UNIVERSIDAD DE LAS CIENCIAS INFORMATICAS

Facultad 5

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Mecanismo de integración de scripting para el SCADA GALBA.

Autor: Gonzalo Martín Rodríguez.

Tutor: Ing. Ignais La Paz Trujillo

Cotutor: Ing. Arianna Gómez Vargas

“La Habana, junio 2015”

“Año 57 de la Revolución”

Declaración de autoría

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

Para que así conste firma la presente a los ___ días del mes de _____ del año 2015.

_____

Firma del autor:

Gonzalo Martín Rodríguez

_____

Firma del cotutor:

Ing. Arianna Gómez Vargas

_____

Firma del tutor:

Ing. Ignais La Paz Trujillo

Datos de contacto:

Tutor: Ignais La Paz Trujillo

Categoría Científica: Ingeniero

Especialidad: Ingeniero en Ciencias Informáticas

Correo Electrónico: ilapaz@uci.cu

Categoría docente: Trabajador

Años de experiencia: 8

Años de graduado: 8

Co-Tutor: Arianna Gómez Vargas

Categoría Científica: Ingeniero

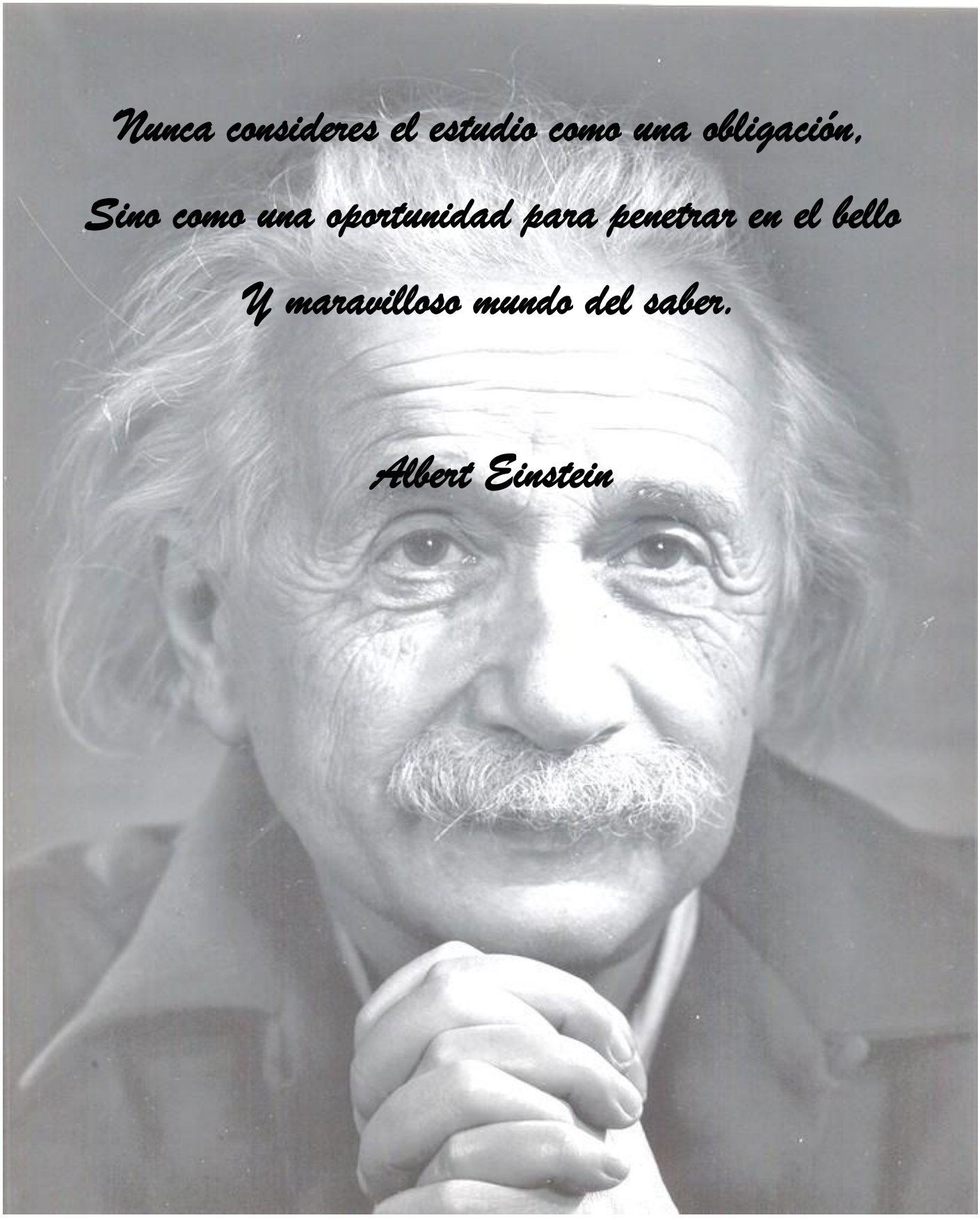
Especialidad: Ingeniero en Ciencias Informáticas

Correo Electrónico: agomezv@uci.cu

Categoría docente: Trabajador

Años de experiencia: 3

Años de graduado: 2


Nunca consideres el estudio como una obligación,
Sino como una oportunidad para penetrar en el bello
Y maravilloso mundo del saber.
Albert Einstein

Agradecimientos

Agradecer profundamente a mis padres, por inspirarme a recorrer este camino y acompañarme en su travesía cada minuto.

A mi hermana Yanelin por brindarme su apoyo siempre que lo necesité y su cariño incondicional.

A mis tutores por apoyarme y confiar en mí cuando más lo necesité.

A mi grupo de aula por todos los momentos felices que allí vivimos y de forma general a todos los amigos que han contribuido a que este logro fuera posible.

Dedicatoria

A mis padres, a mi hermana y a mi familia en general por ser el pilar fundamental en todo lo que soy.

A mi madre Rubidalis:

Por haberme apoyado en todo momento, por sus consejos, sus valores, por la motivación constante que me ha permitido ser una persona de bien, pero más que nada, por su amor.

A mi padre Gonzalo:

Por los ejemplos de perseverancia y constancia que lo caracterizan y que me ha infundado siempre a lo largo de la vida, por el valor mostrado para seguir adelante y por su amor.

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se encuentra la Facultad 5, que cuenta con varios centros productivos como es el Centro de Informática Industrial (CEDIN). Dicho centro, especializado en soluciones aplicadas al área de la automatización industrial, cuenta dentro de sus productos con un Sistema de Supervisión, Control y Adquisición de Datos (SCADA) desarrollado en conjunto con la empresa Petróleos de Venezuela, Sociedad Anónima (PDVSA) que surge en el marco del Plan Siembra Petrolera (1). Este SCADA GALBA está compuesto por varios módulos como: Comunicación, Procesamiento, Interfaz Hombre-Máquina (HMI), Aplicaciones, Seguridad y Base de Datos Históricos. En algunos sistemas SCADA, se han incorporado lenguajes que permiten programar tareas que respondan a eventos del sistema, personalizando las funcionalidades del módulo HMI, logrando así desarrollar proyectos más complejos. El presente trabajo se basa en el desarrollo un mecanismo que permita elevar el nivel de funcionalidades en el proceso de configuración del módulo HMI permitiendo construir proyectos más completos y complejos. El desarrollo de la aplicación está sustentado por la metodología de desarrollo del software AUP, utilizando las técnicas de modelación establecidas por el Lenguaje Unificado de Modelado (UML) y el marco de trabajo para el diseño de interfaces gráficas Qt. Al mecanismo implementado se le realizaron un conjunto de pruebas de aceptación que validaron el cumplimiento de los objetivos trazados en la investigación y los requerimientos establecidos por el cliente. Como resultado de la integración del mecanismo al SCADA se obtuvo una aplicación que permitirá optimizar las configuraciones realizadas en el HMI del SCADA GALBA, disminuyendo la complejidad de realizar de forma visual las configuraciones en el editor.

Palabras clave: SCADA, HMI, Script, JavaScript, Despliegue.

Índice de contenido

Índice de Figuras

Índice de Tablas

Introducción

El avance científico-técnico ha permitido desarrollar diversos sistemas artificiales capaces de ejecutar procesos muy complejos en menor tiempo y con mayor eficiencia. La automatización industrial se ha convertido en un medio fundamental para mejorar el rendimiento de las funciones operacionales de una empresa o industria moderna. La obtención de datos en el momento y punto de origen, al integrarse al ciclo de procesamiento y control de las operaciones y al actualizar las bases de datos en forma automática, permite la toma de decisiones operacionales más eficientes a partir de la naturaleza de la empresa.

Un Sistema de Supervisión Control y de Adquisición de Datos (SCADA, por sus siglas en inglés) es una tecnología que permite obtener y procesar información de procesos industriales dispersos o lugares remotos inaccesibles, y transmitirla a un lugar para supervisión, control y procesamiento, normalmente una sala o centro de control. Un SCADA permite supervisar y controlar simultáneamente procesos e instalaciones distribuidos en grandes áreas, y generar un conjunto de información procesada como presentación de gráficos de tendencias e información histórica, informes de operación y programación de eventos, programas de mantenimiento preventivo, etc (2).

En la Universidad de las Ciencias Informáticas (UCI) se implementa un esquema estructurado de estudio-trabajo, en el cual se instruye a los estudiantes tanto en su formación docente como en su vida laboral, basados en la integración de procesos fundamentales como la formación, investigación y la producción en torno a una temática de un centro de desarrollo. Dentro de la UCI se cuenta con el Centro de Informática Industrial (CEDIN), el cual desarrolla el software SCADA Guardián del ALBA (GALBA). El GALBA está compuesto por varios módulos como: Comunicación, Procesamiento, Interfaz Hombre-Máquina (HMI), Aplicaciones, Seguridad y Base de Datos Históricos.

El HMI del GALBA se divide en dos entornos: Entorno de Configuración (EC) y Entorno de Visualización (EV). El EC permite al operador del sistema editar el ambiente de trabajo, configurar la seguridad del sistema, definir los parámetros de variables a supervisar así como el tipo de datos, alarmas, salva de información, además diseñar los despliegues y componentes gráficos. El EV es el encargado de visualizar la configuración realizada en el EC y permitir al operador interactuar con el sistema, supervisar alarmas y puntos, analizar información a través de tendencias y reportes de estado lo cual brinda una mejor calidad en el control de los procesos.

En ocasiones las tareas de configuración en el EC se tornan complejas para los operadores por el alto nivel de detalle del proceso a visualizar en los despliegues y la ausencia de un mecanismo que permita la configuración de una manera más sencilla y rápida. El proceso de configuración en el EC resulta

engorroso, dado que la modificación simultánea de las propiedades de los objetos dentro de un despliegue debe realizarse paso a paso, o seleccionando uno a uno, para luego modificar sus propiedades en la paleta de componentes. Dicha tarea acarrea una pérdida de tiempo, que puede ser considerable, en dependencia de la cantidad de elementos a modificar. Por otra parte, muchas veces es necesario tener el mismo despliegue en diferentes ubicaciones, hasta la actualidad, el operador se ve obligado a repetir el mismo proceso de configuración en cada terminal, ya que no existe un mecanismo que permite crear plantillas de despliegues que puedan ser almacenadas y distribuidas en diferentes computadoras en dependencia de la necesidad que se tenga.

Lo antes expuesto provoca que a los operadores se les dificulten las tareas de modificación de los objetos gráficos utilizados en el diseño, lo que trae como consecuencias demoras en el proceso de diseño de despliegues y que se puedan introducir errores humanos debido a la complejidad de la tarea.

También resulta importante determinar el comportamiento del sistema ante el manejo de grandes volúmenes de objetos gráficos en los despliegues, con el objetivo de prevenir posibles fallas del sistema en tiempo de ejecución. Además no existe un vínculo con alguna aplicación que facilitaría dicho proceso para aquellos que lo realizan. Por lo antes planteado se puede caracterizar hoy la configuración en el HMI como un proceso complejo que limita a los operadores de realizar configuraciones completas y con el alto nivel de detalle que caracteriza a un SCADA actual.

Por lo antes expuesto se plantea como **situación problemática:**

- El HMI del SCADA no permite crear un conjunto de objetos y modificarlos todos al mismo tiempo, ni transferirlo a otras máquinas.
- No se cuenta con un mecanismo que permita crear plantillas de despliegues complejos y transferir despliegues complejos a varias máquinas.

Se enuncia como **problema de investigación:** ¿Cómo contribuir a la flexibilidad del proceso de configuración de objetos gráficos en el subsistema de ejecución del módulo HMI?

El **objeto de estudio** del trabajo investigativo es el proceso de configuración en la Interfaz Hombre Máquina para los sistemas de supervisión y control.

Por lo que se define como **objetivo general:** Desarrollar un Mecanismo de integración de scripting para el SCADA GALBA.

Para dar cumplimiento al objetivo general se plantean los siguientes **objetivos específicos:**

- Analizar tecnologías que contribuyan con la integración de editor al HMI de SCADA.

- Desarrollar un motor de ejecución para los script de los objetos gráficos.

- Visualizar el funcionamiento del editor.

Se define como **campo de acción** del trabajo investigativo es el proceso de configuración en la Interfaz Hombre Máquina en el SCADA GALBA.

Para dar cumplimiento a los objetivos específicos planteados, se conforman las siguientes **tareas de investigación:**

- Búsqueda y estudio de la bibliografía referente a las funcionalidades de los despliegues en el HMI SCADA.
- Estudio del mecanismo de edición y ejecución del SCADA.
- Estudio de las especificaciones técnicas sobre los despliegues del SCADA.
- Estudio del motor de ejecución de script para los objetos gráficos de los despliegues.
- Solución de no conformidades.

Para dar cumplimiento a las tareas planteadas se hace necesario el empleo de los siguientes **métodos científicos**.

Analítico-Sintético se aplica en la presente investigación para a partir de los estudios e investigaciones hechas y de la información recogida acerca de aplicaciones similares, llegar a conclusiones sobre el problema estudiado y presentar posibles vías para solucionarlo. El **Histórico-Lógico** para explicar los antecedentes de las herramientas para el trabajo con secuencias de comandos en los sistemas SCADA y sus tendencias actuales, así como la necesidad, la importancia y los hitos que surgen del desarrollo de una herramienta como esta. El método **Modelación** para el esbozo de los diferentes diagramas y modelos generados durante el proceso de desarrollo de un componente, que contribuya con la descripción de los objetos gráficos presentes en SCADA.

Métodos empíricos

Consultas de fuentes de información: Se utiliza este método para las consultas de fuentes bibliográficas durante la investigación.

El presente trabajo se estructura en tres capítulos:

- **Capítulo 1:** Se plantea la fundamentación teórica de la investigación. Se realiza la búsqueda de información acerca de las características, funciones y componentes de un SCADA. Por otra parte

se seleccionan las herramientas a utilizar, así como la metodología y tecnologías para el desarrollo de editor de script.

- **Capítulo 2:** Se plantea el análisis y diseño del mecanismo scripting. Se realizan un conjunto de procedimientos establecidos por la metodología seleccionada para guiar el desarrollo de la solución, tales como: el diagrama modelo dominio, diagrama de clases, diagrama de caso uso, se levantan los requisitos funcionales y no funcionales del editor. Se escoge la arquitectura para el desarrollo del mecanismo y se exponen los patrones de diseños utilizados.
- **Capítulo 3:** Se realiza la implementación y prueba del mecanismo scripting se implementan los requisitos del mecanismo planteados en el capítulo 2 y se le realizan las pruebas para comprobar posibles no conformidades en el editor y solucionarlas permitiendo así un funcionamiento correcto del mecanismo.

Capítulo 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se realiza la fundamentación teórica de la investigación, abordando temas de relevancia para la misma como los conceptos y términos asociados al dominio del problema. Se realiza un estudio de las diferentes herramientas que son útiles para la integración del editor de scripting al SCADA. Se realiza la selección de herramientas, metodología y tecnología a utilizar en el ciclo de vida del software. De igual manera, se hace una revisión y selección de lenguajes de programación propuestos para el desarrollo del sistema.

1.2 SCADA (3)

Un SCADA es una aplicación software de control de la producción, que se comunica con los dispositivos de campo y controla el proceso de forma automática desde la pantalla del ordenador.

Dentro de las funciones principales que contiene pueden mencionarse las siguientes:

1. **Adquisición de datos:** para recoger, procesar y almacenar la información recibida.
2. **Supervisión**: para observar desde un monitor la evolución de las variables de control.
3. **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.

