Universidad de las Ciencias Informáticas

Facultad 15

“Diseño e Implementación de la solución de WorkFlow para el Sistema CEDRUX.”

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

Autores:

Yoandry Pardo Escobar

Javier González Rodríguez

Tutor:

Ing. Alain Fernández Deronceré

Co-Tutor:

Ing. Osmar Leyet Fernández

Ciudad de la Habana

“Año 50 del triunfo de la Revolución”

Junio del 2010

Declaración de Autoría.

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 15 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de _____del año_____.

_____ _____

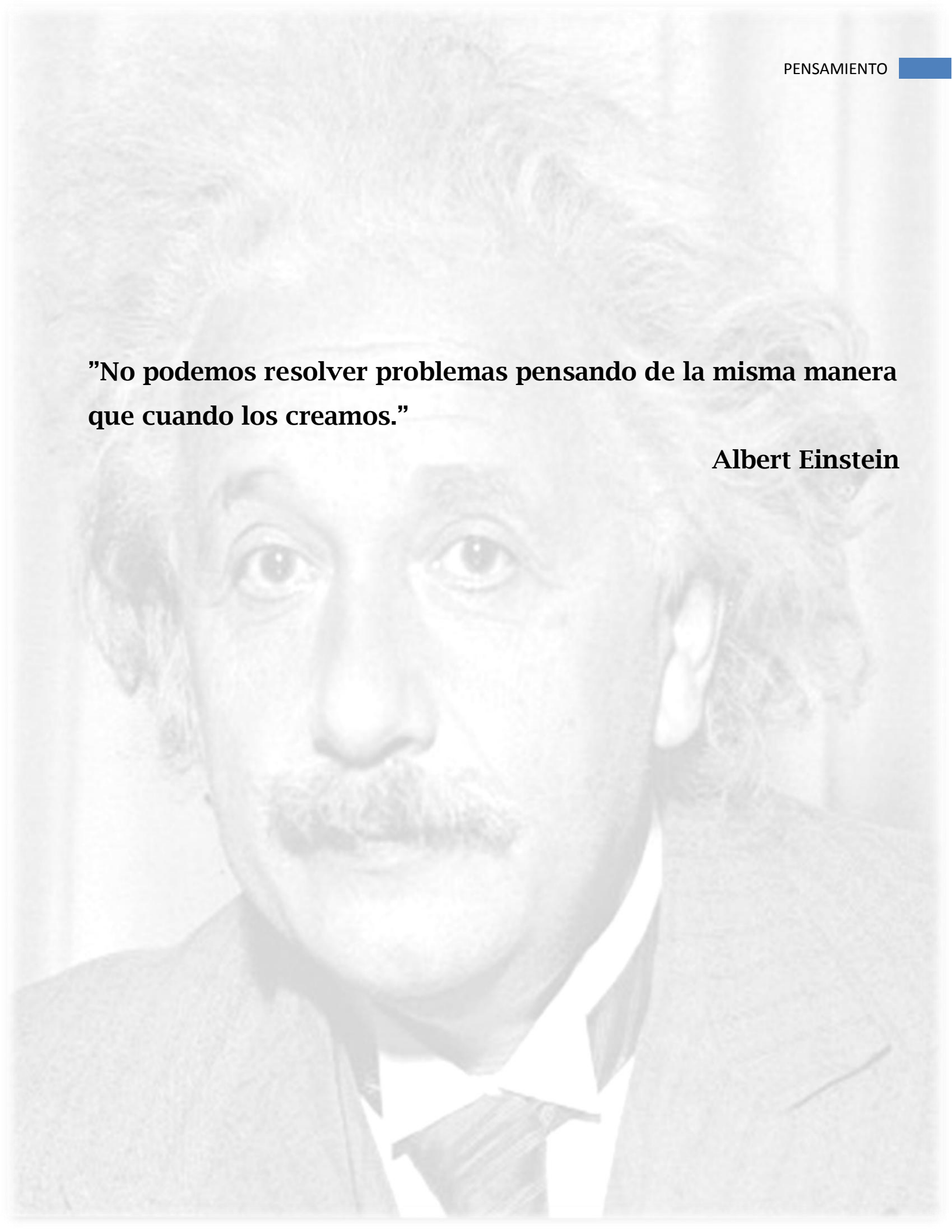
Yoandry Pardo Escobar **Javier González Rodríguez**

Firma del autor

_____

Alain Fernández Deronceré

Firma del Tutor



”No podemos resolver problemas pensando de la misma manera que cuando los creamos.”

Albert Einstein

Yoandry Pardo Escobar

A mis 2 papás y a mi mamá por su apoyo incondicional.

A mi familia en general y amigos.

A la revolución que ha hecho posible la realización de un sueño.

A todas los que me han ayudado durante éstos 5 años.

A mi compañero Javier que gracias a él fue posible el desarrollo y culminación de éste trabajo.

A nuestros tutores que nos han guiado en la realización de éste trabajo.

Dedicado a:

Mis padres, mi pareja y a todos los que de una forma u otra han puesto su granito de arena en mi formación…

Javier González Rodríguez

A mis padres por su apoyo incondicional y por ser los mejores padres del mundo.

A mi familia en general y amigos.

A mi abuelita por todo su amor.

A todas las personas que han puesto su granito de arena en mi formación.

A mi compañero Yoandry que gracias a él fue posible el desarrollo y culminación de éste trabajo.

Dedicado a:

Mis padres, a todos los que de una forma u otra han sido motivo de inspiración durante éstos 5 años…

Al piquete de las canchas.

Los socios del 110 los cuales me han apoyado en todo.

A mi pareja Daynelis la cual me ha brindado todo su cariño y amistad durante estos 5 años.

A mi hermanito que lo quiero con la vida, que siga estudiando para que en la vida sea alguien.

Resumen

Gestionar y organizar los procesos que se generan en el proyecto CEDRUX, dándoles una secuencia lógica dentro de los flujos de trabajo es una tarea de importancia clave en el desarrollo de cada una de las etapas del software. El actual diseño hace que sea engorrosa y en ocasiones muy trabajosa tanto la comunicación como la validación de rutinas entre los diferentes subsistemas, de manera que se hace necesario una nueva solución; la propuesta fue un Sistema de Gestión de Flujos de Trabajo (SGFT), para ello se diseñó e implementó un componente[1] para el Sistema CEDRUX que realiza un manejo centralizado de los procesos comunes a los cambios de estado los documentos, de manera que mejoró la comunicación y adaptación entre los subsistemas y entre sus procesos internos.

[1] Un componente es una parte no trivial, casi independiente, y reemplazable de un subsistema que llena claramente una funcionalidad dentro de un contexto en una arquitectura bien definida.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN.

En estos momentos de constantes cambios y ante una creciente competitividad en el ámbito empresarial, la capacidad de adaptarse al entorno se está convirtiendo en un factor determinante del éxito de las organizaciones. Las empresas han de ser más ágiles y flexibles con capacidad de innovar para poder reaccionar rápidamente a los cambios en el mercado y la competencia, para lograrlo es necesaria entre otros factores la adopción de sistemas informáticos que realicen tareas de análisis y gestión de información de manera automática.

Los Sistemas Empresariales de Planificación de Recursos (Enterprise Resource Planning ERP en inglés) proporcionan una plataforma tecnológica para que las organizaciones puedan integrar y coordinar sus principales procesos internos de negocios. Existen una gran variedad de paquetes de software que facilitan los procesos empresariales: VERSAT-Sarasola, RODAS XXI, SISCONT5, desarrollados en Cuba y CONDOR, SAP, entre otros de producción extranjera; sin embargo no se ha logrado obtener un estándar de todas las funcionalidades que beneficie a las entidades cubanas debido que no se ajustan totalmente a las particularidades de la economía del país tales como la dualidad monetaria y la multi-moneda unido a ello, se encuentra el hecho de que determinadas aplicaciones estén protegidas o que fueran desarrolladas sobre tecnologías y herramientas privadas trae consigo que las posibilidades de usarlo, modificarlo o redistribuirlo sean limitadas.

La Universidad de las Ciencias Informáticas (UCI) por solicitud de los organismos centrales del estado y en conjunto con otras entidades se encuentran actualmente desarrollando un ERP denominado CEDRUX con el objetivo de responder a las necesidades de las entidades cubanas; dicho sistema no presenta actualmente un diseño en el Flujo de los Procesos (WorkFlow) basado en los estándares sobre los que se desarrollan los software de este tipo, dicho factor sumado a que el modelo de negocio del sistema se ve sometido a variaciones en los procesos, producto a los ajustes económicos que se llevan a cabo en el país y a la incorporación de las nuevas funcionalidades previstas para la segunda fase produce situaciones tales como:

- ✓ Redundancia de código.
- ✓ Los implementadores de un subsistema se ven obligados a conocer de la lógica de otros subsistemas.
- ✓ Integrar nuevos subsistemas implica muchos cambios en los que ya están funcionales.
- ✓ Se hace engorroso validar que determinados procesos se hayan ejecutado.

Dichos inconvenientes harían que el software resultante presente dificultades de rendimiento, que los plazos de entrega de cada una de las iteraciones sean más largos o no podrían ser cumplidos debido a demoras en la re-implementación y adicionalmente, la complejidad y cantidad de código harían más difícil el mantenimiento y el soporte del producto final. Por lo que se hace necesario resolver el siguiente **problema**: La implementación del flujo de los procesos descritos en el negocio afecta la escalabilidad del Sistema CEDRUX.

Para dar respuesta al problema, se designa como **objeto de estudio**: los Sistemas de Gestión de Flujos de Trabajo (SGFT) y como **campo de acción**: la implementación del flujo de los procesos en el Sistema CEDRUX. Con el propósito de ofrecer solución al problema se formuló como **objetivo general**: realizar diseño e implementación de una solución para el flujo de los procesos del negocio que mejoren la escalabilidad del Sistema CEDRUX y como **objetivos específicos:**

1. Identificar objetos[2], procesos, flujos, mensajes[3] y eventos inmersos en la solución actual.
2. Realizar estudio del arte sobre soluciones que gestionen los artefactos anteriores.
3. Diseñar la propuesta de solución.
4. Implementar la propuesta de solución.
5. Validar la solución propuesta.

Para cumplir éstos objetivos se plantean las siguientes **tareas:**

1. Estudio del marco de trabajo desarrollado por la Línea de Arquitectura del ERP.
2. Entrevista con los encargados del diseño de la solución actual del Sistema CEDRUX
3. Elaboración de los artefactos de la solución propuesta.
4. Implementación de la solución apoyándose en los artefactos generados.
5. Realización de pruebas sobre la factibilidad de la solución del componente Workflow.

En el presente trabajo se propone como idea a defender que: con el diseño e implementación de un WorkFlow el Sistema CEDRUX mejorará su escalabilidad.

Con el presente trabajo se espera lograr el diseño e implementación de un componente para el Sistema CEDRUX de manera que mejore la comunicación y adaptación entre los subsistemas.

[2] Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad. Corresponden a los objetos reales del mundo que nos rodea, o a objetos internos del sistema.
[3] Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

El trabajo consta de introducción, tres capítulos, conclusiones generales, referencias bibliográficas, bibliografía y anexos. En la introducción se plantea el diseño teórico y metodológico del trabajo. En el primer capítulo se realiza un estudio acerca del estado del arte y los fundamentos teóricos del tema de investigación. En el capítulo dos se realiza el diseño e implementación de la solución enfocado a la construcción de los artefactos, se plantean los principios fundamentales. Se definen e integran las clases y el modelo de datos. En el capítulo tres se valida el diseño de la solución aplicando métricas y se le aplican los métodos de prueba de caja blanca y caja negra sobre el componente.

1. CAPÍTULO I FUNDAMENTACIÓN TEÓRICA .

Introducción

Durante los años 70, 80 y hasta mediados de los 90 el principal uso de las computadoras en las organizaciones fue la automatización de las actividades individuales dentro de las mismas. La situación actual es distinta, pues en los últimos años resulta evidente que existe la necesidad en organizaciones de poder adaptarse rápidamente a los cambios en los procesos internos que experimentan.

El interés de las organizaciones ya no está limitado únicamente al desarrollo de diferentes software que automaticen determinadas actividades individuales, sino también tienen como objetivo final la automatización de todo el proceso de negocio, ya que de ello depende en gran parte su competitividad.

Surgen, por lo tanto, nuevas necesidades de capturar, modelar, ejecutar y monitorear los procesos de negocio, vistos como un conjunto de procedimientos o actividades enlazadas, cuya realización permite alcanzar un cierto objetivo o meta en el contexto de una organización, para solucionar dichas necesidades los Workflow y la tecnología asociada a este ofrece un marco adecuado para solucionar el problema, puesto que cubre al menos parcialmente dichas necesidades.

Proceso: La norma internacional ISO-9001 define un proceso como “una actividad que utiliza recursos y que se gestiona con el fin de permitir que los elementos de entrada se transformen en resultados”. (1)

Proceso de Negocio: Un proceso de negocio se puede ver como un conjunto estructurado de tareas llevadas a cabo lógicamente para lograr un resultado de negocio definido. Los procesos de negocio de una organización son parte de su cultura. 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 funcionamiento. Se registran y difunden en manuales de procedimientos, diagramas de flujo y hasta en forma verbal, son la base operativa de una empresa y el éxito de la misma depende fuertemente de la eficiencia con que sean gestionados.

Como respuesta al problema del modelado de procesos de negocio dentro de una organización, en la década del 90 surge la tecnología de Workflow. Esta tecnología permitió representar total o parcialmente los procesos de negocio en un formato entendible por una máquina. (2)

La Workflow Management Coalition (WfMC), una de las principales organizaciones en el mundo que trabaja en temas de Workflow, cuyo objetivo es unificar esfuerzos para proponer estándares en el campo de los SGFT, lo define de la siguiente manera: “un conjunto de uno o más procedimientos o actividades

directamente ligadas, que colectivamente realizan un objetivo del negocio, normalmente dentro del contexto de una estructura organizacional que define roles funcionales y relaciones entre los mismos". (3)

En un Workflow, la información, tareas y documentos pasan de un participante a otro, para que se realicen una serie de acciones de acuerdo con un conjunto de reglas de negocio. Los sistemas que dan soporte a la definición del flujo de trabajo y a su posterior ejecución, se denominan SGFT.

1.1. Sistemas de Gestión de Workflow

Un SGFT es "un sistema que define, crea y gestiona la ejecución de Workflows mediante el uso de software, siendo capaz de interpretar la definición del proceso, interactuar con los participantes y siempre que se requiera, invocar el uso de herramientas y aplicaciones". (4)

De acuerdo con la definición anterior, en un SGFT existen dos actividades claramente diferenciadas, aunque estrechamente relacionadas, por una parte está la definición del flujo de trabajo que implementa al proceso de negocio, lo que se llama modelado del flujo de trabajo y por otra parte está la ejecución de dicho modelo, también conocido en la literatura como enactment. El Anexo 1 muestra los conceptos básicos y la terminología asociada tanto para la fase de modelado como para la fase de ejecución.

En los SGFT convergen diferentes áreas entre las cuales se pueden mencionar como más significativas las siguientes:

Proceso: Los SGFT intentan automatizar la coordinación de todos los elementos que componen un proceso, indicando qué hay que hacer, en qué orden, quién debe hacerlo y qué recurso debe utilizar.

Reingeniería: Forma de definir, analizar y automatizar procesos empresariales.

Isla de Información: Se llama así a las redes o subredes de ordenadores existentes en una organización, que quedan aisladas unas de las otras sin ofrecer una visión global. Los SGFT sirven para conectar estas islas entre sí.

Sistemas basados en componentes: En los SGFT también concurren ideas de los sistemas basados en componentes, teniendo en cuenta que el desarrollo de aplicaciones se basa en la combinación de componentes ya existentes.

Los SGFT abarcan otras áreas como el procesamiento de imágenes, gestión documental, correo electrónico entre otras. Una aplicación WorkFlow automatiza la secuencia de acciones, actividades o

tareas utilizadas para la ejecución del proceso, incluyendo el seguimiento del estado de cada una de sus etapas y la aportación de las herramientas necesarias para gestionarlo.

Los objetivos de un Sistema de Gestión de Flujos de Trabajo son:

- ✓ Reflejar, mecanizar y automatizar los métodos y organización en el sistema de información.
- ✓ Establecer los mecanismos de control y seguimiento de los procedimientos organizativos.
- ✓ Independizar el método y flujo de trabajo de las personas que lo ejecutan.
- ✓ Facilitar la movilidad del personal.
- ✓ Soportar procesos de reingeniería de negocio.
- ✓ Agilizar el proceso de intercambio de información y agilizar la toma de decisiones de una organización, empresa o institución.

Los SGFT pueden ser implementados de diferentes formas como resultado de la utilización de diferentes tecnologías y plataformas. Pueden ir desde un pequeño grupo de trabajo a una gran organización. No obstante, todos los SGFT exhiben ciertas características comunes. En el nivel más alto, todos los SGFT pueden ser caracterizados por proveer tres áreas de funcionalidad:

- ✓ Funciones de tiempo de Construcción (Build-time functions), dedicadas a la definición y modelado de un proceso junto con todas sus actividades concernientes.
- ✓ Funciones de control en tiempo de ejecución (Run-time control functions), las cuales controlan el proceso en el ambiente de ejecución, llevando a cabo cada tarea (o actividad) definida como parte del proceso.
- ✓ Funciones de interacción en tiempo de ejecución (Run-time interaction), las cuales interactúan con los usuarios o aplicaciones externas para que los participantes del proceso puedan llevar a cabo sus tareas.

