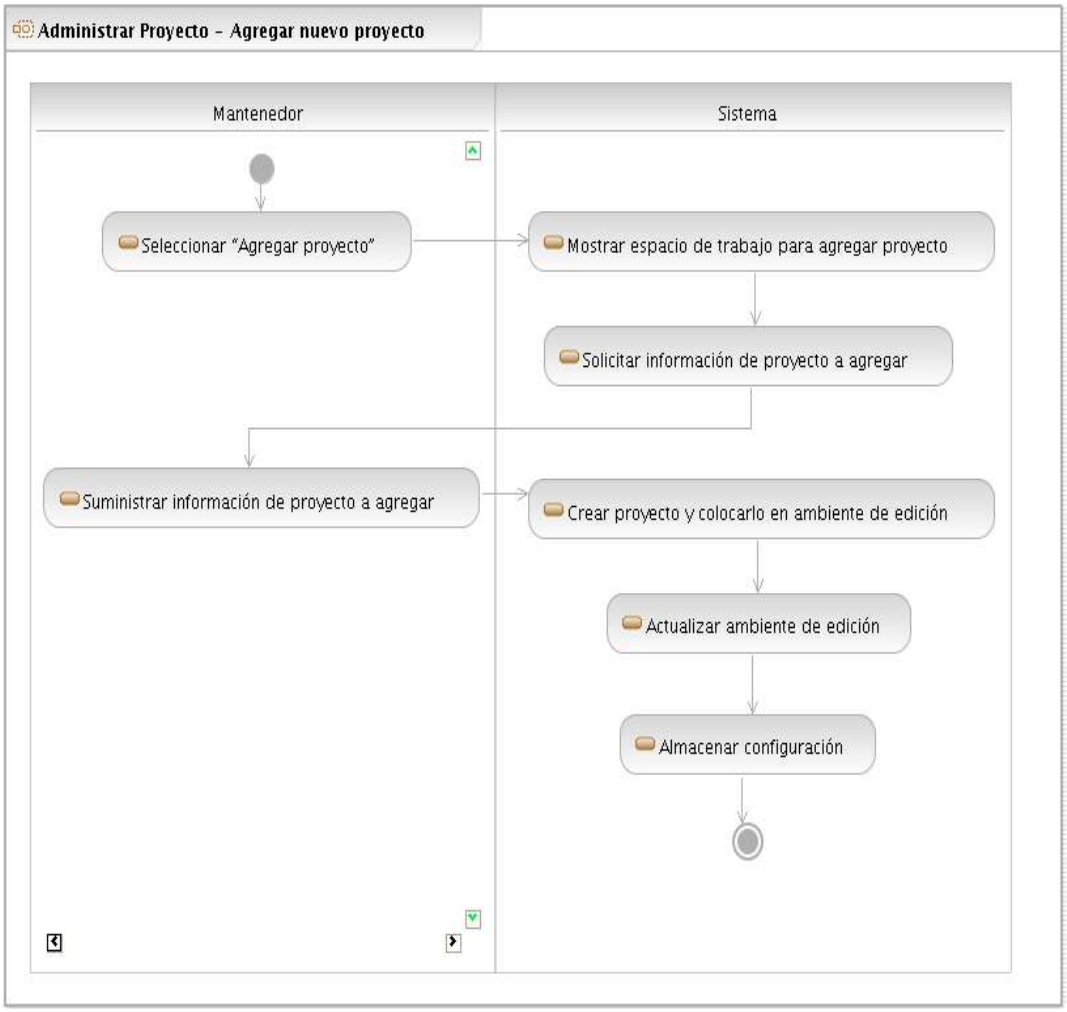


Figura 7. Diagrama de actividades del caso de uso Administrar Proyecto - Agregar nuevo proyecto

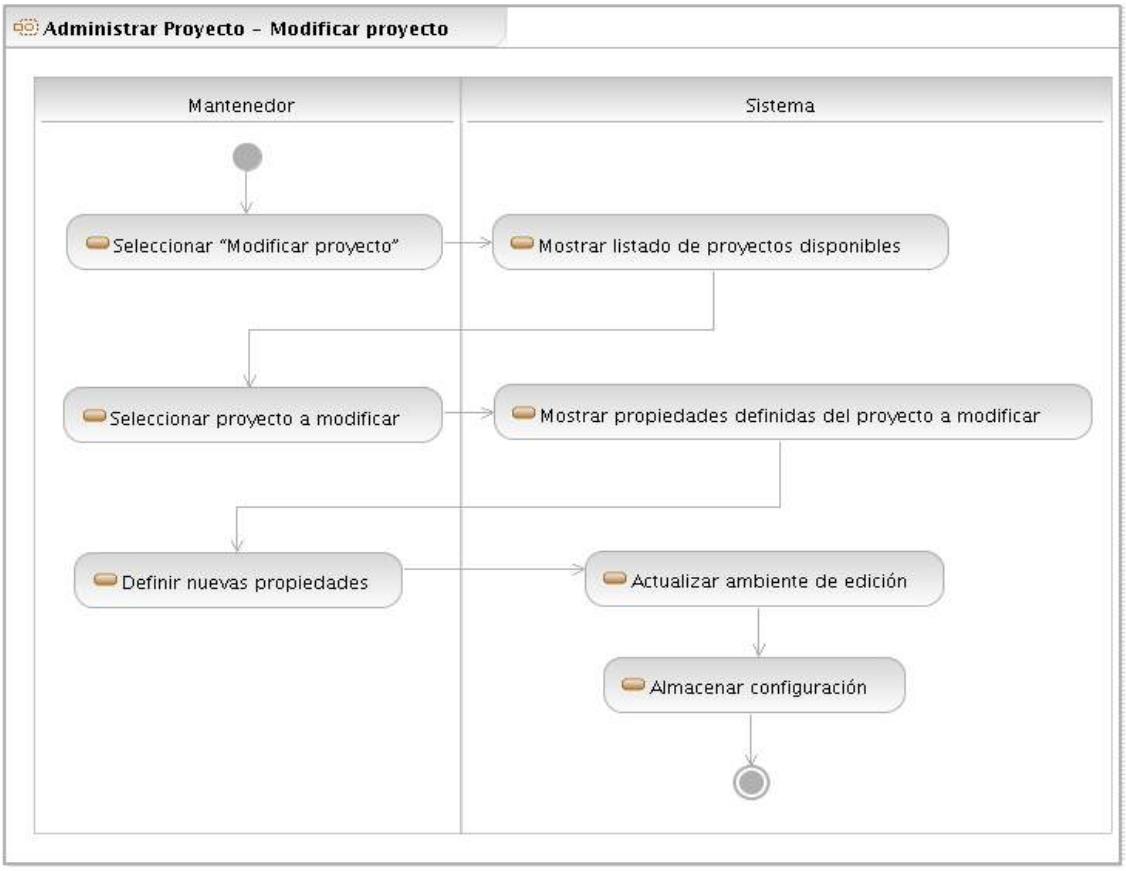


Figura 8. Diagrama de actividades del caso de uso Administrar Proyecto - Modificar proyecto

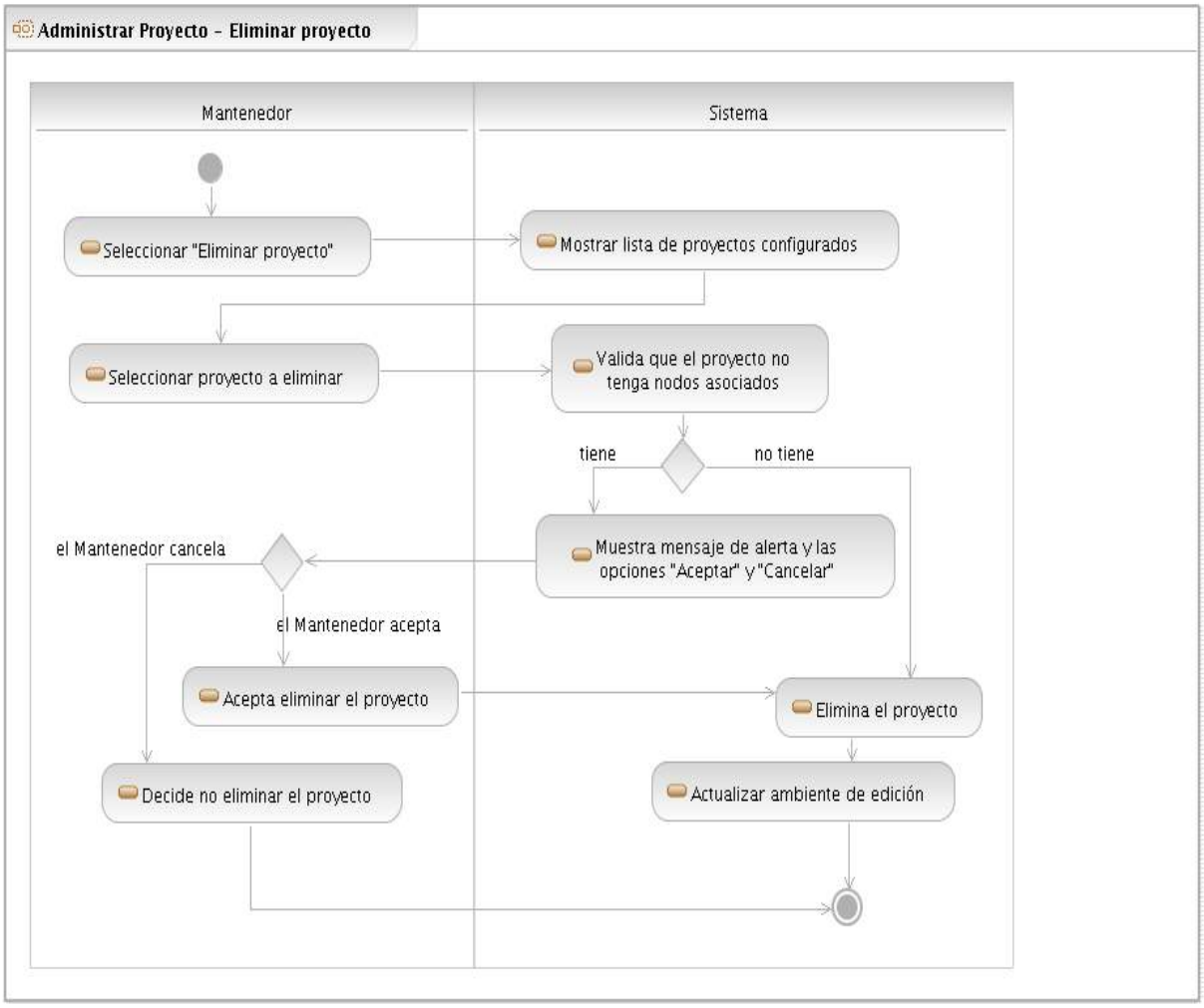


Figura 9. Diagrama de actividades del caso de uso Administrar Proyecto – Eliminar proyecto

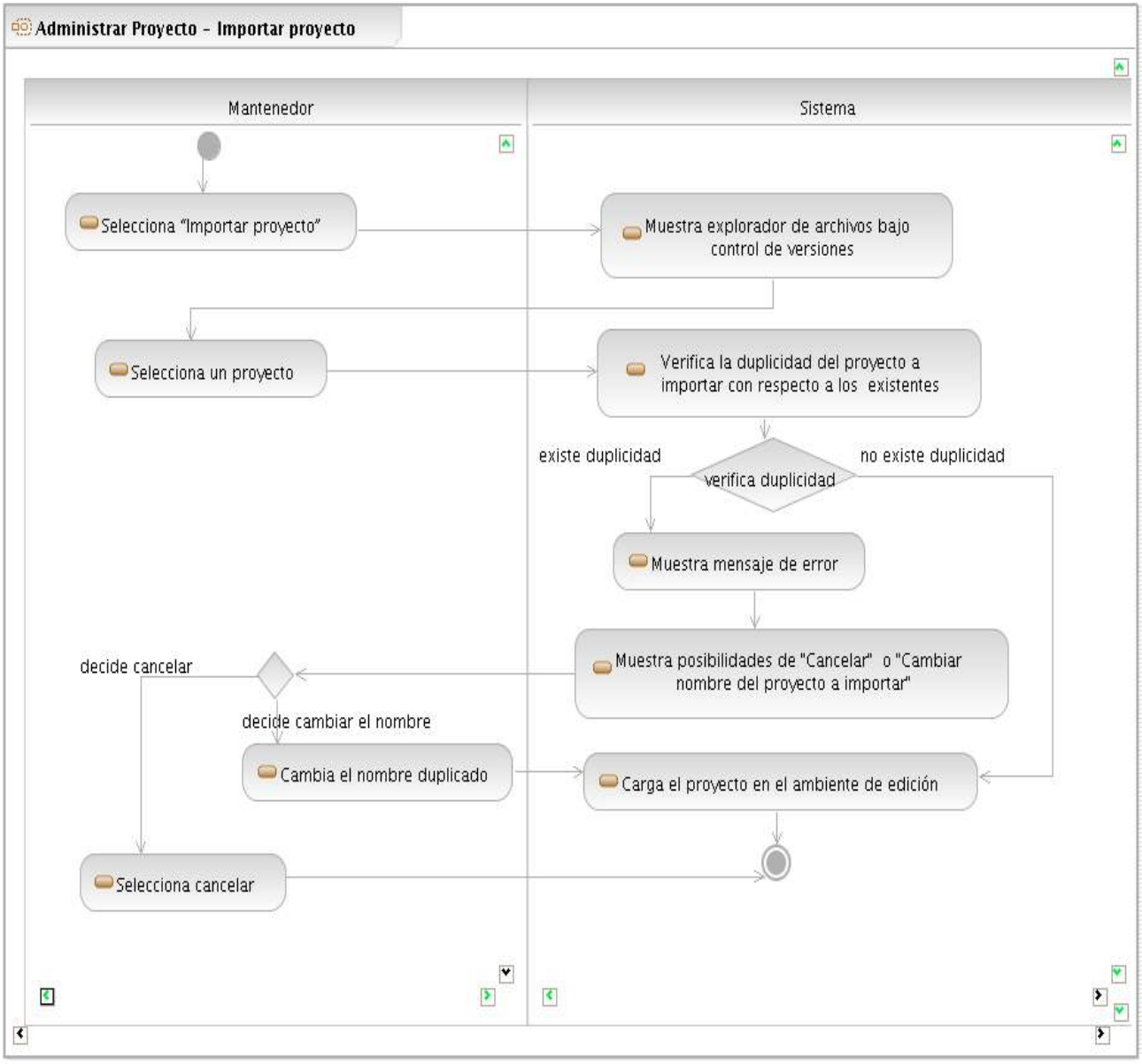


Figura 10. Diagrama de actividades del caso de uso Administrar Proyecto – Importar proyecto

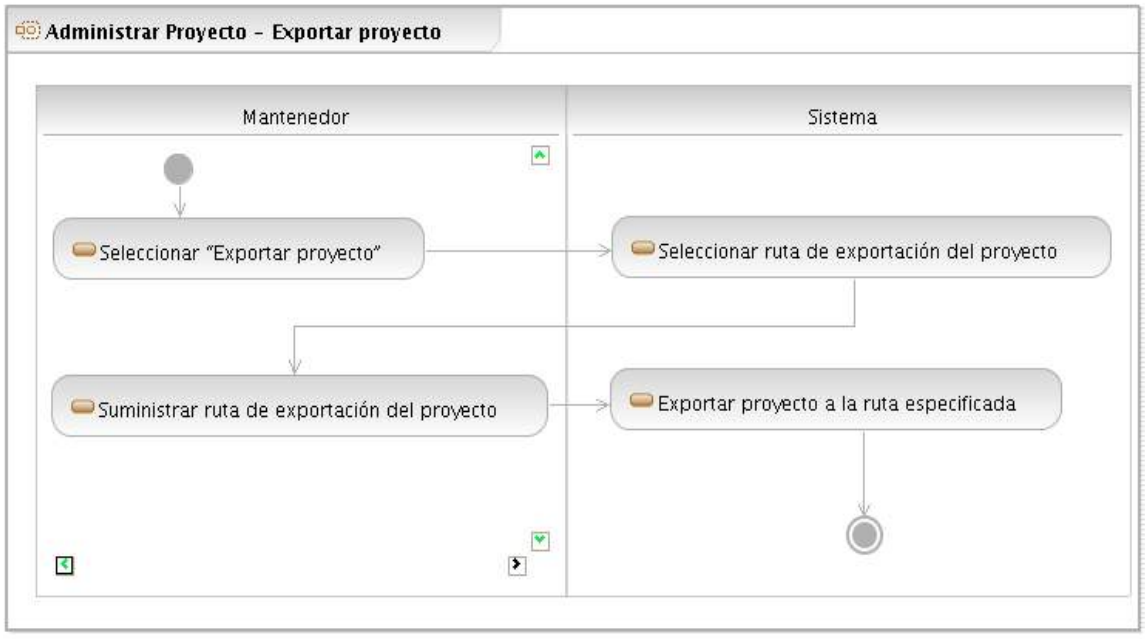


Figura 11. Diagrama de actividades del caso de uso Administrar Proyecto – Exportar proyecto