Funciones más específicas:

1. **Transmisión:** De información con dispositivos de campo y otros PC.
2. **Base de datos:** Gestión de datos con bajos tiempos de acceso.
3. **Presentación:** Representación gráfica de los datos. Interfaz del Operador o HMI (Human Machine interface).
4. **Explotación:** De los datos adquiridos para gestión de la calidad, control estadístico, gestión de la producción y gestión administrativa y financiera.

Ejemplo de Interfaz de Operario:

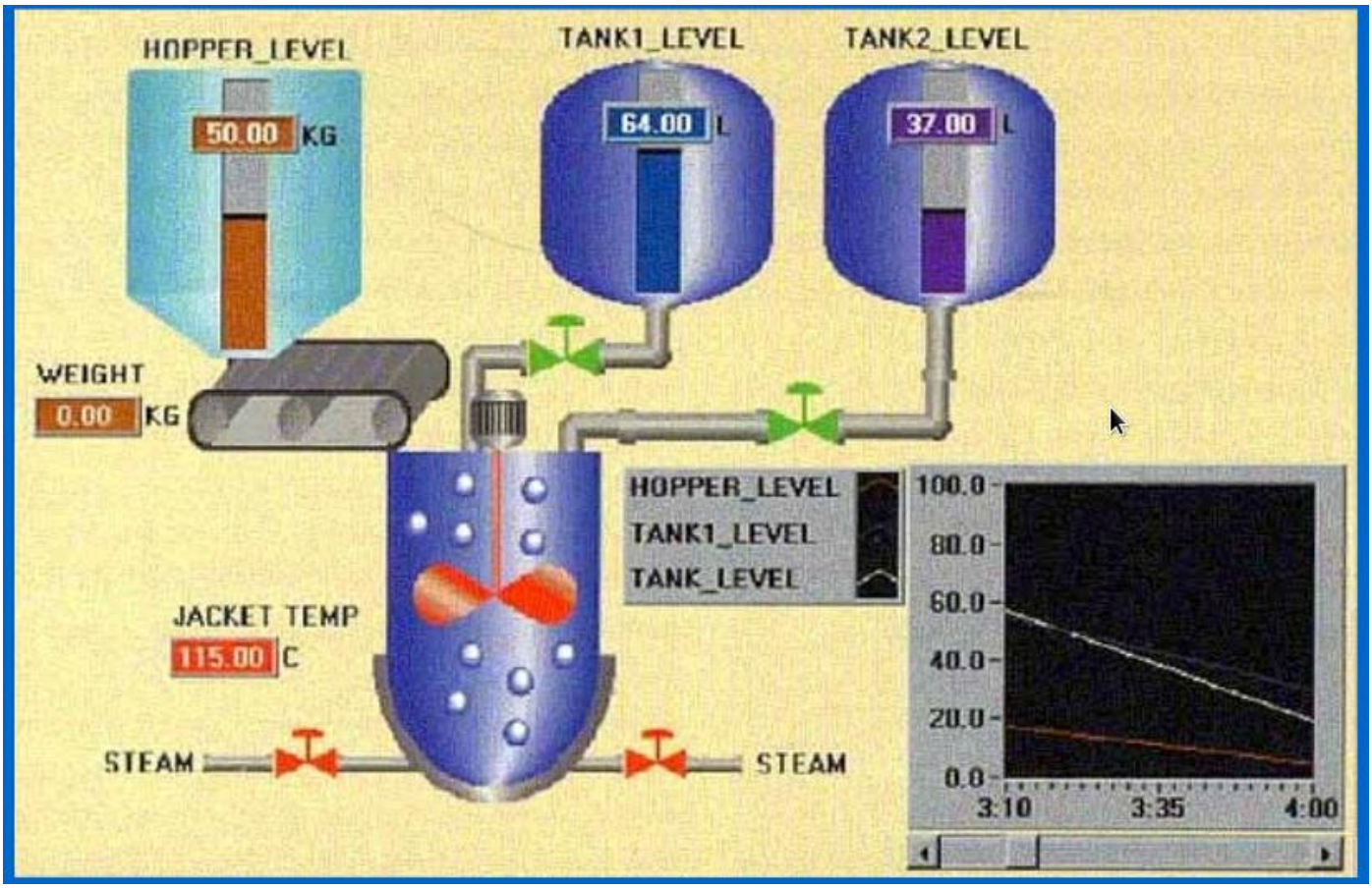


Figura 1: Interfaz de operario de un SCADA

Un SCADA tiene una composición modular, donde cada una de sus partes se encarga de una o varias tareas dentro del sistema y a la vez colaboran entre sí para cerrar el círculo de adquisición, control y supervisión del proceso. A continuación se hará referencia a los componentes de un SCADA, específicamente se centrará mayor atención en el módulo Interfaz Hombre Máquina o HMI como también se le conoce por sus siglas en inglés.

1.3 Componentes de un sistema SCADA (4)

- **Configuración:** permite definir el entorno de trabajo del SCADA, adaptándolo a la aplicación particular que se desea desarrollar.
- **Interfaz gráfica del operador:** proporciona al operador las funciones de control y supervisión de la planta. El proceso se representa mediante sinópticos gráficos almacenados en el ordenador de proceso y generados desde el editor incorporado en el SCADA o importados desde otra aplicación durante la configuración del paquete.
- **Módulo de proceso:** ejecuta las acciones de mando reprogramadas a partir de los valores actuales de variables leídas.

- **Gestión y archivo de datos:** almacenamiento y procesado ordenado de datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos.
- **Comunicaciones:** transferencia de información entre la planta y la arquitectura hardware que soporta el SCADA, y también entre ésta y el resto de elementos informáticos de gestión.

1.3.1 Módulo Interfaz Hombre-Máquina

HMI **(***Human Machine Interface, por sus siglas en inglés*), es un dispositivo o sistema que actúa como interfaz entre la persona (operador) y la máquina (proceso).Tradicionalmente consistían en paneles compuestos por indicadores y comandos, tales como luces pilotos, indicadores digitales y análogos, etc... Cuando se hace uso de una interfaz gráfica de usuario, las prestaciones del sistema aumentan considerablemente. Esto es fácilmente apreciable pues los usuarios poseen un número mayor de posibilidades, como por ejemplo: la visualización del comportamiento de las variables que representan los procesos, el uso de gráficas de tendencia, una mayor facilidad en la realización del control, la generación de reportes también pasa un plano más importante, pues con el uso de los mismos, los datos son presentados al operador de una forma más organizada. Lo antes descrito, refleja que los HMI permiten una interacción más dinámica de los operadores con los procesos supervisados (5).

Funcionalidades generales de un HMI:

- Permitir una comunicación con dispositivos de campo.
- Actualizar una base de datos con las variables del proceso.
- Visualizar las variables mediante pantallas con objetos animados.
- Permitir que el operador pueda enviar señales al proceso, mediante botones,
- Controles encendido y apagado, o ajustes continuos con el mouse o el teclado.
- Supervisar niveles de alarma y alertar/actuar en caso de que las variables
- excedan los límites normales.
- Almacenar los valores de las variables para un análisis estadístico y/o control.
- Controlar en forma limitada ciertas variables de proceso.

1.4 SCADA GALBA

Desde el año 2006 se comienza a desarrollar en la UCI un software SCADA en convenio con la Gerencia Automatización Informática y Telecomunicaciones (AIT) de la empresa Petróleos de Venezuela S.A (PDVSA), conocido en sus inicios como SCADA Nacional o SCADA PDVSA y a partir del 2008 en que fue presentado en la “Cumbre del ALBA” se comienza a conocer como SCADA GALBA, donde se proyectó una futura instalación del sistema en los países integrantes de esta organización. El software, que se da

como solución es realizado en cooperación de distintos equipos de desarrollo de la UCI, DST-AIT PDVSA, Centro de Desarrollo de Automática Integral (CEDAI), Universidad Central “Marta Abreu” de Las Villas (UCLV), Instituto Superior Minero Metalúrgico de Moa (ISMMM), Universidad de los Andes (ULA), DBAccess, IntelCom y la empresa de Ingeniería de Software y Calidad Aplicada (Isca) (6).

1.4.1 HMI del SCADA GALBA

El módulo HMI cuenta con 2 ambientes el ambiente configuración o edición y el ambiente de ejecución. En el ambiente de configuración se pueden crear nodos y agregarle varios módulos, entre ellos los módulos de históricos, los de bases de Datos en tiempo real, que incluyen canales, subcanales, dispositivos, y los distintos tipos de puntos y alarmas con las que cuenta el sistema, también permite configurar y diseñar los esquemáticos del sistema usando una serie de gráficos, controles y widgets que le permitan al usuario la supervisión y el control de los procesos industriales. Por otra parte se encuentra el ambiente de ejecución el cual es el encargado de posibilitar la interfaz hombre-máquina (IHM) para monitorear y supervisar los procesos del sistema previamente configurados. Se refiere a las características o componentes que estarán disponibles al usuario cuando éste inicie al sistema en el ambiente de operación. Se trabaja en el ambiente de configuración o edición para dar solución al Editor de Script ya que es donde se encuentran los componentes gráficos para su edición en este ambiente.

1.5 Herramientas de scripting para sistemas SCADA

Cuando se crea un sistema SCADA se necesita crear un HMI que permita ver de forma animada y amigable el funcionamiento de los equipos y procesos, para poder personalizar estas funciones se necesita programarlas a través de una lógica operacional o comandos. Para realizar estas nuevas tareas se integran a los Sistemas SCADA herramientas de gestión script que permiten facilitarle el trabajo a los operadores.

A continuación, el autor de la presente investigación, considera oportuno realizar un estudio del arte, con el objetivo de identificar posibles soluciones a la problemática planteada.

1.5.1 HMI Script Editor

QuickScript es el lenguaje scripting para HMI de InTouch. Puede ser utilizado para construir aplicaciones más robustas. Posee siete tipos de script definidos y muchas funciones integradas de secuencias de comandos disponibles. Los siete tipos de script se definen según las causas por las que se van a ejecutar. Por ejemplo, los script de aplicación se ejecutan cuando inicia la aplicación, detiene o continúa en funcionamiento. Las funciones de script incorporadas incluyen funciones matemáticas, funciones

trigonométricas, funciones de cadenas, y otras. El uso de estas funciones le permite al usuario ahorrar tiempo en el desarrollo de la aplicación. Los scripts de InTouch pueden incluir vinculación y empotramientos de objetos (OLE) y controles ActiveX. Puede utilizar sentencias condicionales, bucles y variables locales en el lenguaje de scripting para crear efectos complejos en su aplicación (7).

Los tipos de script que define InTouch para su aplicación son:

Script de Aplicación: Se ejecuta de forma continua mientras que WindowViewer se está ejecutando, o una vez que WindowViewer se inicia o se detiene.

Script de Ventana: Se ejecuta periódicamente cuando en InTouch una ventana está abierta o una vez que una ventana se abre o se cierra.

Script de Tecla: Se ejecuta una vez o periódicamente cuando una tecla o una combinación de ellas es presionada o liberada.

Script de Condición: Se ejecuta una vez o periódicamente cuando cierta condición se cumple o no.

Script de Cambio de Datos: Se ejecuta una vez cuando un valor de una etiqueta o de una expresión varía.

Script de Acción: Se ejecuta una vez o periódicamente cuando un operador hace click en un objeto gráfico del HMI.

Script de evento de ActiveX: Se ejecuta una vez que un evento de ActiveX ocurre, como click en el control de ActiveX.

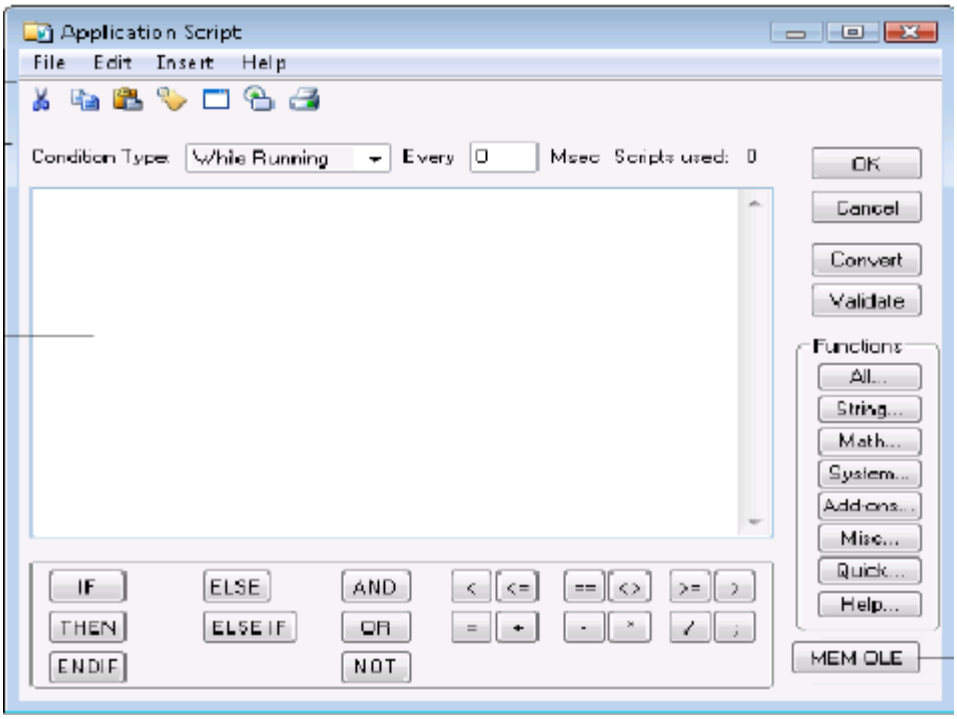


Figura 2: HMI Script Editor de InTouch

Este HMI Script Editor de InTouch es un editor de script potente pero trabaja con funciones y ventanas de Windows que permite desarrollar un mecanismo como el que se acaba de analizar.

1.5.2 Global Script.

El Global Script es un compilador de C incorporado en *WinCC* permite programar en lenguaje JavaScript y realizar compilaciones de código objeto (con un formato un tanto especial, ya que no lo deja en objeto) sin necesidad de salir del propio entorno. Este código generado puede ser añadido directamente al propio intervalo de tiempo en el que un programa de computadora se ejecuta en un sistema operativo (*runtime*) y generarse cada cierto tiempo, o posteriormente asociarse a un evento de un objeto. Si desde el editor del Diseñador Gráfico se selecciona realizar una acción en C, automáticamente se activa el Global Script para permitirnos editar dicha acción en el compilador de C (8).

Dentro del compilador de Global Script existen cuatro tipos de clases o categorías:

Project functions: Son aquellas funciones que deben ser llamadas desde otras partes del programa para devolver valores después de realizar alguna operación en C.

Standard functions: son las funciones estándar del propio *WinCC*, que no residen en el proyecto, sino en el subdirectorio Librería de enlace Dinámico (aplib) de *WinCC*. Estas funciones estándares se pueden modificar, pero dichas modificaciones son permanentes hasta que no se reinstale el *WinCC*.

Internal functions: Funciones que realizan acciones predeterminadas, como son proporcionar valor de una variable, o asignar el valor a una variable.

Actions: Una acción es una subrutina que no se ejecuta cuando es llamada por un evento, sino cuando se desencadena una acción, ya sea por el tiempo o por un cambio de valor de alguna variable.

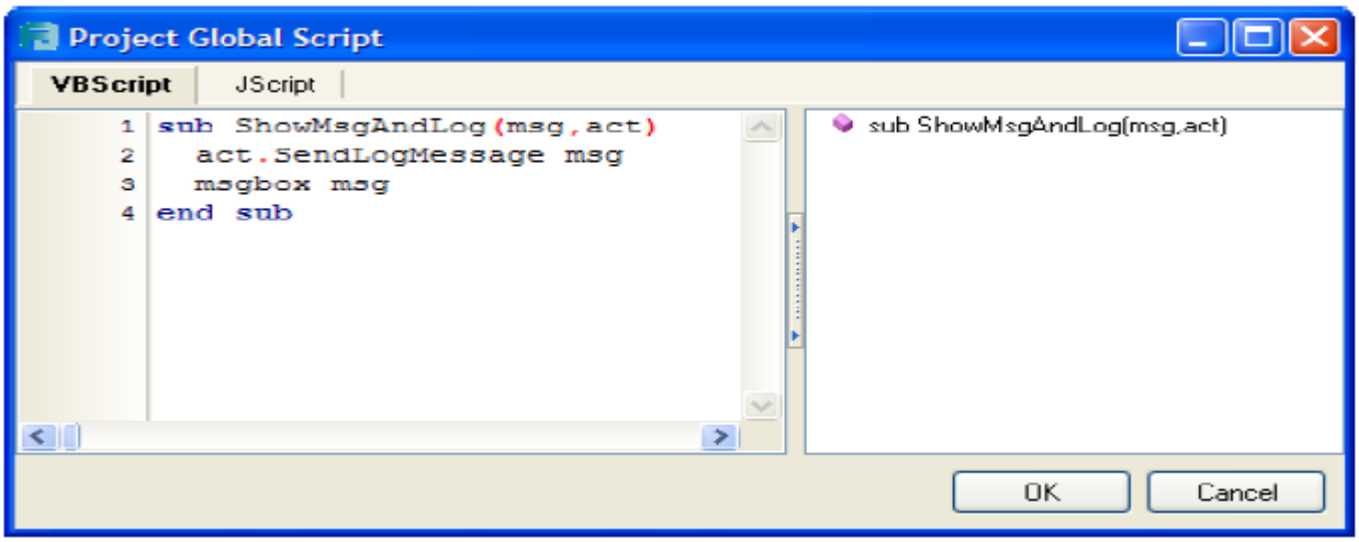


Figura 3 Project Global Script

El Global Script es otra Herramienta que se analiza se logra ver que es una herramienta con gran capacidad de realizar tareas como las funciones estándar de la librería aplib de *WinCC* pero todo utiliza en Windows y no permite el desarrollo en otro sistema operativo como Linux.

