Proceso de mejora del Sistema de Gestión de Proyectos para Cuba y Venezuela

Improvement process of Projects Management System for Cuba and Venezuela

Leodanys Wilber Guerrero Grey[1], Yamira Medel Viltres[1]

[1] Profesor en la Facultad de Ciencias Informáticas de la Universidad de Granma, Cuba

lguerrerogrey@udg.co.cu , ymedelv@udg.co.cu

RESUMEN. El Sistema de Gestión de Proyectos del convenio de cooperación entre Cuba y Venezuela surgió para automatizar el proceso de presentación de proyectos, y realizar un seguimiento de ellos. Debido a problemas encontrados en la versión anterior, se decidió la realización de la siguiente investigación con el objetivo de analizar una serie de medidas teóricas y prácticas para resolverlos. Para esta investigación se llevó a cabo un nuevo análisis de requerimientos, se utilizaron métricas de diseño e implementación, análisis de complejidad de algoritmos y pruebas de carga y estrés, logrando así la satisfacción del cliente y mayor rendimiento de aplicación.

ABSTRACT. The Projects Management System of the cooperation agreement between Cuba and Venezuela emerged to automate the process of submitting projects, and track them. Due to problems encountered in the previous version, was decided the performing of this research to analyze a series of theoretical and practical measures to solve them. For this research was conducted a new analysis of requirements, metrics for design and implementation, analysis of algorithms complexity and stress testing, achieving the customer satisfaction and greater application performance.

PALABRAS CLAVE: Convenio, Proyectos, Cuba, Venezuela, Sistema.

KEYWORDS: Agreement, Projects, Cuba, Venezuela, System.

Guerrero Grey, L. W., y Medel Viltres, Y. (2014). Proceso de mejora del Sistema de Gestión de Proyectos para Cuba y Venezuela. *Campus Virtuales*, Vol. III, Num. 2, pp. 16-22. Consultado el [dd/mm/aaaa] en www.revistacampusvirtuales.es

1. Introducción

El Convenio Integral de Cooperación entre la República de Cuba y la República Bolivariana de Venezuela ha dado muchos frutos desde su surgimiento. Una gran cantidad de proyectos han beneficiado a ambas partes y este número sigue en aumento en el marco de la Alternativa Bolivariana para las Américas (ALBA). En las Comisiones Mixtas Cuba-Venezuela se conciben una serie de proyectos bilaterales, los cuales deben seguir un conjunto de pasos hasta su contratación y firma. Inicialmente y durante algún tiempo la elaboración, aprobación y financiamiento de las Fichas Técnicas en el marco de las Mixtas se realizaba de forma manual lo que provocaba una demora significativa en el proceso y se hacía difícil el control total de la cantidad de proyectos propuestos y del monto que representaban. Por esta razón se decidió la creación del Sistema de Gestión para el Convenio Integral de Cooperación Cuba-Venezuela (CCV); además de brindar un mejor seguimiento a los proyectos que se ejecutaban, facilitar una herramienta que brindara mayor variedad en la obtención de los reportes para la toma de decisiones y mejorar el almacenamiento de la información relacionada con los proyectos de forma íntegra y segura.

La primera versión del sistema fue desarrollada en tiempo de acuerdo al cronograma de desarrollo propuesto y una vez desplegado jugó un papel importante en la IX Mixta Cuba-Venezuela. Es este evento, la presentación de las fichas de los proyectos se realizó a través del sistema, presentándose un total de 305 proyectos; donde estuvieron involucrados 71 ministerios y 273 instituciones por ambas partes. Con estos resultados el sistema fue merecedor de la condición de destacado y fue reconocido por su contribución a que Albet S.A.[1] alcanzara el récord histórico de exportaciones en materia de software.

A pesar de todo esto se detectaron algunas deficiencias que afectaban el rendimiento del sistema. Fue realizado un profundo análisis con vistas a detectar los problemas y darle solución de forma inmediata. Desde el punto de vista visual la interfaz de usuario poseía componentes que provocaban una demora a la hora de mostrar la información solicitada. Desde el punto de vista de la implementación los principales problemas radicaban en la sobrecarga de las clases, la poca utilización de Patrones de Diseño, siendo posible emplear muchos más que ayudaran al desarrollo de la solución y la poca reutilización entre módulos dependientes entre sí debido a la fusión de las fases de Diseño e Implementación. Por todo esto el objetivo de la presente investigación es implementar el módulo de presentación de proyectos del Sistema de Gestión para el Convenio Integral de Cooperación Cuba-Venezuela y aplicar técnicas que permitan mejorar el rendimiento de la misma.

