Universidad de las Ciencias Informáticas

Facultad 5

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

Mecanismo de depuración del módulo de configuración del SCADA Galba.

Autor: Raudel Arencibia Ramírez

Tutores: Ing. Yadira Ramírez Rodríguez

Ing. Miguel Angel Socorro Borges

La Habana, Cuba

Junio 2015

Declaración de autoría

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas con los derechos patrimoniales de la misma, con carácter exclusivo. Autorizo a dicho centro para que haga el uso que estime pertinente con este trabajo.

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

_____

Firma del autor

Raudel Arencibia Ramírez

_____

Firma del tutor

Ing. Miguel Ángel Socorro Borges

_____

Firma del tutor

Ing. Yadira Ramírez Rodríguez

Dedicatoria

Dedico el presente trabajo a mi madre, quien ha sido de todo para mí en esta vida. A mi hermana, a mis abuelos maternos, a mi peluita mortificona (Ofelita), a mis amigos, a todos los que me quieren.

Agradecimientos

A mí querida madre Ileana a quien nunca podré pagarle el tiempo y el amor invertido en mí, gracias por ser mi mamá, eres la mejor que pude haber tenido.

A mis abuelo Idalia y Daniel, más que mis abuelos son mis segundos padres y hasta los terceros, gracias por quererme y amarme tanto.

A mi hermana Claudia, de quien me siento muy orgulloso.

A novia Ofelia por ser mi sostén en el día a día, y por soportar mi forma de ser particular.

A mi padre de crianza Adolfo.

A todo el resto de mi familia.

A mis dos tutores Yadira y Miguel Angel, por la paciencia que han tenido conmigo y por toda la ayuda que me han dado.

A todos los profesores que he tenido en todos los niveles de enseñanza, cada uno de ellos ha aportado un granito de arena a lo que soy hoy, especialmente a Dismer Ronda.

A Yanet por ser mi hermanita cuña.

A Yasely por todo el tiempo que tiene para mí.

A mi hermanito Daynier por estar ahí cuando lo necesito.

A mi gran amigo Liuver por ser uno de mis pilares fundamentales en toda la universidad.

A mis amigas Maiyara, Dayana, Rosmery, Nairobis, Rosmery Pedraza, Beatriz, Yamila, Gladis.

A mis amigos Jorge Alfonso, Julio, Orlando, Tony, Manuel, Reitel, Edward, Emilio Benavides, Sergio, Emilio Bejar, Alberto, Javier, Alejandro, Carlos, Dariel, Yasser, Adolfo, Cajigal, en fin a todos los que han estado a mi lado en estos 5 año.

Resumen

En la actualidad el desarrollo tecnológico ha contribuido con la automatización de las industrias a través de los Sistemas de Control, Supervisión y Adquisición de Datos (SCADA) por sus siglas inglés Supervisory Control and Data Acquisition. Estos sistemas requieren una elevada serie de configuraciones debido a sus dimensiones y complejidad. El módulo de configuración del SCADA Guardián del ALBA (Galba) es el encargado de almacenar y administrar estas configuraciones al resto de los módulos que comprenden el sistema. Resulta de vital importancia saber qué se le está enviando a cada módulo en cada momento, para ello el módulo de configuración cuenta con un modo depuración encargado de generar en ficheros de texto toda esta información. Este modo representa una traba en el funcionamiento del módulo una vez que este se encontraba en ejecución, debido a que su utilización significaría realizar un reinicio del módulo, terminando con las conexiones existentes en ese momento. Con el desarrollo del presente trabajo se logró obtener un mecanismo de depuración aceptado por el cliente, capaz de activar y desactivar dicho modo sin la necesidad de reiniciar el módulo de configuración. Además genera toda la información en ficheros HTML los cuales mejoran el proceso de revisión de la misma al presentarla en forma de árbol en un navegador web, dando la posibilidad de contraer o expandir partes según sea necesario.

Palabras clave: depuración, mecanismo, módulo de configuración, SCADA.

Contenido

Introducción

El desarrollo e innovación de nuevas tecnologías ha posibilitado la automatización de los procesos industriales permitiendo a las compañías implementar procesos de producción eficientes, seguros y competitivos (Carbajal, 2014).

Para lograr dicha automatización han surgido los llamados Sistemas de Control, Supervisión y Adquisición de Datos, por sus siglas en inglés Supervisory Control and Data Acquisition (SCADA), diseñados para funcionar sobre ordenadores en el control de la producción y proporcionar comunicación con los dispositivos de campo. Además, envían la información generada en el proceso productivo a diversos usuarios, tanto del mismo nivel como hacia otros supervisores dentro de la empresa, es decir, permiten la participación de otras áreas como por ejemplo: control de calidad, supervisión y mantenimiento (Hernández, 2010).

Los sistemas SCADA son utilizados ampliamente en el control de infraestructuras críticas o servicios esenciales en el funcionamiento de la economía y la sociedad (Amaya, 2013). Es por ello que la disponibilidad es un factor imprescindible en estos sistemas. Garantiza que los usuarios autorizados puedan acceder a la información y recursos de red cuando los necesiten para tomar decisiones. Estos sistemas son diseñados para controlar grandes procesos industriales distribuidos geográficamente, tal es el caso de las industrias petroleras y plantas nucleares. Sumada a esta característica se encuentra el elevado número de funcionalidades que presentan, es por ello que son desarrollados en módulos. En su representación simple la modularidad es la capacidad que tiene el sistema de verse como la integración de varias partes, donde cada una de ellas puede ser estudiada y analizada por separado, además cumple una función definida (Stutzke, 2005).

La nación cubana no está ajena al desarrollo de sistemas SCADA, específicamente el Centro de Informática Industrial (CEDIN) de la Universidad de Ciencias Informáticas (UCI) tiene experiencia en el desarrollo de estos sistemas. El centro cuenta con el Departamento de Desarrollo de Aplicaciones (DDA), el cual se estructura por proyectos que en su mayoría forman parte de una familia de

productos, como es el caso de la Línea de Productos de Software (LPS) SCADA (Javier, 2015). Hoy en día el CEDIN enfoca el desarrollo de la LPS en función de los productos SCADA, un ejemplo lo constituye el SCADA Guardián del Alba (SCADA Galba). Este sistema cuenta con los siguientes módulos: comunicación, adquisición, seguridad, histórico, comunicación con terceros, planificador de tarea, HMI editor, HMI visualizador y configuración.

El módulo de configuración está formado por un grupo de componentes y una base de datos que contiene las configuraciones de cada uno de los módulos que conformen el proyecto activo. Es el encargado de almacenar y suministrar la información base para el funcionamiento de los demás módulos del SCADA, garantizando la persistencia de dicha información. (Doce, 2010) Esta información resulta ser la configuración que regirá el funcionamiento de los módulos y por ende la del SCADA Galba en general.

En la actualidad el módulo de configuración posee un modo de depuración. Para la utilización de este modo se hace necesario reiniciar el módulo una vez que se encontraba en ejecución, afectándose así su disponibilidad. Cuando el módulo de configuración funciona en este modo genera ficheros de texto con la información que se envió a los restantes módulos. Estos ficheros no contienen la información de forma estructurada, lo cual provoca que la revisión de la misma por parte de los probadores del sistema sea compleja y tome tiempo realizarla; además les resulta complicado comprobar que esté completa y sea igual a la que se tiene almacenada en la configuración general del sistema.

Por todo lo anteriormente planteado se enuncia el siguiente **problema científico**: ¿Cómo obtener desde el módulo de configuración la información enviada a los restantes módulos del SCADA Galba para su revisión sin la necesidad de reiniciar el módulo?

El **objeto de estudio** se centra en: Módulo de configuración de los sistemas SCADA

Introducción

El **objetivo** de este trabajo consiste en: Desarrollar un mecanismo de depuración que permita obtener la información enviada por el módulo de configuración del SCADA Galba a los restantes módulos.

Delimitando como **campo de acción**: Mecanismo de depuración del módulo de configuración del SCADA Galba.

Para darle cumplimiento al objetivo propuesto se plantean las siguientes tareas investigativas:

- Revisión bibliográfica para desarrollar un marco teórico de acuerdo a las tendencias actuales en el desarrollo de los sistemas SCADA y de forma más específica en los mecanismos de depuración.
- Definición de los requisitos para el desarrollo del mecanismo.
- Diseño del mecanismo definido según los requisitos encontrados.
- Implementación del mecanismo como propuesta de solución.
- Realización de pruebas de aceptación al mecanismo.

Para guiar la investigación científica se utilizarán los siguientes métodos científicos:

Métodos teóricos:

- Análisis histórico-lógico para realizar un estudio de la evolución de los sistemas SCADA determinando las técnicas y tendencias actuales en su desarrollo.
- Analítico sintético para el estudio de toda la bibliografía referente al desarrollo de sistemas SCADA, sintetizando los elementos más importantes referentes a módulos de configuración para dar solución al problema existente.

Métodos empíricos:

- Entrevistas para la recolección de la información y el conocimiento existente en los especialistas vinculados a los temas de automatización industrial.
- Experimentos para la elaboración de prototipos funcionales, con el objetivo de comprobar la efectividad de la implementación del mecanismo.

Introducción

El presente trabajo consta con la siguiente estructura capitular:

Capítulo 1: “Marco teórico de la investigación” relacionado con el desarrollo de sistemas SCADA, de forma específica con mecanismos de depuración utilizados por módulos de estos sistemas. Se realiza un estudio de las soluciones existentes en el ámbito nacional e internacional. Además se explica la metodología, herramientas y tendencias actuales a utilizar en el desarrollo del mecanismo de depuración del módulo configuración.

Capítulo 2: “Fases planificación y diseño”. Se presenta la descripción de la solución propuesta al problema de investigación. Se describe el funcionamiento general del mecanismo a desarrollar en relación con sus características y componentes. Se definen los elementos de documentación más significativos tras la aplicación de la metodología de desarrollo seleccionada.

Capítulo 3: “Fases desarrollo y pruebas”. Se presentan las tareas de ingeniería desarrolladas, se muestra la codificación utilizada en la obtención de la solución propuesta. Se realizan las pruebas de aceptación para validar los requerimientos del cliente. Se presenta una carta aceptación emitida por el cliente donde se expresa su conformidad con el mecanismo de depuración obtenido.

Capítulo 1: Marco teórico de la investigación.

Introducción

En este capítulo se abordarán conceptos esenciales relacionados con los sistemas SCADA, así como los usos y funcionalidades que brindan, haciendo un énfasis en los elementos afines al módulo de configuración. Se estudian los sistemas SCADA reconocidos en el mundo para determinar cómo realizan el proceso de depuración. Además se analizan las tecnologías existentes capaces de serializar información para determinar sus características y tenerlas en cuenta en la solución propuesta. Se identifican las herramientas a utilizar en el desarrollo del software.

1.1 Sistemas SCADA

Los sistemas SCADA son un ejemplo de sistemas distribuidos, según (W.Salter, 1999) SCADA significa Sistema de control, supervisión y adquisición de datos. Como su nombre indica, no es un servicio completo de control, sino que se centra también en la supervisión. Como tal, es un paquete de software que es puramente posicionado en la parte superior de hardware a la que está interconectado, en general a través de los controladores lógicos programables (PLC), u otros módulos de hardware comerciales.

Otra definición de un sistema SCADA es la siguiente: tecnología que posibilita a un usuario recolectar datos de uno o más medios distantes y enviar instrucciones de control limitadas a estos medios. Un SCADA hace que sea innecesario que un operador visite frecuentemente localizaciones remotas cuando estas operan normalmente. SCADA incluye una interfaz de operación y una de operación de datos (Boyer, 2009).

Para el desarrollo de la presente investigación se toma como definición de sistema SCADA la siguiente: "*aplicación de software diseñada para funcionar sobre ordenadores en el control de producción, proporcionando comunicación con los dispositivos de campo y controlando el proceso de forma automática desde la*

pantalla del operador" (Cáceres, 2009), al considerarse por parte del autor completa conteniendo los elementos comunes de las definiciones mencionadas.

1.2 Sistema SCADA Guardián del ALBA

El SCADA Guardián del ALBA (Galba) es un sistema distribuido en módulos que trabajan de manera conjunta posibilitando el funcionamiento del sistema como un todo. Estos módulos se encuentran interconectados a través de un software para la distribución de los servicios en la red, conocido como software de comunicación entre aplicaciones. La distribución de los módulos existentes en el SCADA permite obtener configuraciones escalables en dependencia de los requisitos que presente cada aplicación. Es un sistema en tiempo real y presenta una arquitectura distribuida.

