UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

CENTRO INTERNACIONAL DE POSGRADO

PROCEDIMIENTO DE REINGENIERÍA PARA INCREMENTAR LA MANTENIBILIDAD DE APLICACIONES WEB HEREDADAS

Trabajo final presentado en opción al título de

Máster en Informática Aplicada

Autor: Lic. Ladislau Faustino Banza Lutete

Tutor: Dra. C Ailyn Febles Estrada

Ciudad de La Habana, noviembre de 2015

Agradecimientos

A Dios por su infinito amor

A mi familia por me amaren

A la tutora Dra. Ailyn por el voto de confianza y su paciencia

A los profesores por la paciencia y empeño

A los amigos que siempre estuvieron pendientes por los avances de la tesis.

Declaración jurada de autoría

Declaro por este medio que yo Ladislau Faustino Banza Lutete, soy el autor principal del trabajo final de maestría titulado PROCEDIMIENTO DE REINGENIERÍA PARA INCREMENTAR LA MANTENIBILIDAD DE APLICACIONES WEB HEREDADAS, desarrollada como parte de la Maestría en Informática Aplicada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, se firma la presente declaración jurada de autoría en Ciudad de La Habana a los _____ días del mes de _____ del año _____.

Ladislau Faustino Banza Lutete

_____

Ailyn Febles Estrada

_____

Resumen

La mayoría de las aplicaciones web son caracterizadas por constantes operaciones de mantenimiento y evolución por su propia naturaleza y por los continuos cambios en los estándares y tecnologías web. Un alto nivel de mantenibilidad es necesario para que la aplicación se acomode a los continuos cambios que está sometido. No obstante, debido a los ciclos de desarrollos bastante rápidos que generalmente las aplicaciones Web son sometidas y las limitaciones de recursos, muchas aplicaciones Web son desarrolladas sin el cumplimiento en la práctica de los principios de ingeniería de software, y como consecuencia, en muchos casos, estas aplicaciones posen poca o inadecuada documentación y diseño difícil de entender, lo que torna difícil su mantenimiento y amplía la complejidad de su evolución. La reingeniería de software ha sido fuertemente utilizada para mejorar la mantenibilidad de las aplicaciones web. Este trabajo presenta un procedimiento de reingeniería para incrementar la mantenibilidad de aplicaciones web heredadas. El procedimiento ha sido aplicado en el Sistema Integrado de Bibliotecas de la universidad Agostinho Neto de Angola (SIB:UAN), en el cual se ha recuperado los modelos de diseño de la aplicación de acuerdo a la metodología UWE, y se restructuró la aplicación a la arquitectura MVC, lo que permitió incrementar la facilidad de mantenimiento de referido sistema.

Palabras claves: Aplicaciones Web, Mantenibilidad, reingeniería, UWE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En los últimos años, la World Wide Web, comúnmente conocida como la web, se ha difundido en varios contextos, desde la comunicación, educación, industria, gobierno y otras áreas que afectan nuestra vida social y laboral [1]. Paralelo a su expansión, la complejidad de las funcionalidades ofrecidas por las aplicaciones Web se ha ampliado significativamente, de pequeños servicios para el intercambio y desimanación de información, ofrecidos por los primeros sitios Web, a aplicaciones dinámicas y complejas que proporcionan a sus usuarios una gran variedad de funciones para la manipulación de datos, el acceso a bases de datos y otras operaciones [2] [3]. En la actualidad la mayoría de las aplicaciones Web (WebApps) conforman las características de un sistema complejo con considerable costo de mantenimiento [4]. Estas aplicaciones generalmente son desarrollados sobre arquitecturas multiniveles y suportadas por una variedad de estándares y tecnologías con alto nivel de interacción entre ellos, lo que propicia a menudo su constante evolución. Entre los factores que inciden sobre el costo del mantenimiento de las aplicaciones Web están las leyes de Lehman sobre la evolución del software, como la ley de la continuidad de cambio, donde un software en uso debe ser continuamente adaptado, en caso contrario, se tornara progresivamente insatisfactorio en el entorno en que opera, y la ley del incremento de la complejidad, donde a medida que el programa evoluciona, su estructura se torna progresivamente más compleja salvo que se haga un esfuerzo activo para evitar este fenómeno [5]. Adicionalmente, las aplicaciones Web poseen otras características que influyen igualmente en la complejidad y en el costo de su mantenimiento, como la heterogeneidad de componentes, los rápidos avances tecnológicos y la velocidad de evolución [6]. Un alto nivel de mantenibilidad es necesario para que las operaciones de mantenimiento y evolución se realicen con los mínimos costos posibles [7].

La mantenibilidad puede definirse, de acuerdo con [8], como la facilidad con el cual un software o componente puede ser modificado para corregir los errores, mejorar la performance u otro atributo del entorno, y se puede lograr mediante la adopción de un proceso de desarrollo disciplinado y una metodología que se adecue a las especificidades del software, bien como el uso de herramientas de desarrollo y el cumplimiento de un conjunto de buenas prácticas. Los principios y enfoques de Ingeniería Web brindan un potencial aporte para desarrollar aplicaciones Web bajo control, minimizar los riegos, y mejorar su calidad y mantenibilidad [9]. Sin embargo, en

muchos casos el desarrollo de aplicaciones Web se enfrenta con ciclos de desarrollos bastante rápidos asociados a técnicas de desarrollo ad-hoc[1]", y los referidos enfoques no son aplicadas en la práctica, lo que potencialmente resulta en sistemas con baja calidad y poca o inadecuada documentación [10] [11] [3]. Un estudio que dio seguimiento a seis aplicaciones Web durante cinco años encontró que "sólo el 40% de las aplicaciones seleccionadas al azar no exhibían anomalías o fallos" [12]. En la mayoría de estos casos el mantenimiento del software se vuelve difícil con elevados costos, por lo cual es sumamente importante incrementar la mantenibilidad del software para que se minimicen los costos del mantenimiento y se asegure su evolución [13] [14] [3].

El difícil mantenimiento es mencionada en la literatura frecuentemente como "problema de mantenimiento" y no se delimita únicamente a las aplicaciones web, si no, en todo software que ha sido desarrollado sin la observancia de los principios de ingeniería de software [15] [16]. La movilidad de personas presente en la organización según [8], constituye un factor adicional, a medida que puede provocar alguna falta de conocimiento directo del sistema existente y dificultar el mantenimiento por parte de las generaciones subsecuentes.

Según la literatura [3], uno de los métodos más aplicados para incrementar la mantenibilidad del software es la reingeniería, que puede ser entendida como el examen y la alteración de un sistema para reconstruirlo en una nueva forma y la subsiguiente implementación de esa forma [17]. El proceso de reingeniería básicamente incluye dos etapas fundamentales. La primera es la ingeniería inversa que permite obtener una descripción abstracta del sistema, y la segunda etapa consiste en un proceso de ingeniería directa para la reconstrucción del sistema.

La reingeniería de software tiene varias ventajas, entre ellas, la reducción del riesgo, teniendo en cuenta que el proceso se implementa sobre una aplicación que funciona y se conocen sus resultados y, por tanto, ya se dispone de una especificación del sistema, y la reducción del coste, que de acuerdo estudios realizados, puede alcanzar 75% [8] [18].

[1] Desarrollo ad hoc, se suele llamar al tipo de desarrollo que no es regido por un proceso disciplinado de ingeniería de software o sea la no observancia de los principios de ingeniería de software

No obstante, existen determinados inconvenientes que pueden perjudicar el éxito de un proceso de reingeniería, entre ellas, se destacan las siguientes de acuerdo con [19] [8], [20] [21]:

- Adopción de una estrategia/metodología de reingeniería impropia o incompleta al determinado contexto
- Ausencia de estandarización y automatización para facilitar la remodelación y reutilización.

En la actualidad, los proyectos de reingeniería integran los enfoques del paradigma dirigido por modelos (Model-Drive Software Development: MDSD) con principal objetivo de enfrentarse a los desafíos de automatización y estandarización del proceso. En 2007, el Object Management Group (OMG) ha propuesto la iniciativa de modernización dirigido por arquitectura (Architecture-Driven Modernization: ADM), que aboga por la aplicación de técnicas y herramientas propias del MDSD con el objetivo de formalizar y estandarizar los procesos de reingeniería dirigida por modelos [22].

En esta línea de investigación son escasos los trabajos de reingeniería para aplicaciones Web. En la literatura consultada se han encontrado los trabajos de Echeverría en [21], destinado a la modernización de WebApps heredadas a Aplicaciones enriquecidas (Rich Internet Applications*:* RIA), y la de Bernardi, y otros en [23], que define un modelo de proceso de reingeniería dirigido por modelos, destinado a reducir el esfuerzo del mantenimiento y evolución de las WebApps. En ambas las propuestas, aunque se integran los principios de MDSD, todavía subyace una dependencia tecnológica como en la mayoría de procesos de reingeniería dirigida por modelos, a medida que son generalmente concebidas para determinada categoría de aplicación Web, y se centran en un conjunto específico de aspectos, como la arquitectura y las tecnologías utilizadas en la implementación de referida aplicación, lo que limita su implementación en escenarios diferentes de los originalmente concebidos.

Normalmente la realización de proyectos de reingeniería requiere la definición de una estrategia propia que se adecue a las necesidades puntuales y específicas de determinada organización y de las características de los software candidatos al proceso [8] [18] [24].

El departamento de ciencias de la computación de la Universidad Agostinho Neto (UAN), donde el autor pertenece, es responsable para el desarrollo y mantenimiento de software a medida para la universidad. Varios proyectos de software, sobre todo aplicaciones Web

han sido desarrollados y contribuyen considerablemente para la mejora de servicios ofrecidos en la universidad (véase anexo 1). No obstante, las operaciones de mantenimiento y evolución en la mayoría de estas aplicaciones se realizan actualmente con considerable dificultad, lo que provoca en algunos casos el abandono de proyectos por parte del equipo de mantenimiento.

El diagnóstico realizado mediante observación, entrevistas y encuestas a los gestores de proyectos y miembros de equipos de mantenimiento del departamento (véase anexo 2 e 3), ha identificado que las referidas aplicaciones presentan una baja calidad en el diseño de la arquitectura caracterizado por falta de separación de los aspectos, lo que dificulta el mantenimiento y amplía la complejidad de su evolución. Otro factor se prende con la falta de documentación y especificaciones de diseño adecuada y actualizada, lo que influye de forma negativa en la comprensión de la estructura y funcionamiento de las aplicaciones existentes. Adicionalmente, la mayoría de los desarrolladores que inicialmente producirán gran parte de las referidas aplicaciones ya no se encuentran en la universidad, lo que constituye otro inconveniente para el personal nuevo en el departamento, a medida que no posee conocimiento sobre dichas aplicaciones.

Por otra parte, se constató que el departamento no cuenta actualmente con ninguna metodología o procedimiento para lidiar con las aplicaciones Web en las referidas condiciones. Comúnmente, las actividades de mantenimiento se realizan en dependencia del conocimiento que detén el mantenedor, y en la ausencia de personal que conoce la aplicación, se consume considerable tiempo y esfuerzo en la compresión del sistema.

En definitiva, el análisis del diagnóstico destaca como principales causas del difícil mantenimiento de las aplicaciones Web existentes en la universidad Agostinho Neto, las siguientes:

- Poca o ineducada documentación del diseño e baja calidad, sobre todo en la arquitectura de las aplicaciones Web existentes
- Movilidad de personal perteneciente a los equipos de desarrollo y mantenimiento de la universidad
- Ausencia de una estrategia que asegura la continuidad de proyectos después de eventual movilidad de personal en los equipos de proyectos de la universidad.

El escenario descripto lleva al planteamiento del siguiente **problema de investigación**: ¿Cómo incrementar la mantenibilidad de las aplicaciones Web existentes en la Universidad Agostinho Neto?

En este sentido se ha definido como **objeto de investigación**, mantenimiento y reingeniería de software. El **campo de la investigación** recaí sobre el mantenimiento de aplicaciones Web en el departamento de ciencias de la computación de la UAN.

El **objetivo general** de la presente investigación consiste en desarrollar un procedimiento de reingeniería de software dirigido por modelos para incrementar la mantenibilidad de las aplicaciones Web existentes en la Universidad Agostinho Neto.

Para dar respuesta a la problemática fue planteada la siguiente **hipótesis**: si se desarrolla un procedimiento de reingeniería de software dirigido por modelos que incluye la recuperación de la documentación del diseño y reestructuración de arquitectura de las aplicaciones Web existentes en la UAN, entonces se incrementará la mantenibilidad de dichas aplicaciones.

En el sentido de alcanzar el enunciado objetivo general y de acuerdo al resultado del análisis del problema de investigación, fueron delimitados los siguientes **objetivos específicos**:

1 Elaborar el marco teórico de la investigación relacionado con desarrollo, mantenimiento y reingeniería de aplicaciones Web.

2 Elaborar un procedimiento de reingeniería de software dirigido por modelos teniendo en cuenta las características de las aplicaciones Web existentes en la UAN

3 Aplicar el procedimiento propuesto a una aplicación Web existente en la UAN

4 Validar el procedimiento propuesto.

En la presente investigación han sido utilizados métodos científicos, entre los cuales se destacan los siguientes:

- Métodos generales: El método hipotético-deductivo para la elaboración de la hipótesis central de la investigación; el método sistémico para lograr que los elementos que forman parte del procedimiento funcione de manera armónica; el método histórico-lógico y el dialéctico para el estudio crítico de los trabajos

anteriores, y para utilizar estos como punto de referencia para el propósito del trabajo.

- Métodos lógicos: el método analítico-sintético para descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos de forma independiente, para luego sintetizarlos en la solución de la propuesta.

- Métodos empíricos: los métodos de entrevista y encuesta para obtener información sobre el mantenimiento y evolución de las aplicaciones web existentes en la UAN. El método experimental para comprobar la utilidad de los resultados obtenidos a partir del procedimiento propuesto.

Este trabajo ha contribuido con **aportes prácticos** que incluye el procedimiento reingeniería de software dirigido por modelos para el incremento de la mantenibilidad de las aplicaciones Web existentes en la UAN, además de la versión 2.0 del Sistema Integrado de Bibliotecas de la UAN (SIB.UAN), resultante de la aplicación práctica del procedimiento. Mientras que se considera como **aporte económico**, la reducción del esfuerzo en el mantenimiento del referido sistema.

El presente documento está estructurado en tres capítulos. El Capítulo 1 dedicado al marco teorico sobre las tematicas relaccionadas con el mantenimiento y reigeneiria de aplicaciones Web. En el Capítulo 2 se presenta la propuesta de procedimiento de reingeneiria para incrementar la mantenibiliad de las Aplicaciones Web heredadas, que ha sido elaborado teniendo en cuenta el escenario de desarrollo de aplicaciones Web en el departamento de ciencias de la computacion de la UAN. Por último, el Capítulo 3 presenta la apliccion del procedimineto propuesto en el Sistema Integrado de Bibliotecas de UAN con el objectivo de validar la propuestoa mediante la evalucion de mantenibilidad de la aplicación antes y despues del empleo del procedimiento.

CAPÍTULO 1. MARCO TEÓRICO

Introducción

En el capítulo se caracterizan las aplicaciones Web y se presenta el estado de arte sobre su desarrollo y mantenimiento. Se conceptualiza el proceso de reingeniería de software en general y el enfoque dirigido por modelos, y se platean los puntos de vista del autor sobre algunas propuestas de reingeniería relacionados con los aportes principales del trabajo.

1.1 Las aplicaciones Web: sus características principales y estado actual

Las aplicaciones Web poseen características específicas que les diferencian de las de más categorías de software. Comúnmente se suele definir como aplicaciones Web (WebApps), los software basados en las tecnologías y estándares de la Web, que proporcionan recursos específicos de Internet como los contenidos y servicios a través de una interfaz de usuario, el navegador Web. Entre las principales características de las WebApps, se destacan las siguientes [8]:

1. **Evolución continúa**. A diferencia del software tradicional, que evoluciona con una serie de versiones planificadas y cronológicamente espaciadas, las aplicaciones web están en constante evolución. No es inusual que algunas WebApps (específicamente, su contenido) se actualicen cada hora.
2. **Inmediatez**. Las aplicaciones basadas en Web tienen una inmediatez que no se encuentra en otros tipos de software. Es decir, el tiempo que se tarda en comercializar un sitio Web completo puede ser cuestión de días o semanas.

1.1.1 Categorías de aplicaciones Web

Existen varias clasificaciones sobre las categorías de aplicaciones web. De acuerdo con la clasificación propuesta por Tilley y Huang [25], se pueden distinguir tres clases de aplicaciones Web teniendo en cuenta el aumento de la complejidad:

- Clase 1: Son primariamente las aplicaciones estáticas desarrolladas en HTML, y sin interactividad con el usuario.

- Clase 2: proporcionan interacción con el cliente a partir de páginas HTML dinámicas (DHTML), asociando acciones de script con los eventos generados por el usuario (como el clic del ratón o pulsaciones de teclado)
- Clase 3: Las aplicaciones de contenido dinámico, en el cual sus páginas pueden ser creados en el momento de ejecución, en función de la interacción del usuario con la aplicación. Una aplicación de esta categoría se caracteriza por un gran número de tecnologías empleadas, tales como Java Server Pages (JSP), Java Servlets, PHP, CGI, XML, ODBC, JDBC, o tecnologías propietarias como Páginas Active Server de Microsoft (ASP).

Otra clasificación propuesta por Ginige y Murugesan en [26] se presenta en la Tabla.1.1.

Tabla 1.1: Categorías de las aplicaciones Web. Adaptado de [26]

Categoría	*Descripción*	*Ejemplos*	*Clase*
WebApp Informacional	colección jerárquica de documentos HTML que permiten la lectura de información	periódicos en línea, catálogos de producto, anuncios en línea	1
WebApp Interactiva	Permiten contenidos dinámicos en las páginas web, de ahí proveen a los usuarios información personalizada	Formularios de registro, presentación de información personalizada,	2
WebApp Transaccional	Incorporan suporte para almacenamiento persistente de datos, localización de información, control de concurrencia, gestión de fallos y de configuración	Compras electrónicas, ordenamiento de bienes y servicios, banca en línea	3
WebApp basada en el flujo de trabajo	Modelan procesos de negocios estructurados, flujos de actividad, reglas de negocios, infraestructura de alto rendimiento para el almacenaje de datos (gestión de contenidos).	Planificación en línea y sistemas de planificación, gerencia de inventario, supervisión de estado	3
WebApp Colaborativa	Ejecutadas por diferentes grupos de usuarios que acceden los recursos web en el sentido de realizar determinadas tareas	Sistemas de redacción distribuidos, herramientas de diseño colaborativo	2
WebApp Ubicuas	Accesibles en cualquier momento y en cualquier medio de comunicación. WebApps para variedad de plataformas, como teléfonos móviles, PDAs, ordenadores de escritorio, etc.	Grupos de chat, sistemas que recomiendan productos o servicios (recommender), mercados en línea	3
Portales web	Integran recursos (datos, aplicaciones, y servicios) de diferentes fuentes en un solo punto.	Centros comerciales electrónicos, intermediarios en línea	2

En la tabla descripta, la última columna fue añadida por el autor, con objetivo de presentar una clasificación conjunta que integra en las seis categorías propuestas en [26], las tres clases de la perspectiva de evolución de la web planteada en [25].

1.1.2 Arquitecturas utilizadas para el desarrollo Web

Según la guía SWEBOK [27], una arquitectura de software es el conjunto de estructuras necesarias para el sistema, que comprende los elementos de software, las relaciones entre ellos, y las propiedades de ambos. En el transcurrir de los años varias arquitecturas han sido propuestas para la implementación y despliegue de aplicaciones Web. La arquitectura cliente-servidor o arquitectura de dos capas es conocida como la arquitectura tradicional para las aplicaciones Web, y conceptualmente se puede entender como un modelo de aplicación distribuida donde el usuario interactúa con la aplicación a través de un navegador Web, lo que genera petición HTTP que se envía al servidor [28]. El servidor analiza la petición a través de la aplicación del servidor, que es un componente activo del servidor, responsable por la interpretación de la petición del cliente. Tras la interpretación de la petición se genera una respuesta que el servidor envía al navegador, y el navegador Web a su vez muestra el resultado al usuario. Esta arquitectura ha evolucionado a la arquitectura de multiniveles como se muestra en la Figura 1.1, donde la aplicación del servidor, durante la interpretación de la petición del cliente puede comunicarse con un servidor de base de datos o solicitar servicios a una tercera parte, como un servicio Web. De acuerdo a esta arquitectura pueden ser reconocidas en una aplicación web tres niveles o capas:

1. Capa de presentación: que se encarga de la interfaz de usuario
2. Capa de aplicación: contiene toda la lógica que modela los procesos de negocio y es donde se realiza todo el procesamiento necesario para atender a las peticiones del usuario.
3. Capa de administración de los datos y/o servicios, que es responsable de los intercambios de datos entre las aplicaciones y los terceros, tales como bases de datos o proveedores de servicios.