2. Materiales y métodos

Como punto de partida para la nueva versión fue necesario reanalizar los requisitos que la entidad cliente solicitaba, en este caso el Ministerio de Energía y Petróleo de Venezuela, para refinarlos y obtener una mejor descripción de los mismos. Este proceso estuvo regido por lo que establece la fase de Inicio de la metodología RUP, la cual divide el proceso de desarrollo en cuatro fases: Inicio, Elaboración, Construcción y Transición; con 9 flujos de trabajo: Modelamiento del negocio, Requerimientos, Análisis y Diseño, Implementación, Pruebas, Despliegue, Administración de configuración y cambios, Administración del proyecto y por último el flujo Ambiente. Cada uno de estos flujos tienen un mayor o menor peso de acorde a la fase de desarrollo en que se encuentre el proyecto.

[1] Empresa cubana comercializadora de software.

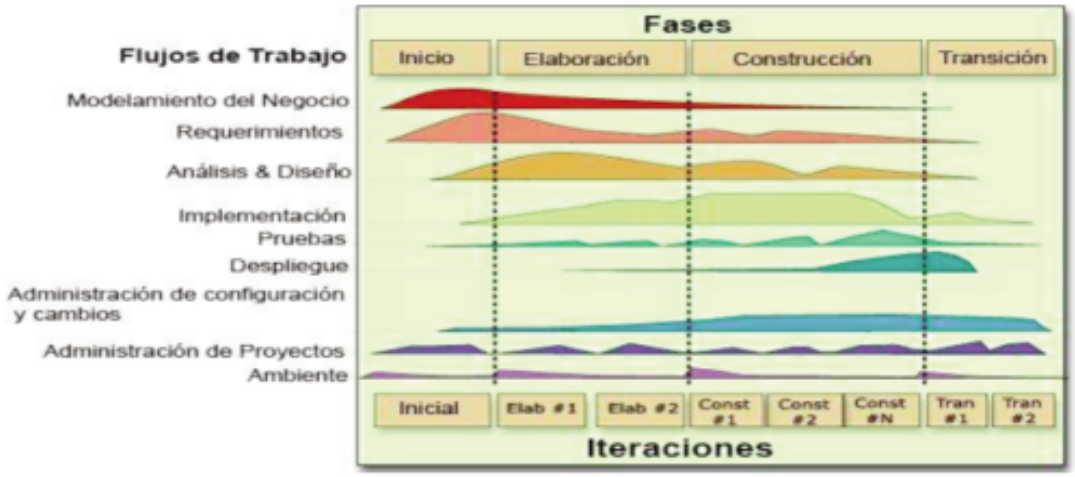

Figura 1. Fases de RUP.

La selección de la metodología RUP fue debido a que permite, entre otras cosas, la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible. Permite además llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para las actividades críticas. Proporciona guías explícitas para áreas tales como modelado de negocios, arquitectura Web, pruebas y calidad. Y por último, unifica el equipo de desarrollo de software y mejora la comunicación al brindar a cada miembro del mismo una base de conocimientos, un lenguaje de modelado y un punto de vista de cómo desarrollar software. (1)

La obtención del nuevo diseño resultó imprescindible pues posibilitó crear una entrada apropiada y un punto de partida para las actividades de implementación, donde jugó un papel muy importante el marco de trabajo OpenXava el cual permite desarrollar aplicaciones JavaEE/J2EE de forma rápida y eficiente. La filosofía subyacente es definir con anotaciones de Java o con XML y programar con Java, pero cuanto más se define y menos se programe, mejor. El objetivo principal de esta herramienta es hacer que las cosas más típicas en una aplicación de gestión sean fáciles de hacer, mientras que ofrece la flexibilidad suficiente para desarrollar las funciones más avanzadas y especificas. Con el uso de OpenXava no es necesario modelar el esquema de la base de datos ya que al estar incluidas las etiquetas JPA[2] en la propia clase, cuando se actualiza el esquema de la base de datos se convierte cada clase declarada en una tabla de la base de datos, siendo los atributos las columnas. Tampoco es necesario crear una interfaz de usuario personalizada, ya que el marco de trabajo genera una interfaz con los atributos de la clase, según su tipo de dato. Estos tipos de datos son soportados perfectamente por PostgreSQL, en cual fue utilizado para la base de datos.