Está dividido en varios subsistemas entre los que se encuentran (Álvarez, 2008):

Comunicación: Es la capa de software encargada de la comunicación entre los diferentes módulos que forman parte del sistema. Este módulo tiene como finalidad proporcionar la capa de comunicación de alto nivel, tanto sincrónica, como asincrónica, para la comunicación de todos los módulos que conforman el sistema SCADA.

Adquisición: Es el encargado de la adquisición, recepción, procesamiento y distribución de los datos provenientes del campo.

Almacenamiento de datos históricos: Es el encargado de almacenar la información del sistema para que posteriormente pueda ser empleada, por ejemplo, en generación de reportes, tendencias o en gestión de producción. La base de datos histórica (BDH) contendrá la información persistente de los datos recolectados de los dispositivos.

Seguridad: Proporciona las funcionalidades necesarias para garantizar el trabajo autorizado a usuarios y módulos, además brinda las herramientas necesarias para la protección contra ataques maliciosos o involuntarios al sistema por parte de personas o recursos, tales como fallas de energía, problemas de red o servidores.

Visualización o HMI: Se encarga de representar en un ordenador, los procesos que ocurren en el campo en tiempo real, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador diferentes niveles de control en dependencia de sus niveles de privilegios. Este módulo permite al operador el contacto directo con el sistema y realizar la supervisión y el control del proceso en general.

Además de los subsistemas mencionados, el SCADA Galba cuenta con el módulo de configuración.

1.3 Módulo de configuración del SCADA Galba

El módulo de configuración del SCADA Galba es el encargado de almacenar y suministrar la información base para el funcionamiento de los demás módulos del SCADA a través del uso de los agentes de configuración. Para lograr un funcionamiento eficiente del mismo es necesario garantizar la escalabilidad del módulo, de manera que soporte desde una pequeña configuración hasta una configuración de gran complejidad y tamaño. El módulo debe ser robusto para soportar cualquier problema de seguridad que atente contra la estabilidad del sistema. Este módulo soporta la configuración en frío, en caliente y operacional.

Estas configuraciones son explicadas a continuación (Carbón, 2014):

Configuración en frío: Cada uno de los módulos del sistema se inicia y solicitan sus parámetros de arranque al servidor de configuración posteriormente se sincronizan. Todos los datos y archivos temporales son limpiados, excepto los valores almacenados en el servidor de datos históricos, que se mantienen de ejecuciones anteriores. Al finalizar esta acción el sistema se encuentra en la condición de inicializado. Para los módulos la configuración en frío implica que deben reiniciarse y volver a leer toda la configuración, lo cual es solventado con la configuración en caliente y operacional.

Configuración en caliente: El sistema se encuentra en ejecución y se realizan modificaciones en la configuración desde una interfaz de edición. Toda configuración en caliente se deriva o está asociada a una configuración actual, de

modo que el sistema se actualiza con los nuevos datos de configuración siguiendo los cambios realizados y no inicializándose como en el caso de la configuración en frío, este proceso ocurre sin afectar aquellos elementos del sistema que no estén involucrados en los cambios.

Configuración operacional: El operador realiza cambios en el ambiente de ejecución del sistema (por lo general cambios mínimos en la configuración que no impliquen crear ni eliminar). El sistema se actualiza con los nuevos datos de configuración sin afectar la ejecución de este.

1.4 Agente de configuración

Se entiende por agente de configuración a la biblioteca que le brinda el módulo de configuración al resto de los módulos del SCADA para que puedan obtener o modificar la configuración. Existe una biblioteca específica por cada tipo de módulo definido en el sistema. La labor fundamental de los agentes es abstraer a los módulos de la complejidad del proceso que es necesario realizar cuando se desea obtener o modificar la configuración. (Rodríguez, y otros, 2014)

Las funcionalidades generales a todos los agentes son (Rodríguez, y otros, 2014):

- Iniciar la comunicación con el servidor de configuración.
- Obtener los datos de la configuración que necesita el módulo.
- Notificar al módulo cuando se produjo un cambio en la configuración que lo afecta a él.
- Notificar al módulo cuando el mismo fue eliminado de la configuración.
- Informar de los cambios operacionales, en frío y en caliente que se produjeron en la configuración afectan a la configuración del módulo.
- Terminar la comunicación con el servidor de configuración.

1.5 Mecanismo de depuración

Un mecanismo es, en términos generales, una entidad o proceso cuya principal característica es la producción regular de cierto aspecto. Por otro lado la depuración

o debugging es el proceso metodológico para encontrar y reducir errores o defectos en un programa informático o en una pieza de hardware. Otro concepto de depurar consiste en eliminar impurezas, pero en informática es un vocablo utilizado en el trabajo de programación, que consiste en revisar y analizar si la sintaxis de un programa creado es correcta y/o genera errores al ejecutarlo. (Mastermegazine, 2010) Una vez analizado ambos concepto se plantea por parte del autor como mecanismo de depuración para la presente investigación, al proceso capaz de encontrar y corregir errores en un programa informático.

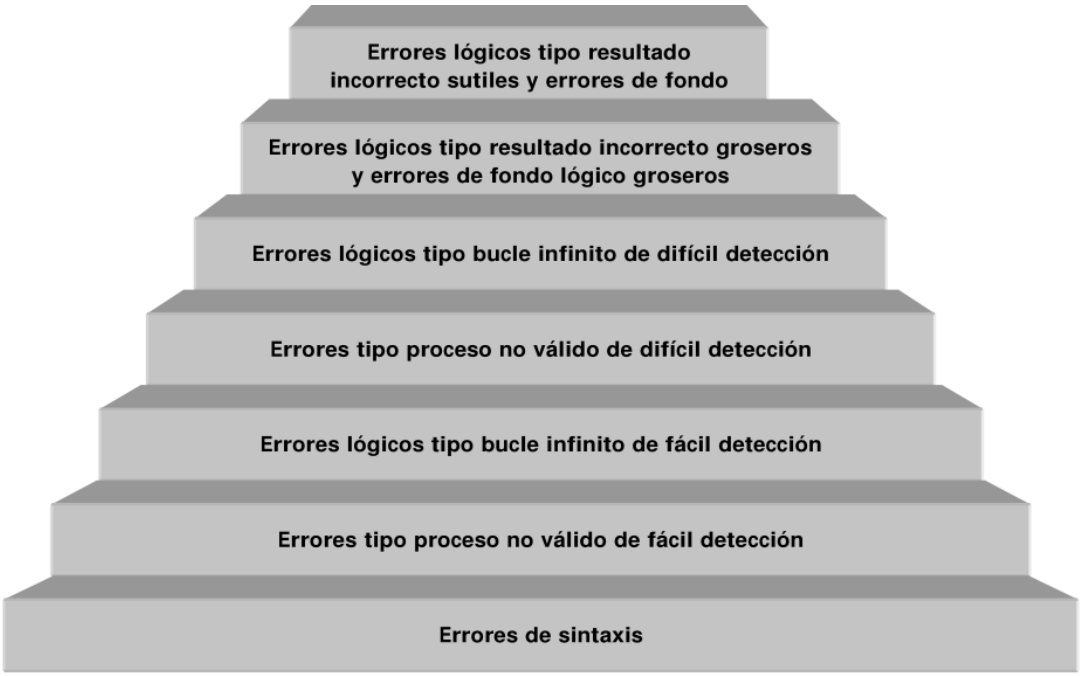


***Figura 1**: Pirámide de errores (Rancel, 2008).*

Como puede observarse en la figura 1 existen disímiles errores en el código fuente a la hora de desarrollar un programa informático, que van desde errores de sintaxis que impiden la ejecución del programa hasta los errores lógicos que hacen que se obtengan resultados distintos a los esperados.

El mecanismo de depuración se centra en los errores tipo lógicos de resultado incorrecto, para ello genera toda la información enviada por los agentes de

configuración en ficheros, con el objetivo de ser revisados por los probadores del sistema.

1.6 Análisis de soluciones existentes

Tanto en el ámbito nacional como internacional no se encontraron evidencias de sistemas SCADA que presenten mecanismo de depuración. En el transcurso de la investigación el autor pudo percatarse que los errores son tratados por las herramientas de depuración presentes en el software de programación utilizado en su desarrollo. Entre los sistemas SCADA que cumplen esta característica se encuentra: PlantScape quien presenta una herramienta de ingeniería del sistema, Control Builder, además incorpora una librería con más de 50 bloques de control analógico predefinidos. (HoneyWellsp, 2000) Otro SCADA analizado fue el Intouch, este sistema permite configurar alarmas y establecerles 999 niveles de prioridad en 8 niveles de jerarquía.

Por otra parte se encontraron algunos sistemas que son desarrollados con sus propias herramientas de depuración, tal es el caso del SCADA LabView, este software contiene potentes herramientas de depuración que identifican las áreas problemáticas en su código. Puede encontrar dos tipos generales de bugs de software: aquellos que impiden que el programa se ejecute y aquellos que generan malos resultados o comportamiento incorrecto. (ni.com, 2014)

En la investigación no fue posible estudiar el código fuente de los sistemas anteriormente mencionados debido a que son privativos, además hay que mencionar sus altos precios de adquisición y en muchos casos vienen acompañados del hardware necesario, haciéndolos aún más difíciles de adquirir. Por lo mencionando con anterioridad se decidió la realización de un mecanismo de depuración propio.

1.7 Herramientas y tecnologías

Para el desarrollo de la solución el entorno de trabajo será el utilizado en el desarrollo del módulo de configuración, debido a que la solución propuesta formaría parte del mismo y por ende debe seguir las pautas de desarrollo empleadas.

A continuación se mencionan las herramientas utilizadas en dicho módulo para el desarrollo de software, además se realiza un análisis de las tecnologías utilizadas en el desarrollo del mecanismo de depuración como propuesta de solución.

1.7.1 Eclipse

Es un entorno de desarrollo integrado (IDE), de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Provee todas las herramientas y funciones necesarias para el trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar (Ceca, 2013). Este IDE al no tener todas las funcionalidades incluidas no consume memoria innecesaria, contribuyendo a la velocidad del desarrollo.

1.7.2 C++

Es un lenguaje imperativo orientado a objetos derivado del C. Posee características superiores a otros lenguajes. Las más importantes son: programación orientada a objetos, portabilidad, brevedad, programación modular, compatibilidad con C y velocidad (Sierra, 2012). En este trabajo se utilizó debido a sus múltiples ventajas, entre ellas el hecho de que como su ancestro C, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción. Además, la definición "oficial" del lenguaje nos dice que C++ es un lenguaje de propósito general basado en el C, al que se han añadido nuevos tipos de datos, clases, plantillas, mecanismo de

excepciones, sistema de espacios de nombres, funciones inline, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería.

1.7.3 Formato de los ficheros generados por el mecanismo de depuración.

Para el desarrollo de la propuesta de solución fue necesario realizar un estudio de los posibles lenguajes capaces de representar información en un formato que facilite su comprensión. Dentro de los lenguajes se analizaron la Notación de Objetos de JavaScript (JSON) por sus siglas en inglés JavaScript Object Notation, el Lenguaje de Marcas Extensible (XML) por sus siglas en inglés Lenguaje de Marcas Extensible (XML) y el lenguaje de marcas de hipertexto (HTML) por sus siglas en inglés HyperText Markup Language. En la figura 2 se muestra el resultado del análisis realizado. Como criterios a evaluar se tuvieron: el objetivo para el cual fue creado el lenguaje, si se puede asociar un estilo visual y si el lenguaje posee funcionalidades para facilitar la revisión de la información.

	XML	JSON	HTML
OBJETIVO DEL LENGUAJE	Describir información	Notación de objetos de JavaScript	Mostrar información
INCORPORAR ESTILO VISUAL	si	no	si
VISUALIZACIÓN EN NAVEGADOR WEB	si	no	si
FACILIDADES PARA REVISIÓN	si	no	no

Figura 2: *Criterios de selección del formato para los ficheros generados. Elaboración propia*

Con el análisis realizado se arriba a las siguientes consideraciones:

- El JSON no permite su visualización en un navegador web.
- El lenguaje XML permite la visualización en un navegador web con un estilo visual asociado, sin embargo se muestran elementos propios del lenguaje

como las etiquetas utilizadas, siendo esta información innecesaria en el proceso de revisión y análisis de los ficheros.

- El lenguaje HTML es un lenguaje utilizado para mostrar información en ambientes visuales agradables en un navegador web, no presenta funcionalidades que agilicen el proceso de revisión, pero permite la incorporación de otros lenguajes como el JavaScript con el que se puede manejar toda información contenida en un fichero.

Por todo lo anteriormente planteado se toma como formato para los ficheros generados por el módulo de configuración el HTML.

