Universidad de las Ciencias Informáticas
Facultad 3

Título: “Componente dinámico para la interoperabilidad de procesos de negocio en el Sistema Integral de Gestión Cedrux.”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Christian García Ramírez

Raydel Pavón Reyes

Tutores: Ing. Magdanis Galván Rey

MSc. Pedro Manuel Nogales Cobas

La Habana, junio de 2013

“Año 55 de la revolución”

"La falla de nuestra época consiste en que sus hombres no quieren ser útiles sino importantes."

Winston Churchill.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Christian García Ramírez
_____

Raydel Pavón Reyes
_____

Ing. Magdanis Galvan Rey
_____

MSc. Pedro Manuel Nogales Cobas
_____

DATOS DE CONTACTO

Autor: Christian García Ramírez.

Correo electrónico: cgramirez@estudiantes.uci.cu

Autor: Raydel Pavón Reyes.

Correo electrónico: rpavon@estudiantes.uci.cu

Tutor: Ing. Magdanis Galvan Rey.

Correo electrónico: mgalvan@uci.cu

Tutor: MSc. Pedro Manuel Nogales Cobas.

Correo electrónico: pmnogales@uci.cu

AGRADECIMIENTOS

Quisiera agradecer a todas las personas que de una forma u otra han incidido en la realización de este trabajo, en especial a mi mamá Cándida, ya que siempre ha sido un sueño para ella ver a sus hijos graduarse. A mi madrasta Aida, a mi papá Raydel por ayudarme en los mementos difíciles y por sus buenos consejos. A mi abuela Eulalia, pues ha sido mi segunda madre. A mis hermanos Reycel, Javier, Dunia, Yoenia. A toda mi familia en general, a mi suegra por ayudarme al final de camino junto, a mi novia Yesmely, pues me ayudó grandemente cuando me ahogaba el estrés, por su preocupación, dedicación, entrega, confianza, amor y cariño, por no dudar de mí. A toda su familia, en especial a su tía Yudelsi, a su prima Magdalena , a Manuel, a una chiquitica malcriada que le dicen Araimis, a Lisandra, Pavel por su ayuda especial. A los compañeros de aula que pasamos 5 años juntos asumiendo los obstáculos como retos y a los profesores que contribuyeron a mi formación profesional. A mis tutores Magdanis y Pedro por apoyarnos y darnos ánimos cuando pensamos que todo se había terminado, por la noches sin dormir tratando de encontrar la solución a los problemas. A mi compañero de tesis La Magia por lidiar juntos hasta el final. A los compañeros de deporte y de edificio, en fin a todos. A la Revolución. ***Raydel***

Agradecer: primeramente a los miembros del tribunal por sus sugerencias que contribuyeron al perfeccionamiento de este trabajo. A mis tutores Magdanis y Pedro por su intención y empeño en dar lo mejor para el cumplimiento exitoso de este trabajo de diploma. A mi compañero de tesis Raydel, por su paciencia, dedicación y perseverancia y sobre todo por ser mi amigo. A Yesmely porque también aportó muchísimo con sus opiniones y consejos. A todos mis compañeros de aula, en especial a Armando y a Wilson por brindarme su apoyo no solo en este trabajo sino durante toda mi carrera, a Addiel y a Raúl porque se han comportado como verdaderos hermanos en los momentos difíciles. A los compañeros de proyecto Abraham, William y Yuniel por su apoyo en la implementación. A toda mi familia en especial a mis tíos Sergio, Nene y Madelín por creer en mí, brindarme su apoyo y cariño. A mi abuela Carmen por todas sus atenciones y cariño. A las dos personas que lo hacen todo posible, que para describir lo que representan solo necesito una frase de dos palabras: “Lo superlativo”. Mis padres, Iván y Pucha por brindarme tanto amor desde el mismo momento que llegue a este mundo, por siempre estar ahí para escucharme y empeñarse en dar los mejores consejos y apreciaciones y por comprenderme y perdonarme todas las veces que me he equivocado. ***Christian***

DEDICATORIA

Dedico este trabajo de diploma a mi mamá Candida Reyes porque siempre ha sido uno de sus sueños, y a mi abuela Eulalia Reyes por ser mi segunda madre.

Raydel

Dedico este trabajo a mi familia en especial a mis padres Iván y Pucha porque sin su apoyo y cariño nada de esto sería realidad, a mi abuela Carmen, por el orgullo que representa para mi tenerla como abuela, a mis tíos Sergio, Nene y Madelín, por lo bien que me hicieron sentir creyendo tanto en mí. A todos mis amigos en especial a Addiel, Raúl, Armando y Raydel porque con su afecto e interés me aportaron fuerzas para seguir adelante.

Christian

RESUMEN

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, por sus siglas en inglés) define la interoperabilidad como: “La habilidad de dos o más sistemas, redes de comunicación, aplicaciones o componentes para intercambiar información entre ellos y para usar la información que ha sido intercambiada”.

En el presente trabajo se hace un estudio del estado del arte sobre las diferentes definiciones, herramientas y modelos de interoperabilidad a fin de determinar puntos de reutilización y definir la posición de los autores respecto al estudio y comportamiento de la interoperabilidad de procesos en software de gestión. Se realiza el análisis y el diseño de la solución, generando los artefactos definidos por el modelo de desarrollo del centro CEIGE, para llevar a cabo el proceso de desarrollo de la solución propuesta. Por último se validan los resultados en correspondencia con el cumplimiento del objetivo propuesto en la investigación.

PALABRAS CLAVE: Interoperabilidad, proceso de negocio, Software de gestión

TABLA DE CONTENIDOS

ÍNDICE DE TABLA

ÍNDICE DE ILUSTRACIONES

TABLA DE CONTENIDOS

INTRODUCCIÓN

El desarrollo acelerado de las tecnologías a nivel mundial, el intercambio y manejo de la información entre sistemas homólogos y la utilización de la misma, han crecido considerablemente. Unido a este crecimiento surge la necesidad de interoperar información entre sistemas que permitan el flujo de datos de un modo factible.

La interoperabilidad es un término dado a conocer por diferentes autores y estándares reconocidos internacionalmente, entre las definiciones más conocidas se encuentra:

"La habilidad de dos o más sistemas, redes de comunicación, aplicaciones o componentes para intercambiar información entre ellos y para usar la información que ha sido intercambiada". [1].

Desarrollar la interoperabilidad para lograr la comunicación entre sistemas homólogos con tecnologías heterogéneas ha sido un tema complicado a lo largo de la historia, pero la misma se caracteriza porque:

- ✓ Mejora el intercambio de datos basado en estándares abiertos y publicados de forma libre.
- ✓ Ignora la heterogeneidad de los sistemas.
- ✓ Fortalece la integridad de la información y elimina su duplicidad. [3].

La necesidad de gestionar y automatizar procesos en casi todos los sectores de la sociedad como instituciones, empresas, escuelas u organizaciones, ha requerido del uso de las tecnologías efectivas para agilizar dichos procesos con el objetivo de lograr una economía de alta calidad. [5]. Esto ha propiciado el surgimiento de los sistemas de Planificación de Recursos Empresariales (ERP, por sus siglas en inglés), los cuales son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa. Se caracterizan por estar compuestos por diferentes partes integradas en una única aplicación". [6].

Como parte del desarrollo informático en Cuba, y la búsqueda de soluciones que contribuyan al avance de la economía; se desarrolla en la Universidad de las Ciencias Informáticas, específicamente en el Centro de Informatización de la Gestión de Entidades (CEIGE), un Sistema Integral de Gestión denominado Cedrux. Este sistema, parte de la idea de contar con una solución a nivel de país, que cumpla con los principios de independencia tecnológica y que esté basado en los procesos y particularidades de la economía cubana. [5].

El sistema Cedrux es un paquete de soluciones integrales de gestión para las entidades presupuestadas y empresariales. [7].

Este sistema cuenta con un componente de interoperabilidad. Sin embargo cuenta con algunas insuficiencias que para el equipo de desarrollo son necesarias mejorar.

- ✓ Depende de los procesos que se deseen interoperar.
- ✓ Provoca la reimplementación de la solución para cada especificación de procesos.
- ✓ Influye en la duplicidad del código.
- ✓ Incrementa el esfuerzo realizado por los programadores.
- ✓ Aumenta el tiempo de desarrollo por parte del equipo de trabajo.

Desarrollando el componente dinámico para la interoperabilidad de procesos de negocio que se desea, el sistema:

- ✓ Permitiría la homogenización del proceso de interoperabilidad.
- ✓ Lograría la generación dinámica de las interfaces específicas por cada subproceso.

Después de un análisis de lo antes expuesto se define como **problema a resolver:** ¿Cómo homogeneizar la interpretación de nuevos procesos de negocio, para el intercambio de información en el Sistema Integral de Gestión Cedrux?

Para ello se plantea como **objeto de estudio** de la investigación el proceso de interoperabilidad de software en sistemas de gestión.

Se define como **objetivo general** desarrollar un componente dinámico que permita homogeneizar la interpretación de nuevos procesos de negocio, a partir del lenguaje de ejecución de procesos de interoperabilidad en el Sistema Integral de Gestión Cedrux.

Para dar solución al objetivo general se definieron los siguientes **objetivos específicos**:

- ✓ Elaborar el marco teórico de la investigación a fin de determinar puntos de reutilización y definir la posición de los autores.
- ✓ Realizar el análisis y el diseño de la solución.

- ✓ Implementar la solución propuesta.
- ✓ Validar la solución a partir de las métricas de diseño y las pruebas de caja blanca y caja negra.

Se especifica como **campo de acción** la interoperabilidad en el Sistema Integral de Gestión Cedrux.

Se define como **idea a defender** si se desarrolla un componente dinámico a partir del lenguaje de ejecución de procesos de interoperabilidad se permitirá homogeneizar la interpretación de nuevos procesos de negocio en el Sistema Integral de Gestión Cedrux.

Durante el transcurso de esta investigación se recurrieron a varios métodos científicos:

Analítico – Sintético: a partir del análisis de los referentes teóricos y la bibliografía en cuestión se realizará una síntesis de los elementos significativos de la investigación.

Hipotético – Deductivo: este método ocupó un papel importante en el proceso de verificación de la idea a defender, permitiendo establecer conclusiones y predicciones a partir de la solución desarrollada.

Histórico – Lógico: para sistematizar las tendencias actuales de la interoperabilidad en el desarrollo de los sistemas de gestión y determinar su influencia en el problema actual de la investigación.

Entrevista: se utilizó para conocer en detalles y comprobar los problemas relacionados con la interoperabilidad, existentes en del Sistema Integral de Gestión Cedrux.

La revisión de documentos: para definir los aspectos importantes y relevantes a tono con la interoperabilidad en los sistemas de gestión, así como para la determinación del estado del arte del objeto de investigación de este trabajo.

La modelación: se utiliza para la creación de abstracciones que explican la realidad, ejemplo de esto se puede ver en los modelos y diagramas presentados en cada fase del proceso de desarrollo.

Estructura del contenido.

Capítulo 1. "*Fundamentación teórica*". Se exponen los fundamentos generales que sirven de soporte teórico en la solución del problema. Se definen elementos de interoperabilidad, algunos modelos, iniciativas y herramientas que permiten la interoperabilidad, el uso de la programación dinámica, el modelo

de desarrollo por el cual es guiado el desarrollo del sistema, así como las tecnologías a utilizar para la implementación de la solución.

Capítulo 2. "*Diseño de la solución*". Se exponen los requisitos a cumplir por el componente de interoperabilidad, los artefactos generados durante el diseño, así como los patrones implícitos en la solución.

Capítulo 3. "*Validación de la propuesta".* Se exponen los estándares de codificación, los artefactos generados durante la implementación de la solución, así como las métricas de diseño, las pruebas de caja negra y caja blanca para la validación, y los casos de pruebas diseñados.

El documento cuenta además con las **Conclusiones** del trabajo, algunas **Recomendaciones** a tener en cuenta para la puesta en práctica del componente para la interoperabilidad de procesos de negocio en el Sistema de Gestión Cedrux, la **Bibliografía** y un **Glosario de Términos**.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se darán a conocer los diferentes conceptos que serán de vital importancia para entender el trabajo investigativo desarrollado. Se realiza un estudio relacionado con la interoperabilidad entre sistemas de gestión a nivel nacional e internacional, sus ventajas, buenas prácticas, tipos o dimensiones y los niveles alcanzados por la misma. Se hace un estudio de un conjunto de herramientas, iniciativas y modelos que proporcionan la interoperabilidad, así como la importancia del uso de la programación dinámica en el desarrollo del componente de interoperabilidad. Se realiza una descripción de las herramientas de modelado, tecnologías de desarrollo para la solución, además del modelo de desarrollo definido por el Centro de Informatización de la Gestión de Entidades (CEIGE).

1.1 Interoperabilidad de software

Dado que la interoperabilidad en muchos casos se trata de un modo estructurado, varias organizaciones e iniciativas han definido diferentes modelos de la misma. El término interoperabilidad tiene muchas connotaciones, incluyendo los objetivos de la comunicación, el intercambio de información, la cooperación y el uso compartido de recursos entre distintos tipos de sistemas. De hecho, la esencia de la interoperabilidad es garantizar las relaciones entre los sistemas, donde cada relación es una forma de compartir, de comunicar, de intercambiar y cooperar. [8]. Existen múltiples definiciones de la interoperabilidad defendidas por varios autores y estándares reconocidos a nivel internacional; entre ellas se encuentran las siguientes:

- ✓ La interoperabilidad se define como la capacidad de los sistemas de información, de las tecnologías de la comunicación y de los procesos de negocio para intercambiar datos, compartir la información y el conocimiento. [8].
- ✓ La interoperabilidad consiste en la disponibilidad de mecanismos que permitan intercambiar procesos y/o datos entre sistemas heterogéneos. [5].
- ✓ La interoperabilidad es la capacidad de intercambiar y compartir datos entre distintos sistemas que utilizan diferentes arquitecturas de equipos y programas informáticos, distintas interfaces y estructuras de datos. [10].

- ✓ La habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada. [1].
- ✓ La capacidad de diferentes productos y servicios de las Tecnologías de la Información y las Comunicaciones (TIC) para intercambiar y usar datos e información con el objetivo de funcionar juntos en un entorno conectado en red. En su acepción más simple, la interoperabilidad trata de asegurar que los sistemas trabajen juntos. [2].
- ✓ La interoperabilidad es la capacidad para comunicar, ejecutar programas, o transferir datos entre varias unidades funcionales sin necesitar que el usuario tenga conocimiento de las características de esas unidades. [11].
- ✓ La habilidad de organizaciones y sistemas dispares y diversos para interaccionar con objetivos consensuados y comunes y con la finalidad de obtener beneficios mutuos. La interacción implica que las organizaciones involucradas compartan información y conocimiento a través de sus procesos de negocio, mediante el intercambio de datos entre sus respectivos sistemas de tecnología de la información y las comunicaciones. [12].

Después de valorar y realizar un análisis de los criterios y definiciones expuestas anteriormente sobre la interoperabilidad, se considera que la interoperabilidad está dada por la cualidad que tienen diferentes sistemas para el envío y recepción de la información, y poder utilizar la información intercambiada.

1.2 Ventajas de interoperabilidad

Desarrollar una plataforma de interoperabilidad, basada en el conocimiento de las dimensiones de la misma, trae grandes beneficios en cuanto a intercambio de información. [53]. Entre las cuantiosas ventajas se pueden mencionar:

- ✓ Mejora el intercambio de datos basado en estándares abiertos y publicados de forma libre.
- ✓ Ignora la heterogeneidad de los sistemas.
- ✓ Posee mecanismos eficientes de intercambio de información.
- ✓ Fortalece la integridad de la información y elimina su duplicidad.

- ✓ Logra la disponibilidad y visibilidad de los datos entre ambos sistemas.
- ✓ Permite la optimización de los procesos de interacción e intercambio de información entre las entidades.
- ✓ Aumenta su reusabilidad en otros proyectos, dando la posibilidad de integrar sistemas en el futuro sin necesidad de grandes inversiones.
- ✓ Los sistemas interoperan de forma transparente, coherente y segura. [3].