1.5.3 CX-Supervisor Script Editor.

Los script, controlan las acciones de los objetos, páginas o proyectos; pueden ser creados o actualizados utilizando el cuadro de diálogo del Editor de Script de CX-Supervisor. El editor de script permite crear o modificar las operaciones que se llevan a cabo en las animaciones mediante una secuencia de comandos. Está codificado por colores para ayudar a mostrar la sintaxis correcta con las palabras claves y los diferentes tipos de objetos se muestran en colores diferentes. Al crear una secuencia de comandos debe elegir una acción, función, etc. Además de tener un menú contextual del cual puede requerir información adicional (9).

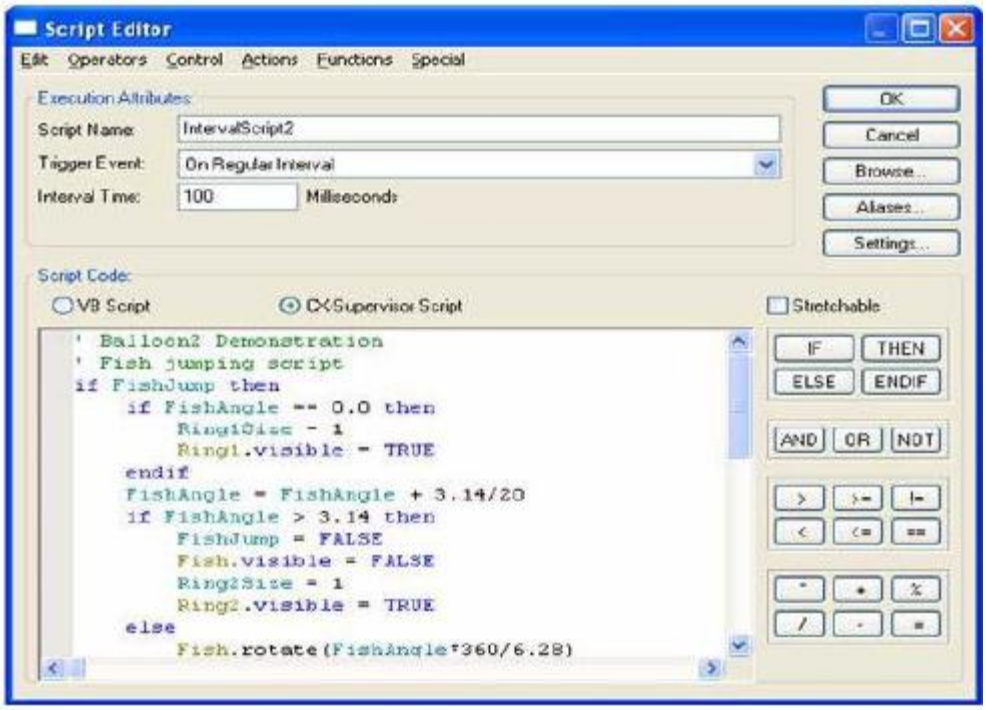


Figura 4: CX-Supervisor Script Editor

El CX-Supervisor Script Editor permite de programar en el lenguaje script propio de CX-Supervisor o en Visual Basic Script pero se necesita otro lenguaje para hacer más fácil a los operadores el uso de mecanismo.

1.6 Resumen del análisis de las herramientas para la incorporación de scripting en los SCADA.

En total se analizaron 3 herramientas para la incorporación de scripting, a continuación una tabla muestra una serie de características observadas en cada una:

Herramienta	Software propietario	Lenguaje de programación	Multiplataforma
HMI Script Editor	Sí	QuickScript	No
Global Script	Sí	JavaScript	No
CX-Supervisor Script Editor	Sí	CX-Supervisor, Visual Basic	No

las características antes mencionadas hacen que el uso de algunas de ellas para adaptarla a la solución sea imposible, partiendo desde el punto de vista que pertenecen a software propietarios, además presentan características particulares para cada SCADA. A pesar de ello contribuyeron como guía para desarrollar una nueva herramienta que permita incorporarse al SCADA-GALBA con sus características y funciones específicas.

1.7 Selección del marco teórico metodológico para el desarrollo de la solución

El SCADA GALBA al ser uno de los proyectos desarrollado por el Centro de Informática Industrial, cuenta con marco tecnológico definido, como la investigación forma parte de dicho sistema, se heredan todas las tecnologías y metodologías para su desarrollo. A continuación se exponen cada una de ellas con sus características fundamentales.

1.7.2 Metodología de desarrollo: AUP versión UCI

Se escoge la metodología de desarrollo del software variación de AUP para la UCI ya que no existe máuna metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas (10).

Descripción de las Fases Variación AUP-UCI

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, la cual se califica como Ejecución y se agrega la fase de Cierre. Para una mayor comprensión ver Anexos.

Descripción de las disciplinas AUP-UCI

AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 8 disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto). Para una mayor comprensión ver Anexos.

Productos de trabajos

Entre las técnicas ágiles que utiliza AUP se encuentra el Modelado ágil, se hará uso de esta técnica para los proyectos que necesiten por sus características encapsular sus requisitos funcionales en **Historias de usuarios** o en **Descripción de requisitos por procesos**. La otra forma de encapsular los requisitos se mantiene por **Casos de Uso (CU)** la cual se utilizó en el desarrollo del mecanismo de scripting.

1.7.3 Entorno de desarrollo: sistema operativo: GNU/Linux

Se selecciona este sistema operativo porque provee alta calidad tecnológica y gran seguridad Debian GNU/Linux es un sistema operativo libre, desarrollado por más de mil voluntarios alrededor del mundo, que colaboran a través de Internet. La dedicación de Debian al software libre, su base de voluntarios, su naturaleza no comercial y su modelo de desarrollo abierto la distingue de otras distribuciones del sistema operativo GNU. Lo que más distingue a Debian de otras distribuciones es su sistema de gestión de paquetes. Estas herramientas otorgan al administrador de un sistema Debian total control sobre los paquetes instalados, operativo incluyendo la capacidad de instalar un sólo paquete o actualizar el sistema por completo. También es posible proteger paquetes individualmente de forma que no se actualicen. Incluso puede indicar al sistema de gestión de paquetes qué programas ha compilado usted mismo y qué dependencias cumplen.
Proporciona un sistema operativo actualizado y estable para el usuario promedio, con un fuerte enfoque en la facilidad de uso y de instalación del sistema. Al igual que otras distribuciones se compone de múltiples paquetes de software normalmente distribuidos bajo una licencia libre o de código abierto. (11)

1.7.4 Lenguaje de programación para la implementación de la solución: C++

C++ es un lenguaje de programación diseñado a mediados de los años 80 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitieran la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma. (12)
El lenguaje utilizado es **C++** para la implementación del mecanismo scripting ya que es el lenguaje utilizado en el desarrollo del sistema SCADA GALBA.

1.7.5 Marco de trabajo gráfico: Qt

Qt Creator es un entorno de desarrollo integrado para la creación de aplicaciones con el Framework Qt. Qt está diseñado para desarrollar aplicaciones e interfaces de usuario una vez y desplegarlas en múltiples sistemas operativos tanto móviles como de escritorio.

Qt Creator proporciona herramientas para desarrollar sus tareas a lo largo de todo el ciclo de vida desarrollo de las aplicaciones.

Las características clave de Qt Creator permiten a los programadores realizar las siguientes tareas:

- Empezar a desarrollar aplicaciones con Qt rápida y fácilmente con el asistente de proyectos, y acceder rápidamente a proyectos recientes y sesiones.
- Diseñar aplicaciones de interfaz de usuario basadas en widgets Qt con el editor integrado, Qt Designer.
- Desarrollar aplicaciones con el editor de código avanzado de C++ que provee nuevas y poderosas características para el completamiento de código y viendo el esquema de archivos (que es, la jerarquía de símbolos de un archivo)
- Compilar, correr y desarrollar proyectos Qt que se dirigen a múltiples plataformas de escritorio y móviles, como Microsoft Windows, Mac OS X, Linux, Symbian, MeeGo, y Maemo (13).

1.7.6 Lenguaje de modelado: UML

Cualquier rama de ingeniería o arquitectura ha encontrado útil desde hace mucho tiempo la representación de los diseños de forma gráfica. Desde los inicios de la informática se han estado utilizando distintas formas de representar los diseños de una forma más bien personal o con algún modelo gráfico. La falta de estandarización en la manera de representar gráficamente un modelo impedía que los diseños gráficos realizados se pudieran compartir fácilmente entre distintos diseñadores. (14)

El Lenguaje utilizado (UML) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Está compuesto por varios elementos gráficos que se combinan mediante reglas para conformar los diferentes diagramas. Permite además la modelación de sistemas con tecnología orientada a objetos. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. UML tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue.

1.7.7 Herramienta CASE: Visual Paradigm

Las herramientas CASE se han venido ampliando y desarrollando, existe una gran variedad de estas con características específicas, a continuación se describe la utilizada en el desarrollo del mecanismo scripting (15).

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. También proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Presenta licencia gratuita y comercial. Es fácil de instalar y actualizar y compatible entre ediciones.

- Características principales:
- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS y Subversion (control de versiones).
- Ingeniería de ida y vuelta.
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas EJB - Visualización de sistemas EJB.
- Generación de código y despliegue de EJB - Generación de beans para el desarrollo y despliegue de aplicaciones.
- Diagramas de flujo de datos.
- Soporte ORM - Generación de objetos Java desde la base de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Importación y exportación de ficheros XMI.
- Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de Microsoft Visio.

- Editor de figuras.

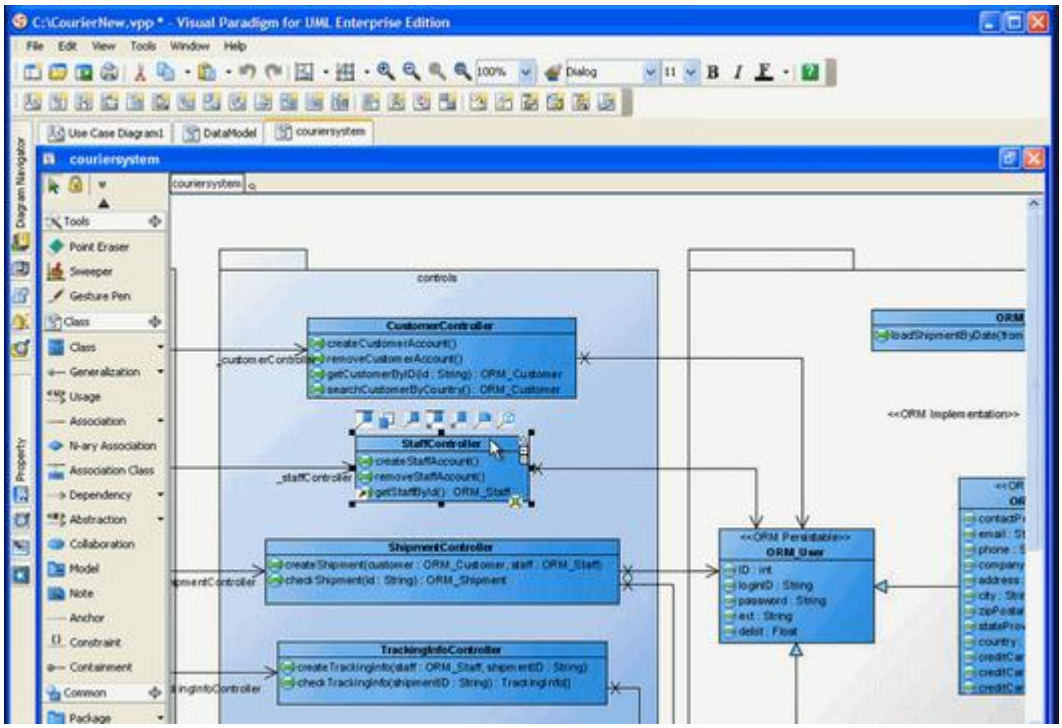


Figura 5: Herramienta Case Visual Paradigm

1.7.8 Lenguaje de programación Script para la secuencia de comandos

Los lenguajes script permiten a las aplicaciones una mayor flexibilidad sobre otros lenguajes de programación, así como una gestión de memoria automática. Dado el estudio realizado sobre los lenguajes script más utilizados en sistemas SCADA, se profundiza el estudio de ellos por las características antes mencionadas y se define para la utilización de JavaScript como lenguaje base para la generación de código script por ser un lenguaje multiplataforma, como también permite la programación orientada a objetos y posibilitando una integración al SCADA GALBA más sencilla y permitiendo un ahorro de tiempo para los operadores del sistema.

A continuación se realiza un estudio para determinar cuáles son las técnicas que se utilizan a nivel mundial para la incorporación de scripting en diversos sistemas SCADA.

Un script (cuya traducción literal es “guión”) o archivo de órdenes o de procesamiento por lotes, es un programa usualmente simple, que por lo general se almacena en un archivo de texto plano. Los script son casi siempre interpretados, pero no todo programa interpretado es considerado un script. El uso habitual de los scripts es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario. Por este uso frecuente es que los *shells* sean a la vez intérpretes de este tipo de programa (16). La aplicación de la tecnología script en los sistemas SCADA permiten desarrollar aplicaciones potentes y complejas para que los operadores puedan observar con más detalles como

intermitencias en la pantalla, alarmas interactivas, movimientos de productos ya sea horizontal o vertical, flujos secuencias, etc, a continuación algunos ejemplos:

- ✓ **Visual Basic for Applications**

Visual Basic for Applications (VBA) es el lenguaje de programación incorporado en Microsoft Office. Es un lenguaje muy extendido y se ha convertido en un estándar de facto, que permite la integración de aplicaciones de terceros y la comunicación directa con cualquier aplicación de MSOffice y de cualquier aplicación compatible con VBA. Este lenguaje se ha extendido y ha sido aceptado por la mayoría de los usuarios. Simatic *WinCC* de SIEMENS es un ejemplo de ello al integrar VBA, en el *WinCC* Graphics Designer. Por otro lado GENESIS32 de Iconics no solo integra esta característica a su editor gráfico, sino que ofrece el módulo ScriptWorX32 que es una poderosa herramienta para crear y ejecutar programas en VBA, que pueden correr simultáneamente basados en intervalos de tiempos o de eventos. El uso de un lenguaje común también facilita la integración de objetos suministrados por terceros. Además permite interactuar directamente con las aplicaciones de Office (Access, Excel y Word) y con otros productos compatibles (17).

- ✓ **HMI Scripting Language**

El HMI Scripting Language utiliza la misma sintaxis y semántica que el Microsoft Visual Basic. Permite la interacción con otras aplicaciones de Windows, además de realizar múltiples acciones como la conversión de unidades, formato e impresión de reportes, la gestión de cuadros de diálogos y entradas de usuario, así como la alternación entre las pantallas del operador (18). Es un lenguaje potente que permite realizar múltiples acciones para el desarrollo del mecanismo pero sobre aplicaciones de Windows y el desarrollo del mecanismo es sobre una aplicación de Linux.

- ✓ **C++ Script.**

Es una biblioteca de C++ que proporciona tipos de secuencias de comando, dinámicos y programación dinámica en C++. El paradigma dinámico permite a C++ ser utilizado de una manera más sencilla, sin requerir el conocimiento de la gestión de memoria, clases, plantillas, los contenedores y los iteradores. Muchos problemas pueden ser expresados de una manera más sencilla utilizando los tipos dinámicos. Como lenguajes de secuencia de comandos, programas en C++ Script se compilan utilizando el estándar de C++, puede incrustar en el estándar código de C para lograr un alto rendimiento (19).