1.7.4JavaScript

Es un lenguaje de programación que se utiliza para crear páginas web dinámicas. Permite incorporar efectos como aparición y desaparición de texto, animaciones, acciones que se activan al pulsar botones u otros elementos y ventanas con mensajes de aviso al usuario. JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos; con el uso de JavaScript se puede probar directamente en cualquier navegador el código, sin necesidad de procesos intermedios (Pérez, 2008)

Entre las principales ventajas que posee este lenguaje de programación se encuentran (Brandendaugh, 2000):

- JavaScript es el lenguaje de programación que más se utiliza en la web.
- La mayoría de los navegadores web pueden trabajar con él.
- Este lenguaje se puede hacer cargo de una gran parte de las funciones del cliente de las cuáles se encargaba el servidor, siendo un ejemplo de esto las validaciones.

El lenguaje JavaScript es parte esencial en el desarrollo del mecanismo de depuración como propuesta de solución debido a que será el encargado de manejar la información dando la posibilidad de contraer o expandir datos en específico.

1.7.5 CSS

Hoja de estilo en cascada (*CSS)* por sus siglas en inglés cascading style sheets es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas. (librosWeb.es, 2013)

1.8 Metodología de desarrollo

El desarrollo de software sin dudas es una difícil tarea, sin embargo existen varias metodologías capaces de guiar este complejo proceso, por una parte están las llamadas metodologías tradiciones que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida (Penadés, y otros, 2006).

Por otra parte están las llamadas metodologías ágiles las cuales se encargan de valorar al individuo y las iteraciones del equipo más que a las herramientas o los procesos utilizados, hacen mucho más importante crear un producto software que funcione que escribir mucha documentación. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad (Penadés, y otros, 2006). Para el desarrollo de la propuesta de solución se plantea, por parte del autor, la utilización de una metodología de desarrollo ágil debido a que el sistema que se desea implementar cumple con estas características mencionadas. Además el cliente forma parte del equipo de desarrollo.

1.8.1 SCRUM

Metodología ágil desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante

los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (Penadés, et al., 2006).

1.8.2 Programación extrema

Programación extrema (XP) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos muy cambiantes, y donde existe un alto riesgo técnico (Penadés, y otros, 2006).

El ciclo de vida de un proyecto XP es muy dinámico como se puede apreciar en la figura 3, basado en pequeñas iteraciones en las cuales se llevan a cabo recursivamente los procesos de análisis, diseño, implementación y pruebas.

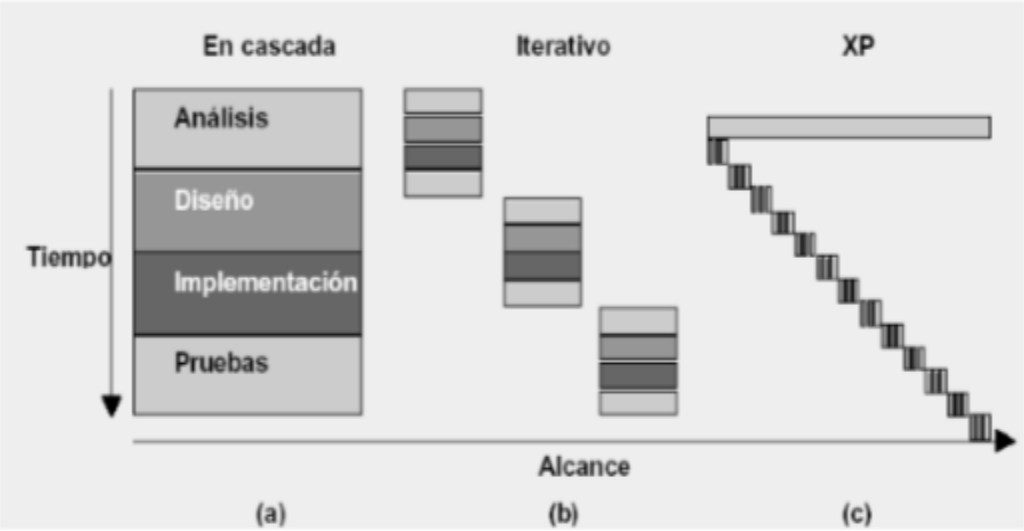


Figura 3: *Ciclos de vida de XP (Escribano, 2002).*

1.8.3 Selección de la metodología de desarrollo

Las metodologías ágiles de desarrollo XP y SCRUM comparten características similares como puede apreciarse en las figuras 4 y 5. Sin embargo debido a que SCRUM está centrada en la gestión de proyectos y XP por su parte se centra en la programación y obtención de un producto final, además genera documentación del software desarrollado, se selecciona esta última como metodología de desarrollo para la presente investigación a través de las fases propuestas por (Escribano, 2002): planificación, diseño, desarrollo y pruebas.

➢ Por las características del Proyecto

Modelo de Proceso	Tamaño del Proceso	Tamaño del Equipo	Complejidad del Problema
XP	**Pequeño / Medio**	**Pequeño**	Medio / Alto
SCRUM	**Pequeño / Medio**	**Pequeño**	Medio / Alto

Figura 4*: Comparación entre XP y SCRUM (Figueroa, 2010)*

➢ Por la curva de Aprendizaje

Modelo de Proceso	Curva de aprendizaje	Herramienta de integración	Soporte Externo
XP	Rápida	No mencionado	Algún Soporte Disponible
SCRUM	Rápida	No mencionado	Algún Soporte Disponible

***Figura 5**: Comparación entre XP y SCRUM (Figueroa, 2010)*

Consideraciones parciales

Después del análisis del estado del arte se puede llegar a las siguientes consideraciones parciales.

- El formato para los ficheros generados es HTML.
- El mecanismo de depuración como solución propuesta está centrado en la depuración de errores tipo lógicos de resultados incorrectos.
- Las herramientas a utilizar para el desarrollo de la solución serán las utilizadas en el entorno de trabajo del módulo de configuración y las tecnologías: HTML, JavaScript y C++ siguiendo las pautas de la metodología de desarrollo XP.

Capítulo 2: Fases planificación y diseño.

Introducción

En el presente capítulo se precisan un conjunto de elementos para conformar la propuesta de solución de la presente investigación. Se realiza un análisis de las características, componentes y particularidades; además de describir el funcionamiento general del mecanismo de depuración que se desea implementar. Se realiza una descripción de los patrones utilizados. Se aplica la metodología de desarrollo seleccionada y se generan los artefactos correspondientes a las fases tratadas en este capítulo.

2.1 Características del mecanismo de depuración

Para determinar las características que debía cumplir el mecanismo de depuración se realizó una entrevista a los trabajadores del módulo de configuración, cliente de la propuesta de solución, obteniéndose al final las siguientes:

- Posibilita que el módulo de configuración del SCADA Galba funcione en modo depuración sin necesidad de reiniciarlo.
- Genera ficheros en formato HTML para cada módulo existente en el SCADA Galba con la configuración referente a los mismos.
- Los ficheros generados pueden ser visualizados en un navegador web cualquiera.
- Los ficheros generados presentan un estilo visual resaltando en otro color que información puede ser contraída o expandida.

2.2 Fase de planificación

En esta fase, los clientes plantean a grandes rasgos las Historias de Usuario (HU) que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto (Joskowicz, 2008).

2.2.1 Descripción de las HU

Las HU sustituyen a los documentos de especificación funcional, y a los "casos de uso". Estas "historias" son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar (Pérez, 2012). Las principales características de las historias de usuario son según (Beck, 1999) independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables.

Para la confección de las HU se realizó diferentes entrevistas a desarrolladores del módulo de configuración del SCADA Galba. Si bien el cliente no fue quien escribió personalmente las HU, fue él quien diseñó su contenido, asignó la prioridad de cada una de ellas y dirigió la redacción de las mismas.

El cliente se encargó de asignarle una prioridad a cada HU. El equipo de desarrollo por su parte revisó esta prioridad analizando la dependencia entre HU y asignó el costo de cada una de ellas, este se traduce en las semanas para su desarrollo. También, es importante destacar, que las HU nuevas pueden describirse en cualquier momento, con esto se comprueba la flexibilidad de la metodología.

Las HU se representan mediante tablas las cuales contienen las siguientes secciones:

- **Código:** Las siglas de HU más un numero consecutivo, este permite la historia de usuario.
- **Nombre:** Nombre que identifica la HU.
- **Referencia:** Es el conjunto de códigos de las diferentes HU de las cuales depende actualmente la que se encuentra en desarrollo.
- **Prioridad:** Esta característica es dada por el cliente con los valores: alta, media o baja en dependencia de la importancia en que desean ser implementadas.
- **Iteración asignada:** Número de la iteración en la cual se desarrollara la HU.
- **Puntos estimados:** Tiempo estimado en semanas que se le asignara.

- **Descripción:** Breve descripción del proceso que define la HU.
- **Observaciones:** Alguna acotación importante de señalar acerca de la HU.

Se confeccionó un total de 6 HU, tomando para los puntos estimados la unidad como una semana de trabajo. En las tablas de la 1 a la 6 se presentan dichas HU. Las HU 1,2,3 y 4 son las encargas de describir el proceso de la generación de los ficheros con la información enviada por el módulo de configuración a los restantes módulos del SCADA Galba y las HU 5 y 6 describen todo el proceso de adoptar el modo depuración sin reiniciar el módulo de configuración. Ambos constituyen procesos de alta complejidad dentro del desarrollo del mecanismo de depuración es por ello que todas las HU tendrían una prioridad alta.

***Tabla 1**: HU1 Crear fichero HTML.*

Historia de usuario	
Código: HU 1	**Nombre:** Crear fichero HTML.
Responsable: Raudel Arencibia Ramírez	**Prioridad:** Alta
Iteración Asignada: 1	**Puntos estimados:** 0.3
Referencia:	
Descripción: El programador introduce la ruta física donde se generará el fichero HTML y el sistema crea en dicha ruta un fichero HTML en blanco.	
Observaciones: **1.** Se crean uno o más ficheros para cada módulo.	

***Tabla 2** HU2 Introducir información al fichero HTML.*

Historia de usuario	
Código: HU 2	**Nombre:** Introducir información al fichero HTML.

Responsable: Raudel Arencibia Ramírez	**Prioridad:** Alta
Iteración Asignada: 1	**Puntos estimados:** 2.6
Referencia: HU1	
Descripción: El sistema escribe la información que se envió a través de un agente de configuración a un módulo específico en el fichero HTML**.** La información está conformada por el nombre del módulo al cual pertenece el fichero HTML generado, seguido de la configuración definida para dicho módulo.	
Observaciones: Se escriben dos ficheros para cada módulo, uno contiene los cambios realizados, el otro contiene toda la configuración referente al módulo**.**	

***Tabla 3**: HU3 Visualizar información en forma de árbol.*

Historia de usuario	
Código: HU 3	**Nombre:** Visualizar información en forma de árbol.
Responsable: Raudel Arencibia Ramírez	**Prioridad:** Alta
Iteración Asignada: 1	**Puntos estimados:** 0.5
Referencia: HU1, HU2	
Descripción: El fichero generado por el sistema es visualizado en un navegador WEB en forma de árbol, donde la raíz es el nombre del módulo, los elementos de configuración de dicho módulo serían los nodos y los atributos asociados a estos representarían las hojas.	
Observaciones:	

***Tabla 4**: HU4 Contraer-expandir información.*

Historia de usuario		
Código: HU 4	**Nombre:** Contraer-expandir información.	
Responsable: Raudel Arencibia Ramírez		**Prioridad:** Alta
Iteración Asignada: 1		**Puntos estimados:** 0.5
Referencia: HU1,HU2,HU3		
Descripción: El fichero HTML, al ser visualizado en un navegador web, brinda la posibilidad de contraer-expandir información específica. Esta información está conformada por la configuración definida para el módulo al cual perteneces el fichero.		
Observaciones:		

***Tabla 5**: HU5 Actualizar parámetros de configuración del servidor.*

Historia de usuario		
Código: HU 5	**Nombre**: Actualizar parámetros de configuración del servidor.	
Responsable: Raudel Arencibia Ramírez		**Prioridad:** Alta
Iteración Asignada: 2		**Puntos estimados:** 0.8
Referencia:		

Descripción: Se realiza una lectura de los parámetros de configuración del servidor. Los modificados se actualizan con sus nuevos valores.
Observaciones: Los parámetros de configuración del servidor deben ser leídos estando en funcionamiento el mismo.