1.2. Estándares de SGFT

Los principales esfuerzos en el desarrollo y promoción de estándares en el campo de los SGFT han sido realizados por la WfMC.

La WFMC ha definido una serie de estándares como es el Modelo de Referencia que describe la arquitectura básica de un SGFT y que junto con el de Terminología y Glosario constituyen el material básico, tomando como base estos se han definidos otras especificaciones mucho más concretas. (Anexo 2)

1.2.1. Modelo de Referencia de los SGFT.

El Modelo de Referencia propuesto por la WfMC intenta reunir las características comunes de cualquier producto para la gestión de flujos de trabajo, de manera que sea posible la interoperabilidad entre ellos, a través de estándares comunes para cada una de las funciones que se puedan realizar.

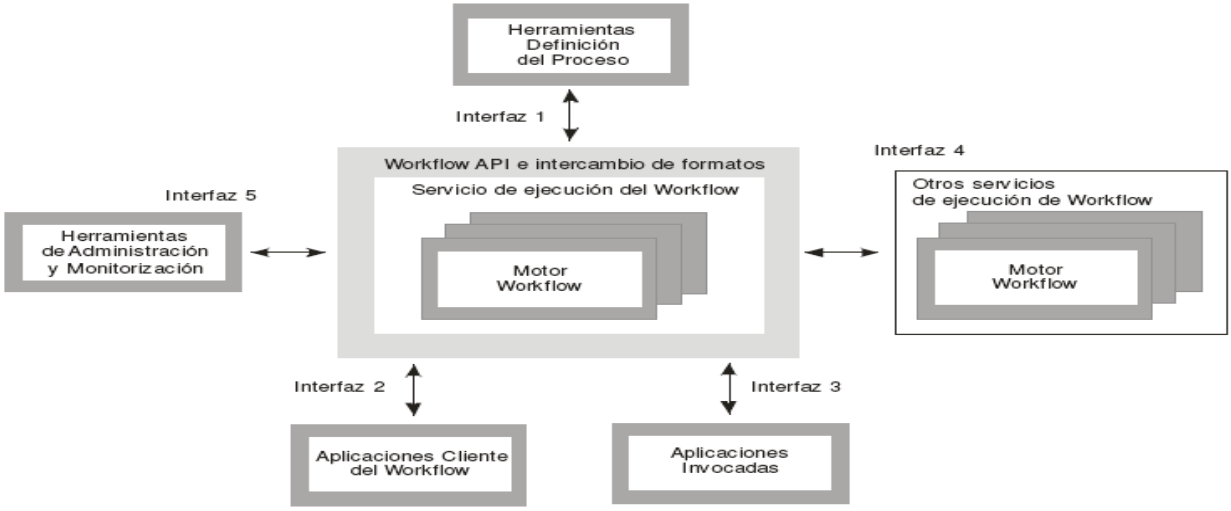


Figura 1 Modelo de Referencia de un SGFT (fuente WfMC)

Servicio de ejecución del flujo de trabajo: Se encarga de crear, gestionar y ejecutar cada una de las instancias del modelo de flujo de trabajo, en este componente se encuentra el Motor de Workflow.

Herramientas de definición del proceso: Permiten modelar, describir y documentar un determinado flujo de trabajo, proceso de negocio o proceso en general.

Interfaz 1: Permite la comunicación entre Herramientas de definición del proceso y el servicio de ejecución del flujo de trabajo.

Aplicaciones clientes del flujo de trabajo: Representa las entidades software utilizadas por el usuario final en aquellas actividades que requieren participación humana para su realización.

Interfaz 2: Si **Aplicaciones clientes del flujo de trabajo** se separa, es necesaria esta interfaz para que defina y maneje claramente el concepto de lista de trabajos o worklist, como una cola de trabajo asignado a un usuario o a un grupo de usuarios por el propio motor de ejecución del flujo de trabajo.

Aplicaciones invocadas: Representa software o aplicaciones ya existentes que un SGFT puede utilizar para la realización de ciertas actividades, sin importar plataforma o lugar en la red.

Interfaz 3: Permite la comunicación entre **aplicaciones invocadas** y el **servicio de ejecución del flujo de trabajo**, no sólo a nivel de invocación del mismo, sino de transformación de datos en formatos entendibles por ambos componentes.

Herramientas de administración y monitorización: Permite que distintos servicios de ejecución de flujo de trabajo compartan las mismas funciones de administración y monitorización del sistema.

1.2.2. Terminología y Glosario.

El objetivo de este estándar es definir los términos utilizados en las especificaciones de la WfMC, con la finalidad de establecer una consistencia en el uso de los mismos, tanto en la industria como en el mundo de los SGFT en general. En el estándar se identifica la terminología usada para describir los conceptos y la estructura general de un SGFT, con sus principales componentes e interfaces. (4)

Interoperabilidad entre SGFT

Define la funcionalidad necesaria para permitir la interoperabilidad entre dos o más motores de distintos SGFT, en este inicialmente existen diferentes niveles de interoperabilidad y para cada uno de ellos un posible modelo que permita dicha integración.

Este estándar se enriquece con estándares de WFMC como son:

- ✓ **Interoperabilidad utilizando XML 1.0:** Su objetivo es utilizar XML como lenguaje de intercambio de mensajes entre SGFT. (5)
- ✓ **El estándar de interoperabilidad "Internet e-mail MIME Binding":** Permite la interoperabilidad entre SGFT utilizando el correo electrónico en internet como medio de comunicación y la codificación MIME como mecanismo de transporte de información. (6)

Workflow Client API Specifications (WAPI)

Este estándar especifica la interfaz de comunicación del motor de flujo de trabajo con el resto de componentes que lo forman. Esta comunicación se realiza a través de Application Programming Interfaces (APIs), definiéndose un conjunto de servicios para la programación de aplicaciones de flujo de trabajo, denominada WAPI. (7)

Análisis de Datos

Especifica la información que debe ser capturada y almacenada respecto a los diferentes eventos ocurrido durante la ejecución del flujo de trabajo, a dicha información se le denomina CWAD y se utilizará tanto para realizar análisis como para realizar pruebas de ejecución. (8)

Definición de Procesos & Workflow Process Definition Language (WPDL)

Este estándar forma parte de la estandarización de la interfaz 1 del modelo de referencia definido por la WfMC. Se incluye un metamodelo para describir la definición de procesos y una propuesta de lenguaje textual. (9)

1.2.3. Clasificación de los SGFT.

Esta clasificación está dada por el criterio a considerar:

I. **Según la complejidad del grafo que representa al flujo de trabajo:** (10)

- ✓ **Ligeramente estructurado:** En este caso, el grafo que representa la coordinación entre las actividades es prácticamente lineal y estas se van ejecutando una a continuación de otra.
- ✓ ***Altamente estructurado:*** El grafo que representa la coordinación entre las actividades ya no es lineal, sino que se representan ejecuciones en paralelo de actividades, sincronización de actividades.

II. **Según el grado de participación humana en el flujo de trabajo:** (10)

- ✓ ***Orientados a personas:*** Cuando la participación humana en la ejecución del SGFT es importante. Estas personas serán agentes que inician o realizan las actividades. Por su naturaleza, la mayor parte de las actividades que conforman el flujo de trabajo son manuales.
- ✓ **Orientados a sistemas:** La participación humana suele ser menor y el proceso está altamente automatizado, por lo que la mayor parte de las actividades del flujo de trabajo son actividades automáticas.

III. **De acuerdo con la tecnología en la que se basa el SGFT:**

- ✓ **Centrados en el correo electrónico:** En estos SGFT se utiliza el correo electrónico como medio de comunicación.
- ✓ ***Centrados en el documento:*** La principal característica es que los documentos circulan e interaccionan con aplicaciones externas. Predominan los aspectos de gestión de documentos.
- ✓ **Centrados en el proceso:** En este caso, lo importante es el propio proceso. Los mecanismos de comunicación entre las actividades y los datos se implementan por encima de la base de datos, proporcionándose interfaces de interacción.

IV. Según la complejidad de las actividades involucradas y la estructura de las mismas, por una parte y por otra, en base a las similitudes de los procesos de negocio involucrados y su valor o importancia en la propia organización:

- ✓ ***administrativo:*** que corresponde a procesos de negocios conocidos de la empresa y que está sujeto a procedimientos preestablecidos. En este caso, la dirección del Workflow es más o menos fija. La principal característica es que las actividades inmersas en el proceso son repetitivas y de baja complejidad.
- ✓ **ad hoc:** que se basa en un modelo de trabajo de grupo en el cual los protagonistas participan en la decisión de hacia dónde dirigir el Workflow. Aquí la dirección del Workflow es dinámica. Las actividades inmersas en el proceso suelen ser únicas, en el sentido de que sólo se ejecutan una sola vez y su complejidad está entre baja y media.
- ✓ **Producción:** Modelan e implementan los procesos del negocio críticos de la organización. Los SGFT de este tipo son sistemas complejos, en los que suelen participar como agentes gran variedad de organizaciones. Las actividades involucradas son repetitivas y de alta complejidad.
- ✓ **Colaboración:** Participan gran cantidad de personas, de forma tal que la mayor parte de la coordinación la realiza el hombre, son sistemas por lo general muy dinámicos.

1.2.4. Especificación de un Workflow.

La aparición masiva de numerosos SGFT puso en evidencia aún más la falta de un modelo formal universal para la especificación de un proceso de negocio. Esto dificulta la comparación de las diferentes tecnologías y herramientas. Afortunadamente la especificación de un Workflow puede ser explicada en sentido general desde diferentes perspectivas.

Perspectiva de Control de Flujo: describe actividades y su orden de ejecución mediante diferentes constructores que permiten controlar el flujo de ejecución (joins, splits, secuencias paralelismo, etc.). Estas actividades se pueden ver como unidades atómicas de trabajo.

Perspectiva de Datos: describe los datos (documentos, objetos, etc.) que fluyen entre las diferentes actividades. Estos datos también pueden ser variables locales que definen pre y pos condiciones en la ejecución de tareas.

Perspectiva de Recursos: muestra una visión más orientada al negocio, describiendo el proceso en función de las responsabilidades que tienen las diferentes personas o dispositivos en la ejecución de una determinada tarea.

Perspectiva Operacional: muestra las acciones elementales que se realizan dentro de las actividades, tales como invocar un determinado servicio de una aplicación con determinados datos.

Si bien todas las perspectivas presentan una visión diferente del mismo sistema, la perspectiva de control de flujo provee un mejor panorama para la especificación de un Workflow describiendo más ampliamente el proceso en sí mismo. La perspectiva de datos sólo se apoya en la mencionada anteriormente, mientras que la operacional es más bien complementaria. Es por esto que los estudios se enfocan en la perspectiva de control de flujo.

1.2.5. Elementos para Modelar un Workflow.

Cuando se modela un sistema de Workflow generalmente se identifican y utilizan definiciones de los distintos elementos que se pueden encontrar dentro de dicho sistema. A continuación se listan estos elementos y se ofrece una breve descripción:

Tareas

Cada tarea es un conjunto de acciones o actividades manejadas como una sola unidad. Generalmente son desempeñadas por una única persona dentro de los roles que pueden realizar dicha tarea. Las tareas surgen del análisis del flujo del trabajo, donde se define por quiénes deben ser ejecutadas.

Personas (Usuarios)

Las tareas son realizadas en un orden definido por determinadas personas (o agentes automatizados tomando el rol de las personas) basadas en las condiciones o reglas del negocio.

Roles

Cada rol define las distintas competencias potenciales que existen en el sistema. Se definen independientemente de las personas físicas a las cuales se les van a asignar dichos roles. Una persona puede tener más de un rol.

Rutas

Una ruta define la secuencia de pasos a seguir por los documentos (o información) dentro de un sistema de Workflow. La capacidad de pre-establecer los caminos a seguir por las tareas a usuarios remotos u ocasionales es vital en una aplicación de Workflow. Para asegurar el éxito del flujo de información y decisiones, todos los miembros del equipo deben ser capaces de tomar parte en este proceso.

Existen varios tipos de rutas:

1. **Rutas Fijas:** en este caso los documentos siguen siempre el mismo camino. Se define de antemano cuál es la próxima etapa a seguir.
2. **Rutas Condicionales:** el camino a seguir depende de la evaluación de condiciones. Estas decisiones se toman en el mismo momento que se pasa por el punto donde hay que evaluar las condiciones.
3. **Rutas Ad-Hoc:** en este caso el usuario elige explícitamente cuál es la siguiente etapa a seguir.

Reglas de Transición:

Son reglas lógicas que determinan la navegación del documento dentro del sistema. Expresan qué acción se va a tomar dependiendo del valor de expresiones lógicas. La definición de las reglas puede ser muy complicada, con múltiples opciones, variaciones y excepciones.

Datos:

Los datos son los documentos, archivos, imágenes, registros de la Base de Datos y otros utilizados como información para llevar a cabo el trabajo. Entre los datos manejados por el Workflow se encuentran:

1. Datos de Control: son los datos internos manejados por la lógica del sistema de Workflow.
2. Datos Relevantes: son aquellos datos utilizados para determinar el ruteo de las distintas tareas del sistema.
3. Datos de la Aplicación: Estos datos son específicos de la aplicación, no son accedidos por la lógica del Workflow.

Eventos: un evento es una interrupción que contiene información, el mismo tiene un origen y uno o más destinatarios. La información contenida en el mensaje que se produjo por el evento puede ser implícita o dada por el usuario. Los eventos pueden ser disparados voluntariamente por el usuario o automáticamente.

Plazos (Deadlines): puede verse a los plazos como los tiempos que se le asignan a ciertos elementos o tareas.

Procesos: conjunto de actividades o eventos interrelacionados que transforman elementos de entrada en elementos de salida.

- ✓ **Políticas:** las políticas son una manera formal de expresar sentencias de cómo serán manejados ciertos procesos.

1.2.6. Patrones de Workflow

La WfMC ha identificado un conjunto de patrones básicos los cuales están de forma recurrente dentro de la implementación de sistemas de Workflow, estos patrones capturan los aspectos elementales para el control de procesos.

Estos patrones se han convertido en un estándar, no sólo como formas normalizadas de encarar la solución de las situaciones que estos patrones atacan, sino también como una forma de evaluar la capacidad de las herramientas de Workflow, de acuerdo a su capacidad para implementar en forma natural dichos patrones. (11) (12)

Patrones de control básico.

1. **Secuencia:** una actividad en un proceso de Workflow es habilitada después de ser completada otra actividad en el mismo proceso.
2. **Separación en paralelo (también llamado AND/split):** dos o más actividades de un proceso se ejecutan en paralelo. En un punto del proceso de Workflow, el hilo de control se divide en múltiples hilos de control, habilitando la ejecución de las tareas en paralelo y sin restricciones de orden entre ellas.
3. **Sincronización:** una actividad es iniciada cuando dos o más hilos completan la ejecución de sus actividades. En un punto del proceso, dos o más hilos de control convergen en un solo hilo.
4. **Opción exclusiva (XOR-Split):** en un punto del proceso, una o más de sus ramas son seleccionadas sobre la base de los datos de control del proceso.
5. **Fusión simple:** se mezclan varios hilos de control en uno solo, en determinado punto del proceso.

Patrones de bifurcación y sincronismo avanzados.

1. **Múltiple Opción (Or-Split):** en un punto del proceso, basado en los datos de control del proceso, uno o más caminos o ramas son seleccionados.
2. **Fusión Sincronizada:** se trata de un punto en el proceso de Workflow donde múltiples caminos convergen en un único hilo de control. Si más de un camino ha sido tomado, entonces debe producirse una sincronización entre los hilos de control, si por el contrario, ha sido tomado sólo uno, los demás hilos deben converger.
3. **Fusión Múltiple:** en una actividad convergen distintas ramas o caminos en los que se abrió el hilo de control en una bifurcación anterior. La actividad donde convergen los hilos será activada una vez por cada hilo de la rama que se activó en la separación.

4. **Discriminador:** es un punto del proceso que espera por una rama o hilo de control antes de activar la siguiente actividad. Los siguientes hilos de control que lleguen a ella luego de activada la tarea siguiente son ignorados, hasta que al llegar el último la tarea finaliza.
5. **Unión, N-de-los-M:** unión N-de-los-M, es un punto en el proceso de workflow donde M caminos de ejecución paralelos convergen en uno. La siguiente actividad es activada sólo cuando N caminos se han completado. La compleción de los siguientes M-N caminos es ignorada.

Patrones que involucran tareas con múltiples instancias.

1. **Múltiples instancias sin sincronización:** una actividad es instanciada múltiples veces, desconociéndose en tiempo de diseño el número de instancias a ser habilitadas. Cuando todas las instancias creadas finalizan, se ejecuta la siguiente actividad.
2. **Múltiples instancias con conocimiento a priori en tiempo de diseño:** se soporta que una actividad sea habilitada o invocada muchas veces, con la cantidad de veces conocida en tiempo de diseño.
3. **Múltiples instancias con conocimiento a priori en tiempo de ejecución:** una actividad es instanciada más de una vez, dependiendo de la información disponible en tiempo de diseño, antes de invocar la actividad. Esta información puede provenir de los datos de control del proceso o de los recursos del sistema.
4. **Múltiples instancias sin conocimiento a priori en tiempo de ejecución:** una actividad es instanciada muchas veces en tiempo de ejecución, sin conocimiento de cuántas instancias se crearán ni en tiempo de diseño ni en tiempo de ejecución, hasta que se haya terminado de crear las instancias.

Patrones estructurales.