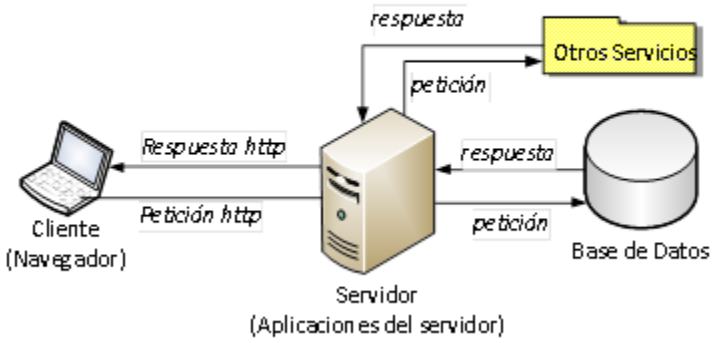

Figura 1.1: Arquitectura general de aplicaciones web

A menudo, es difícil separar estas capas, y en muchos casos, la capa de presentación incluye el código relacionado con la lógica de negocio. Estas situaciones deben ser evitadas, ya que hacen muy difícil el mantenimiento y evolución de las aplicaciones Web [29]. Por este motivo el patrón de diseño Modelo-Vista-Control (MVC) que trata de separar los componentes de un software en tres capas, ha sido ampliamente adaptado en el desarrollo de aplicaciones Web en el sentido de alcanzarse la debida separación entre las capas de la arquitectura.

1.1.2.1 El patrón Modelo Vista Controlador

El patrón de arquitectura Modelo-Vista-Control (Model View Control: MVC) es un concepto introducido por el inventor de Smalltalk[2] (Trygve Reenskaug) para encapsular datos junto con su tratamiento (modelo), y separarlo del proceso de manipulación (controlador) y presentación (Vista) [30]. La arquitectura sigue el enfoque básico de estratificación (capas o niveles). Las capas son una división lógica del código para funciones en diferentes clases. El enfoque de estratificación es bien conocido y ampliamente aceptado, y su principal ventaja es la reutilización de código [31].
La arquitectura MVC se divide en tres capas [32] :

1. El modelo: es la capa que se utiliza para gestionar la información y notificar a los observadores cuando se produjo un cambio de información. El modelo tiene acceso a los datos y las funciones asociadas al procesamiento de ellos.

[2] Smalltalk lenguaje de programación, actualmente extinto. (www.smalltalk.org)

2. La vista: es responsable para la presentación a un dispositivo, sirve de interface entre el usuario y el software. La vista está vinculada a un modelo, y presenta el contenido en la pantalla, de acuerdo a la actualización realizada por modelo.
3. El controlador: es el responsable por recibir la entrada del usuario y ordenar al modelo y la vista para realizar las acciones de acuerdo a la entrada.

Los tres niveles de la arquitectura están relacionados y en contacto constante. Internamente, ellos deben hacer referencia a unos de otros. La Figura 1.2 presenta un esquema de la comunicación entre las tres capas de la arquitectura.

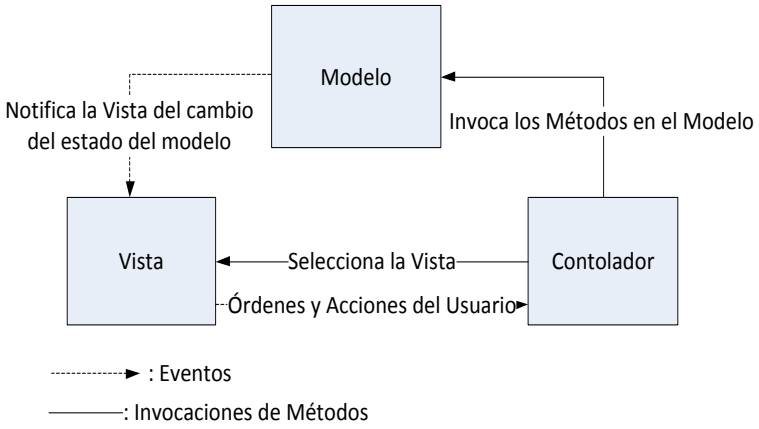

Figura 1.2: Esquema de comunicación en la arquitectura MVC. Adaptado de [33]

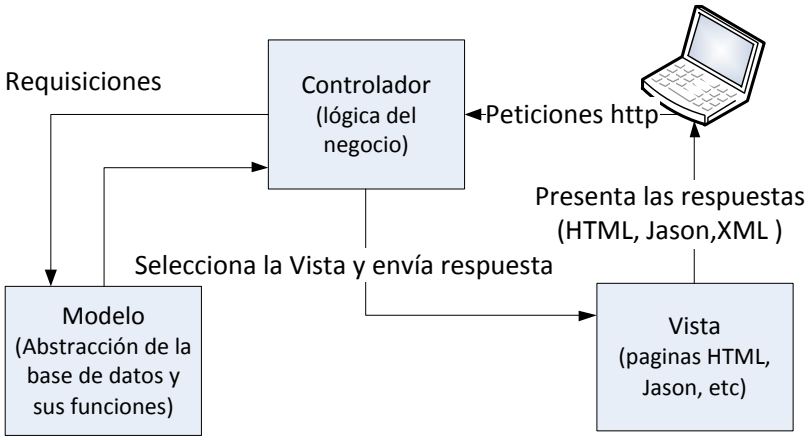

Figura 1.3: Esquema de la arquitectura MVC en aplicaciones web

Aunque originalmente este patrón ha sido desarrollado para software de escritorio, en los últimos años ha sido ampliamente adoptado como arquitectura para diseñar e implementar aplicaciones Web con objectivo de facilitar su mantenimiento y evolución a

través de la separación de los segmentos de código pertenecientes a cada capa de la arquitectura [29] [33]. La Figura 1.3 presenta un esquema básico de la utilización del patrón MVC en las aplicaciones Web, donde el controlador recibe la petición HTTP y actúa sobre el modelo y/o la vista convenientemente con base en la petición. Los resultados de la acción se devuelven al usuario en forma de página HTML u otro archivo de presentación mediante una respuesta HTTP a través de la vista. La clara separación de las capas de la arquitectura permite mayor comprensión de los elementos, lo que ayuda en el mantenimiento y evolución de cada elemento de la arquitectura sin afectar los otros elementos y viceversa.

1.1.3 Tecnologías para el desarrollo Web

La actual complejidad de las aplicaciones Web es alcanzada gracias a una variedad de tecnologías tanto para el lado del cliente como para del servidor. Las páginas de cliente son generalmente escritas alternando HTML con lenguaje de scripts (scripting) de cliente como JavaScript, pero se puede usar VBScript (solo para navegadores Internet Explorer o Google Chrome). JavaScript es un lenguaje sencillo, con una sintaxis similar a la sintaxis de Java, que ofrece un comportamiento dinámico a las páginas de los clientes, debido a que su comportamiento depende de la ejecución del script diferente del código HTML que es estático. Un lenguaje de scripts es un lenguaje de programación en el cual la mayoría de sus instrucciones son ejecutadas directamente, sin una previa compilación del programa en la máquina.

Para el lado del servidor de las WebApps generalmente se utilizan tecnologías como, PHP, JSP, Perl, Rubi, Python y ASP/ASP.NET. Estas tecnologías también son lenguajes de scripts, pero a diferencia de otros lenguajes de scripts como JavaScript, se ejecutan en el lado del servidor para generar páginas dinámicas, y permiten el acceso a bases de datos y otros servicios. Generalmente pertenecen a capa de aplicación (lógica de negocio) de la arquitectura de tres capas como se describió en el epígrafe 1.1.2.

1.1.4 Marcos de trabajo para el desarrollo de aplicaciones Web

Los marcos de trabajos para el desarrollo de software (Frameworks) definen un conjunto estandarizado de conceptos, prácticas y criterios que enfocan determinado problema, y sirven como referencia para enfrentar y resolver nuevos problemas similares. Un marco de trabajo para el desarrollo Web (Web Application Framework: WAF) está diseñado

para apoyar el desarrollo de aplicaciones Web. El objetivo de los WAF es aliviar la carga de actividades comúnmente ejecutadas en el desarrollo Web a partir de un conjunto de librerías que bridan soluciones probadas de referida actividades, tales como, el acceso a base de datos, las operaciones de administración comúnmente conocidas de “CRUD” (Create, Read, Update and delete: CRUD) y otras funcionalidades. La mayoría de los WAFs implementan el patrón de arquitectura MVC para separar el modelo de datos y las reglas de negocio de la interface del usuario. Esta práctica permite la modularización y reutilización de código, que contribuyen para el fácil mantenimiento y evolución de la aplicación.

En los últimos años han sido desarrollados una variedad de frameworks para el desarrollo Web cada vez más eficientes, entre ellos, Django, Zend Framework, Ruby on Rails y Symfony.

1.2 Ingeniería Web

La ingeniería Web surge por la necesidad de aplicarse un enfoque sistemático y disciplinado específico al desarrollo de aplicaciones Web, dado que los principios y técnicas de la ingeniería de software se mostraban insuficientes para la construcción de aplicaciones Web complejas con la calidad deseada y con el cumplimiento de los plazos y otras exigencias del cliente [34]. La Ingeniería Web (IWeb) está relacionado con el establecimiento y utilización de principios científicos, de ingeniería y de gestión, y con enfoques sistemáticos y disciplinados del éxito del desarrollo, despliegue y mantenimiento de sistemas y aplicaciones basados en Web de alta calidad [9]. La naturaleza de las WebApps torna el proceso de desarrollo Web una actividad difícil y desafiante a medida que debe considerar varios aspectos y requisitos, entre ellos, los continuos y frecuentes cambios tecnológicos, los problemas de usabilidad y seguridad, bien como la necesidad de proveer información para varios usuarios con diferentes habilidades [1]. Por estos motivos, generalmente se recomiendan un proceso interactivo e incremental, en el cual las fases del proceso son repetidas y refinadas hasta que la aplicación incorpore los requerimientos definidos. En la literatura se encuentran varias propuestas de marco de proceso para el desarrollo de aplicaciones Web, como los propuestos por [7], [35] y [1]. Básicamente, el proceso de desarrollo Web se inicia con una etapa de modelado que incluye el análisis y el diseño, con el objetivo de especificar los requisitos y describir la aplicación. La etapa de implementación que tiene que ver con la codificación. En seguida, tiene lugar la etapa de despliegue donde la aplicación es

configurada para su entorno operacional y entregue a sus usuarios finales. A partir de esta etapa se comienza un periodo de evaluación de la aplicación y consecuentemente tiene lugar la etapa de mantenimiento y evolución con el objetivo de incrementar los nuevos requisitos subyacentes al entorno en que opera, donde se vuelve a repetir las otras etapas.

1.2.1 Metodologías para el desarrollo de aplicaciones Web

En el dominio de las aplicaciones Web, debido a su propia naturaleza, un proceso de desarrollo debe considerar tres aspectos claves [36] [37]:

- Estructura o Contenido: define la organización de la informaciones a ser tratada por la aplicación, es decir, la representación de los datos y sus relaciones
- Navegación: Representa como las informaciones serán accedidas a través de la aplicación. Este aspecto está relacionado con la estructura de navegación y el comportamiento del sistema
- Presentación: Describe como las informaciones y el acceso a ellos serán presentados al usuario de la aplicación. Este aspecto está relacionado con la interface y el diseño de la presentación.

En este contexto, han sido propuestos varios métodos y técnicas específicos, capaces de capturar los referidos aspectos y brindar mayor eficiencia y disciplina en el proceso de desarrollo de aplicaciones Web. Entre estos métodos se destacan el OOHDM (Object Oriented Hypermedia Design Method) [38], WebML (Web Model Language: WebML) [39], W2000 [40], UWE (UML-based Web Engineering) [41] y UWA (Ubiquitous Web Applications design framework*)* [42]. La mayoría de estas metodologías integran los principios del paradigma de desarrollo dirigido por modelos (Model-Driven Software Development: MDSD) y forman parte de lo que se suele llamar ingeniería Web dirigido por modelos (Model-Driven Web Engineering: MDWE).

El MDWE es la aplicación del MDSD en el dominio del desarrollo de aplicaciones Web. Estas metodologías son dirigidas por modelos porque tratan los diferentes aspectos de las WebApps, mediante la utilización de modelos separados (por ejemplo: modelo de navegación, modelo de presentación, etc.), y tienen el soporte de compiladores de modelos que producen la mayor parte de las páginas web y la lógica de la aplicación basada en los modelos [43]. Entretanto, muchas de estas metodologías no explotan

completamente todos los beneficios del enfoque MDSD, y presentan algunas limitaciones, tales como [44] [45]:

- Falta de completa independencia de las plataformas: Estas metodologías comúnmente están ligadas a un particular estilo arquitectónico y una tecnología específica (JSP, ASP, PHP), y por lo tanto no permiten la construcción parametrizable de aplicaciones Web utilizando diferentes plataformas tecnológicas y estilos de arquitectura.
- Dificultad en describir los diferentes aspectos que intervienen en el desarrollo de aplicaciones: La mayoría de estas metodologías fueron originalmente concebidas para lidiar con específicas categorías de aplicaciones web (véase sección 1.1.1), y solo tratan un determinado conjunto de los aspectos relacionados con específica categoría de WebApps. Por consiguiente, son eficientes en modelar determinados aspectos, pero incapaces de modelar otros.
- Costos de adquisición y mantenimiento de las herramientas CASE que suportan las metodologías e Imposibilidad de intercambio de modelos entre diferentes herramientas: La mayoría de las metodologías poseen notaciones y herramientas propietarias. Esto obliga que los desarrolladores compren las herramientas de modelado (con los costos de aprendizaje) si quieren explotar mejor las ventajas de las metodologías. Peor aún, estas herramientas propietarias no interactúan con el resto de las herramientas utilizadas por el desarrollador, lo que le obliga a trabajar con todo un conjunto de entornos de desarrollo aislados, cada uno diferente y en muchos casos incompatible del resto.

Resolver estas limitaciones no es una tarea trivial. En la actualidad varias de las propuestas MDWE están evolucionado en el sentido de agregar de forma más eficiente los conceptos del enfoque del MDSD, sin embargo la eficiente integración de estas técnicas con la mayoría de las existentes metodologías está por resolverse [44].

1.3 Mantenimiento de software

Según el estándar IEEE 1219 [46] el mantenimiento del software, se define como la modificación de un producto software después de haber sido entregado a los usuarios o clientes con el fin de corregir defectos, mejorar el rendimiento u otros atributos, o adaptarlo a un cambio en el entorno. Similar definición es planteada por estándar

ISO/IEC 12207 [47], que la define como un producto software soporta una modificación en el código y su documentación asociada para la solución de un problema o por la necesidad de una mejora. Su objetivo es mejorar el software existente manteniendo su integridad.

El mantenimiento de software se puede clasificar en cuatro diferentes categorías [8] [16] [48]:

- Mantenimiento correctivo (corrección): consiste en corregir los defectos, cuando el software no conforma las especificaciones (por ejemplo, corregir errores descubiertos en la ejecución del software).
- Mantenimiento adaptativo (adaptación): se refiere a las modificaciones realizadas al software para acomodarlo a los cambios. Consiste en realizar ajustes cuando se registra un cambio en el entorno en que el software opera (por ejemplo, cuando se introduce una nueva versión del sistema operativo).
- Mantenimiento perfectivo (mejora): se refiere a las mejoras para incrementar la performance del producto, y consiste en añadir funciones al software, a fin de satisfacer las solicitudes del cliente/usuario que puede descubrir funciones adicionales. Este mantenimiento tiene lugar cuando los requisitos del software cambian (p. ej., cambio de los impuestos del software para reflejar nuevas leyes tributarias)
- Mantenimiento preventivo (prevención) o Reingeniería: consiste en hacer cambios en software a fin de que se pueda corregir, adaptar y mejorar más fácilmente. En este tipo de mantenimiento incide el enfoque de la presente tesis.

Las actividades de mantenimiento consumen entre 70 a 80 por ciento del esfuerzo y tiempo requerido para el desarrollo de software, mientras los restantes 20 a 30 por ciento son consumidos por otros etapas del ciclo de vida del software [49] [50]. De acuerdo con [6] y [51], entre los factores que inciden sobre el costo del mantenimiento del software figuran las dos de las leyes de Lehman sobre la evolución del software, que tienen particular incidencia sobre los software de categoría E-programs [5]:

- Continuidad de cambio*:* Un programa del tipo E (E-type program) en uso, debe ser continuamente adaptado, en caso contrario, se tornara progresivamente insatisfactorio en el entorno en que opera.

- Incremento de la complejidad: A medida que el programa evoluciona, su estructura se hará progresivamente más compleja salvo que se haga un esfuerzo activo para evitar este fenómeno

Adicionalmente a los factores descriptos, las aplicaciones web poseen otras características que influyen igualmente en la complejidad y en el coste de su mantenimiento, entre ellos, la heterogeneidad de componentes, los rápidos avances tecnológicos y la velocidad de evolución [4] [6]. A diferencia del software tradicional, una aplicación Web cambia y crece rápidamente en sus requerimientos durante su ciclo de vida [8]. La mayoría de las WebApps se caracterizan por continuas operaciones de mantenimiento y evolución a fin de incorporaren los nuevos requisitos que advienen del entorno evolutivo en que operan [14].

Un alto nivel de mantenibilidad es necesario para que las operaciones de mantenimiento y evolución se realicen con los mínimos costos posibles [7] [51]. La mantenibilidad puede definirse como la facilidad con el cual un software o componente puede ser modificado para corregir los errores, mejorar la performance u otro atributo del entorno [8]. Los métodos de ingeniería Web descriptos en el epígrafe 1.2.1, son fuertemente recomendados para que se logre alcanzar una mantenibilidad de mayor calidad en las aplicaciones Web [9].

1.3.1 Problema de mantenimiento de las aplicaciones Web

A pesar de existir varios métodos y técnicas capaces de soportar un proceso de desarrollo disciplinado y eficiente, en muchos casos, debido a los ciclos de desarrollos bastante rápidos que generalmente las aplicaciones Web son sometidas, los desarrolladores son inducidos a implementar directamente el código de aplicación sin el cumplimiento de un proceso disciplinado de desarrollo (el conocido ad-hoc), donde los referidos métodos no son aplicados en la práctica [3]. Adicionalmente, los desarrolladores son igualmente influenciados a no implementar principios de ingeniería importantes a la arquitectura de las WebApps como se planteó en el epígrafe 1.1.2, como la modularidad y la separación de los aspectos, debido a que la mayoría de las tecnologías utilizadas en las implementaciones de aplicaciones Web no alientan la aplicación de referidos principios, a medida que los scripts del servidor, como PHP, JSP, ASP, etc., son fácilmente mezclado con el código de la presentación, como, HTML, WML, JavaScript, etc.

Consecuentemente, el mantenimiento de las aplicaciones Web desarrolladas en esas condiciones se vuelve difícil con elevados costos y se amplía la complejidad de su evolución [10] [14]. Un estudio que dio seguimiento a seis aplicaciones web durante cinco años encontró que “sólo el 40% de las aplicaciones seleccionadas al azar no exhibían anomalías o fallos” [12]. En este escenario se torna sumamente importante incrementar la mantenibilidad del software para que se minimicen los costos del mantenimiento y se asegure su evolución [5] [3].

El difícil mantenimiento es mencionada en la literatura frecuentemente como “problema de mantenimiento” y no se delimita únicamente a las aplicaciones web, si no, en todo software que ha sido desarrollado sin la observancia de los principios de ingeniería de software [15] [16]. La movilidad de personas presente en la organización según [8], constituye un factor adicional, a medida que puede provocar alguna falta de conocimiento directo del sistema existente y dificultar el mantenimiento por parte de las generaciones subsecuentes.

Según la literatura [52] [53] [54] [3], uno de los métodos más aplicados para incrementar la mantenibilidad del software es la reingeniería, que se destina a rediseñar el software para tornarlo más mantenible y con mejor rendimiento.

1.4 Reingeniería de software

La reingeniería de software se destina a reconstruir un sistema existente en su totalidad o en parte con la finalidad de tornar fácil su mantenimiento y mejorar su calidad [8]. Existe en la literatura varias definiciones de reingeniería de software, entre ellas, la planteada por Chikofsky [17], que la define como el examen y la alteración de un sistema para reconstruirlo en una nueva forma y la subsiguiente implementación de esa forma. Arnold en [55], define la reingeniería como una actividad que mejora la compresión del software, o bien, lo prepara o mejora para incrementar su facilidad de mantenimiento, reutilización o evolución

En definitiva, la reingeniería de software consiste en realizar cambios en el software deficientemente diseñado e implementado a fin de que se pueda corregir, adaptar y mejorar fácilmente, reduciendo el costo y esfuerzo de su mantenimiento. La principal razón de la reingeniería radica en la existencia de software difícil de mantener o mejorar y que sin embargo necesita ser adaptado y extendido puesto que es crítico para el negocio de la organización en que opera. En muchos casos la necesidad de reingeniería

surge debido a los cambios subyacentes en el entorno del software, y. aunque el sistema sea realizado con las buenas técnicas de ingeniería de software, dicho sistema se tornara obsoleto en vista de nuevas tecnologías. Estos software se suelen llamar “sistema heredado” (legacy system), y necesitan ser rediseñados para que se adecuen a las nuevas tecnologías, tendencias y exigencias de los clientes, y aseguren su calidad en un mercado cada vez más competitivo [16] [56].