Tabla 6: *HU6 Activar-desactivar modo de depuración.*

Historia de usuario	
Código: HU 6	**Nombre**: Activar-desactivar modo de depuración.
Responsable: Raudel Arencibia Ramírez	**Prioridad:** Alta
Iteración Asignada: 2	**Puntos estimados:** 2.6
Referencia: HU5	
Descripción: El sistema envía una notificación a los agentes conectados al servidor de configuración para que adopten el modo de depuración o cambien al modo normal en caso de ya estar funcionando en dicho modo. Si se adoptó el modo depuración configuración comienza a generar los ficheros HTML con la información enviada a cada módulo, o deja de hacerlo en caso contrario.	
Observaciones:	

2.2.2 Estimación del esfuerzo por HU

En la fase de planificación se priorizan las HU y se acuerda el alcance de cada una de las entregas o release. Los desarrolladores estiman cuánto esfuerzo requiere cada historia y a partir de allí se define el plan de iteraciones. La primera iteración

crea un sistema con la arquitectura del sistema completo. Esto es alcanzado seleccionando las historias que harán cumplir la construcción de la estructura para el sistema completo. El cliente decide las historias que se seleccionan para cada iteración. Las pruebas funcionales creadas por el cliente se ejecutan al final de cada iteración. (Joskowicz, 2008)

A continuación se presenta la tabla 7 donde se resume la estimación del esfuerzo realizada por parte de los desarrolladores y la duración total que tendrá el desarrollo del mecanismo de depuración como solución propuesta.

***Tabla 7**: Estimación de esfuerzo por HU.*

Historia de usuario	Puntos de estimación(semanas)
HU1 Crear fichero HTML.	0.3
HU2 Introducir información al fichero HTML.	2.6
HU3 Visualizar información en forma de árbol.	0.5
HU4 Contraer-expandir información.	0.5
HU5 Actualizar parámetros de configuración del servidor.	0.8
HU6 Adoptar modo de depuración.	2.6
Total	**7.3**

2.2.3 Plan de iteraciones

Luego de identificar y redactar cada una de las HU y de la estimación del esfuerzo necesario para realizarlas, se debe conformar el plan de iteraciones. Las HU seleccionadas para cada iteración son desarrolladas y probadas de acuerdo al orden preestablecido.

El desarrollo del mecanismo fue dividido en 2 iteraciones teniendo en cuenta los procesos que comprende el mecanismo de depuración los cuales son: adoptar el modo depuración sin reiniciar el módulo de configuración y la generación de los

ficheros HTML cuando el módulo funciona en dicho modo. A continuación se describen cada una de las iteraciones.

Iteración 1: En esta iteración se desarrollan las historias de usuario relacionadas con el proceso de generar los ficheros con la configuración enviada a cada módulo. Se realiza una entrega funcional al finalizar la iteración en la cual se puede realizar el proceso de revisión y análisis por parte de los probadores del sistema.

Iteración 2: En esta iteración se desarrolla las historias de usuario relacionadas con el funcionamiento en modo depuración del módulo de configuración sin tener que reiniciar el mismo.

Para aproximar el tiempo de ejecución de cada iteración, se tomó como medida que cada semana constaba de 5 días en los que se trabajaban 6 horas sin distracciones. En la tabla 8 se muestra el plan de iteraciones, este incorpora el tiempo estimado para cada una de las iteraciones y las HU que se van a desarrollar.

***Tabla 8**: Plan de iteraciones*

Iteración	Historias de usuario	Puntos de estimación
1	HU1 Crear fichero HTML.	3.9
	HU2 Introducir información al fichero HTML.	
	HU3 Visualizar información en forma de árbol.	
	HU4 Contraer-expandir información.	
2	HU5 Actualizar parámetros de configuración del servidor.	3.4
	HU6 Activar-desactivar modo de depuración.	

2.2.4 Plan de entregas

El plan o cronograma de entregas establece qué HU son agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma es el resultado de una

reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. (Joskowicz, 2008)

A partir del plan de iteraciones analizado en el acápite anterior, y en correspondencia con el mismo se realiza el plan de entregas, que se muestra en la tabla 9. En el cual se proponen 2 versiones funcionales y una última entrega de iteraciones del producto el 25 de abril, para dar paso a la fase de pruebas.

***Tabla 9**: Plan de entregas*

Historia de usuario	1ra iteración 17 de marzo de 2015	2da iteración 25 de abril de 2015
HU1 Crear fichero HTML.	V1.0	
HU2 Introducir información al fichero HTML.		
HU3 Visualizar información en forma de árbol.		
HU4 Contraer-expandir información.		
HU5 Actualizar parámetros de configuración del servidor.		V 2.0
HU6 Activar-desactivar modo de depuración.		

2.3 Fase de diseño

La metodología XP sugiere que hay que conseguir diseños simples y sencillos, procurando hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible que a la larga costará menos tiempo y esfuerzo desarrollar.

2.3.1 Estructura del módulo de configuración

El módulo de configuración presenta una estructura por paquetes, la misma puede verse representada en la figura 6**,** donde cada paquete tiene una funcionalidad específica y al relacionarse entre sí logran que se cumpla el objetivo del módulo al brindarle toda la información base para su funcionamiento a los restantes módulos.

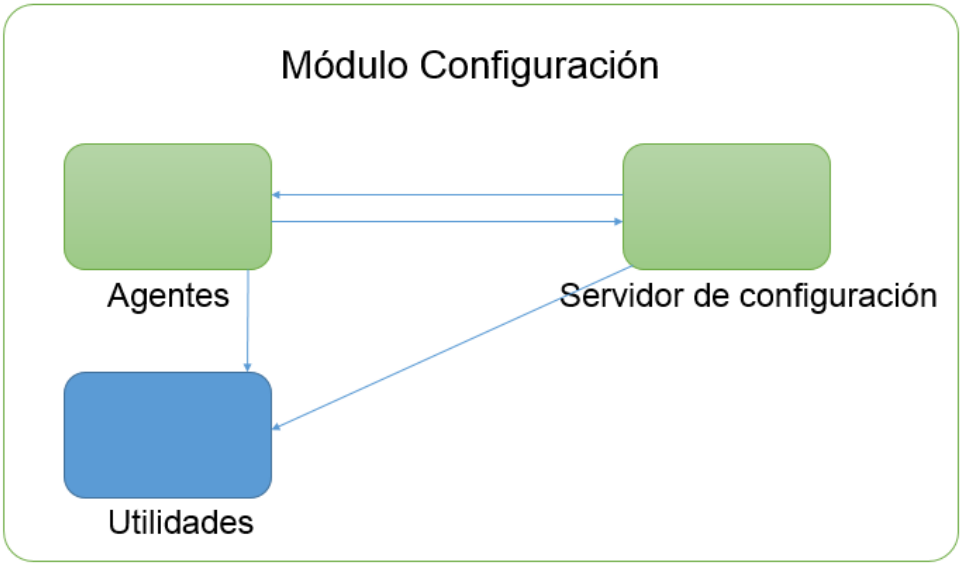


Figura 6: *Estructura en paquetes del módulo de configuración*

Para el desarrollo del mecanismo de depuración se mantendrá dicha estructura, agregando clases y funcionalidades necesarias en los paquetes Agentes y Servidor de configuración.

En el paquete Servidor de configuración se desarrollarán funcionalidades que hacen que el módulo pueda cambiar del modo normal al modo depuración mediante el comando reload en las opciones de inicio del mismo. Una vez que el módulo se encuentra en modo depuración comienza el proceso de la generación de los ficheros HTML o termina en caso contrario.

Dentro del paquete Agentes están contenidos todos los agentes de configuración presente en el SCADA Galba, en cada uno de ellos se desarrollará una funcionalidad encargada de escribir en el fichero HTML toda la información

manejada por los mismos. Cada vez que ocurra un cambio en la configuración general del SCADA quedará registrado en ficheros HTML.

2.3.2 Patrón de arquitectura del módulo de configuración

El módulo de configuración se desarrolla siguiendo la arquitectura cliente-servidor donde los agentes de configuración actúan como clientes y solicitan permiso para leer o escribir en el servidor de configuración quien atiende las peticiones y otorga los permisos requeridos para la realización de las estas operaciones. El mecanismo de depuración será desarrollado siguiendo este patrón de arquitectura presente en el módulo. La figura 7 evidencia la arquitectura mencionada con anterioridad al representar las relaciones existentes entre los agentes y el servidor de configuración en el proceso de guardar una configuración en frío.

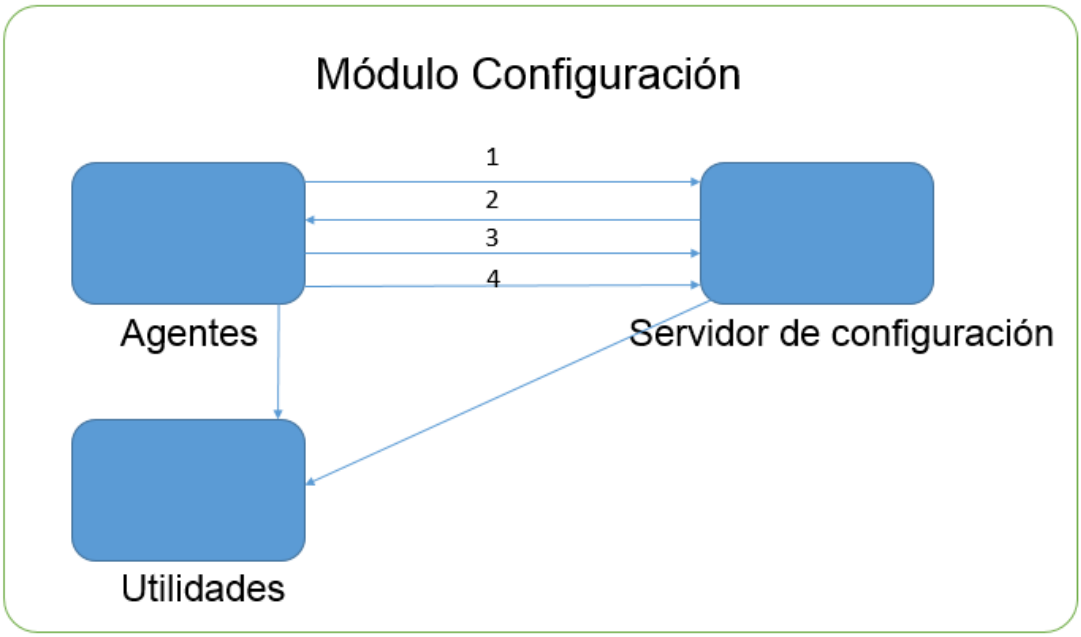


Figura 7: *Proceso de guardar la configuración en frío.*

En este proceso existen cuatro mensajes de sincronización entre el paquete agentes y el servidor de configuración como puede observarse en la figura. Para una mejor comprensión del proceso a continuación se explican cada uno de estos mensajes.

1. El agente solicita al servidor de configuración permiso para almacenar la nueva configuración en el formato persistente.
2. Cuando sea posible almacenar la nueva configuración (no existe ningún otro agente obteniendo información del formato persistente o modificando la misma) el Servidor de configuración procede a darle permiso al agente para que se encargue de almacenar la nueva configuración.
3. El agente almacena la configuración en el formato persistente y procede a generar un fichero con la información enviada a un módulo en específico en caso de estar el módulo de configuración funcionando en modo depuración.
4. El agente le comunica al Servidor de configuración que ha culminado de almacenar la nueva configuración.

Dentro de este proceso se encuentra el mecanismo de depuración, específicamente en los agentes de configuración. Esta ubicación posibilita que cada vez que se realicen cambios en la configuración (configuraciones en frío, en caliente u operacional) toda la información enviada a los diferentes módulos sea guardada en los ficheros HTML generados (ver mensaje 3). Al revisarlos se podrá comprobar si el módulo está funcionando de forma correcta.

2.3.3 Diseño de la interfaz de los ficheros

El mecanismo de depuración no presenta una interfaz de usuario, sin embargo los ficheros generados presentan un diseño, el cual puede observarse en la figura 8. Al abrir un fichero este se visualiza en forma de árbol, donde la raíz será el nombre del módulo al cual se envió la información seguido de la palabra Info. Los nodos estarán compuestos por objetos y conceptos correspondientes a cada módulo en específico, mientras que las hojas serán los atributos de los nodos. Inicialmente el fichero se muestra contraído, la persona encargada de revisarlos determinará qué información expandir-contraer según sea necesario.