1. **Ciclos Arbitrarios:** se trata de un punto en el proceso donde una o más actividades pueden ser realizadas en forma repetitiva. Este patrón existe para diferenciar los Workflows que necesitan de un constructor explícito de repetición.
2. **Terminación Implícita:** un proceso debe terminar cuando no quedan actividades pendientes y cuando no está en tiempo muerto o deadlock. Se evita un terminador explícito que cancela las actividades pendientes.

Patrones basados en el estado.

1. **Selección Diferida:** se trata de un punto en el proceso donde uno o múltiples caminos son seleccionados.

2. **Enrutamiento en paralelo entrelazado:** un conjunto de actividades es ejecutado en un orden arbitrario, decidido en tiempo de ejecución.
3. **Hito:** permite testear si un proceso de Workflow ha alcanzado determinada fase o etapa.

Patrones de cancelación

1. **Cancelar Actividad:** el sistema permite que al completarse una actividad, otra actividad sea cancelada o deshabilitada.
2. **Cancelar Caso:** una instancia completa del proceso es cancelada.

1.3. Tendencias y tecnologías actuales

La industria de software ha mostrado ser una de las áreas más dinámicas y con mayor crecimiento en los últimos años. La evolución hacía a un modelo más racional para los usuarios, con menores costos de licencia , donde se intensifique la prestación de servicios, que permitirá además reducir el tiempo de desarrollo y por supuesto incrementar la calidad viene siendo lo más importante a la hora de desarrollar cualquier tipo de aplicación. A continuación se ofrecerá una valoración sobre algunas de las tendencias y tecnologías que marcan un nivel alto en el mundo del software y que bien contribuye a lo dicho anteriormente:

Enfoque Orientado a Objetos (OO)

El enfoque Orientado a Objetos actualmente se encuentra en una etapa de madurez como paradigma del desarrollo de sistemas de información. Ayuda a explotar el poder expresivo de todos los lenguajes de programación basados en objetos y los orientados a objetos, como: Java y PHP 5, apoya la reutilización no sólo del software, sino de diseños completos, produce sistemas que están construidos en formas intermedias estables y por ello son más resistentes al cambio en especificaciones y tecnología y brinda un mecanismo para formalizar el modelo de la realidad.

Aplicación Web

Con la creación de aplicaciones de este tipo se logra:

Compatibilidad Multiplataforma: una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows, Linux, Mac, etc.

Menos requerimientos de hardware: este tipo de aplicación no consume (o consume muy poco) espacio en disco y también es mínimo el consumo de memoria Random Access Memory (RAM) en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras

con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor en donde reside la aplicación.

Actualización: una de las características fundamentales de las aplicaciones web es que siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones ni realizar tareas de instalación.

Acceso inmediato y desde cualquier lugar: las aplicaciones basadas en tecnologías web no necesitan ser descargadas, instaladas y configuradas. Además pueden ser accedidas desde cualquier computadora conectada a la red en donde se accede a la aplicación.

Seguridad en los datos: los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que usualmente sufren los ordenadores de usuarios comunes como virus y roturas de disco.

La utilización de esta tecnología unida al uso del patrón Modelo-Vista-Controlador (MVC) y la arquitectura en capas conlleva a reducir costos y complicaciones y proporciona mayor libertad a la hora de realizar cualquier tipo de cambios, siendo así una excelente opción para que las pequeñas empresas automaticen sus procesos sin invertir demasiado en equipo, desarrollo y capacitación.

Desarrollo ágil.

Las metodologías ágiles surgen como una extensión a las metodologías tradicionales para mejorar el desarrollo de sistemas según el tipo de proyecto y empresa, añadiendo y mejorando las prácticas de desarrollo de software. Estas representan sin dudas uno de los temas recientes en ingeniería de software que están acaparando gran interés y entre sus principales ventajas encontramos:

Según el Manifiesto Ágil se valora:

- ✓ Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La persona es el principal factor de éxito de un proyecto software.
- ✓ Desarrollar un software que funciona más que conseguir una buena documentación.
- ✓ La colaboración con el cliente más que la negociación de un contrato.
- ✓ Responder a los cambios más que seguir estrictamente un plan.

Software libre:

Catalizador que permite crear soluciones cada vez más sofisticadas y más personalizadas. Se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software y que

suele estar disponible gratuitamente, o al precio de costo de la distribución y dos de sus ventajas fundamentales se encuentran:

1. **Independencia tecnológica:** El acceso al código fuente permite el desarrollo de nuevos productos sin la necesidad de desarrollar todo el proceso partiendo de cero.
2. **Libertad de uso y redistribución:** Las licencias de software libre existentes permiten la instalación del software tantas veces y en tantas máquinas como el usuario desee.

1.4. Patrones

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio, son soluciones de sentido común que deberían formar parte del conocimiento de un diseñador experto. Además, facilitan la comunicación entre diseñadores, pues establecen un marco de referencia.

1.4.1. Patrones de Diseño

Los patrones de diseño (del inglés design patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema del diseño.

Los patrones de diseño pretenden:

- ✓ Proporcionar catálogos de elementos reusables en el diseño de sistemas de software.
- ✓ Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- ✓ Formalizar un vocabulario común entre diseñadores.
- ✓ Estandarizar el modo en que se realiza el diseño.
- ✓ Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

1.4.1.1. Patrones de Software para la asignación General de Responsabilidad (Grasp)

Los patrones Grasp describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Booch y Rumbaugh definen la responsabilidad como "un contrato u obligación de un tipo o clase"

Existen nueve patrones Grasp los cuales son: Experto, Creador, Alta Cohesión, Bajo Acoplamiento, Controlador, Polimorfismo[4], Fabricación Pura, Induración y No Hables con Extraños, de estos se utilizaron con el fin de que contribuya a que el sistema sea más robusto y flexible los que se enumeran a continuación: (13)

Experto: la solución se basa en asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Creador: teóricamente responde la interrogante de ¿Quién debería ser responsable de crear una nueva instancia de alguna clase?, asignando a una clase la responsabilidad de crear una instancia de otra clase en casos diferentes. (13)

Controlador: en la asignación de responsabilidades, es contradictorio definir ¿Quién debería encargarse de atender un evento del sistema?, sin embargo, el uso de este patrón elimina la incógnita anterior sugiriendo asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una de las clases. (13)

Lo descrito a priori significa asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, entre otras cosas.) Si se aplica estos principios, el controlador no realiza las actividades mencionadas sino que las delega en otras clases con las que mantiene un modelo de alta cohesión.

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 tiene una dependencia fuerte de las otras clases para realizar sus tareas, permitiendo que se pueda reutilizar con mayor facilidad y flexibilidad. **(13)**

Alta cohesión: la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realizan un trabajo enorme. Permite que las clases se puedan reutilizar con mayor facilidad y flexibilidad, mejorando la claridad y facilidad del diseño. **(13)**

1.4.1.2. Patrones Gof

Los patrones Gof son patrones de diseño publicados en el libro Design Patterns: Elements of Reusable Object-Oriented Software por Gamma, Helm, Jonson y Vlissides conocidos mundialmente por “Gang of

[4] Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.

Four" o Pandilla de los cuatro. En este libro se encuentran recopilados un total de 23 patrones clasificado en patrones creacionales, estructurales y de comportamiento.

- ✓ Los patrones creacionales se encargan de la creación de los objetos ayudando a que el sistema sea independiente de la creación, composición y representación de los objetos.
- ✓ Los patrones estructurales son los encargados de cómo las clases y objetos están compuestos para formar estructuras más grandes. Los patrones estructurales usan la herencia para componer interfaces u objetos en tiempo de ejecución.
- ✓ Los patrones de comportamiento plantean algoritmos y la asignación de responsabilidades entre objetos. Estos patrones no sólo describen clases y objetos sino también describen la comunicación entre ellos.

1.5. Modelo de desarrollo

La propuesta de modelo de desarrollo fue elaborada por el equipo de producción en colaboración con las Líneas de desarrollo del proyecto ERP de acuerdo con las necesidades presentadas por cada una de ellas y teniendo en cuenta los principales riesgos con los que se cuentan en el proyecto.

1.5.1. Características

Centrado en la arquitectura

La arquitectura determina la línea base, los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

Orientado a componentes

Las iteraciones son orientadas por el nivel de significancia arquitectónicas de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

Iterativo e incremental

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

Ágil y adaptable al cambio

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

1.6. Lenguajes de desarrollo y de modelado

1.6.1. Lenguaje de modelado

Llamamos lenguaje de modelado de objetos a un conjunto estandarizado de símbolos y de modos de disponerlos para modelar (parte de) un diseño de software.

UML

Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Probablemente, una de las innovaciones conceptuales en el mundo tecnológico del desarrollo de software que más expectativas y entusiasmo han generado en muchos años. Es un estándar en la industria del software, creado por Grady Booch, James Rumbaugh e Ivar Jacobson.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos se crea una documentación que cualquier desarrollador con conocimientos de UML será capaz de entender. Su utilización es independiente del lenguaje de programación y de las características de los proyectos ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otra rama. (14)

1.6.2. Lenguajes de programación

El término lenguaje de programación está dado por un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen la estructura, el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

1.6.2.1. Lenguajes del lado del servidor

Se les clasifica así a los lenguajes de programación en la tecnología cliente servidor que se ejecutan del lado del servidor y de los que los cuales los usuarios sólo obtienen el beneficio del procesamiento de la información.

Lenguaje PHP

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. Es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en la web con distintas prestaciones de forma rápida. No requiere definición de tipos de variables, no es un lenguaje de marcas. Su interpretación y ejecución se realiza en el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. Permite la conexión a numerosas bases de datos de forma nativa tales como Postgres, MySQL, Oracle, ODBC, Microsoft SQL Server, entre otras, lo cual permite la creación de Aplicaciones web muy robustas.

PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux, Windows y Mac OS X y puede interactuar con los servidores de web más populares.

Ventajas

- ✓ Es un lenguaje multiplataforma.
- ✓ Capacidad de expandir su potencial utilizando gran cantidad de módulos.
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Es libre, lo que representa que una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.
- ✓ Permite las técnicas de Programación Orientada a Objetos[5].
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida

Desventajas

- ✓ No posee una abstracción de base de datos estándar, sino bibliotecas especializadas.
- ✓ Por sus características promueve la creación de código desordenado y complejo de mantener.

[5] La terminología Programación Orientada a Objetos (POO u OOP según siglas en inglés) representa un paradigma de programación donde se definen los programas en términos de "clases de objetos", tales objetos son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto).

- ✓ Todo el trabajo lo realiza el servidor, por tanto puede ser más ineficiente a medida que aumenten las solicitudes.
- ✓ La orientación a objetos es aún muy deficiente para aplicaciones grandes.

En este caso se estará haciendo uso de PHP 5.2, con los siguientes módulos o extensiones: pdo, pdo_pgsql, pgsql, soap, xsl.

1.6.2.2. Lenguajes del lado del cliente

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.

HTML[6]

Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. (15)

XML

Es el metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos. Por lo tanto el Lenguaje de Marcas Extensible (XML) no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

XML se desarrolló para proporcionar una flexibilidad y consistencia que no se podían alcanzar con HTML, no sólo es un lenguaje de marcado, sino también un metalenguaje cuya particularidad más importante es que no posee etiquetas prefijadas con anterioridad, permitiendo describir otros lenguajes de marcado y definir lenguajes de presentación propios en dependencia del contenido del documento.

[6] Acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. HTML es una aplicación de Lenguajes de Marcas Generalizados (Standard Generalized Markup Language SGML por sus siglas en inglés) conforme al estándar internacional ISO 8879.

JavaScript

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia[7], es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM[8]. JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página Web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros. (16)

XHTML

"XHTML es el acrónimo en inglés de Extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto). Es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos." (17)

XHTML reúne la capacidad de formato de HTML y se consolida con la formalidad del XML a la hora de estructurar documentos para la portación de datos. Está encaminado al uso de un etiquetado correcto, por lo que exige una serie de requisitos básicos a cumplir en cuanto al código. Algunos de estos requisitos son:

1. Elementos correctamente anidados
2. Etiquetas en minúsculas
3. Elementos cerrados correctamente
4. Atributos[9] de valores entrecomillados.

[7] La herencia es específica de la programación orientada a objetos, donde una clase nueva se crea a partir de una clase existente. La herencia (a la que habitualmente se denomina subclases) proviene del hecho de que la subclase (la nueva clase creada) contiene las atributos y métodos de la clase primaria. (3)

[8] DOM: El Document Object Model (Modelo de Objetos de Documento), Es una plataforma que proporciona un conjunto estándar de objetos a través de la cual se pueden crear documentos HTML y XML, proporcionando una interfaz estándar para que otro software manipule los documentos.

[9] Contenedor de un tipo de datos asociados a un objeto, que hace los datos visibles desde fuera del objeto, y cuyo valor puede ser alterado por la ejecución de algún método.

CSS

Es un lenguaje de hojas de estilos (Cascading Style Sheets) creado para controlar la presentación de documentos estructurados y escritos en HTML y XHTML, aspectos como: el color, el tamaño, el tipo de letra, la separación entre párrafos y la tabulación con la que se muestran los elementos de una lista. El propósito del desarrollo de CSS es separar la estructura y el contenido de la presentación estética en un documento, esto permite un control mayor del documento y sus atributos, convirtiendo al HTML en un documento muy versátil y liviano. (18)

1.6.3. ExtJS

ExtJS posee una librería inmensa que permite configurar las interfaces Web de manera semejante a aplicaciones desktop, incluye la mayoría de los controles de los formularios Web basándose en Grids para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería contiene componentes para el manejo de datos, lectura de XML, lectura de datos JSON[10] e implementaciones basadas en AJAX.

Actualmente ExtJS es considerado un Framework[11] independiente, ya que a principios del 2007 se creó una compañía para comercializar y dar soporte al mismo, dicha compañía proporciona los servicios de consultoría necesarios para ayudar a los clientes en el aprovechamiento máximo de las ventajas de ExtJS. Es importante señalar que la ExtJS 2.2 tiene dos tipos de licencias, Lesser General Public License (LGPL) y la comercial, esta última es obligatoria si se desea obtener soporte.
Se estará haciendo uso de Ext. 2.2

1.6.4. Zend Framework

Es un framework de alta calidad y de código abierto para el desarrollo de aplicaciones y servicios web con PHP. Zend Framework brinda facilidades de uso y poderosas funcionalidades. Proporciona soluciones para construir modernas, robustas y seguras aplicaciones web, está diseñado para usar todas las potencialidades de php5.

Presenta entre otras, las siguientes características:

[10] "JavaScript Object Notation", es un formato ligero para el intercambio de datos.

[11] Denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.

- ✓ Proporciona un sistema de caché dividido en frontend y backend, de forma que se puedan almacenar en caché diferentes datos como resultados de funciones, páginas completas, etc. y que esta información se almacene en archivos, en memoria, en base de datos, etc.
- ✓ Proporcionan los componentes que forma la infraestructura del patrón MVC.
- ✓ Proporciona una capa de acceso a base de datos, construida sobre PDO pero ampliándola con diferentes características.
- ✓ Proporciona mecanismos de filtrado y validación de entradas de datos.
- ✓ Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX.

Se estará haciendo uso de Zend Framework 1.9.7

1.6.5. Doctrine

Doctrine es un potente y completo sistema ORM[12] (object relational mapping) para PHP 5.2 con un DBAL (database abstraction layer) incorporado, el mismo cuenta con disimiles funcionalidades, una de ellas es que da la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO), debido a que doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente.

El patrón “Active Record es una extensión del patrón Domain Model (”Modelo de Dominio”), que se entiende como una clase o un grupo de clases que representan a “objetos” o responsabilidades particulares en la aplicación”. (19)

Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario.

Se estará haciendo uso de Doctrine 1.2.1.

1.6.6. Zend_Ext Framework

Es elaborado a partir de Zend Framework cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizada por las vistas hacia el

[12] ORM es un componente de software que me permite trabajar con los datos persistidos como si ellos fueran parte de una base de datos orientada a objetos. (30)

controlador, un motor de reglas para las validaciones en el servidor, se le incluyó el IoC[13] para la comunicación en entre los módulos o componentes, la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos, el ExtJS Framework para el desarrollo de las vistas y un controlador de trazas para controlar las acciones del sistema (acción, excepciones, rendimiento, integración y excepción de integración).

1.6.7. UCID Framework

Es el Framework encargado del trabajo con la vistas. Abarca la integración de ExtJS Framework con el sistema incluyendo el integrador de interfaz, el generador de interfaz dinámica y la impresión de documentos. Integra la iconografía, los diferentes temas de escritorio de la aplicación y el multilenguaje.

1.7. Tecnologías y Herramientas de desarrollo

Para la realización de un proyecto de esta magnitud es necesario que cada uno de los equipos de desarrollo posean un modelo estandarizado de las tecnologías y herramientas a utilizar conjuntamente con sus versiones para la implementación de las capas de presentación, negocio y acceso a datos. De acuerdo a lo planteado anteriormente la dirección del proyecto determino emplear:

1.7.1. Tecnologías

Tecnología AJAX

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting lenguaje) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. (20)

[13] Inversión de control (Inversion of Control, IoC por sus siglas en inglés). Es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones.

1.7.2. Herramientas CASE

CASE son las siglas correspondientes a Computer Aided Software Engineering, que en su traducción al español significa Ingeniería de Software Asistida por Computadoras. Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Están destinadas a:

1. Mejorar la productividad en el mantenimiento y desarrollo del software.
2. Automatizar el desarrollo del software, generación de código, pruebas de errores y gestión del proyecto.
3. Mejorar el tiempo y costo de desarrollo y mantenimiento de los sistemas informáticos.
4. Aumentar la calidad del software.
5. Facilitar el uso de las distintas metodologías propias de la ingeniería del software

Visual Paradigm

Es una herramienta CASE de modelado que utiliza UML como lenguaje de modelado profesional y que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar ingeniería tanto directa como inversa. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como HTML, PDF y permite control de versiones. Dentro de sus características principales se puede destacar su robustez, usabilidad y portabilidad. Para el desarrollo de este trabajo se escogió como herramienta CASE de modelado al Visual Paradigm porque tiene un coste favorable y realiza de una forma íntegra los planes de construcción del software. Permite ingeniería inversa, del modelo físico se puede llegar al modelo lógico. Genera de forma automática códigos desde diagramas, además brinda la posibilidad de generar documentación.