- ✓ **JavaScript.**

JavaScript (abreviado comúnmente "**JS**") es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

- ✓ Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.
- ✓ JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.
- ✓ Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).
- ✓ Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Selección Lenguaje de programación Script para la secuencia de comandos para la incorporación de scripting en el SCADA GALBA

En total se analizaron 4 lenguajes de programación y se elige el lenguaje JavaScript para la incorporación de scripting porque este es un lenguaje fácil de trabajar a la hora de que el operador decida interactuar con la configuración de los componentes gráficos, fácil de entender y además es imperativo, estructurado, dinámico, funcional, prototípico, etc.

1.9 Consideraciones Parciales

Los diferentes conceptos estudiados ayudaron a tener una visión más clara sobre los sistemas SCADA. El estudio que se realizó del SCADA GALBA contribuyó a esclarecer la estructura y funcionalidades que contiene, como también cuales son las situaciones problemáticas que dieron paso a la presente investigación científica. Otros de los aspectos abordados fueron los diferentes mecanismos de integración de scripting en algunos de los SCADA disponibles en la industria, definiéndose que el lenguaje a utilizar es JavaScript, ya que QT, ambiente en el que se desarrolla el GALBA, tiene incorporado el mecanismo QScripting el cual permite el intercambio de objetos entre JavaScript y C++, que en caso de utilizarse otro

lenguaje haría falta otro tipo de herramienta para integrar el mecanismo al módulo HMI. Por otra parte se realizó la selección de metodologías y tecnologías que soportaran el proceso de desarrollo de la solución.

Capítulo 2: ANÁLISIS Y DISEÑO DEL MECANISMO DE SCRIPTING

2.1 Introducción

El presente capítulo tiene como objetivo reflejar las actividades realizadas en el diseño del mecanismo de scripting. En el mismo se exponen los artefactos más importantes que describen el flujo normal de eventos que ocurren en el sistema, tales como especificaciones de requisitos que rigen el desarrollo de la solución, historias de usuarios, diagramas y las principales clases que componen el mecanismo, detallando la información del análisis y del diseño de la solución en cuestión.

2.2 Modelo de dominio

El modelo de dominio se crea con el fin de representar el vocabulario y los conceptos clave del dominio del problema. El modelo de dominio también identifica las relaciones entre todas las entidades comprendidas en el ámbito del dominio del problema, y comúnmente identifica sus atributos. Un modelo de dominio que encapsula los métodos dentro de las entidades se asocia más bien con modelos orientados a objetos. El modelo de dominio proporciona una visión estructural del dominio que puede ser complementado con otros puntos de vista dinámicos, como el modelo de casos de uso.

Una ventaja importante de un modelo de dominio es que describe y limita el alcance del dominio del problema. El modelo de dominio puede ser usado efectivamente para verificar y validar la comprensión del dominio del problema entre las diversas partes interesadas. Define un vocabulario y es útil como herramienta de comunicación. Puede añadir precisión y enfoque para la discusión entre el equipo de negocios, así como entre los equipos técnicos y de negocios. (20)

El modelo de dominio se representa a través de un diagrama de clases: Con asociaciones entre ellas, pero sin cualificar (sin adornos), se pueden incluir atributos en las clases (de forma conceptual).

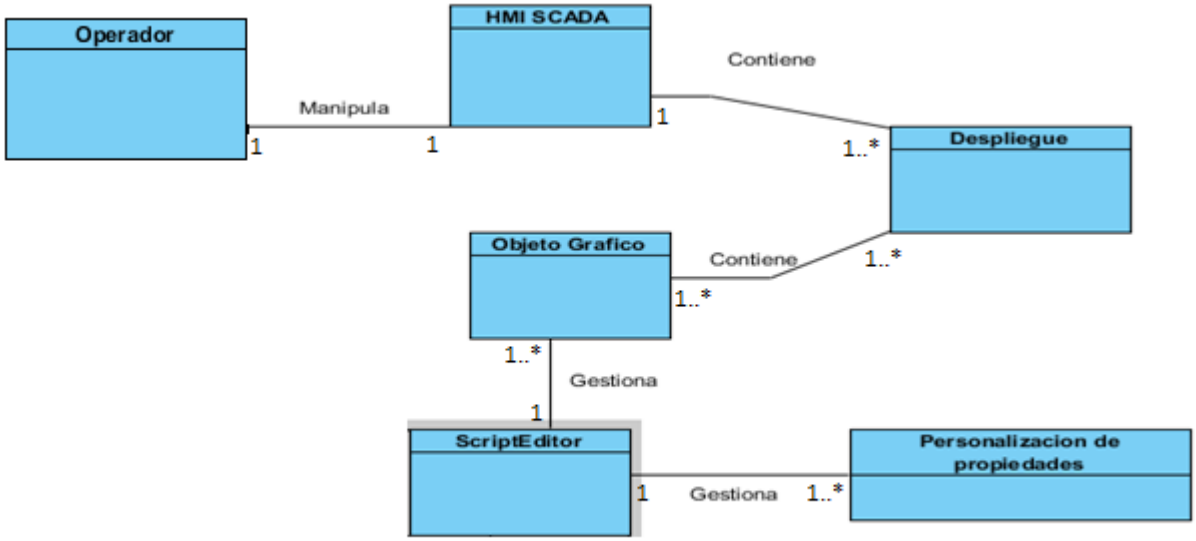


Figura 6: Diagrama de clases del dominio

En la modelación el operador accede al HMI SCADA donde se pueden crear uno o varios despliegues. Estos a su vez contienen Objetos Gráficos que utilizan el ScriptEditor para personalizar varias propiedades mediante código JavaScript. Además con dicha herramienta es posible crear objetos gráficos como rectángulos, líneas, gráficos de pastel, entre otros. Otras de sus funcionalidades son modificar propiedades, salvar el script para su posterior uso e importar scripts previamente definidos por el usuario.

2.3 Requisitos del mecanismo de integración

Un requisito funcional define el comportamiento interno del software, es decir, funcionalidades específicas de la aplicación; son complementados por los requisitos no funcionales, que se enfocan en el cambio en el diseño o en la implementación, describen la interacción entre el sistema y su ambiente independientemente de su implementación, el ambiente incluye usuario y cualquier otro sistema externo que interactúa con el sistema. (21)

2.3.1 Requisitos funcionales:

RF1 Gestionar un script.

- ✓ RF1.1 Adicionar script.
- ✓ RF1.2 Modificar script.
- ✓ RF1.3 Eliminar script.

RF2 Personalización de propiedades de los objetos gráficos.

- ✓ RF2.1 Personalización de propiedades.

RF3 Asociar variables a los componentes gráficos.

RF4 Crear objetos gráficos mediante scripting y adicionarlos al despliegue.

- ✓ RF4.1 Crear objetos gráficos mediante scripting.
- ✓ RF4.2 Agregar objetos gráficos al despliegue mediante scripting.

RF5 Salvar e importar scripts.

- ✓ RF5.1 Salvar scripts.
- ✓ RF5.2 Importar scripts.

2.3.2 Requisitos no funcionales:

Los requisitos no funcionales describen aspectos del sistema que son visibles por el usuario que no incluye una relación directa con el comportamiento funcional del sistema, incluyen restricciones como el tiempo de respuesta (desempeño), la precisión, recursos consumidos, seguridad, etc (22).

Usabilidad

- ✓ RNF1 El mecanismo debe ser fácil de utilizar para los operadores.

Software

- ✓ RNF2 El mecanismo debe ser implementado en el sistema operativo Debian.
- ✓ RNF3 El mecanismo debe estar incorporada al SCADA GALBA.

Hardware

- ✓ RNF4 El mecanismo debe instalarse junto con el SCADA en un ordenador con al menos 1 GB de memoria RAM.

Diseño e implementación

- ✓ RNF5 Permitir el completamiento de código en el editor.
- ✓ RNF6 Resaltar palabras reservadas del lenguaje.

2.4 Propuesta solución del sistema.

La solución que se propone abarca la creación de un editor de código script que permita asociar, agregar o modificar componentes gráficos a un despliegue en el módulo HMI del SCADA GALBA. Se utilizan palabras reservadas por el lenguaje JavaScript, el editor envía el código al despliegue del HMI y este a su vez asocia el código del editor y realiza las funciones entradas por el operador del sistema. Si existe algún error el HMI envía el error y el depurador lo muestra en el editor.

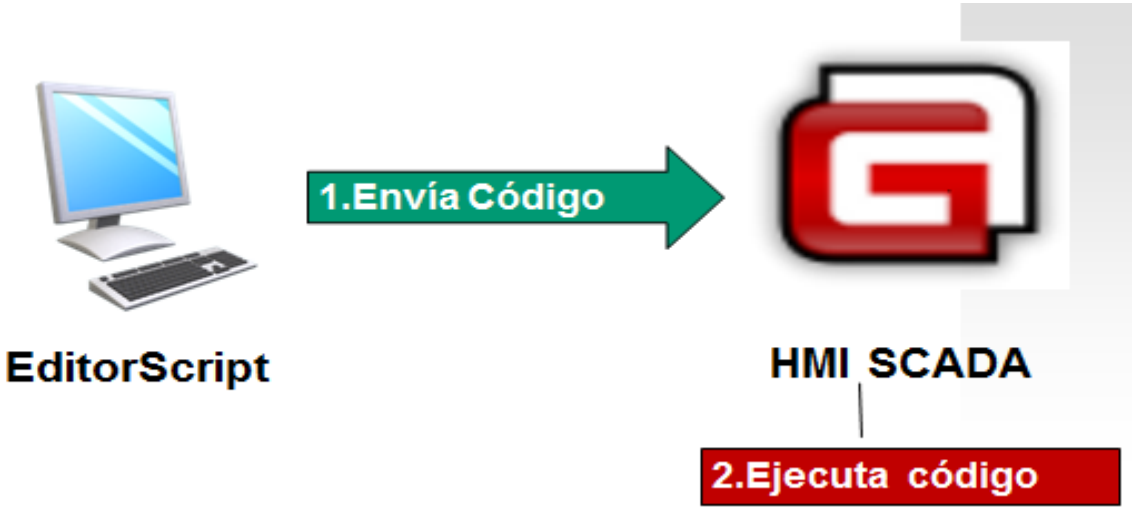


Figura 7: Propuesta solución del sistema.

2.5 Descripción de los actores

Los actores del sistema son abstracciones de las entidades externas a este, son subsistemas o clases que interactúan directamente con el sistema. A continuación una figura que lo demuestra.

Actor	Descripción
Usuario	Puede ser un informático encargado de mantener el software, especialista asociado a la rama con los que el sistema interactúa, alguien que haya consultado con algún operador del sistema.

2.5.1 Diagrama de Caso de Uso del Sistema.

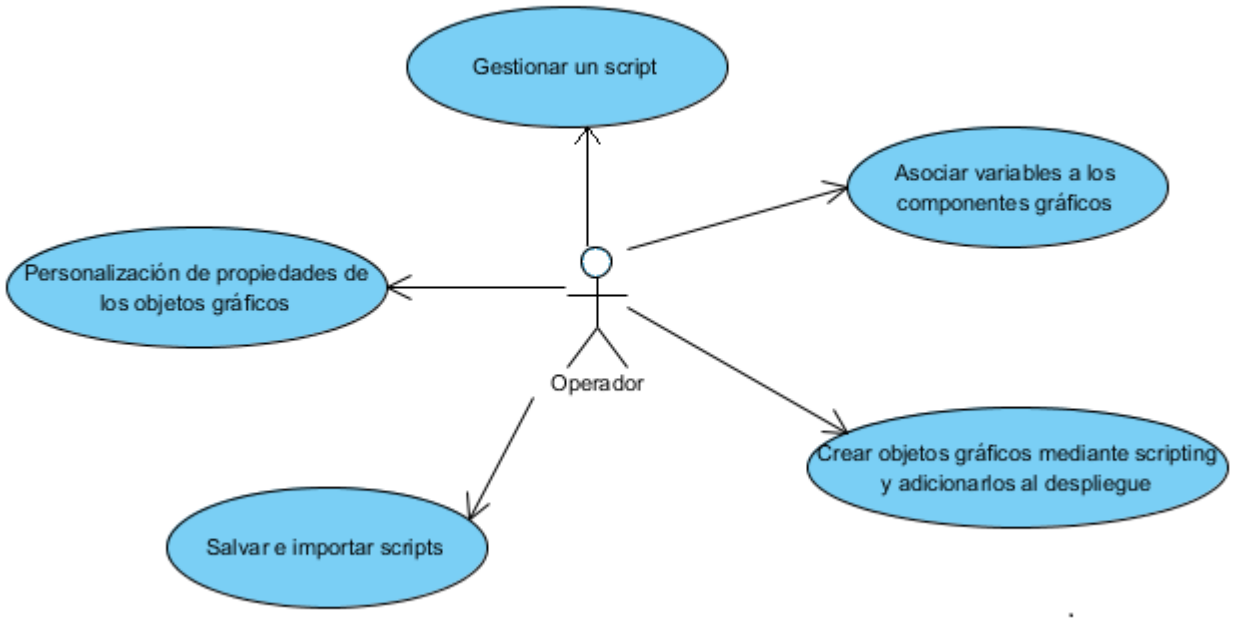


Figura 8: Diagrama de Caso de Uso del Sistema

2.5.2 Especificación de Casos de Usos

Propiedades Scripting

Tabla 1: Especificación del caso de uso Gestionar un Script

Caso de Uso	Gestionar un Script
Actores	Operador
Propósito	Este caso de uso se lleva a cabo con el objetivo de poder agregar o modificar los parámetros de los componentes gráficos del editor.
Resumen	El caso de uso se inicia cuando un operador del sistema decide agregar o modificar un componente gráfico en el editor.
Precondiciones	Editor de Scripting conectado al ambiente de configuración del HMI.
Referencias	RF1
Prioridad	Crítico.
Flujo Normal de eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El caso de uso el operador realiza en el editor las siguientes acciones adicionar, editar y eliminar scripts en el proyecto.	El sistema permite: • Adicionar Script. • Modificar Script. • Eliminar Script.

Sección “ Adicionar Script”	
Acciones del Actor	**Respuesta del Sistema**
2. El operador escribe el código script para adicionarlo y luego selecciona la opción salvar.	El sistema permite adicionar el script en el editor.
Sección “ Modificar Script”	
Acciones del Actor	**Respuesta del Sistema**
3. El operador después de seleccionar el script, modifica el código y lo ejecuta después.	El sistema permite modificar el script y así comprobar que se modificó mediante la ejecución del código.
Sección “ Eliminar Script”	
Acciones del Actor	**Respuesta del Sistema**
1. El operador después de seleccionar el script en la barra lateral derecha donde aparecen los script selecciona la opción eliminar y elimina el código script.	El sistema permite eliminar el script.
Prototipo Interfaz	