1.20.2 Diagramas de actividad del caso de uso "Administrar Nodos" (PROYECTO 2007)

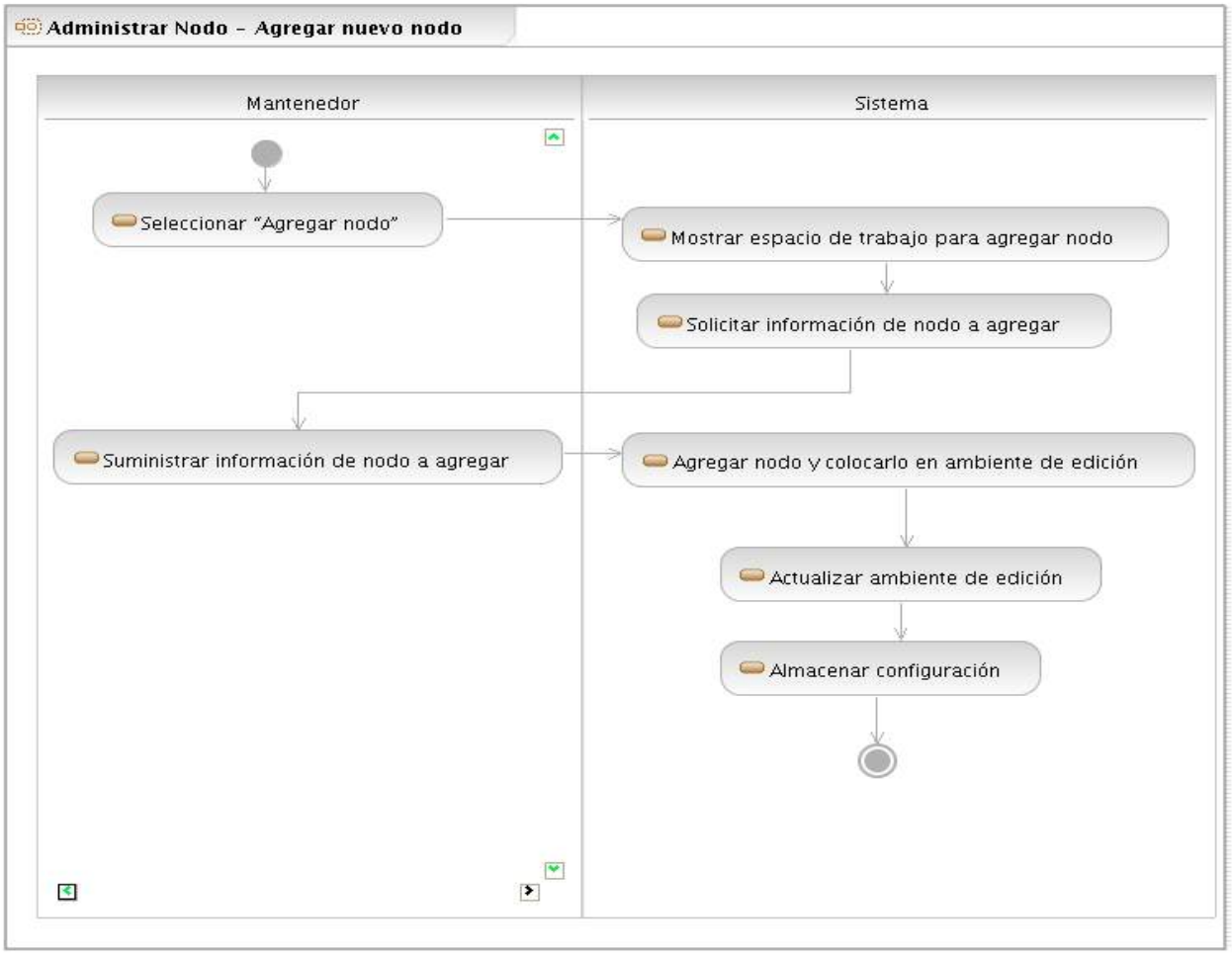


Figura 12. Diagrama de actividades del caso de uso Administrar Nodo – Agregar nodo

uesta

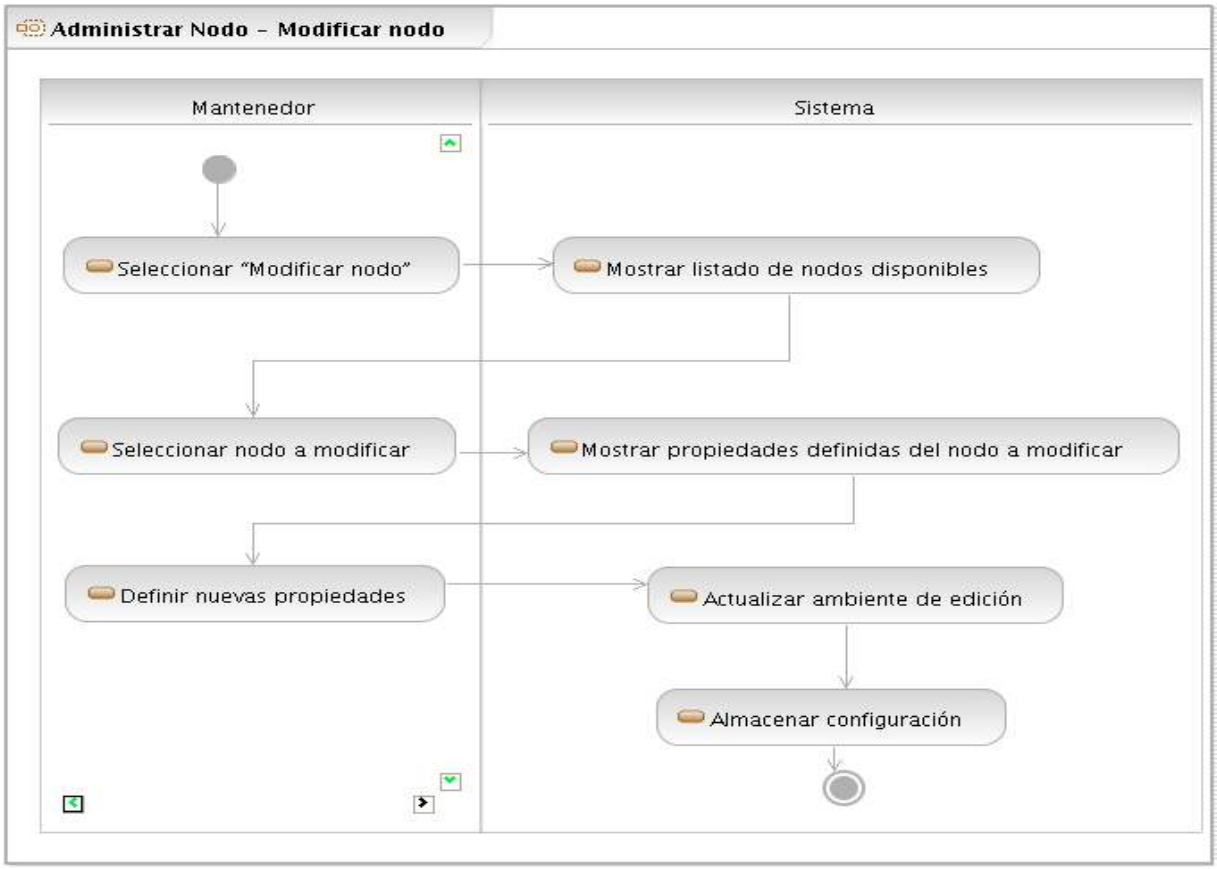


Figura 13. Diagrama de actividades del caso de uso Administrar Nodo – Modificar nodo

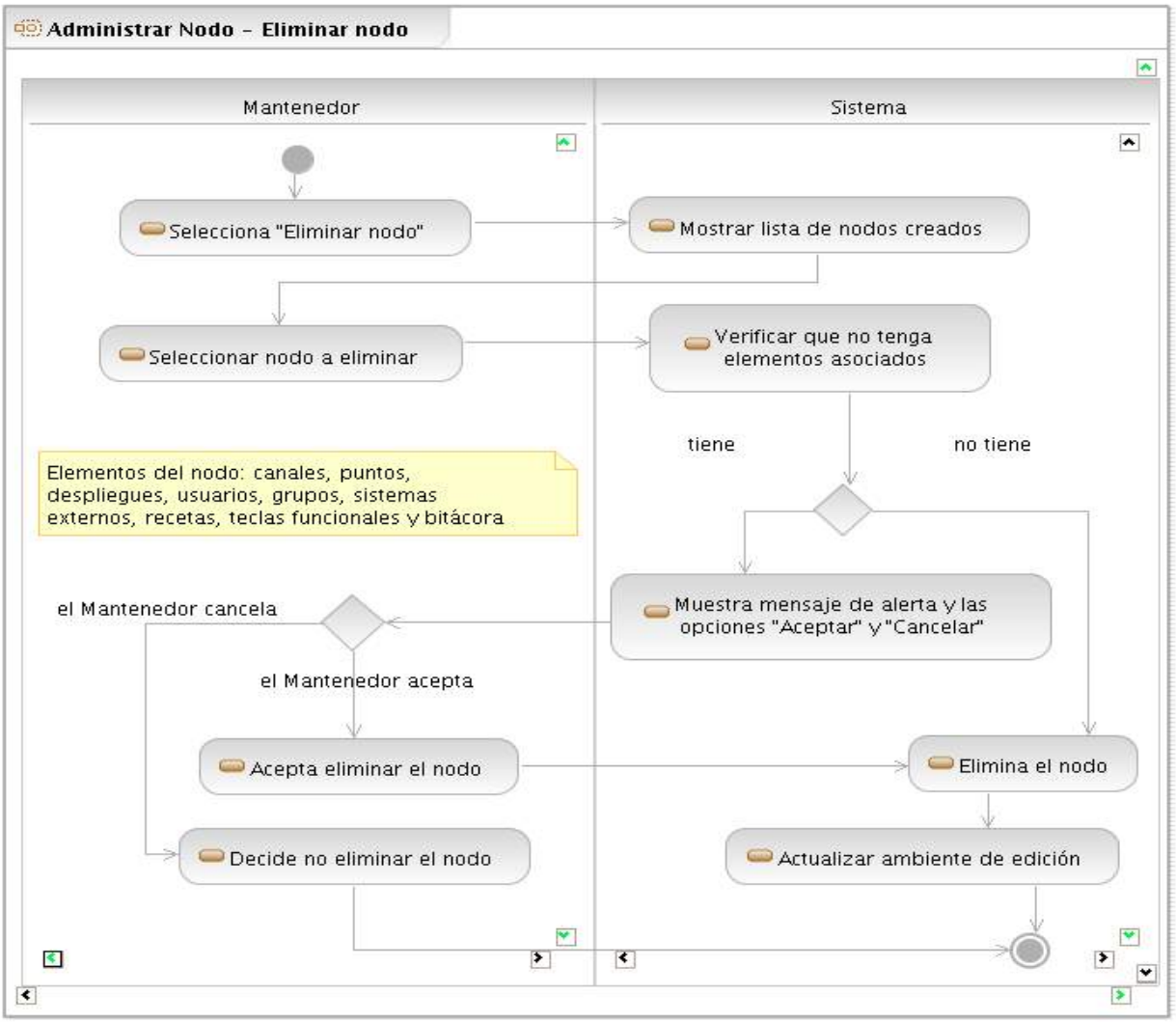


Figura 14. Diagrama de actividades del caso de uso Administrar Nodo – Eliminar nodo

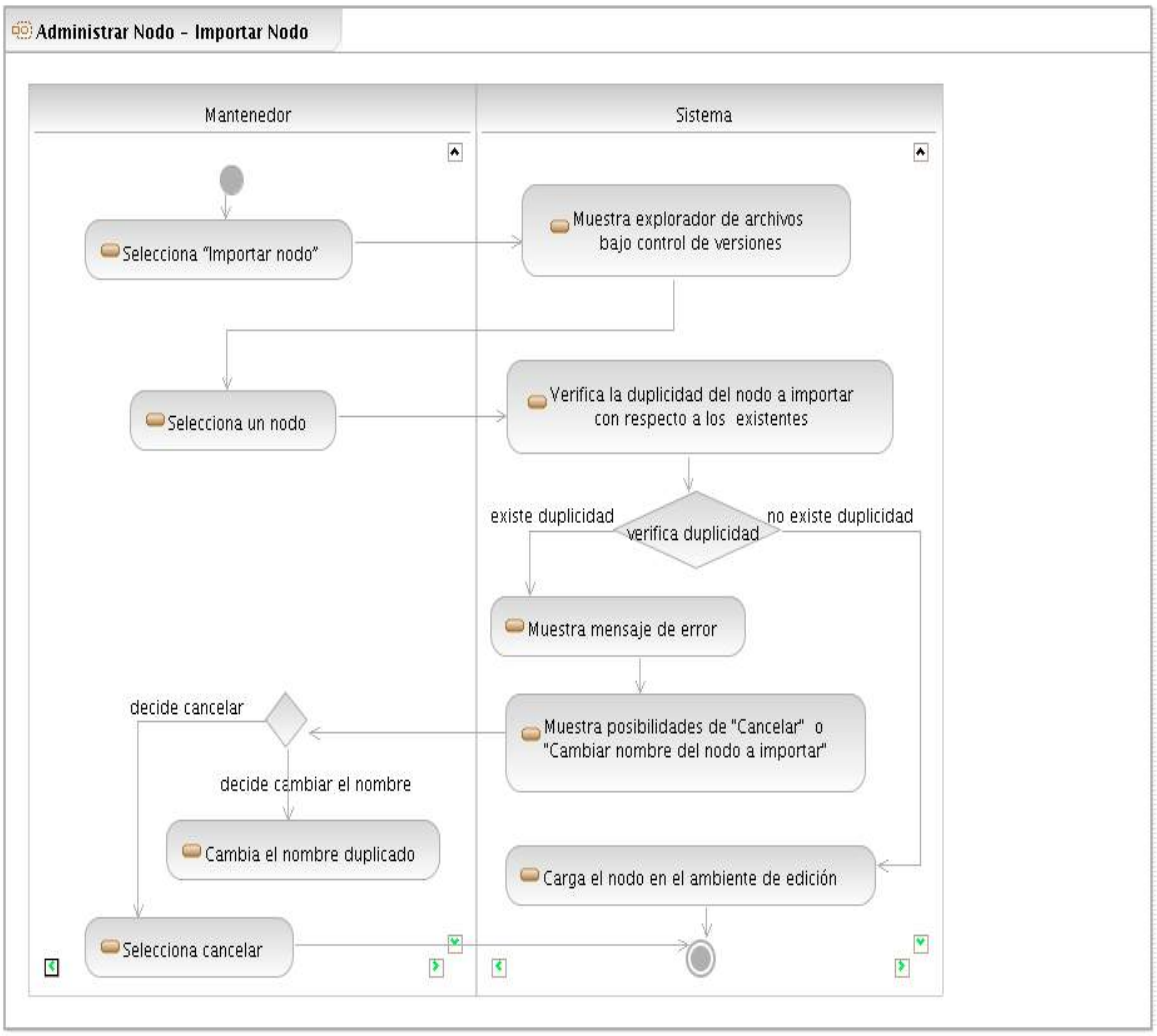


Figura 15. Diagrama de actividades del caso de uso Administrar Nodo – Importar nodo

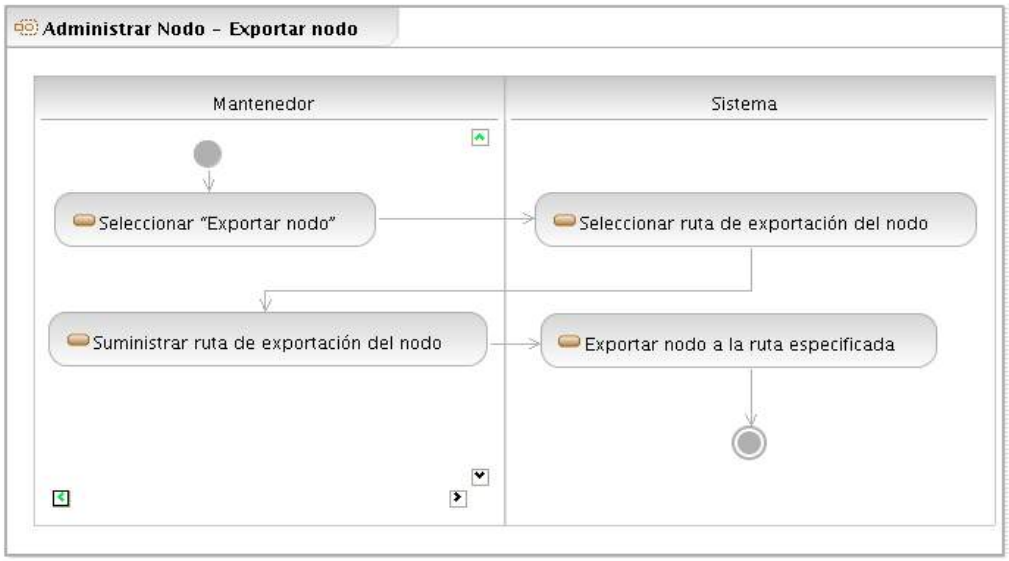


Figura 16. Diagrama de actividades del caso de uso Administrar Nodo – Exportar nodo

1.21 Prototipo de Interfaz de Usuario

Los prototipos de interfaz de usuario nos ayudan a comprender y especificar las interacciones entre los actores humanos y sistema durante la captura de requisitos. No sólo nos ayuda a desarrollar una interfaz gráfica mejor, sino también a comprender mejor los casos de usos. A la hora de especificar la interfaz gráfica de usuario también pueden utilizarse otros artefactos, como los modelos de interfaz gráfica y los esquemas de pantalla. **(RUMBAUGH et al. 2000)**