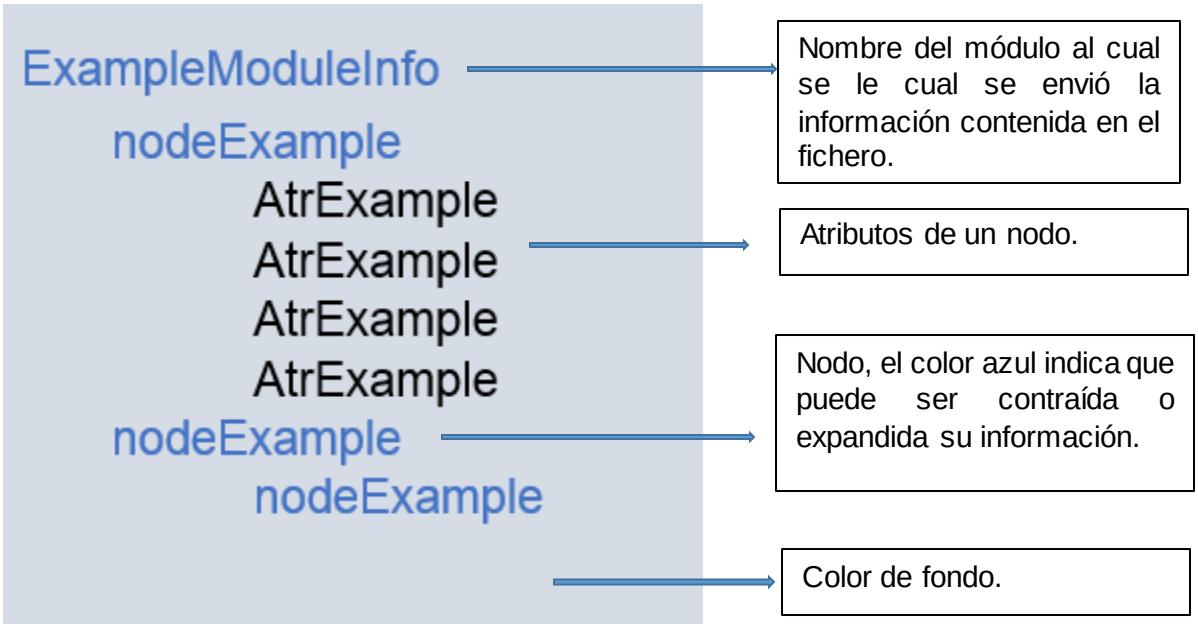


***Figura 8**: Diseño de los ficheros generados*

2.3.4 Patrones generales de software para asignación de responsabilidades

Los Patrones de Software para la Asignación General de Responsabilidad (GRASP) por sus siglas en inglés General Responsabilty Assignment Software Patterns describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. En la implementación de la aplicación se utilizaron los siguientes:

- **Alta cohesión:** Es la meta principal que ha de tenerse en cuenta en cada momento en todas las decisiones de diseño. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Mejora la claridad y la facilidad con que se entiende el diseño. Se simplifican el mantenimiento y las mejoras en funcionalidad. A menudo genera un bajo acoplamiento (Larman, 2003). Este patrón fue utilizado en el diseño de la aplicación de manera general, siguiendo la premisa de que cada clase debe contener operaciones que resuelvan necesidades a fines con ellas.

- **Bajo acoplamiento:** Es un principio que debemos recordar durante las decisiones de diseño: es la meta principal que es preciso tener presente siempre. Estimula asignar una responsabilidad de modo que su colaboración no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios (Larman, 2003). Este patrón fue utilizado para el diseño de las clases utilizadas en el mecanismo de depuración.
- **Experto:** Es un patrón que se usa más que cualquier otro para asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la intuición de que los objetos hacen cosas relacionadas con la información que poseen (Larman, 2003). Este patrón fue utilizado de manera general en el desarrollo del mecanismo de depuración, un ejemplo lo constituye la clase SecurityModuleInfo es la encargada de manejar toda la información referente al módulo de seguridad.

2.3.5 Tarjetas clase-responsabilidad-colaboración

En el anexo 1 se presentan las tarjetas clase-responsabilidad-colaboración (CRC) del sistema.

Consideraciones parciales

- El mecanismo de depuración está compuesto por 6 historias de usuario, las cuales permitieron tener la base para la implementación de la aplicación.
- En la fase de planificación se determinó que la duración del desarrollo del mecanismo de depuración como solución propuesta es de 8 semanas, para lo cual contará con 2 iteraciones, al finalizar cada una se tendrá una entrega funcional de dicho mecanismo.

- El desarrollo del mecanismo de depuración como solución propuesta estuvo basado en el patrón de arquitectura cliente-servidor utilizado en el desarrollo del módulo de configuración del SCADA Galba.

Capítulo 3: Fases desarrollo y pruebas.

Introducción

En el presente capítulo se aborda la implementación del sistema perteneciente a la fase desarrollo y las pruebas realizadas al mecanismo de depuración obtenido pertenecientes a la fase pruebas. Se identifican y definen las tareas de ingeniería y se lleva a cabo las pruebas de aceptación realizadas al sistema con el objetivo de que cumpla con lo esperado.

3.1 Fase de desarrollo

Dentro de la fase de iteraciones se encuentra la implementación o el desarrollo del código el cual se sustenta en buenas prácticas planteadas por el ciclo de vida de XP para esta etapa entre ella se encuentran: disponibilidad del cliente, uso de estándares, programación dirigida por las pruebas ("Test-driven programming"), programación en pares, integraciones permanentes y ritmo sostenido.

3.1.1Tareas de ingeniería o desarrollo

Asociado a cada iteración se encuentra la planificación de las tareas de ingeniería o programación, cada HU se transforma en estas tareas que son desarrolladas por programadores, dentro del equipo de desarrollo, aplicando la práctica de la programación en parejas. Para cada iteración se realizó la distribución de tareas en correspondencia con las HU que se desarrollaron.

Tabla 18: *Tareas de ingeniería*

Historia de usuario	Tareas de ingeniería
HU1 Crear fichero HTML.	1. Desarrollo de una función para crear un fichero HTML.

HU2 Introducir información al fichero HTML.	1. Desarrollo de la clase principal. 2. Desarrollo de una función para escribir en fichero HTML.
HU3 Visualizar información en forma de árbol.	1. Desarrollo de un estilo visual.
HU4 Contraer-expandir información.	1. Desarrollo de una función JavaScript.
HU5 Actualizar parámetros de configuración del servidor.	1. Modificar el script de configuración. 2. Desarrollo de una función para leer parámetros. 3. Desarrollo de una función para actualizar parámetros.
HU6 Activar-desactivar modo de depuración.	1. Desarrollo de una función de aviso. 2. Desarrollo de una función que active-desactive modo depuración.

En el anexo 2 se muestran las tareas de ingeniería realizadas.

3.1.2 Estilos de codificación

Los estilos de codificación o programación definen la estructura y apariencia física del código, lo que facilita su comprensión, mantenimiento y lectura. Los estilos de codificación y documentación explicados en este capítulo son aplicados en el

CEDIN, esto justifica el empleo de los mismos en el mecanismo de depuración como solución propuesta a la problemática existente en el módulo de configuración del SCADA Galba.

3.1.3 Definición de clases

Las declaraciones de clases tienen su llave de apertura una línea más abajo de la declaración y el nombre de la clase comienza con mayúscula. Los nombres de las mismas son sustantivos singulares, que deben reflejar qué hacen y no cómo lo hacen. En la siguiente figura 9 se muestra un ejemplo.

```
/**
 * @brief Clase encargada de codificar la información enviada por un agente en HTML.
 * @author Raudel Arencibia Ramírez rarencibiar@estudiantes.uci.cu
 * @date 22/01/2015
 */
class CodificatorHTML
{

public:
    ...
    CodificatorHTML(){}
    ...
}
```

Figura 9: *Definición de clases*

3.1.4 Definición de métodos

Los nombres de los métodos son frases que incluyen verbos, dado que los mismos generalmente son el producto de concatenar varias palabras, se debe emplear la primera palabra en minúscula, mayúscula para denotar la letra de inicio de cada una de las palabras restantes por las que esté formado y minúscula para las letras intermedias. Los atributos pasados por parámetro se separarán por una coma y un espacio después de ésta en caso de existir más de uno.

3.1.5 Estructuras de control

Al hacer uso de las estructuras de control se deben emplear las letras i, j, k, l, m, p, q, r para contadores en ciclos. Emplear correctamente los tipos de ciclos: si es al menos una vez usar do-while, si es ninguna o más veces usar while-do, y si se

conoce el número exacto de ciclos usar for. En la siguiente figura 10 se muestra un ejemplo.

```
if ( deletedGOPs )
{
    for ( uint32_t i(0); i < deletedGOPs->size(); ++i )
    {

        std::string idValueDeletedGop="Atr: id Value:";
        std::string valueIdValueDeletedGop;
        Utils::ToStdString::uintToStdString(deletedGOPs->at(i),valueIdValueDeletedGop);
        idValueDeletedGop+=valueIdValueDeletedGop;
        obj->writeFile(pFile,idValueDeletedGop);
    }

}
```

Figura 10: *Estructuras de control*

3.1.6 Codificación en general

La codificación de forma general estuvo guiada por los siguientes aspectos (Lorenzo, 2010).

- Se adopta el estilo de bloques de documentación JavaDoc.
- Para hacer una descripción breve se adopta el uso del comando \brief ó @brief.
- A la hora de hacer una descripción de los argumentos de métodos y funciones esta se hará en línea, es decir, luego de la declaración de cada uno de los argumentos se da una breve descripción de cada uno de ellos.
- Se debe especificar el nombre del autor y la fecha de creación de cualquier estructura en un código, para ello se utilizan los comandos @author y @date para el nombre del autor y la fecha respectivamente.

3.2 Fase de pruebas

La metodología XP propone la aplicación de las pruebas de aceptación utilizadas para evaluar si se obtuvo las funcionalidades deseadas por parte del cliente.

3.2.1 Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. Las pruebas de aceptación son consideradas como "pruebas de caja negra", Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos (Joskowicz, 2008).

Como criterio de aprobación de cada iteración se tomó que el 100% de los casos de prueba sean exitosos para pasar de iteración. El objetivo de estas pruebas no es tener un conjunto de casos escritos que cubran el 100% del código, sino poder realizarle pruebas al sistema desde el punto de vista del usuario.

Para la realización de cada una de las pruebas de aceptación se siguieron una serie de pasos que se muestran a continuación:

- Identificar todas las acciones en la HU.
- Para cada acción escribir al menos una prueba.
- Para algunos datos, reemplazar las entradas que hacen que la acción ocurra y llenar en la casilla resultados esperados los resultados obtenidos.
- Para otros datos, reemplazar las entradas que hacen que la acción falle, y registrar los resultados.

Para representar las pruebas de aceptación se definieron los siguientes elementos:

- **Código:** Representa al caso de prueba, incluye el número de HU, de la prueba y si posee diferentes escenarios.
- **HU:** Número de la HU a la cual pertenece.
- **Nombre:** Junto al código conforma el identificador del caso de prueba.
- **Descripción:** Acción que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- **Entrada:** Incluye las entradas necesarias para realizar el sistema.
- **Pasos de ejecución:** Pasos para realizar el caso de prueba**.**

- **Resultados Esperados:** Descripción de la respuesta del sistema ante el caso de prueba.
- **Resultado Obtenido:** Respuesta visual del sistema después de realizar el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

3.2.2 Pruebas de aceptación de la primera iteración

En esta primera iteración se realizaron pruebas a la generación de los ficheros HTML. En la tabla 29 se muestra la prueba realizada a la generación de un fichero HTML en una ruta física especificada por el programador.

***Tabla 29 HU1_P1**: Crear fichero HTML*

Código: HU1_P1	HU: 1
Nombre: Crear fichero HTML.	
Descripción: Se procede a iniciar los agentes de configuración del SCADA Galba.	
Condiciones de ejecución: Módulo de configuración en modo depuración.	
Entrada:	
Pasos de ejecución: 1. Se abre la herramienta Terminator y se divide en 6 consolas. 2. Se inicia el módulo de comunicación. 3. Se inicia el servidor de configuración. 4. Se inicia el módulo de seguridad. 5. Se inicia el módulo de adquisición. 6. Se inicia el módulo de histórico. 7. Se inicia el módulo de HMI editor.	

Resultado esperado: Al iniciarse los agentes de configuración el sistema crea un fichero HTML respectivamente a cada uno de los módulos en la dirección especificada por el programador.
Resultado obtenido: Se crearon los ficheros en el área de la generación de trazas definidas por el módulo.
Evaluación de la prueba: Satisfactoria

En el **anexo 3** Pruebas de aceptación de la primera iteración se muestran los demás casos de prueba desarrollados.