1.3 Buenas prácticas de interoperabilidad

Con el desarrollo actual de las tecnologías se ha incrementado el intercambio de datos entre sistemas. Existen varias formas o propuestas que ayudan a desarrollar la interoperabilidad de un software. Algunas de las buenas prácticas se mencionarán a continuación para que puedan ser aplicadas en el desarrollo de la presente investigación:

- ✓ Establecer políticas de seguridad y privacidad coherentes y fáciles de llevar a cabo, acordes con el nivel de amenaza y tipo de información a proteger.
- ✓ Buscar la accesibilidad de los contenidos y el conocimiento, con independencia de la tecnología que se use para acceder a ellos.
- ✓ Si es posible, se debe utilizar un lenguaje común para que los diferentes sistemas integrales de gestión puedan interactuar e intercambiar información de manera óptima y eficiente. [13].
- ✓ Se debe construir una plataforma como solución de interoperabilidad para el intercambio de información, independiente de hardware o software.
- ✓ Uso exclusivo de estándares abiertos, convirtiendo a estos estándares toda la información que se encuentre en formatos privados, o que se encuentre almacenada usando estándares abiertos en situación de abandono. [13].
- ✓ Siempre que sea posible, se debería optar por el uso de software libre. Así mismo, es frecuente que el software libre ayude a definir nuevos estándares abiertos y sus especificaciones están disponibles públicamente.

- ✓ Si es posible, se deben usar aplicaciones o herramientas de desarrollo, que puedan funcionar en más de una plataforma tecnológica. En este sentido, las aplicaciones JAVA, o tecnologías como el XML, serán de gran ayuda para lograr los objetivos de interoperabilidad sin caer en las trampas de la dependencia tecnológica. [13].

1.4 Dimensiones de interoperabilidad

La interoperabilidad es una característica esencial para arquitecturas de información, enlazada para trabajar en entornos heterogéneos a lo largo del tiempo. Sin embargo, emplear y entender el concepto es aún muy heterogéneo: está concebido en una relación con objetos o en una perspectiva funcional, desde la perspectiva de un usuario o de una institución, en términos de significados técnicos y protocolos. [14]. Esta puede analizarse desde diferentes dimensiones (catorce dimensiones), dependiendo del contexto en que se desarrolle.

Interoperabilidad técnica

Es aquella que permite el intercambio de información en su nivel más básico, abarca las cuestiones técnicas (hardware, software, telecomunicaciones), necesarias para interconectar sistemas computacionales y servicios. Se preocupa de los problemas que existen para intercomunicar sistemas y servicios heterogéneos". En esta interoperabilidad existen aspectos claves como el uso de interfaces y estándares abiertos, servicios de interconexión, integración de datos, presentación de datos e intercambio de información, accesibilidad y la garantía de seguridad de los servicios. [8].

Interoperabilidad semántica

Es aquella que posibilita el intercambio de información, utilizando un vocabulario común y compartido que posibilite la interpretación del significado de los términos. Se ocupa de asegurar que el significado preciso de la información que se intercambia es entendido por otra aplicación que no fue diseñada inicialmente para ese propósito". Este tipo de interoperabilidad es la que permite que las aplicaciones recombinen información de varias fuentes y la puedan procesar de una forma coherente. [8].

Interoperabilidad organizacional

Está relacionada con las metas que se desean conseguir, el modelado de los procesos y la necesaria colaboración entre los elementos participantes, que deben poder intercambiar información a pesar de tener distintas estructuras internas y procesos. Se ocupa de definir los objetivos de negocios, modelar los

procesos y facilitar la colaboración de administraciones que desean intercambiar información y pueden tener diferentes estructuras organizacionales y procesos internos. [8].

Después de revisar y analizar la bibliografía a tono con el concepto de interoperabilidad, varios autores definen un conjunto importante de dimensiones (catorce dimensiones) a tener en cuenta para desarrollar la interoperabilidad. Aunque la identificación de estas depende mucho del contexto en que se quiera desarrollar; se considera tomar en cuenta las dimensiones técnicas, semánticas y organizacionales.

1.5 Niveles de interoperabilidad

El modelo de referencia Niveles de Interoperabilidad de los Sistemas de Información (LISI, por sus siglas en inglés) incrementa las capacidades de interconexión e interoperabilidad entre las interacciones de sistemas con el objetivo de incrementar los niveles de sofisticación en la interacción entre sistemas. Este modelo define 5 niveles numerados del 0 al 4, definidos a continuación:

Nivel 0. Aislada: sin conexión.

Los sistemas de Nivel 0 no tienen conexión electrónica directa. El intercambio de datos entre estos sistemas se produce mediante una entrada manual con el teclado o con un sistema de almacenamiento común.

Nivel 1. Conectada: conexión electrónica, datos y aplicaciones separados.

Los sistemas de Nivel 1 están conectados de forma electrónica. Estos sistemas permiten el intercambio peer-to-peer (punto a punto) de tipos homogéneos, tales como texto, ficheros gráficos. Generalmente los sistemas de Nivel 1 permiten el intercambio de ficheros de forma simple.

Nivel 2. Funcional: funciones comunes mínimas, datos y aplicaciones separados.

Los sistemas de Nivel 2 son sistemas distribuidos que residen en redes locales que permiten transferir conjuntos de datos complejos y heterogéneos (imágenes, mapas, etc.) de un sistema a otro. Un aspecto clave de los sistemas o aplicaciones de este nivel es su capacidad para proporcionar acceso vía Web a los datos. En estos sistemas están presentes formatos de modelos de datos (lógicos y físicos), pero generalmente existe un modelo de datos aceptado entre los distintos programas, y después cada programa define su modelo de datos propio. Generalmente, los usuarios son capaces de compartir información integrada entre sistemas o funciones.

Nivel 3. Dominio: datos compartidos, aplicaciones separadas.

El nivel 3 se caracteriza por las múltiples interacciones entre aplicaciones. Los sistemas y aplicaciones de nivel 3 son sistemas integrados, capaces de interconectarse mediante Redes de Área Amplia (WAN, por sus siglas en inglés) que permiten a múltiples usuarios acceder a los datos. En este nivel las implementaciones generalmente tienen una visión localizada de un espacio de información distribuido y a través de un dominio operativo o funcional, de manera que la información es compartida por aplicaciones independientes.

Nivel 4. Empresarial: manipulación interactiva, datos y aplicaciones compartidas.

El nivel 4 es el objetivo último de los sistemas de información que persiguen la interoperabilidad, en el que los sistemas son capaces de operar empleando un espacio de información global distribuido a través de distintos dominios. Múltiples usuarios son capaces de interactuar con datos complejos de forma simultánea. Los datos y las aplicaciones son completamente independientes y pueden ser distribuidas en este espacio para soportar la integración o agrupación de información. [15].

Después de estudiar los niveles de interoperabilidad según LISI, y teniendo en cuenta la base tecnológica predominante en los sistemas de Gestión Empresarial a los que se pretende aplicar la solución, se propone el desarrollo de un componente dinámico de interoperabilidad; el cual permita el intercambio y utilización de la información mediante correo o dispositivos electrónicos y con éste alcanzar el primer nivel de interoperabilidad definido por LISI.

1.6 Proceso de negocio

Un proceso es “un conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados”. [16].

Un proceso de negocio es un completo y coordinado conjunto de actividades transaccionales y de colaboración, que entrega valor a los clientes o se encarga de cumplir otras metas estratégicas de la entidad. Cada proceso de negocio tiene sus entradas, funciones y salidas. Las entradas son requisitos que deben tenerse antes de que una función pueda ser aplicada. Cuando una función es aplicada a las entradas de un método, tendremos ciertas salidas resultantes. Un proceso de negocio puede ser parte de un proceso mayor que lo abarque o bien puede incluir otros procesos de negocio que deban ser incluidos en su función. En este contexto un proceso de negocio puede ser visto a varios niveles de granularidad[1].

[1] Es más fácil reutilizar unidades más pequeñas dado, que de este modo, es posible seleccionar aquellas partes que son interesantes y descartar aquellas que no son adecuadas en el contexto donde se encuentre.

El enlace entre procesos de negocio y generación de valor lleva a algunos practicantes a ver los procesos de negocio como los flujos de trabajo que efectúan las tareas de una organización. Estos procesos pueden ser medidos y están orientados al rendimiento, tienen resultados específicos, entregan resultados a clientes o stakeholders[2], responden a alguna acción o evento específico y las actividades deben agregar valor a las entradas del proceso. [17].

1.7 Programación dinámica

La programación dinámica es un enfoque general para la solución de problemas en los que es necesario tomar decisiones en etapas sucesivas. Las decisiones tomadas en una etapa condicionan la evolución futura del sistema, afectando a las situaciones en las que el sistema se encontrará en el futuro (denominadas estados), y a las decisiones que se plantearán en el futuro. Conviene resaltar que a diferencia de la programación lineal, el modelado de problemas de programación dinámica no sigue una forma estándar. Así, para cada problema será necesario especificar cada uno de los componentes que caracterizan un problema de programación dinámica. [19]. La programación dinámica no sólo tiene sentido aplicarla por razones de eficiencia, sino porque además presenta un método capaz de resolver de manera eficiente problemas cuya solución ha sido abordada por otras técnicas y ha fracasado. Donde tiene mayor aplicación la programación dinámica es en la resolución de problemas de optimización. En este tipo de problemas se pueden presentar distintas soluciones, cada una con un valor, y lo que se desea es encontrar la solución de valor óptimo (máximo o mínimo). La solución de problemas mediante esta técnica se basa en el llamado principio de óptimo enunciado por Bellman en 1957. [20].

La programación dinámica es un factor clave en el desarrollo del componente dinámico que se desea, ya que a partir de un lenguaje de ejecución para procesos de interoperabilidad (LEPI), se ejecuta un componente dinámico que posibilita la generación dinámica de las interfaces por cada especificación de subproceso del componente antes mencionado.

1.8 Herramientas, Iniciativas y modelos que brindan la interoperabilidad

A continuación algunas de las herramientas encontradas en la bibliografía a tono con la interoperabilidad en los sistemas de gestión.

[2] Quienes pueden afectar o son afectados por las actividades de una empresa.

1.8.1 Herramientas que proporcionan la interoperabilidad

DSpace

DSpace ha sido desarrollado conjuntamente por los laboratorios Hewlett-Packard y por la biblioteca del Instituto de Tecnología de Massachussets (MIT, por sus siglas en inglés) específicamente para constituir servicios de repositorio institucional. Es un sistema diseñado para la captura, almacenamiento, indización[3], preservación y redistribución de la producción intelectual del personal investigador de la universidad en formato digital. Distribuido mediante una licencia Distribución de Software Berkeley (BSD, por sus siglas en inglés). Está programado en Java y corre sobre sistemas tipo Unix, aunque ya se han realizado con éxito pruebas de instalación en Windows XP. Precisa de un servidor Web Apache y una base de datos relacional de código abierto PostgreSQL 8.0 o superior, permitiendo también el uso de Oracle 9.0 o posterior.

La interfaz Web básica que ofrece por defecto es totalmente configurable según las necesidades de cada institución o comunidad de usuarios, mediante Lenguaje de Marcas de Hipertexto (HTML, por sus siglas en inglés) y Java. Utiliza el motor de búsqueda de software libre Lucene. [21].

OPUS

Repositorio Institucional de la Universidad de Stuttgart (OPUS, por sus siglas en inglés), fue creado como resultado de un proyecto de la biblioteca y el centro computacional de la Universidad de Stuttgart, Alemania. Esta Universidad ofrece en la página Web de OPUS, un interfaz de búsqueda simultánea en todos o algunos (a elección) de los repositorios que emplean este sistema. Programado en PHP, este software corre sobre sistemas Linux o Solaris, y cualquier tipo de servidor Web. Admite cualquier formato de ficheros y ofrece la función del administrador para seleccionar los formatos aceptados. Permite convertir documentos electrónicos a Formato de Documento Portátil (PDF, por sus siglas en inglés) asignando automáticamente algunos metadatos que se almacenan en formato Dublin Core sin cualificar empleando una base de datos MySQL. La recuperación tiene por tanto las funciones típicas de otras bases de datos en línea: varios campos de búsqueda, opciones de truncamiento y operadores booleanos. Permite realizar importación y exportación de ficheros y metadatos, su interfaz es fácilmente configurable a la imagen de la institución que lo implementa, y permite obtener fácilmente estadísticas e informes de uso del sistema. Las principales funciones para futuros desarrollos se centran en servicios

[3] De acuerdo a la norma [52], la indización es el proceso donde se describe o representa la temática o contenido de un recurso de información, cuando se realiza este proceso se elaboran índices que son una herramienta útil de búsqueda y recuperación de información.

de repositorios interoperables, como estadísticas de uso y análisis de citaciones en una red de repositorios certificados. [21].

MyCoRe

El sistema MyCoRe se desarrolla en el marco del proyecto MILESS de la Universidad de Essen, Alemania. Es resultado de la colaboración de un consorcio de universidades para ofrecer un conjunto de herramientas software que permitan crear bibliotecas digitales y soluciones archivísticas (Content Repositories- CoRe) que sean configurables y adaptables a los requisitos y necesidades locales de cada sistema sin esfuerzos de programación. Se compone de un conjunto de módulos que proporcionan las funciones básicas que podrían ser requeridas en una implementación de un repositorio, incluyendo: búsqueda distribuida sobre repositorios MyCoRe dispersos geográficamente, soporte integrado para transmisión en línea de audio y video, gestión de ficheros, y editores de metadatos en línea.

Además, ofrece un paquete para la implementación básica de un servidor de publicaciones, DocPortal. Las implementaciones locales pueden adaptar este conjunto básico para servir a sus requisitos particulares. El modelo de datos de MyCoRe es completamente configurable. Emplea tecnología Web como Java, Lenguaje de Marca Extensible (XML, por sus siglas en inglés) o Lenguajes Extensible de Hojas de Estilo (XSL, por sus siglas en inglés). No está restringido a una base de datos particular, sino que soporta varios sistemas de bases de datos: de tipo comercial como IBM Content Manager, y software libre como MySQL.

Los componentes principales de MyCoRe son: aplicación de metadatos del documento y de las personas, sistema de ficheros internos, sistema de clasificación jerárquica, funciones de gestión de flujo y carga de trabajo; interfaces de usuario y administrador; y búsqueda distribuida e interfaces. Está especialmente dirigido a repositorios de contenido multimedia y bibliotecas digitales, complementando las funcionalidades del sistema OPUS, utilizado de forma generalizada en este país para repositorios de publicaciones académicas. [21].

IBM Web Service Toolkit (WSTK)

Es un conjunto de punta de las herramientas y tutoriales para el uso de Java con Protocolo de Simple Acceso a Objetos (SOAP, por sus siglas en inglés), Lenguaje de Descripción de Servicios Web (WSDL, por sus siglas en inglés). El kit de la herramientas funciona con WebSphere Application Server y viene con

un servidor de aplicaciones incrustado en caso de no tener uno. Provee de un entorno de tiempo de ejecución (tanto del lado del cliente como del servidor), brinda ejemplos de algunas herramientas adicionales para diseñar y ejecutar aplicaciones basadas en servicios Web.

El objetivo del producto es facilitar el desarrollo de aplicaciones basadas en servicios Web. Trabaja como un plug-in para servidores de aplicaciones J2EE y actualmente está en la versión 3.0. Los servidores soportados son: IBM WebSphere y Web Sphere MicroEdition, y Jakarta Tomcat 4.0. Estos servidores ya tienen soporte para servicios Web, pero el WSTK le agrega algunas funcionalidades nuevas y aplicaciones ejemplo. A parte de incluir todos los estándares como SOAP, WSDL, Descripción Universal, Descubrimiento e Integración (UDDI, por sus siglas en inglés); el WSTK provee un conjunto de utilidades extras que ofrecen entre otras cosas servicios de identificación, notificación, medición de uso de servicios Web y contratación de servicios Web y servicios de acceso a datos basado en servicios Web. [22].

GLUE

Provee una plataforma tanto para desarrollar como para construir servicios Web para la plataforma de Java. El mismo soporta y es completamente compatible con los estándares XML, SOAP, WSDL y UDDI. Incorpora una Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés) muy completa que disminuye la complejidad del desarrollo y publicación de servicios Web. Este producto es desarrollado por la empresa The Mind Electric. Se destaca por su facilidad de uso (lo cual fue confirmado al implementar un prototipo para consumir servicios Web de .NET desde Java), su fiabilidad y robustez. [22].

Java XML Pack

Es un paquete provisto por Sun que incorpora un conjunto de APIs para XML y APIs estándar para el manejo de servicios Web. La API provista soporta SOAP, XML Protocolo (XMLP, por sus siglas en inglés) el cual es una especificación provista por W3C para la interacción entre objetos remotos basada en XML (están aún en construcción y se espera que finalmente sea basada en SOAP) y también soporte para WSDL y UDDI. El paquete está compuesto por cuatro APIs, JAXM para mensajería en general, basada en SOAP, JAXP para procesamiento de datos en formato XML, JAXR para registro de servicios (soporta UDDI) y JAX-RPC que permite realizar invocaciones a Web Services (basada en SOAP y con soporte WSDL). [22].