Caso de uso Administrar Proyectos

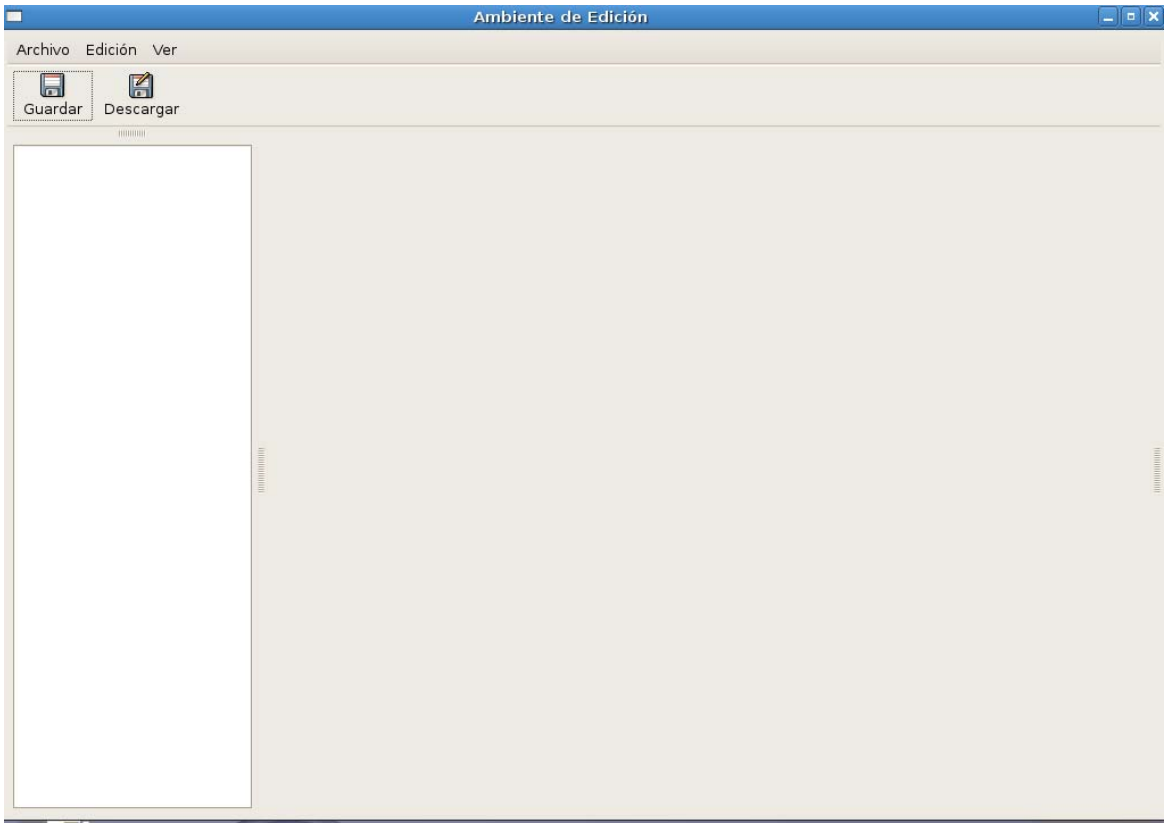


Figura 17. Interfaz de la primera vista del ambiente de edición del sistema.

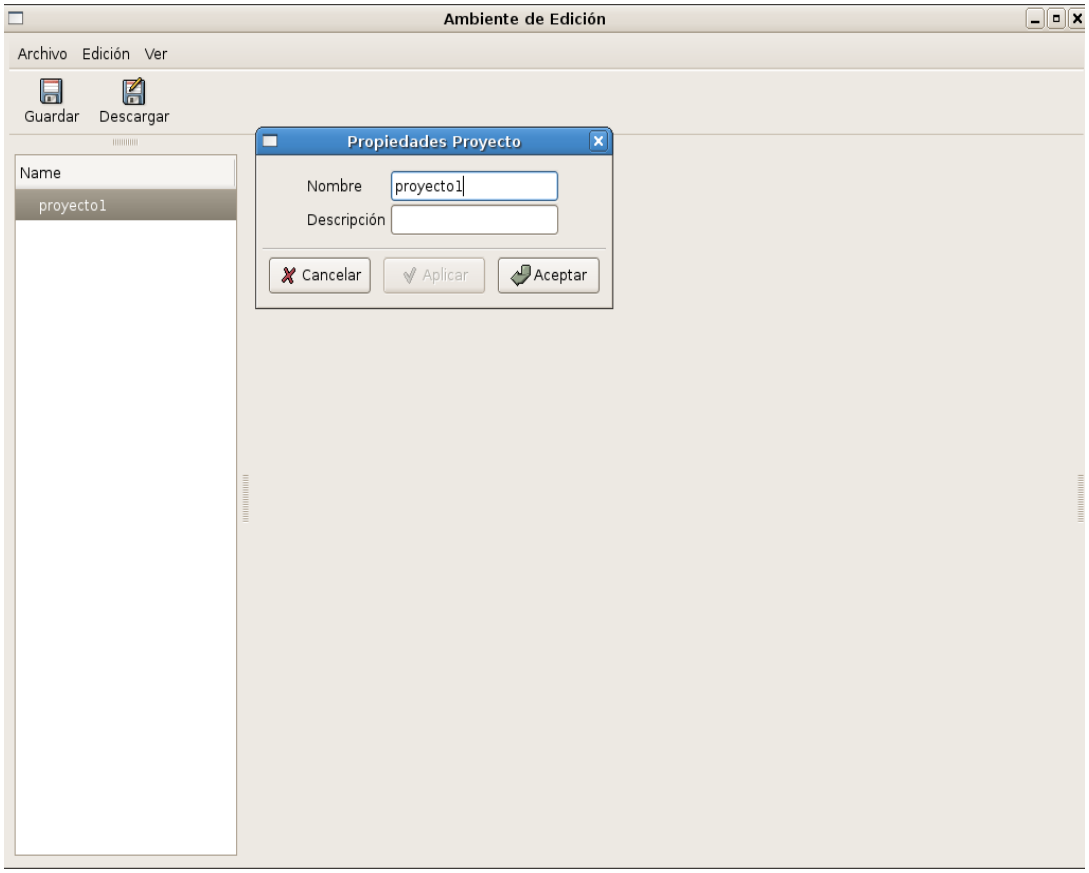


Figura 18. Interfaz del caso de uso Administrar Proyecto con una ventana modal mostrando las propiedades del proyecto a adicionar (o a modificar).

1.21.1 Caso de uso Administrar Nodo

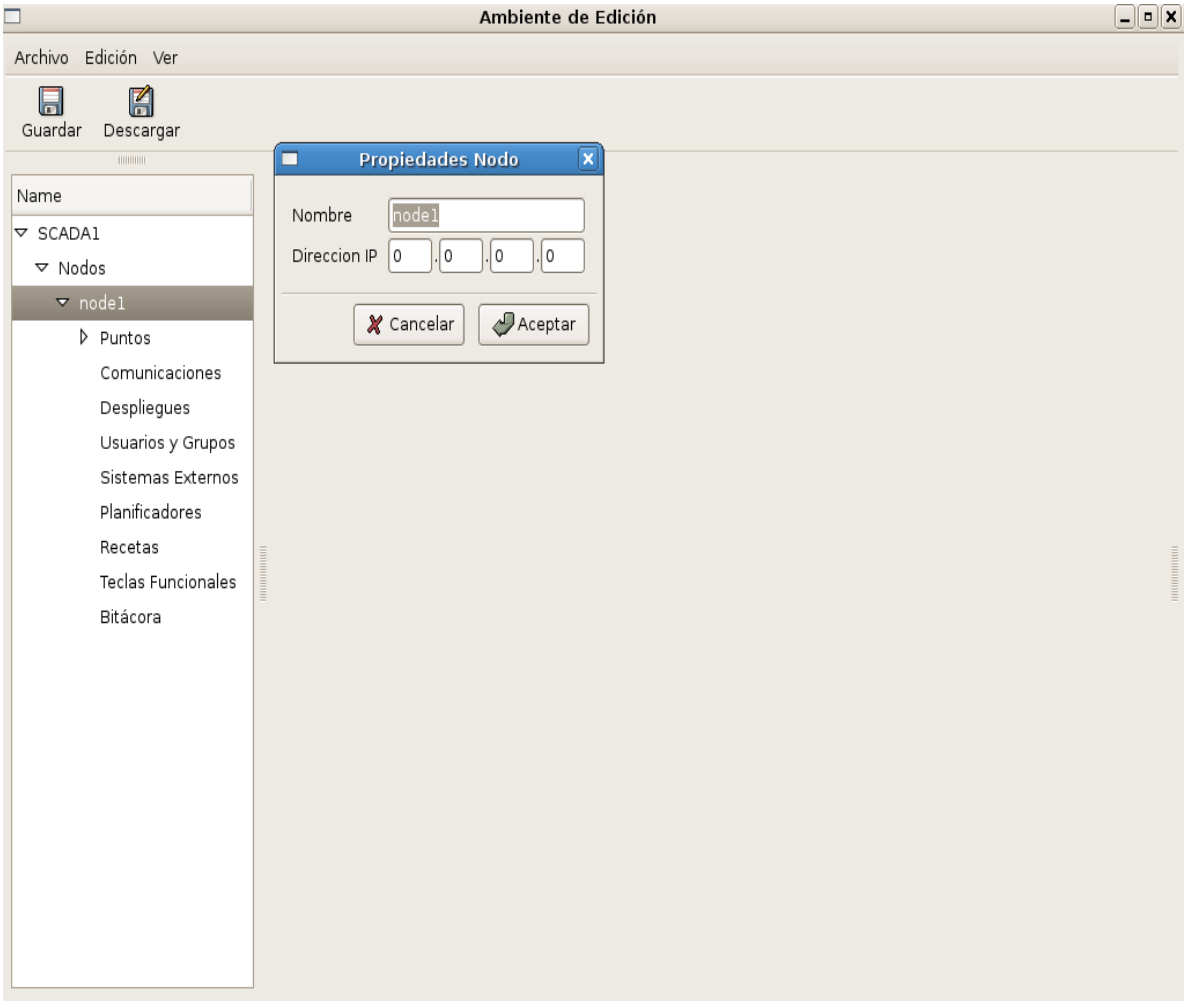


Figura 19. Interfaz del caso de uso Administrar Nodo con una ventana modal mostrando las propiedades del nodo.

1.21.2 Caso de uso Administrar Punto Analógico

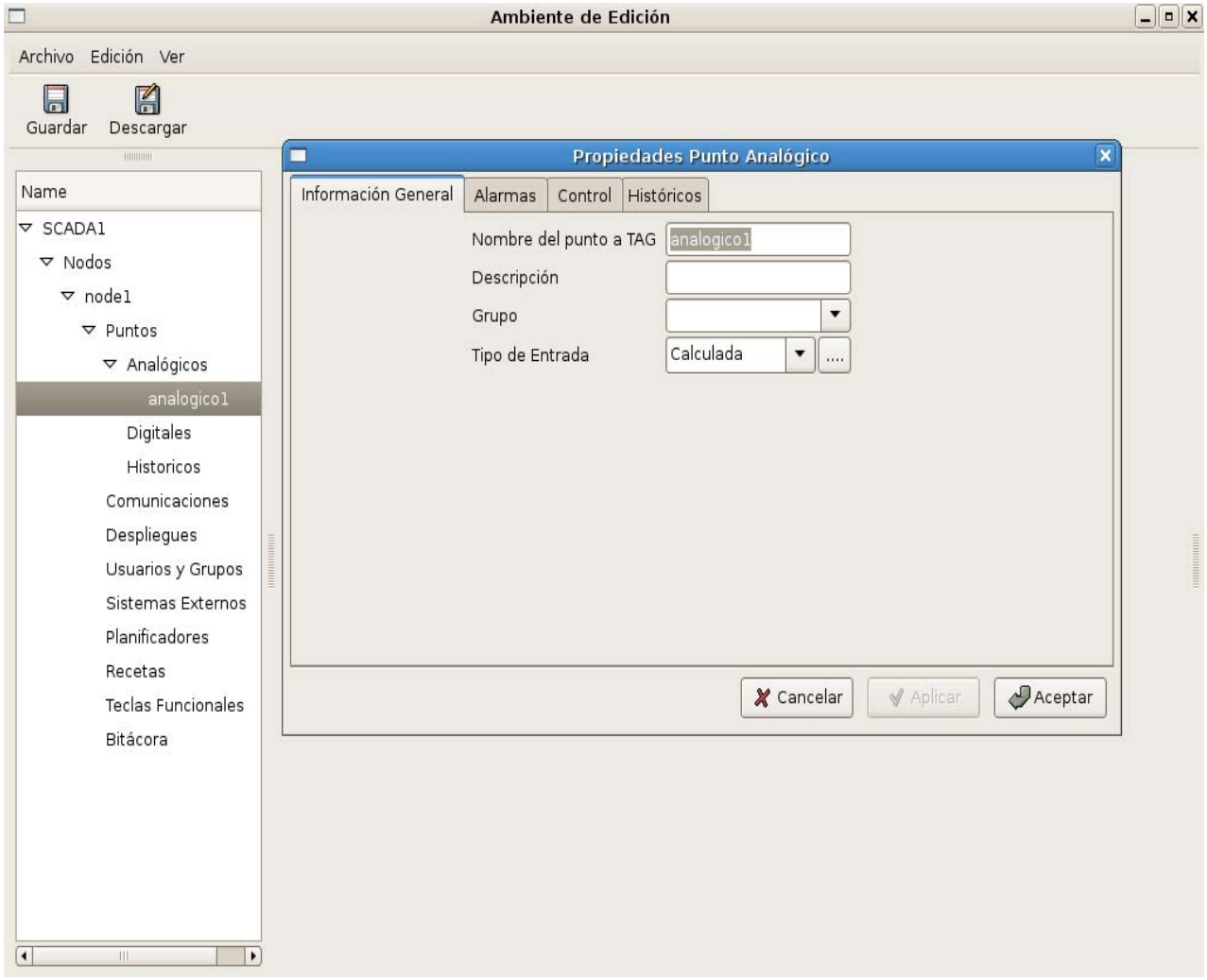


Figura 20. Interfaz del caso de uso Administrar Punto Analógico con una ventana modal mostrando la información general del punto.

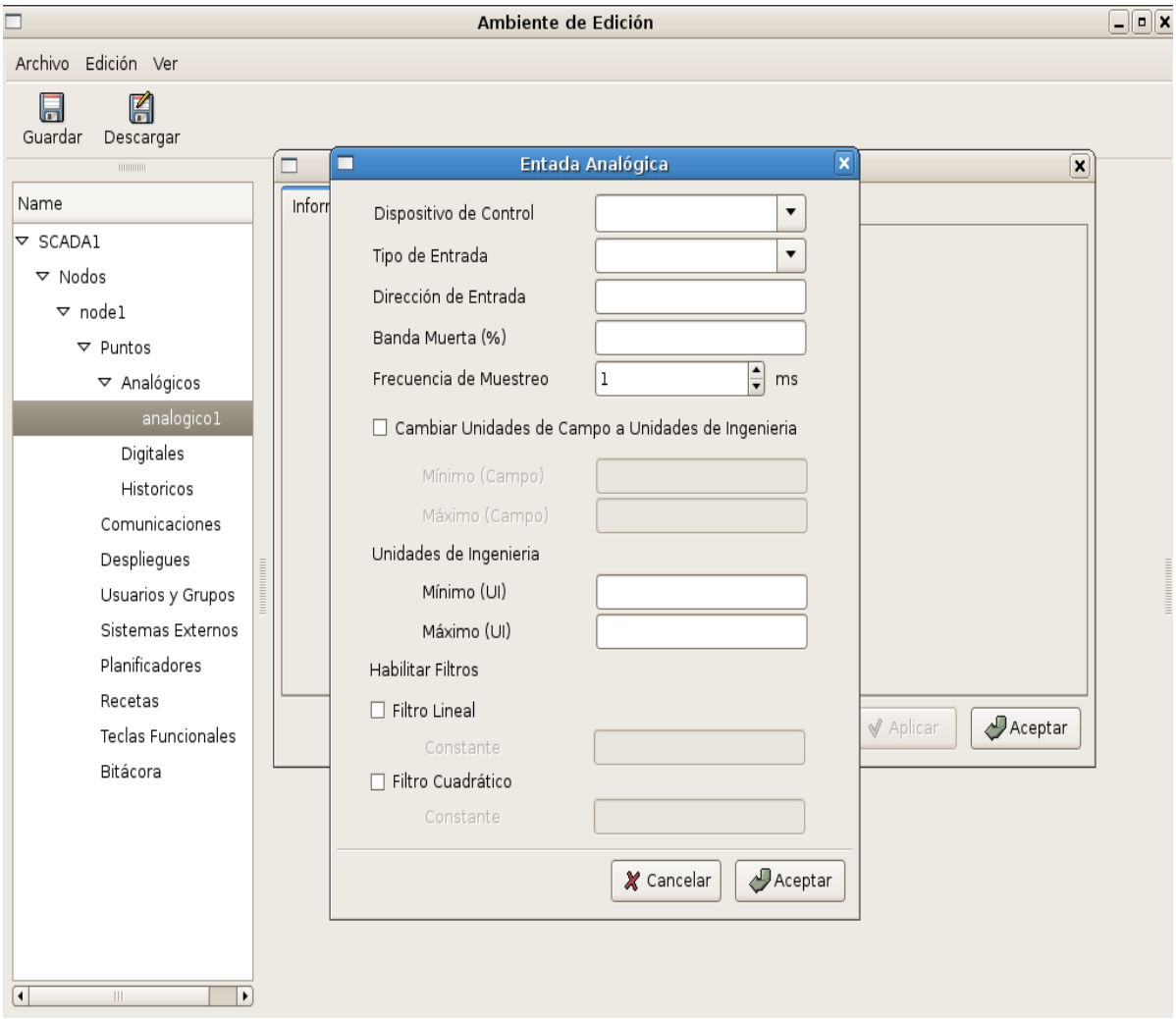


Figura 21. Interfaz del caso de uso Administrar Punto Analógico con una ventana modal mostrando los datos de entrada del punto.