1.7.3. Herramienta de desarrollo colaborativo

Control de versiones

Una versión, revisión o edición de un producto, es el estado en que se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Un sistema de control de versiones debe proporcionar un mecanismo de almacenaje de los elementos que deba gestionar y un registro histórico de las acciones

realizadas con cada elemento o conjunto de elementos (normalmente brindando la posibilidad de volver o extraer un estado anterior del producto) entre otros aspectos. Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados. Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial.

Subversion

También conocido como SVN[14], es un sistema de control de versiones que se ha popularizado bastante, en especial dentro de la comunidad de desarrolladores de software libre. Está preparado para funcionar en red y se distribuye bajo licencia libre.

- ✓ Mantiene versiones no sólo de archivos, sino también de directorios
- ✓ Mantienen versiones de los metadatos asociados a los directorios.
- ✓ Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- ✓ Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- ✓ Soporte tanto de ficheros de texto como de binarios.
- ✓ Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.

Tortoise SVN 1.4.5

Tortoise SVN es el cliente gratuito para el sistema de control de versiones Subversion, con código abierto y software libre bajo la licencia GNU[15] General Public License (GPL). Está disponible en 28 idiomas diferentes y puede ser usado sin un entorno de desarrollo está basado en las bibliotecas SVN y actúa de cliente gráfico para el acceso al repositorio SVN, tanto si éste es remoto como si es local. Este cliente gráfico ha sido diseñado para su integración contextual en el Explorador de Windows proporcionando el acceso a la mayoría de funciones que proporciona SVN. (21)

[14] Es el nombre que se utiliza en la línea de comandos Ej: **$ svn export nombre-de-repositorio**
[15] Acrónimo recursivo que significa No es Unix.

1.7.4. IDE

Entorno de Desarrollo Integrado que (en inglés IDE) permite de forma cómoda y ágil editar, compilar, ejecutar y depurar programas.

Spket

Spket es un plugin para Eclipse y Aptana que provee un conjunto de utilidades para la edición de JavaScript, sobre todo para la edición de clases que extienden el framework JavaScript ExtJS o que usan la librería. Proporciona un editor de código JavaScript muy parecido al editor Java de Eclipse, es decir, incluye autocompletado de código, resaltado de texto, muestra errores, etc.

Spket IDE es una excelente aplicación que ofrece la posibilidad de editar en lenguaje de programación JavaScript, para la creación de utilidades menores. Dentro de sus características, se destacan el autocompletado de comandos, diferenciación por colores de la sintaxis, etc. Cuenta con un funcionamiento totalmente sencillo para todo aquel programador profesional o aficionado y posee una interfaz gráfica verdaderamente eficiente y completa para la edición de aplicaciones. (22)

1.7.5. Servidor de Aplicaciones web

Es un programa que permite crear un Servidor de Protocolos de Transferencia de Hipertexto (HTTP) en un ordenador de una forma rápida y sencilla. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Es además un programa que implementa el protocolo HTTP el cual se encarga de transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Servidor web Apache

Apache 2.0 es un servidor Web potente, flexible y disponible para distintas plataformas y entornos. Es altamente configurable y de diseño modular, posibilitando que los administradores de sitios Web puedan elegir los módulos que serán incluidos y ejecutados en el servidor.

Características de Apache:

- ✓ Es una tecnología gratuita y de código abierto, lo que proporciona transparencia en todo el proceso de instalación.
- ✓ Es prácticamente universal, por su disponibilidad en multitud de sistemas operativos.
- ✓ Posee una alta configurabilidad en la creación y gestión de logs, de este modo es posible tener un mayor control sobre lo que sucede en el servidor.

Este servidor Web tiene una fácil integración con varios lenguajes de programación como: Java, Perl y especialmente PHP. Dicha relación a dado a lugar el desarrollo de aplicaciones como el APPSERV y XAMPP los cuales instalan el Apache y el PHP configurados para su uso.

Se estará haciendo Servidor web Apache 2.0 o superior.

1.7.6. Sistemas de gestión de base de datos

Los servidores de bases de datos surgen producto de la necesidad de las empresas de manejar grandes y complejos volúmenes de datos, al tiempo que requieren compartir la información con un conjunto de clientes de una manera segura.

Una de las funciones que se empieza a exigir a los SGBD, es la de proporcionar herramientas de apoyo a toma de decisiones ("Datawarehouse[16]") al tiempo que proporciona una plataforma de transacciones "on-line" (OLTP) que hacen que la información esté siempre actualizada y consistente.

Los SGBD proporcionan herramientas de administración completas que simplifican la tarea de la configuración, seguridad, creación y gestión de bases de datos, al tiempo que proporcionan mecanismos de integración con otros sistemas y políticas de copias de seguridad y herramientas que permitan su programación tanto a nivel de diseño como a nivel de reglas y procedimientos que encapsulen la arquitectura de la base de datos. Estos sistemas deben proporcionar mecanismos de comunicación con otras plataformas que actúen también como clientes o servidores de datos.

PostgreSQL 8.3

PostgreSQL es un servidor de base de datos relacional, libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, sub-consultas y joins de gran tamaño.

Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que dicha herramienta es multiplataforma. Fue diseñado para ambientes de alto volumen.

[16] Colección de datos orientada a un determinado ámbito, integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza; trata sobre todo, de un expediente completo de una organización, más allá de la información transaccional y operacional, almacenado en una base de datos diseñada para favorecer el análisis y la divulgación eficiente de datos.

Escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos.

PGAdmin III

"Es una aplicación gráfica usada para la gestión de PostgreSQL, siendo la más completa y popular con licencia Open Source[17]. PGAdmin está escrito en C++ y utiliza la librería gráfica multiplataforma wxWidgets[18], permitiendo que se pueda usar en sistemas operativos como: GNU/Linux[19], MacOS y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3, ejecutándose en cualquier plataforma. PGAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente su administración."

1.7.7. Navegador

El navegador es una especie de aplicación capaz de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en las páginas que son el resultado de dicha orden. Permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web.

Mozilla Firefox

Mozilla Firefox es el nuevo e innovador navegador open source. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. Se trata de un práctico y ágil navegador, que está en renovación constante. Tiene la capacidad de modificarlo totalmente a gusto del usuario y según las necesidades del mismo. Esto se consigue gracias a la multitud de "extensiones" que existen y que cada día aparecen más, que permiten añadirle nuevas funciones de todo tipo. (23)

Se estará haciendo uso de Mozilla Firefox 3.0.

[17] Open Source: Código abierto, es el término con el que se conoce al software distribuido y desarrollado libremente.
[18] Son unas bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++. Están publicadas bajo una licencia LGPL, similar a la GPL con la excepción de que el código binario producido por el usuario a partir de ellas, puede ser propietario, permitiendo desarrollar aplicaciones empresariales sin coste.
[19] GNU/Linux: Es el término empleado para referirse al sistema operativo Unix-like que utiliza como base las herramientas de sistema de GNU y el núcleo Linux.

1.7.8. Herramientas para la realización de Pruebas de Caja Blanca.

JMeter

Herramienta de carga para llevar a cabo simulaciones sobre cualquier recurso de software, inicialmente diseñada para pruebas de estrés en aplicaciones web, hoy en día, su arquitectura ha evolucionado no sólo para llevar acabo pruebas en componentes habilitados en Internet (HTTP), sino además en Bases de Datos, programas en Perl y prácticamente cualquier otro medio.

Además, posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de peticiones que permiten diagnosticar el comportamiento de una aplicación en condiciones de desarrollo, en éste sentido, simula todas las funcionalidades de un navegador o de cualquier otro cliente, siendo capaz de manipular resultados en determinada petición y ser reutilizables para ser empleados en una nueva secuencia. JMeter es una herramienta código abierto muy completa, implementada en Java. (24)

Conclusiones

- ✓ Se realizó un estudio profundo de las características de los WorkFlow y de los SGFT confirmando la necesidad e importancia de la adopción de un componente que implemente sus principales funciones.
- ✓ Se hizo un estudio de las tendencias y tecnologías actuales y se estuvo de acuerdo con la elección del equipo de arquitectura del proyecto.
- ✓ Se analizaron los patrones de software a utilizar, haciendo énfasis en su importancia.
- ✓ Se caracterizaron los lenguajes de modelado y de programación a utilizar analizando sus ventajas y desventajas.
- ✓ Se realizó un evaluación de las herramientas a utilizar destacando la importancia de su elección en la futura elaboración de la solución.

2. CAPÍTULO II DISEÑO DE LA SOLUCIÓN

Introducción

En el presente capítulo se propone modelar la solución, a partir de artefactos que tendrán un valor significativo en la posterior etapa de implementación, de ésta manera se obtendrán los modelos de Componentes, de Entidad Relación, de Datos y el de Clases, compuestos por los diagramas y la descripción de los mismos.

Dada la arquitectura previamente definida por la dirección del proyecto se valorará la posibilidad de reutilización de tanto servicios como componentes, ya que ofrece ventajas significativas en la rapidez del proceso y adaptación de la solución a los módulos ya existentes. Se determinarán los estándares de codificación y la estructura de datos con el fin de mantener la uniformidad en cada una de las etapas y se definirán las clases y las funcionalidades necesarias. Por último, se describirá la estrategia de integración al proyecto.

2.1. Requisitos Funcionales

Los requisitos funcionales identificados y aprobados se agruparon en un conjunto de funcionalidades con una finalidad u objetivo específico:

1. Gestionar Estado.
   - ✓ Adicionar Estado.
   - ✓ Modificar Estado.
   - ✓ Eliminar Estado.
   - ✓ Consultar Estado.
   - ✓ Listar Estados.
2. Gestionar Tipo de Documento.
   - ✓ Adicionar Tipo de Documento.
   - ✓ Modificar Tipo de Documento.
   - ✓ Eliminar Tipo de Documento.
   - ✓ Consultar Tipo de Documento.
   - ✓ Asociar Estado al Tipo de Documento.
   - ✓ Desasociar Estado al Tipo de Documento.
   - ✓ Asociar Subsistema al Tipo de Documento.
   - ✓ Desasociar Subsistema al Tipo de Documento.

- ✓ Listar Tipo de Documentos.
- ✓ Listar Subsistemas.

3. Gestionar Flujo de Trabajo.

- ✓ Adicionar Transición.
- ✓ Modificar Transición.
- ✓ Eliminar Transición.
- ✓ Consultar Transición.
- ✓ Asociar Usuarios Simultáneos a la Transición.
- ✓ Asociar Usuarios Participantes a la Transición.
- ✓ Desasociar Usuarios Simultáneos a la Transición.
- ✓ Desasociar Usuarios Participantes a la Transición.
- ✓ Asociar Pre-condiciones a la Transición.
- ✓ Desasociar Pre-condiciones a la Transición.
- ✓ Asociar Post-condiciones a la Transición.
- ✓ Desasociar Post-condiciones a la Transición.
- ✓ Listar Post-condiciones.
- ✓ Listar Pre-condiciones.
- ✓ Listar Usuarios.
- ✓ Listar Estados a Modificar el Tipo de Documento.
- ✓ Listar Estados a Ver el Tipo de Documento.
- ✓ Ejecutar Flujo de Trabajo.
- ✓ Ejecutar Post-condiciones.
- ✓ Ejecutar Pre-condiciones.

Valoración crítica de la Especificación de Requisitos

El artefacto Especificación de Requisitos del Software para el componente Workflow como resultado del previo análisis efectuado constituye un elemento clave para el diseño y la posterior implementación.

Básicamente se caracteriza por presentar los requisitos de forma completa, presentando una única interpretación evitando la ambigüedad de las definiciones y funcionalidades, están clasificados por la importancia arquitectónica y desde el punto de vista del cliente. Viabiliza las modificaciones y comprobación de los requisitos especificados.

Según lo descrito con anterioridad, la Especificación de Requisitos presentada puede ser tomada como artefacto de entrada al Diseño para facilitar la compresión del equipo de desarrollo.

2.2. Diagramas de Clases de Diseño

Los diagramas de clases según la clasificación UML son diagramas de estructura estática donde la representación de los requerimientos se lleva a cabo a través de las clases del sistema y sus interrelaciones. Representan una abstracción del dominio de modo que es formalizado el análisis de conceptos y constituyen el pilar básico del modelado, mostrando en términos generales qué debe hacer el sistema.

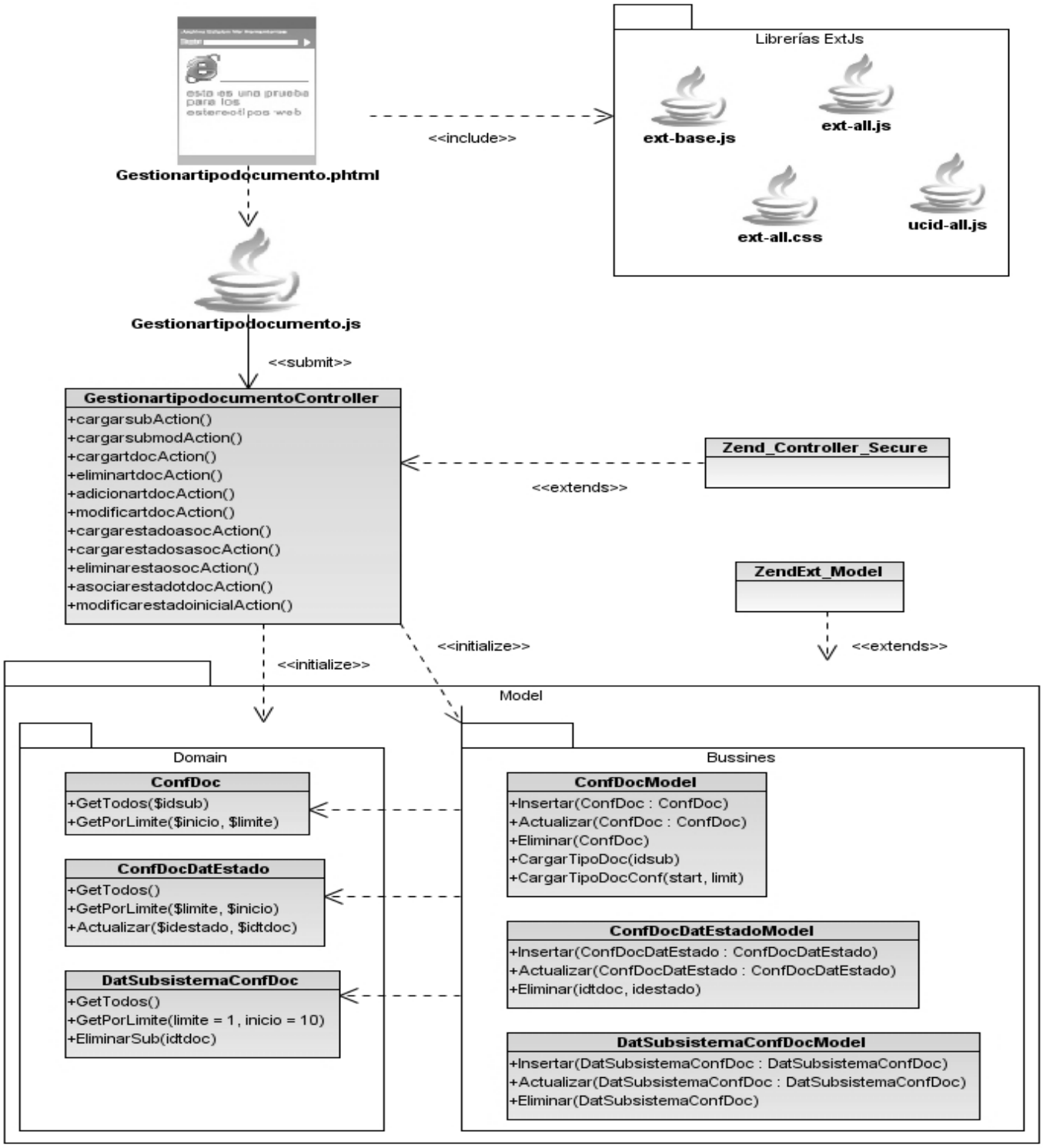


Figura 2. Diagrama de Clases del Escenario Gestionar Documento.

Tabla 1. Descripción del diseño de clases para el escenario de Gestionar Documentos.

Clases	Descripción
Librería ExtJs	Contiene los componentes generados a través de la librería JavaScript Extjs.
GestionarTipoDocumento.phtml	Es la página que se encargaría de visualizar a través de los js que debe incluir, la información relacionada con la configuración inicial del WorkFlow.
Configuración.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería ExtJS los componentes que manejen la información en la vista necesaria para realizar la configuración mínima del workflow. Debe enviar y recibir los datos de la controladora utilizando tecnología AJAX.
ConfiguraciónController	Clase controladora responsable de configurar los destinos donde se deben ejecutar las diferentes funcionalidades, teniendo en cuenta las peticiones del usuario.
ZendExt_Controller_Secure	Encargada de gestionar la seguridad en la página controladora.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el paquete Bussines y el Domain.
ZendExt_Model	Encargada de gestionar la seguridad de los datos persistentes ubicados en el Model.