Entre los principales objetivos de la reingeniería de software, se destacan los siguientes [15]:

- Preparación para aumentar funcionalidades: En muchos casos la reingeniería es utilizada en la preparación del software para el proceso de mejora continua. La reingeniería permite especificar las características del sistema existente, que pueden ser comparadas con las especificaciones del sistema esperado. El sistema puede ser rediseñado para que se torne más fácil de mejorar.
- Mejorar la mantenibilidad: Los costos de mantenibilidad incrementan a medida que el sistema crece y evoluciona, porque los cambios se tornan difíciles y consumen mucho tiempo. Uno de los objetivos de reingeniería es rediseñar el sistema con apropiados módulos funcionales e interfaces explicitas, además de actualizar la documentación, lo que incrementa la mantenibilidad del software. En este enfoque se centra el propósito del presente trabajo.
- Migración: Debido a los continuos y rápidos avances en la tecnología, los software rápidamente se tornan obsoletos. Las organizaciones tienen cada vez más la necesidad de migrar sus sistemas para las nuevas plataformas de hardware, nuevos sistemas operativos o lenguajes.

El proceso de reingeniería de software, generalmente incluye una etapa de ingeniería inversa para la obtención de una descripción abstracta del sistema, seguida de un proceso de ingeniería directa o reestructuración. Este enfoque ha sido propuesto por Chikofsky en [17]. Otros autores como Pressman, conciben el proceso de reingeniería de software como una actividad de reconstrucción, en el cual se procede inicialmente una etapa de ingeniería inversa del software y en seguida se implementa la reconstrucción tanto a nivel de los datos y del código fuente. Mientras Somerville [16], especifica el proceso de reingeniería con las etapas de redocumentación, rediseño de la arquitectura del sistema, migración hacia un lenguaje de programación moderna, y

modificación y actualización de la estructura y valores de la base de datos respetivamente.

En líneas generales, el proceso de reingeniería de software comprende un conjunto de actividades que se destinan en la reutilización de un software heredado como base para la creación de otro más eficiente y fácil de mantener. El proceso conforma básicamente las etapas de ingeniería inversa, reestructuración y/o ingeniería directa como se presenta en la Figura 1.4.

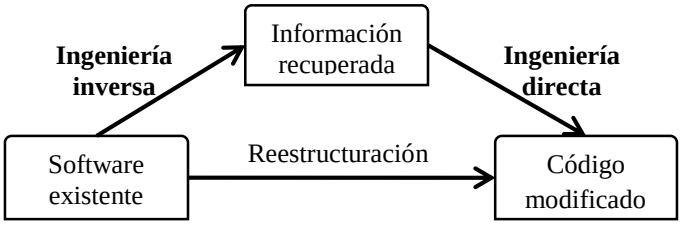

Figura 1.4: Modelo de proceso básico de reingeniería de software. Fuente: [57]

1. Ingeniería inversa: consiste en obtener una representación abstracta del sistema heredado.
2. Reestructuración: consiste en la trasformación de la representación abstracta a otra representación al mismo nivel de abstracción, en el sentido de mejorar propiedades del sistema, desde la estructura de representación o atributos de calidad como la introducción de nuevos requisitos de negocio.
3. Ingeniería directa: consiste reconstruir el sistema a partir de los modelos extraídos y transformados (mejorados) en las etapas anteriores respectivamente.

1.4.1 Reingeniería de software dirigido por modelos

En los últimos años el paradigma de desarrollo dirigido por modelos (Model-Driven Software Development: MDSD) ha sido ampliamente aceptado para el desarrollo de aplicaciones complejas.El enfoque MDSD se refiere al uso sistemático de modelos como artefactos clave de ingeniería a lo largo de todo ciclo de vida del desarrollo de software, desde la especificación del sistema, análisis y diseño e implementación. El enfoque más prometedor para el MDSD es la arquitectura dirigida por modelos (Model Driven Architecture: MDA) definido por el grupo de gestión de objeto (Object Management Group: OMG) [58], que básicamente está organizado en torno de los llamados modelos independiente de la plataforma (platform-independent models: PIMs) y modelos

específico de la plataforma (platform-specific models: PSMs), bien como las transformaciones entre los referidos modelos. El enfoque MDA cuenta con el suporte de un conjunto de herramientas y estándares definidos por el OMG, entre ellos, UML (Unified Modelling Language), MOF (Meta Object Facility), XMI (XML Metadata Interchange), ATL (Atlas Transformations Language) y QTV (Query View Transformation) [59].

La integración de este enfoque en los procesos de reingeniería de software se hizo necesaria principalmente debido a las limitaciones tecnológicas que caracterizaban la mayoría de los proyectos de reingeniería tradicional y que contribuía en gran parte para el fracaso de varios proyectos [60]. En 2007, el OMG ha propuesto la iniciativa de modernización dirigido por arquitectura (Architecture-Driven Modernization: ADM) [22], que aboga por la aplicación de técnicas y herramientas propias del MDSD con el objetivo de formalizar y estandarizar los procesos de reingeniería dirigida por modelos. Básicamente, este enfoque trata los artefactos del software como modelos, y provee transformaciones de modelos entre ellos. De este modo, los mantenedores realizan proyectos de reingeniería de forma más automática y reusables porque los modelos son basados en metamodelos estándar y los modelos de transformación son implementados de acuerdo los lenguajes especificados [20].

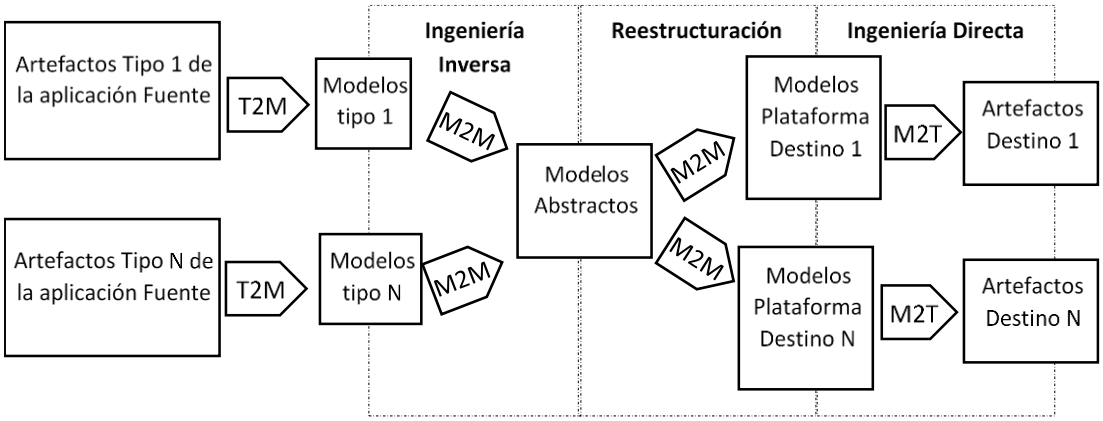

Figura 1.5: Proceso de reingeniería dirigido por modelos. Adaptado de [21]

La Figura 1.5 describe la estructura general de un proceso de reingeniería dirigido por modelos. Los mentamodelos y transformaciones son las técnicas básicas sobre las que se construyen las soluciones MDSD. En el caso de la reingeniería, las etapas de ingeniería inversa, reconstrucción e ingeniería directa son organizadas como una cadena de trasformaciones de modelo. Inicialmente los artefactos involucrados, como código fuente, base de datos, diagramas de diseño, etc., son previamente transformados

para un determinado tipo de modelo (Transform to Model: T2M). En seguida se aplica la ingeniería inversa como una cadena de transformaciones modelo a modelo (Model to Model: M2M). La etapa siguiente es la reestructuración, que implica otra cadena de transformaciones M2M, y finalmente se desarrolla la nueva versión del sistema mediante la ingeniera que puede ser llevado a cabo por transformaciones directas de los modelo obtenidos y refinados en la etapa de ingeniera inversa y reestructuración respetivamente, lo que se suele llamar enfoque translationist, o por intermedio de un proceso de ingeniería directa que alterna generación automática y elaboración manual, lo que se conoce como enfoque elaborationist [43]. Hoy día, la mayoría de los enfoques MDA son elaborationist, y tienen que lidiar con algunas situaciones de sincronización de modelo para modelo y/o modelo para código.

La iniciativa ADM ha también creado el Knowledge Discovery Metamodel (KDM) que ofrece un formato común para el intercambio de información entre las herramientas de reingeniería. KDM es comparable con el estándar UML. Mientras UML se utiliza para generar nuevo código de una manera de arriba hacia abajo, un proceso de reingeniería basado en KDM se inicia desde el código existente y construye modelos de nivel superior de manera ascendente. Las herramientas de reingeniería tradicional se han construido de forma específica para determinada tecnología lo que limita la reutilización de los artefactos producidos en dichas implementaciones [20]. La iniciativa ADM alienta los mantenedores para construir herramientas de reingeniería basados en KDM para aumentar la estandarización del proceso. Todavía la mayoría de las herramientas de reingeniería no alcanzan este enfoque, lo que se traduce en considerable dependencia tecnológica en los proyectos de reingeniería que utilizan dichas herramientas.

1.4.1.1 Herramientas para reingeniería de software

La mayoría de las técnicas aplicadas en el proceso de reingeniería de software son complejas y, como cualquier otro proceso de desarrollo de software complejo, son mejor suportados por herramientas de ingeniería de software asistida por computador (Computer-Aided Software Engineering: CASE) o por Frameworks especializado para el proceso. Estas herramientas pueden apoyar desde la ingeniería inversa, la reconstrucción hasta la ingeniería directa, permitiendo la realización del proceso de una manera semiautomatizada. La Tabla 1.2 muestra algunas de las herramientas que apoyan el proceso de reingeniería.

Tabla 1.2: Herramientas para reingeniería de software. Adaptado de [20]

Herra	Tipo	Etapas que suporta	técnicas	MDD	artefactos de entrada	artefactos de salida	Utilidad para el mantenedor
Amelio	Comercial	Todas	Análisis estática, refactorización, y Generación automática de código	No	Código fuente y Base de datos relacional	Flujo de datos y Gráficos de flujo de proceso, código	Planificación eficiente, cálculos y seguimiento de proyectos de modernización
Blue Age	Comercial	Todas	Análisis estática, refactorización, y Generación automática de código	Sí	Código fuente (varias lenguajes)	Diagramas de diseño del sistema y código fuente(varias lenguajes)	Migración y transformación
Doctrine2	Académico	Ingeniería inversa	Introspección a esquema de datos	No	Base de datos relacional	Clases PHP	Migración y transformación
MoDisco	Académico	Ingeniería inversa	Análisis estática y refactorización	Sí	Código fuente, base de datos	Modelos de representación de artefactos	Recuperación de modelos de diseño dirigido por metamodelo
PHP_UML	Académico	Ingeniería inversa	Transformación de metamodelos	Sí	Código fuente PHP	Diagramas de clases UML	Recuperación de modelos de diseño dirigido por metamodelo

1.4.2 Trabajos de reingeniería dirigido por modelos para aplicaciones Web

La integración de técnicas de ingeniería inversa con la ingeniería Web dirigido por modelos resulta en un excelente proceso de reingeniería para aplicaciones Web [14]. En esta línea de investigación son escasos los trabajos que plantea una solución completa del proceso. Entre los propuestos, se encuentran los trabajos de Echeverría en [21], destinado a la modernización de WebApps heredadas a Aplicaciones enriquecidas (Rich Internet Applications*:* RIA), y la de Bernardi, y otros en [23], que define un modelo de proceso de reingeniería dirigido por modelos, destinado a reducir el esfuerzo del mantenimiento y evolución de las WebApps. Estas propuestas han sido concebidas para determinado dominio de aplicaciones Web, y se centran en un conjunto específico de aspectos, como la arquitectura y las tecnologías utilizadas en la implementación de referida aplicación, lo que limita su implementación en escenarios diferentes de los originalmente concebidos. Parte de los inconvenientes de estos procesos subyace de las limitaciones descriptas en el epígrafe 1.2.1, que caracteriza la mayoría de las

metodologías de la ingeniería web dirigida por modelos, como la falta de completa independencia de las plataformas, la dificultad en describir los diferentes aspectos que intervienen en el desarrollo de aplicaciones, bien como costos de adquisición y mantenimiento de las herramientas CASE que suportan las metodologías e Imposibilidad de intercambio de modelos entre diferentes herramientas:

No obstante, la propuesta planteada por Bernardi y otros en [23], presenta un modelo de proceso de reingeniería que puede ser fácilmente adecuado a cualquier metodología de ingeniería web dirigido por modelos, y teniendo en cuenta las herramientas disponibles para el suporte semiautomático de las actividades de ingeniería inversa y directa respectivamente, lo convirtiendo en un aceptable marco de referencia para procesos de reingeniería dirigido por modelos para las aplicaciones Web. Esta propuesta se enfoca en incrementar la mantenibilidad de aplicaciones Web a partir de la recuperación del documento del diseño de la aplicación y la realización de un conjunto de tareas de mantenimiento segundo los fallos de calidad encontrados en el software. Por este motivo, considerando el propósito de la presente tesis, este proceso ha sido seleccionado como referencia de proceso de reingeniería de aplicaciones Web dirigido por modelos, por lo que se describe en el siguiente epígrafe.

1.4.2.1 Proceso de evolución de aplicaciones Web dirigido por modelo

Basado en estudios anteriores, Bernardi y otros en [23] han propuesto un proceso de evolución de aplicaciones Web dirigido por modelos, compuesto por dos etapas fundamentales, específicamente, ingeniería inversa, en el cual, los modelos de diseño de la aplicación son recuperados; y posteriormente, tiene lugar un proceso de ingeniería directa dirigido por modelos para modificar el diseño de la aplicación de acuerdo a los nuevos requisitos y producir una nueva versión de la aplicación. Los pasos claves de cada una de las fases se describen a continuación:

Fase 1. Ingeniería inversa: Esta etapa consiste en obtener una descripción abstracta de la aplicación Web, e incluye los siguientes pasos:

- Selección/definición de los modelos a recuperar: La selección de modelos a recuperar puede realizarse de acuerdo a los nuevos requisitos para la evolución de la aplicación. Los tipos de modelos pueden ser los que fueron utilizados a lo largo del desarrollo y mantenimiento u otros según los nuevos métodos y técnicas que los desarrolladores pretendieren aplicar para la nueva versión de la aplicación. Las metodologías de desarrollo web dirigido

por modelos mencionadas en el epígrafe 1.2.1, proveen excelentes modelos para la descripción de aplicaciones Web.

- Selección/identificación de herramientas capaces de recuperar los modelos seleccionados: Tras la selección de los modelos, deben ser identificadas herramientas de ingeniería inversa (véase Tabla 1.2) para la ejecución de la actividad de recuperación. La selección de modelos a recuperar puede ser afectada por la disponibilidad de dichas herramientas
- Análisis o comprensión de los modelos recuperados: El análisis de los modelos recuperados se destina a la comprensión de la aplicación, fundamentalmente con respecto a las características a modificar.

En la literatura se han encontrado propuestas de ingeniería inversa de aplicaciones Web dirigido por modelos, entre ellos se destacan WARE [10], que recupera el diseño para los modelos de una extensión UML propuesta por Conallen (Conallen 1999). Otra propuesta ha sido planteada en [61], que recupera a partir del código de PHP el diseño de la aplicación de acuerdo a metodología WebUML. El trabajo reciente de Bernardi, y otros [62], proponen RE-UWA, que recupera los modelos de la metodología UWA.

Fase 2. Ingeniería directa: Esta fase recibe como entrada los modelos recuperados en la etapa anterior, y por consiguiente, a partir de estos modelos y teniendo en cuenta los nuevos requisitos para la aplicación, así como la (nueva) metodología de diseño a ser utilizado para el mantenimiento o evolución, se deben identificar los fallos y debilidades del diseño actual, definir los cambios en sentido de superar las debilidades del diseño, y realizar las modificaciones ara incorporar los nuevos o redefinidos requisitos

Las operaciones de evolución pueden basarse en los enfoques de ingeniería dirigido por modelos, entre ellos, OOHDM, WebML, UWA, donde los modelos recuperados son utilizados como punto de partida para modificar o rediseñar la aplicación.

Los componentes claves del proceso de evolución dirigido por modelos son los siguientes:

- Un lenguaje de modelado estándar para describir formalmente los conceptos estandarizados, es decir un mentamodelo, como UML, para describir la aplicación según los modelos especificados

- Una manera para las transformaciones entre las tecnologías. La tendencia actual es identificar un lenguaje de transformación de modelos, como, ATL o QTV, para definir un conjunto de reglas de transformación para aplicar a los modelos de entrada en el sentido de obtener automáticamente los modelos o artefactos de salida. La disponibilidad de las tecnologías o lenguajes de transformación y modelado puede afectar la selección de los tipos de modelos a recuperar como el diseño de la aplicación.

El presente proceso fue especializado para los modelos UWA, donde la etapa de ingeniería inversa es implementada mediante el enfoque RE-UWA propuesto en [63] [62], y el proceso de ingeniería directa sigue el enfoque propuesto en [64]. En el procedimiento, los modelos de diseño UWA son recuperados de la aplicación Web heredada, y en seguida se produce un diseño basado en MVC a partir de los modelos recuperados, donde fácilmente la aplicación puede ser modificada para incorporar nuevos requisitos, tales como la migración para otras plataformas o lenguajes de programación. El proceso es suportado por las herramientas RE-UWA y UWA-MDD [62]. La primera es empleada para la recuperación automática de los modelos UWA, mientras la otra suporta la etapa de ingeniería dirigida por modelos. No obstante, las referidas herramientas no se encuentran disponibles, además los modelos UWA recuperados se limitan en los aspectos de las aplicaciones Web ubicuas y por consiguiente son ineficientes para modelar otras categorías de aplicaciones.

1.5 Modelo para la evaluación de la mantenibilidad de Aplicaciones Web

Existen varios enfoques para medir la mantenibilidad de todos los elementos software sometidos a mantenimiento, entre ellos, el código, los documentos de usuario, documentos de diseño, etc. Según el propósito de la presente tesis, la documentación del diseño y el código de la aplicación son los elementos claves a considerar en la mantenibilidad de una aplicación Web. En este sentido, a partir de un estudio comparativo de enfoques específicos para la medición de la mantenibilidad de aplicaciones Web realizado en [65], se constató que el Web Applications Maintainability Model (WAMM) propuesto por DiLuca y Tramatoma en [13], es lo que centra en el código fuente de las aplicaciones Web, en particular en las sub-características de control y de información

WAMM se basa en el modelo HPMAS (Hierarchical Multidimensional Assessment Model) utilizado en el dominio de software tradicional, y especifica un conjunto de métricas para medir la mantenibilidad de las aplicaciones tanto a nivel del sistema como de componente como se describe en la Tabla 1.3.

Tabla 1.3: Métricas de aplicaciones a nivel del sistema y componente. Fuente: [13]

Métricas a nivel del Sistema		**Métricas a nivel del Componente**	
TotalWebPage#	Número total de páginas	WebPageTag#	Número de tags en la página
TotalLOC#	Número total de líneas de código	WebPageScript#	Número de script en la página
ServerScript#	Número total de scripts del servidor	PageCodeSize#	Número de líneas de código de la página
ClientScript#	Número total de script del cliente	WebPageRelationships#	Número de relaciones de la página con otras páginas
TotalConnectivity#	Número total de relaciones en las páginas	PageLanguage#	Número de lenguajes de programación/scripting utilizadas para implementar la página
TotalLanguages#	NT de lenguajes de programación		

La Tabla 1.4 describe los atributos considerados y las métricas necesarias para evaluar la mantenibilidad de las WebApps de acuerdo con WAMM. En esta tabla solo se describen los atributos de mantenibilidad con respecto al control de estructura del código, debido al propósito de la tesis.