2 Java Persistence API.

```
package presentacion.modelo;
@Entity
@Table(schema = "presentacion")
@EntityValidators({
    @EntityValidator(value = base.validadores.ValidadorCamposErroresRepetidos.class, properties = {
        @PropertyValue(name = "mensaje", value = "required")})
})
public class Proyecto extends Documento {
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "duracion_meses")
    @DescriptionsList(descriptionProperties = "numero",orderByKey=true)
    @NoCreate
    @NoModify
    @NoSearch
    @ReadOnly(forViews = "vista_revisar_Modificar_Ficha_seguimiento,vista_Modificar_Ficha_seguimiento
    @Required
    @DefaultValueCalculator(
            value = base.calculadores.PrimeroEnlaLista.class,
            properties={ @PropertyValue(name="idDefecto", value="1")}
    )
    private Meses duracionMeses;
```

Figura 2. Ejemplo de Clase con JPA.

Todo este proceso de desarrollo implicó la búsqueda de mejores soluciones desde el punto de vista teórico y práctico. Dentro de estas soluciones estuvo la aplicación de patrones GRASP y GoF. GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns, que se traduce como: patrones generales de software para asignar responsabilidades. Los patrones GRASP son parejas de problema solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Por su parte, los patrones GoF son un conjunto de 23 patrones de diseño propuestos por Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides; autores conocidos como la Pandilla de los Cuatro (Gang of Four) de los cuales toman nombre estos patrones. Estos patrones están agrupados en 3 categorías: Creacionales, Estructurales y de Comportamiento.

```
package presentacion.acciones;
import org.openxava.actions.ViewBaseAction;

public class CancelarFinanciamiento extends ViewBaseAction {
    public void execute() throws Exception {
        // TODO Auto-generated method stub
        returnToPreviousView();
    }
    @Override
    public String getNextMode() {
        return LIST;
    }
}
```

Figura 3. Aplicación del patrón Bajo Acoplamiento en el proyecto.

Otro de los elementos que se tuvieron en cuenta fue la complejidad de los algoritmos y métodos implementados. Desde el punto de vista computacional, fue necesario disponer de alguna forma de comparar una solución algorítmica con otra, para conocer cómo se comportarían cuando fuesen implementadas, especialmente ante gran cantidad de datos de entrada. La complejidad algorítmica es una medida teórica que se aplica a los algoritmos en este sentido. Si bien existe el concepto de Complejidad Espacial referido a la cantidad de memo-

ria utilizada por determinado algoritmo para ejecutarse, el concepto de Complejidad Temporal cobra mucha más importancia debido a que problemas de memoria pueden ser solucionados aumentado la misma pero no es factible que un algoritmo consuma enormes cantidades de tiempo para realizar determinada tarea. La utilización de un marco de trabajo en la nueva versión, hizo necesario el cálculo de la complejidad de los métodos reutilizados del mismo. Los métodos utilizados del marco de trabajo fueron en su mayoría métodos de acceso, sin ningún procedimiento en extremo complicado; basados en estructuras condicionales. Una muestra de los resultados de estos cálculos es la siguiente tabla:

Función	Clase	Versión	Complejidad
crearFichaProyecto	FichaProyectoController	Anterior	$O(n^2)$
crearFichaProyecto	FichaProyectoController	Nueva	O(n)

El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software. (2) Las métricas utilizadas fueron: Tamaño Operacional de Clase, Relaciones entre Clases y Errores por KLOC[3] Dentro de los principales atributos de calidad que se incluyen en la aplicación de las métricas, se encuentran: Responsabilidad, Complejidad de implementación, Reutilización, Acoplamiento y Complejidad del mantenimiento.

Para la métrica Tamaño Operacional de Clase, con un total de 40 clases muestredas, los resultados fueron los siguientes:

Figura 4. Porciento de clases por responsabilidad asignada.

Figura 5. Porciento de clases por su complejidad.

[3] Medida de errores por Mil (Kilo) Líneas de (Of) Código, se considera buena si es menor que 5.(3)

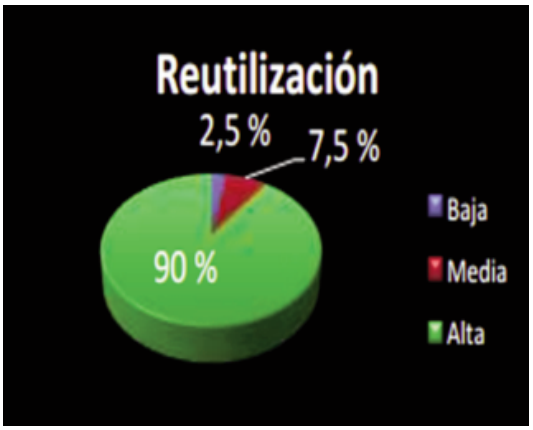

Figura 6. Porciento de clases por su nivel de reutilización.