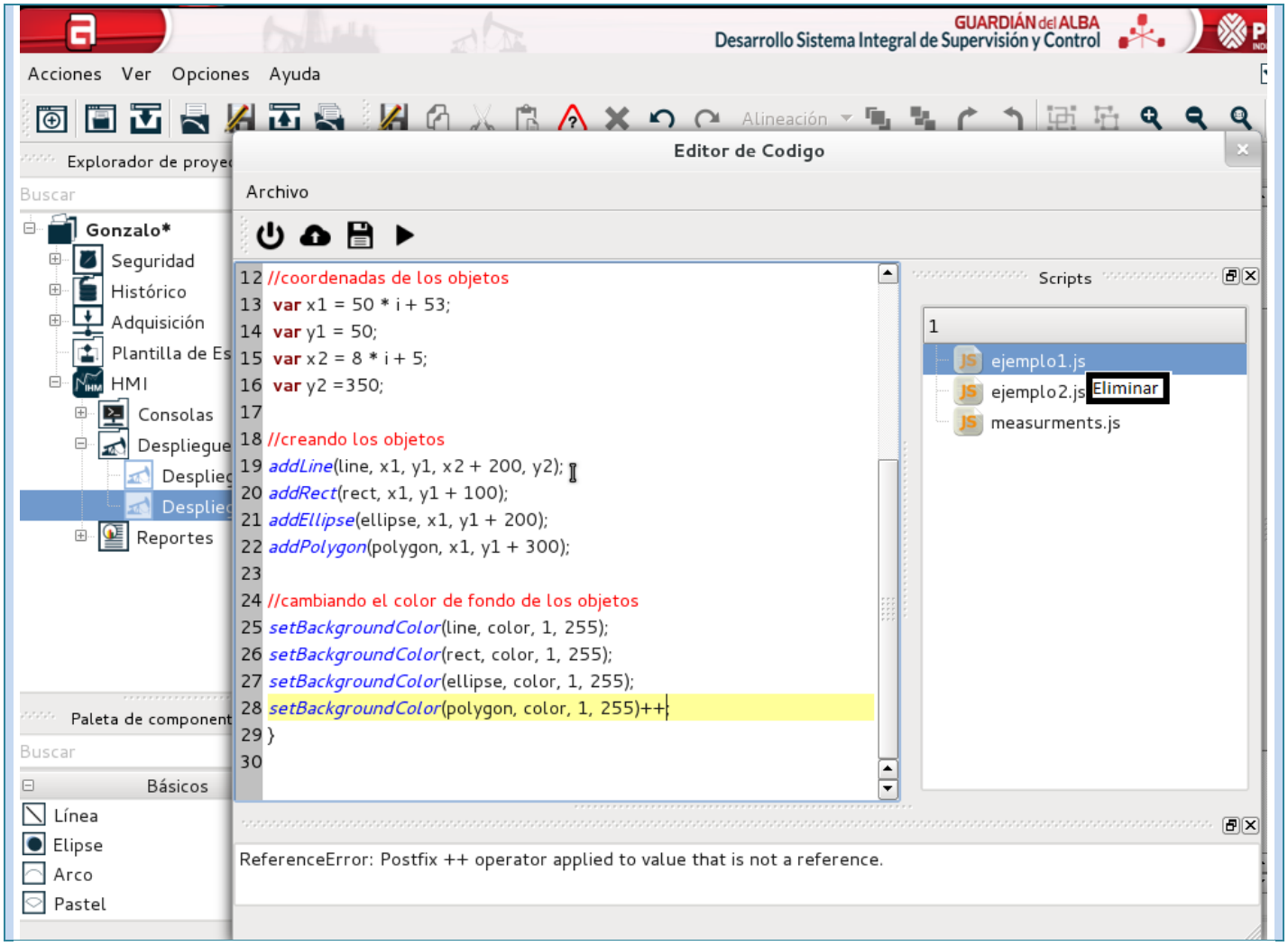


Tabla 2 : Especificación del caso de uso Personalización de propiedades

Caso de Uso	Personalización de propiedades de los objetos gráficos
Actores	Operador
Propósito	Este caso de uso se lleva a cabo con el objetivo de permitir al editor utilizar códigos script para modificar las propiedades de los objetos gráficos; posibilitando asociar, cambiar propiedades para dicho objeto
Resumen	El caso de uso se inicia cuando un operador del sistema decide personalizar alguna propiedad de un componente gráfico.
Precondiciones	Editor de Scripting conectado al ambiente de configuración del HMI.
Referencias	RF2
Prioridad	Crítico.
Flujo Normal de eventos	

Acciones del Actor	Respuesta del Sistema
1. El operador en el editor escribe los comandos de los componentes gráficos que desea personalizar y le da click en ejecutar.	El sistema coge esos comandos y los ejecuta, personalizando así la propiedad del o los componentes que se personalizaron.

Tabla 3: Especificación del caso de uso Asociar variables a los componentes gráficos

Caso de Uso	Asociar variables a los componentes gráficos
Actores	Operador
Propósito	Este caso de uso se hace con el objetivo el editor permita, asociarle variables a los componentes gráficos existentes en un despliegue.
Resumen	El caso de uso se inicia cuando un operador del sistema asociarle variables a los componentes gráficos existentes despliegue.
Precondiciones	Editor de Scripting conectado al ambiente de configuración HMI.
Referencias	RF3
Prioridad	Crítico.
Flujo Normal de eventos	

Acciones del Actor	Respuesta del Sistema
1. El operador asocia la variable al componente gráfico en el editor.	• El editor asocia la variable a los componentes gráficos. • El sistema permite que la variable adquiera los valo componente gráfico.

Prototipo Interfaz

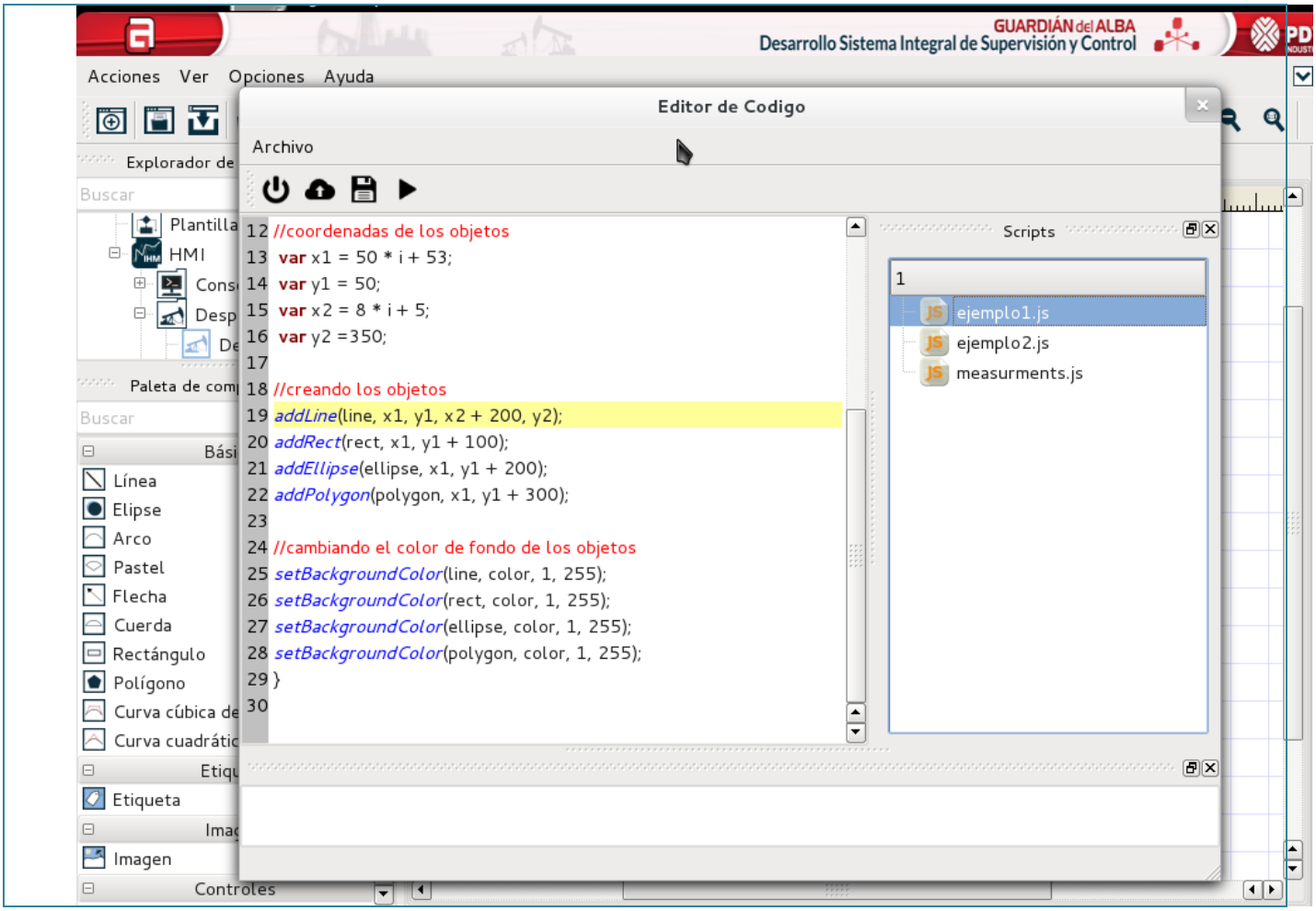


Tabla 4: Especificación del caso de uso Crear objetos gráficos mediante scripting y adicionarlos al despliegue.

Caso de Uso	Crear objetos gráficos mediante scripting y adicionarlos al despliegue.
Actores	Operador
Propósito	Este caso de uso se lleva a cabo con el objetivo de crear componentes gráficos mediante la programación y agregarlos al despliegue del proyecto.
Resumen	El caso de uso se inicia cuando el operador del sistema decide crear componentes gráficos mediante la programación y agregarlos al despliegue del proyecto.
Precondiciones	Editor de Scripting conectado al ambiente de configuración del HMI.
Referencias	RF4
Prioridad	Crítico.

Flujo Normal de eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El operador crea mediante el código script todos los componentes gráficos que desea adicionar a un despliegue del proyecto	El sistema crea componentes gráficos mediante la programación y los agrega al despliegue del proyecto.
Sección "Crear objetos gráficos mediante scripting"	
Acciones del Actor	**Respuesta del Sistema**
2. El operador insertar el código para crear objetos gráficos mediante el editor script.	El sistema crea objetos gráficos mediante el código script escritos en el editor.
Sección "Agregar objetos gráficos al despliegue mediante scripting"	
Acciones del Actor	**Respuesta del Sistema**
3. El operador una vez insertado el código script de crear objetos gráficos le da click a la opción ejecutar.	El sistema agrega al despliegue si el código no presenta ningún error, los componentes gráficos escritos en el anterior requisito.
Prototipo Interfaz	

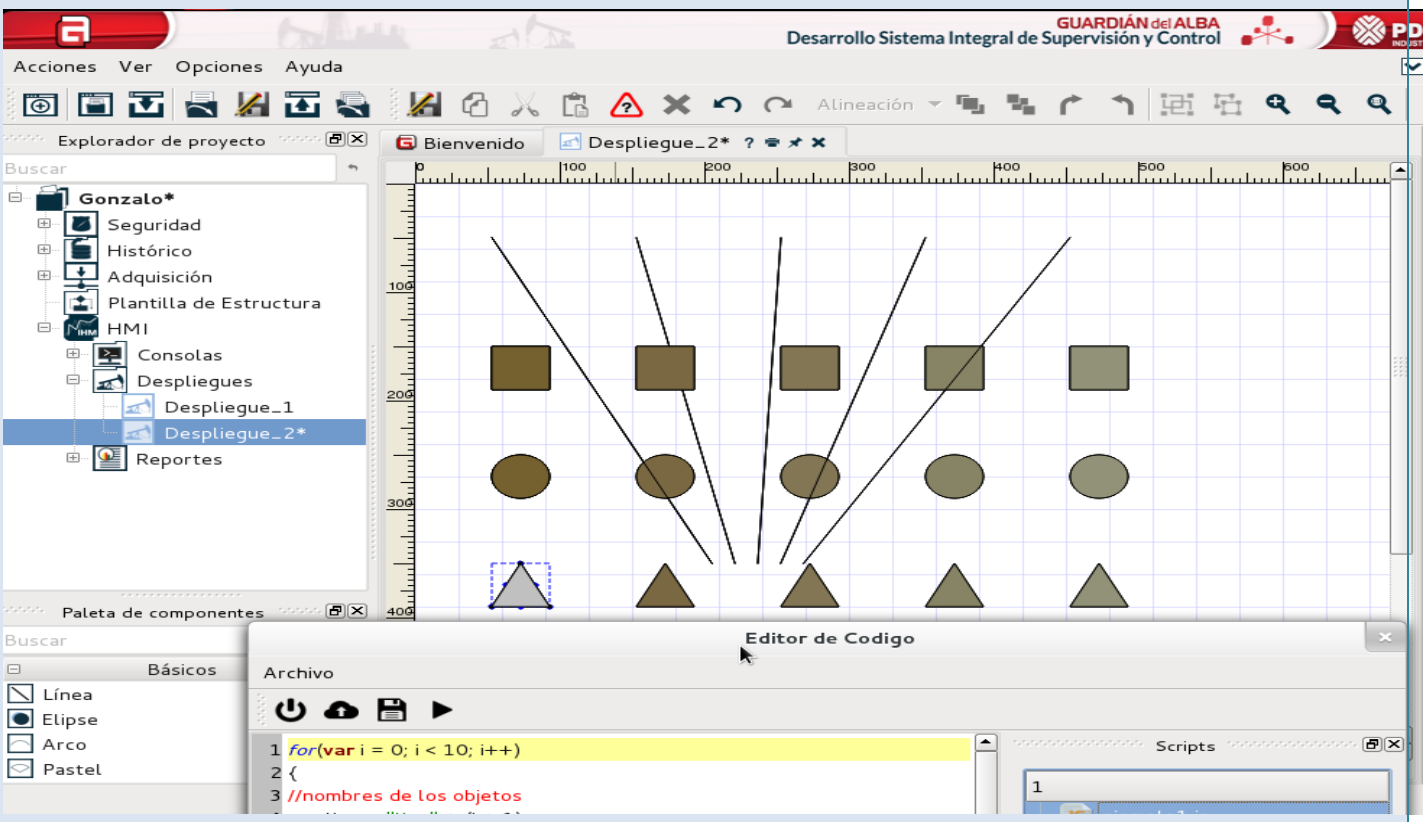
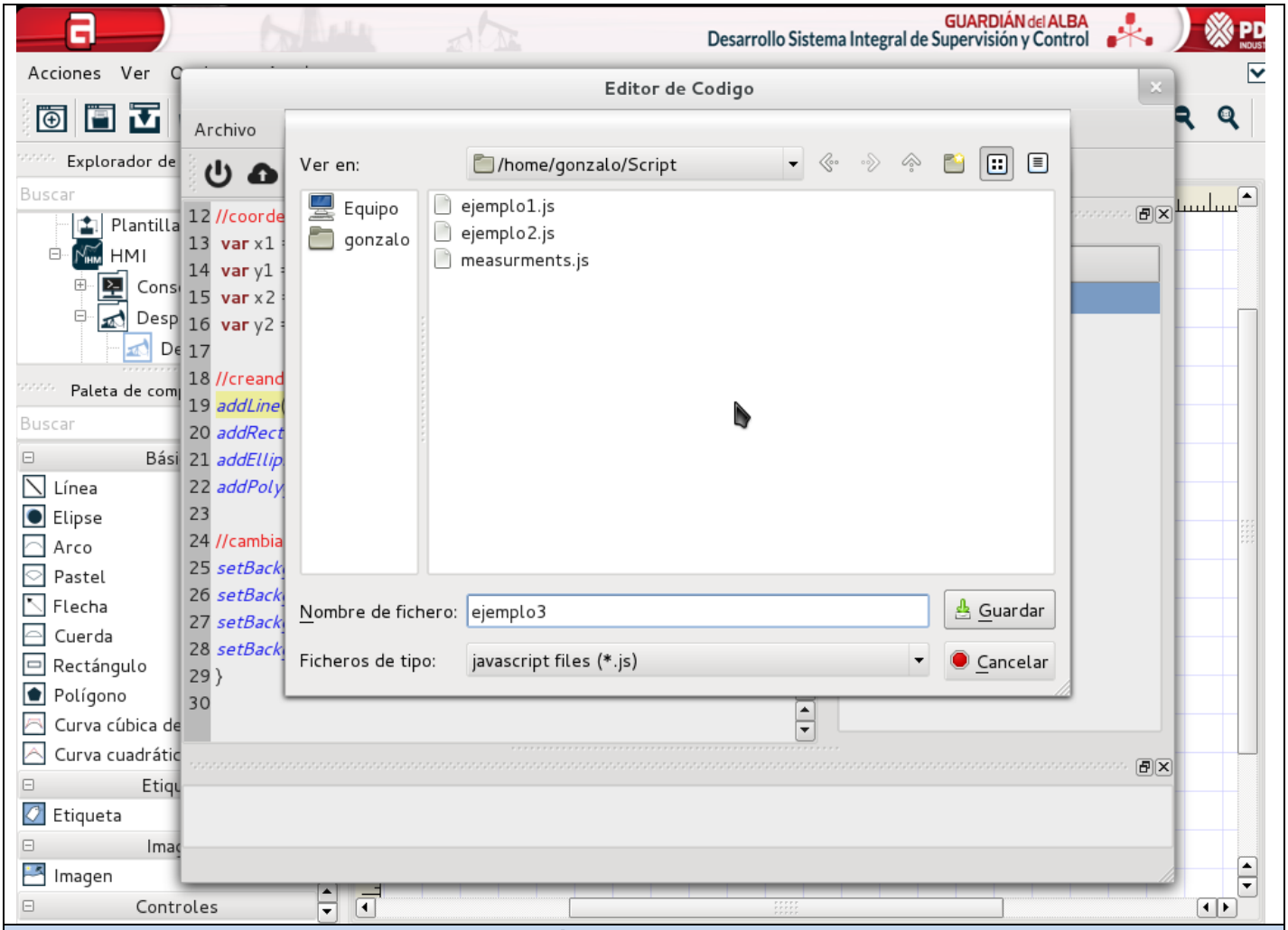