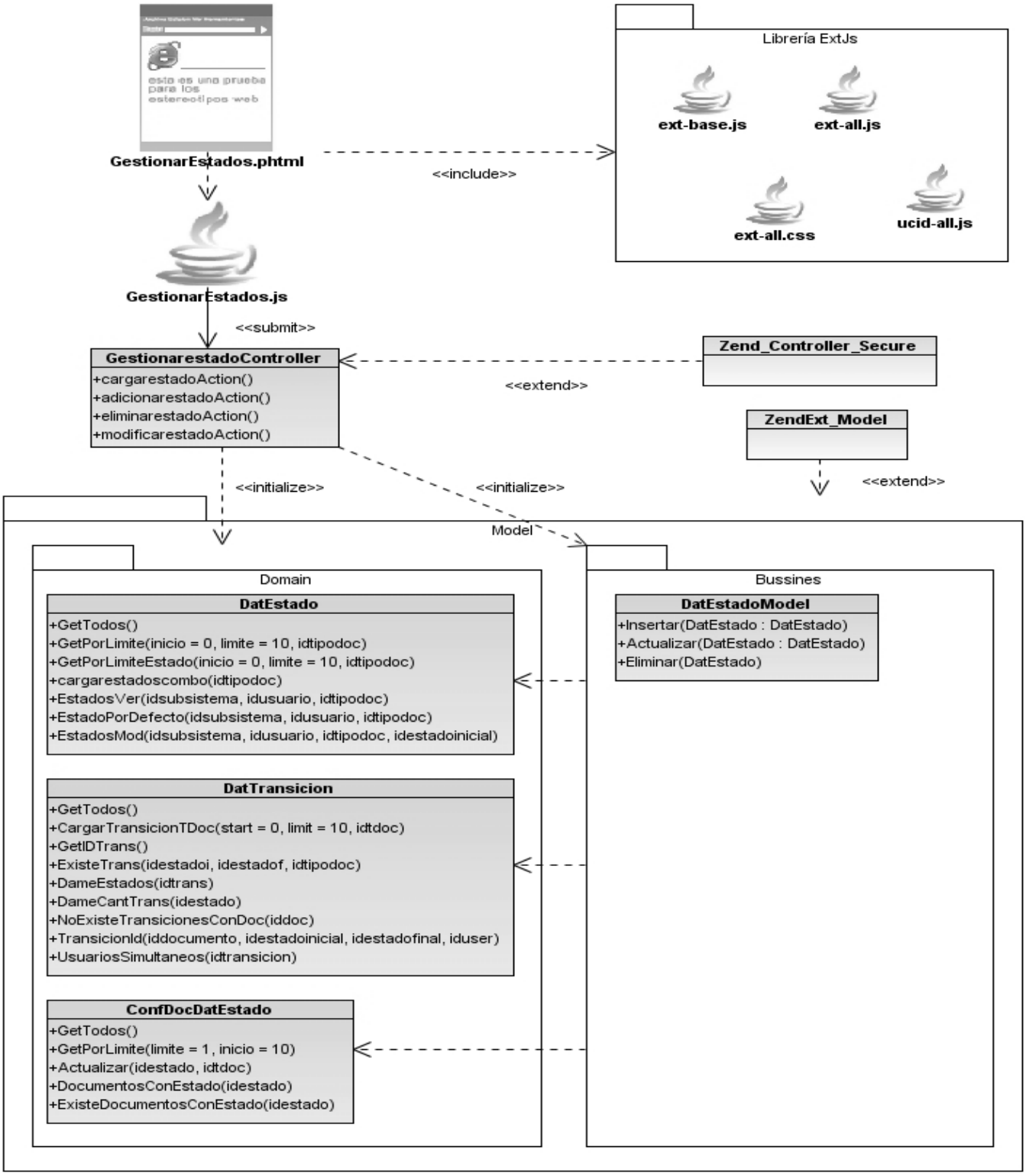


Figura 3. Diagrama de Clases del Escenario Gestionar Estados

Tabla 2. Descripción del diseño de clases para el escenario de GestionarEstados.

Clases	Descripción
Librería ExtJs	Contiene los componentes generados a través de la librería JavaScript Extjs.
GestionarEstados.phtml	Es la página que se encargaría de visualizar a través de los js que debe incluir, la información relacionada con los estados de los documentos.
GestionarEstados.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería Extjs los componentes que manejen la información en la vista necesaria para gestionar los estados asociados a los diferentes tipos de documento asociados al los diferentes subsistemas. Debe enviar y recibir los datos de la controladora utilizando tecnología AJAX.
GestionarEstados Controller	Clase controladora responsable de configurar los destinos donde se deben ejecutar las diferentes funcionalidades, teniendo en cuenta las peticiones del usuario.
ZendExt_Controller_Secure	Encargada de gestionar la seguridad en la página controladora.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el paquete Bussines y el Domain.
ZendExt_Model	Encargada de gestionar la seguridad de los datos persistentes ubicados en el Model.

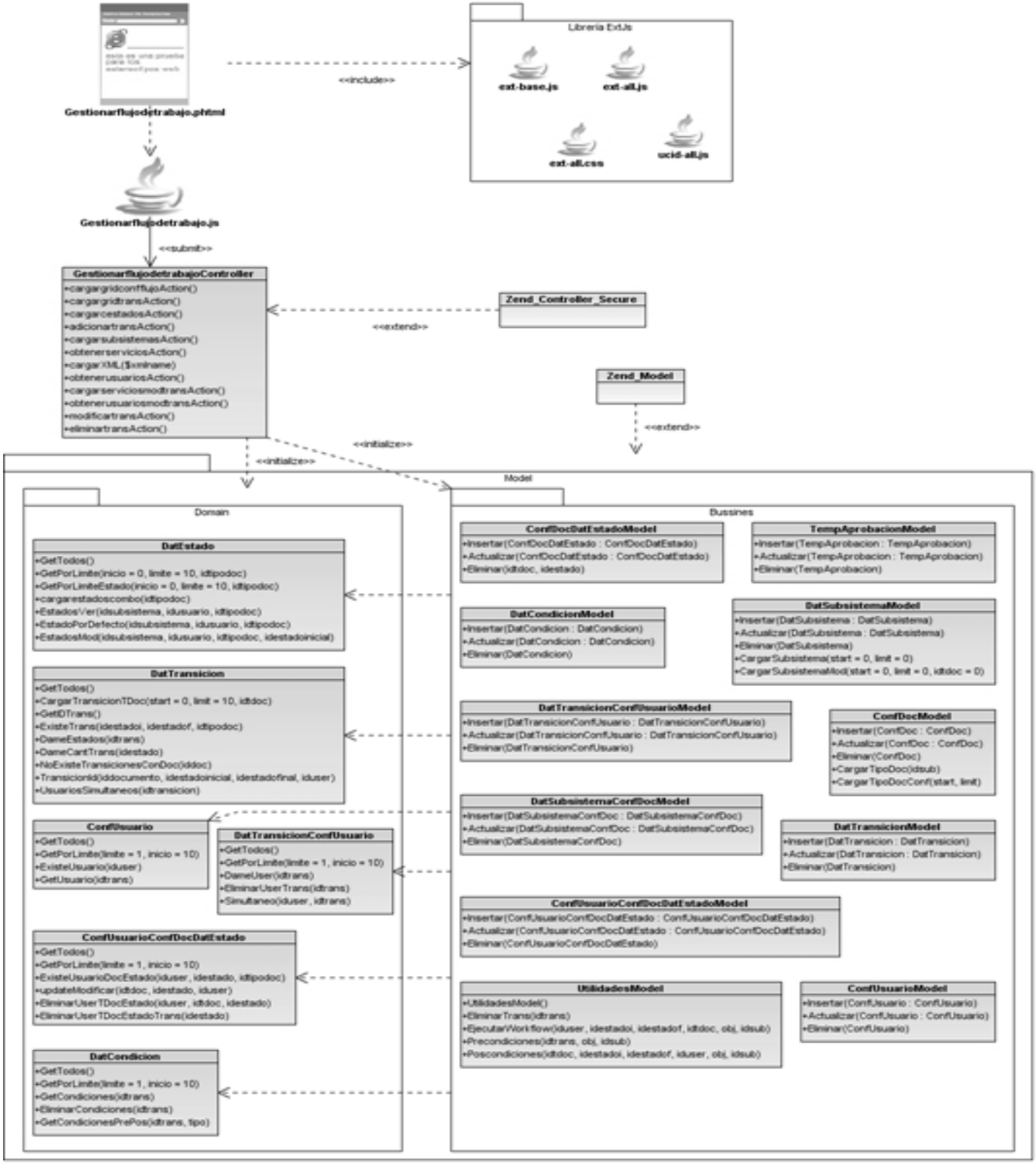


Figura 4. Diagrama de Clases del Escenario Gestionar Flujo de Trabajo.

Tabla 3. Descripción del diseño de clases para el escenario de WorkFlow.

Clases	Descripción
Librería ExtJs	Contiene los componentes generados a través de la librería JavaScript Extjs.
gestionarflujodetrabajo.phtml	Es la página que se encargaría de visualizar a través de los js que debe incluir, la información relacionada con el funcionamiento del WorkFlow.
gestionarflujodetrabajo.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería ExtJS los componentes que manejen la información en la vista necesaria para gestionar los flujos de trabajos por subsistemas. Debe enviar y recibir los datos de la controladora utilizando tecnología AJAX.
gestionarflujodetrabajoController	Clase controladora responsable de configurar los destinos donde se deben ejecutar las diferentes funcionalidades, teniendo en cuenta las peticiones del usuario.
ZendExt_Controller_Secure	Encargada de gestionar la seguridad en la página controladora.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el paquete Bussines y el Domain.
ZendExt_Model	Encargada de gestionar la seguridad de los datos persistentes ubicados en el Model.

2.3. Arquitectura

Un elemento clave en el desarrollo del proceso de software es el diseño de la arquitectura, sobre ella se apoyan todas las representaciones de la estructura general de la aplicación a desarrollar, la misma es la que le da forma al software para que soporte todos los requisitos y establece los fundamentos para que todos los involucrados en el equipo de desarrollo trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema.

La arquitectura proporciona una visión global del sistema a construir. Describe la estructura y la organización de los componentes del software, sus propiedades y las conexiones entre ellos. Los componentes del software incluyen módulos de programas y varias representaciones de datos que son manipulados por el programa. (25)

2.3.1. MVC

Para la arquitectura de sistema, los componentes presentan una estructura, elaborada a partir del framework con que se trabaja (ExtJS, Zend, Doctrine) y de las necesidades del diseño. El elemento de peso que pone el framework es la implementación del estilo o patrón arquitectónico Modelo-Vista-Controlador (MVC). La estructura es la que se muestra en el Anexo 3.

Ventajas del modelo

- ✓ Clara separación entre interfaz, lógica de negocio y de presentación, que además provoca parte de las ventajas siguientes.
- ✓ Sencillez para crear distintas representaciones de los mismos datos.
- ✓ Facilidad para la realización de pruebas unitarias de los componentes, así como de aplicar desarrollo guiado por pruebas.
- ✓ Reutilización de los componentes.
- ✓ Simplicidad en el mantenimiento de los sistemas.
- ✓ Facilidad para desarrollar prototipos rápidos.
- ✓ Los desarrollos suelen ser más escalables.

Desventajas

- ✓ Tener que ceñirse a una estructura predefinida, lo que a veces puede incrementar la complejidad del sistema. Hay problemas que son más difíciles de resolver respetando el patrón MVC.
- ✓ La curva de aprendizaje para los nuevos desarrolladores se estima mayor que la de modelos más simples.
- ✓ La distribución de componentes obliga a crear y mantener un mayor número de ficheros.

De ésta manera se desarrolla usando la librería ExtJS para la vista, Zend Framework para el controlador y el Doctrine para el modelo.

2.3.2. Basada en componentes

La vista de componentes es una de las subdivisiones que se encuentran en la vista de sistema, es la encargada de definir los tipos de componentes posibles en el proyecto, de la especificación de sus características, así como de la composición estructural interna de cada uno de estos componentes (vista vertical de la arquitectura).

Basado en las características del sistema a construir se propone cinco clasificaciones para los componentes que conformarán el sistema, cada uno de ellos con un objetivo claro de gran impacto para el desarrollo y construcción de la arquitectura.

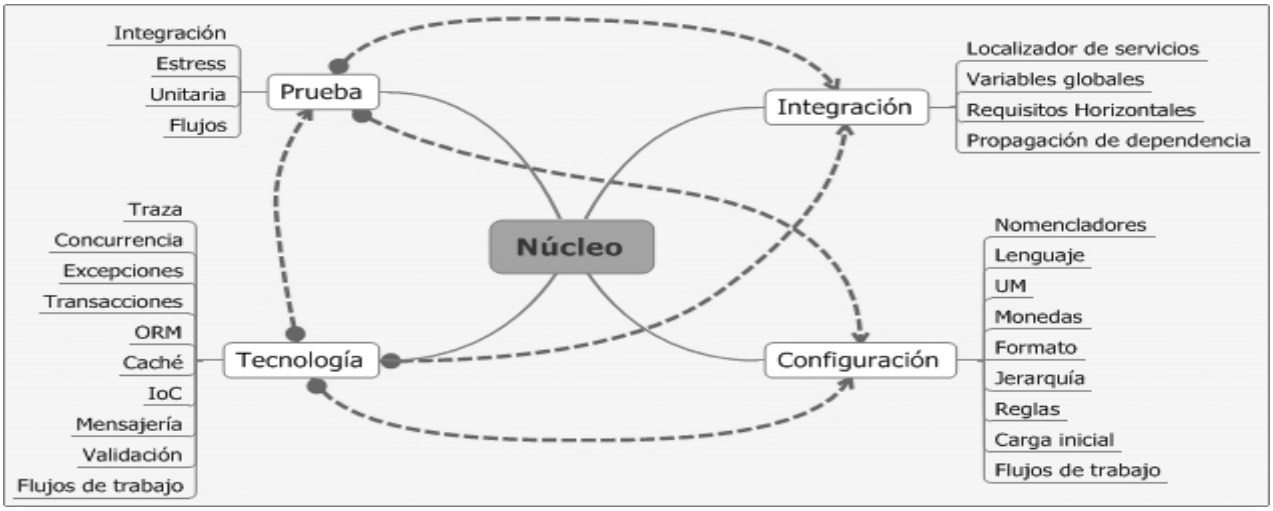


Figura 5 Clasificación de los componentes.

El componente WorkFlow se encuentra específicamente en la categoría Integración ya que representa a aquellos componentes con la responsabilidad de abstraer la estrategia de integración y colaboración del sistema, así como los flujos y subflujos de trabajo.

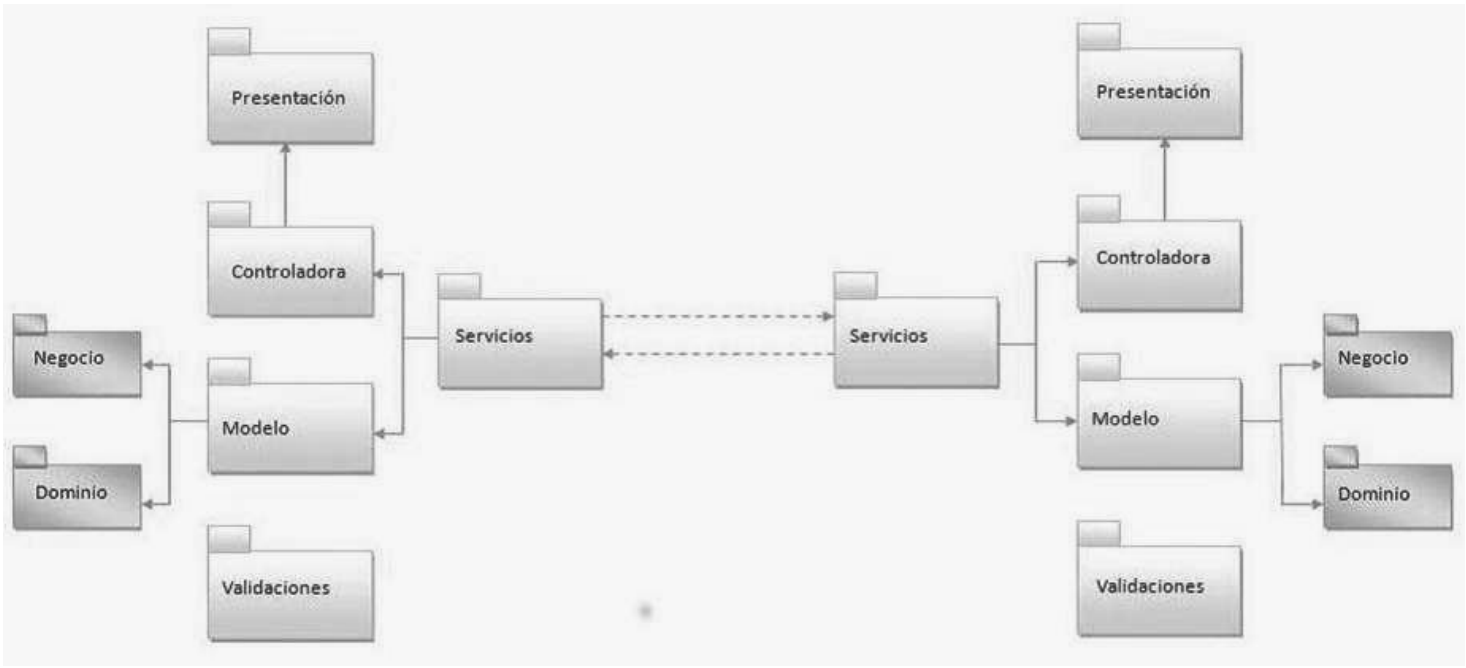


Figura 6 Vista Vertical de la Arquitectura.