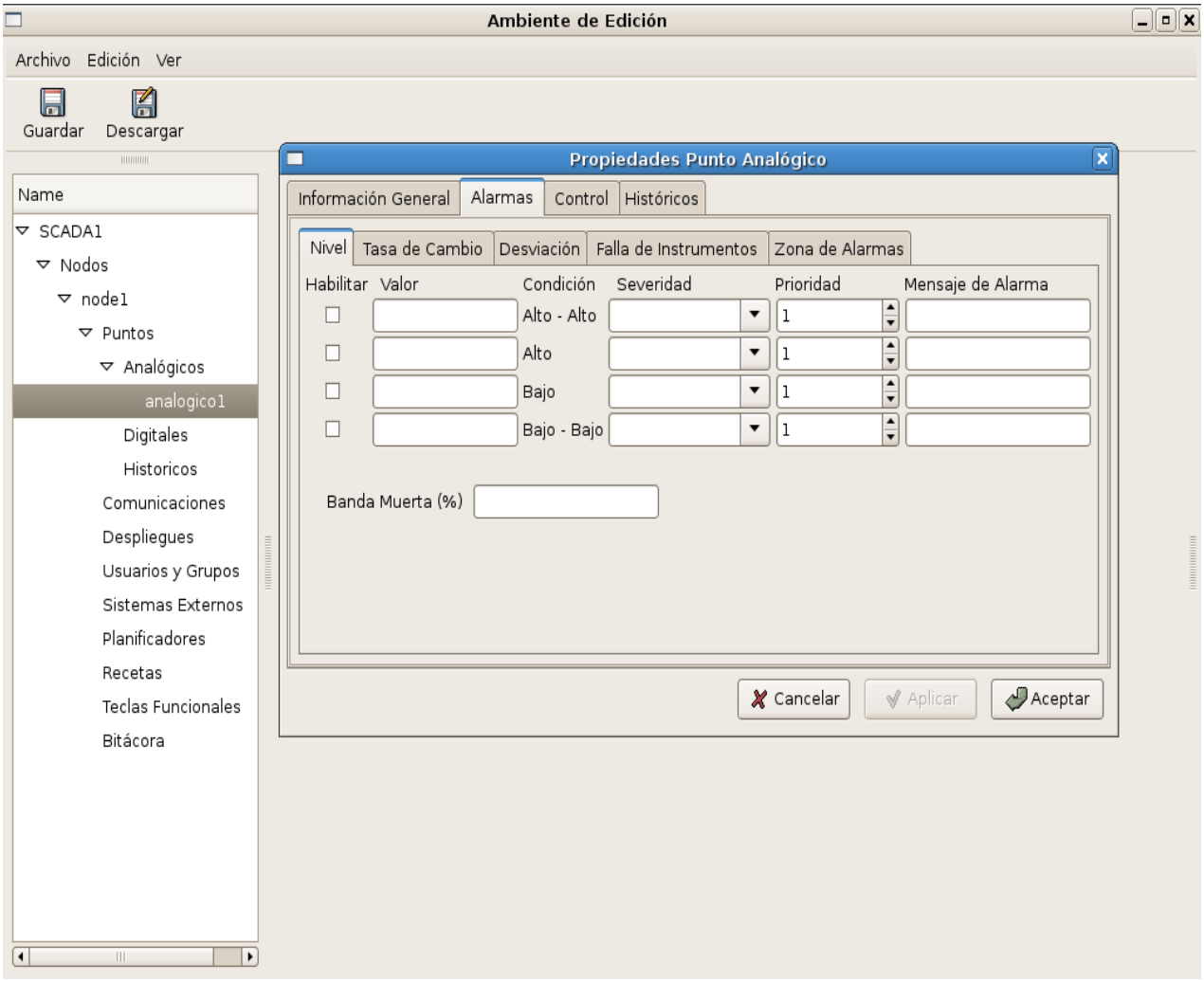


Figura 22. Interfaz del caso de uso Administrar Punto Analógico con una ventana modal mostrando los datos de la alarma de Nivel del punto.

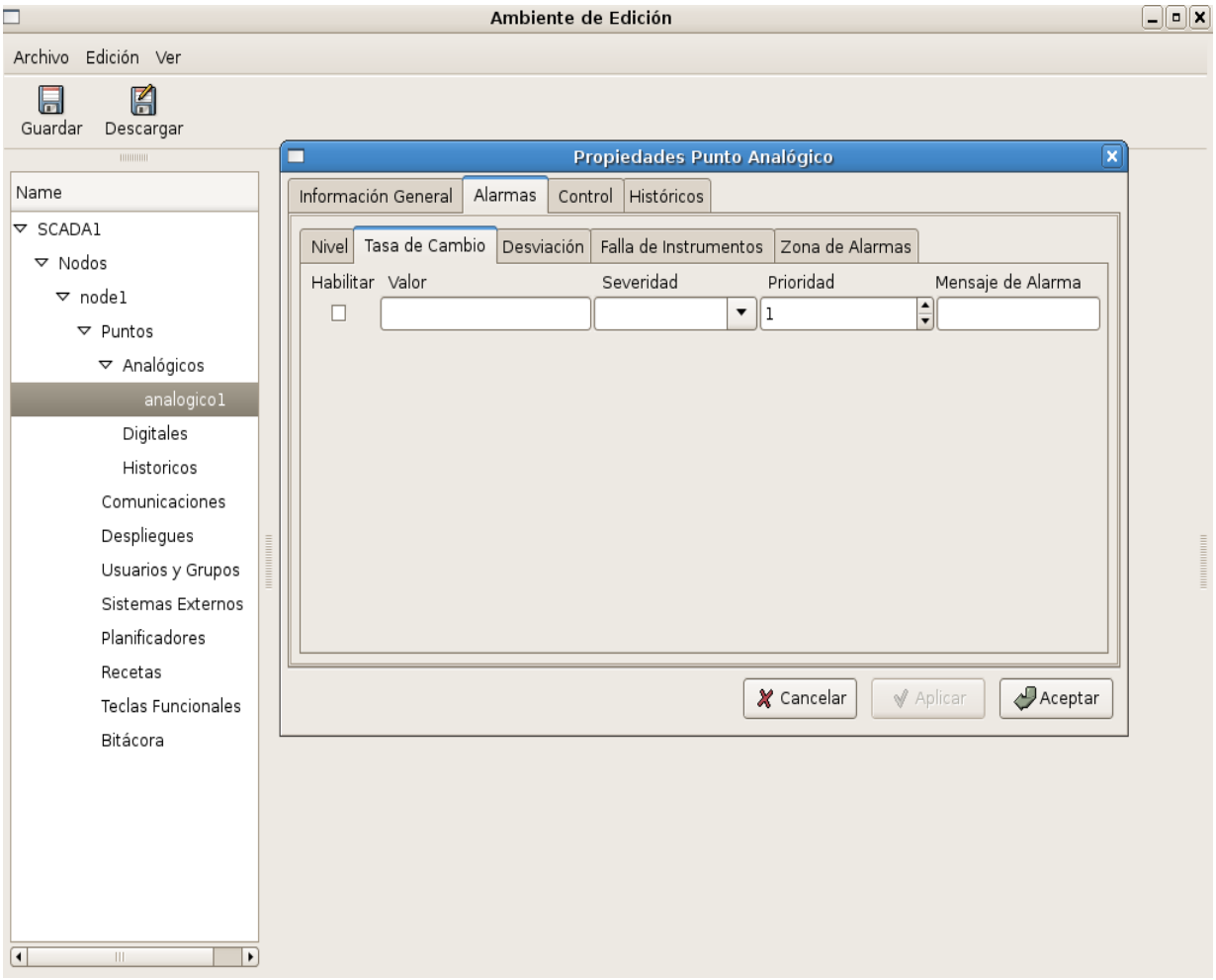


Figura 23. Interfaz del caso de uso Administrar Punto Analógico con una ventana modal mostrando los datos de la alarma Tasa de Cambio del punto, que coinciden con los de la alarma de Desviación.

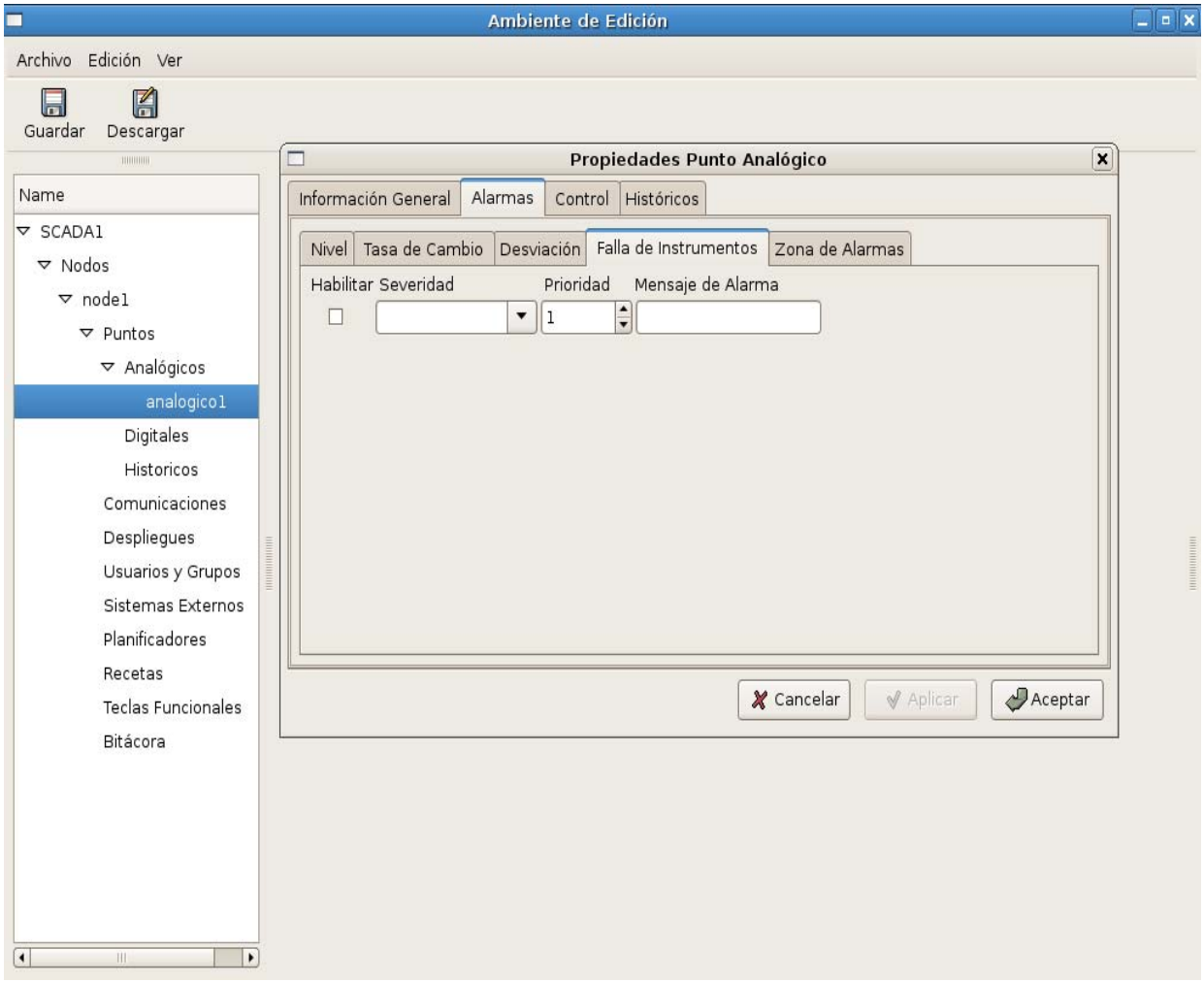


Figura 24. Interfaz del caso de uso Administrar Punto Analógico con una ventana modal mostrando los datos de la alarma Falla de Instrumentos del punto.

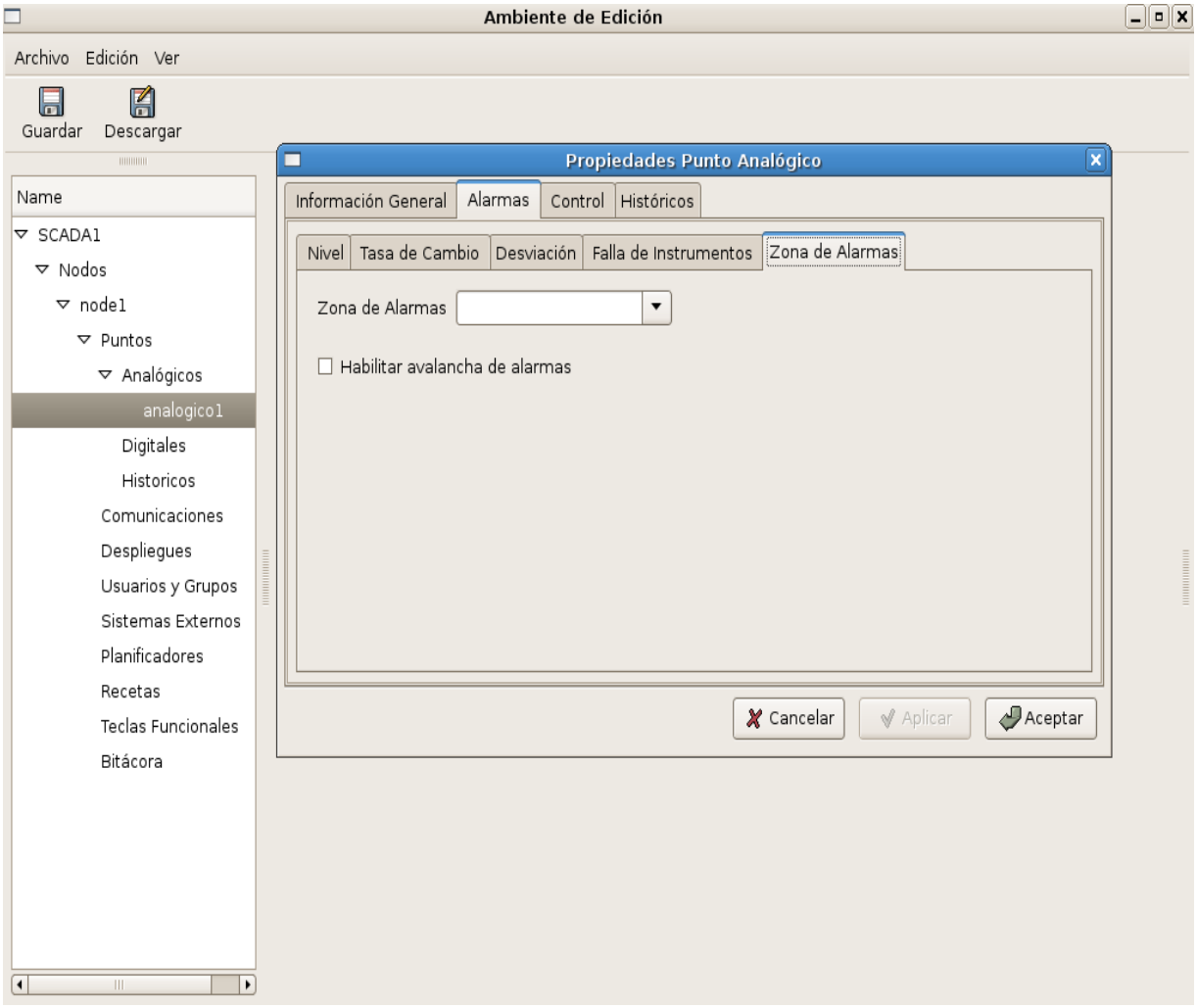


Figura 25. Interfaz del caso de uso Administrar Punto Analógico con una ventana modal mostrando la opción de seleccionar la zona de alarmas a la que pertenece el punto.