Tabla 5: Especificación del caso de uso Salvar e importar scripts

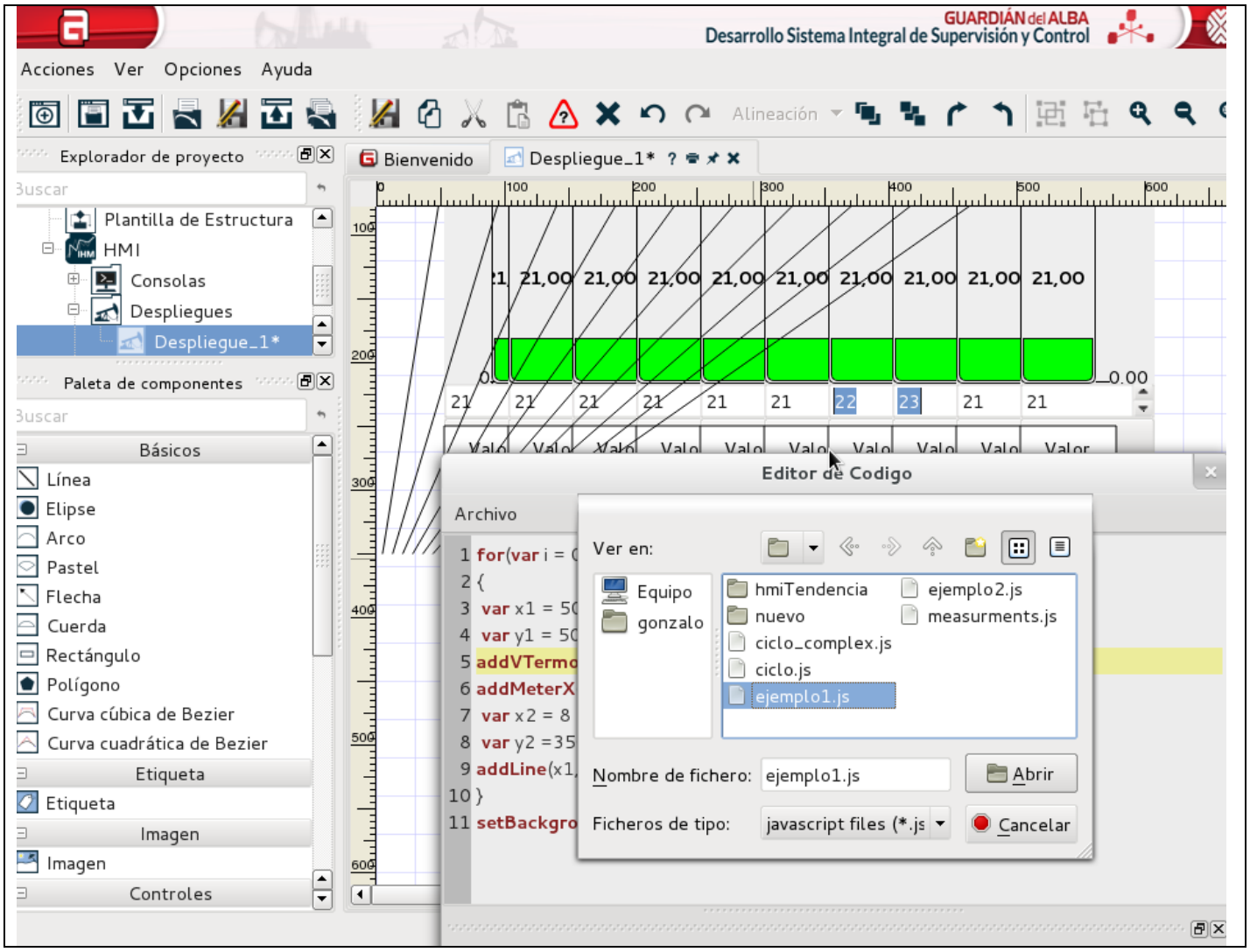
Caso de Uso	Salvar e importar scripts

Actores	Operador
Propósito	Este caso de uso se lleva a cabo con el objetivo de importar un script guardado con anterioridad o salvar un script creado por el operador.
Resumen	El caso de uso se inicia cuando un operador del sistema decide salvar o importar un script.
Precondiciones	Editor de Scripting conectado al ambiente de configuración del HMI.
Referencias	RF5
Prioridad	Crítico.

Flujo Normal de eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El caso de uso comienza cuando un operador decide salvar o importar un script.	El sistema permite que el operador salve o importe un código script.
Sección "Salvar script"	
Acciones del Actor	**Respuesta del Sistema**
2. El operador selecciona la opción salvar script y hace click en salvar.	El sistema permite salvar el script dentro de la carpeta destinada.
Prototipo Interfaz	



Sección “Importar script”	
Acciones del Actor	**Respuesta del Sistema**
3. El operador selecciona la opción importar script y hace click en salvar.	El sistema permite importar el script para su posterior uso dentro de la carpeta destinada.
Prototipo Interfaz	



2.6 Arquitectura propuesta

Se escoge **la arquitectura basada en N-Capas** ya que esta se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades (23). El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada. Por ejemplo, una aplicación está compuesta por una capa de presentación (funcionalidad relacionada con la interfaz de usuario), una capa de negocios (procesamiento de reglas de negocios) y una capa de datos (funcionalidad relacionada con el acceso a datos), en la tesis se usan las dos primeras capas: la capa de presentación (interfaz de usuario) compuesta por las clases *TreeWidget, JavaScriptCodeEditor y CodeEditor* y la capa de negocios (reglas de negocios) está compuesta por las clases *ScriptManager, JavaScriptCodeExecuter, ScriptModel y Highlighter*.

Principios fundamentales

Los principios comunes que se aplican cuando se diseña para usar este estilo de arquitectura incluyen:

Abstracción. La arquitectura basada en capas abstrae la vista del modelo como un todo mientras que provee suficiente detalle para entender las relaciones entre capas.

Encapsulamiento. El diseño no hace asunciones acerca de tipos de datos, métodos, propiedades o implementación.

Funcionalidad claramente definida. El diseño claramente define la separación entre la funcionalidad de cada capa. Capas superiores como la capa de presentación envían comandos a las capas inferiores como la capa de negocios y la capa de datos y los datos fluyen hacia y desde las capas en cualquier sentido.

Alta cohesión. Cada capa contiene funcionalidad directamente relacionas con la tarea de dicha capa.

Reutilizable. Las capas inferiores no tienen ninguna dependencia con las capas superiores, permitiéndoles ser reutilizables en otros escenarios.

Beneficios

Los principales beneficios del estilo de arquitectura basado en capas son:

Abstracción. Las capas permiten cambios que se realicen en un nivel abstracto. Usted puede incrementar o disminuir el nivel de abstracción usado en cada capa de la “pila” jerárquica.

Aislamiento. El estilo de arquitectura de capas permite aislar los cambios en tecnologías a ciertas capas para reducir el impacto en el sistema total.

Rendimiento. Distribuir las capas entre múltiples sistemas (físicos) puede incrementar la escalabilidad, la tolerancia a fallos y el rendimiento.

2.7 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Es un tema importante en el desarrollo de software actual permite capturar la experiencia, busca ayudar a la comunidad de desarrolladores de software a resolver problemas comunes, creando un cuerpo literario de base y un lenguaje común para comunicar ideas y experiencia acerca de los problemas y sus soluciones. El uso de patrones ayuda a obtener un software de calidad (reutilización y extensibilidad) (24).

2.7.1 Patrones GoF (Gang of Four).

Los patrones GoF según sus características se utilizan en situaciones frecuentes, debido a que se basan en la experiencia acumulada al resolver problemas reiterativos, favorecen la reutilización de código, ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar además indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje. (25).

Se usa **el patrón *Observer*** es uno de los más importantes (y utilizados) de todos los patrones de diseño vistos hasta el momento. Su filosofía es simple: un objeto, denominado *sujeto* (*Subject*) posee un estado. Cuando su estado cambia, es capaz de "avisar" a sus *subcriptores* (*Observers*) de este cambio de estado. De este modo, los objetos suscritos al objeto no tienen que preocuparse de cuándo se produce un cambio de estado: éste se encargará de informar de forma activa a todos aquellos objetos que hayan decidido suscribirse (26). Este patrón se usa en la clase codeeditor donde se hace el trabajo con las señales *(signal)*.

2.7.2 Patrones GRASP

Los Patrones de diseño GRASP son patrones generales de software para asignación de responsabilidades, es el acrónimo de "GRASP (object-oriented design General Responsibility Assignment Software Patterns)". Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

Se usa **el patrón Experto.** El GRASP de experto en información es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento).

Se usa **el patrón Creador.** El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase contiene o agrega la clase.

Se usa **el patrón Controlador** se usa en la clase codeeditor ya que es la clase encargada de controlar los procesos**.** El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar

separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

Se usan **los patrones Alta Cohesión y Bajo Acoplamiento.** Los conceptos de cohesión y acoplamiento no están íntimamente relacionados, sin embargo se recomienda tener un mayor grado de cohesión con un menor grado de acoplamiento. De esta forma se tiene menor dependencia y se especifican los propósitos de cada objeto en el sistema. **Alta Cohesión** dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase, **Bajo Acoplamiento** Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (27).

2.8 Diagrama de clases

El diagrama de clases expresa la estructura u organización del software mediante las clases. Es un reflejo abstracto de los componentes y las relaciones entre ellos. Los diagramas de clase forman una parte básica del modelado con UML, siendo utilizados tanto para mostrar lo que el sistema puede hacer, como para mostrar cómo puede ser construido. Si se crea un diagrama de clases, se está modelando una parte de los elementos y relaciones que configuran la vista de diseño del sistema. A continuación se muestra el diagrama de clases que representa la estructura del sistema. (28)

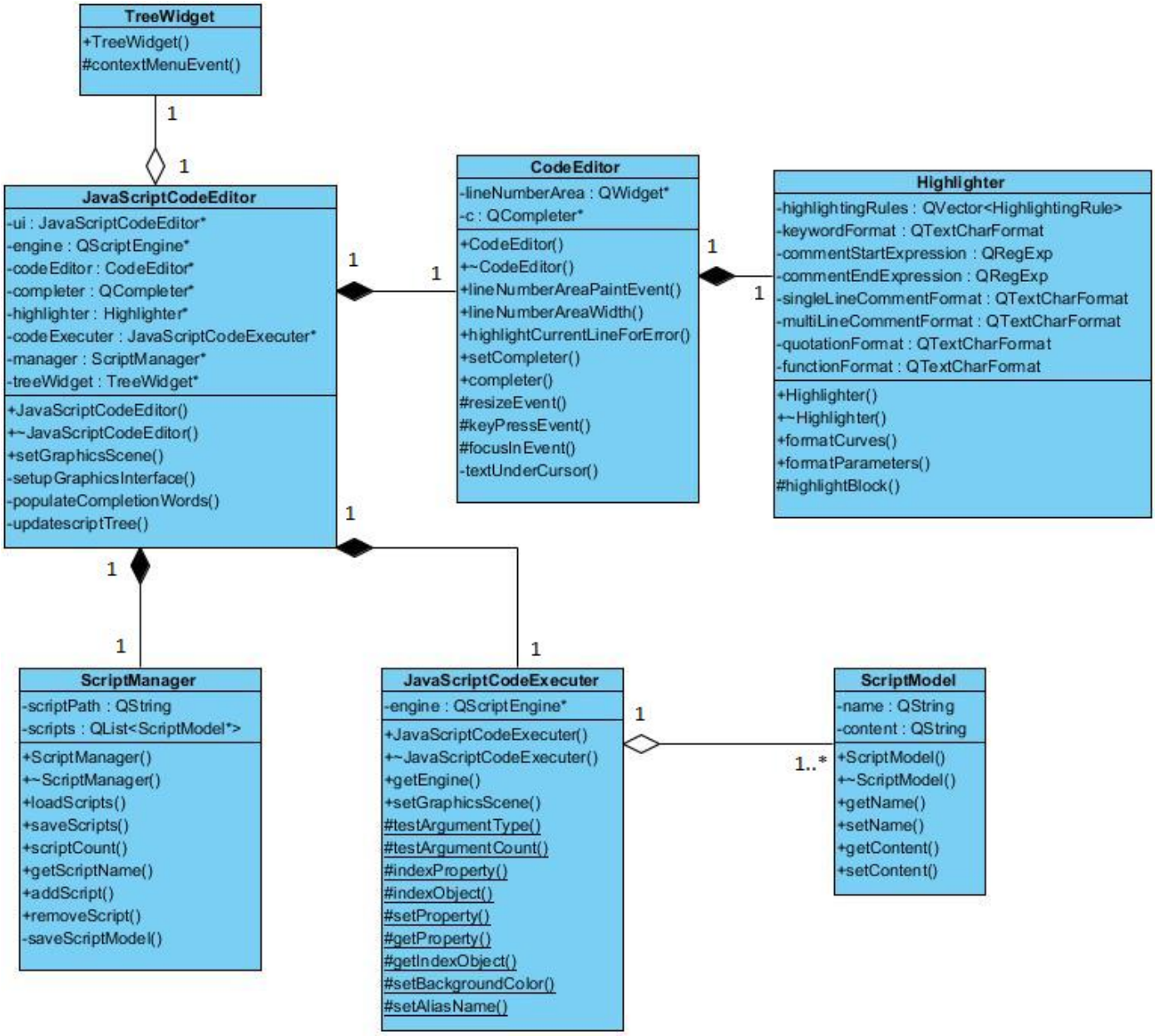


Figura 9: Diagrama de clases

2.8.1 Descripción del Diagrama de clases

El diagrama representa la relación entre clases haciendo el uso de arquitectura definida Arquitectura basada en N-Capas a continuación una descripción de las clases queda resumida en la siguiente tabla.

Clase	Descripción
TreeWidget	Esta clase es la encargada de guardar los códigos y mostrarlos en la barra lateral derecha del editor. Permite mostrar el script y eliminarlo de la lista.

JavaScriptCodeEditor	Esta clase muestra una vista a la ventana del editor, dentro de ella se ejecutan acciones importantes como Abrir_Codigo_desde_fichero, Guardar_codigo_a_fichero y Exec (Ejecutar).
CodeEditor	Clase donde se escribe el código para ejecutarlo luego, ventana donde el operador escribe el código a ejecutar. Utiliza el uso del highlighter para las palabras reservadas.
Highlighter	Clase que permite colorear las palabras reservadas para el editor, contiene formato de textos para permitirle al operador una mejor vista del código que está escribiendo.
ScriptManager	Clase que permite cargar, adicionar (addScript) y eliminar (removeScript).
JavaScriptCodeExecuter	Clase encargada de las funciones c++ que se van a exportar a JavaScript se usan propiedades básicas como setBackgroundColor, setAliasName, setDescription, etc. También permite adicionar objetos o componentes del despliegue con las funciones addLine, addVTermometer, addMeterX, etc.
ScriptModel	Clase que contienes el nombre y el contenido del editor. Permite cambiar el nombre y el contenido del script.

2.9 Diagrama de secuencia

En el flujo de diseño se utilizan los diagramas de secuencia, estos incluyen las interacciones entre clases del diseño a través de mensajes, que describen las operaciones que realiza cada clase para colaborar con otras con el objetivo de dar cumplimiento a la petición del operador del sistema.