El componente asume además características horizontales en la arquitectura del sistema, implementando patrones de integración para mejorar la calidad del diseño arquitectónico y aplicando los

patrones GRAPS a los componentes creados, con el objetivo de eliminar lazos de dependencia funcional entre los mismos.

2.3.3. Híbrido

La solución definitiva se puede clasificar como híbrida debido a la participación de los dos modelos anteriores.

Dicha característica combina las ventajas de los modelos explicados con anterioridad permitiendo el desarrollo en paralelo entre las capas y a la vez el encapsulamiento[20], además mejora la flexibilidad de la aplicación al permitir que se añadan nuevos módulos y de ésta manera nuevas funcionalidades incidiendo positivamente en el mantenimiento y el soporte al ser más sencillo cambiar un módulo que el sistema completo.

2.4. Patrones

Dentro de los patrones utilizados en la aplicación, se encuentran los GRASP de ellos se utilizaron:

Experto: se evidencia cuando las clases controladoras delegan la responsabilidad de ejecutar determinadas acciones a las clases del modelo cuya información es más adecuada para la resolución del problema en cuestión.

Creador: es adaptable a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos.

Controlador: se pone de manifiesto a través del uso de una clase controladora que es la que administra el flujo de acciones invocadas desde la interfaz de usuario y en cada una de ellas asocia la lógica del negocio al modelo responsable. En el modelo, donde está implementada la lógica del negocio, las clases del modelo pueden instanciar otras clases en dependencia de la información que se requiera.

Bajo acoplamiento: en el Modelo de Datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

[20] El encapsulamiento es un mecanismo que consiste en organizar datos y métodos de una estructura, conciliando el modo en que el objeto se implementa, es decir, evitando el acceso a datos por cualquier otro medio distinto a los especificados. Por lo tanto, la encapsulación garantiza la integridad de los datos que contiene un objeto

Alta cohesión: en el componente existe afinidad entre cada clase y los métodos[21] que implementan, éstas poseen responsabilidades vinculadas acordes a la información que manejan y colaboran con otras clases para compartir el esfuerzo si la tarea es grande, facilitando su mantenimiento y reutilización. El patrón se evidencia también en el diseño de clases al darle a cada clase persistente la funcionalidad que necesita.

De entre los patrones GOF se usó además el patrón Cadena de Responsabilidad que entra en la clasificación de patrón de comportamiento, el mismo se utiliza en el tratamiento de excepciones. Un ejemplo de su uso es cuando se produce un error al insertar en la base de datos, el cual es captado por las capas superiores, reenviando la excepción hasta la capa de aplicación donde se traduce al lenguaje del usuario.

2.5. Estándares de código

Los estándares de codificación en el marco del proyecto CEDRUX van a permitir una mejor integración entre las líneas de producción y se establecerán las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que permita un mejor entendimiento por parte del equipo de desarrollo y de soporte, el componente se rige por estos estándares, pues son una guía para el desarrollo y desde el punto de vista arquitectónico estandarizan el código a implementar.

Dentro de los estándares utilizados se encuentran los siguientes:

- ✓ **Notación Húngara:** se utilizó en la definición de las variables.
  Ej. intCantidad : como se observa la variable es de tipo entero y representa una cantidad.

Tabla 4. Prefijos a utilizar en la creación de las variables.

Tipo de Datos	Prefijos
Arreglos	Arr
Objetos	Obj
Enteros	Int
Cadena	str
Boolean	bool

- ✓ **Notación PascalCasing:** se usó para los nombres de las clases del modelo.

Ejemplo de clase del negocio: DatEstado

[21] Algoritmos asociados a un objeto, cuyas ejecuciones se desencadenan tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, son lo que el objeto puede hacer.

Ejemplo de clase del dominio: DatEstadoModel

- ✓ **Notación CamelCasing:** se usó en los nombres de los atributos y de las funciones.

Ej. idEstructura
adicionarEstado()

- ✓ **Nomenclatura para las constantes:** En caso de ser una constante el nombre entero de la variable se escribirá con mayúscula.

Ej. ESTADOINICIAL

2.6. Descripción de las clases y las funcionalidades necesarias

2.6.1. Clases Controladoras

Tabla 5. Descripción de la clase controladora Gestionarestado.

Nombre: GestionarEstado	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
cargarEstado	Devolver todos los estados.
adicionarEstado	Insertar un estado.
modificarEstado	Actualizar un estado.
eliminarEstado	Eliminar un estado.

Tabla 6. Descripción de la clase controladora Gestiontipodocumento.

Nombre: GestionTipoDocumento	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
cargarSub	Cargar los subsistemas.
cargarSubMod	Cargar los subsistemas asociados a un documento.
cargarTDoc	Cargar todos los tipos de documento.
eliminarTDoc	Eliminar un tipo documento.
adicionarTDoc	Adicionar un tipo de documento asociado a un subsistema o varios
modificarTDoc	Modificar un tipo de documento.
cargarEstadoAsoc	Cargar los estados asociados al tipo documento.
cargarEstadoSAsoc	Cargar los estados sin asociar al tipo documento.
eliminarEstaAsoc	Eliminar la asociación de un estado con un tipo documento.
asociarEstadoTDoc	Asociar un estado a un tipo documento.
modificarEstadoInicial	***Establecer un estado inicial al tipo documento.***

Tabla 7. Descripción de la clase controladora Gestionarflujodetrabajo.

Nombre: GestionarFlujoDeTrabajo	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
cargarGridConfFlujo	Cargar tipos de documentos asociados a un subsistema.
cargarGridTrans	Cargar las transiciones asociadas tipo de documento.
cargarCEstados	Cargar los estados iniciales y finales posibles para el tipo documento.
adicionarTrans	Adicionar una transición al tipo de documento.
cargarSubsistemas	Cargar subsistemas.
obtenerServicios	Cargar las pre-condiciones y pos-condiciones asociadas al subsistema.
cargarXML	Mapear el xml del IoC
obtenerUsuarios	Obtener los posibles usuarios asociados al subsistema
cargarServiciosModTrans	Cargar los servicios de la transición seleccionada.
obtenerUsuariosModTrans	Obtener los usuarios de la transición seleccionada.
modificarTrans	Modificar la transición.
eliminarTrans	Eliminar la transición.

2.6.2. Clases Auxiliares

Tabla 8. Descripción de la clase auxiliar TemAprobacion.

Nombre: TempAprobacion	
Tipo de clase: Auxiliar	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
usuariosSimultaneos	Devolver los usuarios simultáneos dada una transición.
eliminarUsuariosTransicion	Elimina los usuarios simultáneos que se encuentran en espera en la BD.
existeUserSimultaneo	Devolver si el usuario se encuentra en espera dado su id, el id del tipo de documento y el id de la transición.
eliminarTempUser	Elimina un usuario simultáneo en específico de la tabla temporal dado su id, el id del tipo de documento y el id de la transición en que se encuentra involucrado.

Tabla 9. Descripción de la clase auxiliar UtilidadesModel.

Nombre: UtilidadesModel	
Tipo de clase: Auxiliar	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
eliminarTrans	Eliminar una transición.
ejecutarWorkflow	Ejecutar el flujo de trabajo a partir de un id de usuario, un

	id de estado inicial, un id de estado final, un id de tipo de documento, un objeto y un id de subsistema.
precondiciones	Ejecutar las pre-condiciones a partir de un id de transición, un objeto y un id de subsistema.
poscondiciones	Ejecutar las post-condiciones a partir de un id de tipo de documento, un id de estado inicial, un id de estado final, un id de usuario, un objeto y un id de subsistema.
eliminarUserTemp	Eliminar un usuario de la tabla temporal a partir de un id de usuario, un id de transición y un id de tipo de documento.

2.6.3. Clases Entidades

Tabla 10. Descripción de la clase entidad ConfDoc.

Nombre: ConfDoc	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
getTodos	Devolver todos los documentos dado el id del subsistema en que se encuentra.
getPorLimite	Devolver una cantidad limitada de documentos dado el id del subsistema en que se encuentra.
getIDTDoc	Devolver el último documento insertado.

Tabla 11. Descripción de la clase entidad DatCondicion.

Nombre: DatCondicion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
getTodos	Devolver todas las condiciones.
getPorLimite	Devolver una cantidad limitada de condiciones
getCondiciones	Devolver todas las condiciones asociadas a una transición dado su id.
eliminarCondiciones	Eliminar todas las condiciones asociadas a una transición dada su id.
getCondicionesPrePos	Devolver las condiciones asociadas a una transición dada su id y el tipo.

Tabla 12. Descripción de la clase entidad DatEstado.

Nombre: DatEstado
Tipo de clase: Entidad

Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
getTodos	Devolver todos los estados.
getPorLimiteEstado	Devolver una cantidad limitada de Estados
cargarEstadosCombo	Cargar los estados asociado a un tipo de documento
estadosVer	Devolver los estados en los que se puede ver un tipo de documento dado el id del subsistema en que se encuentra, el id del usuario y el id del tipo de documento.
estadoPorDefecto	Devolver el estado por defecto del documento dado el id del subsistema en que se encuentra, el id del usuario y el id del tipo de documento.
estadosMod	Devolver el los estados a los que se puede modificar el tipo de documento dado el id del subsistema en que se encuentra, el id del usuario, el id del tipo de documento y el id del estado en que se encuantra.

Tabla 13. Descripción de la clase entidad DatSubsistema.

Nombre: DatSubsistema	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
getTodos	Devolver todos los subsistemas

Tabla 14. Descripción de la clase entidad DatTransicion.

Nombre: DatTransicion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción**.
getTodos	Devolver todas las transiciones.
cargarTransicionTDoc	Cargar las transiciones asociadas a un tipo de documento.
getIDTrans	Devuelve la última transición adicionada.
existeTrans	Verificar si existe una transición dado su id de estado inicial, su id de estado final y su id de tipo de documento asociada.
dameEstados	Devolver todos los estados asociados a una transición dado su id.
dameCantTrans	Devolver la cantidad de transiciones que tienen asociado un estado dado su id.
noExisteTransicionesConDoc	Verificar si no existen transiciones asociadas a un

	documento específico dado su id.
transicionId	Devolver el id de la transición dado su id de documento, su id de estado inicial, su id de estado final y el id de usuario asociado.
usuariosSimultaneos	Devolver los usuarios simultáneos de una transición.

2.7. Estrategia de integración

La propuesta utilizada por el proyecto ERP para realizar la interacción entre los módulos que componen el Sistema CEDRUX es la utilización del IoC, esta interacción se realiza mediante la llamada de diferentes servicios los cuales se encuentran implementados en los diferentes módulos y son invocados de forma interna al propio módulo o de forma externa cuando se solicita un servicio que se encuentra fuera del alcance del propio módulo. Su principal ventaja es que permite reducir el acoplamiento entre una clase y las clases de las cuales depende (26)

Dentro de los principales servicios que brinda el componente están:

Tabla 15. Principales Servicios que brinda el componente WorkFlow.

Servicio	Descripción	Parámetros
EjecutarPostCondiciones ()	Servicio que ejecuta las post-condiciones de un cambio de transición.	Identificador del usuario, identificador del subsistema, identificador del Tipo Documento, identificador del estado actual, identificador del estado al que desea llegar.
ejecutarWorkFlow()	Servicio que ejecuta la transición de un estado a otro de un documento.	Identificador del usuario, identificador del subsistema, identificador del Tipo Documento, identificador del estado actual, identificador del estado al que desea llegar.
EstadosVerDoc()	Servicio que devuelve los estados en que un usuario puede ver un documento.	Identificador del usuario, identificador del subsistema, identificador del Tipo Documento.
EstadoPorDefecto()	Servicio que devuelve el estado por defecto de un documento en caso de que el usuario pueda verlo, en caso contrario devuelve un mensaje informativo.	Identificador del usuario, identificador del subsistema, identificador del Tipo Documento.
EstadosModDoc()	Servicio que devuelve los estados a los que se puede modificar un documento teniendo en cuenta el estado en que se encuentra y el usuario.	Identificador del usuario, identificador del subsistema, identificador del Tipo Documento, identificador del estado en que se encuentra.
DenomEstado()	Servicio que devuelve la denominación de un estado.	Identificador del estado.

2.8. Modelo de Datos

Un modelo de datos es un lenguaje encaminado a describir las estructuras de datos, las restricciones de integridad y las operaciones de manipulación de los datos; orientado a resolver el problema y que describa los elementos de la realidad que intervienen en el mismo.

El modelo propuesto consta de 11 tablas, para su elaboración se tuvo en cuenta la reducción a la mínima expresión de los campos nulos.

Teniendo en cuenta el requisito funcional Gestionar Estados se crea la tabla dat_estado, en la misma se guardan los id de cada uno de los estados y una denominación del mismo.

Según el requisito funcional Gestionar Tipo de Documento se especificaron las tablas dat_subsistema y la tabla conf_doc, dichas tablas se relacionan con dat_subsistema_conf_doc, la cual se utiliza para identificar los documentos asociados a un subsistema.

Entre las principales tablas del modelo se encuentra dat_transición que es la encargada de guardar la información referente al estado inicial y final por el que puede transitar un documento, la tabla dat_condición que guarda la referencia al servicio que se ejecutará como post o pre-condición y la tabla temp_aprobación que se encargará de guardar los usuarios que simultáneamente estén solicitando un cambio de transición mientras la misma no se haya ejecutado, garantizando de ésta forma que se cumpla el patrón de WorkFlow Sincronización; todas éstas tablas tributan al requisito Gestionar Flujo de Trabajo así como las que surgen de las relaciones M-M entre ellas.

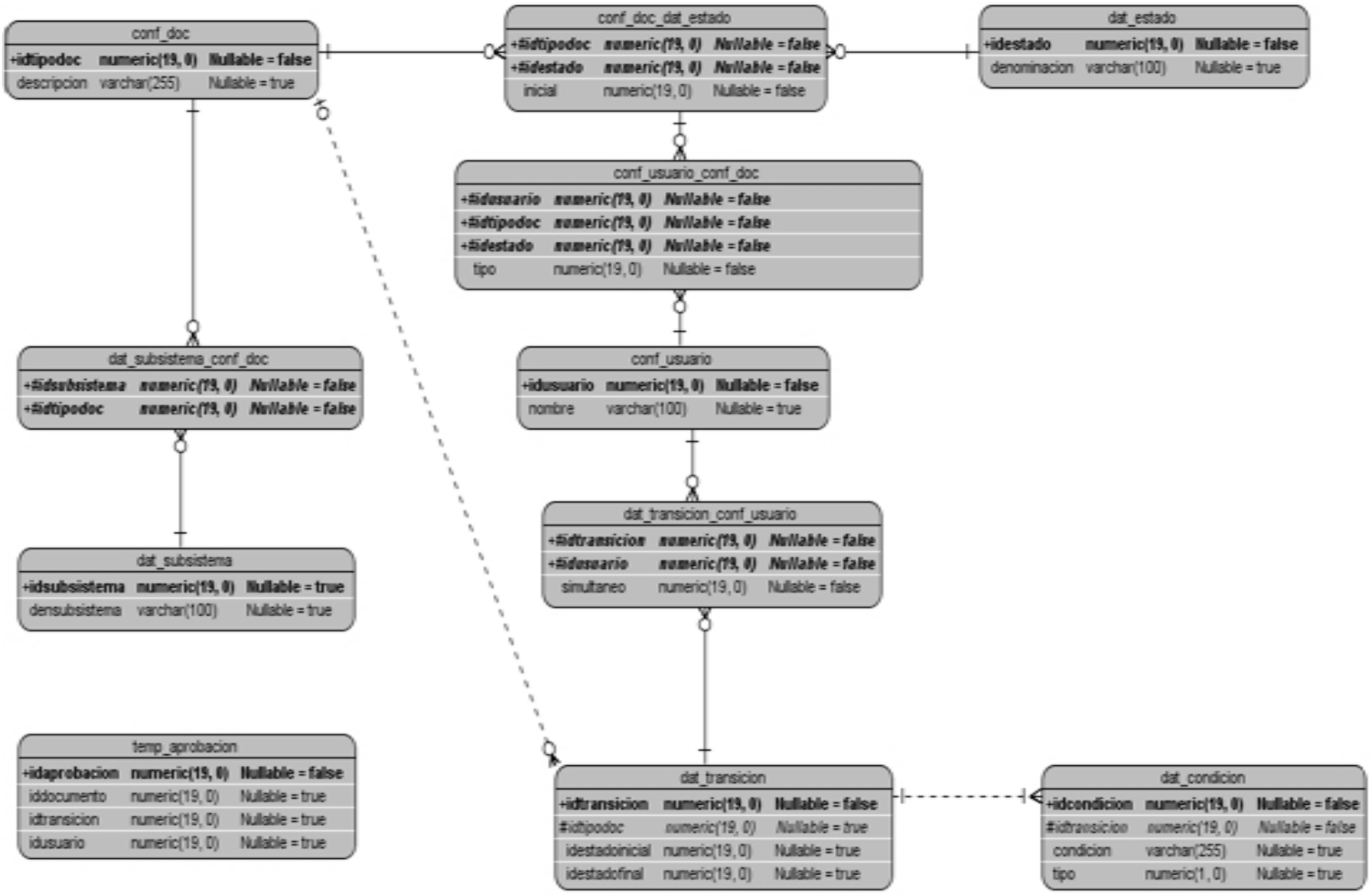


Figura 7. Modelo Entidad-Relación del componente WorkFlow

Conclusiones

- ✓ Se elaboraron los artefactos Modelo Entidad-Relación y los Diagramas de Clases de Diseño, así como las descripciones de las clases teniendo en cuenta el marco de trabajo del proyecto y los patrones de software y de WorkFlow.
- ✓ Se definió la estrategia de integración del componente con el sistema y se crearon los principales servicios que brinda el componente.