Tabla 1.4: Métricas y atributos de Mantenibilidad del Control de aplicaciones Web. Fuente: [13]

Atributo y Métricas a nivel del Sistema		**Atributo y Métrica a nivel del Componente**	
Tamaño	[TotalWebPage#, TotalLOC#, WebObject#, ServerScripts#, ClientScripts#,TotalLanguages#]	**Tamaño**	PageCodeSize
Modularidad	[TotalWebPage#, Average PageCodeSize]	**Modularidad**	[PageCodeSize, TotalWebPage#]
Control de acoplamiento	Total WebPageRelationships# / TotalWebPage#	**Control de acoplamiento**	WebPageRelationships#
Reusabilidad	Include# / TotalWebPage#		

El índice de mantenibilidad (Maintainability Index: MI) de las aplicaciones Web es calculado como una función de los atributos descriptos en la Tabla 1.4, como se muestra en la ecuación 1.1

$$MI = F(y_i A_i) \quad (1.1)$$

Donde, y_i es el valor *i-th* del atributo de mantenibilidad y A_i es el peso asignado a cada atributo de acuerdo con el impacto que posee en la manatenibilidad. Normalmente la definición de los valores del peso requiere un análisis de la información proveniente de las diferentes operaciones de mantenimiento. Por default el peso de cada atributo es 1.

Conclusiones parciales

Después de un análisis del estado del arte referido a las aplicaciones Web, su desarrollo y mantenimiento, bien como los enfoques de reingeniería para mejorar su mantenibilidad, se concluye el siguiente:

- El desarrollo ad-hoc que a menudo caracteriza el desarrollo de aplicaciones Web en muchas organizaciones provoca en muchos casos efectos desastrosos en el mantenimiento de dichas aplicaciones.
- La reingeniería de software es uno de los métodos fuertemente recomendado para incrementar la mantenibilidad de las aplicaciones Web. Los procesos de reingeniería de software que integran el paradigma de desarrollo dirigido por modelos son la actual tendencia de las soluciones de reingeniería debido a los desafíos de automatización y estandarización del proceso.
- El proceso de evolución dirigido por modelos de aplicaciones propuesto por Bernadi y otros en [23], sirve como una excelente referencia para soluciones de reingeniería dirigido por modelos para aplicaciones Web. Este proceso define dos elementos claves para la reingeniería, la especificación de un metamodelo y una manera de transformar los modelos.
- La efectividad de los procesos de reingeniería puede ser evaluado con la medición de la mantenibilidad del software tras su realización. En el dominio de las WebApps el Web Applications Maintainability Model es uno de los más aplicados. En este trabajo se propone este modelo para medir la mantenibilidad del software para el análisis de los resultados del proceso de reingeniería.

CAPÍTULO 2. PROCEDIMIENTO DE REINGENIERÍA PARA INCREMENTAR LA MANTENIBILIDAD DE APLICACIONES WEB HEREDADAS

Introducción

La reingeniería es un método fuertemente recomendado para incrementar la mantenibilidad del software. Este Capítulo presenta una propuesta para la reingeniería de aplicaciones Web heredadas. Se plantea inicialmente breves consideraciones sobre la propuesta, y en seguida se describen la estructura general del procedimiento, su alance, los roles involucrados en el proceso bien como los requisitos necesarios para su realización. Tras la presentación de estos elementos, son finalmente detalladas las actividades planteadas en el procedimiento.

2.1 Consideraciones generales

El presente procedimiento ha sido concebido con base en la estructura general de un proceso de reingeniería de software dirigido por modelos descripto en el epígrafe 1.5.1, con principal referencia en el proceso propuesto por Bernardi y otros en [23], descripto en el epígrafe 1.5.2.1. Básicamente, la propuesta es una especialización del referido proceso para los modelos de la metodología UWE y cuenta con el soporte de librerías de código abierto, específicamente, el ORM Doctrine2, PHP_UML, la herramienta de modelado ArgoUWE para la etapa de ingeniería inversa, y Symfony2 para la etapa de ingeniería directa. La presente especialización es un enfoque novedoso a medida que hasta la elaboración de la tesis, el referido proceso solo fue especializado para la metodología UWA con el suporte del framework propietario RE-UWA para la etapa de ingeniería inversa y UWA-MDD para el desarrollo dirigido por modelos, y se destina a aplicaciones Web ubicuas implementadas con la tecnología Java Server Face y basadas en la arquitectura MVC. Mientras, el presente procedimiento no se limita a ninguna categoría de aplicación Web y se destina a las implementadas con la tecnología PHP y que no necesariamente se basan en la arquitectura MCV. El procedimiento ha sido concebido con las referidas particularidades teniendo en cuenta el escenario de desarrollo de software en la Universidad Agostinho Neto y las características de las aplicaciones Web existente en la universidad, donde la mayoría son implementadas con la tecnología PHP y no obedecen a la arquitectura MCV.

2.2 Estructura del procedimiento propuesto

La Figura 2.1 describe de forma general el procedimiento propuesto. En la etapa inicial la aplicación fuente es analizada para la identificación de posibles módulos, que servirán de entrada al ciclo intermedio del procedimiento bajo un enfoque incremental e interactivo por cada módulo. En el ciclo intermedio del procedimiento se deberá ejecutar la trasformación del módulo seleccionado a la arquitectura MVC con el objetivo de separar el modelo de datos de la lógica de negocio y de la presentación y, facilitar el proceso de la etapa siguiente que es la ingeniería inversa para UWE, que se destina obtener una descripción más abstracta de la aplicación con base en los modelos UWE. En seguida, se procede una etapa de ingeniería directa mediante refactorización del código de la aplicación, concluyendo las actividades intermedias del proceso. Finalmente, después que todos módulos identificados cumplieren el ciclo intermedio del proceso, se procede la integración de los referidos módulos y se obtiene la nueva versión de la aplicación.

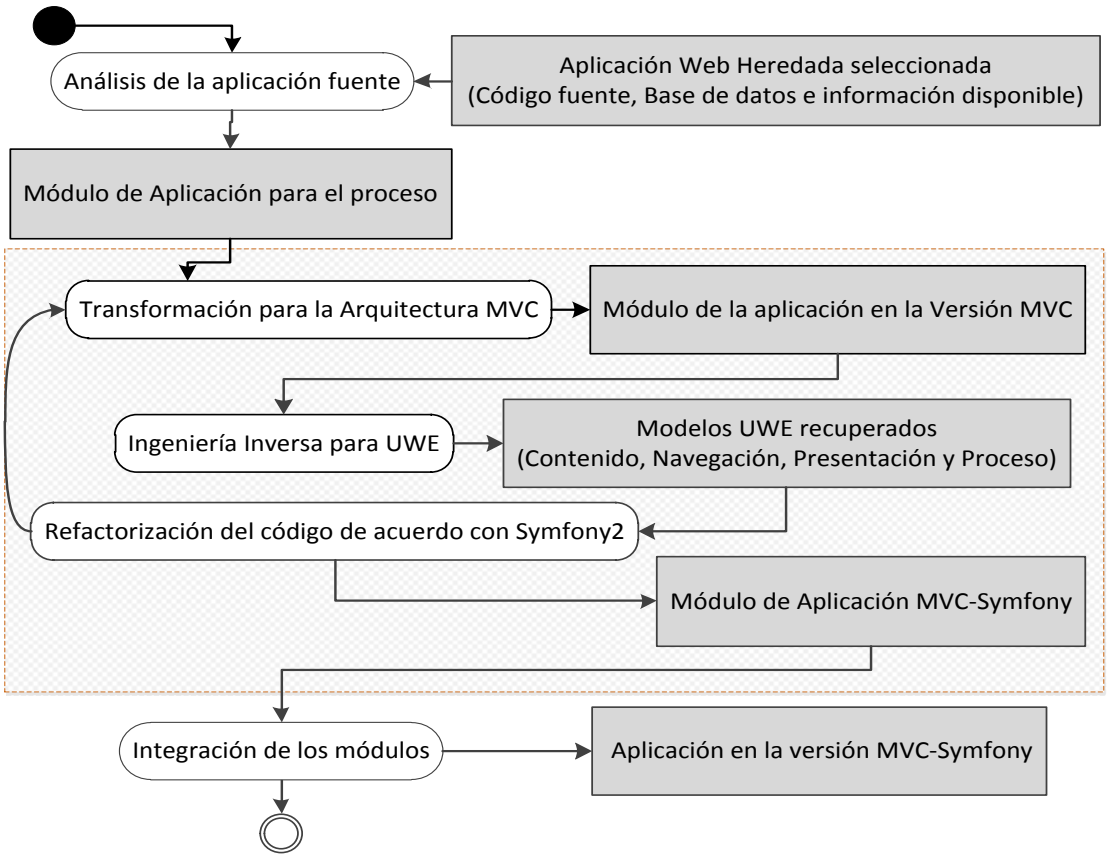

Figura 2.1: Diagrama de Actividades del procedimiento propuesto

2.3 Dominio

La propuesta presentada en este trabajo se circunscribe al dominio de aplicaciones Web desarrolladas en PHP implementadas sobre una arquitectura de tres capas y que puede o no estar basada en el patrón de diseño MVC. Normalmente estas aplicaciones presentan una arquitectura defectuosa caracterizada por la no separación de las capas del diseño de la arquitectura y consecuentemente son difíciles de mantener y evolucionar. Además, posee en muchos casos, poca o inadecuada documentación, lo que incrementa aún más la dificultad de su mantenimiento.

2.4 Roles

La definición de un proceso de reingeniería de software debe ser flexible y adecuarse a las características de la organización. Como se ha mencionado en las consideraciones generales, el procedimiento propuesto ha sido concebido teniendo en cuenta el escenario de desarrollo de software en el departamento de ciencias de computación de la Universidad Agostinho Neto. En esta organización, el desarrollo de software es realizado por un número reducido de ingeniero de software coordinados por un líder de proyecto. En este sentido existe la necesidad de una mayor organización para minimizar los esfuerzos y brindar un mayor resultado. A continuación se definen los roles participantes como miembros de la actividad de mantenimiento:

- Líder del proyecto: Representa a persona con autoridad en la organización y de inmediato interés en que se le realice el proceso de reingeniería a la aplicación Web en cuestión para mejorar su calidad.
- Mantenedor: Ingeniero responsable por las actividades intermedias y finales del procedimiento, desde la planificación de la nueva versión de la aplicación con base en los módulos seleccionados por el líder del proyecto hasta su rediseño e integración para la nueva versión de la aplicación.

2.5 Requisitos para el procedimiento

La reingeniería es normalmente suportada por herramientas que automatizan ciertas actividades del proceso. Según el proceso base del procedimiento, para la realización del proceso de reingeniería de aplicaciones Web dirigido por modelos, deben ser especificados los siguientes elementos en las respectivas etapas del proceso.

Ingeniería inversa

- Definir los modelos a recuperar, lo que se traduce en definir una metodología de desarrollo Web dirigida por modelos, mencionadas en el epígrafe 1.2.1, como WebUML, UWA, UWE, etc.
- Definir la herramienta para recuperar los respectivos modelos, mencionados en el epígrafe 1.4.1.1, como MoDisco, BlueAge, etc.

Ingeniería Directa:

- Un lenguaje de modelado estándar para describir formalmente los conceptos estandarizados, como UML.
- Una manera para las transformaciones entre las tecnologías, como, ATL o QTV, para definir un conjunto de reglas de transformación de los modelos de entrada para los especificados artefactos de salida.

Contundo, el modelo advierte que la definición o selección de estos elementos es realizada de acuerdo al escenario de la organización y de las características de las aplicaciones Web en causa, como su tecnología y arquitectura. En este sentido se proponen para el presente procedimiento los siguientes requisitos para las actividades planteadas en la estructura general del procedimiento (epígrafe 2.2).

2.5.1 Requisitos para la etapa de transformación MVC

2.5.1.1 Symfony2

Symfony es un framework orientado a objeto y basado en el patrón MVC destinado a optimizar el desarrollo de aplicaciones PHP. Este framework proporciona varias herramientas y clases destinadas a reducir el tiempo de desarrollo de una aplicación PHP compleja. Symfony2 es la versión más reciente de Symfony publicada en 2009 ideada para exprimir al límite todas las nuevas características de PHP 5.3, y por eso es, en la actualidad, uno de los frameworks PHP con mejor rendimiento [66]. El alto rendimiento de Symfony2 y su popularidad ha influenciado para su utilización en este procedimiento.

2.5.1.1.1 Implementación MVC en Symfony2

Symfony2 basa su funcionamiento interno en la famosa arquitectura MVC descripta en el epígrafe 1.1.2.1. En la Figura 2.2 se presenta un esquema simplificado del funcionamiento interno de Symfony2.

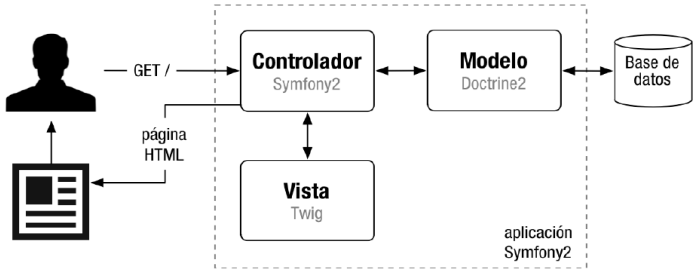

Figura 2.2: Implementación MVC en Symfony2. Fuente: [66]

La implementación del patrón MVC en Symfony2 conforma el siguiente flujo:

1 El sistema de enrutamiento de la aplicación determina qué controlador está asociado con cada URL (página) relacionada con la petición del usuario

2 Symfon2 ejecuta el controlador asociado a URL. En este momento el controlador puede solicitar al Modelo datos dependiendo de la petición, y en seguida solicita a la Vista que crea una página para la presentar la respuesta referente a la petición del usuario

3 Finalmente, el controlador entrega al servidor la página creada por la Vista.

2.5.1.1.2 Estructura de directorio de Symfony2

Las aplicaciones Symfony2 generalmente tienen la siguiente estructura de directorios:

- app/: contiene la configuración de la aplicación.
- src/: aquí se encuentra todo el código PHP de la aplicación.
- vendor/: por convención aquí se guardan todas las librerías creadas por terceros.
- web/: este es el directorio web raíz y contiene todos los archivos que se pueden acceder públicamente.

Paralelo a la estructura de directorios, otro concepto importante para que se construyan proyectos en Symfony2, es el bundle, que es un conjunto estructurado de archivos que se encuentran en un directorio y que implementan una sola característica [67].Los bundles son la base de la nueva filosofía de trabajo de Symfony2. El código de tus

aplicaciones y el propio código fuente de Symfony2 se estructura mediante bundles. Los bundles de las aplicaciones Symfony2 suelen contener clases PHP y archivos web (JavaScript, CSS, imágenes, etc.).

2.5.1.2 ORM Doctrine2

Un ORM (Object-Relational Mapping) es una interfaz que traduce la lógica de los objetos a la lógica relacional. Doctrine2 es un ORM del *Proyecto Doctrine*[3]*,* que proporciona una capa de persistencia para objetos PHP.

El proyecto Docrtine2 tuvo su inicio en 2006 por Konsta Vesterine, y actualmente se encuentra en su versión 2.5. Esta librería utiliza como base el patrón Data Mapper[4] con el objetivo de alcanzar una completa separación de la lógica de negocio de la persistencia en un sistema de gestión de base de datos relacional (SGBD). Doctrine2 se sitúa en la parte superior de una poderosa capa de abstracción de base de datos (Data Base Abstraction Layer: DBAL) del mismo proyecto. El DBAL del *Proyecto Doctrine* es una potente capa de abstracción de base de datos con muchas características para la introspección a esquema de base de datos, gestión de esquema y abstracción PDO (PHP Data Objects). PDO es una extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se logra hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos SGBD.

Entre las características clave de Doctrine2 está la capacidad de introspección a base de datos, que permite generar metadatos que representan la estructura de las tablas que la forman. Se trata de una opción muy útil cuando se hace ingeniería inversa en el dominio de aplicaciones Web implementadas en PHP.

La introspección a esquema de base de datos, gestión de esquema y la abstracción PDO, además de su alta popularidad, tornan Doctrine2 un ideal framework para el procedimiento propuesto en este trabajo.

---

[3] El Proyecto Doctrina es el hogar de varias librerías de PHP que se centran principalmente en el almacenamiento de base de datos y el mapeo de objeto

[4] Es una un patrón de arquitectura (www.datamapper.org)

2.5.2 Requisitos para la etapa de ingeniería inversa

2.5.2.1 Definición de los modelos a recuperar

La tabla 2.1 presenta parte de un estudio comparativo entre las principales metodologías para el desarrollo Web planteado en [68], que tiene como criterio de comparación las técnicas utilizadas por cada metodología para las etapas claves del proceso de desarrollo de aplicaciones Web, específicamente, requisitos, análisis, diseño, implementación, y pruebas[5].

Tabla 2.1: Metodologías para el desarrollo de Aplicaciones Web. Adaptado de [68]

Método	Análisis	Diseño	Implementación	MDA
OOHDM	Análisis de Casos de USO; UIDs	Diagrama de Contenido; y Abstract Data VIEW (ADV)	OOHDM-WEB (comercial); CGILua (comercial)	No
WebML	Plantillas; Lenguaje Natural; Análisis de Casos de Uso; Acceptance Test	Esquema Estructural en XML, Notación, grafica e Textual	Webratio (comercial)	parcial
UWE	Diagrama de Clase, Escenarios; Análisis de Casos de Uso; Prototipo	Diagrama de Clase, Estado, Actividades; Desarrollo; despliegue, y Estereotipo UML	ArgoUWE (académica) MagiCUWE (comercial)	completa

Sobre este estudio se ha basado la selección de la metodología para este trabajo, tomando como criterio de selección la cobertura de técnicas para la etapa de diseño, debido a que el procedimiento debe ser capaz de describir cualquier categoría de aplicación Web de forma más apropiada posible, y se tuve igualmente en cuenta la disponibilidad de las herramientas utilizada para el diseño, en la cual se optó por las metodologías que pueden ser suportadas por herramientas académicas. Por otra parte se analizó el nivel de cobertura del enfoque MDSD a partir de la adopción de los principios de MDA por las referidas metodologías.

El estudio constató que UWE es la metodología que propone mayor cobertura para la etapa de diseño, lo que se traduce en una descripción más generalizada para cualquier categoría de aplicación Web. Y a nivel de la integración de los principios MDSD, UWE

[5] No se muestra en la tabla descripta por ser irrelevante con base en los objetivos del trabajo

es igualmente la que posee enormes avances en la adopción de los principios del enfoque MDA, y como resultado, la metodología ha reestructurado su conjunto original de modelos en términos de metamodelos, y su proceso de desarrollo en términos de transformaciones entre dichos modelos [69] [43]. El análisis descripto llevó a que se seleccionasen los modelos UWE como la propuesta planteada en este trabajo, por lo que se describen en el epígrafe siguiente.

2.5.2.1.1 Modelos de la metodología UWE

La metodología UWE (UML-based Web engineering: UWE) es una propuesta para el desarrollo de aplicaciones Web, orientada a objeto, iterativa e incremental, basada en UML y en el proceso de desarrollo de software unificado. El principal enfoque de la metodología es el diseño sistemático a través de generación semiautomática de la aplicación web [41]. Para el diseño de los modelos, UWE utiliza un perfil UML[6], lo que permite que sea utilizada en cualquier herramienta CASE que suporta UML. Actualmente UWE cuenta con el suporte de la herramienta CASE de libre acceso ArgoUWE [70], y la comercial MagicDraw a partir del plug-in MagicUWE.

La notación utilizada en UWE cubre los aspectos de contenido, navegación, presentación y proceso del dominio de las aplicaciones Web como se muestra en la Figura 2.3.

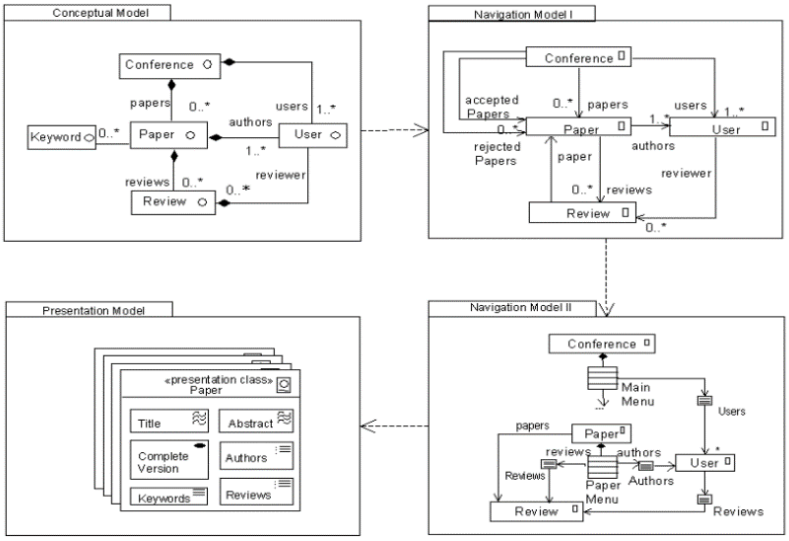

Figura 2.3: Ejemplo de proceso de diseño UWE. Fuente: [70].