A continuación se muestra el diagrama de secuencia para un caso de uso crítico Crear objetos gráficos mediante scripting y adicionarlos al despliegue:

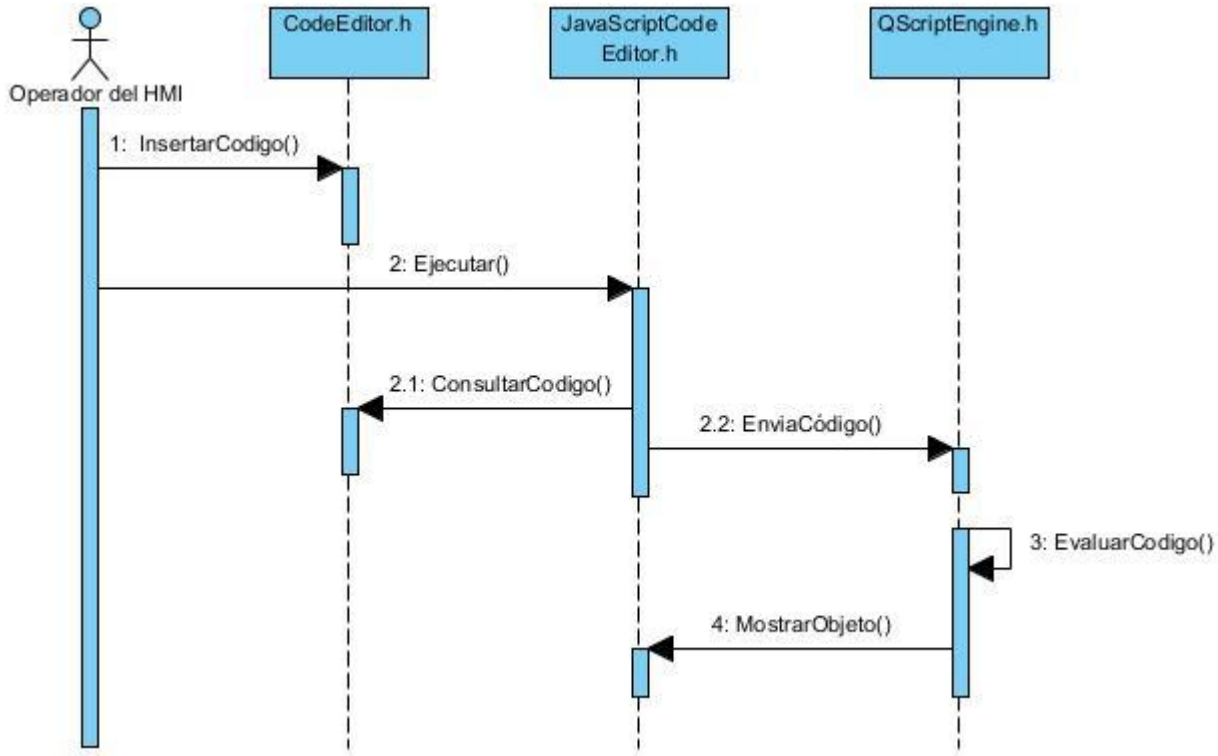


Figura 10: Diagrama de secuencia

2.9.1 Descripción del diagrama de secuencia

El diagrama de secuencia inicia cuando el operador del HMI decide crear objetos gráficos mediante scripting y adicionarlos al despliegue, lo primero que debe hacer es insertar el código de los objetos que desea crear, esta información queda registrada en la clase *CodeEditor.h* y luego presiona la opción Ejecutar y los datos son enviados a la clase *JavaScriptCodeEditor.h* para consultar el código entrado, por otra parte después de consultado el código se envía a la clase *QScriptEngine.h* para evaluar dicho código y luego se hace la llamada de MostrarObjeto() a la clase *JavaScriptCodeEditor.h* para mostrar el resultado final.

2.10 Consideraciones parciales

En el presente capítulo se han seleccionado las tecnologías y herramientas que se adecuan a las características del módulo que se desea desarrollar. Se asegura el uso de tecnologías, promoviendo la utilización de desarrollo de la tesis. Se realizó un estudio de las metodologías de desarrollo existentes con el propósito de seleccionar la más adecuada para la solución de la investigación y fue seleccionada la metodología AUP como guía en el proceso de desarrollo del software ya que permite que en un tiempo relativamente corto el proceso de desarrollo quede debidamente documentado.

Capítulo 3: IMPLEMENTACIÓN Y PRUEBA DEL MECANISMO DE SCRIPTING

3.1 Introducción

En el presente capítulo se abordan los temas relacionados con la implementación del mecanismo, basándose en el trabajo desarrollado en los capítulos anteriores. Posteriormente se valida la solución propuesta mediante el correcto funcionamiento de las principales funcionalidades de la aplicación.

3.2 Estándar de codificación

Los estándares de codificación, también conocidos como estilos de programación o convenciones de código, son convenios para escribir código fuente en ciertos lenguajes de programación. Permiten que el código en consecuencia sea mantenible y que todos los participantes lo puedan entender en un menor tiempo.

-En los archivos cabecera debe incluir el copyright y la licencia, o una referencia de la misma, al estilo GNU GPL.

-Se adopta el estilo de bloques de documentación de JavaDoc, el cual consiste de un bloque de comentario de estilo C.

-Es importante especificar el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @autor y @date.

-El código será escrito en inglés y la documentación en español.

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

Implementación

El modelo de implementación describe como los elementos del modelo de diseño y las clases, se implementan en términos de componentes, ficheros de código fuente, ejecutables, entre otros. Como resultado se obtiene un sistema ejecutable que incluye todas las funcionalidades propuestas en la captura de requisitos funcionales.

3.3 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y las dependencias lógicas de componentes de software. El diagrama de componentes del Mecanismo de Integración de Scripting para el SCADA se corresponde con la arquitectura definida que da soporte a la aplicación (29).

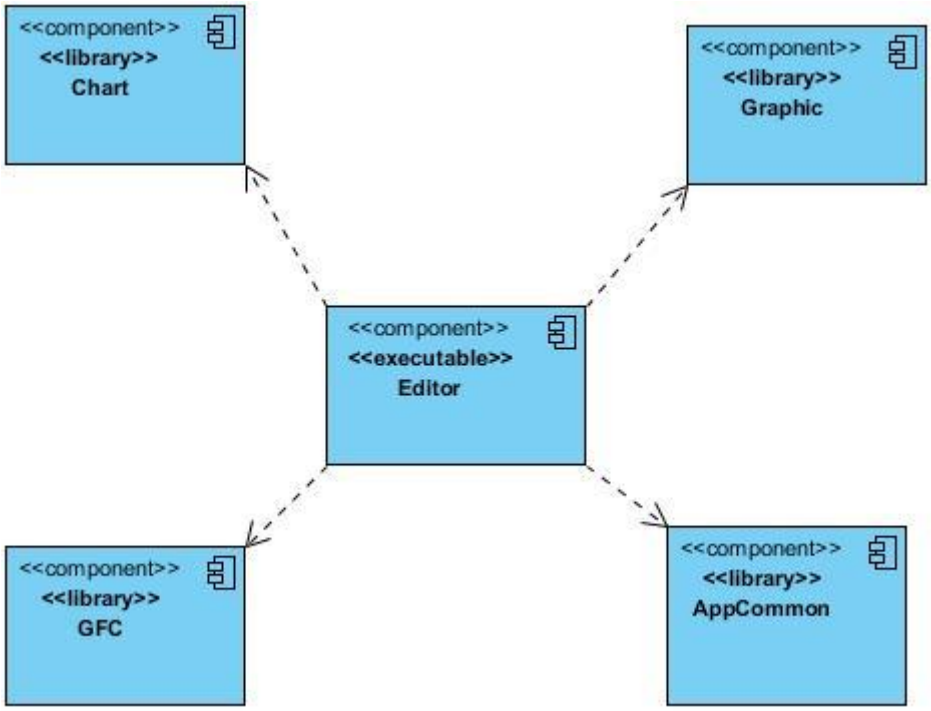


Figura 11: Diagrama de componentes

En el desarrollo del Editor se utilizan diferentes bibliotecas para dar cumplimiento a los requisitos propuestos para el sistema se utiliza la biblioteca Chart que permite interactuar con los gráficos de barras, xy y algunas tendencias utilizadas, por otra parte se utiliza la biblioteca Graphic que es donde se encuentran los objetos que son widgets tales como meters, label, combo box, bottoms, etc. Se usa GFC[1] para el trabajo con los componentes básicos, como elipse, línea, polígono, rectángulo, etc y se usa la biblioteca AppCommon porque presenta las características específicas del editor.

3.4 Modelo de Despliegue

Un modelo de despliegue es el encargado de capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. Consiste en uno o más nodos, dispositivos y conectores, estos últimos estarán ubicados entre nodos y dispositivos. A continuación se propone la distribución física de los elementos que conforman el módulo.

[1] GFC es una biblioteca de animaciones de QT.

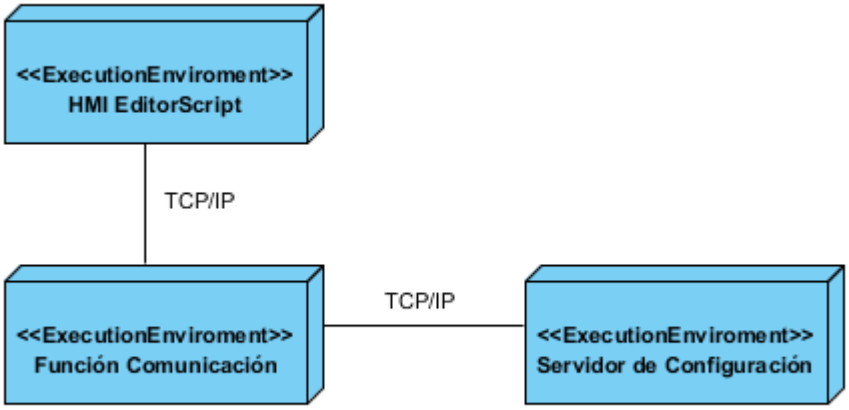


Figura 12: Modelo de Despliegue

En el diagrama se muestra como el HMI EditorScript encargado de enviar el código se comunica por el protocolo TCP/IP con la Función de Comunicación para luego enviarle la información con el mismo protocolo al Servidor de Comunicación el cual permite realizar las tareas del EditorScript.

3.5 Pruebas

Como define la metodología AUP, uno de sus pilares es el proceso de pruebas. Esto permite aumentar la calidad del sistema reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. Por lo que permitió aumentar la seguridad al evitar efectos colaterales no deseados, realizando modificaciones. Esta metodología realiza pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de cada iteración se consiguió la funcionalidad requerida por el cliente.

3.5.1 Pruebas de Aceptación

La ejecución de esta prueba de aceptación, permite la evaluación de las funcionalidades del mecanismo desarrollado antes de implantarlo en su entorno real de explotación. Los resultados obtenidos se muestran a continuación:

Tabla 6: Caso de prueba Crear script asociándolo al componente gráfico seleccionado

Caso de Prueba de Aceptación	
Numero:	**Caso de Uso: 1**
Nombre: Crear script asociándolo al componente gráfico seleccionado.	
Descripción: Verificar la correcta asignación de un script al componente gráfico seleccionado y la personalización de propiedades mediante el script.	

Condiciones de ejecución: El cliente debe comprobar que se pueda acceder a las propiedades del componente gráfico, y mediante la ejecución del script, modificar las mismas.
Entradas/Pasos de ejecución: Probar el acceso y modificación de las propiedades de cada uno de los componentes gráficos del proyecto utilizando estas en el script.
Resultado esperado: Se accede a las propiedades de los componentes gráficos del proyecto, modificando las mismas mediante scripting.
Evaluación de la prueba: Satisfactorio

Tabla 7: Asociar variable al componente gráfico seleccionado.

Caso de Prueba de Aceptación	
Numero:3	**Caso de Uso: 3**
Nombre: Asociar variable al componente gráfico seleccionado.	
Descripción: Verificar la correcta asignación de variables al componente gráfico seleccionado	
Condiciones de ejecución: El cliente debe comprobar que se pueda acceder a las variables de los puntos de adquisición y realizar la asignación de las mismas al componente gráfico.	
Entradas/Pasos de ejecución: Probar el acceso y modificación de las variables de los puntos creados en adquisición y asignar las mismas a cada uno de los componentes gráficos del proyecto.	
Resultado esperado: Se accede a las variables de los puntos de adquisición y se realiza la asignación de estas a los componentes gráficos del proyecto.	
Evaluación de la prueba: Satisfactorio	

Tabla 8: Creación de componentes gráficos mediante script y agregarlos al despliegue del proyecto.

Caso de Prueba de Aceptación	
Numero:5	**Caso de Uso:** 4
Nombre: Creación de componentes gráficos mediante script y agregarlos al despliegue del proyecto.	
Descripción: Verificar la correcta creación de los componentes gráficos del sistema mediante script y su agregación al despliegue del proyecto activo.	

Condiciones de ejecución: El cliente debe comprobar que la herramienta permita crear componentes gráficos mediante la programación y agregarlos al despliegue activo del proyecto.
Entradas/Pasos de ejecución: Crear un código script en el que se creen todos los componentes gráficos del sistema y agregarlos a un despliegue activo del proyecto.
Resultado esperado: Se crean mediante el código script todos los componentes gráficos y se adicionan a un despliegue del proyecto.
Evaluación de la prueba: Satisfactorio

Tabla 9: Importar códigos script desde archivos locales.

Caso de Prueba de Aceptación	
Numero:6	**Caso de Uso:** 5
Nombre: Importar códigos script desde archivos locales.	
Descripción: Verificar la correcta importación de códigos script al editor.	
Condiciones de ejecución: El cliente debe comprobar que la herramienta permita importar correctamente códigos script desde archivos locales.	
Entradas/Pasos de ejecución: Seleccionar la opción de abrir código existente y verificar la importación del archivo seleccionado.	
Resultado esperado: Se selecciona la opción de abrir y se importa el código script desde el fichero seleccionado.	
Evaluación de la prueba: Satisfactorio	

Tabla 10: Salvar códigos script en archivos locales.

Caso de Prueba de Aceptación	
Numero:7	**Caso de Uso:** 5
Nombre: Salvar códigos script en archivos locales.	
Descripción: Verificar el correcto almacenamiento de códigos script en ficheros.	
Condiciones de ejecución: El cliente debe comprobar que la herramienta permita almacenar correctamente códigos script en archivos locales.	

Entradas/Pasos de ejecución: Seleccionar la opción de salvar el código creado y verificar el almacenamiento en el archivo seleccionado.
Resultado esperado: Se selecciona la opción de salvar y se crea un fichero con el código script en caso de seleccionar un archivo inexistente.
Evaluación de la prueba: Satisfactorio

3.6 Resumen del resultado de las iteraciones

Cada iteración ejecutada concluyó con la realización de un conjunto de pruebas de aceptación para verificar el cumplimiento de los requisitos. Dichas pruebas permitieron al operador comprobar el grado de satisfacción con el resultado. Las no conformidades detectadas en cada iteración fueron mitigadas antes de dar paso a una nueva iteración. El gráfico que se muestra a continuación resume los resultados arrojados por cada iteración:

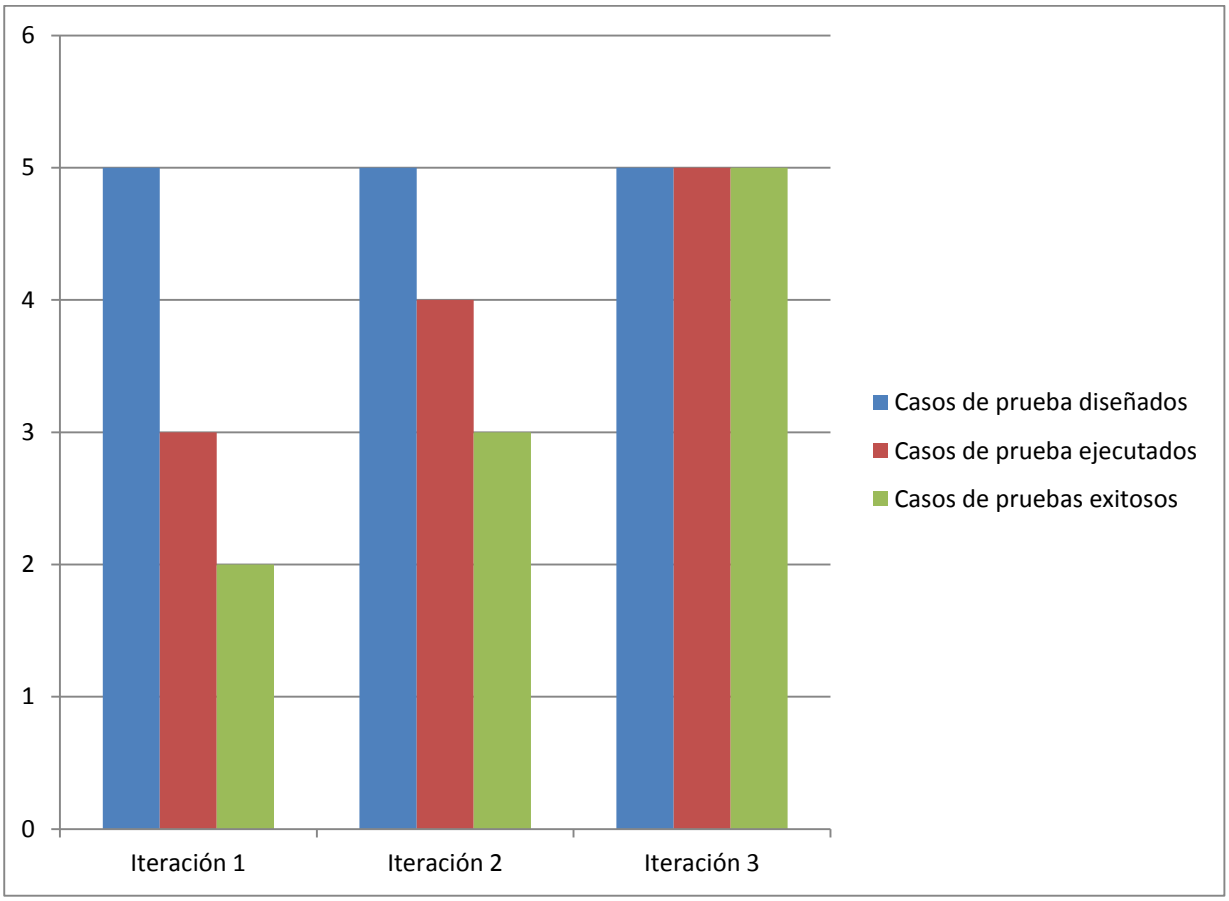


Figura 13: Resumen de los resultados las pruebas realizadas

3.7 Resultados del mecanismo de integración de scripting sobre el HMI del SCADA GALBA.

El mecanismo ya integrado al SCADA permite optimizar las configuraciones realizadas en el HMI del SCADA GALBA, los operadores se vieron beneficiados con el uso de la herramienta de gestión de la lógica operacional generada por el Editor Script, disminuyendo la complejidad de realizar de forma visual las configuraciones en el editor. Logrando así una integración entre los diferentes componentes de manera sencilla y rápida, permitiendo la creación de aplicaciones complejas.