En esta primera iteración de pruebas se realizaron 3 etapas de pruebas con un total de 6 casos de prueba, en la primera etapa se detectaron 2 no conformidades representando 33.33%, luego de corregir los errores se realizó la segunda etapa de pruebas en la cual se obtuvieron 1 no conformidades representando un 16.7% de resultados insatisfactorios, después de la corrección de los mismos se pasó a desarrollar la tercera etapa de pruebas donde no se encontraron errores para un 100% de resultados satisfactorios cumpliendo así con el criterio de aceptación por lo que se decidió pasar a la segunda iteración de pruebas.

3.2.3 Pruebas de aceptación de la segunda iteración

En las tablas 35 y 36 se muestran las pruebas realizadas a la HU6 activar-desactivar modo depuración con lo cual se puede comprobar que la HU5 actualizar parámetros de configuración funciona correctamente debido a que es un proceso continuado.

Tabla 35 HU6_P1: *Cambio del modo normal al modo depuración mediante el comando reload*

Código: HU6_P1	HU: 6

Nombre: Cambio del modo normal al modo depuración mediante el comando reload
Descripción: Se ejecuta el script de inicio para el servidor de configuración con el comando reload.
Condiciones de ejecución: Servidor de configuración ya iniciado en modo normal.
Entradas: Script de inicio del módulo de configuración
Pasos: 1. Se cambia el parámetro modo de funcionamiento dentro del archivo de configuración del módulo y se pone en modo depuración. 2. Se ejecuta el script de inicio con el comando reload.
Resultado esperado: El módulo de configuración cambia al modo depuración sin tener que reiniciar el mismo.
Resultado obtenido: El módulo cambia al modo depuración.
Evaluación de la prueba: Satisfactoria

***Tabla 36 HU6_P2**: Cambio del modo depuración al modo normal mediante el comando reload*

Código: HU6_P2	HU: 6
Nombre: Cambio del modo depuración al modo normal mediante el comando reload	
Descripción: Se ejecuta el script de inicio para el servidor de configuración con el comando reload.	
Condiciones de ejecución: Servidor de configuración ya iniciado en modo depuración.	
Entradas: Script de inicio del módulo de configuración	

Pasos: 1. Se cambia el parámetro modo de funcionamiento dentro del archivo de configuración del módulo y se pone en modo normal. 2. Se ejecuta el script de inicio con el comando reload.
Resultado esperado: El módulo de configuración cambia al modo normal sin tener que reiniciar el mismo.
Resultado obtenido: El módulo cambia al modo normal.
Evaluación de la prueba: Satisfactoria

En la segunda iteración de pruebas se desarrollaron 2 etapas de pruebas con un total de 2 casos de pruebas. En la primera etapa se detectaron un total de 1 no conformidades representando 50% de resultados insatisfactorios, una vez corregidos, se desarrolló la segunda etapa de pruebas donde no se encontraron errores representando el 100% de resultados satisfactorios.

3.2.4 Análisis de los resultados obtenidos

Las no conformidades encontradas en cada una de las etapas de pruebas realizadas se clasificaron en errores en funcionalidades que no realizaban lo que estaba previsto. En la clase GOP, Profile y Resuorce pertenecientes al agente de seguridad, el sistema escribía en los ficheros de forma incorrecta, dejaba los atributos sin sus respectivos valores, en las clases GTH, DigitalPoint, PrefileGOP y ProfileUser pertenecientes al agente histórico, el sistema creaba nodos incorrectos. De forma general, en las clases se detectaron nodos que no expandían su información los cuales fueron solventados con el transcurso de las etapas de prueba. Además el módulo no activaba el modo depuración mediante el comando reload.

Una vez culminada la fase de pruebas puede apreciarse como quedó la relación de pruebas de aceptación de cada iteración en la figura 11, además se muestran el total de no conformidades encontradas en cada una de las fases por las cueles pasó el mecanismo de depuración.

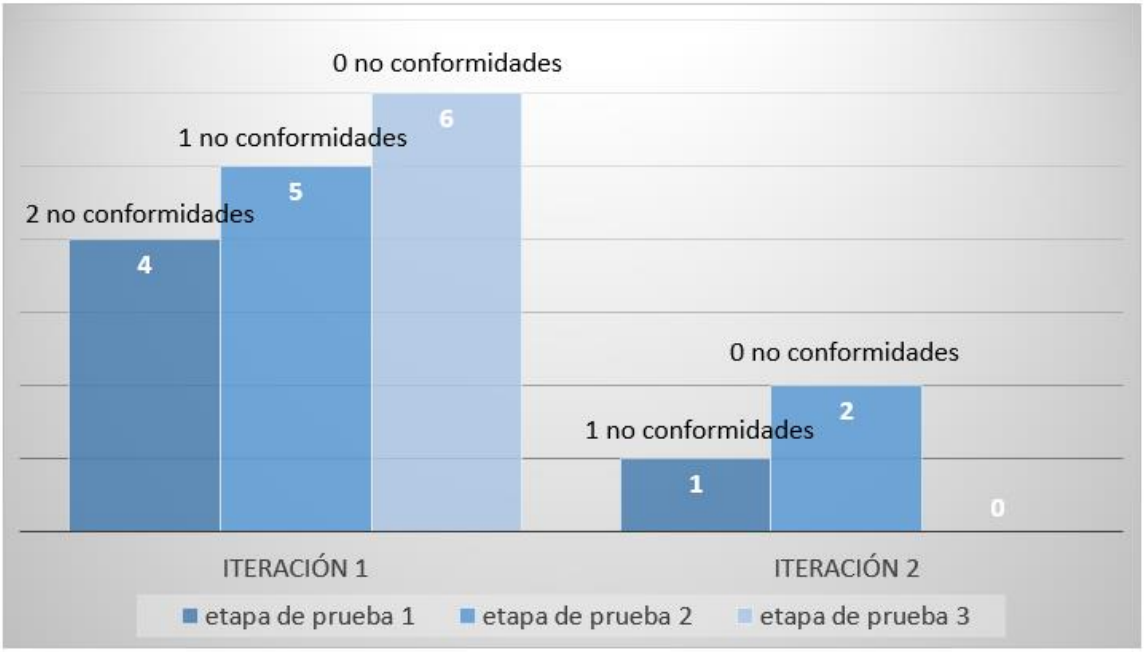


Figura 11: *Pruebas de aceptación satisfactorias por iteración.*

Las pruebas de aceptación se desarrollaron dentro del módulo de configuración. Al finalizar la fase de pruebas realizadas al mecanismo de depuración el cliente valoró el resultado obtenido mediante un aval, el mismo puede verse en el anexo 4 Carta de aceptación del cliente.

Consideraciones parciales

- En la fase de desarrollo se definió la codificación utilizada para las clases, métodos y estructuras de control definidas en el sistema SCADA Galba, además se desarrollaron las tareas de ingeniería.
- La realización de las pruebas de aceptación posibilitaron la obtención de un mecanismo de depuración aceptado por el cliente.

Conclusiones generales

Luego de realizado el trabajo se concluye lo siguiente:

- Se obtuvo un mecanismo de depuración que permite obtener la información enviada por el módulo de configuración a los restantes módulos del SCADA Galba.
- La propuesta de solución garantiza la disponibilidad del módulo de configuración al activar el modo depuración.
- La propuesta de solución genera ficheros HTML los cuales facilitan la revisión de la información enviada a cada uno de los módulos al presentar una estructura en forma de árbol.
- Al desarrollar el mecanismo de depuración siguiendo el patrón de arquitectura cliente servidor se puede integrar fácilmente al módulo de configuración.
- La propuesta de solución fue aceptada por el cliente mediante un aval emitido.

Recomendaciones

Se recomienda:

Adicionar una función JavaScript que indique si existen errores en un fichero HTML al abrirlo.

Añadir al fichero de configuración del módulo de configuración una opción que permita al probador del sistema mostrar la información en HTML o en texto en plano.

Bibliografía

Schwaber, Ken y Sutherland, Jeff . 2013. *La Guía Definitiva de Scrum: Las Reglas del Juego.* 2013.

Álvarez, Moisés. 2008. *Introducción a la Arquitectura del Guardián del ALBA .* 2008.

Amaya, Camilo Gutierrez. 2013. welivesecurity. [En línea] 2013. http://www.welivesecurity.com/la-es/2013/01/25/criticos-sistemas-scada/.

arPUG, Grupo de usurios postgresSQL de Argentina. 2012. postgressql.org.ar. [En línea] 2012. http://www.postgresql.org.ar/trac/wiki/PgAdmin.

automatas.org. 2006. automatas. [En línea] 2006. http://www.automatas.org/redes/scadas.htm.

Beck, kent. 1999. *Extreme Programming Explained.* 1999.

Boyer, Stuart A. 2009. *SCADA: Supervisory Control and Data Acquisition.* s.l. : Isa; Edición: 0004, 2009. 1936007096.

Brandendaugh. 2000. *Aplicaciones JavaScript.* Madrid,España : O'Reilly & Associates, 2000. ISBN 84-415-1079-9.

Burnette, Ed. 2005. *Eclipse IDE Pocket Guide.* s.l. : O'Reilly Media, 2005.

Cáceres, José Antonio Aragón. 2009. *Servicio de Integración con Terceros para el Acceso de Variables del sistema SCADA Guardián del ALBA.* 2009.

Carbajal, Martha. 2014. *Evolución de la automatización industrial.* 2014.

Carbón, Yanet Cedeño. 2014. *Especificación de requisitos de software. Sistema de Supervisión y Control Guardián del ALBA.* 2014.

Ceca, Jesus Moreno. 2013. *Nuevas tecnologías de manipulación usando sensorización integrada.* 2013.

Doce, Yanet Nieto. 2010. *Sistema SCADA.* 2010.

Bibliografía

Escribano, Gerardo Fernández. 2002. *Introducción a Extreme Programming.* 2002.

Figueroa, Roberth G. 2010. *METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* 2010.

Galdámez, Kevin. 2014. *Creación de páginas web, formularios y base de datos en access.* 2014.

Hernández, Jimmy. 2009. patronesdediseno.blogspot.com. [En línea] 2009. http://patronesdediseno.blogspot.com/2009/05/patron-iterator.html.

Hernández, María Isabel. 2010. *Desarrollo de un sistema SCADA para la medicón de voltajes con sistemas embebidos para le laboratorio de mecatrónica de la facultad de mecánica.* 2010.

HoneyWellsp. 2000. HoneyWellsp. [En línea] 2000. [Citado el: 17 de enero de 2015.] http://www.honeywellsp.com/hw_productos_servicios/hw_industrial/Hw_Plantscape.htm#Control%20Anal%C3%B3gico.

Javier, Maikel Pérez. 2015. *Manual Organizacional.* 2015.

Jorge Savedra. 2007. [En línea] 2007. http://www.ecured.cu/index.php/Patrones_de_Asignaci%C3%B3n_de_Responsabilidades.

Joskowicz, Jose. 2008. *Reglas y prácticas en Extreme Programing.* Universidad de Vigo.España : s.n., 2008.

Lapuente, María Jesús Lamarca. 2013. *Hipertexto, el nuevo concepto de documento en la cultura de la imagen.* Madrid : Creative Commons, 2013.

Larman, Craig. 2003. *UML y patrones. Introducción al análisis y diseño orientado a objetos.* s.l. : Pearson, 2003.

Bibliografía

librosWeb.es. 2013. LIBROSWEB. [En línea] 12 de agosto de 2013. https://librosweb.es/libro/css/capitulo_1.html.

Linacre, Elsevier. 2003. *Practical SCADA for industry.* 2003.

Lorenzo, Ariel Chávez. 2010. *Estándares de codificación para C++.* 2010.

Madrid, Universidad informatica de. *Patrones del "Gang of Four".*

Martinez, Rafael. 2013. PostgresSQL.org.es. [En línea] 2013. http://www.postgresql.org.es/sobre_postgresql.

Mastermegazine. 2010. www.mastermegazine.info. [En línea] 2010. http://www.mastermagazine.info/termino/4582.php.

Moreno, Emilio Garcia. 2009. *Automatización de procesos industriales: robótica y automática.* Valencia : Ed. Univ. Politéc, 2009. 8477217599.

ni.com. 2014. ni.com. [En línea] 2014. http://www.ni.com/getting-started/labview-basics/esa/debug.

Penadés, Carmen y Letelier Torres, Patricio Orlanndo. 2006. *Métodologías ágiles para el desarrollo de software.* 2006. Vol. 5. ISSN-e 1666-1680.

Pérez, Javier Eguíluz. 2008. *Introducción a JavaScript.* 2008.

Pérez, María J. Pérez. 2012. *Guia comparativa de metodologías ágiles.* Segovia. Universidad de Valladolid : s.n., 2012.

Pérez, Mariñán. *Patrones de Diseño (Design Patterns)". .*

Radvanovsky, Robert. 2003. *Handbook of SCADA/Control System Security.* s.l. : CRC Press, 2003.