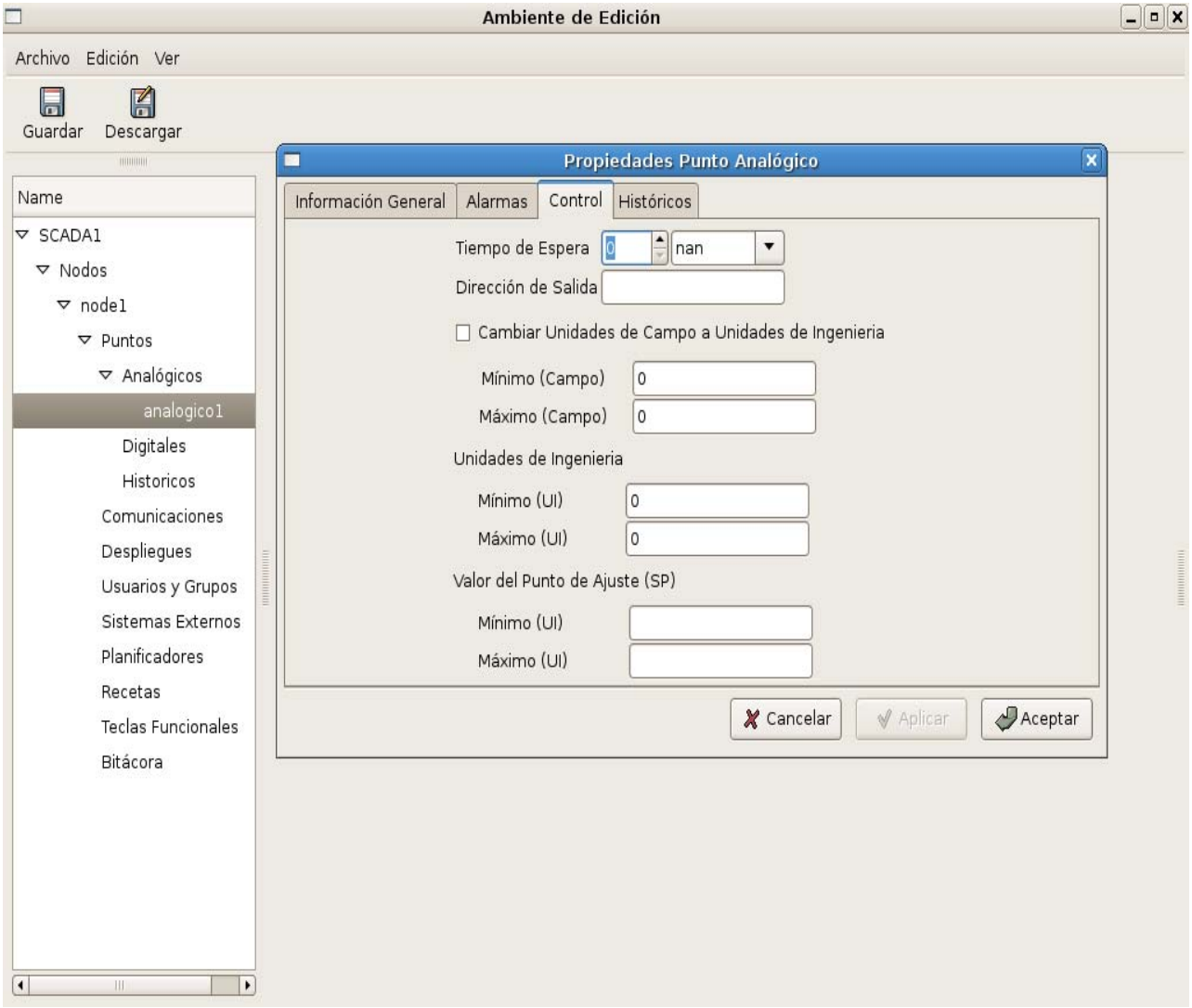


Figura 26. Interfaz del caso de uso Administrar Punto Analógico con una ventana modal mostrando los datos de control del punto.

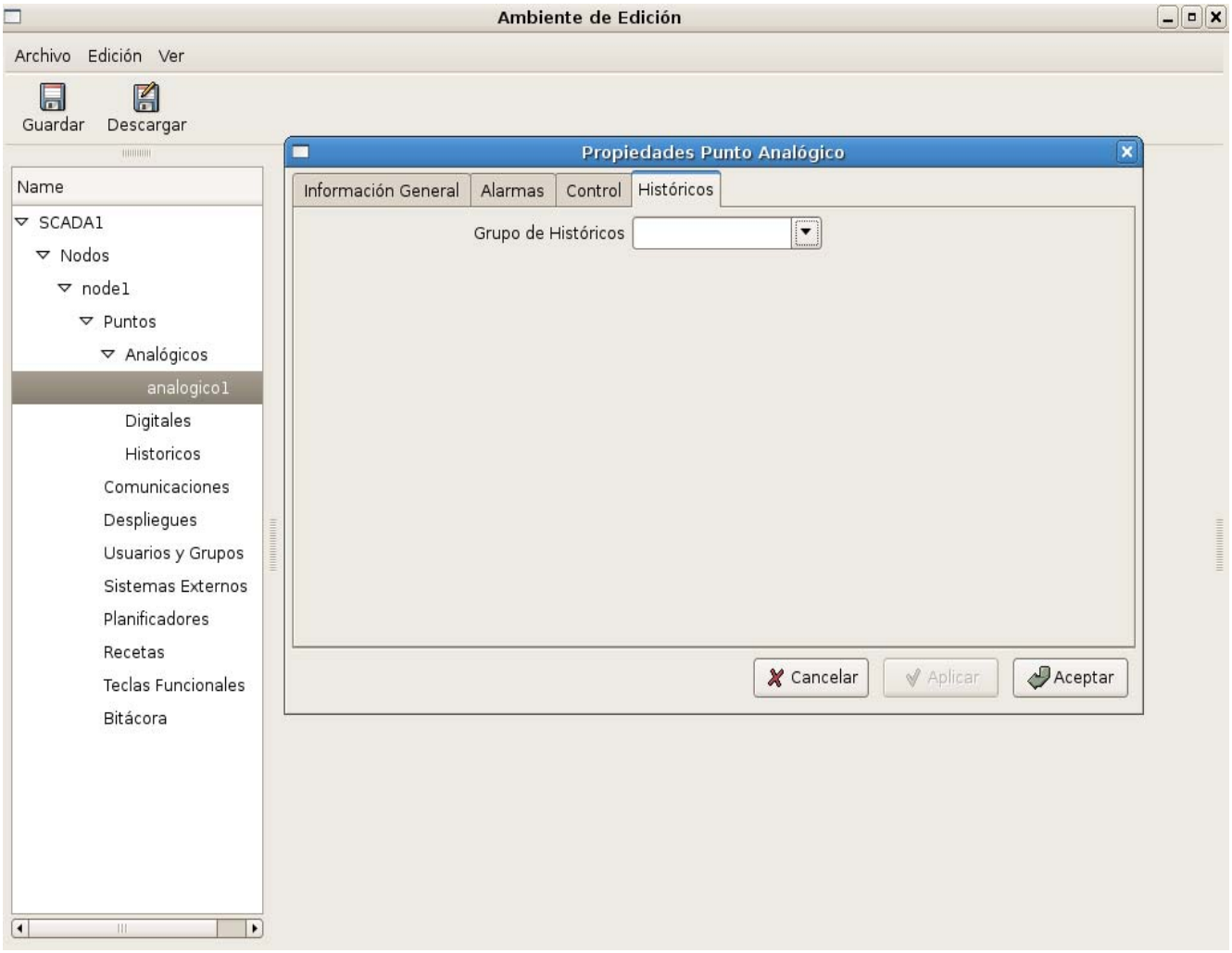


Figura 27. Interfaz del caso de uso Administrar Punto Analógico con una ventana modal mostrando la opción de seleccionar el grupo de históricos al que pertenece el punto.

1.21.3 Caso de uso Administrar Punto Digital

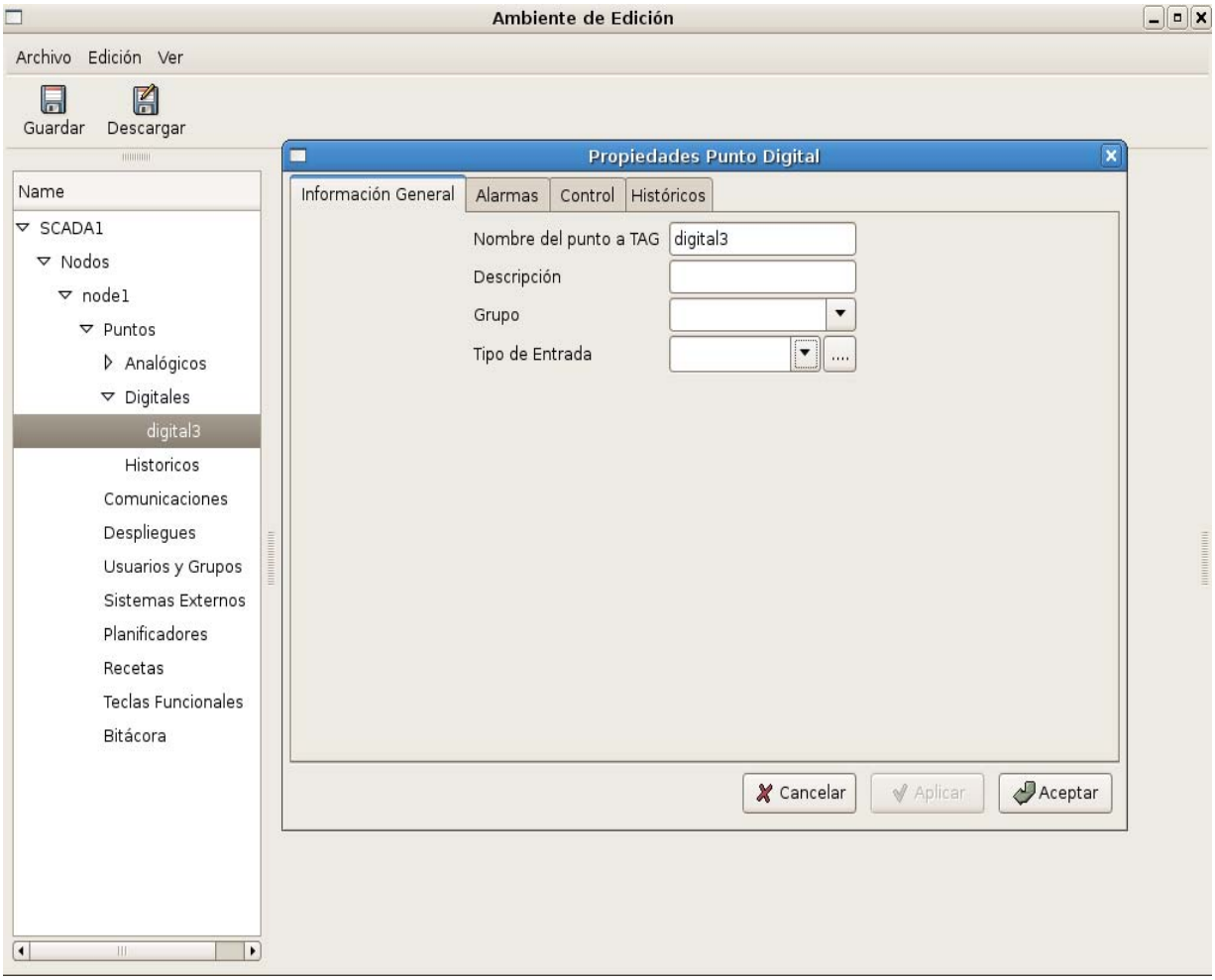


Figura 28. Interfaz del caso de uso Administrar Punto Digital con una ventana modal mostrando la información general del punto.

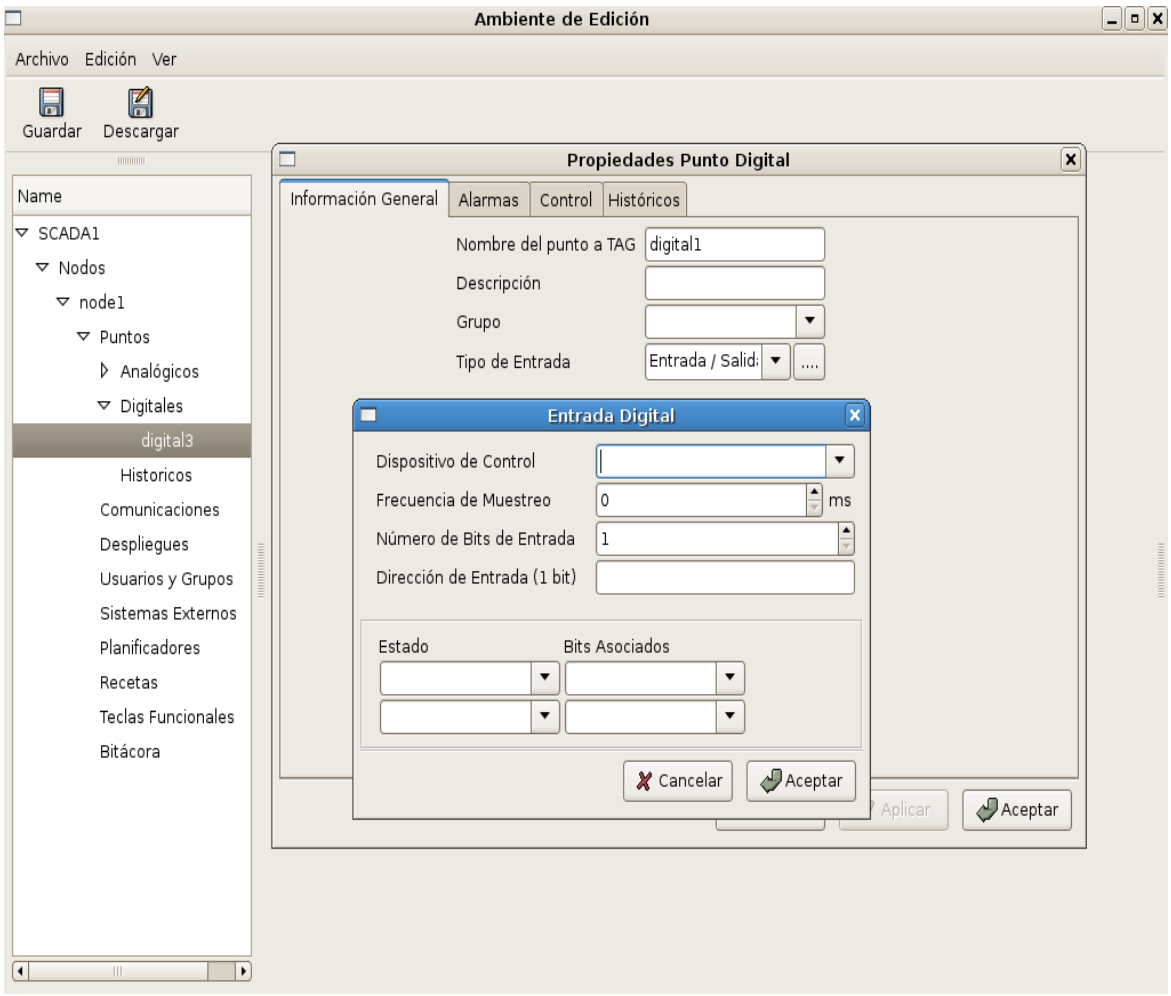


Figura 29. Interfaz del caso de uso Administrar Punto Digital con una ventana modal mostrando los datos de entrada del punto para el tipo de entrada seleccionado.

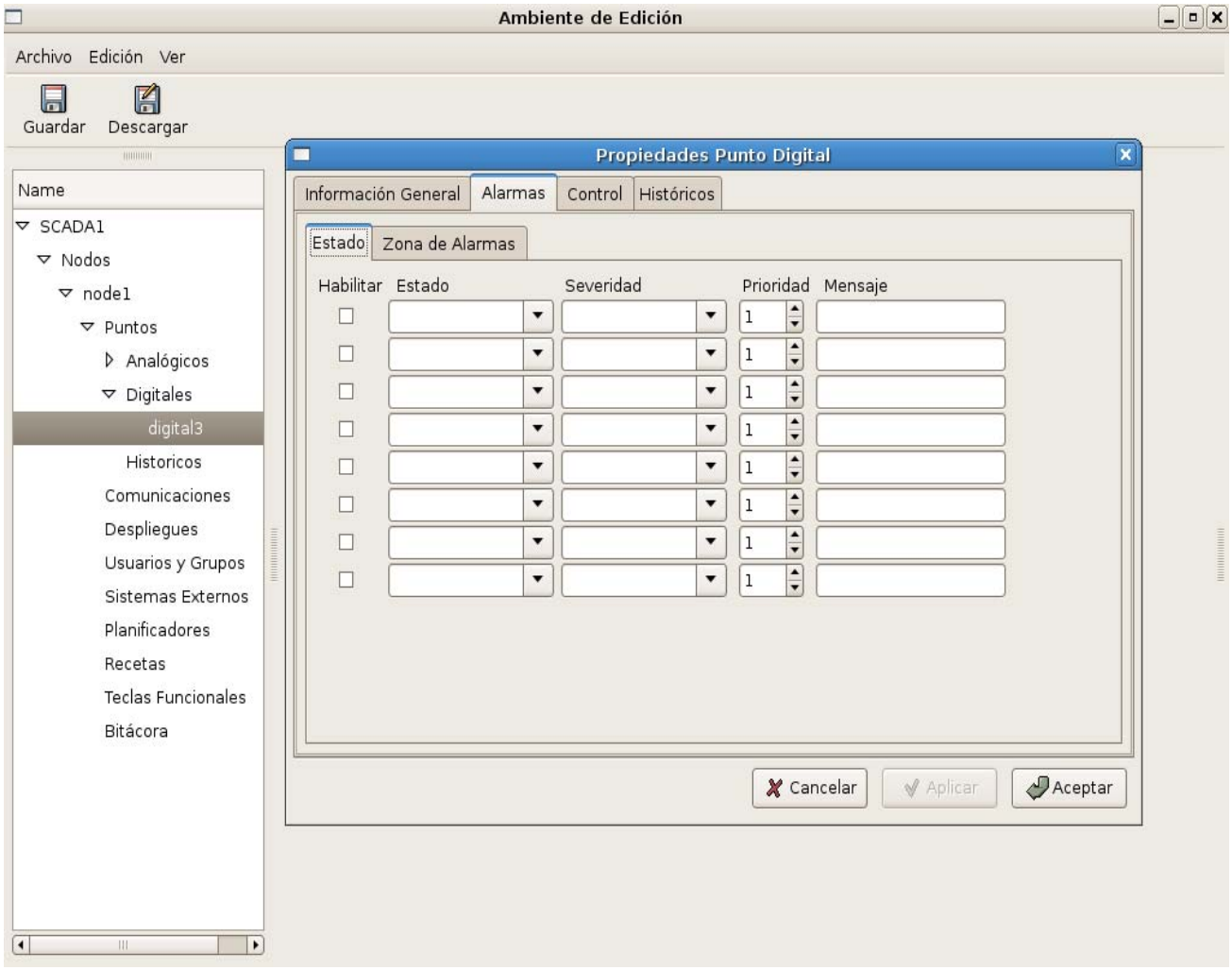


Figura 30. Interfaz del caso de uso Administrar Punto Digital con una ventana modal mostrando los datos de la alarma del punto.