ETD-db

ETD-database o ETD-db es un depósito de documentos basado en un conjunto de páginas Web y guiones Perl que interactúan con una base de datos MySQL. Los guiones proporcionan una interfaz estándar para internautas, investigadores, autores, personal de universidades y de biblioteca, para que remitan y gestionen ficheros y metadatos relacionados con una colección de tesis y disertaciones electrónicas. Su descarga está disponible de forma gratuita para los miembros de la NDLTD, acompañada de diversos manuales para su instalación, mantenimiento y actualización. Para ello precisa de un servidor en activo, preferentemente con un sistema operativo tipo Unix, en el que se deberá instalar una base de datos MySQL y un servidor Web Apache.

Para cumplir con la versión 2.0 del protocolo OAI-PMH, precisa de un módulo adicional también disponible para su libre descarga, que incorpora utilidades para la transformación del esquema de metadatos específicos, definidos por la NDLTD a los metadatos que define el protocolo, es decir, Dublin Core sin cualificar. [21].

CDSware

CDSware permite ejecutar su propio servidor de pre-impresión electrónica, su propio catálogo de la biblioteca en línea o un sistema de documentos en la Web. Se cumple el OAI recolección de metadatos (protocolo Iniciativa de Archivos Abiertos-Protocolo de Recolección de Metadatos (OAI-PMH, por sus siglas en inglés)) y utiliza MARC como su estándar bibliográfico subyacente. Utiliza como base de datos MySQL y servidor Apache (PHP, Python) servidor de aplicaciones Web. Posee un potente motor de búsqueda con Google. Otros servicios independientes de la plataforma se pueden integrar en CDSware, tales como el servidor de conversión como un servicio adicional proporcionado dentro de la instalación CERN CDSware. Se puede utilizar después de la instalación. Posee dos formas de intercambio de metadatos: la recolección jerárquica y la recolección recíproca. [21].

Tabla 1. Comparación entre herramientas.

Herramientas	Plataforma	Libre / Privativa	Tecnologías de desarrollo	Mecanismo para generar interfaces	Lenguajes de programación
CDSware	Unix	Privativa	MySQL, Oracle, Apache, WML	x	PHP, Python, Java

DSpace	Unix	Privativa	Apache Tomcat, Apache Maven, JDK, PostgreSQL 8.0, Oracle 9.0, Lucene	x	Java, HTML
OPUS	Solaris, Linux	Libre	MySQL	x	PHP
WSTK	Windows	-	SOAP, UDDI, Servidores IBM WebSphere, Jakarta Tomcat 4.0	-	WSDL
GLUE	Windows	-	UDDI, SOAP	-	XML, WSDL, Java
Java XML Pack	-	-	SOAP, UDDI, JAXP, JAXM, JAXR, JAX-RPC	-	XMLP, WSDL, XML
MyCoRe	Solaris, Linux	libre	MySQL, eXist XML: DB	x	Java, XML, XSL
EDT-db	Unix	libre	MySQL, Apache	-	-

Fuente: Elaboración propia.

1.8.2 Iniciativas que brindan la interoperabilidad

DoKS

Documento y la aplicación de Intercambio de Conocimiento (DoKS, por sus siglas en inglés) es una iniciativa de la biblioteca de la Escuela Superior Católica Kempen, en colaboración con la Universidad Católica de Lovaina y otros socios, trata de proporcionar una herramienta que facilite el almacenamiento, intercambio y recuperación de documentos (e información en general) a través de Internet. La arquitectura del software se basa en un conjunto de estructuras Java de código abierto entre las que se encuentra OAIcat para la implementación del protocolo OAI-PMH. Funciona sobre cualquier plataforma con Java, pudiéndose utilizar sobre Windows o Linux. Es necesario instalar los paquetes Java, un sistema de gestión de bases de datos, Apache Ant y el servidor Web Tomcat. Es independiente de la base de datos, por lo que puede funcionar sobre MySQL, Microsoft SQL Server, Oracle, PostgreSQL, etc. Esta

herramienta se construye sobre un sistema de registros (*records*) y carpetas (*folders*). Contiene un módulo de guiones similar a JavaScript, BeanShell, que hace a los ficheros y carpetas extensibles y configurables, y cuenta con un mecanismo de permisos para controlar el acceso a registros y carpetas, lo que implica que DoKS permite definir usuarios y grupos de usuarios. Cuenta con un conjunto de funcionalidades implementadas comunes a otros sistemas de archivos abiertos, como la configuración de múltiples conjuntos de metadatos, interfaz Web multilingüe y configurable, la importación y exportación en XML, exportación de metadatos en formato MARCXML, o la búsqueda por texto libre y por campos de metadatos. [49].

Diseño de servicios Web para dar soporte a la Gestión de Procesos de Negocio

Esta iniciativa es una propuesta basada en SOA para facilitar el diseño de servicios de negocio reutilizables, identificando aquellos de mayor valor para el negocio y facilitando un marco para su implementación. Tiene como aspectos fundamentales: Eliminar la redundancia de funcionalidades, facilitar la reutilización de funcionalidad de negocio haciendo invisible la complejidad tecnológica, facilitar los mecanismos para poder escoger, de forma dinámica, la mejor oferta en cada momento; son aspectos que reflejan una adecuada visión acerca de la gestión integrada de los Sistemas y Tecnologías de la Información. Todas estas características se pueden encontrar en las SOA, específicamente diseñadas para promover la interoperabilidad. Este enfoque enfatiza la creación de redes virtuales de cooperación que persiguen un objetivo superior común, para garantizar un nivel de colaboración aceptable y la circulación segura de la información compartida entre los miembros de la red. [23].

Tabla 2. Comparación entre iniciativas.

Iniciativas	Tecnologías de desarrollo
DoKS	Apache Ant, Servidor Web Tomcat, MySQL, Oracle, PostgreSQL.
Diseño de servicios Web para dar soporte a la Gestión de Procesos de Negocio	SOAP

Fuente: Elaboración propia.

1.8.3 Modelos que definen la interoperabilidad

Modelo Middleware para la Integración de Agentes Móviles Inteligentes con Redes de Sensores Inalámbricos

La propuesta de este modelo de interoperabilidad middleware es para facilitar la integración entre la tecnología de programación basada en agentes y la tecnología basada en dispositivos de WSN (acrónimo inglés para Wireless Sensor Networks).

La interoperabilidad en este sistema busca que las partes en el modelo se integren a través de capas de abstracción, estas de manera transparente proveen los mecanismos necesarios para su integración. [24].

Modelo de interoperabilidad IDA

Este modelo resume la descripción de una arquitectura acordada por la comunidad Intercambio de Datos entre Administraciones (IDA, por sus siglas en inglés) para establecer interoperabilidad entre redes trans-Europeas, y por lo tanto, facilitar el intercambio de datos entre administraciones públicas en Europa. La arquitectura IDA es de vital importancia para el intercambio de datos y la colaboración entre estado miembros e instituciones. El Consejo Europeo continuamente presta una atención especial a su desarrollo, implementación y recomendación de guías de diseño. La Comisión Europea espera que a través del programa IDA se alcance un elevado grado de interoperabilidad entre las redes telemáticas en los estados miembros**.** Los estándares de interoperabilidad recomendados por IDA usan XML y esquemas XML como protocolo destacado para intercambio de datos.

Para simplificar la interconexión y descripción de servicios, IDA recomienda, además, unas serie de tecnologías: SOAP y Servicios Web, Presentación e intercambio de datos, entre otras. [25].

Tabla 3. Comparación entre modelos.

Modelos	Plataforma	Tecnologías de desarrollo	Lenguajes de programación
IDA	Plataforma Web CIRCA	SOAP, navegador Web	XML, XSD
Middleware		WSN, FIPA, ACL	

Fuente: Elaboración propia.

Luego de haber buscado en la bibliografía consultada, referente al objeto de estudio y campo de acción se encontraron los modelos anteriormente expuestos, lo que no se descarta la existencia de otros modelos en este sentido.

Después de un estudio y análisis de las herramientas, iniciativas y modelos descritos anteriormente que proporcionan la interoperabilidad, se concluye los siguiente: están orientados a servicios Web, la generación dinámica de las interfaces o vistas no está presente en ellas, un gran porciento son privativas, y las condiciones tecnológicas y de infraestructura predominante en el sistema de Gestión Empresarial al que se pretende aplicar la solución, se ve limitado por una arquitectura SOA utilizada por la mayoría de éstas; por lo que cambiar la filosofía hacia una arquitectura de este tipo requiere tiempo y recursos.

1.9 Lenguajes de ejecución

A continuación se realiza el análisis de los lenguajes BPEL y WS-BPEL por la necesidad de entender y poder redefinir posteriormente el Lenguaje de Ejecución para Procesos de Interoperabilidad (LEPI).

BPEL

Lenguaje de Ejecución de Procesos de Interoperabilidad (BPEL, por sus siglas en inglés), es un lenguaje de programación destinado para la ejecución de procesos empresariales, derivado de XML y persigue lograr un modelo de programación a gran escala. Un documento BPEL define el proceso, o la orquestación y la lógica de las acciones que serán ejecutadas por los motores de orquestación. El programa constituye en sí, el código fuente de la aplicación que ejecuta el proceso; el motor de orquestación actúa como una máquina virtual capaz de ejecutar código BPEL. El objetivo principal del diseño BPEL es definir una serie de conceptos de orquestación de servicios Web que pretenden ser usados por vistas internas o externas de un proceso de negocio, aunque también puede utilizarse para otras cuestiones como son:

- ✓ Proveer sistemas de control jerárquicos y de estilo gráfico, que permitan que su uso sea lo más fusionado e íntegro posible. Esto reduciría la fragmentación del espacio del modelado de procesos.

- ✓ Proveer funciones de manipulación simple de datos, requeridas para definir datos de procesos y flujos de control.

- ✓ Soportar un método de identificación de instancias de procesos que permita la definición de identificadores de instancias a nivel de mensajes de aplicaciones. Los identificadores de instancias deben ser definidos por socios y pueden cambiar.

BPEL le permite a los desarrolladores crear programas que automatizan las interacciones entre los servicios Web, jugando un papel clave en las arquitecturas orientadas a servicios. Esta automatización de la interacción entre los servicios Web es comúnmente referida como orquestación de servicios Web. [25].

WS-BPEL

El Lenguaje de Ejecución de Procesos de Interoperabilidad Orientados a Servicios (WS-BPEL, por sus siglas en inglés) es basado en XML, el cual permite especificar el comportamiento de un proceso de negocio basado en interacciones con servicios Web. [26]. Posee una sintaxis estándar definida en XML, puede mantener temporariamente estructura de datos en variables, las cuales pueden ser inspeccionadas y actualizadas para enviar datos desde un servicio a otro, poseen también un alcance determinado, donde su declaración es efectiva. Las actividades básicas incluyen la invocación de un servicio remoto, recepción de mensajes de manera determinística y no determinística, asignación de estructuras XML a variables. Define un modelo y una gramática para describir el comportamiento del proceso de negocios basados en la interacción entre un proceso y sus partners. La interacción de un proceso con cada uno de sus partners, se realiza a través de interfaces de Servicios Web y la estructura de la relación a nivel de interfaces se encapsula en lo que se denomina partner link. Define la manera de coordinar interacciones de servicios múltiples con estos partners para lograr el objetivo del negocio, tanto como el estado y la lógica necesaria para esa coordinación. Posee un mecanismo de concurrencia y necesita primitivas que permitan la exclusión mutua cuando se comparten variables que son accedidas por los flujos concurrentes. [27].

1.10 Tecnologías para la descripción estructural de los datos

Definición de tipo de documento (DTD)

Los **DTD** son definiciones de los elementos que puede incluir un documento XML, de la forma en que deben hacerlo (qué elementos van dentro de otros) y los atributos que se les puede dar. El DTD y los esquemas, proporcionan la gramática para una clase de documentos XML.

Estos mecanismos se utilizan para la llamada validación tanto estructural como formal del documento, esto es, enviar un documento a un destinatario junto con las condiciones que deben cumplir los documentos. [28].

Esquema XML (XSD)

Al igual que los **DTD**, los **XSD** describen el contenido y la estructura de la información, pero de una forma más precisa. Los esquemas indican tipos de dato, número mínimo y máximo de ocurrencias y otras características más específicas. Según la especificación del W3C *XML Schema*, los esquemas expresan vocabularios compartidos que permiten a las máquinas extraer las reglas hechas por las personas. Los esquemas proveen un significado para definir la estructura, contenido y semántica de los documentos XML. Un esquema XSD es algo similar a un DTD, es decir, define qué elementos puede contener un documento XML, cómo están organizados, y qué atributos y de qué tipo pueden tener sus elementos, pero la utilización de *schemas* ofrece nuevas posibilidades en el tratamiento de los documentos. [28].

Tabla 4. Ventajas de XSD respecto a DTD.

XSD	DTD
Usa la misma sintaxis que el XML.	Usa una sintaxis distinta al XML.
Gran flexibilidad para expresar tipos de datos.	Capacidad limitada para expresar tipos de datos.
Tipos de datos soportados (similares a los que se están en las bases de datos): más de 44.	Tipos de datos soportados (similares a los que se encuentran en las bases de datos): 10.
Permiten crear sus propios tipos de datos.	No permiten crear sus propios tipos de datos.
Orientado a objetos.	No orientado a objetos.
Puede definir conjuntos.	No puede definir conjuntos.
Puede especificar elementos únicos (clave).	No puede especificar elementos únicos (clave).
Puede definir elementos de contenido nulo.	No puede definir elementos de contenido nulo.
Puede definir elementos subtitulables.	No puede definir elementos subtitulables.

Fuente: Elaboración propia, a partir de [28].

Teniendo en cuenta que para modelar y validar la información se encuentra el uso de DTD y XSD. La utilización de XSD se ve favorecida por la facilidad de tratamiento que se deriva de la escritura en XML, en

lugar de emplear otros *parsers*[4] para su reconocimiento, y también porque el estándar está muy extendido. Otra gran ventaja es que gracias a *XSD* se pueden expresar un gran número de tipos de datos y jerarquizarlos. Además es posible expresar la carga semántica y esto es muy importante para el tratamiento de la información. Gracias a XML se han podido realizar intercambios de información entre aplicaciones distintas y mediante *XSD* se ha podido validar y estructurar su contenido.

1.10.1 Lenguaje de ejecución para procesos de interoperabilidad (LEPI)

El lenguaje de ejecución para procesos de interoperabilidad (LEPI) es un trabajo de pregrado realizado con el objetivo de sentar las bases para la posterior implementación de un componente dinámico de interoperabilidad, capaz de cargar de este las especificidades del subsistema a interoperar. Este lenguaje de ejecución está sustentado en las sintaxis de XML, se ajusta a una estructura según la necesidad de los procesos que describirá, teniendo en cuenta las reglas de etiquetado definidas por W3C[5] para documentos XML. [29]. Debido a que el LEPI es lo que recibe como entrada el componente dinámico que se desea desarrollar y teniendo en cuenta que el mismo adolece de elementos necesarios en la definición de la estructura, se hace imprescindible su redefinición. Pues XSD presenta una gran flexibilidad para expresar tipos de datos, es decir, para predefinir los tipos de datos que tendrán los datos a introducir en el XML a validar, da la posibilidad de crear nuevos tipos de datos, se decidió realizar la sustitución de la validación del LEPI con DTD por la de XSD. La ejecución de los subprocesos de Exportación e Importación fue estructurada en secuencia de actividades, se modificaron y añadieron etiquetas correspondientes a las vistas y servicios a utilizar.

1.11 Modelo de desarrollo orientado a componentes

El desarrollo del componente propuesto se guiará por el modelo de desarrollo del Centro para la Informatización de la Gestión de Entidades (CEIGE). Es un modelo basado en componentes, detalla el ciclo de vida (**Ilustración 1**) de sus proyectos, tiene en cuenta las actividades de cada una de las fases y áreas de procesos que plantea el nivel II de CMMI establecido en la UCI. El mismo establece los artefactos necesarios para el desarrollo de cada fase, los cuales se listan a continuación.

1. Modelo de Procesos de Negocio o Modelo de dominio.

2. Descripción de requisitos del cliente: Modelo Conceptual.

[4] Analizador gramatical; programa analizador, que divide la entrada de texto en pequeñas partes y las procesa.
[5] World Wide Web Consortium.

3. Especificación de requisitos de software (Descripción de requisitos).

4. Plan de Desarrollo de Software.

5. Diagrama de clases del diseño (por componentes o módulos).

6. Diagrama de interacción (secuencia o colaboración por requisito).

Ilustración 1. Ciclo de vida de proyectos del CEIGE.

Fuente. Elaboración propia, a partir de: Modelo de desarrollo CEIGE.