El análisis de errores por mil líneas de código, para un total de 5420 líneas entre todas las clases de diferentes tipos y aproximando el total a 5 mil líneas de código como establece la métrica, arrojó que para los 3 errores detectados la media es de 0.6 errores por mil líneas de código, resultado que indica una alta calidad en la implementación realizada.

3. Resultados

Como último paso necesario para medir el rendimiento de la aplicación y compararlo con el rendimiento de la versión anterior se hicieron pruebas de Carga y Estrés con la herramienta JMeter. Las pruebas se realizaron en una computadora con CPU Core 2 Duo, 2 Gb RAM, Ubuntu 12.04 instalado, con un disco duro de 250 Gb, Postgres SOL 8.4, Tomcat 6.014 y JDK 6. Los resultados fueron los siguientes:

- **Muestras**: 8500 (Cantidad de páginas que simulan la cantidad de usuarios que están interactuando con el sistema desde la misma URL).
- **Media**: 3918 (Media de páginas que se cargaron de manera satisfactoria).
- **Mediana**: 2452 milisegundos (Tiempo promedio que han tardado en cargarse las páginas).
- **Mín**: 0 (Tiempo mínimo que ha demorado en cargarse una página).
- **Max**: 81244 milisegundos (Tiempo Máximo que ha tardado en cargarse una página).
- **Línea 90 %**: 7785 milisegundos (tiempo en que el 90 por ciento de las páginas se cargaron de manera satisfactoria).
- **% Error**: 0.08 (Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria).
- **Tiempos de Respuestas**: 2.27 segundos (Total del tiempo que demoró en cargarse la cantidad de usuarios de esa prueba).

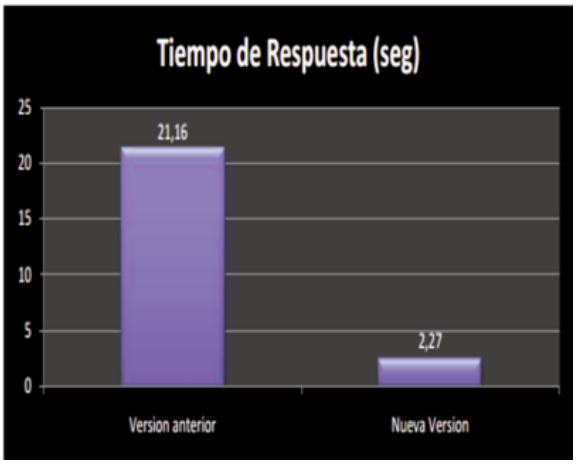

Figura 7. Comparación de tiempos de respuestas.

4. Discusión

El desarrollo de aplicaciones de gestión implica una serie de elementos que se deben tener en cuenta. Claridad en los requerimientos, selección de la tecnología adecuada para el tipo de desarrollo definido, metodología y validación son algunos de ellos. El sistema para la gestión de proyectos entre Cuba y Venezuela constituyó, por su necesidad, una herramienta imprescindible, que permitió llevar un control más eficiente de los mismos en los aspectos más importantes: coste, tiempo, alcance y recursos humanos.

Para que esta efectiva gestión fuese posible, el sistema debía tener unos parámetros de rendimiento aceptables, puesto que iba ser usado por varias entidades del estado venezolano y del estado cubano, por eso el tema del tiempo de respuesta fue tan importante. Luego de llevado a cabo todo el proceso de mejora, aplicando medidas teóricas y prácticas, el resultado fue satisfactorio. Se logró reducir casi en 20 segundos el tiempo de respuesta, una mejor organización del proyecto, una mejor arquitectura modular y una mejor comprensibilidad del código generado. Todo esto contribuyó a que el proceso de presentación, aprobación y seguimientos de proyectos y contratos entre Cuba y Venezuela fuera más eficiente.

Cómo citar este artículo / How to cite this paper

Guerrero Grey, L. W., y Medel Viltres, Y. (2014). Proceso de mejora del Sistema de Gestión de Proyectos para Cuba y Venezuela. Campus Virtuales, Vol. III, Num. 2, pp. 16-22. Consultado el [dd/mm/aaaa] en www.revistacampusvirtuales.es

Referencias

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). El Proceso Unificado de Desarrollo de Software. Addison Wesley.
Pressman, R. (2002). Ingieneria de software un enfoque práctico. McGraw Hill.
Grompone, J. (2004). Gestión de Proyectos de Software. Olmer S.A.