[6] Extensión de metamodelos UML especifico a un dominio

Modelo de Contenido

El modelo de contenido incluye los objetos necesarios para suportar la funcionalidad que el sistema ofrece a sus usuarios. El objetivo de este modelo es proveer una especificación visual de la información relevante del dominio de la aplicación Web ignorando los aspectos de navegación, presentación e interacción cuanto posible. Los elementos UML precisados en el modelo de contenido de UWE no se diferencian de un modelo que define otro dominio; por tal motivo se utilizan las clases, atributos, asociaciones y otros elementos UML, sin necesitar de ninguna extensión o estereotipo en su construcción [71].

Modelo de Navegación

El modelo de navegación especifica cuáles son los objetos a los que se puede acceder a través de la navegación y cómo el usuario puede acceder a ellos [41]. Este modelo se base en el modelo de contenido, y tiene como objetivo representar la estructura de navegación de la aplicación entre las diferentes entidades y diseñar el camino para el acceso a referidas entidades. Los principales elementos para el diseño del modelo de navegación son estereotipos de elementos UML, específicamente, un estereotipo de clase *«NavigationNode»* y un estereotipo de asociación *«Link»* [71]

Modelo de presentación

El modelo de presentación provee una vista abstracta y describe la estructura básica de la interface del usuario (user interface: UI) de una aplicación web. Este modelo se basa en el modelo de navegación, y especifica cuales elementos de la UI (texto, imágenes, anclas, formularios, etc.) van ser usados para representar los nodos de navegación, dejando de parte los aspectos concretos de la UI, tales como el uso de colores, fuentes, y la ubicación de los elementos UI en la página Web [72]. Los elementos UI no representan componentes concretos de cualquier tecnología de presentación sino que describen qué funcionalidad se requiere en determinado punto particular de la interface del usuario [71]. Los principales elementos del modelo de presentación son las clases de presentación *«PresentationElmenet»*, estereotipos del elemento clase UML, que se basa directamente en los nodos del modelo de navegación (*«NavigationNode»*, *«menu»*, *«index»*, «*guide tou*r», etc.). Las relaciones entre los elementos de presentación son por composición (composition), es decir, el elemento debe ser dibujado dentro del otro elemento. Generalmente el contenido de distintos nodos de navegación es presentado

en una *«page»*, que puede contener, entre otras cosas, clases de presentación y grupos de presentación *«PresentacionGroup»*. Un grupo de presentación puede contener en sí grupos de presentación y clases de presentación.

Modelo de proceso

El modelo de proceso provee los elementos necesarios para la integración de los procesos del negocio en los modelos UWE descriptos en los epígrafes anteriores. Los dos elementos claves del diagrama del proceso son los estereotipos *«ProcessClass»* y *«ProcessLink,* y su integración puede ser separada en las siguientes tareas [71]:

a) Integración de los procesos de negocio en el modelo de navegación: En las aplicaciones Web que manejan procesos de negocio, los procesos deben ser integrados en el modelo de navegación. Para eso se utilizan dos estereotipos adicionales: *«ProcessClass»* y *«ProcessLink»* que extienden las clases Nodo y Link, respectivamente, y que permiten la definición de cómo un proceso puede ser accedido a través de la navegación.

b) Definición del comportamiento del proceso: El comportamiento del proceso se define a través de un diagrama de actividad UML y representa cuando el sistema tiene la acción en determinado punto del proceso «*SystemAction*» o cuando se le solicita al usuario ingresar datos e informaciones. La Figura 1.11 presenta un ejemplo de definición de un proceso.

2.5.2.2 Identificación de las herramientas para recuperar los modelos definidos

Las herramientas de ingeniería inversa existentes, algunas descriptas en el epígrafe 1.4.1.1, generalmente son concebidas para específicas tecnologías. En la actualidad, según pesquisa efectuada, una de las más utilizadas es Eclipse MoDisco, que permite el proceso de ingeniería inversa de la tecnología Java y a partir del metamodelos MOF o UML en el formato XMI. La mayoría de las otras herramientas tienen como artefacto de entrada base de datos relaciones para ingeniería inversa de base de datos y código de la tecnología java, como Java Server Face. Entretanto son considerablemente escasas las herramientas que suportan ingeniería inversa desde código PHP. El PHPModeler [73], un plug-in eclipse académico, propuesto para para ingeniería inversa de código PHP no se ha actualizado desde 2009 y se tornó obsoleta para las recientes versiones de eclipse. Finalmente, la alternativa planteada en la pesquisa efectuada fue

el PHP_UML, una librería de código abierto para la ingeniería inversa de clases PHP para diagramas UML:

2.5.2.2.1 PHP_UML

PHP_UML es un paquete de ingeniería inversa del framework PEAR, que escanea archivos PHP o directorios, y ofrece una representación UML / XMI de las clases y paquetes PHP encontrados [74]. PEAR (PHP Extension and Application Repository), es un framework y sistema de distribución para componentes de código PHP, gestionado por el proyecto PEAR[7].

PEAR consiste en una lista bastante grande de librerías de código PHP que permiten hacer ciertas tareas de manera más rápida y eficiente reutilizando código escrito previamente por otras personas. Entre estas librerías, se encuentra PHP_UML, que es un analizador (parser) de PHP, un generador de XMI y un instrumento para la documentación, que permite entre otras tareas generar archivos UML / XMI en la versión 1.4, o en la versión 2.1 en diferentes vistas, (como, lógico, componentes e implementación), que pueden ser directamente utilizados en una herramienta CASE UML, como Rational Rose o ArgoUML, lo que permite obtener una visión instantánea de una aplicación PHP, con todas las funciones habituales de una herramienta de diseño de software (como, exportación de diagramas de clases, refactorización de aplicaciones orientadas a objetos, o la generación automática de código) [74].

El procedimiento propuesto en esta tesis utiliza PHP_UML debido a simplicidad de su utilización además de contar con un suporte amplio de la comunidad de desarrolladores de proyecto PEAR.

2.5.2.2.2 ArgoUWE

ArgoUWE es una herramienta CASE que suporta la etapa de diseño del proceso de desarrollo UWE, implementada como un módulo plug-in de la herramienta de modelado de código abierto ArgoUML[8]. La herramienta ArgoUWE implementa el metamodelo UWE, y permite la realización del proceso de desarrollo UWE de forma semiautomática a través de la manipulación directa de los modelos UWE, además, cuenta con un

[7] Proyecto PEAR fue fundado por Stig S. Bakken en 1999 para promover la reutilización de código PHP.
[8] http://www.argouml.org.

conjunto de reglas OCL (Object Constraint Language) que evalúan la consistencia de los modelo durante el diseño [70].

El procedimiento propuesto utiliza ArgoUWE debido su cobertura total al proceso de desarrollo UWE y también por ser de libre acceso. Entretanto, ArgoUWE posee algunas restricciones, como el hecho de que todavía se basa en la versión 1.3 de UML, por lo que sólo metamodelos basado en UML 1.3 pueden ser importados para la herramienta. No obstante, esta restricción no ha constituido inconveniente para el procedimiento propuesto, porque los archivos XMI utilizados corresponden a la versión 1.3 de UML.

2.6 Descripción de las etapas del procedimiento

La descripción de las etapas del procedimiento específica las tareas necesarias con las respectivas entradas y salidas, los roles involucrados bien como los soportes necesarios para la realización del procedimiento.

2.6.1 Análisis de la aplicación fuente

Rol: Líder del proyecto

Entrada: Código, base de datos e información disponible de la aplicación fuente

Salida: Módulo de la aplicación seleccionado para reingeniería

Soporte: Plantilla para planificación de proyecto MVC

Descripción:

La mayoría de los proyectos de reingeniería de software encontrados en la literatura suelen incluir en la primera etapa una actividad de análisis que se destina a comprender la estructura y funcionamiento de la aplicación fuente. Este procedimiento se basa en este principio y considera esta actividad en el inicio del proceso, en el sentido de no solo comprender el funcionamiento de la aplicación, sino identificar posibles módulos de la aplicación fuente. La Figura 2.4 describe los pasos y el respectivo rol involucrado en esta actividad.

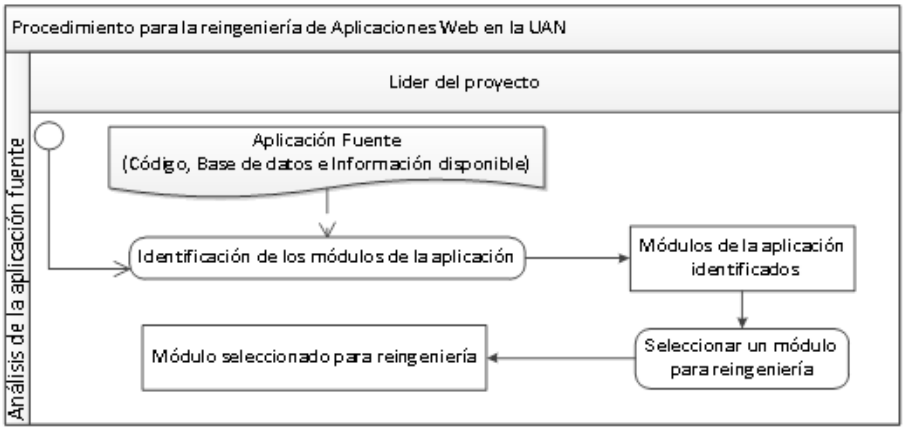

Figura 2.4: Diagrama de actividad del análisis de la aplicación fuente.

- **Identificar los módulos de la aplicación:** Un módulo en una aplicación Web generalmente conforma un conjunto específico de páginas, funciones y datos asociados a ellos, que realizan una actividad específica ofrecida por la aplicación. Este procedimiento propone al inicio del proceso la identificación de posibles módulos de la aplicación con base en la información disponible. Esta tarea es fundamental para el cumplimiento del enfoque incremental adoptado en el procedimiento. En este sentido, se implementará de forma incremental en cada módulo identificado una interacción completa del proceso hasta que todos estén rediseñados.
- **Seleccionar un módulo para la reingeniería:** Tras la identificación de los módulos de aplicación fuente, el líder del proyecto también es responsable de seleccionar un módulo para el proceso de reingeniería. Esta tarea deberá ser realizada de forma repetida en cada interacción del proceso hasta que todos los módulos de aplicación estén rediseñados. No obstante, se puede desde el inicio crear una lista ordenad para la reingeniería secuencial de los módulos de la aplicación.

2.6.2 Transformación de la aplicación a la arquitectura MVC

Rol: Mantenedor

Entrada: Código, base de datos e información disponible del módulo seleccionado

Salida: Módulo de la aplicación en MVC

Soporte: Symfony2 y Doctrine2.

Descripción:

La trasformación de la aplicación a la arquitectura MVC consiste en reestructurar la aplicación para separar el modelo de datos de la lógica de negocio y presentación. Esta actividad no consta en el proceso general en el cual se basa el procedimiento, sin embargo se considera en esta propuesta, con el objetivo de facilitar la etapa de ingeniería inversa, debido a que la descripción abstracta que se pretende recuperar, corresponde a diferentes aspectos de la aplicación Web (contenido, navegación, etc.), y por eso, es necesario que los aspectos estén debidamente separados en la aplicación. Esto se puede lograr fácilmente si la arquitectura de la aplicación obedece el patrón de diseño MVC, que encapsula los datos y su respectivo tratamiento (modelo), y lo separa de la lógica de negocio (controlador) y presentación (Vista) como se planteó en el epígrafe 1.1.2.1. Entretanto, las aplicaciones Web en análisis no obedecen al patrón MVC, y presentan una arquitectura en que no se verifica la separación de los aspectos, lo que no solo dificulta la realización del proceso de ingeniería inversa como también es una de las causas para la dificultad del mantenimiento y evolución de dichas aplicaciones. La Figura 2.5 presenta el flujo de proceso de esta actividad.

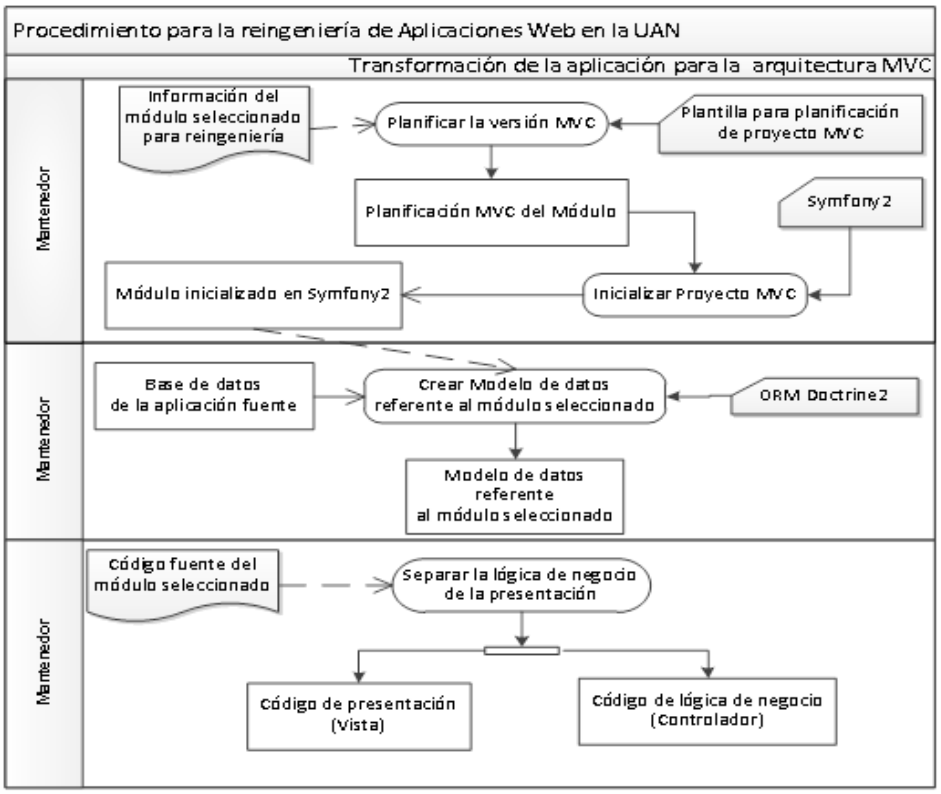

Figura 2.5: Diagrama de actividad - trasformar la aplicación a la arquitectura MCV

- **Planificar la versión MVC:** Consiste en esbozar la estructura de la aplicación en su versión MVC. Esta tarea recibe como entrada el módulo seleccionado en la etapa anterior, y se puede utilizar como suporte una plantilla que permita describir los elementos claves de estructura de un proyecto MVC. El procedimiento propone una plantilla de planificación MVC[9] que describe la estructura de un proyecto de acuerdo a implementación MVC en Symfony2. Esta planificación es realizada según Symfony2, porque ha sido el framework selccionado en este trabajo para el desarrollo de la nueva versión de la aplicación.

- **Inicializar proyecto MVC:** Consiste en crear un proyecto en el entorno Symfony2. El proyecto es inicializado a partir de la orden `php symfony new nombe_del_proyecto` o `symfony new nombe_del_proyecto`[10]. En seguida es necesario que se configuren los parámetros de conexión del archivo `app/config/parameters.yml` para apuntar a la base de datos de la aplicación fuente. La otra acción consiste en crear el bundle que corresponden al módulo seleccionado en la etapa anterior. Esta acción se realiza con la orden `generate:bundle` como se muestra en la Figura 2.6.

```
$ php app/console generate:bundle –namespace= nome_proyecto/nome_móduloBundle
```

Figura 2.6: línea de comando para generar un bundle en Symfony2

- **Crear el modelo de datos a partir de la base de datos de la aplicación fuente:** Esta tarea consiste en crear la capa de acceso a base de datos y el modelo de datos de la aplicación, con el objetivo de separar el modelo de la presentación. En Symfony2 es posible crear el modelo de la aplicación a través de una de las librerías ORM, Propel o Doctrine. Este procedimiento propone Doctrine2 descripto en el epígrafe 2.5.1.2, y para realizar la tarea, primeramente se define el esquema de la base de datos a partir de la orden "`doctrine:mapping:import`" que le pide a Doctrine2 que inspeccione la estructura de la base de datos y genere los archivos metadatos que forman el esquema de datos. La Figura 2.7

[9] Véase en la aplicación del procedimiento epígrafe 3.3
[10] En el sistema operativo Windows y Linux respectivamente

presenta un ejemplo de archivo XML generado a partir de una base de datos existente.

```xml
<?xml version="1.0" encoding="utf-8"?>
<doctrine-mapping >
<entity name="Sib\AdminiBundle\Entity\Cdu1" table="cdu1">
    <id name="idcdu1" type="string" column="idcdu1" length="8">
      <generator strategy="IDENTITY"/>
  </id>
  <field name="nomcdu1" type="string" column="nomcdu1" length="200" />
    <field name="dscrccdu1" type="text" column="dscrccdu1" nullable="true"/>
  </entity>
</doctrine-mapping>
```

Figura 2.7: Metamodelo XML generado desde una base de datos con Doctrine2

Una vez generados los archivos de metadatos, la segunda acción es construir las clases de entidades, con la ejecución de las órdenes `doctrine:mapping:convert` y `doctrine:generate:entities` respectivamente. La Figura 2.6 presenta una clase de entidad generada a partir del archivo XML descripto en la Figura 2.8.

```php
class Cdu1
{
    /**
     * @var string
     *
     * @ORM\Column(name="nomcdu1", type="string", length=200, nullable=true)
     */
    private $nomcdu1;

    /**
     * @var string
     *
     * @ORM\Column(name="dscrccdu1", type="text", nullable=true)
     */
    private $dscrccdu1;

    /**
     * @var string
     *
     * @ORM\Column(name="idcdu1", type="string", length=8)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $idcdu1;
}
```

Figura 2.8: Clase PHP generado desde un metamodelo XML utilizando Doctrine2

- **Separar la lógica de negocio de la presentación**: Consiste en separar la capa de la lógica de negocio de la presentación. El procedimiento propuesto realiza esta tarea siguiendo la filosofía de Symfony2, y la primera acción es retirar de las páginas Web el código referente a lógica del negocio que corresponden a instrucciones en PHP. Este código es debidamente separado en otro archivo de extensión PHP, y puede ser inicialmente ubicado en el directorio para los controladores (véase estructura de directorios de Symfony2, epígrafe 2.5.1.1.2). En seguida, el archivo inicial que ahora deberá contener apenas el código para

la presentación (código HTML) debe ser cambiado para la extensión "*html.php*" conforme las plantillas en Symfony2.

2.6.3 Ingeniería inversa para UWE

Rol: Mantenedor

Entrada: Módulo de aplicación en la arquitectura MVC, información disponible sobre la aplicación fuente

Salida: Modelos UWE recuperados

Soporte: PHP_UML, ArgoUWE

Descripción:

La entrada principal para esta etapa es el modelo de datos del módulo en análisis formado por las clases PHP generadas en la etapa anterior. Los pasos de esta actividad son descriptos en la Figura 2.9.