1.12 Entorno de desarrollo

1.12.1 Marco de trabajo Sauxe 2.0

El Marco de Trabajo Sauxe se ha estructurado de manera tal que facilite la reutilización de los diferentes componentes y que sea de fácil entendimiento para todos los programadores que desarrollen sobre el mismo. Sauxe utiliza la librería ExtJS para implementar la capa de presentación, se apoya en Zend, una extensión de Zend Framework para el desarrollo de la lógica del negocio y para la gestión de los datos utiliza para el acceso a la base de datos un sistema Mapeador de Objeto Relacional (ORM) denominado Doctrine. Este utiliza como patrón arquitectónico el Modelo Vista Controlador (MVC, por sus siglas en inglés). También tiene como propósito insertar la programación orientada a aspectos así como la inversión de controles. [30].

1.12.2 Zend Framework 1.11

Zend Framework (ZF) es un framework de código abierto para desarrollar aplicaciones Web y servicios Web con PHP 5. ZF es una implementación que usa código completamente orientado a objetos. La estructura de los componentes de ZF es única; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. [31].

1.12.3 ExtJS 2.2

ExtJS es una librería JavaScript ligera y de alto rendimiento, compatible con la mayoría de los navegadores que permite crear páginas e interfaces Web dinámicas. Es utilizado en la capa de presentación, por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional. [32].

Esta librería incluye componentes Interfaz de Usuario (UI, por sus siglas en inglés) personalizable, modelo de componentes extensibles, un API fácil de usar y licencias de códigos abiertos y comerciales. [32].

1.12.4 Visual Paradigm 8.0

Aunque es un software propietario; la UCI posee una licencia para el trabajo con Visual Paradigm, es una herramienta multiplataforma y orientada a Gestión de Procesos de Negocio (BPM, por sus siglas en inglés). Proporciona soporte al modelado visual con Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) 6.0, ofreciendo distintas perspectivas del sistema independiente de la plataforma y dotada de

una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software, así como garantizar la calidad del producto final. Posee entre sus principales características la posibilidad de crear un amplio conjunto de artefactos, utilizados con frecuencia durante la confección de software. Posibilita generar código a partir de los diagramas, para plataformas como .Net, Java y PHP.

1.12.5 PHP 5.2.6

Entre los lenguajes que se ejecutan en el servidor y procesan la lógica del negocio están PHP, JAVA, ASP y Perl. Se decide usar PHP, debido a que es el lenguaje definido en el marco de trabajo. El componente de interoperabilidad está desarrollado con el lenguaje PHP y la construcción de la plataforma de interoperabilidad tiene que adoptar PHP como lenguaje del lado del servidor. No obstante, sus características lo hacen ser uno de los lenguajes de programación Web más usado en el mundo, y éstas son también ventajas para el equipo de desarrollo que busca la solución de interoperabilidad.

Es un lenguaje muy eficiente mediante el uso de un único servidor, puede brindar millones de acceso al día, cuenta con un conjunto de bibliotecas incorporadas (como se ha diseñado para su uso en la Web, incorpora una gran cantidad de funciones integradas para realizar útiles tareas), es gratuito, su uso y aprendizaje es muy sencillo, está disponible para una gran cantidad de sistemas operativos diferentes. Se puede escribir código PHP en todos los sistemas operativos gratuitos del tipo Unix, como Linux y Fedora, versiones comerciales de Unix o en las diferentes versiones de Microsoft Windows, a diferencia de los productos comerciales de código cerrado, con PHP se pueden realizar modificaciones a los programas con toda libertad. [33].

1.12.6 XML

Lenguaje de Marca Extensible (**XML**, por sus siglas en inglés), deriva del lenguaje Estándar de Lenguaje de Marcado Generalizado (SGML, por sus siglas en inglés) y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones se deben comunicar entre sí o integrar información. Los expertos nombran varias ventajas que derivan de la utilización del XML. Es extensible (se puede añadir nuevas etiquetas tras el diseño del documento), su analizador es estándar (no requiere de cambios para cada versión del metalenguaje) y facilita el análisis y procesamiento de los documentos XML creados por terceros. Entre los lenguajes creados con XML, se destacan el Lenguajes Extensible de Hojas de Estilo (XSL, por sus siglas en inglés), entre otros. La validez de los documentos (su estructura sintáctica

se encuentre desarrollada correctamente) depende la relación especificada entre los distintos elementos a partir de una definición o documento externo. [34].

Se caracteriza porque: Permite proporcionar diferentes vistas sobre los datos (HTML, PDF, entre otros), dependiendo de quién sea el cliente, facilita la integración desde fuentes de datos heterogéneas, por ejemplo, páginas Web, distintas bases de datos, los documentos tienen una estructura que los hace legibles e inteligibles no sólo para los ordenadores, sino también para los humanos, las aplicaciones de XML son fácilmente extensibles mediante definiciones de nuevos tipos de documento (DTD). [34].

1.12.7 JavaScript

JavaScript es un lenguaje interpretado, esta característica lo hace especialmente idóneo para trabajar en la Web, son los navegadores que se utilizan para viajar por ella los que interpretan y, por tanto, ejecutan los programas escritos en JavaScript.

De esta forma, se envían documentos a través de la Web que llevan incorporados el código fuente de programas, convirtiéndose de esta forma en documentos dinámicos, dejando de ser simples fuentes de información estática. JavaScript comparte muchos elementos con otros lenguajes de alto nivel. [35].

1.12.8 UML 2.0

El Lenguaje de Modelado Unificado (UML, por sus siglas en inglés) es una especificación de notación orientada a objetos. Es utilizado con el fin de especificar y documentar un sistema de software, de un modo estándar. Incluye aspectos conceptuales tales como procesos de negocios y funciones del sistema. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto.

UML implementa un lenguaje de modelado común para todos los desarrollos por lo que se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

1.13 CONCLUSIONES PARCIALES

Después de realizar un análisis del marco teórico de la investigación referente a la interoperabilidad y algunas de las herramientas, iniciativas y modelos que la brindan, se puede llegar a las siguientes conclusiones:

- ✓ La programación dinámica es un factor clave en el desarrollo de esta investigación, a partir del LEPI se posibilita la generación dinámica de las interfaces por cada especificación de proceso de negocio que desee interoperar.
- ✓ Luego de revisar y analizar la bibliografía a tono con el concepto de interoperabilidad, varios autores definen un conjunto importante de dimensiones (14) a tener en cuenta para desarrollar la interoperabilidad. Aunque la identificación de éstas depende del contexto en que se quiera desarrollar; se considera tomar en cuenta las dimensiones técnicas, semánticas y organizacionales.
- ✓ Después de estudiar los niveles de interoperabilidad según el modelo LISI, y teniendo en cuenta la base tecnológica predominante en los sistemas de Gestión Empresariales a los que se pretende aplicar la solución, se propone el desarrollo de un componente dinámico de interoperabilidad, alcanzando el primer nivel definido por LISI.
- ✓ Las herramientas, iniciativas y modelos descritas anteriormente, que proporcionan la interoperabilidad, están orientados a servicios Web; un gran porciento son privativas, no permiten la generación dinámicas de las interfaces (vistas), y las condiciones tecnológicas existentes en el sistema Integral de Gestión Cedrux, imposibilitan la adopción de una arquitectura SOA utilizada por la mayoría de estas herramientas.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción

El siguiente capítulo muestra los resultados obtenidos durante el proceso de desarrollo de la solución, así como los artefactos generados. Se describe el modelo conceptual, se realiza la captura y descripción de los requisitos funcionales, así como los requisitos no funcionales. Se muestra el diagrama de clases del diseño, así como los diagramas de secuencias.

2.1 Modelo Conceptual

La construcción del modelo conceptual permite una mejor comprensión del dominio del problema; no es más que una representación visual de los conceptos u objetos del mundo real que son significativos para el problema; representando las clases conceptuales y los componentes de software. [36].

El modelo conceptual (**Ilustración 2)** muestra los conceptos y relaciones del componente dinámico de interoperabilidad.

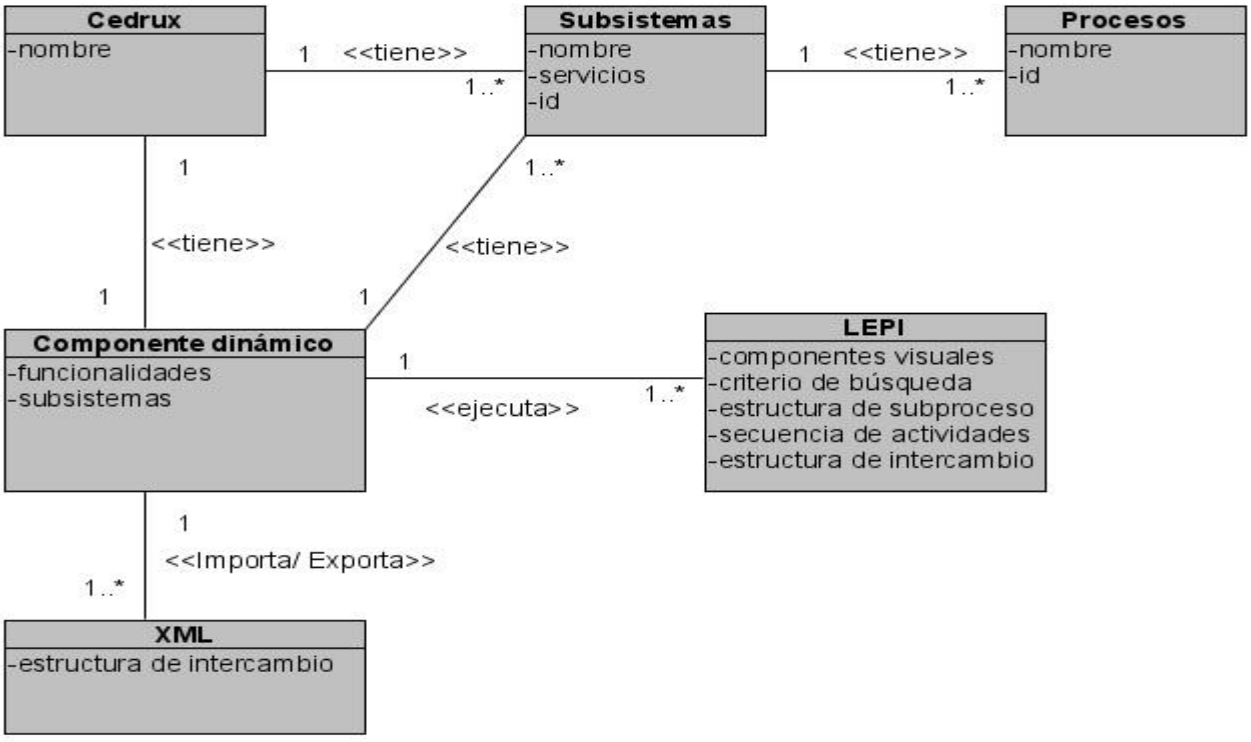

Ilustración 2. Modelo conceptual del componente de interoperabilidad.

Fuente: Elaboración propia.

A continuación se describen los conceptos implícitos en el modelo conceptual mostrado en la figura anterior.

Cedrux

En el modelo conceptual aparece como primer concepto Cedrux; el cual es un paquete de soluciones integrales de gestión para las entidades presupuestadas y empresariales, desarrollado siguiendo los principios de independencia tecnológica.

Subsistemas

Los subsistemas están contenidos dentro del sistema Cedrux, entre los que se pueden encontrar: Inventario, Capital Humano, Contabilidad, Seguridad, Estructura y Composición, Finanzas, Logística, Configuración, entre otros.

Procesos

Los procesos están incluidos dentro de los subsistemas, siendo estos de diferentes tipos como son: Configuración, Nomenclador de cuentas, Comprobante de operaciones, Análisis de consistencia, Cierre, Recuperaciones y Cuadre financiero.

Componente dinámico

El componente dinámico de interoperabilidad dará la posibilidad a varias soluciones desarrolladas sobre el marco de trabajo Sauxe, utilizarlo para la importación y exportación de datos, la generación dinámica de las vistas, evitando su reimplementación cuando cualquier subsistema necesite interoperar y permitirá reducir el tiempo de desarrollo por parte del equipo de trabajo.

Lenguaje de ejecución para procesos de interoperabilidad (LEPI)

El lenguaje de ejecución para procesos de interoperabilidad es un fichero que contiene los componentes visuales, las variables, peticiones, condiciones, secuencias, los servicios, y la estructura de intercambio definida para su utilización. El LEPI es lo que recibe como entrada el componente dinámico de interoperabilidad para la generación dinámica de las vistas de dicho componente.

XML

XML es el formato de salida y entrada del componente dinámico de interoperabilidad, o sea de la exportación e importación, el lenguaje de ejecución LEPI está estructurado en formato XML.

2.2 Requisitos de software

Los requisitos cumplen un papel primordial en el proceso de desarrollo de software. Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales describen qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados. [36].

2.2.1 Requisitos Funcionales

Para la captura de requisitos se aplicaron dos técnicas de las que se proponen en las bibliografías consultadas, Tormenta de ideas y Entrevistas, las cuales fueron realizadas en conjunto con profesionales del departamento de Tecnología del proyecto Sauxe. Se identificaron un total de 5 requisitos funcionales con una alta complejidad que debe cumplir el componente dinámico, a continuación una lista de ellos:

- ✓ RF: Cargar estructura de procesos.
- ✓ RF: Ejecutar proceso de interoperabilidad.
- ✓ RF: Generar vistas dinámicas.
- ✓ RF: Exportar procesos de negocio.
- ✓ RF: Importar procesos de negocio.

A continuación las descripciones de cada uno de los requisitos expuestos anteriormente.

✓ ***Requisito Funcional: Cargar estructura de procesos***

Tabla 5. Especificación del requisito Cargar estructura de procesos.

Precondiciones	El usuario debe tener permisos en el sistema
Descripción	
Flujo de eventos	
Flujo básico	
1	Acceder al sistema de gestión Cedrux.
2	Seleccionar el **Componente de Interoperabilidad**.
3	El sistema consume el servicio que devuelve todos los subsistemas.
3	El sistema muestra la interfaz principal del componente.

6	El sistema muestra la interfaz del proceso seleccionado y carga la estructura del mismo.
7	Concluye el requisito
Pos-condiciones	
1	El Sistema carga la estructura de proceso.
Flujos alternativos	
1	N/A
Pos-condiciones	
1	N/A
Validaciones	
1	N/A
Conceptos	Procesos Subsistemas
Requisitos especiales	N/A
	N/A

- ✓ ***Requisito Funcional: Ejecutar proceso de interoperabilidad***

Tabla 6. Especificación del requisito Ejecutar proceso de interoperabilidad.

Precondiciones	El usuario debe tener permisos en el sistema Debe seleccionarse un proceso a interoperar
Descripción	
Flujo de eventos	
Flujo básico	
1	Introducir los valores de los criterios de búsqueda del proceso que se desea interoperar.
2	El sistema dado los datos introducidos consume el servicio necesario para obtener los elementos que coinciden con dicho criterio.
3	El sistema genera las vistas necesarias para mostrar el proceso.
4	El sistema muestra los elementos que coinciden con el criterio de búsqueda.
5	Marcar los elementos que se desean interoperar.
6	Concluye el requisito
Pos-condiciones	
1	Se ejecuta el procesos de interoperabilidad.
Flujos alternativos 4.	
Pos-condiciones	

1	El sistema muestra mensaje de información **"**Debe seleccionar al menos un elemento.**"**
Validaciones	
2	N/A
Conceptos	Vistas Estructura Proceso
Requisitos especiales	N/A
Asuntos pendientes	N/A

- ✓ ***Requisito Funcional: Generar vistas dinámicas***

Tabla 7. Especificación del requisito Generar vistas dinámicas.

Precondiciones	El usuario debe tener permisos en el sistema
Descripción	
Flujo de eventos	
Flujo básico	
1	Seleccionar un subsistema en el árbol izquierdo denominado ***Subsistemas***.
2	Desplegar el subsistema seleccionado y seleccionar el proceso que desee interoperar.
3	El sistema comprueba que exista un **LEPI** para el proceso seleccionado.
4	El sistema carga del **LEPI** la descripción de las interfaces.
5	El sistema interpreta la descripción de las interfaces obtenidas del **LEPI**.
6	El sistema muestra la interfaz del proceso seleccionado.
7	Concluye el requisito
Pos-condiciones	
1	Se generan las vistas dinámicas.
Flujos alternativos	
2	N/A
Pos-condiciones	
1	N/A
Validaciones	
3	N/A
Conceptos	Procesos Subsistemas

Requisitos especiales	N/A

- ✓ ***Requisito Funcional: Exportar procesos de negocio***

Tabla 8. Especificación del requisito Exportar procesos de negocio.