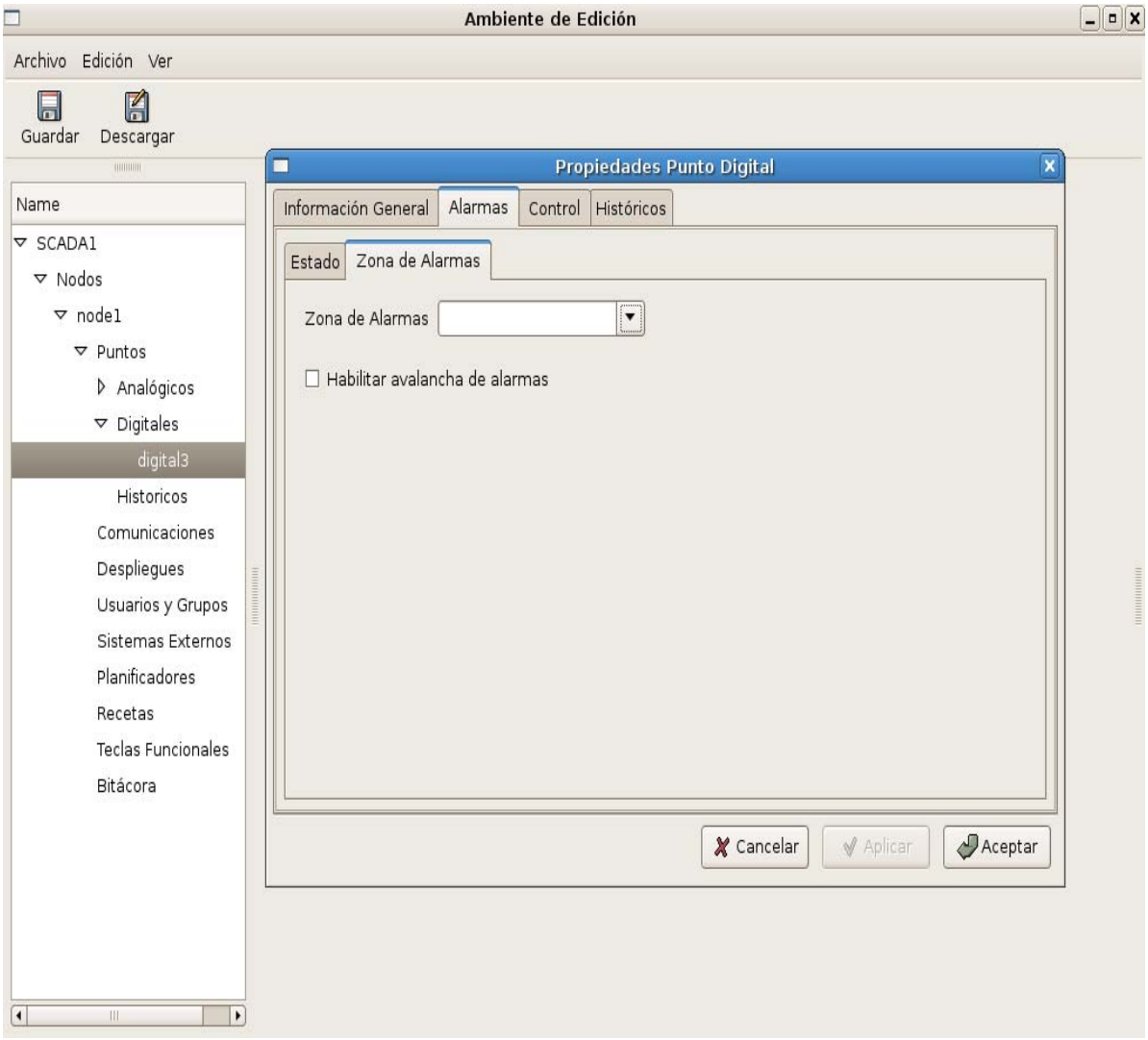


Figura 31. Interfaz del caso de uso Administrar Punto Digital con una ventana modal mostrando la opción de seleccionar la zona de alarmas a la que pertenece el punto.

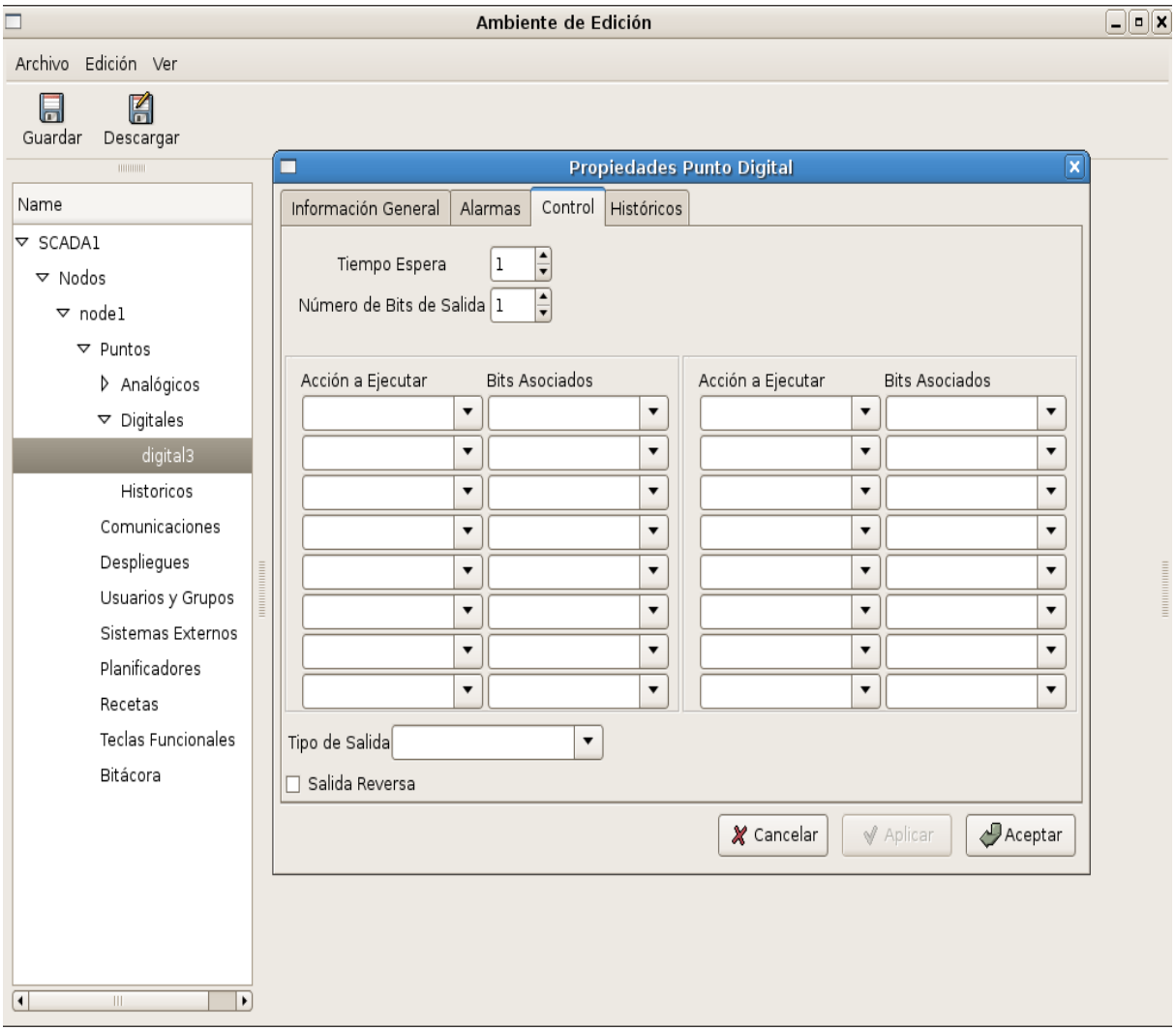


Figura 32. Interfaz del caso de uso Administrar Punto Digital con una ventana modal mostrando los datos de control del punto.

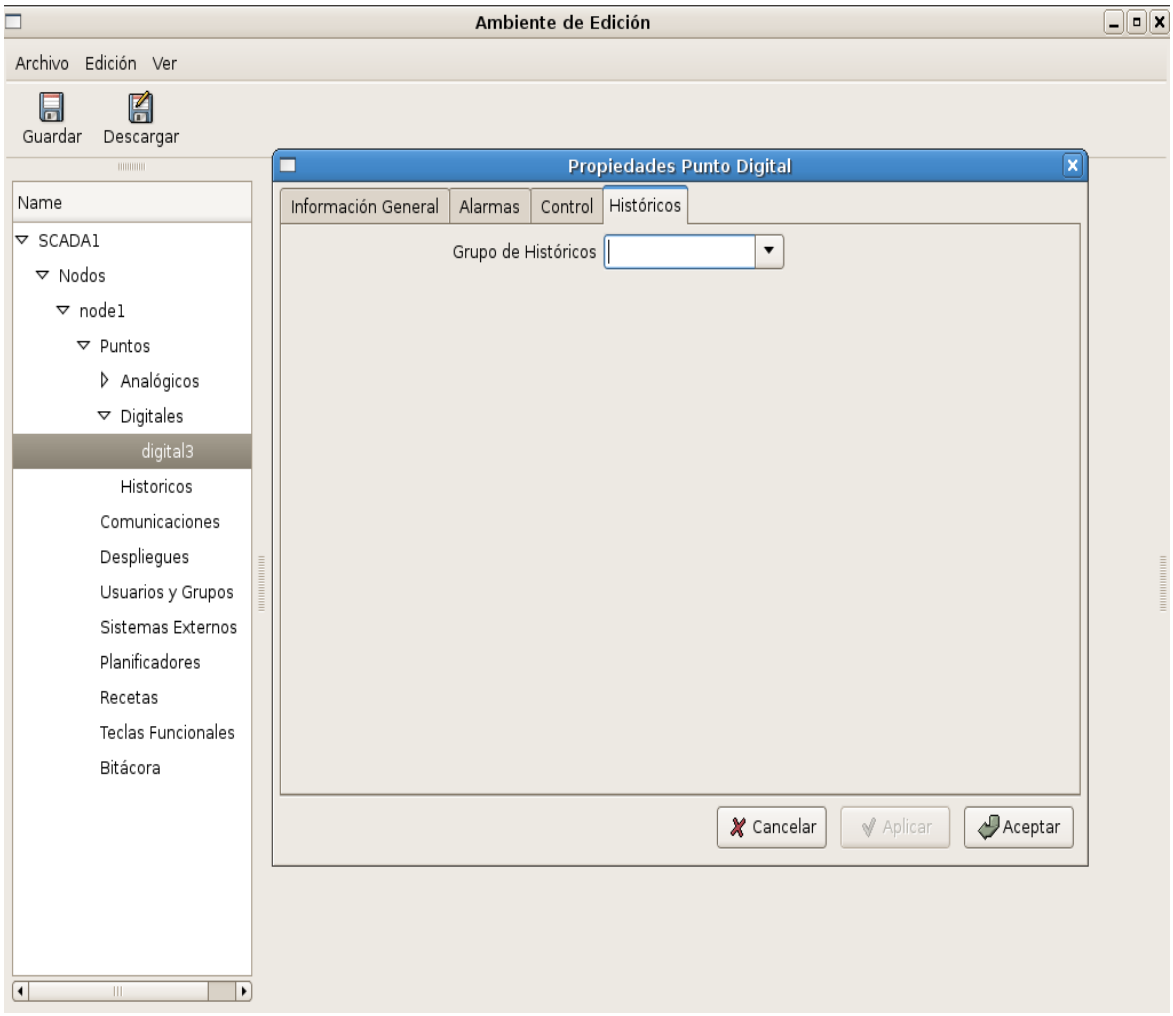


Figura 33. Interfaz del caso de uso Administrar Punto Digital con una ventana modal mostrando la opción de seleccionar el grupo de históricos que pertenece el punto.

1.21.4 Caso de uso Administrar Canal

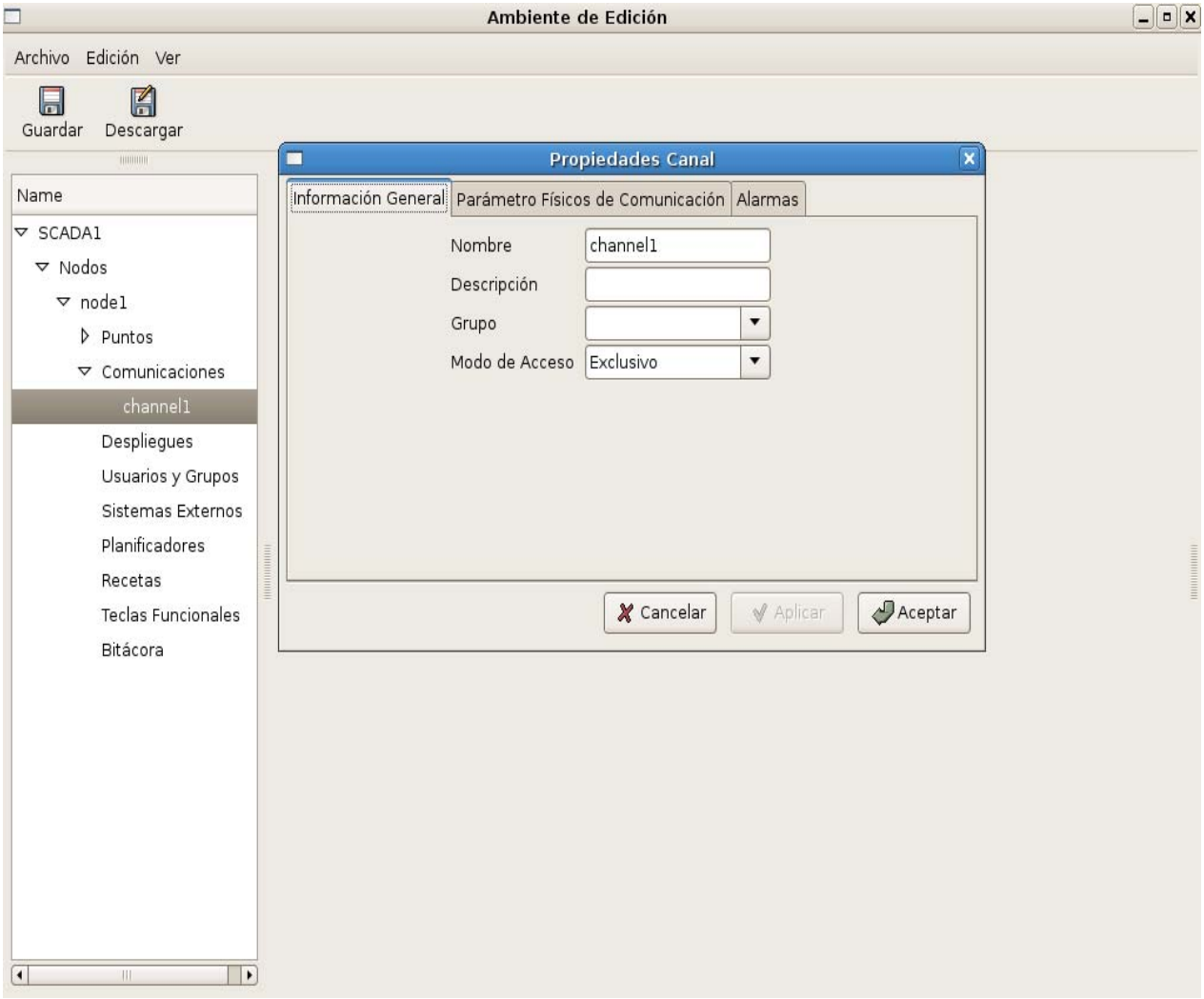


Figura 34. Interfaz del caso de uso Administrar Canal con una ventana modal mostrando las propiedades del canal.

