Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 10, Mes Octubre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 48-58
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: Soluciones informáticas
Recibido: 20/06/2018 | Aceptado: 01/10/2018 | Publicado: 20/10/2018

Entorno Integral de Desarrollo para Prácticas de Control Automática en un Sistema de Laboratorios Remotos

Integral Development Environment for Automatic Control Practices in a Remote Laboratory System

Taimí Torres Cuello[1*], Lázaro Oropesa Mejías[2]

[1] Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas. ttorres@uci.cu
[2] Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas. loropesa@uci.cu

[*] Autor para correspondencia: yalena@uart.edu.cu

Resumen

El presente trabajo está enmarcado en la implementación de un Entorno Integrado de Desarrollo el cual va a estar integrado a un Sistema de Laboratorios Virtuales y a Distancia, con el objetivo de crear una interfaz web con conexión al software MATLAB para que el usuario pueda elaborar prácticas de Control Automático. Para desarrollar el IDE fue elaborado el marco teórico, lo que aportó un gran conocimiento acerca de los Sistemas de Laboratorios Virtuales y a Distancia en el Control Automático y de los Entornos Integrales de Desarrollo. La metodología utilizada durante la investigación fueOpenUp y como lenguaje de programación se utilizó Java Web. Se diseñaron e implementaron las clases del Entorno Integral de Desarrollo, aplicando las buenas prácticas del patrón arquitectónico Modelo Vista Controlador y de los patrones de diseño GRASP. Fueron realizadas varias pruebas, con el objetivo de comprobar el funcionamiento y la calidad la aplicación.

Palabras clave: IDE, Sistema de Laboratorio Remoto, Aprendizaje electrónico, sistema informático

Abstract

The present work is framed in the implementation of an Integrated Development Environment which will be integrated into a Virtual and Distance Laboratory System, with the aim of creating a web interface with connection to the MATLAB software so that the user can elaborate practices of Automatic control. In order to develop the IDE, the theoretical framework was elaborated, which contributed a great knowledge about the Virtual and Distance Laboratory Systems in the Automatic Control and the Integral Development Environments. The methodology used during the investigation was OpenUp and Java Web was used as a programming language. The Integral Development Environment classes were designed and implemented, applying the best practices of the Architectural

Model Vista Controller and the GRASP design patterns. Several tests were carried out, in order to verify the operation and quality of the application.

Keywords: IDE, Remote Laboratory System, Electronic Learning, computer system

Introducción

Desde los inicios de la humanidad, la educación ha sido una de las áreas fundamentales en el desarrollo de la misma. Cada día es mayor el esfuerzo del hombre para mejorarla, con el objetivo de formar más y mejores profesionales en cada una de las esferas de la sociedad. En la actualidad al sistema educacional se le han presentado disímiles dificultades, entre las cuales se encuentra la necesidad de dar respuesta a los diferentes cambios sociales, económicos y culturales, que surgen en la sociedad.

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) se han presentado nuevas formas para perfeccionar y desarrollar la educación, como es el caso de la educación a distancia (ED) (Sartorius C, et al., 2005)**.** La ED se presenta como la solución idónea para un conjunto de colectivos que exigen el disponer de sistemas enseñanza más flexible, accesibles y adaptativos. Dentro de la educación a distancia, el control automático es una de las áreas técnicas que más ha explotado las nuevas tecnologías para desarrollar herramientas que faciliten el aprendizaje. Dicha área aporta sus Sistemas de Laboratorios Remoto (SLR) con el objetivo de compartir determinados recursos tecnológicos mediante el Internet, ya que de otra forma resultaría difícil de generalizar debido a su elevado costo de implementación.

En Cuba se hace uso también de los laboratorios virtuales. Por la importancia que tienen los mismos, la Universidad de las Ciencias Informáticas (UCI) es una de las universidades que lo utilizan, garantizando que los estudiantes tengan una mejor preparación y formación, para el apoyo en el Proceso de Enseñanza-Aprendizaje (PEA), en el trabajo docente para profesores y estudiantes. Algunos ejemplos de ellos son los utilizados en Matemática, Física y Sistema Operativo. Además se está desarrollando un SLR que va a estar integrado por varios módulos entre los que se encuentra un Entorno Integral de Desarrollo (por sus siglas en inglés IDE) al cual el estudiante pueda acceder desde una plataforma.

Los IDEs están diseñados para maximizar la productividad del programador previendo de componentes muy unidos con interfaces similares del usuario, presentan un único programa en el cual se realiza toda la programación. Generalmente, este programa ofrece muchas características para la creación, modificación, compilación,

implementación y depuración de software. Además permite ir al archivo donde están declaradas funciones o variables, validación de código y conocer los ficheros en los que se tiene error de sintaxis.