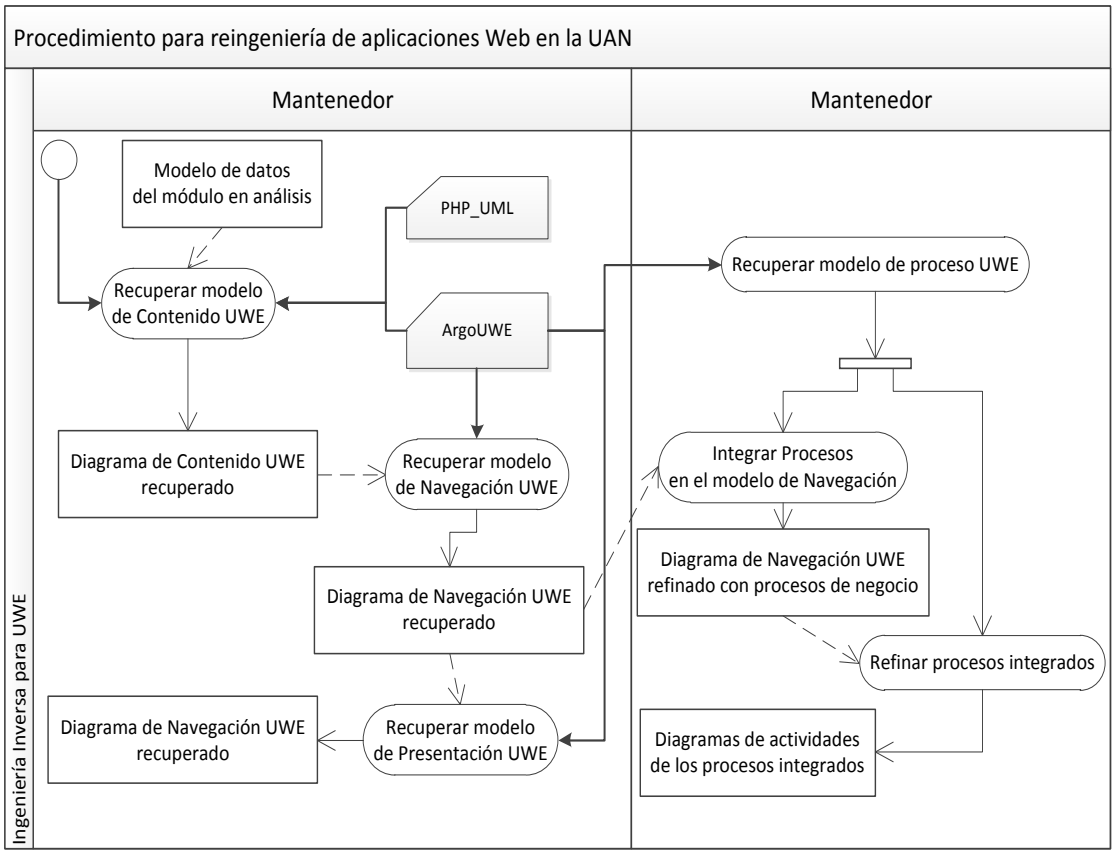

Figura 2.9: Diagrama de actividad del proceso de ingeniería inversa para UWA

El proyecto en la versión MVC es analizado para que se recuperen los modelos de diseño de acuerdo a metodología UWE. Forman parte de los modelos a recuperar aquellos considerados claves en la metodología UWE, específicamente el modelo de contenido, navegación, presentación y de proceso, descriptos en el epígrafe 2.5.2.2.3. Las tareas planteadas en la Figura 2.9 son detalladas a continuación:

- **Recuperar modelo de contenido UWE**: El modelo de contenido UWE incluye los objetos necesarios para suportar las funcionalidades que el sistema ofrece a sus usuarios. El metamodelo del diagrama de contenido UWE corresponde a un diagrama de clases UML. En este contexto, para la recuperación de este modelo, las clases de entidades PHP que constituyen el modelo de datos obtenido en la etapa anterior deberán de ser convertidos en un diagrama de clases UML .El procedimiento utiliza la librería PHP_UML, a partir del cual se ha formalizado un script para convertir las clases de entidades PHP en clases UML. La Figura 2.10 presenta el script formalizado.

```php
<?php

//importa el paquete PHP_UML
require_once 'PHP/UML.php';

//instacia la clase PHP_UML
$p = new PHP_UML;

// para encontrar los archivos en PHP
$p->setMatchPatterns(array('*.php'));
$p->dollar          = false;    // retira los $ en los nombres de la variables
$p->componentView   = false;    // no incluye los aspectos de componentes
$p->deploymentView  = false;    // no incluye los aspectos de despliegue

//Ruta del directorio con las clases en PHP
$p->setInput('c:xampp/htdocs/Sib2/src/Sib/AdminiBundle/Entity/SibAdmini');

//Nombre del elemento root del modelo
$p->parse('SibAdmini');

//instacia de la clase para la transformación
$u = new PHP_UML_Output_Xmi_Exporter();
$u->setModel($p->getModel());
// Version del XMI
$u->setXmiVersion(1.0);
// ecoding del documento
$u->setEncoding('UTF-8');

// Directorio y nombre de archivo xmi
$u->export('E:\Proyecto PHP_UML\SibAdmini_UML\sibadmini.xmi');

//vizualiza el archivo generado en el navegador
echo htmlentities($u->getXmiDocument()->dump());

//Mensaje de conlusion de la transformacion
echo '<br/>Conluido!!';
?>
```

Figura 2.10: Extracto de código para el *parser* PHP_UML

El script propuesto recibe como entrada un directorio donde se encuentran las clases PHP, y los convierte en un archivo XMI versión 1.0 que corresponde a un

diagrama de clases UML versión 1.3, de modos que sea legible en ArgoUWE, la herramienta utilizada para el diseño de los modelos UWE. Importa mencionar que la versión XMI de los archivos generados depende de la herramienta disponible para su edición.

El archivo UML 1.3 generado representa el modelo de datos de la aplicación, constituido por clases UML, sin embargo, no corresponde todavía al diagrama de contenido UWE que se pretende, para tal, deberá ser editado mediante la sustitución de la tag `Foundation.Core.Class` que representa los elementos clase UML, para `UWE.Foundation.Core.Conceptual.ConceptualClass` que corresponde las clases UML del perfil UWE. La figura 2.9 presenta un archivo XMI 1.3 editado para que corresponda a un modelo de contenido UWE.

Finalmente, tras la edición, el archivo deberá ser importado para la herramienta ArgoUWE donde se obtendrá la visualización del diagrama de contenido UWE referente al módulo en análisis, para la edición y otras operaciones ofrecidas por la herramienta. En la Figura 2.11 se visualiza un ejemplo de diagrama de contenido UWE recuperado y refinado en la herramienta de modelado ArgoUWE.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE XMI SYSTEM "UMLX13.dtd">
3 <XMI xmi.version="1.0" timestamp="2015-11-26 18:31:23">
4    <XMI.header>
5       <XMI.metamodel xmi.name="UML" xmi.version="1.3"/>
6    </XMI.header>
7    <XMI.content>
8       <Model_Management.Model xmi.id="MX_EAID_0E836A33_1109_44d6_8B43_69F7B510EE2F">
9          <Foundation.Core.ModelElement.name>sib</Foundation.Core.ModelElement.name>
10         <Foundation.Core.ModelElement.isSpecification xmi.value="false"/>
11         <Foundation.Core.GeneralizableElement.isRoot xmi.value="false"/>
12         <Foundation.Core.GeneralizableElement.isLeaf xmi.value="false"/>
13         <Foundation.Core.GeneralizableElement.isAbstract xmi.value="false"/>
14         <Foundation.Core.Namespace.ownedElement>
15         <Model_Management.Package xmi.id="EAPK_0E836A33_1109_44d6_8B43_69F7B510EE2F">
16                <Foundation.Core.ModelElement.name>sibAdmini</Foundation.Core.ModelElement.name>
17                <Foundation.Core.ModelElement.isSpecification xmi.value="false"/>
18                <Foundation.Core.GeneralizableElement.isRoot xmi.value="true"/>
19                <Foundation.Core.GeneralizableElement.isLeaf xmi.value="false"/>
20                <Foundation.Core.GeneralizableElement.isAbstract xmi.value="false"/>
21                <Foundation.Core.Namespace.ownedElement>
22                   <UWE.Foundation.Core.Conceptual.ConceptualClass xmi.id="EAID_DCEBA86B_0EF6_4078_85A0_F91DF5B9E5C6">
23                        <Foundation.Core.ModelElement.name>Bbltc</Foundation.Core.ModelElement.name>
```

Figura 2.11: Extracto de archivo XMI 1.3 editado para diagrama de contenido UWE

- **Recuperar el modelo de navegación UWE**: El modelo de navegación UWE se basa en el modelo de contenido recuperado en la tarea anterior. Este modelo especifica cuáles son los objetos a los que se puede acceder a través de la navegación y cómo el usuario accede a ellos. La herramienta ArgoUWE brinda la posibilidad de construirse un modelo de navegación de forma

semiautomática con base en un modelo de contenido existente. Para tal el diseñador deberá marcar las clases del modelo de contenido consideradas “relevantes”, como se presenta en la Figura 2.12, para que en la creación del modelo de navegación la herramienta esté habilitada para copiar todas las clases marcadas bien como sus relaciones para un nuevo modelo, transformándolas directamente como nodos de navegación y los respectivos *links* entre los nodos. En seguida se debe refinar el modelo generado con suporte de la información disponible de la aplicación, en el sentido de integrar al modelos todos los aspectos referentes a la estructura de la navegación de la aplicación

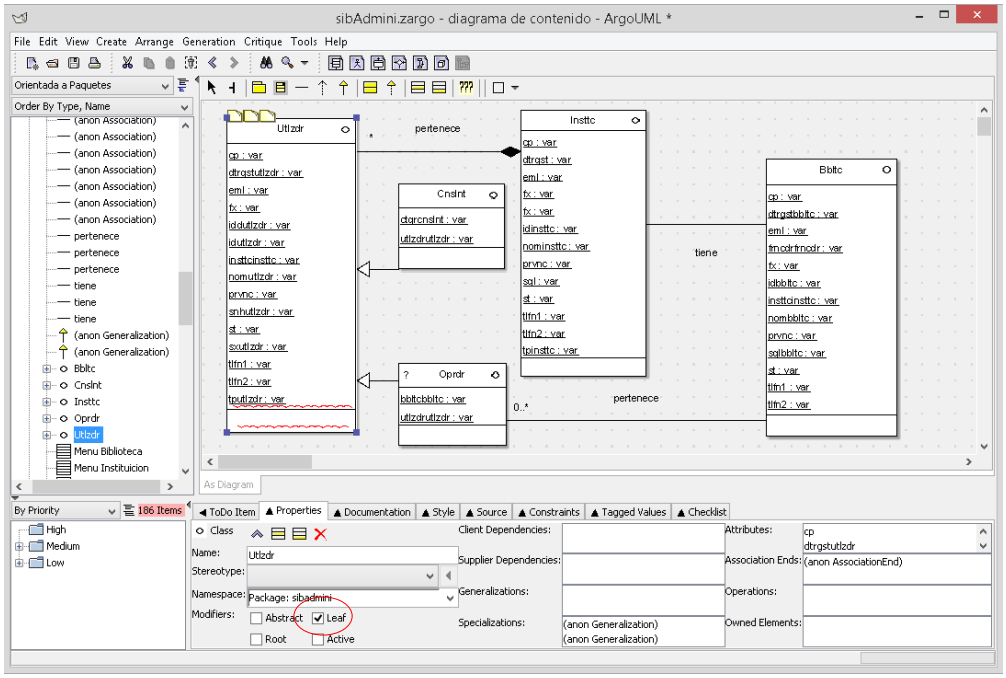

Figura 2.12: Diagrama de contenido UWA refinado

- **Recuperar el modelo de presentación UWE**: El diagrama de presentación describe la estructura básica de la interface del usuario de una aplicación Web. La recuperación de este modelo se realiza de forma similar a la recuperación del modelo de navegación descripto en la etapa anterior. En este caso, la recuperación parte del modelo de navegación.
- **Recuperar el modelo de proceso UWE**: Esta tarea consiste básicamente en integrar la lógica de negocio a los modelos recuperados anteriormente. El metamodelo de proceso UWE provee los elementos necesarios para la referida

integración, y con base en la información disponible sobre la estructura y funcionamiento de la aplicación fuente, esta tarea debe realizarse de dos maneras:

- Integración de los procesos de negocio al respetivo modelo de navegación que deberá corresponder a determinado módulo (bundle) conforme la estrategia incremental adoptada por el procedimiento. Los procesos de negocio son integrados a través de los estereotipos adicionales: *«ProcessClass»* y *«ProcessLink»*.
- Especificación el comportamiento interno de cada proceso integrado mediante diagramas de actividades UML.

2.6.4 Ingeniería directa

Rol: Mantenedor

Entrada: Modelos UWE recuperados y código fuente de la aplicación original

Salida: Aplicación MVC-Symfony

Soporte: Symfony2

Descripción:

La ingeniería directa en los procesos de reingeniería generalmente corresponde a la etapa en que se reconstruye el software a partir de los modelos recuperados en la etapa de ingeniería inversa, dando lugar a una versión mejorada del software. Este procedimiento propone la reconstrucción de la aplicación original de acuerdo al patrón de arquitectura MVC, dando lugar a una versión moderna de la aplicación con la debida separación del modelo de datos, la lógica de negocio y de la presentación a fin de facilitar el mantenimiento de la misma. La nueva versión de la aplicación es desarrollada de acuerdo a los modelos recuperados en la etapa anterior y teniendo en cuenta otras informaciones disponibles con respecto al funcionamiento de la aplicación original. El procedimiento cuenta con el suporte de Symfony2, y propone un conjunto de refactorización del código de la aplicación fuente para aplicarlo de acuerdo a la filosofía de Symfony en la nueva versión. Importa mencionar que en este momento ya existirá parte de la versión MVC de la aplicación debido a etapa de trasformación de la aplicación en la arquitectura MVC descripta en el epígrafe 2.6.2. Por lo tanto, la refactorización solo se centrará en la lógica de negocio y presentación según la filosofía de Symfony2.

2.6.5 Integración de los módulos

Rol: Mantenedor, Líder del proyecto

Entrada: Módulo de la aplicación rediseñado

Salida: Aplicación MVC-Symfony

Soporte: Symfony2

Descripción:

La integración de los módulos de la aplicación es la actividad final de procedimiento, donde los respectivos módulos rediseñados son integrados de forma incremental a la nueva versión de la aplicación. Esta actividad puede ser realizado por un mantenedor o mismo por un líder de proyecto. Se realizan pruebas de conformidad de cada módulo rediseñado con base en la funcionalidad del módulo en la versión original, en el objetivo de verificarse si el módulo ha conservado el comportamiento inicial. En paralelo deben ser programadas de acuerdo con el líder del proyecto las pruebas de software que se aplican en un ciclo normal de desarrollo de software, en particular de aplicaciones Web.

Conclusiones parciales

- La reingeniería de software es un método recomendado para mejorar el mantenimiento de sistemas heredados, en particular de las aplicaciones Web. La definición de un procedimiento que tiene en cuenta el escenario de desarrollo de la organización y las características de las aplicaciones Web contribuye para el éxito del proceso
- En el sentido de dar respuesta al objetivo general de la presente investigación, la elaboración del procedimiento ha sido concebido teniendo en cuenta las características de las aplicaciones Web heredadas en la UAN, a medida que el dominio de la propuesta se consigan a aplicaciones Web implementadas en PHP con una arquitectura que no necesariamente obedezca el patrón de diseño MVC, lo que caracteriza la mayoría de las aplicaciones Web en la UAN
- La definición de los requisitos del procedimiento se guio teniendo en cuenta un escenario de desarrollo académico, donde las herramientas utilizadas son de libre acceso y/o de código abierto, lo que permite mayor aporte de varios investigadores y consecuentemente el mejoramiento del procedimiento.

CAPÍTULO 3. APLICACIÓN DEL PROCEDIMIENTO Y ANÁLISIS DE RESULTADOS

Introducción

En el capítulo se valida el procedimiento propuesto mediante su aplicación en una de las aplicaciones Web de la universidad Agostinho Neto, seguida de la medición de la mantenibilidad del referido software antes y después de la aplicación del procedimiento para comprobar el incremento de su mantenibilidad, que es la variable cuya mejora se dedicó esta investigación.

3.1 Caso de estudio: Sistema Integrado de Bibliotecas de la UAN – SIB.UAN

Para la aplicación del procedimiento se seleccionó intencionalmente el Sistema Integrado de Bibliotecas de la universidad, debido a participación del autor en el desarrollo de la referida aplicación. El SIB.UAN es una aplicación Web destinada a gestionar el proceso de catalogación y respetiva consulta de los materiales bibliográficos de la biblioteca central de la universidad *Agostinho Neto*. La Figura 3.1 presenta el diagrama conceptual del sistema.

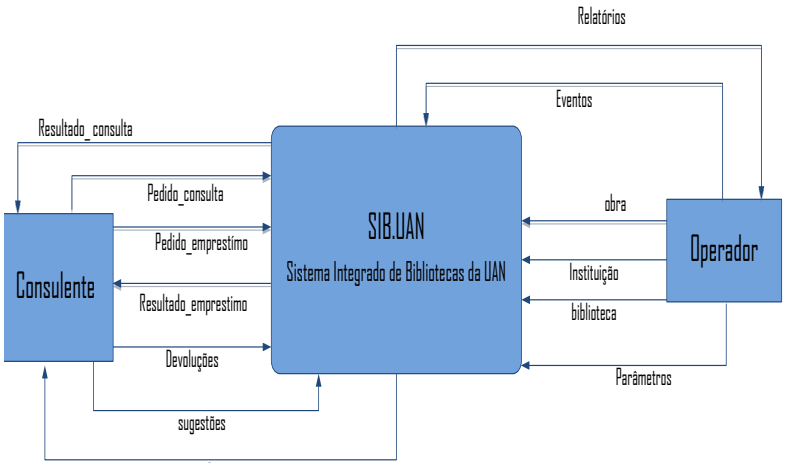

Figura 3.1: Diagrama conceptual del SIB.UAN. Fuente: [75]

El sistema fue desarrollado en 2012, como resultado de una tesis de licenciatura en ciencias de la computación en la UAN [76]. La aplicación fue desplegada en entorno experimental en el mismo año de su desarrollo. No obstante, debido la ausencia del personal que desarrolló la aplicación y su baja calidad con respecto a la capacidad de mantenimiento, el proyecto fue abandonado en el año de 2014.

La aplicación fue desarrollada en una arquitectura de tres capas con la integración de una base de datos implementada en MySQL. Se utilizó para su implementación los lenguajes PHP para el lado del servidor, y HTML y JavaScript para el lado del cliente. Según los mantenedores, la aplicación paso de forma progresiva a bajar de calidad, principalmente en su arquitectura caracterizada por la no separación de los aspectos. Además, la documentación existente de la aplicación no especifica de forma coherente la actual implementación del sistema, debido a los cambios realizados sin la debida actualización del documento. Otro dato importante, tiene que ver con la versión de las tecnologías con los cuales fue implementada la aplicación, que pasar a ser obsoletas actualmente, a medida la aplicación fue implementada con la versión 5.2 de PHP y utilizando el HTML 4, mientras que la versión estable de PHP es 5.5.y la mayoría de los navegadores Web están se actualizando para la versión 5 de HTML. Esto se traduce en la necesidad adicional de migración para las versiones modernas de las referidas tecnologías.

3.2 Entorno de desarrollo para implementación del procedimiento

El procedimiento requiere el suporte de un determinado conjunto de herramientas que han sido descriptas en el epígrafe 2.5. El entorno de desarrollo incluye el ORM Doctrine2 para la recuperación del modelo de datos, la librería PHP_UML para la obtención de una representación UML del modelo de datos, y la herramienta ArgoUWE para la recuperación y refinado de los modelos UWE.

La librería Doctrine2 puede utilizarse a partir de Symfony2 en su edición estándar disponible en (www.symfony2.com). No obstante, a menudo se sugiere utilizar Symfony2 a partir de un entorno de desarrollo integrado (IDE). Esta implementación utiliza NetBeans 7.4, que es uno de los IDEs ampliamente utilizado para el desarrollo de aplicaciones Web PHP. Los pasos para la instalación de Symfony2 en NetBeans están disponible en www.netbeans.org.

El paquete PHP_UML pertenece al framework Pear, y su instalación se realiza a partir de la línea de comando con la orden `Pear install PHP_UML.` Los detalles de la instalación de PHP_UML están disponibles en www.pear.net/PHP_UM.

La herramienta CASE ArgoUWE es de libre acceso y está disponible en www.uwe.com, juntamente con toda información necesaria para su instalación y utilización.

3.3 Implementación del procedimiento

Fase 1. Análisis de la aplicación Fuente

El SIB.UAN descripto en el epígrafe 3.1, ha sido analizado y se identificaron los módulos del sistema, como se presenta en la Tabla 3.1. En seguida se el líder del proyecto seleccionó el módulo de Administración para la primera interacción del proceso de reingeniería.

Tabla 3.1: Descripción de los módulos identificados en el SIB.UAN

NRO.	MÓDULO	DESCRIPCIÓN
1	Administración	Configurar el sistema. Operaciones CRUD
2	Adquisición	Gestionar el proceso de adquisición de materiales bibliográficos
3	Catalogación	Gestiona el proceso de catalogación de los materiales bibliográficos
4	Consulta	Gestiona el proceso de consultas y préstamos de los materiales bibliográficos

Fase 2. Transformación de la aplicación a la arquitectura MVC

En esta etapa la aplicación es transformada a la arquitectura MVC. De acuerdo al procedimiento se ha **planificado la versión MVC** del módulo SIB.ADMINI seleccionado en la etapa anterior, como se presenta en la Figura 3.2.

Tabla 3.2: Planificación MVC del módulo de Administración del SIB.UAN

Nombre del Bundle: SIB.ADMINI	**Descripción:** Las funciones del módulo son crear, ver, actualizar y borrar los objetos de datos que se describen en la fila de clases de entidades gestionadas.		
Entidades gestionadas	Institución (insttc.php), Utilizador (utlzdr.php), operador (oprdr.php), consultante (cnslnt.php). biblioteca (bbltc.php),		
Controlador	Función	Ruta	Página
Inicio	Index	/inicio	admini.php
Institución	Crear	/insttc_admini	insttc.php/c
	Ver	/insttc_admini_ver	insttc.php/v
	Actualizar	/insttc_admini_actualizar	insttc.php/a
	Borrar	/insttc_admini_borrar	insttc.php/b

Tras la planificación del módulo, se **inicializó el proyecto MVC** utilizando Symfony2 a partir del entorno NetBeans 7.4, y fueron configurados los parámetros de conexión del archivo `app/config/parameters.yml` para apuntar a la base de datos de la aplicación fuente, como se muestra en la Figura 3.2. La otra acción realizada fue generar el bundle

referente al módulo seleccionado y planificado. La Figura 3.3 muestra la línea de comando Symfony2 del entorno de Netbeans 7.4 utilizado para generar el referido bundle.