Precondiciones	El usuario debe tener permisos en el sistema Debe existir al menos un proceso interoperable
Descripción	
Flujo de eventos	
Flujo básico	
1	Dar clic en el botón **Exportar.**
2	El sistema comprueba que se haya seleccionado al menos un elemento.
3	El sistema carga la estructura de intercambio desde el **LEPI**.
4	El sistema asigna los valores de los elementos seleccionados a las instancias de la estructura de intercambio.
5	El sistema comprime todas las instancias de la estructura de intercambio.
6	Seleccionar la dirección donde se desea guardar el comprimido.
7	Dar clic en el botón **Aceptar.**
3	El sistema descarga en la dirección especificada el comprimido con las instancias de la estructura de intercambio.
Pos-condiciones	
1	El sistema guarda el archivo con la estructura de intercambio en la dirección seleccionada.
Flujos alternativos	
3	N/A
Pos-condiciones	
1	N/A
Validaciones	
4	N/A
Conceptos	Estructura Proceso Subsistema
Requisitos especiales	N/A
Asuntos pendientes	N/A

- ✓ ***Requisito Funcional: Importar proceso de negocio***

Tabla 9. Especificación del requisito Importar proceso de negocio.

Precondiciones	El usuario debe tener permisos en el sistema Debe existir al menos un proceso interoperable
Descripción	
Flujo de eventos	
Flujo básico	
1	Dar clic en el botón **Importar.**
2	Seleccionar la dirección donde se encuentra la estructura de intercambio.
3	Seleccionar el subsistema donde serán introducidos los datos en la base de datos. .
4	Dar clic en el botón **Aceptar.**
4	El sistema comprueba que en la dirección especificada haya un archivo xml.
5	El sistema comprueba que el archivo xml cumpla con la nomenclatura predefinida en la estructura de intercambio.
6	El sistema consume un servicio, el cual guarda en base de datos en la sección correspondiente al subsistema seleccionado, los datos contenidos en el xml cargado.
Pos-condiciones	
1	El sistema importa el proceso seleccionado.
Flujos alternativos	
4	N/A
Pos-condiciones	
1	N/A
Validaciones	
5	N/A
Conceptos	Estructura Proceso Subsistema
Requisitos especiales	N/A
Asuntos pendientes	N/A

2.2.3 Validación de requisitos funcionales

Para la validación de los RF se efectuaron un conjunto de reuniones con el personal técnico del proyecto Sauxe, donde se descubrieron omisiones que habían sido cometidas en el proceso de obtención de los requisitos. Mediante la Revisión de Técnicas Formales se hizo una revisión por un equipo de profesionales del departamento de Tecnología, donde fueron detectadas 15 no conformidades en una primera iteración,

éstas relacionadas con errores ortográficos y desfasaje de las interfaces. En la segunda se encontraron 6, y éstas por introducción en los datos y la validación de los mimos. Finalmente en la última revisión no se detectó ninguna no conformidad.

2.2.2 Requisitos no Funcionales

Producto a que el desarrollo de este trabajo es guiado por el modelo de desarrollo del centro CEIGE, y los resultados obtenidos formarán parte del marco de trabajo desarrollado en dicho centro, los requisitos no funcionales a los que se debe ajustar el componente a desarrollar son los que fueron establecidos por el CEIGE al inicio del proceso de desarrollo. A continuación una lista de estos:

Software

Para el cliente:

- ✓ Navegador Mozilla Firefox 3.0 o superior.
- ✓ Sistema operativo Linux o Windows.

Para el servidor:

- ✓ Sistema operativo Linux en cualquiera de sus distribuciones.
- ✓ Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” incluida.

Seguridad

Autenticación y Autorización (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

Hardware

Para el servidor:

- ✓ Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- ✓ Al menos 40Gb de espacio libre en disco duro.

- ✓ Tarjeta de red.

Para el cliente:

- ✓ Requerimientos mínimos: Procesador Pentium III a 1GHz con 512Mb de memoria RAM.
- ✓ Tarjeta de red.

2.3 Diagrama de clases

El diagrama de clases del diseño describe la estructura del sistema, mostrando sus clases, asociaciones, atributos, métodos y las relaciones entre ellos. En la **Ilustración 3** se muestra el diagrama de clases del diseño con estereotipos Web identificado para el desarrollo del componente dinámico de interoperabilidad.

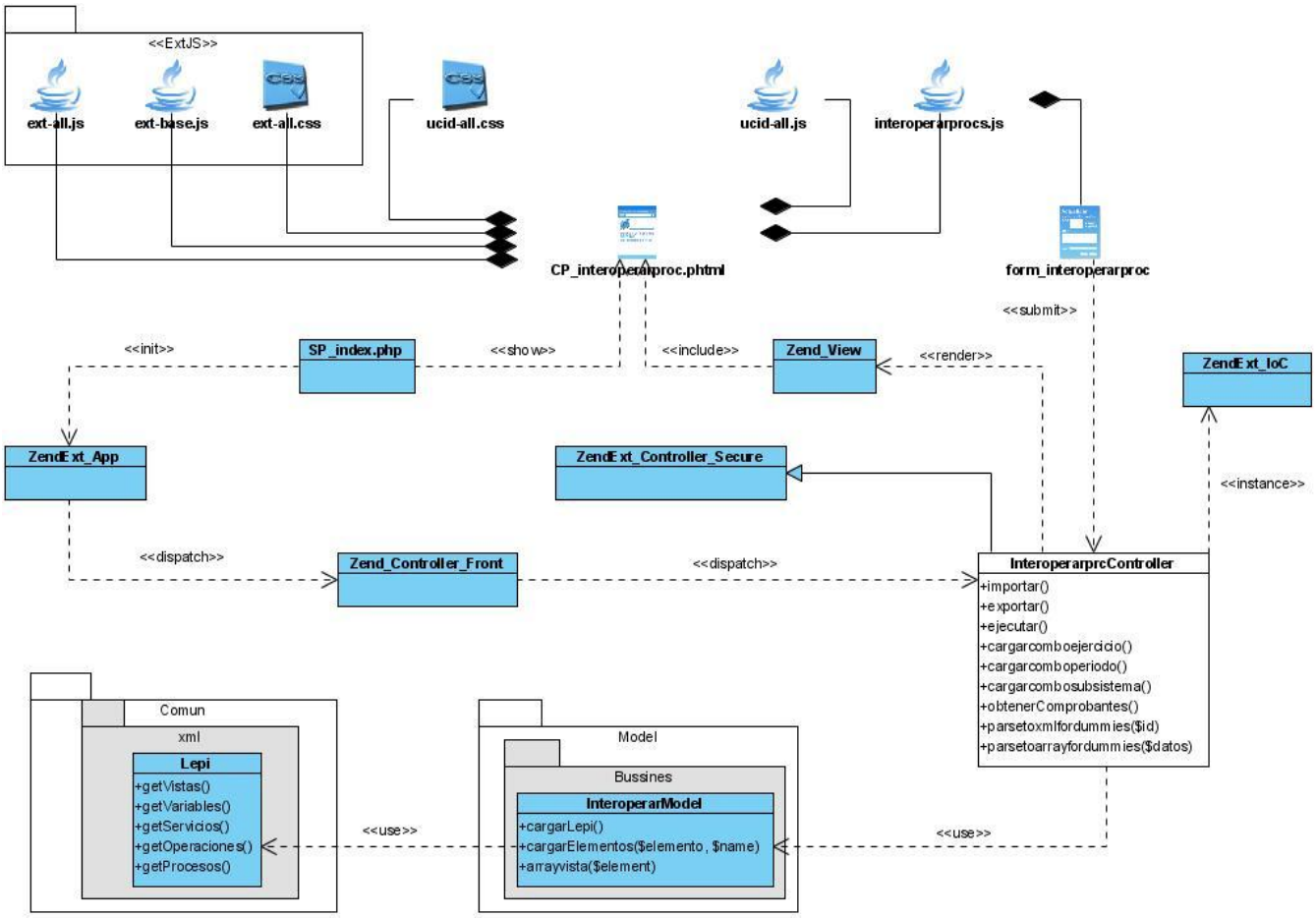

Ilustración 3. Diagrama de clases del diseño.

Fuente: Elaboración propia.

Para brindar un mejor entendimiento a los desarrolladores o programadores de las responsabilidades de cada clase representada en el modelo anterior, a continuación se describen las mismas:

Tabla 10. Descripción de clases.

Clases	Descripción
Framework ExtJS	Es una librería construida con JavaScript que permite construir aplicaciones.
Zend_View	Es la clase que permite trabajar con la vista en el patrón Modelo-Vista-Controlador.
ZendExt_App	Este componente se encarga de realizar las configuraciones iniciales, verificaciones y validaciones. Es el punto de inicio a partir del cual se desencadenan un conjunto de acciones necesarias antes de realizar cualquier operación.
interoperarprocesos.phtml	Plantilla HTML encargada de mostrarle al usuario la interfaz correspondiente al componente con la que va a trabajar.
ZendExt_Controller_Secure	Clase de donde heredan todas las clases controladoras y se encarga de gestionar la seguridad.
interoperarprocesos.js	Formulario js a través del cual se maneja la información que introducen los usuarios del sistema.
Index.php	Se encarga de determinar cuál es la clase controladora que le corresponde a cada página cliente.
InteroperarprocesosController	Clase controladora que hereda de *ZendExt_Controller_Secure*. Encargada de realizar todas las funcionalidades del componente.
ObtenerModel	Se programa la lógica de negocio y las acciones de modificación que se van a realizar con el xml que contiene al LEPI.

ZendExt_IoC	Registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema, especifica respuestas deseadas a sucesos o solicitudes de datos concretas, en el orden necesario y ejecutando el conjunto de sucesos que tienen que ocurrir según los parámetros requeridos para poder hacer uso de un determinado servicio.
Zend_Controller_Front	Representa el controlador frontal de la aplicación. Se encarga de manejar las solicitudes y respuestas. Es manejado por *Index.php*.

Fuente: Elaboración propia.

2.4 Patrones de diseño

”Cada patrón describe un problema que ocurre una y otra vez en un entorno y describe también el núcleo de la solución al problema, de forma que puede utilizarse un millón de veces sin tener que hacer dos veces lo mismo”. Un patrón es la solución general, fruto de la experiencia, a un problema general que puede adaptarse a un problema concreto [37].

Durante el desarrollo de la solución y para un mejor entendimiento se utilizarán varios de los patrones generales de software, los cuales se presentan a continuación:

2.4.1 Patrones Generales de Asignación de Responsabilidades (GRASP)

Experto

Se estableció en el componente con la asignación de responsabilidades específicas por cada una de las clases del sistema. El componente cuenta con las clases controladoras, modelos y entidades, que poseen funcionalidades específicas atendiendo a la actividad que realizan y los procesos que gestionan. Encontrándose presente este patrón en las clases *interoperarprocesos.phtml, ZendExt_Controller_Secure, interoperarprocesos.js, InteroperarprocesosController.*

Creador

Su uso se puede ver en la clase controladora *InteroperarprocesosController*, ya que es la encargada de la creación de objetos de la clase *ObtenerModel*.

Controlador

El uso de este patrón está presente en la clase controladora *InteroperarprocesosController*, ya que es la encargada de llevar el control de todas las peticiones realizadas por el usuario.

Alta Cohesión

El uso de éste patrón indica que la información almacenada en las clases debe ser coherente y relacionada a lo que se maneja en dicha clase, para esta solución se ve su uso igualmente en la clase controladora *InteroperarprocesosController*.

Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Este patrón da soporte a una mínima dependencia y a un aumento de la reutilización; una clase con bajo acoplamiento no depende de "muchas otras" clases para realizar sus tareas, permitiendo que se pueda reutilizar con mayor facilidad[6]. En las clases *Zend_View, ZendExt_App, ZendExt_Controller_Secure, Index.php, interoperarprocesos.js* se puede apreciar el uso de este patrón.

2.4.2 Patrón de Arquitectura

Modelo Vista Controlador (MVC)

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 distintos. El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la interfaz de usuario, el modelo está conformado por el acceso a datos y la lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista. En la **Ilustración 4** se muestra la utilización de este patrón. [51].

6 Cualidad de fácil. Disposición para hacer algo sin gran trabajo.

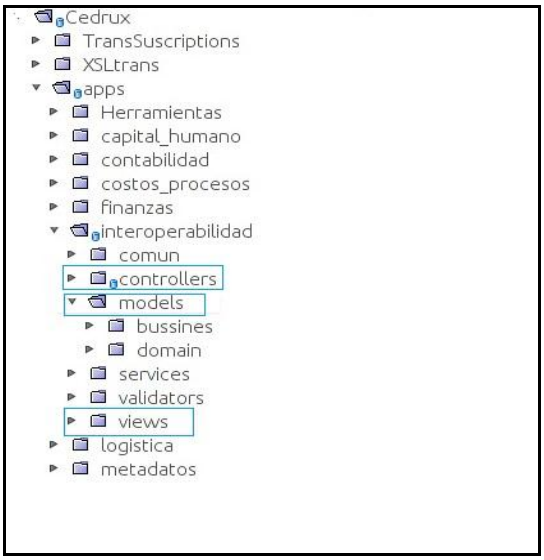

Ilustración 4. Aplicación del patrón Modelo Vista Controlador.

Fuente: Elaboración propia.

El Modelo: es la representación de la información que maneja la aplicación. Son los datos puros que puestos en un contexto del sistema son mostrados al usuario por medio del Controlador, proveen de información al usuario o a la aplicación misma.

La Vista: constituye la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En una aplicación Web la "Vista " es la página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

El Controlador: responde a las solicitudes del usuario desde la Interfaz, manejando los diferentes eventos a través de las funcionalidades necesarias y la información perteneciente al Modelo. [39].

2.5 Diagramas de secuencia

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con otros objetos y qué mensajes emiten esas comunicaciones. [40]. A continuación el diagrama de secuencia correspondiente al requisito funcional Cargar estructura de procesos.

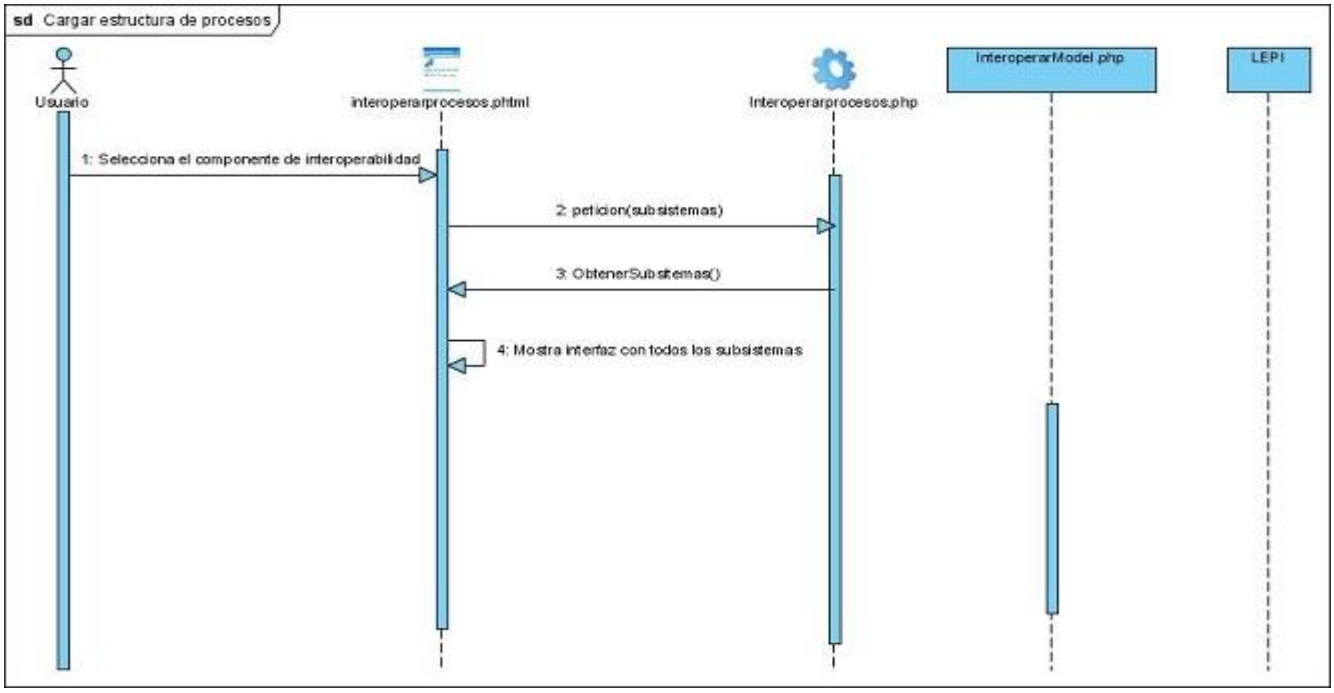

Ilustración 5. Cargar estructura de procesos.

Fuente: Elaboración propia.