Rancel, Mario. 2008. *Depuración de programas frente a errores.* 2008.

Rodríguez, Argelio De la Rosa y Esquijarrosa Valdés, Mayliuvis. 2014. *Informe de análisis y diseño del módulo Configuración.* 2014.

Bibliografía

Santiago, Maria Lourdes. 2009. *Proceso de Desarrollo Unificado(RUP).* 2009.

Schuh, Peter. 2001. *Recovery, redemption, and extreme programming.* s.l. : IEEE Software, 2001. 10.1109/52.965800.

Sierra, Francisco Javier Ceballos. 2012. *Enciclopedia del lenguaje C++.* s.l. : AlfaOmega, 2012. 005.133 C387e RES.

Stutzke, R. D. 2005. *Stimating Software-Intensive Systems : projects ,products and processes.* Massachusetts : Addison Wesley, 2005. ISBN: 0-201-70312-2.

Villa-Alzado, luis. 2010. Alzado org. [En línea] 2010. [Citado el: 17 de enero de 2015.] http://www.alzado.org/articulo.php?id_art=850.

Villafuerte, Victor. 2009. *Extreme Programming.* 2009.

Anexo 1 Tarjetas clase-responsabilidad-colaboración (CRC)

Las tarjetas clase-responsabilidad-colaboración (CRC) permiten desprenderse del método basado en procedimientos y trabajar con una metodología basada en objetos, así el programador se centra y comienza a apreciar el desarrollo orientado a objetos (Beck, 1999). Las tarjetas CRC representan objetos y se describen partir de los siguientes elementos.

- **Clase:** Nombre de la clase a la cual pertenece la tarjeta.
- **Responsabilidad:** Describe cuales son las funcionalidad que deben ser implementadas por la clase.
- **Colaboración:** Enumera las diferentes clases con las cuales tiene relación la clase a la cual pertenece la tarjeta CRC. Se representa la dirección de la clase dentro del código del módulo de configuración.

A continuación se muestra la tarjeta CRC de la clase CodificatorHTML quien es la encargada de escribir la información en un fichero determinado y por ende esta posee una gran importancia dentro del mecanismo de depuración como solución propuesta.

Tabla 10: *tarjeta CRC clase CodificatorHTML paquete Utilidades.*

Tarjeta CRC	
Clase: CodificatorHTML	
Responsabilidad	Colaboración
writeFile (Escribe la información en dicho fichero). writeEnd (Escribe el final de un nodo). writeStyle (Escribe tanto el estilo de los ficheros como la función JavaScript que posibilita contraer-expandir información determinada).	• stdio.h • configuration/utils/ToStdString.h

De la tabla 11 a la 17 se muestran las tarjetas CRC de la clase principal para la generación de los ficheros de cada uno de los agentes de configuración existentes en módulo de configuración.

Tabla 11: *tarjeta CRC clase SecurityModuleInfo paquete Agentes*

Tarjeta CRC	
Clase: SecurityModuleInfo	
Responsabilidad	Colaboración
SecurityModuleInfo (Constructor de la clase). privatePrintToFile (Método encargado de imprimir en fichero HTML la información de la clase)	• configuration/Agents/AbstractAgent/AbstractModuleInfo.h • configuration/common/DataTypeDefinition.h • NamespaceDefinition.h

Tabla 12: *tarjeta CRC clase HMIRuntimeModuleInfo paquete Agentes*

Tarjeta CRC	
Clase: HMIRuntimeModuleInfo	
Responsabilidad	Colaboración
HMIRuntimeModuleInfo (Constructor de la clase). privatePrintToFile (Método encargado de imprimir en fichero HTML la información de la clase)	• configuration/Agents/AbstractAgent/AbstractModuleInfo.h • configuration/HMIConfigModel/RuntimeProject.h • configuration/common/OperationalConfigActions.h • string.h • configuration/utils/CodificatorHTML.h

	• configuration/utils/ToStdString.h • "NamespaceDefinition.h"

Tabla 13: *tarjeta CRC clase HMIEditionModuleInfo paquete Agentes*

Tarjeta CRC	
Clase: HMIEditionModuleInfo	
Responsabilidad	Colaboración
HMIEditionModuleInfo (Constructor de la clase). privatePrintToFile (Método encargado de imprimir en fichero HTML la información de la clase)	• configuration/Agents/AbstractAgent/AbstractModuleInfo.h • configuration/HMIConfigModel/NamespaceDefinition.h • configuration/utils/CodificatorHTML.h • "NamespaceDefinition.h"

Tabla 14: *tarjeta CRC clase HDBModuleInfo paquete Agentes*

Tarjeta CRC	
Clase: HDBModuleInfo	
Responsabilidad	Colaboración
HDBModuleInfo (Constructor de la clase). privatePrintToFile (Método encargado de imprimir en fichero HTML la información de la clase)	• stdio.h • configuration/Agents/AbstractAgent/AbstractModuleInfo.h • "NamespaceDefinition.h" • configuration/utils/CodificatorHTML.h • configuration/utils/ToStdString.h • string.h

Tabla 15: tarjeta CRC clase Comm3ModuleInfo paquete Agentes

Tarjeta CRC	
Clase: Comm3ModuleInfo	
Responsabilidad	Colaboración
Comm3ModuleInfo (Constructor de la clase). privatePrintToFile (Método encargado de imprimir en fichero HTML la información de la clase)	• configuration/Agents/AbstractAgent/AbstractModuleInfo.h • configuration/common/OperationalConfigActions.h • "NamespaceDefinition.h" • "Comm3AgentExports.h" • configuration/utils/CodificatorHTML.h • configuration/utils/ToStdString.h • string.h

Tabla 16: tarjeta CRC clase AppModuleInfo paquete Agentes

Tarjeta CRC	
Clase: AppModuleInfo	
Responsabilidad	Colaboración
AppModuleInfo (Constructor de la clase). privatePrintToFile (Método encargado de imprimir en fichero HTML la información de la clase)	• configuration/Agents/AbstractAgent/AbstractModuleInfo.h • <configuration/common/DataTypeDefinition.h> • <configuration/common/OperationalConfigActions.h> • vector.h

	• "NamespaceDefinition.h"

Tabla 17: *tarjeta CRC clase AcquisitionModuleInfo paquete Agentes*

Tarjeta CRC	
Clase: AcquisitionModuleInfo	
Responsabilidad	Colaboración
AcquisitionModuleInfo (Constructor de la clase). privatePrintToFile (Método encargado de imprimir en fichero HTML la información de la clase)	• Map.h • configuration/Agents/AbstractAgent/AbstractModuleInfo.h • configuration/common/OperationalConfigActions.h • "NamespaceDefinition.h"

Anexo 2 Tareas de ingeniería

Para la confección de cada una de las tareas se utilizó tablas cuyo modelo cuenta con los siguientes campos:

- **No. de tarea:** Numeración continua que identifica a la tarea.
- **No. de HU:** Número de la HU a la cual pertenece.
- **Nombre de la tarea:** Identificación literal de la tarea.
- **Tipo de tarea:** Tipo de tarea, dígase diseño, desarrollo, prueba.
- **Puntos estimados:** Representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizara para su realización.
- **Fecha inicio:** Fecha estimada de inicio de realización.
- **Fecha fin:** Fecha estimada de fin de realización.
- **Descripción:** Se describe en que consiste la tarea y que elementos deben cumplirse para declarar la tarea terminada.

En las tablas de la 19 a la 28 se describen cada una de las tareas de ingeniería realizada.

Tabla 19: *Tarea de ingeniería: Desarrollo de una función para crear un fichero HTML.*

Tarea de ingeniería	
No. de tarea: 1	**No. de HU: 1**
Nombre de la tarea: Desarrollo de una función para crear un fichero HTML.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 16/02/2015	**Fecha fin:**17/02/2015
Descripción: Se crea una función que crea un fichero HTML en una ruta especificada por el programador. Para el desarrollo de esta función se utilizó la biblioteca stdio del lenguaje C++. El fichero se crea con los permisos para la escritura. La información se sobrescribe en caso de no estar vacío dicho fichero.	

Tabla 20: *Tarea de ingeniería: Desarrollo de la clase principal.*

Tarea de ingeniería	
No. de tarea: 1	**No. de HU: 2**
Nombre de la tarea: Desarrollo de la clase principal.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 18/02/2015	**Fecha fin:** 21/02/2015
Descripción: Se desarrolla una clase que enmarcará la información enviada por un agente de configuración a un módulo dentro de las etiquetas UL y LI del lenguaje HTML.	

Tabla 21*: Tarea de ingeniería: Desarrollo de una función para escribir en fichero HTML.*

Tarea de ingeniería	
No. de tarea: 2	**No. de HU: 2**
Nombre de la tarea: Desarrollo de una función para escribir en fichero HTML.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 23/02/2015	**Fecha fin:** 13/03/2015
Descripción: Los agentes de configuración son los encargados de la distribución de la configuración a cada uno de los módulos del SCADA, es por ello se les agregó una funcionalidad capaz de escribir en un fichero HTML dado la información que estos manejan, para ello harán uso de una clase previamente desarrollada.	

Tabla 22: Tarea de ingeniería: Desarrollo de un estilo visual sencillo.

Tarea de ingeniería	
No. de tarea: 1	**No. de HU: 3**
Nombre de la tarea: Desarrollo de un estilo visual sencillo.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 17/03/2015	**Fecha fin:** 19/03/2015
Descripción: Se crea un estilo visual para los ficheros HTML generados, los cuales se visualizaran en forma de árbol resaltando en otro color tanto la raíz como los nodos. La raíz estará compuesta por el nombre del módulo al cual pertenece la información, los nodos serán los parámetros de configuración las hojas los atributos con sus respectivos valores de dichos parámetros.	

Tabla 23: Tarea de ingeniería: Desarrollo de una función JavaScript.

Tarea de ingeniería	
No. de tarea: 1	**No. de HU: 4**
Nombre de la tarea: Desarrollo de una función JavaScript.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 19/03/2015	**Fecha fin:** 20/03/2015
Descripción: Se desarrolla una función en el lenguaje JavaScript la cual permite que tanto la raíz como los nodos del árbol que representa la información enviada por un agente de configuración puedan contraerse o expandirse. Inicial mente el fichero se encuentra contraído.	

Tabla 24: Tarea de ingeniería: Modificar el script de configuración.

Tarea de ingeniería	
No. de tarea: 1	**No. de HU: 5**
Nombre de la tarea: Modificar el script de configuración.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 23/03/2015	**Fecha fin:** 25/03/2015
Descripción: Se modifica el script de inicio del módulo dando la posibilidad de que se lea el fichero de configuración definido para el módulo mediante el comando reload.	

Tabla 25: Tarea de ingeniería: Desarrollo de una función para leer parámetros.

Tarea de ingeniería	
No. de tarea: 2	**No. de HU: 5**
Nombre de la tarea: Desarrollo de una función para leer parámetros.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 25/03/2015	**Fecha fin:** 27/03/2015
Descripción: Al ejecutarse el comando reload en las opciones de inicio del módulo, todos los parámetros contenidos en el fichero de configuración del módulo son leídos nuevamente.	

Tabla 26: Tarea de ingeniería: Desarrollo de una función para actualizar parámetros.

Tarea de ingeniería

No. de tarea: 3	**No. de HU: 6**
Nombre de la tarea: Desarrollo de una función para actualizar parámetros.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 30/03/2015	**Fecha fin:** 1/04/2015
Descripción: Tras la lectura de los parámetros, el módulo de configuración se procede a la actualización de aquellos que hayan sufrido algún cambio.	

Tabla 27: *Tarea de ingeniería: Desarrollo de una función de aviso.*

Tarea de ingeniería	
No. de tarea: 1	**No. de HU: 7**
Nombre de la tarea: Desarrollo de una función de aviso.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 1/04/2015	**Fecha fin:** 3/04/2015
Descripción: Se desarrolla de una función encargada de enviar un aviso a los agentes conectados al servidor de configuración una vez que se ha actualizado el parámetro de configuración referente al funcionamiento del módulo con el objetivo de que estos adopten el modo depuración o lo desactiven en caso de ya estar activado.	

Tabla 28: *Tarea de ingeniería: Desarrollo de una función que active-desactive modo depuración.*

Tarea de ingeniería

No. de tarea: 2	**No. de HU: 7**
Nombre de la tarea: Desarrollo de una función que active-desactive modo depuración.	
Tipo de tarea: Desarrollo	**Responsable:** Raudel Arencibia Ramírez
Fecha inicio: 6/04/2015	**Fecha fin:** 25/04/2015
Descripción: Se desarrolla de una función para el agente abstracto de configuración encargada de activar o desactivar el modo depuración. Este agente al ser abstracto indica que los restantes heredan de él y por ende también contendrían esta función.	