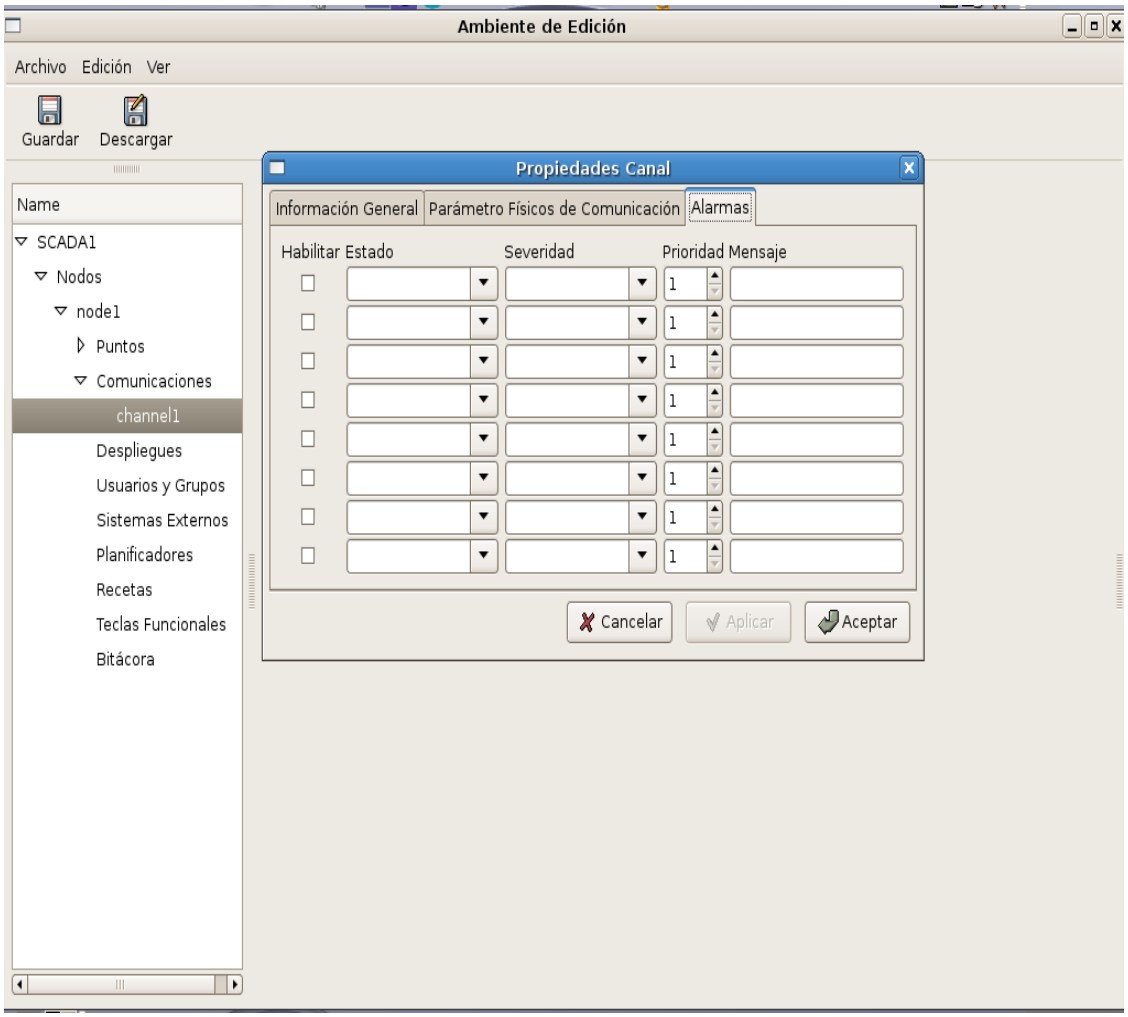


Figura 35. Interfaz del caso de uso Administrar Canal con una ventana modal mostrando las propiedades las alarmas del canal.

1.21.5 Caso de uso Administrar Subcanal

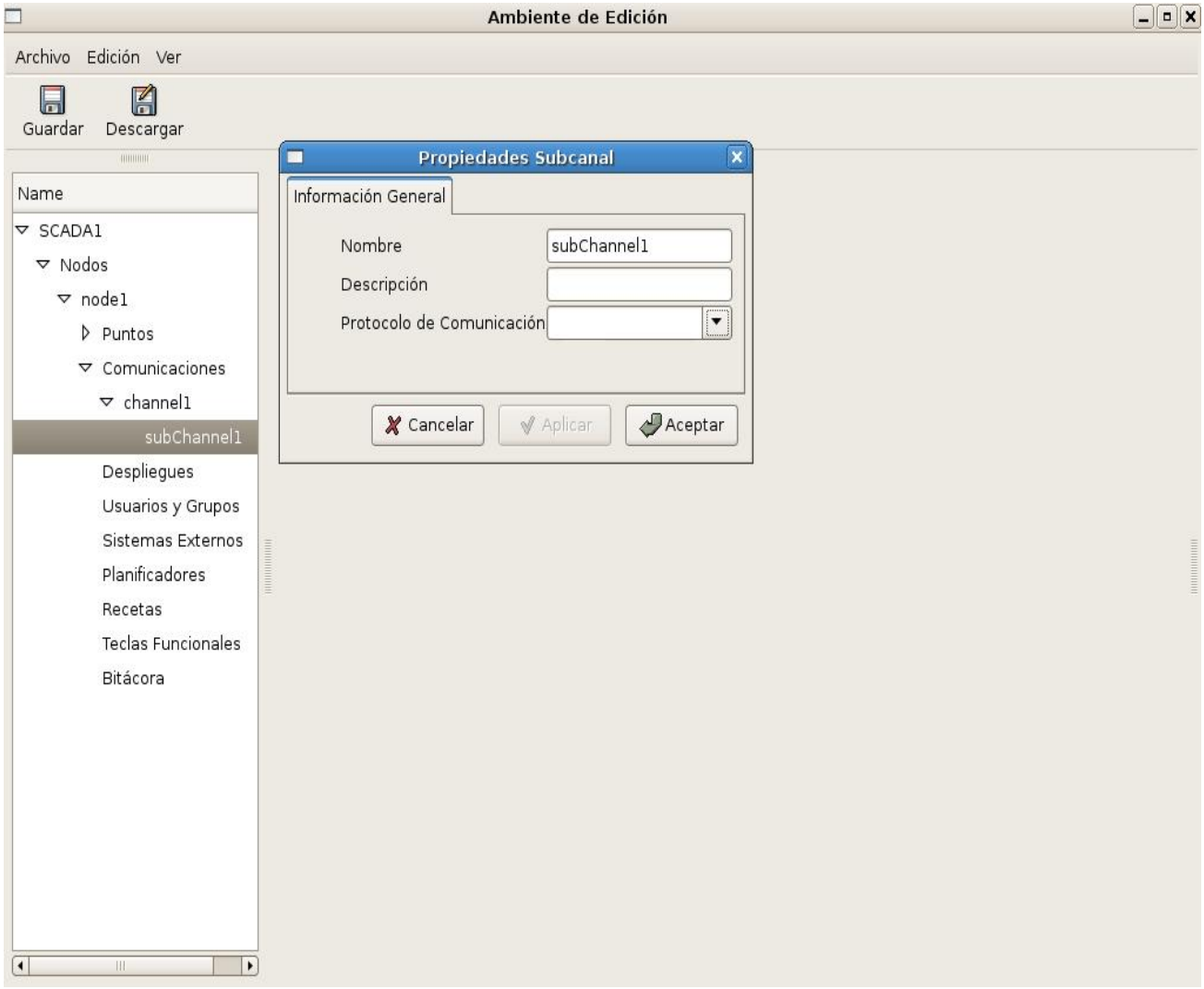


Figura 36. Interfaz del caso de uso Administrar Subcanal con una ventana modal mostrando sus propiedades.

1.21.6 Caso de uso Administrar Dispositivo

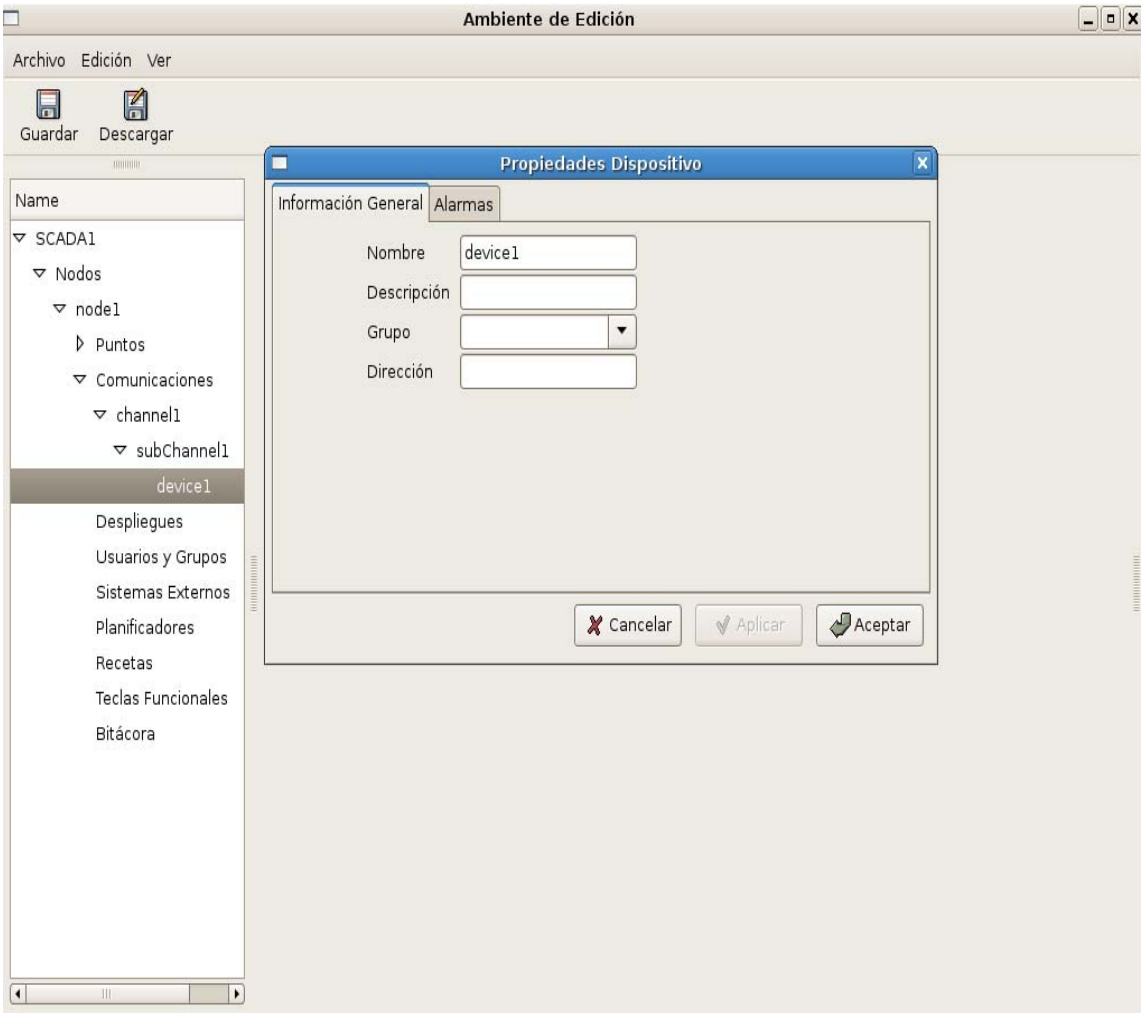


Figura 37. Interfaz del caso de uso Administrar Dispositivo con una ventana modal mostrando sus propiedades.

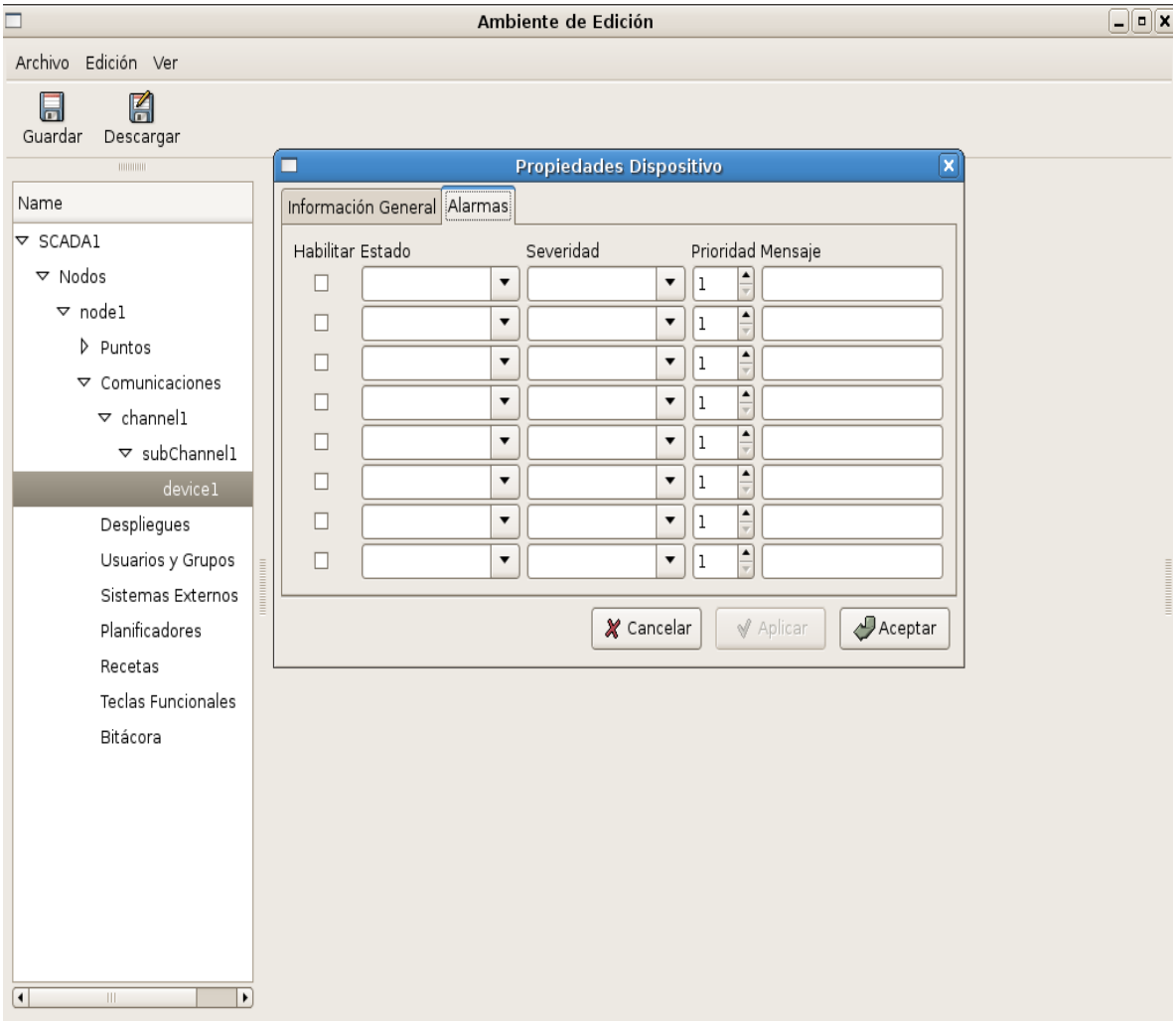


Figura 38. Interfaz del caso de uso Administrar Dispositivo con una ventana modal mostrando los datos de las alarmas.

1.22 Conclusiones

Haciendo uso de las técnicas y herramientas seleccionadas se obtuvo el levantamiento de requisitos del módulo de Gráficos Vectoriales, dándole realización a una parte de las actividades correspondiente a la etapa de análisis. Se concluyó con la validación de algunos requerimientos críticos mediante un prototipo que contó con la aprobación del cliente conciente de que lo que estaba viendo era un prototipo y no el sistema final.

Conclusiones

- Para realizar el levantamiento de requisitos de sistemas complejos y desconocidos es importante la participación constante del cliente como especialista del tema, lo cual evita la falta de identificación de los verdaderos requerimientos.
- La elaboración de diagramas de actividades y el diseño del prototipo de interfaz para el módulo de Gráficos, facilitan la comprensión de los casos de uso descritos.
- Para realizar el diseño sobre plataformas de Software Libre existen un conjunto de tecnologías que de aplicarse correctamente perfeccionan el flujo de captura de requisitos.

Recomendaciones

- Darle continuidad al trabajo basándose en el resto de los requerimientos capturados que no fueron validados.
- Profundizar en el estudio de otros SCADAs buscando ampliar las funcionalidades del sistema.
- Usar los resultados obtenidos en la captura de requerimientos para el diseño del SCADA Nacional.