El resto de los diagramas de secuencia se encuentran en [41].

2.6 CONCLUSIONES PARCIALES

- ✓ Se realizó el modelo conceptual posibilitando una mejor comprensión del dominio del problema; permitiendo así una representación visual de los conceptos u objetos que lo conforman. Encontrándose Cedrux como uno de los conceptos fundamentales junto con LEPI, Componente de interoperabilidad y XML, existiendo una estrecha relación entre ellos.

- ✓ Mediante el uso de las técnicas de captura de requisitos se identificaron 5 requisitos funcionales, todos de complejidad alta, así como los requisitos no funcionales para garantizar la correcta ejecución de la aplicación.

- ✓ Se mostró el diseño de clases para lograr una mayor comprensión de cómo están estructuradas las clases que conforman el componente dinámico de interoperabilidad.

- ✓ Se explicó el uso de diferentes patrones de arquitectura y diseño empleados en la aplicación.

- ✓ Se realizaron los diagramas de secuencia correspondientes a cada uno de los requisitos funcionales del sistema.
- ✓ Una vez concluido el diseño de la solución puede darse paso a los flujos de implementación y prueba de la misma.
- ✓ La solución propuesta, permite que se puedan obtener el conjunto de interfaces visuales para cada proceso a interoperar, tomando como base el Lenguaje de Ejecución del Proceso de interoperabilidad.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

En este capítulo se describen los estándares de codificación a utilizar para garantizar un buen entendimiento y legibilidad del código, y los componentes necesarios para realizar la implementación de la solución. Además se muestran los resultados de la evaluación mediante métricas de diseño, un caso de prueba y pruebas de caja blanca y de caja negra; métodos escogidos para validar la solución.

3.1 Estándares de codificación

Se define estándares de codificación, a un estilo de programación homogéneo en un proyecto, permitiendo que todos los participantes lo puedan entender en menos. [50].

Teniendo en cuenta lo antes referenciado, fueron definidas normas de codificación con el objetivo de obtener un estándar en la implementación, que permitiera asegurar la calidad del software y de esta forma obtener un código más legible y reutilizable.

A continuación se describen estos estándares empleados en la implementación del componente dinámico de interoperabilidad.

3.1.1 PascalCasing

El estándar PascalCasing establece que los identificadores, nombres de clases, variables, métodos o funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula. La nomenclatura de las clases del componente interoperabilidad se realizó sobre la base de este estándar, usando palabras compuestas, sugerentes y acordes al propósito de la misma. [50].

Nomenclatura de clases según su tipo

Controllers: clases controladoras del negocio. El nombre de la clase controladora debe estar estructurado por el nombre propio de la misma en mayúsculas seguido por la palabra Controller y heredar siempre de la súper clase del framework ZendExt_Controller_Secure. Ejemplo: ImportarprocesosController.

Clases de los modelos

Business: Clases modelo del negocio. Las clases modelo tendrán por identificador el nombre de la tabla en la que trabajan seguido por la palabra Model y heredarán de la súper clase del framework Zend_Ext llamada ZendExt_Model. Ejemplo: NomProcesoModel.

Domain: clases entidades del dominio. Los archivos situados en el domain tienen el mismo nombre de las tablas que representan, definiéndolas como clases php, pueden incluir los prefijos Dat o Nom para diferenciar entre sus usos. Ejemplo: NomProceso.

Generated: Clases bases del dominio. Las clases que se encuentran dentro de Generated comienza su nombre con la palabra: “Base”, seguido del nombre de la tabla en la Base de Datos. Ejemplo: BaseNomProceso.

Validators: Clases bases del dominio. El nombre de la clase validadoras se especifica con nombres compuestos con el distintivo que terminan en Validator. Ejemplo: ProcesoValidator.

Services: Clases que ofrecen los servicios de los componentes. Estas clases, de acuerdo con las operaciones que realizan y prestaciones que brindan, se definen con calificativos sugerentes, agregándoles la terminación Service. Ejemplo: ProcesoService.

3.1.2 CamelCasing

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos. [50].

Nomenclatura de las funciones

El identificativo a emplear para las funciones o métodos se escribe con la primera palabra en minúscula utilizando la notación CamelCasing y nombres que deduzcan su propósito. Ejemplo: cargarSubsistemas. Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo: cargarsubsistemasAction.

Nomenclatura de los atributos

Se definió que el nombre a emplear para las variables se escribe en minúscula y en caso de ser un nombre compuesto se escriben ambas palabras en minúscula separadas por un guion bajo. Ejemplo: id_subsistema y nombre_subsistema.

3.1.3 Notación húngara

Esta convención, también conocida como notación: REDDICK por el nombre de su creador, se basa en definir prefijos para cada tipo de datos según el ámbito de las variables con el fin de brindar mayor

información al nombre de la variable, método o función. La notación húngara fue utilizada para la definición de variables de acuerdo con los siguientes prefijos:

Tabla 11. Prefijos para la creación de variables.

Tipos de Datos	Prefijos
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
Float	flt
Boolean	bool

Fuente: Elaboración propia.

3.2 Modelo de implementación

En el modelo de implementación se comienza con el resultado de la etapa de diseño e implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El objetivo principal de la etapa de implementación es desarrollar la arquitectura y el sistema como un todo. De forma más específica, define la organización del código, planifica las integraciones del sistema necesarias para cada iteración e implementa los subsistemas y las clases encontrados durante el diseño. [42].

3.2.1 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre sus elementos. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo, especificando los subsistemas de implementación y sus dependencias.

Muestra como el sistema está dividido en componentes y las dependencias entre ellos, proveen una vista arquitectónica de alto nivel del sistema, ayuda a los desarrolladores a visualizar el camino de la implementación, permite tomar decisiones respecto a las tareas de implementación y las habilidades requeridas. [43].

En la **Ilustración 6** se muestra el diagrama de componentes correspondiente al componente dinámico de interoperabilidad.

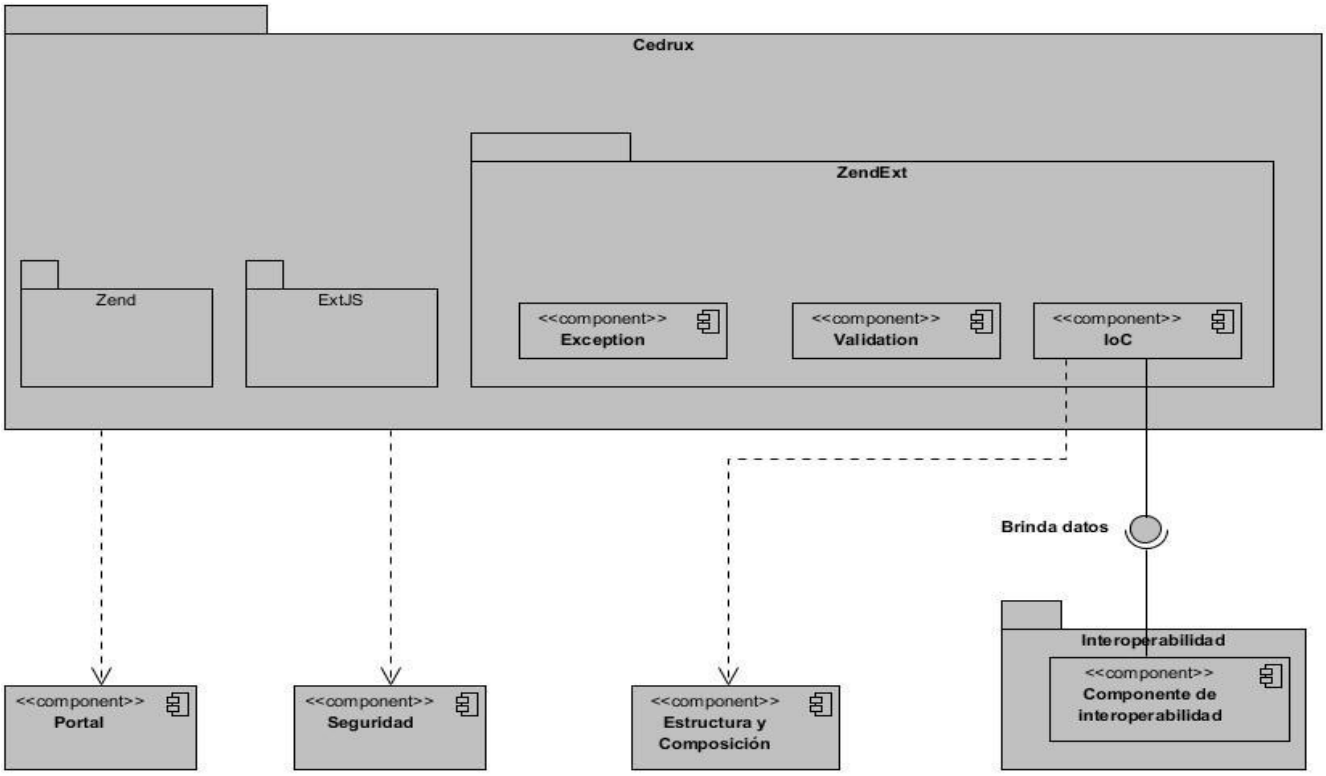

Ilustración 6. Diagrama de componentes.

Fuente: Elaboración propia.

3.3 Métricas de diseño

El Glosario de estándares del IEEE define métrica como una “medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo determinado”. Las métricas se centran en cuantificar tanto la funcionalidad, como la complejidad y eficiencia inmersa en el desarrollo de software, inclinando sus objetivos a mejorar la comprensión de la calidad del producto, estimar efectividad del proceso y mejorar la calidad del trabajo. [1].

3.3.1 Tamaño operacional de clase (TOC)

Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

A continuación se describen los tipos de afectaciones que se pueden observar al evaluar el diseño según la métrica.

Tabla 12. Atributos de calidad evaluados por la métrica TOC.

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento en la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Fuente: Elaboración propia.

Tabla 13. Criterios de evaluación para la métrica TOC.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Fuente: Elaboración propia.

Resultados obtenidos al aplicar la métrica de Tamaño Operacional de Clase (TOC)

A continuación se muestran los resultados obtenidos al aplicar la métrica TOC para un mejor entendimiento de los mismos.

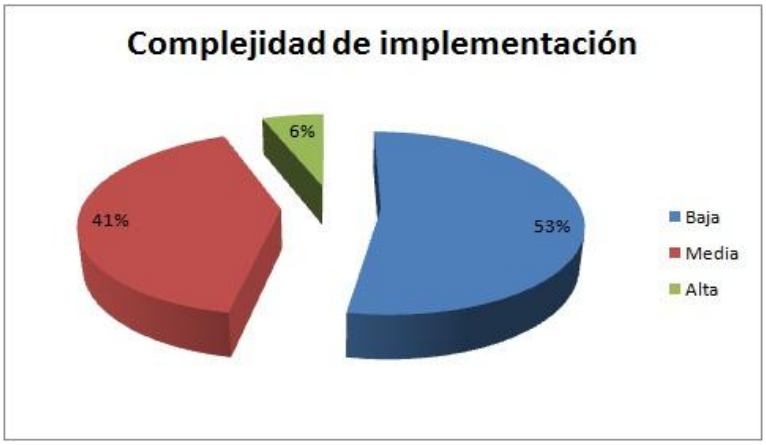

Ilustración 7. Resultados de la evaluar la métrica TOC para el atributo Complejidad de implementación. Fuente: Elaboración propia.

Ilustración 8. Resultados de la evaluación de la métrica TOC para el atributo Responsabilidad. Fuente: Elaboración propia.

Ilustración 9. Resultados de la evaluación de la métrica TOC para el atributo Reutilización.
Fuente: Elaboración propia.

Durante la evaluación de la métrica TOC los resultados obtenidos demostraron que los atributos de calidad de las clases se encuentran por encima del nivel medio satisfactorio a un 54%; de manera que se puede confirmar la elevada reutilización a un 75 % (elemento clave en el proceso de desarrollo de software) y la reducción en menor grado de la responsabilidad y la complejidad de implementación.

3.3.2 Relaciones entre clases (RC)

Dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

A continuación se describen los tipos de afectaciones que se pueden observar al evaluar el diseño según la métrica.

Tabla 14. Atributos de calidad evaluados por la métrica RC.

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento en la complejidad de mantenimiento de la clase.

Reutilización	Un aumento del RC implica una disminución del grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Fuente: Elaboración propia.

Tabla 15. Criterios de evaluación de la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Fuente: Elaboración propia.

Resultados obtenidos al aplicar la métrica de Relaciones entre Clases (RC)

A continuación se muestran los resultados obtenidos al aplicar la métrica TOC para un mejor entendimiento de los mismos.

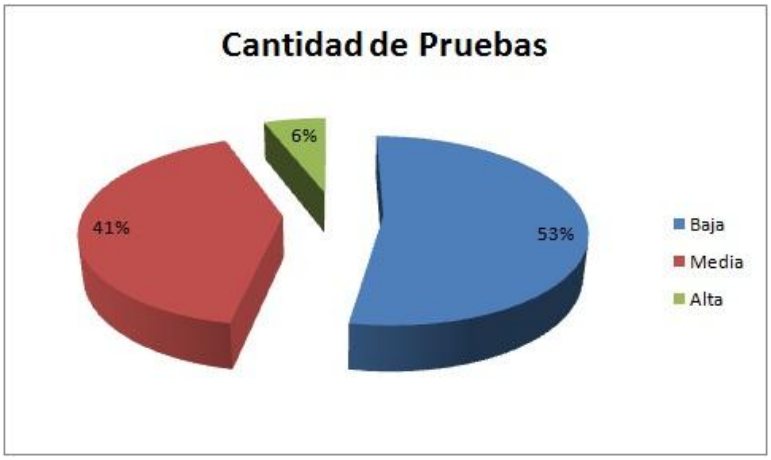

Ilustración 10. Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

Fuente: Elaboración propia.

Ilustración 11. Resultados de la evaluación de la métrica RC para el atributo Reutilización.

Fuente: Elaboración propia.

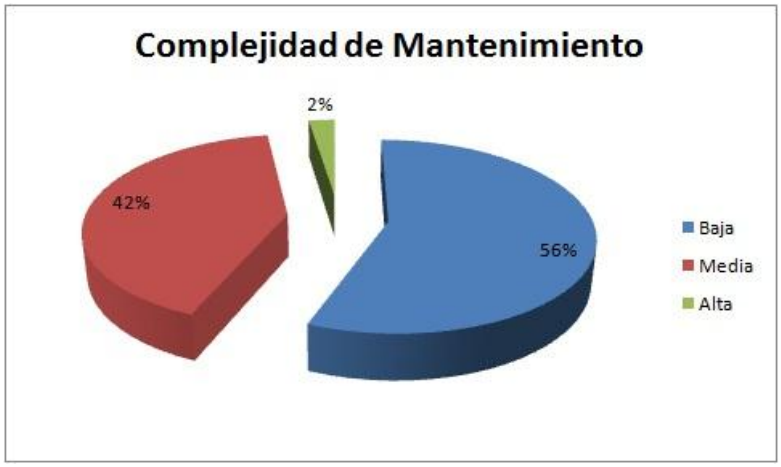

Ilustración 12. Resultados de evaluar la métrica RC para el atributo Complejidad de Mantenimiento. Fuente: Elaboración propia.

Ilustración 13. Resultados de la evaluación de la métrica RC para el atributo Acoplamiento. Fuente: Elaboración propia.

Después de realizarse la evaluación de la métrica RC los resultados obtenidos demostraron que la mayoría de las clases (50%), poseen menos de 3 dependencias entre clases, por lo que se encuentra dentro de los niveles aceptables de calidad. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 40% de las clases el grado de dependencia o acoplamiento es mínimo. La complejidad de mantenimiento y la cantidad de pruebas son bajas a un 56%, lo que representa valores favorables en el diseño realizado. Así mismo, existe un alto grado de reutilización al 77%, comportamiento también favorable para este atributo de calidad.

3.3.3 Evaluación del resultado de las métricas.

Para la evaluación final del diseño se promedian los valores obtenidos y se evalúan en los umbrales (**Tabla 16**) de los parámetros establecidos para la evaluación de la calidad del diseño. A continuación se muestra una gráfica con el resultado final de promediar los atributos medibles según las métricas.

Tabla 16. Umbrales de evaluación.

Categoría	Rango de valores
Malo	<= 0.4
Regular	> 0.4 y < 0.7
Bueno	>= 0.7

Fuente: Elaboración propia.