3. CAPÍTULO III VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

En el desarrollo del software, las posibilidades de error son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso inadecuado de las estructuras de datos, errores al enlazar módulos, entre otras. Para resolver este problema, se incluyó a nivel metódico un nuevo proceso en la confección de los sistemas informáticos: el proceso de prueba. “El testing puede probar la presencia de errores pero no la ausencia de ellos.”(Edsger Dijkstra)

3.1. Pruebas de Software

Las pruebas de software, son los procesos que permiten verificar y revelar la calidad de un producto de software, son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa.

“Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces. La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.” (27)

La fase de pruebas añade al producto final el valor para afirmar que ya se ha alcanzado la calidad requerida. Un gran porcentaje de los programas que se desarrollan tienen errores y es en la fase de pruebas donde se descubren, esa es la importancia fundamental de esta etapa, el objetivo específico de la fase de pruebas es encontrar el mayor número de errores.

3.2. Pruebas de Caja Blanca o Estructurales.

Se denomina cajas blancas a un tipo de prueba de software que se realiza 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. Las pruebas de caja blanca 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:

- ✓ **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.

Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. 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.

3.2.1. Realización de la Prueba del Camino Básico.

El algoritmo a analizar es el que se presenta a continuación:

```
function modificarTDocAction() {
    $idtdoc = $this->_request->getPost('idtdoc');//1
    $denom = $this->_request->getPost('denomtdoc');//1
    $arrSub = json_decode(stripslashes($this->_request->getPost('arrSub')));//1
    $datdoc = Doctrine::getTable('ConfDoc')->find($idtdoc);//1
    $datdoc->denominacion = $denom;//1
    if($this->model[1]->Actualizar($datdoc)) { //2
        if($this->model[8]->EliminarSub($idtdoc)) { //3
            foreach ($arrSub as $index => $row) { //4
                $tdocsub = new DatSubsistemaConfDoc(); //5
                $tdocsub->idsubsistema = $row->idsubmod; //5
                $tdocsub->idtipodoc = $idtdoc; //5
                $this->model[7]->Insertar($tdocsub); //5
            } //6
            echo("{'codMsg':1,mensaje:'El tipo documento se actualizo correctamente'}"); //6
        } //7
    }
    else { //8
        echo("{'codMsg':2,mensaje:'No se pudo actualizar el tipo documento correctamente'}");
    } //9
} //10
```

Figura 8. Representación del algoritmo modificarTDocAction()

Después de este paso, es necesario representar el grafo de flujo asociado, en el cual se representan distintos componentes como es el caso de:

Nodo: Son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: Son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

Regiones: Son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

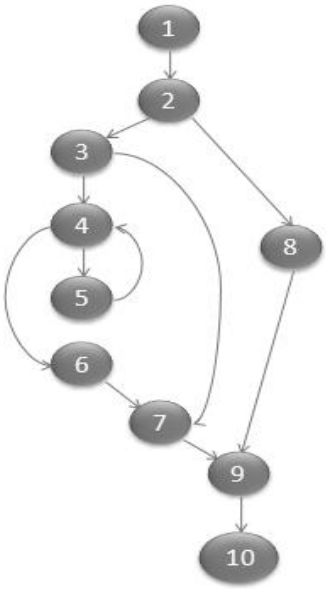


Figura 9. Grafo de Flujo asociado al algoritmo modificarTDocAction()

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías o fórmulas que para concluir que el cálculo fue correcto es necesario que por las tres vías el resultado sea el mismo, las fórmulas para calcular son las siguientes:

1. V(G) = (A - N) + 2

Donde **"A"** es la cantidad de Aristas y **"N"** la cantidad de Nodos.

V(G) = (12 - 10) + 2

V(G) = 4

2. V(G) = P + 1

Siendo **"P"** la cantidad de Nodos predicados[22]

V(G) = 3 + 1

V(G) = 4

3. V(G) = R

Donde **"R"** representa la cantidad de regiones en el grafo.

V(G) = 4

[22] Nodos de los cuáles parten 2 o más aristas.

Como el cálculo por las 3 vías arroja el mismo resultado podemos concluir que el algoritmo anteriormente representado tiene una complejidad ciclomática de 3. Lo que significa que existen a lo sumo 3 caminos por el cuál recorrer el grafo. Éste valor representa además el número máximo de pruebas que se le pueden realizar el algoritmo.

Con los datos anteriores representamos los posibles caminos por los que se puede recorrer el grafo:

Camino Básico No. 1: 1 - 2 – 8 – 9 - 10

Camino Básico No. 2: 1 – 2 – 3 – 7 – 9 – 10

Camino Básico No. 3: 1 - 2 – 3 – 4 – 6 – 7 – 9 – 10

Camino Básico No. 4: 1 - 2 - 3 - 4 – 5 – 4 – 6 – 7 – 9 – 10

Luego de haber determinado los caminos básicos se procede a ejecutar los casos de prueba para el procedimiento, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Para realizar los casos de pruebas es necesario 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.

Entrada: Se muestran los parámetros que entran al procedimiento

Resultados Esperados: Se expone el resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico # 1:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del tipo de documento es un número entero, la denominación es una cadena de caracteres y el array de los id de los subsistemas.

Condición de ejecución: El id del tipo de documento es igual a 10, denominación es igual a “Planes” y el array de los id de subsistemas contiene [15, 10, 5].

Entrada: $idtdoc = 10, $denom=”Planes”, $arrSub=[15, 10, 5].

Resultados esperados: Se espera modificar un tipo de documento y asociarlo a uno o varios subsistemas.

No se pudo modificar el documento debido a que el id recibido por parámetros no existe.

Caso de prueba para el camino básico # 2:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del tipo de documento es un número entero, la denominación es una cadena de caracteres y el array de los id de los subsistemas.

Condición de ejecución: El id del tipo de documento es igual a 10, denominación es igual a “Planes” y el array de los id de subsistemas contiene [120, 12].

Entrada: $idtdoc = 0, $denom=”Planes”, $arrSub=[120, 12].

Resultados esperados: Se espera modificar un tipo de documento y asociarlo a uno o varios subsistemas.

Ocurrió un error al eliminar un subsistema asociado al documento que no existía.

Caso de prueba para el camino básico # 3:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del tipo de documento es un número entero, la denominación es una cadena de caracteres y el array de los id de los subsistemas.

Condición de ejecución: El id del tipo de documento es igual a 10, denominación es igual a “Planes” y el array de los id de subsistemas contiene [].

Entrada: $idtdoc = 0, $denom=”Planes”, $arrSub=[].

Resultados esperados: Se espera modificar un tipo de documento y asociarlo a uno o varios subsistemas.

No se pudo asociar los subsistemas al documento debido a que el array de subsistemas se encuentra vacío.

Caso de prueba para el camino básico # 4:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del tipo de documento es un número entero, la denominación es una cadena de caracteres y el array de los id de los subsistemas.

Condición de ejecución: El id del tipo de documento es igual a 10, denominación es igual a “Planes” y el array de los id de subsistemas contiene [15, 10, 5].

Entrada: $idtdoc = 0, $denom=”Planes”, $arrSub=[15, 10, 5].

Resultados esperados: Se espera modificar un tipo de documento y asociarlo a uno o varios subsistemas.

El documento fue modificado satisfactoriamente.

Teniendo en cuenta los resultados arrojados se puede concluir que el algoritmo modificarTDocAction() cumple se comporta en dependencia a los datos de entrada acorde a los esperado, siendo capaz de lanzar un evento que informe de un fallo en la ejecución de la acción que se trata de realizar en dependencia de la validez o no de los datos de entrada, a la vez que modifica satisfactoriamente un documento si los datos de entrada son los correctos; por lo que se puede decir que el algoritmo la prueba de camino básico arroja resultados positivos al aplicarla sobre el mismo.

3.2.2. Prueba de Rendimiento.

Para realizar la prueba se utilizó un software nombrado JMeter, el cual permitió medir uno de los indicadores más importante en el desempeño de una aplicación que es el rendimiento, permitiendo realizar pruebas distribuidas en distintos ordenadores que actuarán como clientes. Se pueden realizar pruebas de estrés y se puede obtener muchas otras estadísticas del software, como por ejemplo la mediana, el muestreo, porciento de errores, entre otras.

Caso de prueba para probarlo en el JMeter

Descripción: Se accede a la aplicación Cedrux por el portal hasta abrir el componente WorkFlow y probar las funcionalidades de Gestionar Flujo Trabajo, Gestionar Estado, y Gestionar Tipo Documento.

Condición de ejecución:

- ✓ Deben estar corriendo los servidores de web y base de datos.
- ✓ Características de los servidores debe ser 1Gb de RAM.
- ✓ Características de la red 100Mbps.

Tabla 16. Resultados obtenidos al utilizar la herramienta JMeter.

ID del escenario	Escenarios de la sección	Carga de Trabajo	Descripción	Resultado esperado	Resultado de la prueba
EC 1: Gestionar Estado	EC 1.1 Adicionar Estado	50	En esta página se adicionan estados que se definan.	Respondido a una velocidad menor de 15 seg	11,5 seg
	EC 1.2 Modificar Estado	50	En esta página se modifican los estados que se definan.	Respondido a una velocidad menor de 15 seg	12,6 seg
	EC 1.3 Eliminar Estado	50	En esta página se eliminan estados que se definan.	Respondido a una velocidad menor de 15 seg	12,7 seg
EC 2 Gestionar	EC 2.1 Adicionar Tipo	50	En esta página se adicionan los Tipo	Respondido a una velocidad	4,7 seg

Tipo Documento	Documento		Documento definidos.	menor de 15 seg	
	EC 2.2 Modificar Tipo Documento	50	En esta página se modifican los Tipo Documento definidos.	Respondido a una velocidad menor de 15 seg	4,7 seg
	EC 2.3 Eliminar Tipo Documento	50	En esta página se eliminan los Tipo Documento definidos.	Respondido a una velocidad menor de 15 seg	4,8 seg
	EC 2.4 Asociar Estado al Tipo Documento	50	En esta página se asocia los estados a los Tipo Documento definidos.	Respondido a una velocidad menor de 15 seg	5,1 seg
	EC 2.5 Desasociar Estado al Tipo Documento	50	En esta página se desasocia los estados a los Tipo Documento definidos.	Respondido a una velocidad menor de 15 seg	5,1 seg
EC 3 Gestionar Flujo Trabajo	EC 3.1 Adicionar Transición	50	En esta página insertan las transiciones definidas.	Respondido a una velocidad menor de 15 seg	4,3 seg
	EC 3.2 Ejecutar WorkFlow	50	En esta página se ejecuta el workflow.	Respondido a una velocidad menor de 15 seg	7,2 seg
	EC 3.3 Ejecutar Post-condición	50	En esta página se ejecuta las Post-condiciones.	Respondido a una velocidad menor de 15 seg	6,7 seg

Teniendo en cuenta que los tiempos de respuesta del componente son menores que los tiempos esperados se puede concluir que para una muestra de 50 o menos usuarios simultáneamente conectados a la aplicación los mismos se encuentran dentro de los límites establecidos, por lo que se puede decir que el componente presenta un buen rendimiento.

3.3. Validación del modelo de diseño propuesto

Son varios los puntos de vista relacionados con la calidad del software. Desde metodologías hasta las distintas normas de calidad, que pueden estar orientados tanto a los procesos de desarrollo como a los productos de software. No es objetivo de este epígrafe abundar sobre los temas de calidad, pero si desarrollar una evaluación del diseño propuesto del componente Workflow del módulo de Configuración.

Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen entre otras medidas la cohesión, acoplamiento y complejidad, medidas que pueden ayudar al desarrollador de software a evaluar la calidad de un diseño a nivel de componente.

Atributos de calidad que se consideran:

- ✓ **Responsabilidad (Cohesión):** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ **Complejidad del mantenimiento:** Consiste en el grado de esfuerzo necesario para realizar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ✓ **Complejidad de implementación:** Consiste en el grado de dificultad que tiene que implementar un diseño de clases determinado.
- ✓ **Reutilización:** Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ **Acoplamiento:** Consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.
- ✓ **Cantidad de pruebas:** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas empleadas para evaluar la calidad del diseño de los componentes y su relación son las siguientes:

- ✓ **Tamaño operacional de clase (TOC):** Está dado por el número de métodos asignados a una clase. Los atributos que afecta son: la Responsabilidad, Complejidad de implementación y la Reutilización. De manera que mientras mayor sea el tamaño operacional de clase mayor será la Responsabilidad y Complejidad de implementación, mientras que su Reutilización disminuye.

Ver instrumentos y tabla de resultados en (Anexo 4 Instrumento de medición de la métrica Tamaño operacional de clase (TOC)).

Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:

Tabla 17. Resultados del instrumento de evaluación de la métrica T.O.C.

Atributos	T.O.C.(%)

Procedimientos	62 (tiene entre 1 - 4 Procedimientos)
Responsabilidad	62 (Baja)
Complejidad de Implementación	62 (Baja)
Reutilización	62 (Alta)

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del componente Workflow tienen una calidad aceptable teniendo en cuenta que el 62% de las clases incluidas en este componente posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones. Además el 62% de las clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad, Complejidad de implementación y Reutilización).

- ✓ **Relaciones entre clases (R.C.):** Esta dado por el número de relaciones de uso de una clase. Los atributos que afecta son el Acoplamiento, la Complejidad de mantenimiento, la Reutilización y la Cantidad de pruebas. De manera que mientras mayor sean las relaciones entre las clases, mayor será el Acoplamiento, la Complejidad de mantenimiento y la Cantidad de pruebas, mientras que su Reutilización disminuye.

Ver instrumentos y tabla de resultados en (Anexo 5 Instrumento de medición de la métrica Relación entre Clases (RC)).

Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:

Tabla 18. Resultados del instrumento de evaluación de la métrica R.C.

Atributos	RC (%)
Cantidad de Relaciones	85 (entre 1 – 4 Dependencias)
Acoplamiento	85 (Baja)
Complejidad de Mantenimiento	88 (Baja)
Cantidad de Pruebas	88 (Baja)
Reutilización	88 (Alta)

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del componente Workflow tienen una calidad aceptable teniendo en cuenta que el 85% de las clases incluidas en este componente poseen menos de 5 dependencias de otras clases. Además el 85% de las clases no poseen acoplamiento con otras, así como

los atributos de calidad Complejidad de mantenimiento, Cantidad de pruebas y Reutilización se comportan satisfactoriamente en un 88% de las clases.

Conclusiones

- ✓ Se realizó la prueba del Camino Básico concluyendo que el componente responde adecuadamente a entradas de datos erróneas y situaciones excepcionales, realizando además las acciones solicitadas sobre un juego de datos correcto.
- ✓ Se utilizó la herramienta JMeter para comprobar el rendimiento del componente bajo situaciones de carga de usuarios, donde los resultados arrojados en cuanto a tiempo de respuesta fueron relativamente bajos.
- ✓ Se evaluó el diseño de solución propuesto aplicando las métricas TOC y RC, donde los indicadores se mantuvieron en un rango de valores satisfactorio.

Conclusiones Generales

- ✓ Se realizó un estudio de los eventos y el flujo de los procesos del Sistema CEDRUX concluyendo que no permitían al mismo aumentar o modificar sus funcionalidades a un bajo costo.
- ✓ Se realizó una investigación profunda de los estándares para los software tipo ERP y se identificó como posible solución un SGFT capaz de gestionar los procesos comunes a los subsistemas.
- ✓ Se diseñó el componente WorkFlow basándose en las necesidades del proyecto y teniendo en cuenta el marco de trabajo propuesto por el equipo de arquitectura.
- ✓ Se implementó la solución teniendo en cuenta el uso de patrones de software y patrones de WorkFlow posibilitando que el componente fuera más eficiente y reutilizable.
- ✓ Se validó la propuesta de solución haciendo uso de la herramienta Jmeter la cual arrojó resultados positivos en cuento a rendimiento y se le realizaron pruebas de camino básico donde se evidenció que el componente se comporta dentro de los parámetros establecidos.

Recomendaciones

- ✓ Crear los servicios definidos como pre y post condiciones en cada módulo de manera que el componente pueda identificarlos.
- ✓ Integrar el componente con el resto de los módulos con el objetivo de ganar en estandarización.
- ✓ Continuar agregando funcionalidades que gestionen no sólo las transiciones entre los documentos sino también otros procesos comunes a cada módulo.
- ✓ Implementar mayor cantidad de patrones de Workflow con el objetivo de ganar en eficiencia.
- ✓ Realizar pruebas de conceptos al código del componente Workflow para así obtener un mejor rendimiento del mismo.