Permite realizar experimentos de control por medio de aplicaciones gráficas, donde esta forma de programación, le brinda al usuario la posibilidad de realizar las tareas en un tiempo menor al que le toma actualmente. Cuenta además con un sistema implícito de detección de errores y la posibilidad al usuario de simular aplicaciones de Control Automático. (Carabana, 2010), (MAR, O. and GULÍN 2018) . Permite realizar controladores mediante la herramienta matemática Matlab donde el programa realiza las lecturas de los valores del sistema y variables a controlar mediante ficheros, con la capacidad de modificar los mismos mediante el envío de nuevos valores (Reyes, et al., 2010), (MAR, O. *et al.* 2017).

Sin embargo surge la investigación debido a necesidad que las aplicaciones existentes en la actualidad para un SLR sea capaz de proporcionar al usuario una interfaz a la cual se pueda acceder vía web para realizar las prácticas de Control Automático con el objetivos de realizar cambios en los controladores y el sistema en general, debido a que solo es posible el acceso al sistema con una conexión directa al servidor vía telnet.

Materiales y métodos o Metodología computacional

En la presente sesión, se realiza el análisis y diseño de la solución. En él quedan definidas todas las funcionalidades y características que debe cumplir el IDE a través de los requisitos funcionales y requisitos no funcionales. Se identifican los casos de usos del sistema, realizando la descripción de cada uno de ellos, agrupando dentro de estos, los diferentes requisitos funcionales. Se modela el diagrama de clases del diseño, abordando acerca de los patrones utilizados, patrones de diseño y patrones arquitectónicos.

Especificación de requisitos.

La especificación de requisitos es un proceso fundamental para el desarrollo de cualquier software. Este tiene como principales objetivos permitir a los clientes describir con claridad lo que desea obtener, una vez terminado el software, por lo que el cliente debe tener activa participación durante este proceso. Por otra parte, permite a los integrantes del equipo de desarrollo construir un software que satisfaga las necesidades del cliente.

Una buena especificación de requisitos de software ofrece una serie de ventajas entre las que destacan el contrato entre cliente y desarrolladores, reduce el esfuerzo en el desarrollo, brinda una base para la estimación de costes y planificación, además de ser un punto de referencia para procesos de verificación y validación, y sirve como base para la identificación de posibles mejoras en los procesos analizados (Monferrer Agut, 2001), (MAR, O *et al.* 2016).

Requisitos funcionales.

Los requisitos funcionales son los encargados de describir las determinadas funciones o condiciones que debe brindar el sistema. Describen además las posibles entradas y salidas, así como la respuesta que tiene el sistema ante algunas situaciones que se presenten en su interacción con el cliente.

Para la realización de la investigación se identificaron los siguientes requisitos funcionales:

RF1: Completar código en el editor de texto.

RF2: Refactorizar código fuente en el editor de texto.

RF3: Ejecutar código fuente.

RF4: Mostrar gráfica de coordenadas de una función.

RF5: Crear nuevo proyecto.

RF6: Crear nuevo proyecto mediante plantillas.

RF7: Crear plantillas.

RF8: Abrir plantillas.

RF9: Cargar plantillas.

RF10: Exportar proyecto desde la extensión m de Matlab.

RF11: Generar códigofuente de la función seleccionada.

Requisitos no funcionales.

Los requisitos no funcionales son las propiedades o cualidades que el producto debe presentar. Estos requisitos definirán las características del producto final y muchas veces están implicados en el éxito final que se desea alcanzar.

Requisitos de Software

Cliente: debe tener una computadora con sistema operativo(GNU/Linux y Windows7 o superior)que cuente con navegador web, donde se recomienda que sea Mozilla Firefox 32.0.

Servidor: Debe tener un servidor con Sistema Operativo GNU/Linux y Windows7 o superior y se requiere que cuente con un Servidor Web apache-tomcat-7 o superior.

Requisitos de Hardware

Cliente: Debe poseer una tarjeta de red a 100 Megabytes (Mb) o superior, donde se requiera que sea un Procesador Intel Pentium IV o superior, de 1Gb de RAM.

Servidor: Debe poseer un Procesador Corei3 o superior que cuente con una memoria RAM 1.0 Gigabytes (Gb) o superior y que tenga de capacidad de disco duro 80 Gb libres o superior.

Requisitos de usabilidad: Resaltado de sintaxis, el editor de código resalta mediante colores las palabras reservadas del lenguaje.