```
2  #Configuraciones de la base de datos Sib1
3  parameters:
4      database_driver: pdo_mysql
5      database_host: 127.0.0.1
6      database_port: null
7      database_name: sib1
8      database_user: root
9      database_password: teste
```

Figura 3.2: Archivo *parametrs.yml* configurado para la base de datos de SIB.UAN

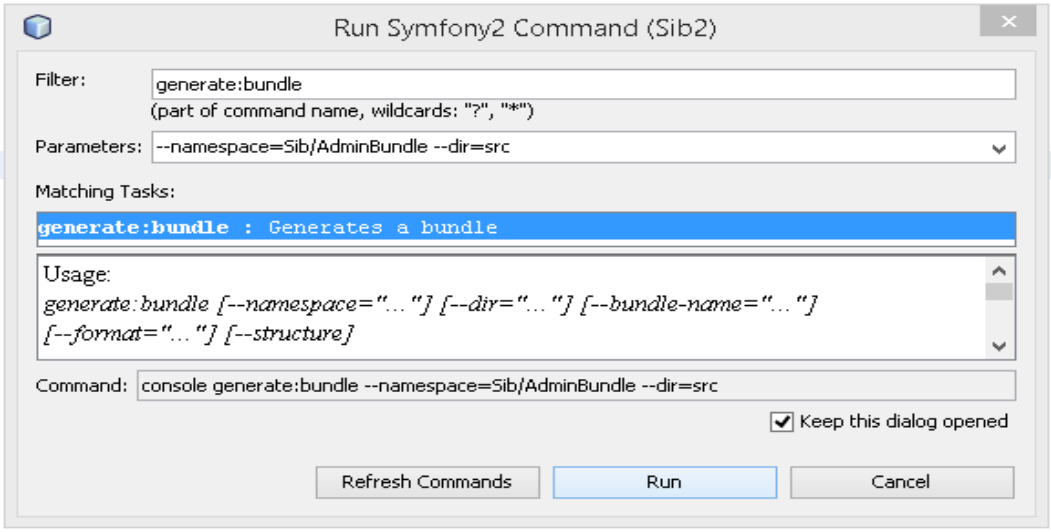

Figura 3.3: Generar bundle SIB.ADMINI a partir de la línea de comando Symfony2

Tras la realización de las descriptas acciones previas se ha logrado crear el modelo de datos a partir de las órdenes "`doctrine:mapping:import`" para los archivos metadatos del esquema de datos de la base de datos del sistema, seguida de las órdenes `doctrine:mapping:convert` y `doctrine:generate:entities` para la generación de las clases de entidades PHP. La Figura 3.4 presenta extracto de una clase de entidad PHP obtenido como resultado final de esta tarea.

```php
class Insttc
{
    /**
     * @var string
     *
     * @ORM\Column(name="nominsttc", type="string", length=100, nullable=true)
     *
     * @Assert\NotBlank()
     *
     */
    private $nominsttc;

    /**
     * @var string
     *
     * @ORM\Column(name="sgl", type="string", length=10, nullable=true)
     *
     * @Assert\NotBlank()
     *
     */
    private $sgl;
```

Figura 3.4: Extracto de código de clase PHP obtenido desde Doctrine2

Fase 3. Ingeniería Inversa para UWA

Las clases PHP obtenidas en la etapa anterior fueron convertidas en un archivo XMI versión 1.0 utilizando el script propuesto en el procedimiento. El archivo XMI generado corresponde a un diagrama de clases UML en la versión 1.3, por lo que fue editado para corresponder a un diagrama de contenido UWA. En seguida se importó el archivo para ArgoUWE para visualización y refinado del diagrama de contenido UWA correspondiente al módulo en análisis.

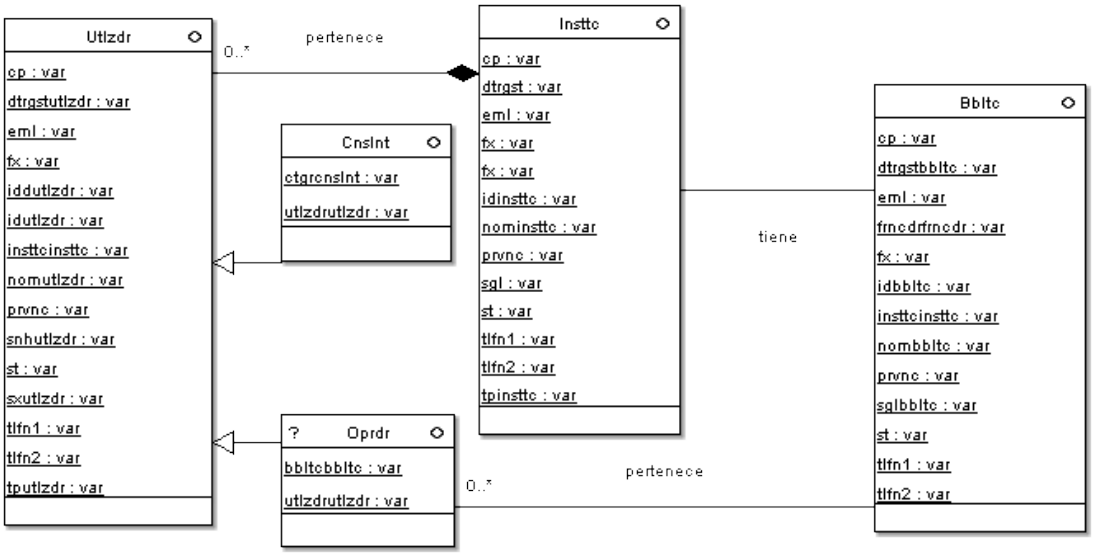

Figura 3.5: Modelo de Contenido UWA del módulo SIBADMINI

El diagrama fue refinado como se presenta en la Figura 3.5, y se le asignó “relevante” a las clases navegables de la aplicación, de modos a generar de manera automática el

modelo de navegación del respetivo módulo, que en seguida fue igualmente refinado, como se muestra en la Figura 3.6.El modelo de navegación fue refinado con la adición de otras clases de navegación, como la clase de *Modulo_Administracion*, que representa el nodo de navegación inicial del módulo en análisis. Además, se ha integrado los procesos de la lógica de negocio, como *login,* necesario para el acceso al nodo del módulo en análisis. Tras el refinado del modelo de navegación, se ha generado el modelo de presentación que especifica los elementos de interface del usuario de cada nodo de navegación como se presenta en la Figura 3.7.

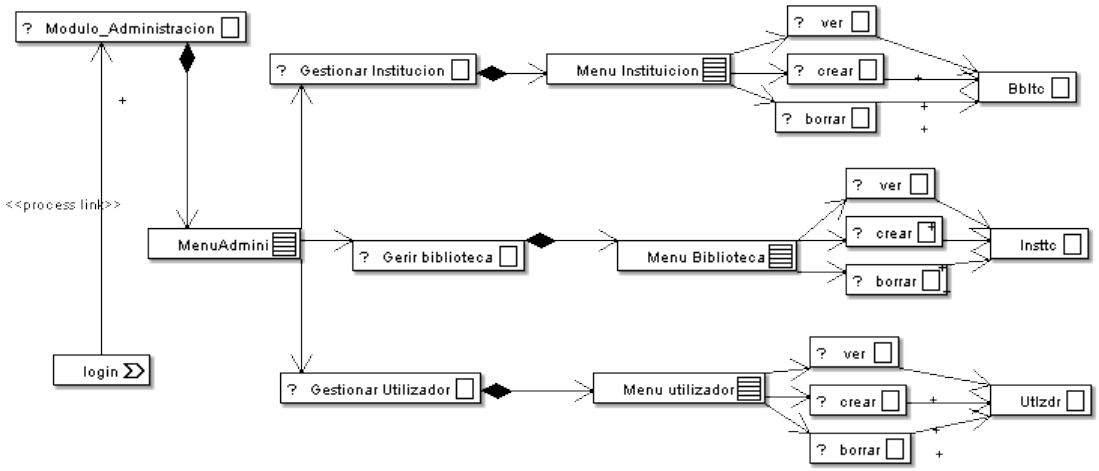

Figura 3.6: Modelo de Navegación UWA del módulo SIB.ADMINI refinado

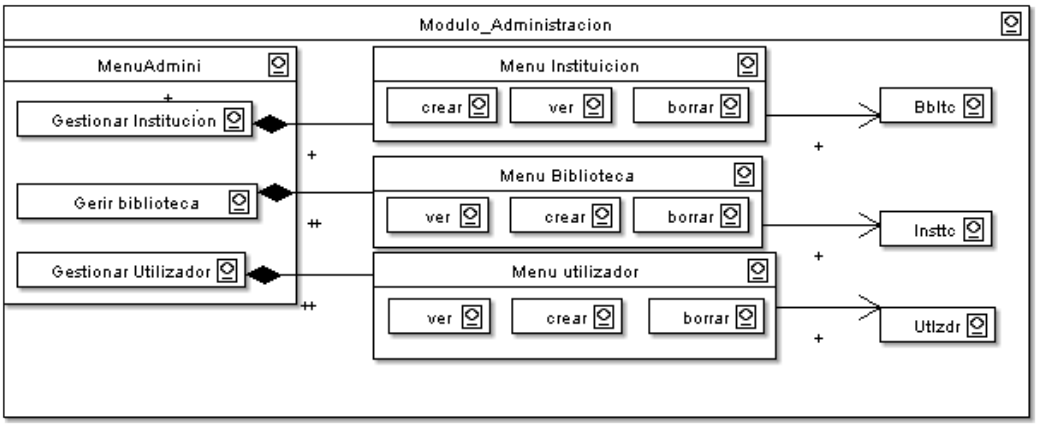

Figura 3.7: Modelo de presentación del módulo SIB.ADMIINI

Finalmente, los procesos añadidos en el modelo de navegación fueron refinados mediante diagramas de actividades. La Figura 3.8 presenta el diagrama de actividades del proceso login del módulo en análisis.

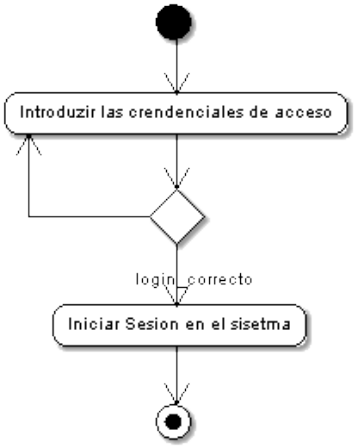

Figura 3.8: Diagrama de actividad del proceso login del módulo SIB.ADMINI

Fase 4. Ingeniería directa

En esta etapa el código de la aplicación fuente de la lógica de negocio (instrucciones PHP) y de la presentación (Código HTML, CSS y JavaScript) fue reutilizado en la versión MVC de la aplicación a través de refactorización basada en la filosofía de Symfony2. La Figura 3.9 muestra extracto del código de la versión original, y la Figura 3.10 presenta el código de la versión MVC de la aplicación.

```
2   <?php
3
4   header('Content-Type: text/html; charset=utf-8');
5   require "ssn_control.php";
6   require "cnct.php";
7
8   if(isset($_POST['cntnr']))
9   {
10    $nom = $_POST['nom'];
11    $sgl = $_POST['sgl'];
12    $tp = $_POST['tp'];
13   $prvnc = $_POST['prvnc'];
14    $tlfn1= $_POST['tlfn1'];
15    $tlfn2= $_POST['tlfn2'];
16    $fx= $_POST['fx'];
17    $cp= $_POST['cp'];
18    $eml= $_POST['eml'];
19    $st= $_POST['st'];
20
21    @$addmais= $_POST['addmais'];
22    @$erro = $_POST['erro'];
23    $msg = "";
24
25  //variável que informará a ocorrência de erros
26  if($erro == "")
27  {
28  //executa aa querys na Base de dados(MySql)
29  $query1 =mysql_query( "insert into endrc
30  values (null, '$prvnc', '$tlfn1', '$tlfn2','$fx','$st','$eml','$cp')" )
31
32  $regist1 =mysql_affected_rows();
```

Figura 3.9: Extracto de código para insertar Institución (versión antigua)

En la versión antigua, el código PHP que maneja el acceso a base de datos bien como la manipulación de los datos se encuentran en el mismo archivo, además, en este mismo está el código HTML para la presentación.

```
40          * Insertar nueva Institución.
41          *
42          * @Route("/", name="insttc_admini_create")
43          * @Method("POST")
44          * @Template("AdminiBundle:Default:rgst_insttc.html.php")
45          */
46         public function createAction(Request $request)
47         {
48             $entity = new Insttc();
49             $form = $this->createCreateForm($entity);
50             $form->handleRequest($request);
51
52             if ($form->isValid()) {
53                 $em = $this->getDoctrine()->getManager();
54                 $em->persist($entity);
55                 $em->flush();
56
57                 return $this->redirect($this->generateUrl('insttc_admini_show'));
58             }
59
60     return array(
61                 'entity' => $entity,
62                 'form'   => $form->createView(),
63             );
```

Figura 3.10: Extracto de código para gestionar Institución (versión MVC)

En la versión MVC, las funciones PHP fueron refactorizados para correspondieren a las acciones (Actions) agrupados en controladores (Controllers) a partir del cual se ejecutan las peticiones del usuario. Por otra parte, el código para la presentación fue separado en archivos HTML, y en seguida fueron igualmente refactorizados con base en la filosofía de Symfony2. Entre las altercaciones, se destacan la jerarquización de los archivos HTML, en el cual se crea una plantilla base que incluye apenas los elementos básicos de presentación subyacentes en todas páginas de la aplicación, bien como el código CSS correspondiente. Posteriormente, cada módulo y/o página extiende la plantilla base y solo se añaden otros elementos de presentación específicos al respectivo módulo y/o página como se muestra en la Figura 3.11