Referencias Bibliográficas

1. **Barros, Oscar.** *Reingeniería de Procesos de negocio.* Chile : Editorial Dolmen, 1994.
2. Kioskea Computing Comunity. [En línea] 2010. http://es.kioskea.net/contents/poo/heritage.php3ote.
3. **Workflow, Management Coalition Members.** Workflow Audit Data Specifications. Technical. [En línea] 1998. http://www.wfmc.org/.
4. —. Terminology & Glosary. Technical report WfMC-TC-1011, WfMC. [En línea] 1999. http://www.wfmc.org/.
5. **W3C.** Extensible Markup Language 1.0 (XML), REC-xml-19980210, version 1.0. [En línea] Febrero de 1998. http://www.w3.org/.
6. **Workflow, Management Coalition Members.** Workflow Standard-Interoperability. Internet e-mail Technical report WFMC-TC-1018, WfMC,MIME Binding. [En línea] Enero de 2000. http://www.wfmc.org/.
7. —. Workflow Client API Specifications (WAPI).Technical report WfMC-TC-1002, WfMC. [En línea] Julio de 1998. http://www.wfmc.org/.
8. —. Workflow Audit Data Specifications.Technical report WfMC-TC-1015, WfMC. [En línea] Septiembre de 1998. http://www.wfmc.org/.
9. —. Workflow Standard-Process Definition MetaModel Technical report WFMC-TC-1016, WfMC & WPDL. [En línea] Octubre de 1999. .http://www.wfmc.org.
10. **Georgakopoulos D., Hornick M., Sheth A.** *An Overview of Workflow Managenement: from Process Modeling to Workflow Automation Infraestructure, Distributed and Parallel Databases.* 1995.
11. **Patterns, Workflow.** [En línea] http://is.tm.tue.nl/research/patterns/.
12. **Prestedge, Jessica.** Workflow Patterns. [En línea] http://www.workflowpatterns.com/.
13. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico.* s.l. : McGraw-Hill Companies, 2002. ISBN: 8448132149.
14. **Systems Popkin, Software.** Modelado de Sistemas com UML. [En línea] 2008. http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.
15. **HtmlCastellano.** HtmlCastellano. [En línea] 2009. http://www.programacion.com/html.
16. **territoriopc.** territoriopc. [En línea] 2001. http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php..
17. **Javier, Eguíluz Pérez.** Introducción a XHTML. [En línea] http://www.librosweb.es/xhtml.
18. Introducción a CSS. [En línea] http://www.librosweb.es/css/..
19. **Celis, Ismael.** ESTADOBETA desarrollo web con estándares. Active Record. [En línea] http://www.estadobeta.com/2006/05/02/active-record/.
20. AbartiaTeam. [En línea] 2006. http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.
21. Tortoise SVN. *Tigris.org.* [En línea] 2009. [Citado el: 15 de Enero de 2010.] http://tortoisesvn.tigris.org/.
22. **MP3es.** Spket IDE. [En línea] 2009. http://wiki.mp3.es/Es/Spket_IDE.
23. Mozilla Firefox. [En línea] http://www.mozilla-europe.org/es/firefox/3.0/releasenotes/.
24. Osmosis Latina. [En línea] http://www.osmosislatina.com/jmeter/basico.htm.
25. **Somerville, Ian.** *Ingeniería de Software.* s.l. : Addison-Wesley, 2002. 9270-26-0206-8.
26. SOA. [En línea] 2006. [Citado el: 2 de 29 de 2010.] http://arquitecturaorientadaaservicios.blogspot.com/2006/03/pero-qu-es-realmente-soa.html.

27. **Marañón Gonzalo, Álvarez.** Características del lenguaje Java. 2009. [En línea] http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html.
28. **Vega, Yanet.** *L. S. Definición del ciclo de vida del proyecto.* Habana : s.n., 2009.
29. **Metodologiasdesistemas.** [En línea] 2007. [Citado el: 1 de 3 de 2010.] http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html.
30. **Almada, Federico.** [En línea] 22 de 1 de 2008. http://www.techtear.com/2008/01/22/zend-studio-for-eclipse-desarrollo-profesional-en-php.
31. **Doctrine.** Doctrine. [En línea] 2008. http://www.doctrine-project.org.
32. **Javier, Eguíluz Pérez.** librosweb.es. *Introducción a CSS.* [En línea] http://www.librosweb.es/css/.
33. **HTML.net.** Tutoriales sobre HTML y CSS - Construye tu propio sitio web. [En línea] http://es.html.net/tutorials/css/lesson1.asp.
34. **Coello, Costa Helkyn R. informatizate.** [En línea] Noviembre de 2002. http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_23082004.html.
35. Burbujas en .NET. IoC o el poder de ceder el control. [En línea] http://geeks.ms/blogs/etomas/archive/2008/10/28/ioc-o-el-poder-de-ceder-el-control.aspx.
36. **Larman, Craig.** *UML y patrones, Introducción al análisis y diseño orientado a objeto.* 1999.
37. territoriopc. [En línea] http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php.

Bibliografía

Introducción a JSON. (n.d.). Retrieved Enero 16, 2009, from http://www.json.org/json-es.html
JohannA. (n.d.). Modelo N-Capas. Retrieved Febrero 17, 2009, from http://n capas.blogspot.com/2008/11/modelo-n-capas_13.html
kynetia. (n.d.). Retrieved Abril 12, 2009, from Tipos de Pruebas: http://www.kynetia.es/calidad/tipos-de-pruebas.html
Leopoldo Magaña, C. (2005-2008). Zend Framework, una introducción. Retrieved Enero 29, 2009, from http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/
mozilla.org. (1998–2009). Retrieved Febrero 8, 2009, from http://www.mozilla.org/about/
PgAdmin III. (n.d.). Retrieved Enero 22, 2009, from http://guia-ubuntu.org/index.php?title=PgAdmin_III
php. (2001 - 2009). Retrieved Diciembre 19, 2008, from http://php.net/
Spket IDE. (2005 - 2008). Retrieved Febrero 6, 2009, from http://www.spket.com
TortoiseSVN. (n.d.). Retrieved Enero 11, 2009, from Un cliente de Subversion para Windows: https://forja.rediris.es/docman/view.php/123/117/TortoiseSVN-1.4.1-es.pdf
Zend Technologies. (2006 - 2009). Retrieved Marzo 23, 2009, from http://framework.zend.com/
Leopoldo Magaña, C. (2005-2008). Zend Framework, una introducción. Retrieved Enero 29, 2009, from http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/
maestros del web. (n.d.). Retrieved Enero 26, 2009, from ¿Qué es CSS?: http://www.maestrosdelweb.com/editorial/introcss/
Guía Breve de CSS. (n.d.). Retrieved Enero 23, 2009, from http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo
Guía Breve de Tecnologías XML. (1994-2005). Retrieved Enero 11, 2009, from http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML
El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha. (n.d.). Retrieved Diciembre 24, 2008, from 50. El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha. [Online] [Cihttp://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html.
Etapa de pruebas . (2007, Septiembre 16). Retrieved Febrero 22, 2009, from http://fabian-quintosemestre.blogspot.com/2007/09/etapa-de-pruebas.html
Ext JS. (n.d.). Retrieved Diciembre 19, 2008, from http://extjs.es/
Workflow Patterns http://www.workflowpatterns.com/patterns/data/routing/wdp38.php
Arquitectura de procesos para modelos de Workflow del Ing. Pablo Morales Lithium Software, Montevideo - Uruguay
Software para la automatización de procesos Administrativos http://www.workflow.com.mx/
Workflow management: models, methods, and systems http://books.google.com.cu/books?id=O1xW1_Za-I0C&printsec=frontcover&dq=workflow&source=bl&ots=I7eta3zkWT&sig=lRyy6nRsv4x_lGr62eqUT_fJ2EI&hl=es&ei=EpkGTLKeHoGC8gbVtITuCw&sa=X&oi=book_result&ct=result&resnum=11&ved=0CEMQ6AEwCg#v=onepage&q&f=false
Symantec http://www.symantec.com/es/mx/business/workflow-solution

ANEXOS

Anexo 1.

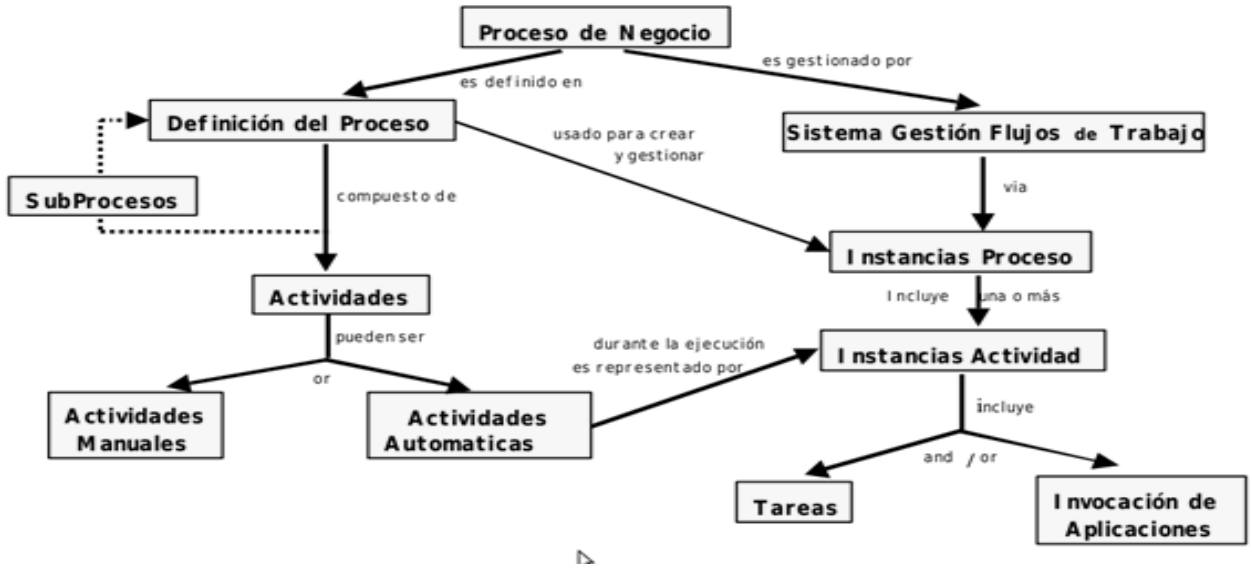


Figura 10. Conceptos básicos y la terminología asociada tanto para la fase de modelado como para la fase de ejecución de un Sistema de Gestión de Workflow.

Anexo 2.

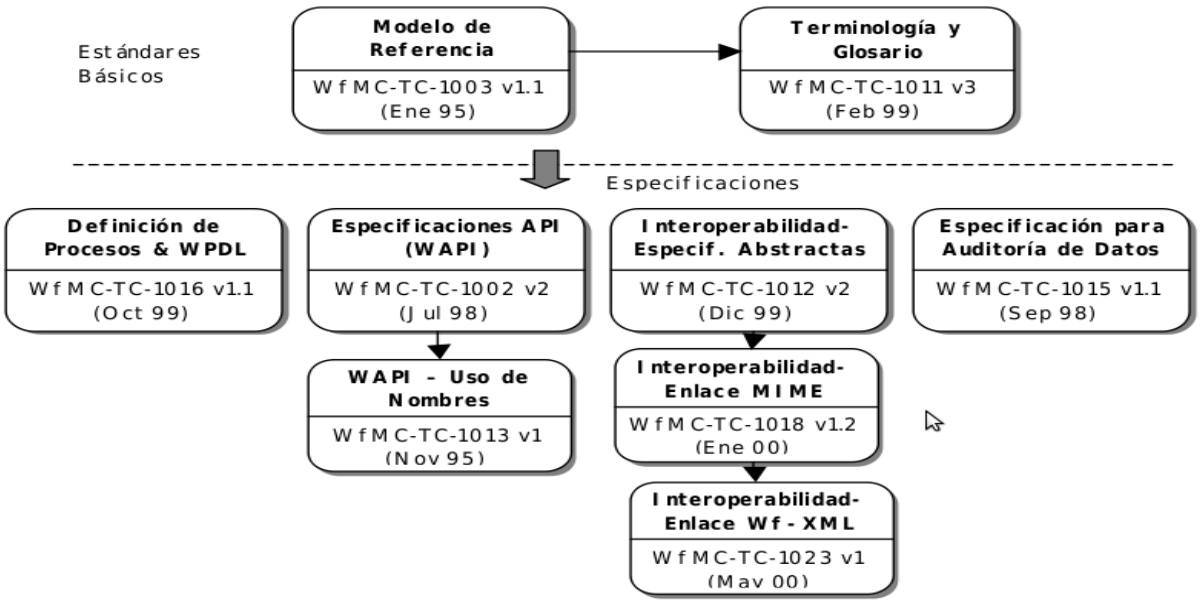


Figura 11. Estándares de la WFMC.

Anexo 3:

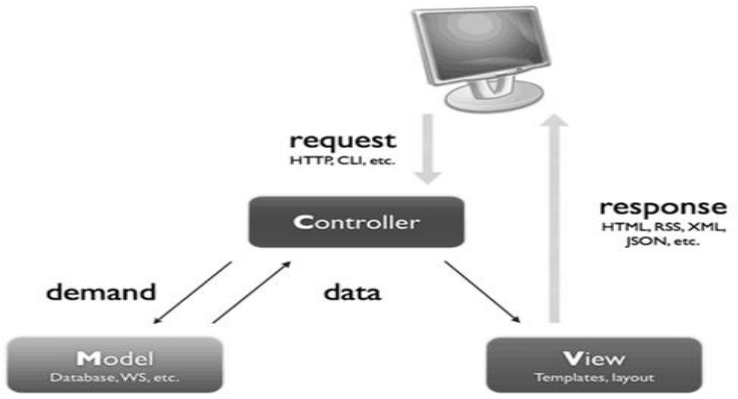


Figura 12. Estructura del patrón Modelo-Vista-Controlador.

Anexo 4: Instrumento de medición de la métrica Tamaño Operacional de Clase (TOC).

Tabla 19. Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC[23].

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.

Tabla 20. Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad.

No	Clases	Procedimientos	Responsabilidad	Complejidad	Reutilización
1	GestionarestadoController	4	Baja	Baja	Alta
2	GestionartipodocumentoController	11	Alta	Alta	Baja
3	GestionarflujodetrabajoController	12	Alta	Alta	Baja
4	UtilidadesModel	5	Baja	Baja	Alta
5	ConfDocModel	4	Baja	Baja	Alta
6	ConfDocDatEstadoModel	3	Baja	Baja	Alta
7	ConfUsuarioConfDocDatEstadoModel	3	Baja	Baja	Alta

[23] Responsabilidad, Complejidad de Implementación y Reutilización.

8	ConfUsuarioModel	3	Baja	Baja	Alta
9	DatCondicionModel	3	Baja	Baja	Alta
10	DatEstadoModel	3	Baja	Baja	Alta
11	DatSubsistemaConfDocModel	3	Baja	Baja	Alta
12	DatSubsistemaModel	5	Media	Media	Media
13	DatTransicionConfUsuarioModel	3	Baja	Baja	Alta
14	DatTransicionModel	3	Baja	Baja	Alta
15	TempAprobacionModel	3	Baja	Baja	Alta
16	ConfDoc	3	Baja	Baja	Alta
17	ConfDocDatEstado	5	Media	Media	Media
18	ConfUsuario	4	Baja	Baja	Alta
19	ConfUsuarioConfDocDatEstado	6	Media	Media	Media
20	DatCondicion	5	Media	Media	Media
21	DatEstado	7	Media	Media	Media
22	DatSubsistema	2	Baja	Baja	Alta
23	DatSubsistemaConfDoc	3	Baja	Baja	Alta
24	DatTransicion	9	Alta	Alta	Baja
25	DatTransicionConfUsuario	5	Media	Media	Media
26	TempAprobacion	4	Baja	Baja	Alta

Anexo 5: Instrumento de medición de la métrica Relación entre Clases (RC).

Tabla 21. Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Bajo	1
	Medio	2
	Alto	>2
Complejidad del mantenimiento	Baja	< =Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Cantidad de pruebas unidad	Baja	< =Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Reutilización	Baja	> 2* Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	< =Promedio.

Tabla 22. Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización).

No	Clases	Cantidad de Relaciones	Acoplamiento	Complejidad de Mantenimiento	Cantidad de Pruebas	Reutilización
1	GestionarestadoController	5	Medio	Baja	Baja	Alta
2	GestionartipodocumentoController	10	Alta	Alta	Alta	Baja

3	GestionarflujodetrabajoController	14	Alta	Alta	Alta	Baja
4	UtilidadesModel	7	Alta	Alta	Alta	Baja
5	ConfDocModel	1	Baja	Baja	Baja	Alta
6	ConfDocDatEstadoModel	1	Baja	Baja	Baja	Alta
7	ConfUsuarioConfDocDatEstadoModel	1	Baja	Baja	Baja	Alta
8	ConfUsuarioModel	1	Baja	Baja	Baja	Alta
9	DatCondicionModel	1	Baja	Baja	Baja	Alta
10	DatEstadoModel	1	Baja	Baja	Baja	Alta
11	DatSubsistemaConfDocModel	1	Baja	Baja	Baja	Alta
12	DatSubsistemaModel	1	Baja	Baja	Baja	Alta
13	DatTransicionConfUsuarioModel	1	Baja	Baja	Baja	Alta
14	DatTransicionModel	1	Baja	Baja	Baja	Alta
15	TempAprobacionModel	1	Baja	Baja	Baja	Alta
16	ConfDoc	1	Baja	Baja	Baja	Alta
17	ConfDocDatEstado	1	Baja	Baja	Baja	Alta
18	ConfUsuario	1	Baja	Baja	Baja	Alta
19	ConfUsuarioConfDocDatEstado	1	Baja	Baja	Baja	Alta
20	DatCondicion	1	Baja	Baja	Baja	Alta
21	DatEstado	1	Baja	Baja	Baja	Alta
22	DatSubsistema	1	Baja	Baja	Baja	Alta
23	DatSubsistemaConfDoc	1	Baja	Baja	Baja	Alta
24	DatTransicion	1	Baja	Baja	Baja	Alta
25	DatTransicionConfUsuario	1	Baja	Baja	Baja	Alta
26	TempAprobacion	1	Baja	Baja	Baja	Alta