Requisitos de Seguridad: Disponibilidad: el sistema deberá estar disponible las 24 horas del día para todos los usuarios del mismo.

Modelo de casos de uso del sistema.

El modelo de casos de uso del sistema representa las relaciones existentes entre actores y casos de uso. Los actores son terceros fuera del sistema que interactúan con él, puede ser cualquier persona, individuo, grupo, entidad, organización, máquina o sistema de información externo; con los que el sistema interactúa y los casos de uso son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. (López Duque, y otros, 2010)

Durante la investigación fue identificado un único actor del sistema, que se describe a continuación:

Tabla 1: Descripción de los actores del sistema.

Actor	Descripción
Usuario	Cualquier persona capacitada para interactuar con el software.

Casos de usos del sistema.

CU1: Completar código en el editor de texto.RF1

CU2: Refactorizar código fuente en el editor de texto.RF2

CU3:Ejecutar código fuente.RF3

CU4: Mostrar gráfica de coordenadas de una función. RF4

CU5: Crear proyecto. RF5, RF6

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 10, Mes Octubre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 48-58
http://publicaciones.uci.cu

CU6: Crear plantillas. RF7

CU7: Abrir plantillas. RF8, RF9

CU8: Exportar proyecto de la extensión m de Matlab. RF10

CU9: Generar código de la función seleccionada.RF11

Diagrama de caso de uso del sistema.

El diagrama de casos de uso representado, describe las acciones que el usuario realiza mediante la utilización del IDE, así como las funciones que este brinda.

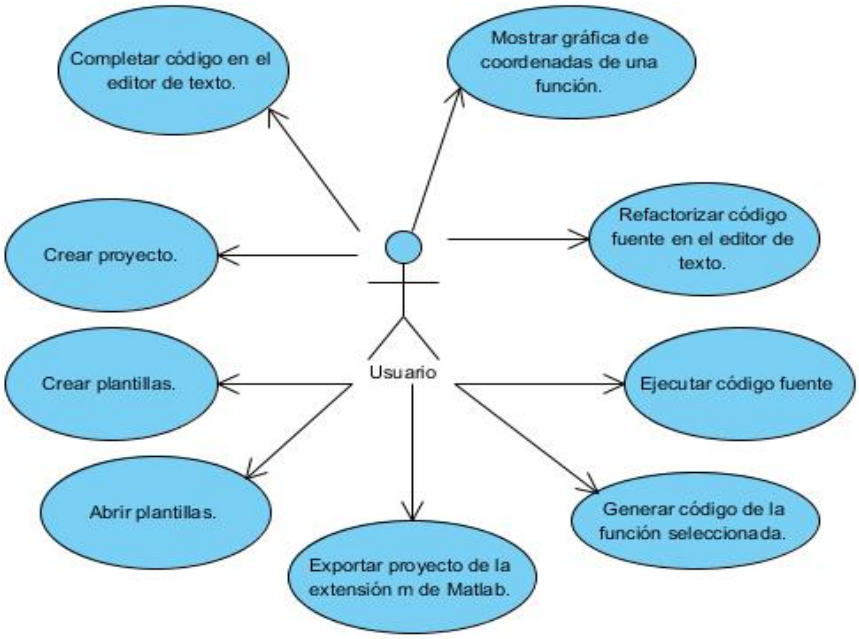

Fig.1 Diagrama de caso de uso del sistema.

Resultados y discusión

Modelo del diseño del sistema.

El modelo de diseño detalla los modelos de análisis, tomando en cuenta todas las implicaciones y restricciones técnicas. El propósito es especificar una solución que trabaje y pueda ser fácilmente convertida en código fuente y construir una arquitectura simple y extensible. Las clases definidas en el análisis se detallan y se añaden nuevas clases para manejar áreas técnicas como bases de datos, interfaces de usuario, dispositivos, entre otros.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 10, Mes Octubre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 48-58
http://publicaciones.uci.cu

Diagrama de clases del diseño del sistema.

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro.