Anexo 3 Pruebas de aceptación de la primera iteración

En las tablas de la 30 y 31 muestran las pruebas de aceptación satisfactorias realizadas la HU2.

Tabla 30 HU2_P1: *Fichero escrito.*

Código: HU2_P1	HU: 2
Nombre: Fichero escrito.	
Descripción: Se realiza cambios de configuración del SCADA Galba referentes a los recursos que son manejados por cada uno de los módulos.	
Condiciones de ejecución: Módulo de configuración en modo depuración.	
Entrada: Módulos del SCADA Galba	
Pasos de ejecución: 1. Abrir editor del SCADA Galba. 2. Acceder a la opción **abrir proyecto desde el servidor** de la barra de herramientas. 3. Acceder a la opción **explorador de proyecto** de la barra de herramientas. 4. Seleccionar la configuración referente a un módulo. 5. Desplegar los recursos asociados al módulo seleccionado. 6. Se agrega un nuevo recurso al sistema. 7. Acceder a la opción **Guardar proyecto en el servidor** de la barra de herramientas.	
Resultado esperado: El sistema escribe en el fichero HTML el nombre del módulo al cual se le agregó el recuro como raíz del mismo, los parámetros de	

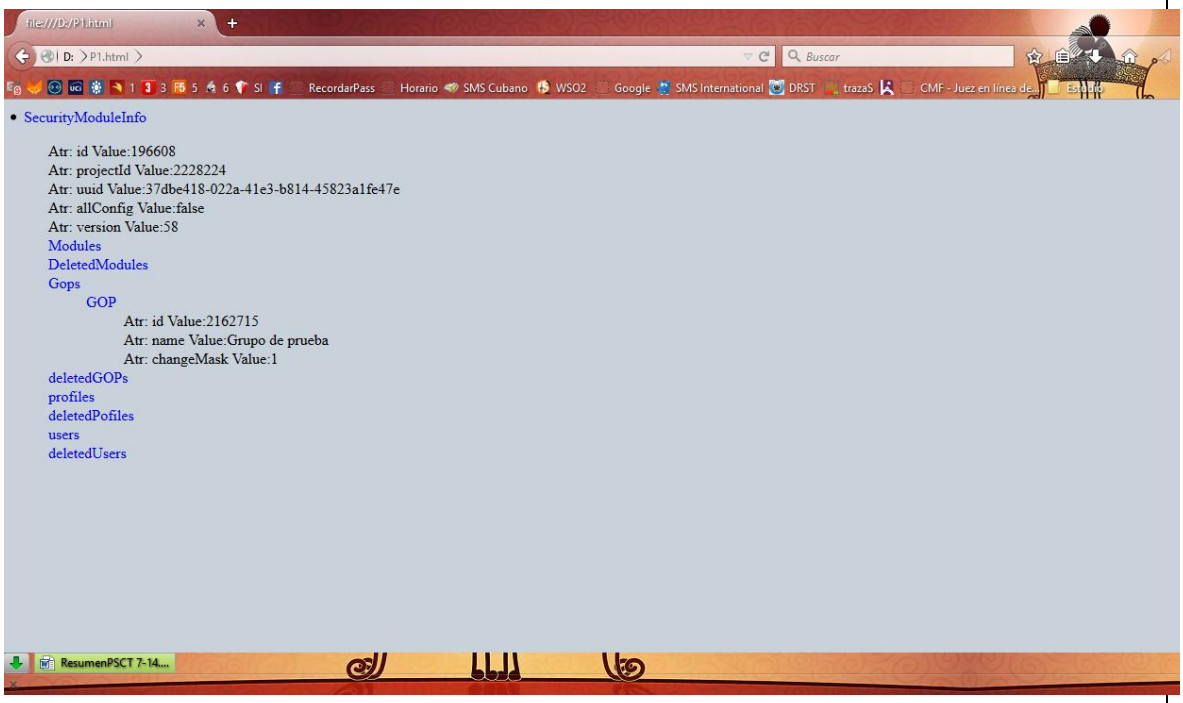
configuración del módulo, en el nodo referente al recurso añadido escribe el nuevo recurso con sus respectivos parámetros de configuración.
Resultado obtenido: 
Evaluación de la prueba: Satisfactoria

Tabla 31 HU2_P2: *Datos incorrectos*

Código: HU2_P2	HU: 2
Nombre: Datos incorrectos	
Descripción: Se realiza cambios de configuración del SCADA Galba referentes a los recursos que son manejados por cada uno de los módulos con datos incorrectos.	
Condiciones de ejecución: Módulo de configuración en modo depuración.	
Entrada/Pasos de ejecución: Módulos del SCADA Galba	

Pasos: 1. Abrir editor del SCADA Galba. 2. Acceder a la opción **abrir proyecto desde el servidor** de la barra de herramientas. 3. Acceder a la opción **explorador de proyecto** de la barra de herramientas. 4. Seleccionar la configuración referente a un módulo. 5. Desplegar los recursos asociados al módulo seleccionado. 6. Se agrega un nuevo recurso al sistema con datos incorrectos. **7.** Acceder a la opción **Guardar proyecto en el servidor** de la barra de herramientas.
Resultado esperado: Se muestra una notificación emitida por el HMI editor señalando los campos incorrectos en color rojo.
Resultado obtenido: El HMI editor muestra la notificación señalando los datos incorrectos de color rojo.
Evaluación de la prueba: Satisfactoria

En la tabla 32 se muestra las pruebas de aceptación realizadas a la HU3.

Tabla 32 HU3_P1: *Visualización de la información en forma de árbol.*

Código: HU3_P1	HU: 3
Nombre: Visualización de la información en forma de árbol.	
Descripción: Se procede a la ejecución de un fichero generado por el mecanismo de depuración referente a un módulo del sistema.	
Condiciones de ejecución: Fichero previamente generado.	

Entradas:

Fichero HTML generado con la información enviada a un módulo específico.

Pasos de ejecución:

1. Abrir la ruta física definida para la generación de las trazas.
2. Realizar doble clic en un fichero generado.

Resultado esperado: El fichero se visualiza en el navegador web en forma de árbol donde se puede contraer o expandir información específica.

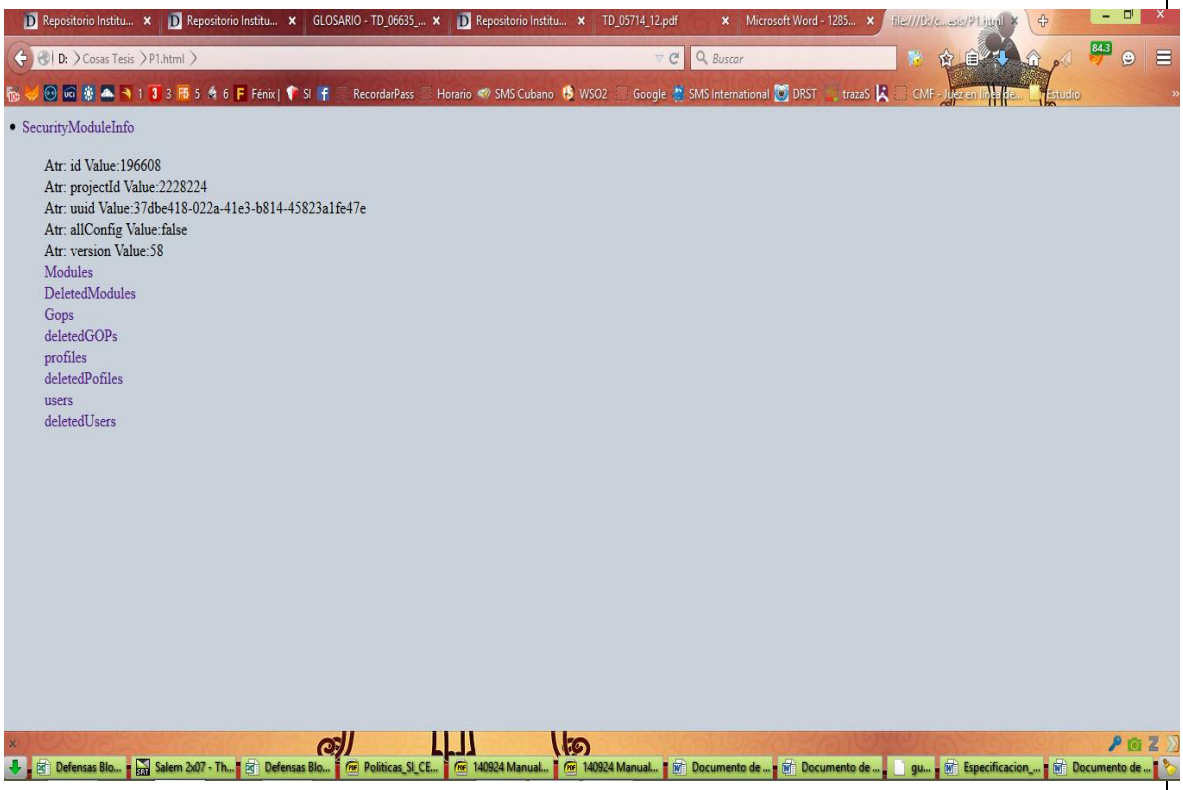
Resultado obtenido:

Evaluación de la prueba: Satisfactoria

Tabla 33 HU4_P1: Expandir información.

Código: HU4_P1	HU: 4
Nombre: Expandir información.	
Descripción: Se procede a visualizar un fichero generado por el mecanismo de depuración con el objetivo de expandir información.	
Condiciones de ejecución: Módulo de configuración en modo depuración.	
Entrada: Fichero HTML generado con la información enviada a un módulo específico.	
Pasos de ejecución: 1. Abrir la ruta física definida para la generación de las trazas. 2. Realizar doble clic en un fichero generado.	
Resultado esperado: El fichero se visualiza contraído, al dar clic en la raíz este se expande mostrando los parámetros de configuración del módulo correspondiente y los nodos asociados a dicho módulo.	

Resultado obtenido:



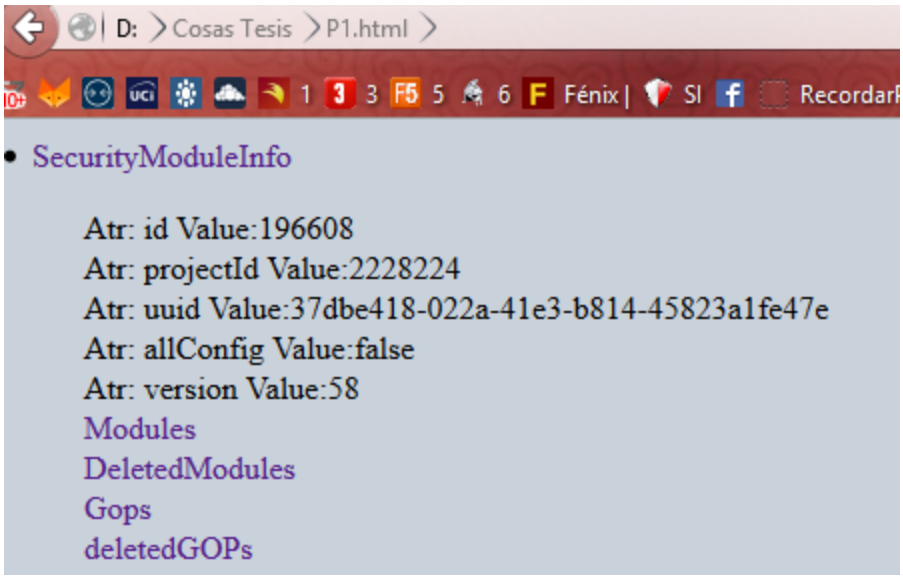
Evaluación de la prueba: Satisfactoria

Tabla 34 HU4_P2: Contraer información.

Código: HU4_P2	HU: 4
Nombre: Contraer información.	
Descripción: Se procede a visualizar un fichero generado por el mecanismo de depuración con el objetivo de contraer información.	
Condiciones de ejecución: Fichero HTML previamente generado.	
Entrada: Fichero HTML generado con la información enviada a un módulo específico.	
Pasos: 1. Abrir la ruta física definida para la generación de las trazas. 2. Realizar doble clic en un fichero generado.	

Resultado esperado: Al visualizarse el fichero en el navegador este se encuentra totalmente contraído, al dar clic en la raíz se despliegan todos los nodos pertenecientes al fichero, al realizar clic nuevamente en la raíz de dicho fichero este contrae toda la información.

Resultado obtenido:



Evaluación de la prueba: Satisfactoria