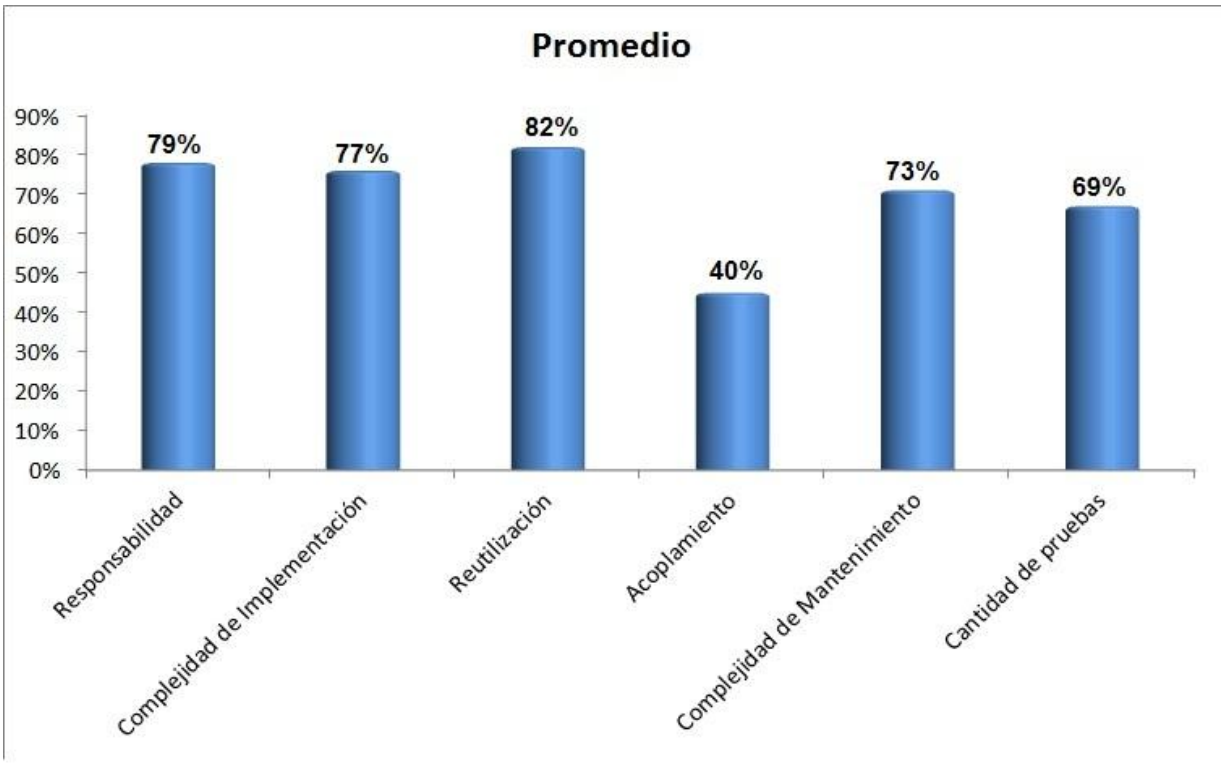

Ilustración 14. Promedio de valores óptimos obtenidos durante la aplicación de las métricas.

Fuente: Elaboración propia.

Como se muestra en la figura anterior, la complejidad de implementación, reutilización, complejidad de mantenimiento, cantidad de pruebas y responsabilidad, se encuentran dentro del rango de buena, lo que demuestra que el diseño posee una baja complejidad de implementación, mantenimiento y responsabilidad. La cantidad de pruebas necesarias aplicables al sistema no es elevada y la reutilización que provee es alta. En sistema posee un bajo acoplamiento lo que le da la posibidad de una alta reutilización. De manera general el promedio de calidad del diseño obtenido por la evaluación de las métricas es de 70%, lo que demuestra que el diseño obtenido es bueno y cumple con los requerimientos de calidad.

3.4 Pruebas de software

La prueba de software es un elemento esencial y crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. La prueba se enfoca sobre la lógica interna del software y las funciones externas. Es un proceso que tiene como objetivo la ejecución de un programa con la intención de descubrir un error. La fase de pruebas es una de las más valiosas del ciclo de vida de un software y en ese sentido, deben evaluarse todos los artefactos generados, lo que incluye especificaciones de requisitos, diagramas de diversos tipos, el código fuente y el resto de productos que forman parte de la aplicación.

“Las pruebas no pueden asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software.” [45].

3.4.1 Pruebas de caja blanca o Estructurales

“La prueba de caja blanca (**Ilustración 15**) del software se basa en un examen cercano al detalle procedimental[7]. Se prueban las rutas lógicas del software y la colaboración entre los componentes, al proporcionar casos de prueba que ejerciten conjuntos específicos de condiciones, bucles o ambos.” [46]. Estas pruebas software también se conocen porque se realizan sobre las funciones internas de un módulo. Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo, las de caja blanca están dirigidas a las funciones internas. Se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas.

Algunas técnicas de prueba de caja blanca son:

[7] Perteneciente o relativo al procedimiento.

Prueba de Condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

Prueba de Flujo de Datos: Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de Bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

Prueba del Camino Básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

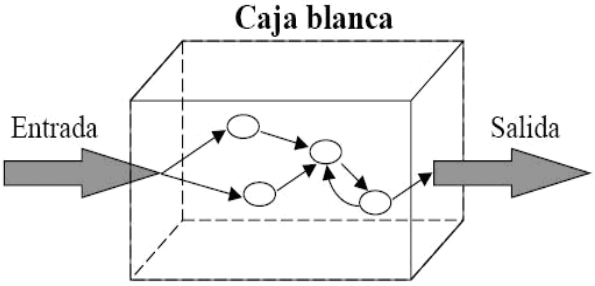

Ilustración 15. Pruebas de caja blanca.

Fuente: [47].

Pasos a seguir en la aplicación de esta técnica:

1ro. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.

2do. Se calcula la complejidad ciclomática del grafo.

3ro. Se determina un conjunto básico de caminos independientes.

4to. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del camino básico. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos

una vez cada sentencia del programa. Para realizar la prueba es necesario realizar primeramente el análisis de complejidad del algoritmo sobre el que se va a realizar la prueba, con el propósito de calcular los valores de la complejidad ciclomática.

```
function cargarElementos($elemento, $name) {
    $domDocument = $this->cargarLepi(); (1)
    $elem = $domDocument->getElementsByTagName($elemento); (1)
    $longitud = $elem->length; (1)
    if ($longitud == 0) { (2)
        return "El elemento no está definido en el LEPI"; (3)
    }
    for ($i = 0; i < $longitud; $i++) { (4)
        if ($elem->item($i)->getAttribute("name") == $name) { (5)
            return $elem->item($i); (6)
        }
    } (7)
    return  "No existe ningún elemento con el nombre especificado"; (8)
} (9)
```

Ilustración 16. Código fuente de la funcionalidad cargar Elementos.

Fuente: Elaboración propia.

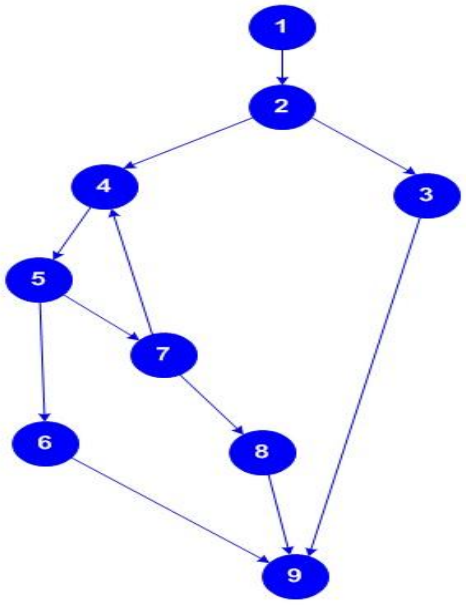

Ilustración 17. Grafo de flujo asociado a la funcionalidad cargar Elementos.

Fuente: Elaboración propia.

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Siendo "**A**" la cantidad total de aristas y "**N**" la cantidad total de nodos.
V (G) = (A - N) + 2
V (G) = (11 - 9) + 2
V (G) = 4

Siendo "**P**" la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).
V (G) = P + 1
V (G) = 3 + 1
V (G) = 4

Siendo "**R**" la cantidad total de regiones, para cada formula "**V (G)**" representa el valor del cálculo.
V (G) = R
V (G) = 4

Mediante los cálculos realizados anteriormente basados en las formas en que se puede obtener el valor de la complejidad ciclomática, se puede observar que todos arrojaron como resultado "**V (G) = 4**", lo que significa que existen cuatro posibles caminos por donde puede circular el flujo. Este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado. A continuación (**Tabla 17**) se muestran los caminos básicos por los que puede recorrer el flujo.

Tabla 17. Caminos básicos del flujo.

Número	Camino básico
1	1, 2, 3, 9
2	1, 2, 4, 5, 6, 9
3	1, 2, 4, 5, 7, 4
4	1, 2, 4, 5, 7, 8, 9

Fuente: Elaboración propia.

Para cada camino se realiza un caso de prueba, y es preciso cumplir con las siguientes exigencias:

- ✓ ***Descripción:*** se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- ✓ ***Condición de ejecución:*** se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- ✓ ***Resultados esperados:*** se expone el resultado que se espera que devuelva el procedimiento.
- ✓ ***Evaluación de los resultados:*** se exhibe la evaluación que dio el resultado final del procedimiento.

3.4.2 Casos de Prueba para pruebas de caja blanca

Caso de prueba para el camino básico # 1.

Descripción: Los datos que van adoptar las variables $elemento, $name, $domDocument cumplirán con los siguientes requisitos: No pueden ser nulos.

Condición de ejecución:

Variables $elemento, $name, $domDocument no pueden ser nulos.

$elemento = "no_existe", **$name** = "Inicial", **$domDocument** = DOMDocumentObject ( ), **$elem** = DOMNodeList Object ( ), **$longitud** = 0.

Resultados esperados: Al ejecutar el procedimiento se mostrará un mensaje informado que el elemento no está definido en el LEPI.

Evaluación de los resultados obtenidos: Los resultados obtenidos de la realización del caso de prueba de caja blanca para el caso de cargar elementos, fueron satisfactorios al mostrar el mensaje de la no definición del elemento buscado.

Caso de prueba para el camino básico # 2.

Descripción: Los datos que van adoptar las variables $elemento, $name, $domDocument cumplirán con los siguientes requisitos: No pueden ser nulos.

Condición de ejecución:

Variables $elemento, $name, $domDocument no pueden ser nulos.

$elemento = "view", **$name** = "Inicial", **$domDocument** = DOMDocumentObject (), **$elem** = <view name="Inicial" method="POST" action="invocarServicio"><slmodel name='sm' xtype="slmodel" singleSelect="false"/><array name="arregloEjercicio" xtype="array" edemt="01"/> …, **$longitud** = 2.

Resultados esperados: Al ejecutar el procedimiento se cargarán los elementos que coinciden con el tipo de elemento pasado por parámetro y con el nombre del elemento.

Evaluación de los resultados obtenidos: Los resultados obtenidos de la realización del caso de prueba de caja blanca para el caso de cargar elementos, fueron satisfactorios al lograr que se cargaran los elementos.

Caso de prueba para el camino básico # 3.
Descripción: Los datos que van adoptar las variables $elemento, $name, $domDocument cumplirán con los siguientes requisitos: No pueden ser nulos.

Condición de ejecución:
Variables $elemento, $name, $domDocument no pueden ser nulos.

$elemento = "view", **$name** = "Upload", **$domDocument** = DOMDocumentObject (), **$elem** = DOMNodeListObject (), **$longitud** = 2.

Resultados esperados: Al ejecutar el procedimiento la lista **$elem** va a tener dos posiciones con elementos **"view"** de diferentes nombres.

Evaluación de los resultados obtenidos: Los resultados obtenidos de la realización del caso de prueba de caja blanca para el caso de cargar elementos, fueron satisfactorios al lograr que se encontrara el elemento "view", con nombre "Upload" en la segunda iteración.

Caso de prueba para el camino básico # 4.
Descripción: Los datos que van adoptar las variables $elemento, $name, $domDocument cumplirán con los siguientes requisitos: No pueden ser nulos.

Condición de ejecución:
Variables $elemento, $name, $domDocument no pueden ser nulos.

$elemento = "view", **$name** = "No_existe", **$domDocument** = DOMDocumentObject (), **$elem** = DOMNodeListObject (), **$longitud** = 2.

Resultados esperados: Al ejecutar el procedimiento se mostrará un mensaje informando que no existe ningún elemento con ese nombre.

Evaluación de los resultados obtenidos: Los resultados obtenidos de la realización del caso de prueba de caja blanca para el caso de cargar elementos, fueron satisfactorios al lograr que se mostrara el mensaje de error.

3.4.3 Pruebas de caja negra o Funcional

Se denominan pruebas funcionales o Functional Testing (**Ilustración 18**), a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios finales, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta, el paso siguiente es el pase a producción. A este tipo de pruebas se les denomina también pruebas de comportamiento o pruebas de caja negra, ya que los testers[8] o analistas de pruebas, no enfocan su atención a cómo se generan las respuestas del sistema, básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas. [48].

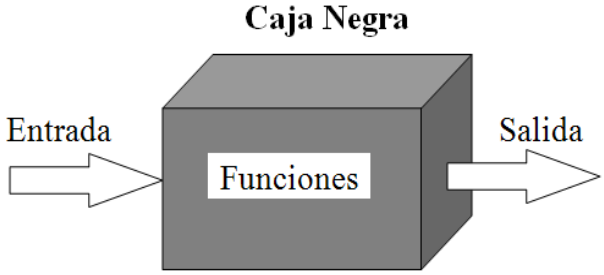

Ilustración 18. Pruebas de caja negra.
Fuente: [47].

En esencia permite encontrar:

- ✓ Funciones incorrectas o ausentes.

[8] Persona que realiza las pruebas.

- ✓ Errores de interfaz.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.
- ✓ Errores en estructuras de datos o en accesos a las bases de datos externas.

Dentro de las pruebas de caja negra se incluyen las siguientes técnicas de pruebas:

Partición equivalente: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Análisis de valores límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Gafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

3.5 Diseño de casos de prueba para pruebas de caja negra

Los casos de prueba están basados en diferentes entradas que puede recibir el sistema con sus correspondientes valores de salida. Se centran en realizar pruebas de software para comprobar su funcionalidad. Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces.

Los casos de prueba demuestran que:

- ✓ Las funciones del software son operativas.
- ✓ Las entradas se aceptan de la forma adecuada produciendo el resultado esperado.
- ✓ La integridad de la información externa (por ejemplo archivos de datos) se mantiene.

Para verificar que la aplicación cumpla con los requisitos establecidos por el cliente, se diseñan los casos de prueba y se realizan las pruebas internas. Seguidamente se especifica el caso de prueba “Exportar

proceso de negocio", el cual permite exportar los comprobantes de operaciones acordes con el criterio de búsqueda seleccionado por el usuario.

Condiciones de ejecución.

- ✓ Se debe autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- ✓ Se debe seleccionar el subsistema de Contabilidad.
- ✓ Se debe seleccionar la opción **Contabilidad/Contabilidad financiera/Interoperabilidad**.
- ✓ Se debe seleccionar la opción **Interoperar**.

Tabla 18. Escenario de prueba del requisito Exportar proceso de negocio.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Exportar proceso de negocio.	Se exportarán los comprobantes deseados por el usuario.	EP 1.1 Exporta los comprobantes acorde con el criterio de búsqueda.	-Dar clic en el botón **Exportar.** -Seleccionar la dirección donde se desea guardar la estructura de intercambio. -Dar clic en el botón **Aceptar.**
		EP 1.2 No existen comprobantes de operaciones acordes con el criterio de búsqueda.	-El sistema emite un mensaje informando que no existen comprobantes de operaciones acordes con el criterio de búsqueda. -Dar clic en el botón **Aceptar.**

Fuente: Elaboración propia.

Tabla 19. Descripción de la variable del caso de prueba para el requisito Exportar proceso de negocio.

No	Nombre del campo	Tipo	Válido	Inválido
1	Periodo, Ejercicio, Estado	Checkbox	Caracteres y números.	N/A

Fuente: Elaboración propia.

Tabla 20. Datos de prueba del caso de prueba para el requisito Exportar proceso de negocio.

Id del escenario	Escenario	Nombre	Respuesta del sistema
EP 1.1	Exportar los comprobantes acorde con el criterio de búsqueda.	**V** (2010, Marzo, Ambos Estados).	El sistema muestra los comprobantes acorde con el criterio de búsqueda.
EP 1.2	Exportar los comprobantes acorde con el criterio de búsqueda.	**I** (2011, Diciembre, Asentados).	El sistema emite un mensaje informando que no existen comprobantes asociados a los parámetros definidos.

Fuente: Elaboración propia.

Resultados de los escenarios para el caso de prueba diseñado en la tabla anterior.

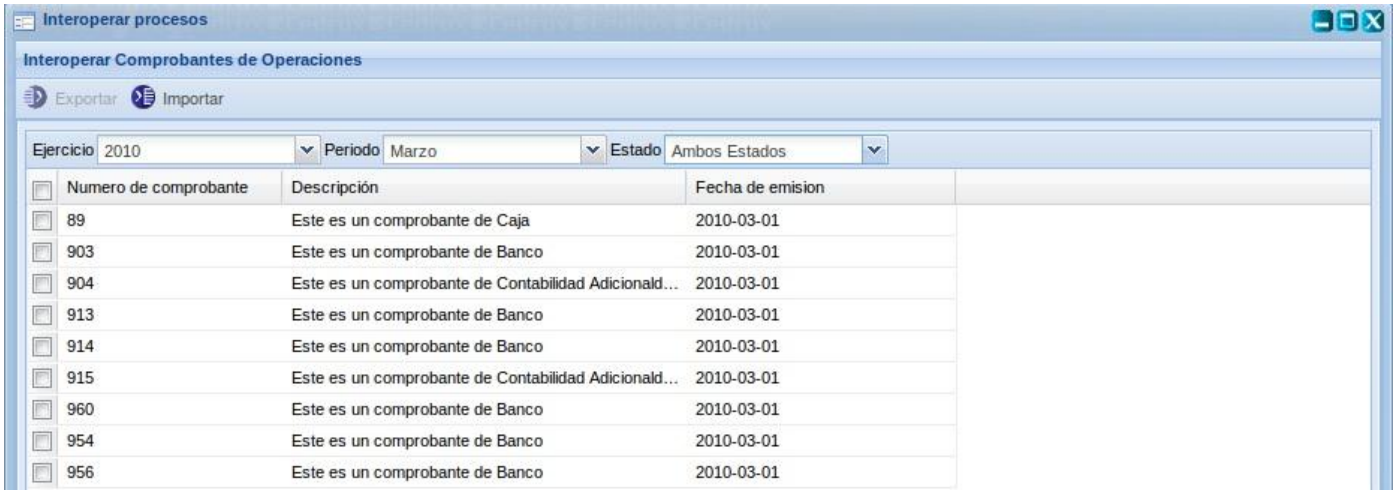

Ilustración 19. Resultados para el escenario valido.

Ilustración 20. Resultados para el escenario invalido.

En la tabla anterior aparecen **V** (válido) y **I** (inválido), indicando los escenarios correctos e incorrectos.

3.6 CONCLUSIONES

- ✓ Se expusieron los estándares de codificación para que los desarrolladores puedan entender en menos tiempo el código y se mostró el diagrama de componentes mostrando una vista arquitectónica de alto nivel del sistema.

- ✓ Se aplicaron las métricas para validar el diseño: Relaciones entre Clases y Tamaño Operacional de la Clase, las cuales de manera general el promedio de calidad del diseño obtenido por la evaluación de las mismas es de 70%, lo que demuestra que el diseño realizado es bueno y cumple con los requerimientos de calidad.

- ✓ Se efectuaron pruebas de software tanto de caja blanca como de caja negra con el objetivo de detectar errores tanto en el código como en las funcionalidades del sistema.

- ✓ El componente desarrollado realiza la generación de vistas dinámicas, permitiendo la disminución del tiempo de desarrollo de las soluciones de interoperabilidad.

CONCLUSIONES GENERALES

Con el desarrollo y culminación del componente dinámico para la interoperabilidad de los procesos de negocio en el Sistema de Gestión Cedrux y producto a la obtención de los resultados generados por esta investigación, se puede arribar a las siguientes conclusiones:

- ✓ Luego de un análisis de los criterios de interoperabilidad estudiados, los autores de esta investigación la definen como: la cualidad que poseen diferentes sistemas para el envío, recepción y manejo de la información intercambiada.

- ✓ Haciendo uso del modelo de desarrollo orientado a componentes del centro CEIGE, se generaron los artefactos necesarios para guiar el proceso de desarrollo.

- ✓ La programación dinámica fue un factor clave en el desarrollo de esta investigación, ya que a partir del LEPI, se posibilita la generación dinámica de las interfaces por cada especificación de subproceso.

- ✓ El componente desarrollado permite la generación de vistas dinámicas, permitiendo así la disminución del tiempo de desarrollo de las soluciones de interoperabilidad.

- ✓ Las pruebas de software permitieron detectar y corregir los errores no identificados durante la implementación, posibilitando cumplir con las especificaciones requeridas y la validación del componente de interoperabilidad desarrollado.

RECOMENDACIONES

- ✓ Realizar un componente que además de permitir la generación dinámicas de las interfaces a partir del lenguaje de ejecución LEPI; sea capaz de interpretar la descripción de la lógica de negocio.
- ✓ Utilizar el componente dinámico de interoperabilidad desarrollado como guía de estudio en futuras investigaciones.

REFERENCIAS BIBLIOGRÁFICAS

1. **Institute of Electrical and Electronics Engineers**. 1990. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York, NY: 1990.
2. **Borges, Alejandro E. 2007.** La interoperabilidad y los estándares abiertos, base del desarrollo de la Sociedad de la Información. *Colegio Oficial de Ingenieros de Telecomunicación.* [En línea] Coit, marzo de 2007. [Citado el: 20 de marzo de 2013.] http://www.coit.es/publicaciones/bit/bit161/36-39.pdf.
3. **Thompson, R.J; Redstone, L**. 1997. *"Business process management-maintaining control in an environment of rapid change", Factory 2000 - The Technology Exploitation Process at Fifth International Conference, (Abril 1997).* IEEEXplore [On line data base]. (Consultada en Febrero de 2013).
4. **Martínez Usero, José Angel**. *La necesidad de interoperabilidad de la información en los servicios de administración electrónica: xml, una posible solución.*, 2004. En VIII edición de las Jornadas sobre Tecnologías de la Información para la Modernización de las Administraciones Públicas (Tecnimap 2004), Murcia (Spain), 28 de Septiembre - 1 de Octubre de 2004. Citado el 14 de enero de 2013. Enlace: http://eprints.rclis.org/7944/.
5. **Nogales Cobas, Pedro M.; Galván Rey, Magdanis.** 2010. *Estrategia de interoperabilidad para la transferencia de datos entre sistemas ERP en Cuba. Cuba. 2010*.
6. **Florencia Chiesa. 2004.** *METODOLOGÍA PARA SELECCIÓN DE SISTEMAS ERP*. Argentina. http://www.ucla.edu.ve/dac/departamentos/informatica-II/metodologia-para-seleccion-de-sistemas-erp.PDF.
7. **Quevedo Campins, Virgen D.; Martínez Chong, Meylin; González Gutierrez, Lester A.; Fernández Fuentes, Alien.** 2010. Cedrux. *Solución para sistemas de control logístico*. Cuba, noviembre-diciembre 2010.
8. **Gradmann**, **S**. (2009). INTEROPERABILIDAD. *Un concepto clave a bibliotecas digitales a gran escala persistentes.* 2009. Disponible en: http://www.digitalpreservationeurope.eu/publications/briefs/es_interoperabilidad.pdf.
9. **Manso Callejo, Miguel Ángel. 2009.** El uso de los metadatos para el desarrollo de un modelo de interoperabilidad para las Infraestructuras de Datos. *Archivo Digital UPM.* [En línea] 2009. [Citado el: 14 de febrero de 2013.] http://oa.upm.es/1870/. 1870.

10. ***NISO*** *(2004). National Information Standards Organization. Understanding Metadata. Documento Web. 2004. Visitado el 28 de febrero de 2013. Enlace: http://www.niso.org/standards/resources/UnderstandingMetadata.pdf*.
11. **ISO 19119.** *International Organization for Standardization*.
12. *Bases para una estrategia iberoamericana de interoperabilidad*. Documento marco iberoamericano de interoperabilidad ratificado en 2010 por la XX Cumbre Iberoamericana de Jefes de Estado y de Gobierno.
13. **Acero, Fernando. 2009.** Interoperabilidad (y IV): Estrategias para las organizaciones. *Bitacoras.* [En línea] kriptopolis, 2009. Citado el 23 de febrero de 2013. http://www.kriptopolis.org/interoperabilidad-4.
14. **Lamarca Lapuente, María Jesús**. (2006). Hipertexto: *El nuevo concepto de documento en la cultura de la imágen.* Madrid, 2006. Visitado el 28 Enero 2013. Dispinible en: http://www.hipertexto.info/documentos/dtds.htm.
15. **LISI Model. 1997.** Levels of Information Systems Interoperability (LISI) Reference Model. *Best Manufacturing Practices.* [En línea] bmpcoe, 1997. [Citado el: 22 de marzo de 2013.] http://www.bmpcoe.org/library/books/lisi%20model.
16. Norma **ISO 9000: 2000.**
17. **Quesada, Víctor Manuel; Vergara, Juan Carlos.** (2007). *ANÁLISIS CUANTITATIVO CON WINQSB.* ISBN: 978-84-690-3681-5. Universidad de Cartagena. 2007. Visitado el 15 de febrero de 2013. Disponible en: http://metodoscuantitativos.50webs.org/winqsb.htm.
18. **Tamara, Igor**. 2000. Tiempo de desarrollo. Agosto de 2000. Visitado el 24 de febrero del 2013. Disponible en: http://igor.tamarapatino.org/escritos/proptlinux/node7.html.
19. **Quesada, Víctor Manuel; Vergara, Juan Carlos.** (2007). *ANÁLISIS CUANTITATIVO CON WINQSB.* ISBN: 978-84-690-3681-5. Universidad de Cartagena. 2007. Visitado el 15 de febrero de 2013. Disponible en: http://metodoscuantitativos.50webs.org/winqsb.htm.
20. **Ramos Arias, Taimé; Torres Salas, Pedro A.** 2010*. Diseño e implementación del módulo Banco del Sistema Integral de Gestión Cedrux.* La Habana, junio de 2010.
21. W3C-DTD. 2011; Available from: http://www.w3.org/TR/html4/sgml/dtd.html.
22. **Rodríguez, Leonardo; Vignaga, Andrés; Zipitría, Felipe.** 2002. *Estudio de Interoperabilidad .NET/J2EE*. Uruguay, 2002. Citado el 29 de enero de 2013. Enlace: http://www.fing.edu.uy/~ruggia/publications/CITA03_InteropServAplic%28Final%29.pdf.

23. **Darío Franco, Rubén; Gómez, Pedro; Navarro Varela, Rosa; Ortiz Bas, Ángel**. 2007. *Diseño de servicios web para dar soporte a la Gestión de Procesos de Negocio.* Valencia. 2007.
24. **Piedrahita Ospina, Alberto Alejandro**. (2012*). Modelo de Interoperabilidad Middleware para la Integración de Agentes Móviles Inteligentes con Redes de Sensores Inalámbricos o WSN (acrónimo inglés para Wireless Sensor Networks).* Medellín, Colombia, 2012.
25. Red de Gobierno Electrónico de América Latina y el Caribe. Interoperabilidad e Intranet Gubernamental. Visitado 11 Febrero 2013. Disponible en: http://www.gobiernofacil.go.cr/e-gob/gobiernodigital/eventos/presentacionesGEALC/Interop_intranet.pdf.
26. **Alves, Alexandre; Arkin, Assaf; Askary, Sid** y otros. 2007. Web Services Business Process Execution Language Version 2.0. 2007. Visitado el 27 de enero de 2013. Disponible en: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf.
27. **Gálvez Rojas, Sergio.** *2008. Tecnologías Emergentes Multiplataforma. Programa de Doctorado en Ingeniería de Sistemas y Computación.* Argentina, 2008. Disponible en: http://www.sicuma.uma.es/sicuma/independientes/argentina08/Bernal-Karanik/index.htm.
28. ECURED. Visitado 8 Febrero 2013. Disponible en: http://www.ecured.cu/index.php/BPEL.
29. **Meadón Mecí, Rogelio.** (2012). *Propuesta de un lenguaje de ejecución para el proceso de interoperabilidad en los sistemas que emplean el marco de trabajo Sauxe.* La Habana, junio de 2012.
30. **O Gomes Baryolo, Mariela Tenrero Cabrera, Nemuris Silega Martinez.** Plantilla Registro de propiedad Intelectual (Sauxe). La Habana: s.n., 2008.
31. **Rodríguez Urrutia, Misael.** 2010. *Manual de diseño proyecto TPC.* Habana: s.n., 2010.
32. **Baryolo, Gómez, Baryolo, Tenrero, Cabrera, Marianela y Silega, Martínez, Nemuris.** 2008. *Plantilla Registro de la Propiedad intelectual (Sauxe).* 2008.
33. **Henst, Christian Van Der**. 2001. ¿Qué es el PHP? [Online] 2001. http://www.maestrosdelweb.com/editorial/phpintro/.
34. **Brown, Eric.** 2003: *The myth of self-describing XML.* Citado el 21 de marzo de 2013. Disponible en: http://workflow.healthbase.info/monographs/XML_myths_Browne.pdf.
35. **Merelo, Juan J.** 2010. *Metodologías de desarrollo del software*. Citado el 14 de febrero de 2013. Enlace en: http://latecladeescape.com/ingenieria-del-software/metodologias-de-desarrollo-del-software.html.

36. **Sommerville, Ian y Sawyer, Pete.** 1997**.** *Requirements Engineering: A good practice guide.* Fayetteville, AR, USA. 1997. ISBN: 9780471974444. Lancaster: s.n. Citado el 1 de abril de 2013. Disponible en: http://books.google.es/books/about/Requirements_engineering.html?hl=es&id=Ag7qotD-sukC.
37. **Larman, Crain**. 1999. *UML y Patrones. Introducción al anális y diseño orientado a objetos.* Mexico, 1999. Visitado el 23 de enero de 2013. Disponible en: http://elblogerperu.blogspot.com/2011/01/uml-y-patrones-introduccion-al-analisis.html.
38. **Ruiz Durán, Javier; González Marcaida, Eyonys.** 2010. *Diseño e implementación de una herramienta para la gestión de servicios web sobre aplicaciones PHP.* La Habana. 2010.
39. **Fernández González, Mairelys y Zorrilla Rivera, Osley.** 2010. *Diseño e implementación del componente Ajuste al Costo del Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades Cedrux.* Ciudad de la Habana: s.n., 2010.
40. **Almazán María, Belén.** (2012). *Herramientas para la Interoperabilidad y Normalización de datos en RI.* La Plata, Octubre de 2012. [Consulta: 2013-02-7]. Disponible en: http://www.oaforum.org/resources/tvtools.php.
41. Expediente de Proyecto de Sauxe.
42. Tablespace Postgresql. Citado el: 20 de marzo de 2013. Enlace: http://www.postgresql.org/docs/8.1/static/manage-ag-tablespaces.html.
43. **Eduardo Rivera Alva. 2011.** Scribd. [En línea] 2011. [Citado el: 9 de mayo de 2013.] http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue.
44. **ECURED.** Métrica de diseño. Citado el 23 de mayo de 213. Disponible en: http://www.ecured.cu/index.php/M%C3%A9trica_de_dise%C3%B1o#Matriz_de_inferencia_de_indicadores_de_calidad.
45. **Pressman, R.** *Ingeniería de Software. Un enfoque práctico.* 1998.
46. **Pressman, Roger S.** 2010. Ingeniería de Software. Un enfoque práctico. Quinta edición. 2010. ISBN: 978-0-07-337597-7.
47. **Blanco Bueno, Carlos.** Ingeniería de Software II. *Construcción y Pruebas de Software*. Citado de 12 de abril de 2013. Disponible en: http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/materiales/tema1-pruebasSistemasSoftware.pdf.

48. **B, Ing. Alexander Oré.** Calidad y software. Citado el 24 de abril de 2013. Disponible en: http://www.calidadysoftware.com/testing/pruebas_funcionales.php.
49. **Bueno-de-la-Fuente, Gema; Rodríguez-Mateos, David**. Herramientas de software para OAIPMH. *La Iniciativa de Archivos Abiertos (OAI): situación y perspectivas en España y Latinoamérica*. Bogotá: Rojas Eberhard, 2007, pp. 247-302. ISBN 978-958-9121-89-4. Disponible en: http://www.rojaseberhard.com.co/rojaseberhard/bibliotecologia/lib_oai.html.
50. **Arias Calleja, Manuel**. *Estándares de codificación.* Disponible en: http://www.cisiad.uned.es/carmen/estilo-codificacion.pdf.
51. **Pantoja Vascón, Ernesto.** El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing. Disponible en: http://revistatelematica.cujae.edu.cuindex.phpteleartícleview15.
52. **ISO** 5963 (1985).
53. **Osoria, J.R.M.R.-E.R**., *Guía metodológica de interoperabilidad para sistemas integrales de gestión en Cuba*, in *UCI*. 2010, UCI: Ciudad de la Habana.