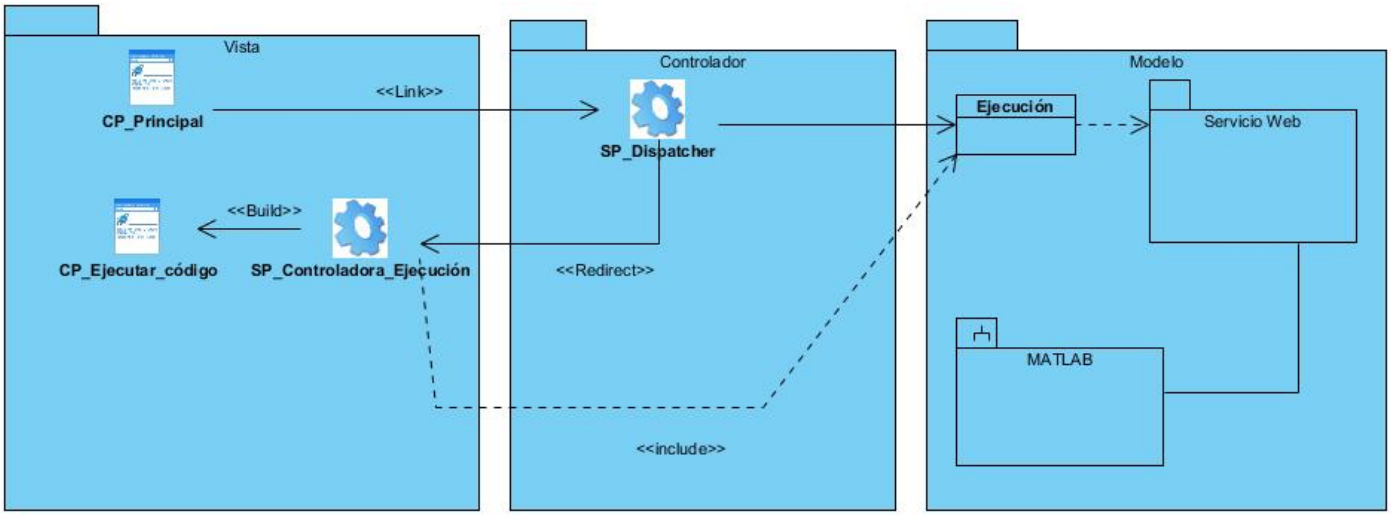

Fig.2: Diagrama de clases CU"Ejecutar código fuente".

Patrones utilizados.

Patrón arquitectónico.

Un patrón arquitectónico es de vital importancia ya que la estructura de un sistema influye directamente sobre la capacidad que presenta un sistema para satisfacer los atributos de calidad. Algunos de estos ejemplos de estos atributos de calidad son el tiempo de respuesta del sistema a las peticiones que se le hacen, la usabilidad, que tiene que ver con qué tan sencillo les resulta a los usuarios realizar determinadas operaciones con el sistema, como determinar qué tan sencillo es realizar cambios al mismo. (Cervantes, 2010)

Patrón arquitectónico Modelo Vista Controlador (MVC).

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes diferentes. El patrón MVC se ve frecuentemente en aplicaciones web. (Trott, 2000)

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 11, No. 10, Mes Octubre, 2018
ISSN: 2306-2495 | RNPS: 2343
Pág. 48-58
http://publicaciones.uci.cu

Fig.3: Patrón Arquitectónico Modelo-Vista-Controlador. La utilización de este patrón evita en gran medida el acoplamiento de los componentes del sistema.

Modelo: Maneja losdatos del sistema y las operaciones asociadas a esos datos.

Vista: Define y gestiona cómo se presentan los datos al usuario.

Controlador: Dirige la interacción del usuario.

Ventajas del patrón MVC (Sommerville, 2011)

1. Permite que los datos cambien de manera independiente de su representación y viceversa.
2. Soporta en diferentes formas la presentación de los mismos datos, y los cambios en una representación se muestran en todos ellos.
3. Clara separación entre interfaz, lógica de negocio y de presentación.
4. Sencillez para crear distintas representaciones de los mismos datos.
5. Reutilización de los componentes.
6. Simplicidad en el mantenimiento de los sistemas.
7. Facilidad para desarrollar prototipos rápidos.
8. Los desarrollos suelen ser más escalables.

Desventajas del patrón MVC (Sommerville, 2011)

1. Puede implicar código adicional y complejidad de código cuando el modelo de datos y las interacciones son simples.
2. La curva de aprendizaje para los nuevos desarrolladores se estima mayor que la de modelos más simples.

3. La distribución de componentes obliga a crear y mantener un mayor número de ficheros.

Patrones de diseño.

Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos con consejos acerca de cómo aplicarlo en nuevas situaciones y discusiones sobre sus compromisos (Mendoza, 2013). Un patrón de diseño es una solución repetible a un problema recurrente en el diseño de software. Esta solución no es un diseño terminado que puede traducirse directamente a código, sino más bien una descripción sobre cómo resolver el problema. (Visconti, 2011)

En resumen los patrones de diseños se pueden definir de la siguiente manera:

1. Describen un problema recurrente y una solución.
2. Cada patrón nombra, explica, evalúa un diseño recurrente en sistemas orientados a objetos.