```
1  <?php $view->extend('AdminiBundle::base.html.php') ?>
2
3    <?php $view['slots']->start('stylesheets') ?>
4   <link href="<?php echo $view['assets']->getUrl('bundles/sib/SpryAssets/SpryColl
5   <link href="<?php echo $view['assets']->getUrl('bundles/sib/SpryAssets/SpryMenu
6   <?php $view['slots']->stop() ?>
7
8    <?php $view['slots']->start('wrappercontent') ?>
9
10  <div class="CollapsiblePanelTab"  align="justify" >
11
12  Inserir Instituição </div>
```

Figura 3.11: Archivo HTML que extiende la plantilla base del sistema (versión MVC)

3.4 Evaluación de la mantenibilidad del sistema sometido al procedimiento

El módulo de administración del sistema en análisis ha sido evaluado antes y después de la implementación del procedimiento. Se ha utilizado el Web Applications

Maintainability Model descripto en el epígrafe 1.5, y las métricas del referido modelo fueron calculadas de manera manual con auxilio del IDE NetBeans 7.4 que provee algunas facilidades para el cálculo de determinadas métricas, como por ejemplo el número de líneas de código de las páginas.

En primera instancia se calcularan las métricas a nivel del componente, que representan las páginas web, tanto del servidor como del cliente. En seguida, con base en los resultados de las métricas de componentes, se calcularan las métricas del sistema, que totalizan las métricas de todos los compontes involucrados en la medición. Por razones de brevedad solo se presentan en el documento el cálculo de las métricas a nivel del sistema, que en este caso corresponde al módulo SIB.ADMINI. Los resultados obtenidos se muestran en la Tabla 3.3.

Tabla 3.3: Evaluación de la mantenibilidad del módulo SIB.ADMINI

ATRIBUTO	**MÉTRICA**	**MÓDULO SIB.ADMINI**	
TAMAÑO		Antes	Después
	TotalServerPage# (1)	12	6
	TotalClientPage# (2)	24	24
	TotalWebPage# = 1 + 2	36	30
	TotalServerPageLoc# (a)	2088	1474
	TotalStaticClientPageLoc# (b)	5446	2061
	TotalLOC# = a + b	7534	3535
	TotalBuiltClientPages#	6	6
	WebObject#	6	6
	TotalLanguages#	4	4
MODULARIDAD	Promedio de código por página = TotalLOC#/ TotalWebPage#	209.3	117.83
CONTROL DE ACOPLAMIENTO	TotalWebPageRelationships# / TotalWebPage#	24/36 = 0.7	24/30 = 0.8
REUSABILIDAD DE COMPONENTE	Include# / TotalWebPage#	26/36= 0.7	37/30 = 1.2

Las métricas descriptas se centran en los atributos considerados relevantes de acuerdo a las principales alteraciones realizadas en el código de la aplicación tras la aplicación del procedimiento. En particular, fueron considerados como atributos claves, el tamaño, la modularidad, el control de acoplamiento y la reusabilidad del módulo.

- El tamaño está relacionado con la cantidad de páginas y respectivo número de líneas de código que conforman la aplicación. Las aplicaciones de mayor tamaño suelen ser las más complejas de mantener.
- La modularidad es un indicador de reusabilidad de la aplicación. Las aplicaciones complejas suelen ser desarrolladas por módulos o componentes que pueden ser reutilizados. Esta características no solo brinda la reutilización, sino la facilidad de mantenimiento, a medida facilita en el mantenimiento de característica que involucran varias páginas de la aplicación. Este atributo se mide a través del promedio de líneas de código por página. Cuantas más líneas de código poseyeren las páginas de la aplicación se supone que existe poca reutilización en las páginas de la aplicación, lo que podrá contribuir para considerable esfuerzo en las alteraciones genéricas de la aplicación.
- El control acoplamiento indica el nivel de relaciones entre las páginas de la aplicación. Un elevado nivel de acoplamiento puede causar dificultad de mantenimiento.
- La reusabilidad de componente que está relacionado con la modularidad, sin embargo, este atributo indica qué nivel los componentes de la aplicación están a ser reutilizados.

En este contexto, el análisis de los resultados obtenidos constató que el sistema antes de la aplicación del procedimiento era de mayor tamaño y consecuentemente más difícil de mantener que su nueva versión, debido principalmente al mayor número de TotalLOC# y en particular el número de TotalServerPageLOC#. Este resultado se debe a falta de reutilización del código de la presentación en la versión original del sistema, lo que ha sido comprobado a través del cálculo de la modularidad, donde la versión original del sistema poseía un promedio de 209.3 líneas de código por página, mientras en la nueva versión se acortó para 117.83.

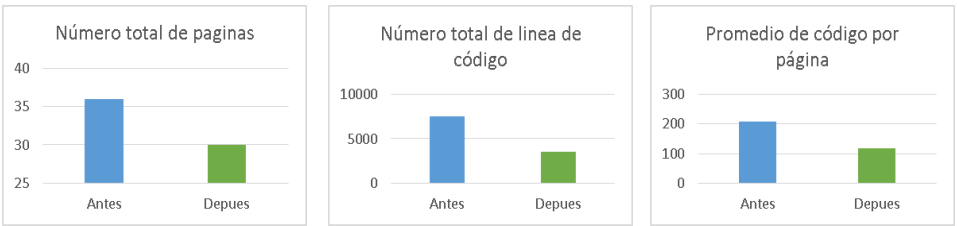

Figura 3.12: Tamaño del SIB.ADMINI antes y después del procedimiento

Los gráficos agrupados en la Figura 3.12 muestran un resumen de la comparación de la mantenibilidad de la aplicación con respecto a su tamaño, donde claramente se constata la reducción del tamaño de la aplicación en su nueva versión, lo que se traduce en la reducción de la esfuerzo de mantenimiento.

Relativamente al control del acoplamiento, los resultados muestran que antes el sistema poseía 0.7 de acoplamiento, menor que la actual versión, que es de 0.8. El acoplamiento de la aplicaciones Web está relacionado con la cantidad de relaciones entre las páginas que a conforman. El elevado acoplamiento puede causar un impacto negativo en la mantenibilidad de aplicaciones Web debido al número de conexiones que puede ser difícil de controlar. En esta perspectiva, la nueva versión parece ser más difícil de mantener, sin embargo, no es una diferencia considerable, y si se considera otro importante atributo de mantenibilidad, que es la reusabilidad de componentes del sistema, donde según los datos, la nueva versión es considerablemente más reusable que la versión anterior, como se ilustra en la Figura 3.13. Esto explica de forma parcial porque esta versión tiene un mayor acoplamiento que la anterior, y entretanto brinda mayor reusabilidad de componentes, lo que contribuye considerablemente para su facilidad de mantenimiento.

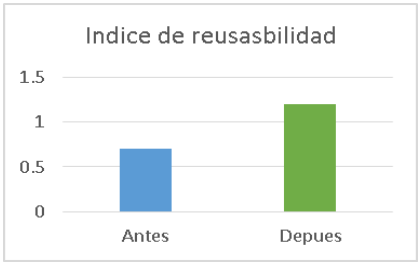

Figura 3.13: Reusabilidad de componentes del SIB.ADMINI

3.5 Resultados obtenidos

Luego de realizada la reingeniería al Sistema Integrado de Bibliotecas de la UAN aplicando el procedimiento propuesto se obtuvo importantes resultados, entre los cuales se destacan los siguientes:

- Se recuperó una adecuada y actualizada documentación del diseño del sistema referente al módulo SIB.ADMINI, lo que ciertamente contribuirá para la facilidad de mantenimiento y evolución del sistema

- Se transformó la arquitectura de la aplicación para el patrón de diseño MVC, uno de los estándares de arquitectura ampliamente aceptados en el desarrollo de software, en particular aplicaciones Web, que a su vez contribuirá igualmente para mayor mantenibilidad del sistema
- Se migró y mejoró el código del sistema, tanto a nivel del servidor como del cliente, mediante la refactorización de acuerdo a filosofía de Symfony2, que actualmente es uno de los frameworks ampliamente utilizado para el desarrollo de aplicaciones Web en PHP, con aporte para la aplicación de buenas prácticas de desarrollo de software, además del número considerable de librerías de gran utilidad.

Conclusiones parciales

- El procedimiento propuesto para la reingeniería de aplicaciones Web heredadas fue exitosamente aplicado al Sistema de Bibliotecas de la UAN
- La especificación detallada del procedimiento, desde el entorno de desarrollo, flujo de proceso de las actividades y respectivas tareas, bien como la definición de los roles involucrados en el proceso permitió guiar el proceso de forma segura.
- La disponibilidad y simplicidad de las herramientas propuestas en el procedimiento también fueron claves para que lograse la implementación del proceso, a medida que son todas de carácter académico y cuentan con aporte de una gran comunidad de desarrolladores.
- El análisis de mantenibilidad arrojó resultados favorables fundamentalmente en la documentación del diseño del sistema, y las características internas del sistema, como el diseño de la arquitectura.
- El éxito en la aplicación del procedimiento y los resultados obtenidos con respecto a mantenibilidad del sistema contribuyó a demostrar la aplicabilidad del procedimiento y su contribución al incremento de la mantenibilidad de aplicaciones Web heredadas implementadas en PHP.

CONCLUSIONES

La investigación realizada permite llegar a las siguientes conclusiones:

1 Se elaboró el marco teórico sobre el desarrollo y mantenimiento de aplicaciones Web, y procesos de reingeniería de software, lo que permitió comprobar la necesidad del proceso en el mantenimiento de las aplicaciones Web e identificar los elementos claves del proceso de reingeniería en el dominio de las aplicaciones Web

2 Se diseñó un procedimiento para la reingeniería de aplicaciones Web en la UAN que contribuirá para incrementar la mantenibilidad de dichas aplicaciones

3 Se aplicó el procedimiento propuesto al Sistema Integrado de Bibliotecas de la UAN, lo que permitió la reconstrucción de la aplicación en una versión mejorada

4 Se validó la propuesta a partir de la aplicación del proceso seguida de la evaluación y comparación de la mantenibilidad del sistema antes y después de la aplicación del procedimiento.

RECOMENDACIONES

El autor recomienda para los estudios futuros el siguiente:

- Extender la librería PHP_UML para permitir la generación de clases PHP para diagrama de contenido UWA de forma directa, lo que contribuirá para mayor agilidad en el proceso de ingeniería inversa, dado que ya no será necesario la edición de los archivos XMI que representan los diagramas de clase UML, pasando estos a representar directamente los diagramas de contenido UWE
- Especificar reglas de trasformaciones en ATL o QVT que permitan la generación automática de código PHP desde los modelos UWE recuperados, lo que contribuirá para mayor agilidad en la etapa de ingeniería directa, a media que obtendrá de manera automática un prototipo del sistema a partir de los modelos recuperados.

REFERENCIAS BIBLIOGRÁFICAS

[1] R. S. Pressman y D. Lowe, Web Engineering: A Practitioner's Approach, vol. 1st ed., New York: McGraw-Hill Higher Education, 2009.

[2] S. Murugesan, «Web Application Development: Challenges and the Role of Web Engineering,» de *Web Engineering: Modelling and Implementing Web Applications*, Londres, Springer, 2008, pp. 7-32.

[3] H. M. Kienle y D. Distante, «Evolution of Web Systems,» de *Evolving Software Systems*, Berlin Heidelberg, Springer, 2014, pp. 201-228.

[4] E. Ghosheh, «A Novel Model for Improving the Maintainability of Web-Based Systems,» 2010.

[5] M. M. Lehman, «Programs, life cycle, and laws of Software Evolution,» *Proceedings of the IEE,* vol. 68, nº 9, pp. 1060-1076, Setiembre 1980.

[6] L. S. Maurya y G. Shankar, «Maintainability assessment of web based application,» *Journal of Global Research in Computer Science - JGRCS,* vol. 3, nº 7, pp. 30-33, 7 Julio 2012.

[7] S. Murugesan y A. Ginige, «Web Engineering: Introduction and Perspectives,» de *Software Engineering for Modern Web Applications: Methodologies and Technologies*, New York, Information Science Reference, 2008, pp. 1-24.

[8] R. S. Pressman, Ingeniería de Sotware: Un Enfoque Práctico, Sexta edción ed., Europa: Mc Graw Hill, 2005.

[9] S. Murugesan, Y. Deshpande, S. Hansen y A. Ginige, «Web Engineering: A New Discipline for Development,» de *Web engineering*, Springer Berlin Heidelberg, 2001, pp. 3-13.

[10] G. A. Di Lucca, A. R. Fasolino y P. Tramontana, «Reverse engineering Web applications: the WARE approach,» *Journal of software maintenance and evolution: research and practice,* nº 16, pp. 71-101, 2004.

[11] F. Ricca y T. Paolo, «Program Transformations for Web Application Restructuring,» de *Web Engineering: Principles and Techiques*, W. Sush, Ed., London, Idea Group Inc., 2005, pp. 242-260.

[12] F. Ricca y P. Tonella, «Anomaly detection in web applications: a review of already,» *European Conference on Software Maintenance and Reengineering,* pp. 385-394, 2005.

[13] P. Tramontana, «Reverse Engineering of Web Applications,» 2007.

[14] M. L. i. Bernard, G. A. Di Lucca y D. Distante, «Improving the Design of Existing Web Applications,» *Quality of Information and Communications Technology (QUATIC),* pp. 499-409, 2010.

[15] L. H. Rosenberg, «Software Re-engineering,» *Software Assurance Technology Center.*

[16] I. Sommerville, Software Engineering, 9ª Edición ed., Addison-Wesley, 2011.

[17] E. J. Chikofsky y J. H. Cross II, «Reverse Engineering and Desing Recovery: A Taxonomy,» *IEEE Software (January),* pp. 13-17, 1990.

[18] F. G. Tosca y R. M. Fernánadez, «¿Reingeniería de software, un camino o el camino?,» *La Nueva Gestión Organizacional,* pp. 13-25, Enero-Junio 2009.

[19] J. Bergey, D. Smith, S. Tilley, N. Weiderman y S. Woods, «Why Reengineering Projects Fail,» 1999.

[20] R. Pérez-Castillo, I. G.-R. d. Guzmán, M. Piattini y C. Ebert, «Reengineering Technologies,» *IEEE,* pp. 13-17, 2011.

[21] R. R. Echeverría, «Un Proceso de Modernización de Aplicaciones Web a RIA,» 2013.

[22] W. M. Ulrich y J. Smith, «Architecture-Driven Modernization,» 11 2015. [En línea]. Available: http://adm.omg.org/. [Último acceso: 11 2015].

[23] M. L. Bernardi, G. A. Di Lucca, D. Distante y M. Cimitile, «Model Driven Evolution of Web Applications,» *Web Systems Evolution (WSE), 2013 15th IEEE International Symposium on,* pp. 45-50, 27 setiembre 2013.

[24] F. Bianchiotti y S. Casas, «Guía para la Reingeniería de Sistemas Legados: Una Experiencia Práctica y Real,» *Revista Latinoamericana de Ingeniería de Software,* vol. 2, nº 2, pp. 99-106, 2014.

[25] S. Tilley y S. Huang, «Evaluating the reverse engineering capabilities of web tools for understanding site content and structure: A case study,» *23rd International Conference on Software Engineering,* pp. 514-523, 2001.

[26] A. Ginige y S. Murugesan, «Web Engineering: An Introduction,» *IEEE MultiMedia,* pp. 14-18, 2001.

[27] P. Bourque y R. E. Fairley, Edits., Guide to the Software Engineering (SWEBOK) version 3.0, IEEE Computer Society, 2014.

[28] L. Shklar y R. Richard, Web Application Architecture: Principles, protocols and practices, London: John Wiley & Sons, Ltd, 2003.

[29] M. D. Acyntho, D. Schwabe y G. Rossi, «A software architecture for structuring complex web applications,» *Journal of Web Engineering,* vol. 1, nº 1, pp. 37-60, 2002.

[30] J. Deacon, «Model-View-Controller (MCV) Archiecture,» 2009. [En línea]. Available: http://www.jdl.co.uk/briefings/indes.html#mvc. [Último acceso: 27 agosto 2015].

[31] Y. Ping, K. Kontogiannis y T. C. Lau, «Transforming Legacy Web Applications to the MVC Architecture,» *Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice,* 2004.

[32] S. Burbeck, «Applications Programming in Smalltalk-80™: How to use Model-View-Controller (MVC),» ParcPlace Systems, Inc, 1992.

[33] S. 'Uyun y M. R. Ma'arif, «Implementation of model view controller (MVC) architecture,» *Seminar Nasional Aplikasi Teknologi Informasi 2010 (SNATI 2010),* pp. 47-50, 2010.

[34] A. Ginige, «Web Engineering: Managing the Complexity of Web Systems Development,» *SEKE '02,* pp. 721-729, 15-19 Julio 2002.

[35] M. Brambilla, S. Ceri y P. Fraternali, «Process Modeling in Web Applications,» *ACM Transactions on Software Engineering and Methodology (TOSEM),* vol. 15, nº 4, pp. 360-409, 2006.

[36] T. Conte, E. s. Mende y G. H. Travassos, «Processos de Desenvolvimento para Aplicações: Uma Revisão Sistemática,» *ACM,* 2005.

[37] G. Rossi, D. Schwabe, L. Olsina y O. Pastor, «Overview of Design Issues for Web Applications Development,» de *Web Engineering: Modelling and Web Engineering: Modelling and*, G. Rossi, O. Pastor, D. Schwabe y L. Olsina, Edits., Londres, Springer, 2008, pp. 49-64.

[38] G. Rossi y D. Schwabe, «Modeling and implementing web modeling and implementing web,» de *Web Engineering: Modelling and Implementing Web Applications*, Londres, Springer, 2008, pp. 109-156.

[39] S. Ceri, P. Fraternali y A. Bongio, «Web Modeling Language (WebML): a modeling language for designing Web sites,» *Computer Networks,* nº 33 , p. 137–157, 2000.

[40] L. Baresi y S. Colazzo, «A modelling notation for complex Web applications,» de *Web Engineering*, Springer Berlin Heideberg, 2006, pp. 335-364.

[41] N. K. A. Koch, «The Expressive Power of UML-based Web Engineering,» *Second International Workshop on Web-oriented Software Technology (IWWOST02).,* vol. 16, April 2002.

[42] M. L. Bernardi, M. Cimitile, D. Distante y F. Mazzone, «Web applications design evolution with UWA,» *12th IEEE International Symposium on Web Systems Evolution (WSE),* pp. 3-10, 2010.

[43] A. Kraus, «Model Driven Software Engineering for Web Applications,» vorgelegt, 2007.

[44] N. Koch, S. Koch, N. Moreno-Vergara, V. Pelechano-Ferragud, F. Sánchez-Figueroa, Vara-Mesa y Juan-Manuel, «Model-Driven Web Engineering,» *UPGRADE: The European Journal for the Informatics Professional,* vol. IX, nº 2, pp. 20-25, abril 2008.

[45] N. Moreno, J. R. Romero y A. Vallecillo, «An overview of model-driven web engineering and the MDA,» de *Web Engineering: Modelling and Implementing Web Applications*, London, Springer, 2007, pp. 353- 382.

[46] IEEE, «Mantenimiento de software,» de *IEEE*, 1993.

[47] ISO, «SO/IEC 12207:2008 - Systems and software engineering — Software life cycle processes,» 2008. [En línea]. Available: https://www.iso.org/obp/ui/#iso:std:iso-iec:12207:ed-2:v1:en. [Último acceso: 10 2015].

[48] IEEE, «Software Engineering — Software Life Cycle Processes — Maintenance». Patente ISO/IEC 14764:2006, 2006.

[49] J. Foster, «Cost Factors in Software Maintenance,» 1993.

[50] K. Bennet y V. Rajlich, «Software Maintenace and Evolution,» *Sofware Engineering Limerick Ireland,* pp. 73-87, 2000.

[51] S. Ghosh, S. K. Dubey y A. Rana, «Comparative Study of the Factors that Affect Maintainability,» *International Journal on Computer Science and Engineering (IJCSE),* vol. 3, pp. 3763-3769, 12 Deciember 2011.

[52] I. G.-R. de Guzmán, M. Polo y M. Piattini, «Estado del arte de la reingeniería y la ingeniería inversa: 2001-2003,» 2004. [En línea]. Available: http://www.academia.edu/download/30484593/30.pdf. [Último acceso: 15 noviembre 2014].

[53] A. S. Abbas, W. Jeberson y V. Klinsega, «The Need of Re-engineering in Software Engineering,» *IJET (International Journal of Engineering and Technology) ,* vol. II, nº 2, pp. 292-295, febrero 2012.

[54] R. Pérez-Castillo, I. G. R. de Guzmán, F. García y M. Piattini, «A Practical Teaching Experience about Software Reengineering,» *Procedia - Social and Behavioral Sciences,* nº 83, pp. 254-260, 2013.

[55] R. S. Arnold, «Software Reengineering,,» *IEEE Press,* 1992.

[56] D. Buzatto, «Formalizacao de um modelo de processo de reengenharia centrado no usuario para conversao de aplicaccoes desktop em RIAS,» Sao Carlos, 2010.

[57] J. C. Á. García, M. M. Sánchez y M. N. M. García, «Metodología de reingeniería del software para la remodelación de aplicaciones científicas heredadas,» Departamento de Informática y Automática - Universidad de Salamanca, Salamanca, 2004.

[58] Soley, Richard, «Model Driven Architecture,» OMG, 2000.

[59] OMG, «OMG Formal Specifications,» 2015. [En línea]. Available: http://www.omg.org/spec/index.htm. [Último acceso: 11 2015].

[60] J.-M. Favre, «Foundations of model (driven)(reverse) engineering: Models,» *International Seminar on Language Engineering for Model-Driven Software Development,* pp. 1-31, 2005.

[61] R. Max, «From legacy web applications to webml models,» Wien, 2009.

[62] M. L. Bernardi, M. Cimitile y D. Distante, «Web Applications Design Recovery and Evolution with RE-UWA,» *Journal of software maintenance and evolution: research and practice,* pp. 1-25, 2013.

[63] M. L. Bernardi, G. A. Di Lucca y D. Distante, «The RE-UWA Approach to Recover User Centered Conceptual Models from Web Applications,» *International Journal on Software Tools for Technology Transfer,* pp. 485-501, 2009.

[64] D. Distante, G. Rossi, P. Pedone y G. Canfora, «Model-Driven Development of Web Applications with UWA, MVC and JavaServer Faces,» *7th International Conference on Web Engineering (ICWE2007),* pp. 457-472, 2007.

[65] E. Ghosheh, J. Qaddour, M. Kuofie y S. Black, «A Comparative Analysis of Maintainability Approaches for Web Applications,» *IEEE - International*

Conference on Computer Systems and Applications, vol. 1, nº 4244, pp. 1155-1158, 2006.

[66] J. Eguiluz, «Desarrollo web ágil con symfony2,» Symfony - Trademark, 2012.

[67] F. Potencier y R. Weaver, «Symfony 2.0, el libro oficial,» 2015. [En línea]. Available: http://librosweb.es/libro/symfony_2_0/. [Último acceso: 12 10 2015].

[68] T. Pessini y S. V. F., «Avaliando metodologias de desenvolvimento de aplicações web,» *Revista Eletrônica Científica Inovação e Tecnologia,* vol. 02, nº 02, pp. 12-16, 2013.

[69] A. Kraus, A. Knapp y N. Koch, «Model-Driven Generation of Web Applications in UWE,» *MDWE,* vol. 261, 2007.

[70] A. Knapp, N. Koch, M. Flavia y G. Zhang, «ArgoUWE: A CASE Tool for Web Applications,» *Engineering Methods to Support Information Systems Evolution (EMSISE'03),* pp. 37-50, 2003.

[71] C. Kroiß, N. Koch y S. Kozuruba, «UWE Metamodel and Profile 1.9 - User Guide and Reference,» München, Germany, 2011.

[72] N. Koch, A. Knapp, G. Zhang y H. Baumeister, «UML-based web engineering: An Approach Based on Standards,» de *Web Engineering: Modelling and Implementing Web Applications*, Londres, Springer, 2008, pp. 157-192.

[73] J. Maras y A. Petričić, «Reverse engineering legacy Web applications with phpModeler,» *Mälardalen University Software Enginnering Workshop (MUSE'09),* 2009.

[74] «Pear:: Manual Pear,» PEAR Documentation Group, 2015. [En línea]. Available: http://pear.php.net/manual/en/. [Último acceso: 8 2015].

[75] L. F. B. Lutete y P. F. D. L. V. L. Calado, «Sistema Integrado de Bibliotecas para UAN,» Luanda, 2013.

ANEXOS

Anexo 1: Lista de la aplicaciones web desarrolladas en la UAN de 2012-2013

Software	Tipo de software	Tecnóloga utilizada	objetivo
Sistema Integrado de gestión académica universitaria de la UAN	Aplicación Web	PHP	Automatizar o proceso universitario, desde el acceso, matrícula y otros procesos académicos
Sistema de reconocimiento de estudios de UAN	Aplicación web	PHP	Automatizar el proceso de reconocimiento de estudios realizados en exterior del país
Sistema integrado de Recursos Humanos de la UAN	Aplicación web	PHP	Biblioteca de documentos electrónicos de la UAN
Sistema Integrado de Bibliotecas de la UAN	Aplicación web	PHP	Automatizar el proceso gestión bibliotecaria de las bibliotecas de la UAN

Anexo2: Encuesta para el diagnóstico del mantenimiento de las aplicaciones Web en la UAN

1. ¿Cuál el grado de dificultad con que se realiza el mantenimiento de los software en análisis?

   ___Muy alta; ___Alta; ___Regular; ___Baja; ____Muy baja

2. ¿Qué factores han influido negativamente en la mantenibilidad de los software en análisis?

____ La falta de personal para dedicar a esta actividad.

____ La ausencia del personal que desarrolló inicialmente el software (Movilidad)

____Baja calidad del software

____ Falta de documentación y especificaciones de diseño o la existente es incomprensible, incorrecta o insuficiente.

____ El esfuerzo que se consume en la ejecución del mantenimiento

___Otro. Diga Cuál: _____

3. ¿Considera necesaria la definición de un procedimiento de reingeniería para incrementar la mantenibilidad de los software en análisis?

   Si_____; No_____; No sé ____

Anexo 3: Perfil de los encuetados en el departamento de Ciencias de computación de la UAN

Nro.	Nombre	Función	Cargo
1	Dr. Padoca Calado	Profesor Titular de Ingeniería de Software e Inteligencia artificial	Jefe del departamento y Líder de Proyectos
2	Dra. Darlines Rodríguez	Profesora Auxiliar de Ingeniería de Software	Líder de Proyectos
3	MrsC. Dikiefu Fabiano	Profesor Auxiliar de Programación y sistemas distribuidos	Líder de Proyectos
4	MrsC. Antonio Vicente	Profesor Auxiliar de Inteligencia Artificial	Líder de Proyectos
5	Lic. Sampaio Velho	Profesor Asistente de Programación	Mantenedor (senior)
6	Lic. Dizando Norton	Profesor Asistente de Matemática Computacional	Mantenedor (senior)
7	Élio Santana	Estudiante del 5 año	Mantenedor (senior)
	Kissema Eduardo	Estudiante del 5 ano	Mantenedor (senior
8	Mauro Fernandes	Estudiante del 5 ano	Mantenedor
9	Ricardo Costa	Estudiante del 4 año	Mantenedor
10	María Bastos	Estudiante del 4 ano	Mantenedor
11	Ana Goureth	Estudiante del 4 año	Mantenedor
12	Reginaldo Santos	Estudiante del 5 año	Mantenedor
13	Zenildo Roberto	Estudiante del 4 año	Mantenedor

Anexo 3: Resultados de la Encuesta

Fecha: 10-14 de Agosto del 2015

Lugar: Departamento de Ciencias de la Computación de la UAN – Luanda – Camama