REFERENCIAS BIBLIOGRÁFICAS

1. ALARCOS, G. *Ingeniería Inversa*, 20/04/07]. Disponible en: http://alarcos.inf-cr.uclm.es
2. ANACHE, I. and M. JOEL. *OMG UML 2.0 -Marcando un hito en el desarrollo de software.* Habana, 2005. p.
3. CHAVEZ, H. *Decreto No. 3.390* Gaceta Oficial No. 38.095, 2004. [Disponible en: www.gobiernoenlinea.ve
4. DEBIAN, C. *Acerca de Debian*, 2007. [21/03/07]. Disponible en: www.debian.org
5. ECLIPSE, F. D. *Eclipse (software)*, 2007. [13/12/2006]. Disponible en: http://es.wikipedia.org
6. EQUIPO, N. S. *Propuesta para la selección de la plataforma de desarrollo de software*, 2005.
7. INDUSTRIALES, G. D. A. *Sistema de visualización WINCC*, 2005. [17/02/07]. Disponible en: www.automatas.org
8. ISW, P. D. *Contenido de Residencias Profesionales en el área de Software*, 2006. [03/003/07]. Disponible en: www.itchihuahuaii.edu.mx
9. KOCH, N. *Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo*, 2007. [Disponible en: www.pst.informatik.uni-muenchen.de
10. LUGONES, H. L. A. *Sistema Automatizado de Diagnóstico, Supervisión y/o Control del Consumo de Energía Eléctrica en Instalaciones Industriales y de Servicio.*, [En línea]. 2005. [Disponible en:
11. MOYA, R. *Introducción al proyecto GNOME.*, 2004. [Disponible en: http://www.es.gnome.org
12. PDVSA Soberanía plena en soluciones AIT para el sector energético aportando valor social.
13. PRESSMAN, R. S. *Ingeniería de software. Un enfoque práctico*. 5a ed., Mc Graw Hill, 2005. 23-25 p. 0-. 9711647-3-8
14. PROGEA, I. A. S. *Movicon X*, 2007. [09/11/2006]. Disponible en: www.movicon-services.de
15. PROYECTO, S. *Documentación de anexo*, 2007. [12/05/2007]. Disponible en: http://10.35.1.2/repos/branches/irina/Documentos/
16. RUMBAUGH, J.; I. JACOBSON*, et al. El proceso unificado de desarrollo de software*. La Habana, Addison Wesley Longman, 2000. p.
17. SANGUINETTI, C. *Análisis y Diseño de Sistema*.
18. SCADA, N. *Ingeniería Conceptual Proyecto Desarrollo SCADA Nacional*, 2005.
19. TORRES, L. S. *El analista de sistemas y el paradigma estructurado*. Disponible en: ww.monografias.com
20. WIKIPEDIA, C. D. *Ingeniería Inversa*, 2007. [20/04/07]. Disponible en: http://es.wikipedia.org

ANEXOS

Anexo 1 Guía de entrevistas realizadas para el levantamiento de requisitos. (PROYECTO 2007)

Paso 1: Establecer el perfil del Stakeholder o Usuario.

1. Nombre y Apellidos.
2. Cargo.
3. ¿Cuáles son sus responsabilidades claves?
4. ¿Qué entregables produce? ¿Para quién?

Paso 2: Comprender el ambiente del usuario.

5. ¿Quiénes son los usuarios?
6. ¿Cuál es su nivel educativo?
7. ¿Los usuarios tienen experiencia con este tipo de aplicación?
8. ¿Qué plataformas están en uso? ¿Cuáles son sus planes para futuras plataformas?
9. ¿Qué tipos de documentación escrita y en línea UD. Necesita?

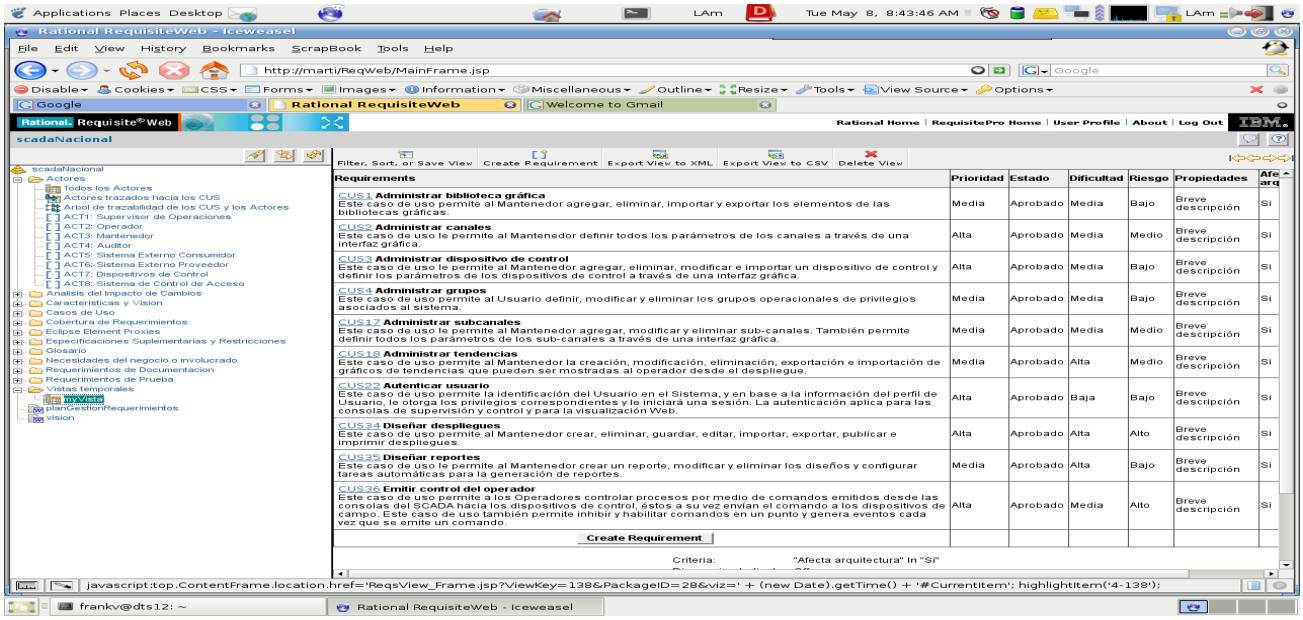
Paso 3: Evaluando la confiabilidad, desempeño y necesidades de mantenimiento.

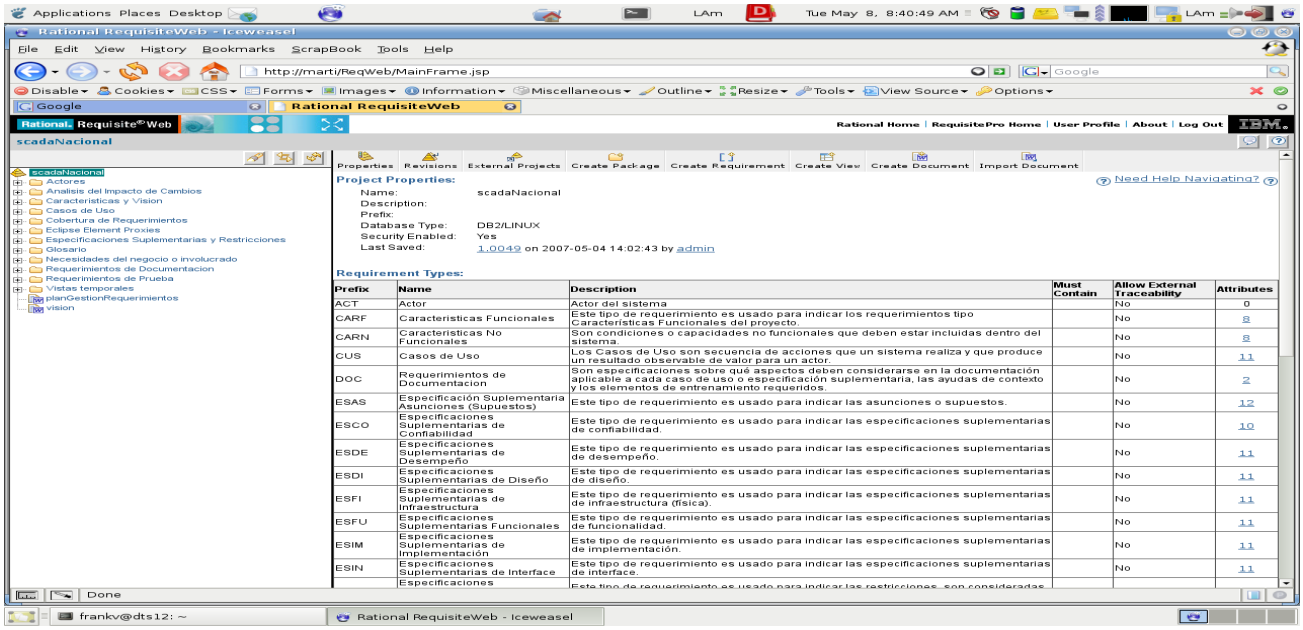
10. ¿Cuáles son sus expectativas de confiabilidad?
11. ¿Cuáles son los requerimientos especiales para la licencia?
12. ¿Conoce de algún otro requerimiento que debamos saber o tener en cuenta?

Paso 4: Resumiendo.

13. ¿Hay otras cuestiones que deba preguntarle?
14. ¿Si necesito hacer seguimiento, puedo llamarlo o escribirle al correo?
15. ¿Estaría dispuesto a participar en una revisión de los requerimientos?

Anexo 2 Imágenes de la utilización de la Herramienta Requisite Web. (PROYECTO 2007)





GLOSARIO DE TÉRMINOS

ADOCE: Con los objetos de datos ActiveX (ActiveX Data Objects, ADO) de Microsoft, las aplicaciones clientes pueden obtener acceso y manipular datos de un servidor de base de datos a través de un proveedor de base de datos OLE. Los objetos de datos ActiveX para Microsoft Windows CE (ADOCE) proporcionan un subconjunto de ADO para Windows CE. Al habilitar el acceso a bases de datos almacenadas de forma local en un dispositivo, ADOCE agrega nueva funcionalidad de base de datos al sistema operativo Windows CE y proporciona sincronización de datos a una base de datos de red.

Alarma: evento generado por un dispositivo o una función que señala la existencia de una condición anormal a través de un cambio discreto audible o visible, o ambas, que requiere su atención inmediata.

AIT: Automatización, Informática y Telecomunicaciones. Es la Dirección de Petróleos de Venezuela, S.A. que rige, provee y garantiza los servicios y soluciones integrales de tecnologías de Automatización, Informática y Telecomunicaciones (AIT) de la Corporación, por ello contribuye a mantener su continuidad operativa y a ejecutar los planes; innovando y actuando como agentes de transformación con responsabilidad social, económica y ambiental.

Ambiente de Edición: es donde se diseñan, editan y configuran los recursos gráficos, así como los parámetros necesarios para la recolección de datos de un proceso y su representación gráfica. Dentro de estos recursos se encuentran los gráficos que se corresponden con los procesos, los mímicos, la configuración de menús, alarmas, dispositivos de control y vías de comunicación.

Ambiente de Ejecución: es donde se actualizan los mímicos desarrollados en el ambiente de edición a partir de la recolección y actualización de las variables en tiempo real. Esto se logra teniendo en cuenta las correspondencias definidas entre el estado de las variables y los parámetros establecidos así como la configuración del ambiente de edición.

Canal: en comunicaciones, es el medio físico de transmisión de datos. Se encarga del transporte de la señal sobre la que viaja la información que intercambian el emisor y el receptor. Es definido desde el punto de vista telemático por sus propiedades físicas: naturaleza de la señal, velocidad de transmisión, capacidad de transmisión (ancho de banda), nivel de ruido que genera, longitud, etc.

GNOME: GNU **N**etwork **O**bject **M**odel **E**nvironment es un entorno de escritorio basado en las bibliotecas GTK diseñadas para GIMP para sistemas operativos de tipo Unix bajo tecnología X Window. Al igual que KDE, permite la interacción entre programas. Se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

GUI: Graphical User Interface (**Interfaz gráfica de usuario**) es un método para facilitar la interacción del usuario con el ordenador a través de la utilización de un conjunto de imágenes y objetos pictóricos (iconos, ventanas) además de texto.

Controlador Lógico Programable (PLC): dispositivos electrónicos usados en automatización industrial para realizar estrategias de control básicas. Por su robustez y características sencillas de control, están cercanas al proceso, permitiendo ejecutar las tareas básicas del control, aun cuando no tenga conexión a las capas superiores del control.

CORBA: Common Object Request Broker Architecture (arquitectura común de intermediarios en peticiones a objetos) es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.

HMI: Human Machine Interface (Interfaces hombre-máquina).

Información general del punto: conjunto de atributos comunes a todos los tipos de punto (analógico, digital, vector, texto, acumulador, diagnóstico).

KDE: K Desktop **E**nvironment es otro entorno de escritorio de gran popularidad, con gestor de ventanas propio y barra de tareas. Está programado en el lenguaje C++ y se basa en las bibliotecas QT+, lo que le ha significado ser víctima de muchas críticas por parte de la comunidad GNU/Linux, ya que estas bibliotecas eran propiedad de una empresa comercial.

Despliegue: Es el área de configuración de un proyecto, donde se representan los elementos gráficos, sus animaciones en dependencia de los valores que registran sus variables, los gráficos de tendencia, toda la representación visual de un SCADA.

Dispositivo de campo: son los elementos físicos que miden, monitorean y, en algunos casos almacenan, los datos de las variables del proceso. Estos dispositivos no se conectan directamente al SCADA.

LDAP: siglas de Lighweight Directory Access Protocol, es un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP puede considerarse una base de datos (aunque su sistema de almacenamiento puede ser otro diferente) al que pueden realizarse consultas.

Multiplataforma: es un término utilizado frecuentemente en informática para indicar la capacidad o características de poder funcionar o mantener una interoperabilidad de forma similar en diferentes sistemas operativos o plataformas.

Nodo: Espacio real o subestracto en el que confluyen parte de las conexiones de otros espacios reales o abstractos que comparten sus mismas características y que a su vez también son nodos. Una arquitectura está compuesta por diversos nodos. Los nodos deben tener interconexiones o conexión con otros nodos.

ODBC: siglas de Open DataBase Connectivity, un estándar de acceso a Bases de Datos desarrollado por la Corporación Microsoft, el objetivo de ODBC es hacer posible el acceder a cualquier dato de cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos (DBMS por sus siglas en Inglés) almacene los datos, ODBC logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas.

OMG: Object Management Group (Grupo de Gestión de Objetos). Es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización NO lucrativa que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para tecnologías orientadas a objetos.

OPC: (OLE for Process Control) es un estándar de comunicación en el campo del control y supervisión de procesos. Este estándar permite que diferentes fuentes de datos envíen datos a un mismo servidor OPC, al que a su vez podrán conectarse diferentes programas compatibles con dicho estándar. De este modo se elimina la necesidad de que todos los programas cuenten con drivers para dialogar con múltiples fuentes de datos, basta que tengan un driver OPC.

Proyecto: Se define como un proyecto a la configuración de un conjunto de recursos del SCADA que se orientan a supervisar una aplicación total. Los proyectos contienen sub-proyectos que contienen la solución a una sección de la aplicación total, estos sub-proyectos pueden representarlos servicios que debe ejecutar determinado nodo del sistema, por ejemplo la aplicación específica que se debe mostrar en

una consola, o la configuración de un servidor de recolección. Cada sub-proyecto contiene la configuración de un nodo, definiendo los servicios y recursos que le corresponda a dicho nodo, de recolección, visualización, etc.

Punto: los puntos se refieren a las direcciones de memoria que se configuran en el SCADA y que referencian a una variable capturada por un dispositivo de campo, o una variable del proceso. Pueden ser de tipo analógico, digital, texto y vector.

Recurso: Son todos los elementos configurables en el sistema como variables, alarmas, elementos gráficos, despliegues, recetas.

Scripts: Conjunto de líneas de código que permite interactuar con los objetos del proyecto, llámense objetos gráficos, drivers, alarmas, etc, para dotar al sistema de funcionalidades avanzadas que no pueden lograse con recursos, eventos y comandos que propicia el sistema. Los scripts pueden ser de tres tipos: el primer tipo son los denominados Script Básicos, dado que pueden ser invocados desde cualquier contexto del proyecto, desde menús desde otros scripts, etc; los segundos son denominados scripts de objetos, estos scripts pueden ser ejecutados a partir de la ocurrencia de algún evento sobre objetos gráficos; a los últimos se les denomina Expresiones, los cuales no son más que ecuaciones que pueden ser evaluadas y que pueden contener, funciones matemáticas y variables de proceso asociadas, estos comúnmente son asignados a propiedades de los objetos que pueden ser animadas, que son en función de expresiones(no sólo de variables).

Sistemas de Adquisición de Datos y Control Supervisorio (SCADA): Aplicación de software especialmente diseñada para funcionar sobre computadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la pantalla del operador, proveyendo toda la información asociada al proceso.

SMS: El servicio de mensajes cortos o SMS (Short Message Service) es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos (también conocidos como mensajes de texto, o más coloquialmente, textos o incluso txts o msjs) entre teléfonos móviles, teléfonos fijos y otros dispositivos de mano.

Subcanal: es parte constitutiva de un canal de transmisión, en el SCADA, es utilizado para agrupar lógicamente los dispositivos de campo que utilizan el mismo protocolo y que por consiguiente son accedidos por un mismo manejador.

Tendencia: en un sentido general, es un patrón de comportamiento de los elementos de un entorno particular durante un período de tiempo. Es representada en un gráfico bidimensional donde se refleja el comportamiento histórico de una variable adquirida en una de las coordenadas, y el tiempo en la otra.

Unidad Terminal Remota (RTU): dispositivos basados en microprocesadores, el cual permite obtener señales independientes de los procesos y enviar la información a un sitio remoto. Las RTU en los tiempos han sido desplazadas por los PLC quienes han fortalecido sus facilidades de comunicación a través de protocolos para sistemas de control (MODBUS, DNP3, IEC-101, etc.)

Widget (Objeto gráfico): Objeto gráfico que puede contener gráficos vectoriales o imágenes raster, (imágenes de mapas de bits) y presenta un conjunto de propiedades que pueden ser editadas y asociadas a puntos (variables) del SCADA o a expresiones que contengan variables del sistema. Por medio de los objetos gráficos se pueden crear animaciones en función de los valores de los puntos asociados. Los widgets pueden agruparse para formar nuevos objetos gráficos más complejos. Un ejemplo de widget puede ser un contenedor de textos.(MOYA 2004)