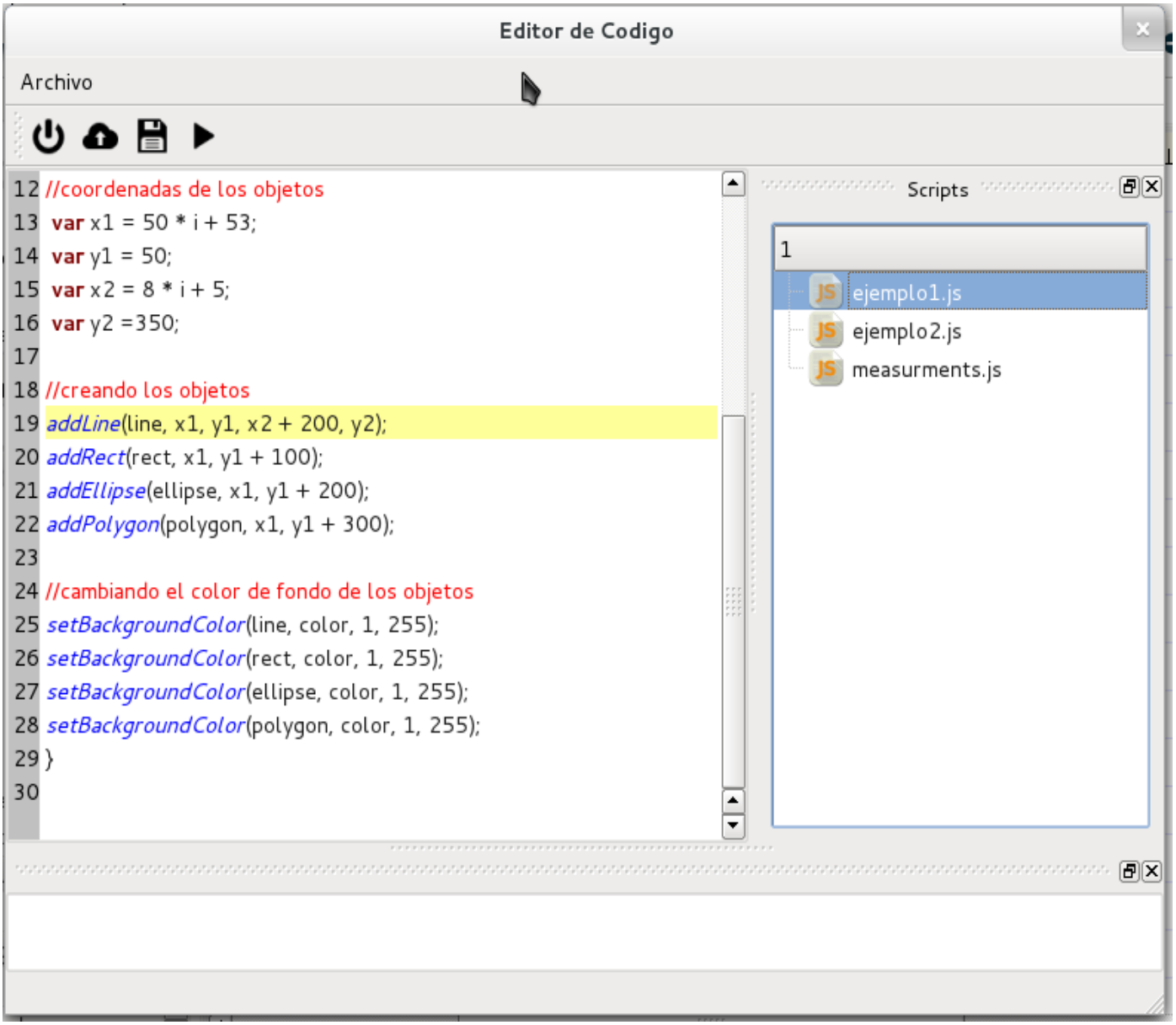


Figura 14: Resultados del mecanismo de integración de scripting

3.8 Consideraciones parciales

En este capítulo se definió el estándar de codificación a utilizar. Se desarrollaron las tareas correspondientes para dar solución a los Casos de Uso. Además se ejecutaron las pruebas de aceptación diseñadas en el capítulo anterior. Se llega a la conclusión de que el mecanismo se encuentra listo para su puesta en funcionamiento dado el resultado de las pruebas realizadas.

Conclusiones generales

- Como resultado del proceso de desarrollo se obtuvo un mecanismo de integración de scripting para el SCADA GALBA, que cumple con los requisitos mínimos para este tipo de aplicación.
- El estudio de las herramientas de scripting arrojó que las mismas no pueden ser usadas por sus características y especificidades técnicas pero contribuyeron al desarrollo del editor de script.
- La ejecución de las pruebas de aceptación al mecanismo desarrollado, contribuyó a concluir que cumple satisfactoriamente con los requerimientos especificados por el cliente.

Recomendaciones

- Elevar el nivel de funcionalidades del editor, agregarle la personalización de animaciones.
- Seguir actualizando el ScriptEditor a medida que se agreguen componentes nuevos.
- Mejorar el Depurador de código del ScriptEditor.

Bibliografía

1. **S.A., Sitio oficial de Petróleos de Venezuela.** [En línea] [Citado el: 18 de febrero de 2015.] http://www.pdvsa.com/index.php?tpl=interface.sp/design/.

2. Control supervisorio remoto presentacion. [En línea] [Citado el: 25 de abril de 2015.] http://www.academia.edu/8469802/Control_supervisorio_remoto_presentacion.

3. Carlos de Castro Lozano, Cristobal Romero Morales. [En línea] [Citado el: 3 de enero de 2015.] http://www.uco.es/grupos/eatco/automatica/ihm/descargar/scada.pdf.

4. [En línea] [Citado el: 15 de febrero de 2015.] http://www.instrumentacionycontrol.net/cursos-libres/automatizacion/cursos-sistemas-scada.

5. human machine interface. [En línea] [Citado el: 25 de abril de 2015.] https://books.google.com.cu/books?id=NWHRciSLmuoC&pg=PA195&dq=human+machine+interface+definition+scada&hl=es&sa=X&ved=0CB4Q6AEwAGoVChMI6o3euYGExgIV7WOMCh1HUwCl#v=onepage&q=human%20machine%20interface%20definition%20scada&f=false.

6. **Galba, Proyecto Scada Nacional. Nombre Código:.** [En línea] [Citado el: 25 de enero de 2015.] publicaciones.uci.cu/index.php/SC/article/download/435/251.

7. Invensys Systems. [En línea] [Citado el: 3 de enero de 2015.] InTouch HMI Scripting And Logic Guide.

8. Manual de WinCC. [En línea] [Citado el: 26 de abril de 2015.] http://es.scribd.com/doc/57325173/Curso-WinCC-V6..

9. OMRON. [En línea] [Citado el: 25 de marzo de 2015.] CX-Supervisor. User Manual. Software Release 3.1.

10. **UCI., Metodología de desarrollo para la Actividad productiva de la.** Metodología UCI.pdf. Habana,cuba : s.n., 2015.

11. GNU/Linux. [En línea] [Citado el: 28 de enero de 2015.] https://www.debian.org/releases/lenny/sparc/ch01s02.html.es.

12. Lenguaje de programacion C++. [En línea] [Citado el: 29 de enero de 2015.] http://lenguajedeprogramacion21.blogspot.com/.

13. Qt - Developer Resources - documentation, guides, forums. [En línea] [Citado el: 15 de enero de 2015.] www.qt.io/developers/.

14. El Lenguaje Unificado(UML). [En línea] [Citado el: 6 de febrero de 2015.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

15. **Software, Herramientas CASE para el proceso de desarrollo de.** [En línea] [Citado el: 23 de febrero de 2015.] http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml.

16. Manel Redondo Sol. [En línea] 25 de enero de 2015. http://www.recercat.cat/bitstream/handle/2072/13784/PFC .

17. Visual Basic for Applications. [En línea] [Citado el: 12 de febrero de 2015.] www.lynda.com/Visual-Basic-for-Applications.

18. HMI Visual Basic Scripting Language. [En línea] [Citado el: 13 de febrero de 2015.] www.isagraf.com/pages/.../app_notes/using_vb.pdf.

19. C++ Script a scripting language built on C++. [En línea] [Citado el: 24 de febrero de 2015.] http://calumgrant.net/cppscript/index.html..

20. Ingeniería del software. [En línea] [Citado el: 2 de febrero de 2015.] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/practicas-1/is1-p03-trans.pdf.

21. Requisitos Funcionales y no Funcionales. [En línea] [Citado el: 23 de febrero de 2015.] https://sistemas.uniandes.edu.co/~csof5101/dokuwiki/lib/exe/fetch.php?media=principal:csof5101-requerimientos.pdf.

22. Gabriel Mendez. [En línea] [Citado el: 23 de enero de 2015.] https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf.

23. **Pelaez, Juan.** [En línea] 26 de mayo de 2009. http://www.juanpelaez.com/geek-stuff/arquitectura/arquitectura-basada-en-capas/.

24. **Microsoft.** [En línea] 25 de enero de 2015. https://msdn.microsoft.com/es-es/library/bb972272.aspx#EDAA.

25. Patrones de diseño - Departamento de Informática. [En línea] Félix Prieto. [Citado el: 3 de marzo de 2015.] www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

26. Daniel Garcia. [En línea] 26 de enero de 2015. https://danielggarcia.wordpress.com/2014/06/02/patrones-de-comportamiento-vi-patron-observer/.

27. Fundamentos de la Ingeniería del Software. [En línea] [Citado el: 23 de abril de 2015.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

28. **Berzal, Fernando.** Diagramas de clases. [En línea] 1 de abril de 2015. http://elvex.ugr.es/decsai/java/pdf/3C-Relaciones.pdf.

29. Diagramas UML: Componentes y despliegue - SlideShare. [En línea] [Citado el: 22 de abril de 2015.] http://es.slideshare.net/joshell/diagramas-uml-componentes-y-despliegue.

30. Evaluacion de estandares HMI. [En línea] 12 de enero de 2015. iull.ull.es/xmlui/bitstream/handle/915/657/Evaluacion de estandares HMI Aplicacion de la guia GEDIS a los Sistemas SCADA del NAP.

31. [En línea] 16 de noviembre de 2014. http://iaci.unq.edu.ar/materias/laboratorio2/HMI%5CIntroduccion%20HMI.pdf.

32. [En línea] 3 de febrero de 2014. [Citado el: 22 de enero de 2015.] javascript.espaciolatino.com/.

33. **Méndez, Correa.** [En línea] [Citado el: 3 de febrero de 2015.] http://repositorio.espe.edu.ec/bitstream/21000/3401/1/T-ESPEL-0406.pdf.

34. **oficial, Documento.** Metodología de Desarrollo para la Actividad productiva de la UCI.

35. **creator, qt.** Qt Documentation Snapshots. [En línea] [Citado el: 30 de enero de 2015.]

36. Introducción a JavaScript . [En línea] [Citado el: 26 de abril de 2015.] librosweb.es/libro/javascript/.

37. Introdución a HMI. [En línea] [Citado el: 3 de marzo de 2015.]

Glosario de términos

API: Del inglés Application Programming Interface. Interfaz de Programación de Aplicaciones. Una serie de rutinas usadas por una aplicación para gestionar generalmente servicios de bajo nivel, realizados por el sistema operativo de la computadora.

Aplicación: Cualquier programa que corra en un sistema operativo y que haga una función específica para un usuario. Por ejemplo, procesadores de palabras, bases de datos, agendas electrónicas, etc.

Archivo: Archivo es el equivalente a "file", en inglés. Es data que ha sido codificada para ser manipulada por una computadora. Los archivos de computadora pueden ser guardados en CD-ROM, DVD, disco duro o cualquier otro medio de almacenamiento.

Despliegue: Vista del HMI dentro del sistema SCADA GALBA.

Depuración: Es el proceso donde se identifican errores en la programación.

DLL o dll: "Dynamic Link Library" es un archivo para el sistema operativo Windows y un mini-programa ejecutable, que hace un enlace entre en el disco duro y los programas cuando son cargados.

HTML: Siglas del inglés Hipertexto Markup Language (Lenguaje de Marcado Hipertexto). Es un lenguaje para crear documentos de hipertexto para uso en el www o intranets, por ejemplo.

Interface: Interfaz o interface es el punto de conexión ya sea dos componentes de hardware, dos programas o entre un usuario y un programa.

Java: Lenguaje de programación que permite ejecutar programas escritos en un lenguaje muy parecido al C++. Se diferencia de un CGI ya que la ejecución es completamente realizada en la computadora cliente, en lugar del servidor.

JavaScript: Lenguaje desarrollado por Sun Microsystems en conjunto con Netscape; aunque es parecido a Java se diferencia de él en que los programas están incorporados en el archivo HTML.

Mac OS: Sistema operativo desarrollándose desde 1984 aprox., por la empresa Apple, para la Macintosh.

PHP: Lenguaje de script diseñado para la creación de páginas web activas (similares a ".asp" de Microsoft), multiplataforma (puede correr en Windows, Mac, Linux). Usualmente se usa en conjunto con la base de datos MySQL, pero puede usar cualquier otro tipo de base de datos como por ejemplo Oracle, SQL o Postgres.

POO: Programación Orientada a Objetos (POO) es una filosofía de programación que se basa en la utilización de objetos. El objetivo de la POO es "imponer" una serie de normas de desarrollo que aseguren y faciliten la mantenibilidad y reusabilidad del código.

SCADA proviene de las siglas de Supervisory Control And Data Adquisition.

SCADA GALBA: software desarrollado por la UCI, nombrado Guardián del Alba.

Software: Es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo.

Script: archivo de órdenes, archivo de procesamiento por lotes o guión es un programa usualmente simple.

Scripting: es un lenguaje de programación que soporta scripts, programas escritos para un ambiente especial en tiempo de ejecución.

SQL: Structured Query Language. Es un lenguaje especializado de programación que permite realizar consultas (queries en inglés) a bases de datos.

Visual Basic: Lenguaje de programación de Microsoft orientado a eventos y utilizado principalmente para realizar consultas a bases de datos de Microsoft como Fox Pro, SQL, etc. que funcionan en servidores Windows.

InTouch: es una pantalla que mediante un toque directo sobre su superficie permite la entrada de datos y órdenes a un sistema interactivo diseñado para la visualización, la supervisión y el control de procesos industriales.

Anexos

Tabla 11: Fases de la Metodologia AUP.

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración Construcción Transición	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Tabla 12: Disciplinas de la Metodologia AUP.

Disciplinas AUP	Disciplinas Variación AUP-UCI	Objetivos Disciplinas (Variación AUP-UCI)
Modelo	Modelado de negocio (opcional)	El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el

		software desarrollado va a cumplir su propósito.
	Requisitos	El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
	Análisis y diseño	En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis.
Implementación	Implementación	En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
Prueba	Pruebas internas	En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posibles componentes de prueba ejecutables para automatizar las pruebas.
	Pruebas de liberación	Pruebas diseñadas y ejecutadas por una

		entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
	Pruebas de Aceptación	Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.
Despliegue	Despliegue (Opcional)	Constituye la instalación, configuración, adecuación, puesta en marcha de soluciones informáticas y entrenamiento al personal del cliente.
Gestión de configuración	Se cubren con las áreas de procesos PP, PMC y CM. Las mismas son áreas de procesos de gestión y soporte respectivamente.	Consultar en mejoras.prod.uci.cu los libros de procesos de cada una de estas áreas.
Gestión de proyecto		
Entorno		