En la construcción del diseño de la aplicación informática a implementar, se utilizaron los patronesGRASP (acrónimo del inglés *General ResponsibilityAssignment Software Patterns*). Se considera quemás que patrones son una serie de "Buenas Prácticas" de aplicación recomendable en el diseño desoftware.

Para la realización fueron necesarios utilizar varios de estos patrones de diseño, a continuación se mencionan los que fueron utilizados:

Creador: El Patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que conecte con el objeto producido en cualquier evento.

La nueva instancia del objeto deberá ser creada por la clase que: tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase. Este patrón brinda soporte de bajo acoplamiento, lo cual supone menos dependencias entre clases y posibilidades.

Experto: Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Este patrón se puede evidenciar en más de una clase del sistema como por ejemplo en la figura 4, la clase Controladora Ejecución tiene toda la información necesaria para ejecutar el código.

Controlador: Este patrón se ve evidenciado en la figura 4. En la clase SP_Dispatcher ya que esta es la clase encargada de asignar la responsabilidad de manejar los eventos de un sistema a una clase que represente un sistema global.

Bajo acoplamiento: Este patrón asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. La clase controladora no realiza las actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.

En la figura 4 se ve evidenciado el patrón ya que las clases que se encuentran en la capa del modelo no tienen mucha dependencia con las demás clases presentes en la Vista o el Controlador, garantizando bajo acoplamiento.

Alta cohesión: El patrón alta cohesión describe cuan fuertemente los contenidos internos de una rutina están relacionados entre sí. Este patrón propone asignar la responsabilidad de manera que la complejidad se mantenga dentro de límites manejables asumiendo solamente las responsabilidades que deben manejar, evadiendo un trabajo excesivo.

El patrón se evidencia en la clase del modelo ya que solo tienen la información necesaria para realizar su labor garantizando que ninguna de ellas realice una labor muy compleja.

Conclusiones

Con la descripción del modelo de dominio y su elaboración se logra una comprensión de la solución, para así potenciar el desarrollo del IDE. Se definieron los requisitos funcionales y los no funcionales, que permitió conocer las funcionalidades y características que debe brindar la aplicación. Se identificaron los casos de usos del sistema, realizando la descripción de cada uno de ellos y se estableció qué requisitos funcionales se cubren encada caso de uso, permitiendo solucionar problemas existentes. Se realizó el diagrama de clases del diseño, en el cual se identificaron los patrones utilizados, los de diseño y el patrón arquitectónico, el uso de estos patrones posibilitó una correcta asignación de responsabilidades a cada una de las clases.

Referencias

Sartorius C, Aldo R, Hernández S, Luis y Aracil Santonja, Rafael. 2005. Laboratorio a distancia para la prueba y evaluación de controladores a través de Internet. Universidad Central "Marta Abreu" de las Villas : s.n., 2005, Vol. 16.

Reyes, Cesar, y otros. 2010.Control de la planta de los cuatro tanques mediante la realización de una pasarela Matlab. Universidad de Sevilla : s.n., 2010.

Monferrer Agut, Raúl. 2001.Especificación de Requisitos Software según el estándar de IEEE 830. s.l. : Universitat Jaume I, 2001.

MAR, O. and J. GULÍN Modelo para la evaluación de habilidades profesionales en un sistema de laboratorios a distancia Revista científica, 2018, 3(33): 332-343.

MAR, O.; J. GULÍN, et al. Sistema de Laboratorios a Distancia para la práctica de Control Automático Revista Cubana de Ciencias Informáticas, 2016, 10(4): 171-183.

MAR, O.; I. SANTANA, et al. Competency assessment model for a virtual laboratory system and distance using fuzzy cognitive map Revista Investigación Operacional, 2017, 38(2): 170.178.

López Duque, Marleysi y Oceguera Ravelo, Raide. 2010.Herramienta en Matlab para la obtención de datos y ficheros electroencefalograma del proyecto Mapeo Cerebral Humano Cubano,. La Habana : s.n., 2010.

Cervantes, Humberto. 2010. Arquitectura de Software. s.l. : SG, 2010.

Trott, J. y Shalloway, A. 2000.Design Patterns Explained. MODELER. 2000.

Sommerville, Ian. 2011.Ingeniería de Software,Novena edición. México : s.n., 2011. 978-607-32-0603-7.

Mendoza, J. 2013. Diseño del sistema de tarjeta de crédito con UML. [En línea] 2013. http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/mendoza_nj/Cap5.pdf..

Visconti, M y Astudillo, H. 2011. Fundamentos de Ingeniería de Software. [En línea] 2011. http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf..