UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Herramienta para generar productos de trabajos de la metodología variación AUP-UCI

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

Autor:
Drymon Alfonso Benítez

Tutora:

Ing. Claudia Bravo Batista

La Habana, junio de 2017
"Año 59 de la Revolución"

DECLARACIÓN JURADA DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Drymon Alfonso Benítez	**Ing. Claudia Bravo Batista**
Autor	Tutora

Datos de Contacto

Claudia Bravo Batista: Ingeniero en Ciencias Informáticas, Graduado en el 2012.

DEDICATORIA

A mi mamá, por haber estado a mi lado siempre apoyándome en las buenas y en las malas, por la confianza que depositó en mí, por darme los mejores consejos, que pese a toda la guerra que di, tuvo paciencia, supo guiarme y orientarme de forma correcta, por darme tanto amor y ser la mejor mamá del mundo.

A mi papá por haber sido mi ejemplo a seguir durante toda mi vida, por haber luchado por mí desde el principio sin rendirse, por haberme enseñado tantas cosas e inculcarme sus creencias morales, espirituales y revolucionarias desde pequeño.

A mis abuelas Zoila y Georgina Inés por haber estado siempre pendientes de mí y de mi trayecto, por el apoyo constante y los ánimos para llegar hasta donde estoy hoy y estar dispuestas siempre a ayudarme en lo que necesité.

A mi abuelo Pastor, por haberme dado su cariño incondicional desde mis primeros días de vida hasta hoy y por no fallarme nunca, por ser el que me guió en mis primeros pasos y me inició en esta hermosa ciencia que es la informática, por ser el abuelo que tanto necesité y mi amigo, por haber estado dispuesto a ayudarme en todo y ser un ejemplo de persona con grandes de valores humanos y patrióticos.

A mi hermano Bryan Miguel, por haberme brindado el privilegio de ser el hermano mayor y ejemplo a seguir de una persona tan cariñosa, amable y gentil, por darme su apoyo moral siempre. Quisiera que este logro le sirva de incentivo para que siga luchando por sus propias metas y le sirva como ejemplo de que con sacrificio y constancia, sin perder de vista el objetivo, pueden lograrse cosas increíbles..

AGRADECIMIENTOS

A todo el tribunal, por sus correcciones y sugerencias con el objetivo de aumentar la calidad del trabajo de diploma presentado.

A toda mi familia que me ha estado apoyando durante todo este proceso.

A toda la comunidad de amigos, los que están hoy con nosotros y los que no, de forma general por haber compartido todos estos años juntos. A Liorge, Nivaldo, Flavio, Marcos, Daniel, Soler, Eugenio, el Ferry, el Ruso... por escuchar mis problemas, darme sus consejos y brindarse a ayudarme en lo que pudieran. Por todos los momentos que pasamos y estar ahí siempre dispuestos a tomarnos un café.

A Yoandry Freire Pérez por tener tanta paciencia y explicarme tantas cosas de Synfony2.

A Jorge Lázaro por ser tan buen amigo y aguantarme durante tantos años.

A Raidel por estudiar conmigo siempre el día antes de los exámenes y prestarme su laptop siempre que la necesité para ultimar detalles.

A Adrián por ser más que un amigo, casi un hermano mayor durante todo este tiempo y brindarme su apoyo y consejo cuando más lo necesité.

A Julio por su enorme ayuda, dedicarme tanto tiempo extra y su guía certera durante todo el proceso de implementación de la aplicación.

A todo el equipo de desarrollo del laboratorio 10 del centro CEIGE: a Daniel y Pavel por compartir conmigo cinco años y brindarme su ayuda como estudiantes y ahora como profesionales. A Dannel, Daniel Arturo, y a todos en general... por ser también mis compañeros durante todo este proceso.

Por último y muchísimo más importante a mi tutora Claudia Bravo Batista, por su enorme paciencia a pesar de mis locuras, por sus regaños, por las noches sin dormir que pasamos juntos y por tantas correcciones. Por brindarme su apoyo incondicional en todo momento y estar siempre halándome las orejas. Por su enorme ayuda tanto en la implementación de la solución como en la documentación de la misma. Porque sin su ayuda no hubiera podido terminar este proceso de forma satisfactoria. En conclusión: por ser la mejor tutora del mundo.

RESUMEN

El presente trabajo tiene como objetivo desarrollar una herramienta informática que permita generar la documentación correspondiente a los productos de trabajo de la disciplina Modelado de Negocio de la metodología de desarrollo de software AUP-UCI. Con la solución propuesta se pretende facilitar y lograr que se estandaricen los documentos según el formato estándar que genera la herramienta. Se definieron las herramientas, tecnologías y lenguajes de programación a utilizar en la implementación de la solución y se identificaron los requisitos funcionales con que contaría la misma mediante técnicas de captura y validación de requisitos. Además, se identificaron los principales elementos de la metodología variación AUP-UCI, para definir el alcance de la investigación. Se describen los estilos arquitectónicos, patrones de diseños y estándares de codificación utilizados en la implementación de la solución propuesta. Y se evalúa por separado el correcto funcionamiento de los códigos que componen la aplicación mediante pruebas unitarias de software con enfoque de caja blanca y caja negra.

Palabras clave: disciplina, estándar, metodología de desarrollo de software, modelado de negocio, productos de trabajos.

Abstract

The present work aims to develop a computer tool to generate the documentation corresponding to the work products of the Business Modeling discipline of the AUP-UCI software development methodology. With the proposed solution is intended to facilitate and achieve standardization of documents according to the standard format generated by the tool. The tools, technologies and programming languages to be used in the implementation of the solution were defined and the functional requirements were identified using the capture and validation techniques. In addition, the main elements of the AUP-UCI variation methodology were identified to define the scope of the research. It describes the architectural styles, design patterns and coding standards used in the implementation of the proposed solution. And separately evaluate the correct functioning of the codes that compose the application by means of unit tests of software focused on white box and black box.

Keywords: business modeling, disciplines, software development methodology, standard, work products.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

El empleo de una metodología impone un proceso disciplinario sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. La Universidad de las Ciencias Informáticas (UCI) cuenta con 14 centros productivos, logrando converger a una metodología que se adapte al ciclo definido para su actividad productiva. Esta metodología es una variación de AUP, la cual tiene entre sus objetivos aumentar la calidad del software, aplicando buenas prácticas, sustentadas en el Modelo CMMI-DEV v1.3 (Tamara, 2015).

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio con el uso de técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Divide el proceso de desarrollo en ciclos, en cada uno, se obtiene un producto final y se divide en cuatro fases: Inicio, Elaboración, Construcción y Transición, que concluye con un hito bien definido. Para el ciclo de vida de los proyectos de la UCI se decide mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes tres fases en la fase de Ejecución y se agrega la fase de Cierre (Tamara, 2015).

AUP propone siete disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno). Se decide para el ciclo de vida de los proyectos de la UCI mantener las mismas disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajo: Modelado de negocio, Requisitos, Análisis y Diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (Tamara, 2015).

El Modelado de negocio propone tres variantes a utilizar en los proyectos: Casos de Uso del Negocio (CUN), Descripción de Proceso de Negocio (DPN) o Modelo Conceptual (MC). Además, existen tres formas de encapsular los requisitos: por Casos de Usos del Sistema (CUS), Historias de Usuarios (HU) y Descripción de requisitos por proceso (DRP). Surgiendo cuatro escenarios para modelar el sistema en los proyectos. Para cada escenario se generan diferentes productos de trabajos definidos en el Expediente de proyecto, independientemente del tipo de proyecto en base a las particularidades del mismo[1] (Tamara, 2015).

[1] Los productos de trabajo se encuentran en documentos publicados en el excriba.prod.uci.cu y en mejoras.prod.uci.cu.

En un análisis realizado a los proyectos de la universidad por el departamento de Calidad, como se representa en el Anexo1 Escenarios que aplican los proyectos que utilizan la metodología AUP-UCI. Se obtiene que para una muestra de 25 proyectos, cinco proyectos no modelan negocio que representa el 20% de la muestra, 13 proyectos modelan negocio con MC que representa el 52% de la muestra, cinco proyectos modelan negocio con DPN que representa el 20% de la muestra y dos proyectos modelan negocio CUN que representa el 8% de la muestra. Por tal motivo, la cantidad de productos de trabajos que se generan de la variación de la metodología AUP para los proyectos productivos de la UCI, continúa siendo exhaustiva, así como sus correspondientes procesos de revisión y actualización. Provocando que en la mayoría de los casos la planificación y las estimaciones de tiempo incurran en desviaciones y aplazamientos de los hitos de proyectos.

En las entrevistas realizadas a especialistas de los proyectos Centro de Informática Médica, Nova Servidores y Arquitectura de Referencia de PHP Bosón, como se representa en el Anexo 2 Entrevistas realizadas a los especialistas de diferentes centros de producción de la UCI, coincidían que los documentos generados de los productos de trabajos de la metodología AUP-UCI en los proyectos, no mantenían el mismo estándar. Presentando como principales deficiencias: las fuentes, estilos y tamaños de letras no se ajustaban a las definidas en el modelo del producto de trabajos propuesto por el centro de Calidad de la Universidad, detectada como no conformidad en las revisiones realizadas. Las imágenes digitales presentan diferentes formatos, cada uno se corresponde con una extensión específica que lo contiene, siendo las más utilizadas: BMP, GIF, TIF y PNG, las cuales no se correspondía con el tamaño y posicionamiento que se requiere en el producto de trabajo. En cuanto a la estructura: los márgenes y sangría de la plantillas son diferentes, las filas y columnas de las tablas no presentan uniformidad.

Sobre la base del planteamiento anterior constituye un **Problema a resolver:** ¿Cómo estandarizar los productos de trabajos que se generan en la disciplina modelado de negocio de la metodología variación AUP-UCI?

Se identifica como **Objeto de estudio:** Proceso de desarrollo de software, enmarcado en el **Campo de acción:** Estandarizar los productos de trabajos de la disciplina Modelado de negocio que define la metodología variación AUP-UCI.

Para darle respuesta al problema planteado se identifica el siguiente **Objetivo General:** Desarrollar una herramienta que permita generar productos de trabajos de la metodología variación AUP-UCI, para estandarizar los productos de trabajos que se generan en la disciplina de modelado de negocio.

Objetivos específicos:

1. Construir el marco teórico conceptual de la investigación a partir del proceso de desarrollo de software, para estandarizar los productos de trabajos de la disciplina Modelado de negocio que define la metodología variación AUP-UCI.

2. Analizar y diseñar la herramienta para generar productos de trabajos de la metodología variación AUP-UCI.
3. Implementar la herramienta para generar productos de trabajos de la metodología variación AUP-UCI.
4. Validar la solución propuesta mediante la automatización de pruebas unitarias de códigos PHP.
5. Validar la variable dependiente de la investigación mediante un Cuasiexperimento.

Se propone como **Idea a defender:** Si se desarrolla una herramienta para generar productos de trabajos de la variante de la metodología AUP-UCI, se logrará estandarizar los productos de trabajos que se generan en la disciplina de modelado de negocio de la metodología AUP-UCI.

Se pretende obtener como **Posibles resultados:** Herramienta para la generación de productos de trabajos de la metodología variación AUP-UCI.

Métodos teóricos

Método histórico-lógico: Se aplicó para recolectar la información sobre los temas relacionados con la aplicación en un ámbito nacional e internacional, de las tecnologías que presentan un diseño de una metodología de desarrollo de software y que pueda generar productos de trabajos de la misma.

Método analítico sintético: Se aplicó para analizar la información extraída y sintetizar la información en cuanto a los conceptos asociados al dominio del problema, las disciplinas, escenarios y productos de trabajos de la metodología AUP. En sentido general englobado en el campo de acción de la investigación.

Métodos empíricos

Entrevista: Se aplicó para obtener información de los proyectos de la universidad que utilizan la metodología AUP-UCI, como se representa en el Anexo1 Escenarios que aplican los proyectos que utilizan la metodología AUP-UCI. Se realizó con el grupo de calidad de la universidad, quienes proporcionaron la información de los proyectos por cada uno de los centros y los escenarios en que se encontraban según la metodología. Además se realizaron entrevistas a los especialistas de los proyectos de la universidad Centro de Informática Médica, Nova Servidores y Arquitectura de Referencia de PHP Bosón, en las cuales se identificaron las principales deficiencias encontradas durante el proceso de creación, actualización y revisión de los productos de trabajos propuestos en la metodología variación AUP-UCI.

La presente investigación se encuentra estructurada de la siguiente manera:

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se hace referencia a los principales conceptos asociados al dominio del problema, se realiza un análisis de las herramientas para la Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) que intervienen en casi todos los aspectos del ciclo de vida de desarrollo del

software. Por último, se identifica y caracterizan los lenguajes de programación, las herramientas y metodologías que permitirán desarrollar la solución propuesta.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA HERRAMIENTA PARA GENERAR PRODUCTOS DE TRABAJOS

En este capítulo se realiza el análisis y diseño de la aplicación, se obtienen los requisitos funcionales y no funcionales y se construyen los diagramas de clases del diseño que permiten modelar el código fuente de la herramienta. Se define la arquitectura y los patrones arquitectónicos utilizados.

CAPÍTULO 3: IMPLEMENTACIÓN, PRUEBA Y VALIDACIÓN DE LA HERRAMIENTA PARA GENERAR PRODUCTOS DE TRABAJOS

En este capítulo se elabora el modelo de componentes de la aplicación representando los componentes y las relaciones entre ellos, se valida la solución propuesta mediante la automatización de pruebas unitarias de códigos PHP y se valida la variable dependiente de la investigación mediante un Cuasiexperimento.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se presentan los conceptos fundamentales asociados al dominio del problema. Se describe las disciplinas, productos de trabajos y escenarios de la metodología AUP-UCI. Se realiza un análisis de las herramientas para la Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) que intervienen en casi todos los aspectos del ciclo de vida de desarrollo del software. Por último, se caracterizan las tecnologías y los lenguajes de programación a utilizar durante el desarrollo de la solución.

1.1. Conceptos asociados al dominio del problema

1.1.1. Proceso de desarrollo de software

Un proceso de desarrollo de software es un conjunto de personas, estructuras de organización, reglas, políticas, actividades y sus procedimientos, componentes de software, metodologías, y herramientas utilizadas o creadas específicamente para definir, desarrollar, ofrecer un servicio, innovar y extender un producto de software (Mara, 2017).

1.1.2. Metodología AUP-UCI

Resulta necesario establecer un enfoque sistemático y disciplinario para desarrollar un software. El uso de una metodología permite el dominio del proceso descrito. Una metodología es un enfoque, una manera de interpretar la realidad o la disciplina en cuestión, que en caso particular correspondía a la ingeniería del software. Se elabora a partir del marco definido por uno o varios ciclos de vida (Mara, 2017). Describe cómo se organiza un proyecto. Establece el orden en el que la mayoría de las actividades tienen que realizarse y los enlaces entre ellas. Indica cómo tienen que realizarse algunas tareas proporcionando las herramientas concretas e intelectuales.

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. La Universidad de las Ciencias Informáticas (UCI) ha definido una variación de la metodología AUP en unión con el modelo CMMI-DEV v 1.3, para que pueda emplearse en los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI).

1.1.3. Disciplinas de la Metodología Variación AUP-UCI

Modelado de negocio: Destinada a comprender los procesos de negocio de una organización. Se define cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes: Casos de Uso del Negocio (CUN), Descripción de Proceso de Negocio (DPN) y Modelo Conceptual (MC). A partir de las

variantes anteriores se condicionan cuatro escenarios para modelar el sistema en la disciplina Requisitos (Tamara, 2015).

La investigación se centra en la disciplina de Modelado de negocio que describe la metodología variación AUP-UCI.

Requisitos: El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos: Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP), agrupados en cuatro escenarios condicionados por el Modelado de negocio (Tamara, 2015).

Análisis y diseño: Los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos que el de análisis (Tamara, 2015).

Implementación: En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema (Tamara, 2015).

Pruebas internas: Se verifica el resultado de implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberada. Se deben desarrollar artefactos de prueba como: diseños de de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas (Tamara, 2015).

Pruebas de liberación: Diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación (Tamara, 2015).

Pruebas de Aceptación: Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido (Tamara, 2015).

Descripción de los escenarios

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (CUN, DPN o MC) y existen tres formas de encapsular los requisitos (CUS, HU, DRP), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

Escenario No 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema.

Escenario No 2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio.

Escenario No 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad.

Escenario No 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos (Tamara, 2015).

1.1.4. Productos de trabajos

En el sistema de Mejoras de Procesos de Software de la Universidad (url: mejoras.prod.uci.cu), en el Área de proceso: Administración de Requisitos (REQM[2]) se listan los productos de trabajos que se realizan durante la administración de requisitos. En la investigación se definieron como productos de trabajos a generar por la herramienta: los productos principales de la Disciplina de Modelado de Negocio: Reglas del negocio (RN), Modelo de negocio con casos de uso (CUN) y Descripción de procesos de negocio (DPN).

1. Criterios para validar requisitos del cliente: Conjunto de criterios que permiten decidir si se aceptan o se rechaza un requisito del cliente.
2. Criterios para validar requisitos del producto: Se establecen o analizan un conjunto de criterios que permiten decidir si un requisito es aprobado o rechazado.
3. Descripción de requisitos por proceso: Producto de trabajo donde se realiza la descripción de los requisitos del producto.
4. Especificación de casos de uso: Se describe el sistema en función de los casos de uso.
5. Especificación de requisitos de software: En este producto de trabajo quedan documentadas las características y capacidades con las que el software debe cumplir. Es un producto de trabajo fundamental en el desarrollo de software.

[2] Área que permite gestionar los requisitos de los productos y los componentes de producto del proyecto y asegurar la alineación entre esos requisitos, los planes y los productos de trabajo del proyecto.

6. Evaluación de casos de uso: Provee una guía para la evaluación de los casos de uso lo que permite obtener una valoración de la complejidad y la prioridad de los mismos en función del desarrollo del producto.

7. Glosario de términos: Permite obtener un entendimiento de términos o acrónimos que se utilizan durante la documentación del proceso de desarrollo.

8. Modelo conceptual: Este documento modela los objetos de dominio o conceptos de los procesos de negocio o áreas funcionales de la organización.

9. Modelo de negocio con casos de uso: Este artefacto permite describir en función de casos de uso del negocio las actividades que desarrolla la entidad y que son candidatas a ser informatizadas.

10. Lista de verificación de revisiones de inconsistencia: Se utiliza como Lista de chequeo para registrar las inconsistencias encontradas en las revisiones planificadas en la Herramienta de Gestión de Proyecto.

11. Registro de proveedores de requisitos: Este producto de trabajo permite registrar los posibles proveedores de requisitos y evaluar si son aptos o no para ejecutar esta función.

12. Reglas del negocio: Producto de trabajo para registrar las reglas de negocio que rigen los procesos establecidos en la organización.

13. Reporte de trazabilidad: Este documento contiene un análisis del impacto de los cambios a un artefacto determinado o grupo de artefactos. Es válido utilizar el Excel que se genera en la herramienta de Trazabilidad.

14. Requisitos rechazados: Se documentan los requisitos rechazados y los motivos por los cuales se han rechazado.

15. Salida del sistema: En este producto de trabajo se modelan las salidas del sistema, los reportes que emite el sistema.

16. Historias de usuario: Producto de trabajo donde se realiza la descripción de los requisitos del producto identificados en los proyectos que encapsulan sus requisitos en Historias de usuario.

17. Matriz EntidadesBD_Conceptos: Matriz de trazabilidad de Entidades de la bases de datos contra el Modelo conceptual.

18. Matriz EntidadesBD_DiagClaseDiseño: Matriz de trazabilidad de Entidades de la bases de datos con el Diagrama de Clases del Diseño.

19. Matriz Requisito_CUN: Matriz de trazabilidad de Requisitos con Caso de Uso del negocio.

20. Matriz Requisito_Proceso de negocio: Matriz de trazabilidad de Requisitos con Procesos de Uso del negocio.

21. Matriz Requisito_Artefactos: Matriz de trazabilidad de Requisitos con Artefactos.

22. Matriz Requisito_Conceptos: Matriz de trazabilidad de Requisitos con Modelo conceptual.

21. Matriz Requisito_CUS: Matriz de trazabilidad de Requisitos con Caso de Uso del Ssitema.

23. Matriz Requisito_DCP: Matriz de trazabilidad de Requisitos con Diseño de Casos de Pruebas.

24. Matriz Requisito_DiagClaseDiseño: Matriz de trazabilidad de Requisitos con Diagrama de Clases del Diseño.

25. Descripción de procesos de negocio: Producto de trabajo donde se describen los elementos del negocio identificados con el cliente.

26. Diseño de casos de prueba: Producto de trabajo que permite realizar las comprobaciones de los requisitos funcionales de software a partir de escenarios de pruebas.

27. Documentación del negocio entregada por el cliente: La documentación del negocio entregada por el cliente constituye un conjunto de documentos necesarios para dar inicio al entendimiento de los procesos de la organización por parte del equipo de proyecto.

28. Mapa de procesos: Producto de trabajo que permite tener una visión general de los procesos de negocio identificados con el cliente.

29. Matriz Requisito_Paquete funcional: Trazabilidad de requisitos con el código fuente de la aplicación.

30. Herramientas instaladas y configuradas: Conjunto de herramientas necesarias para llevar a cabo la Administración de requisitos.

31. Acta de inicio: Formaliza la fecha de inicio del proyecto y las personas naturales que trabajarán como representantes de las entidades involucradas.

32. Ficha técnica del proyecto: Contiene los elementos necesarios y suficientes para realizar una ficha técnica de productos o servicios al cliente.

33. Oferta: Contiene los elementos necesarios y suficientes para realizar una oferta de productos o servicios al cliente (Excriba).

1.1.5. Concepto de estandarizar

Estandarizar es ajustar a alguien o algo a un estándar (WordReference.com, 2017). Los especialistas de los proyectos deben ajustar con el uso de la herramienta al formato estándar de los productos de trabajo que se generan. Se generarán plantillas automatizadas de los productos de trabajos con estándares previamente definidos para asegurar que se cumplan los mismos.

1.2. Tecnologías existentes de diseño, elaboración y generación de productos de trabajos

En la actualidad existe una amplia gama de tecnologías y aplicaciones que son empleadas por los usuarios para generar documentación relacionada con temas de interés para su trabajo, estudio u otra finalidad. Las herramientas para la Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) intervienen en casi todos los aspectos del ciclo de vida de desarrollo del software, específicamente en tareas tales como el diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente, compilación automática, documentación o detección de errores, entre otras. Estas herramientas están destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de capital (Visual Paradigm, 2014). A continuación, se

describen las principales herramientas CASE que generan documentación referente a las metodologías de desarrollo de software utilizadas actualmente.

Herramienta: EasyCASE

Descripción: Producto para la generación de esquemas de base de datos e ingeniería inversa. Permite automatizar las fases de análisis y diseño dentro del desarrollo de una aplicación, desde el procesamiento de transacciones a la aplicación de bases de datos de cliente/servidor, así como sistemas de tiempo real. Permite capturar los detalles del diseño de un sistema y comunicar las ideas gráficamente. Para un diseño legítimo y modelado de datos, procesos y eventos, permite crear y mantener diagramas de flujo de datos, diagramas de entidad-relación y mapas de estructura. Posee herramientas de corrección avanzadas que permiten revisiones generales. Permite re-usar diagramas o partes de diagramas para economizar el diseño de un proyecto. Soporta una gama amplia de metodologías estructuradas, permitiendo escoger los métodos más apropiados para realizar las tareas. Determina los tipos de esquemas según la metodología del proyecto seleccionada y notifica de errores a medida que el modelo vaya construyéndose. Posee desde el editor de diagramas flexible y un diccionario de los datos, así como una extensa cantidad de reportes y análisis (José Manuel González Peña, 2017).

Herramienta: Oracle Designer

Descripción: Es un juego de herramientas para guardar las definiciones que necesita el usuario y automatizar la construcción rápida de aplicaciones cliente/servidor flexibles y gráficas. En el lado del Servidor, Oracle Designer soporta la definición, generación y captura de diseño de los siguientes tipos de bases de datos, por conexión nativa de Oracle y por conectividad ODBC: Oracle7 y más, Personal Oracle Lite, Rdb, ANSI 92, DB, MVS y Microsoft SQL Server. Oracle Designer soporta las siguientes metodologías: Desarrollo Rápido de Aplicaciones (RAD), Ingeniería de la Información (IE), Modelado Asistido de Procesos y Captura de Diseño Asistido (Designer, 2017).

Herramienta: System Architect

Descripción: Posee un repositorio único que integra todas las herramientas, y metodologías usadas. En la elaboración de los diagramas conecta directamente al diccionario de datos, los elementos asociados, comentarios, reglas de validaciones y normalización. Posee control automático de diagramas y datos, normalizaciones y balanceamiento entre diagramas "Padre e Hijo", además de balanceamiento horizontal, asegurando la compatibilidad entre el Modelo de Datos y el Modelo Funcional. Traduce modelos de entidades, a partir de la enciclopedia, en esquemas para Sybase, DB2, Oracle, Ingress, SQL Server, RDB, XDB, Progress, Paradox, SQL Base, AS400, Interbase, OS/2, DBMS, Dbase 111, Informix, entre otros. Genera también Windows DDL y definiciones de datos para lenguaje C/C++. Posibilita a través de ODBC, la creación de bases de datos a partir del modelo de entidades, para los diversos manejadores de bases de datos. Posee un módulo específico para Ingeniería Inversa desde las Bases de Datos SQL, incluyendo Sybase, DB2, Infonmix, Oracle y SQL Server (DLL). Posee múltiples metodologías para

diseño y análisis, incluyendo: Análisis Estructurado en los modelos De Marco/Yourdon y Gane/Sarson, análisis de tiempo real en el modelo Ward & Mellor; análisis esencial de sistemas; análisis orientado a objetos en los modelos UML, Booch, Coad/Yourdon, Rumbaugh, Shaler/Mellor; Diagrama de entidad - relación en los modelos Peter Chen, James Martin, Bachman o Booch, Gráfico de Estructuras, Diagramas de Descomposición y Planeamiento Estratégico de informaciones (Architect).

Herramienta: JDeveloper

Descripción: Entorno integrado desarrollado por Oracle trabaja con la ingeniería inversa, es decir primero se crea el código y después el diagrama. Es un software propietario pero gratuito desde 2005. Sus características principales: Netamente desarrollado para Java., posee diagrama de clases (UML), funciona en los siguientes sistemas operativos: Windows, Linux, Mac OSX (Alonso, 2017).

Herramienta: Rational Rose

Descripción: Instrumento operativo conjunto que utiliza el Lenguaje Unificado (UML) como medio para facilitar la captura de dominio de la semántica, la arquitectura y el diseño. Sus características principales:

La ingeniería de código (directa e inversa) es posible para ANSI C++, Visual C++, Visual Basic 6, Java, J2EE/EJB, CORBA, Ada 83, Ada 95, Base de datos: DB2, Oracle, SQL 92, SQL Server, Sybase, Aplicaciones WEB. Provee plantillas de código que pueden aumentar significativamente la cantidad de código fuente. Adicionalmente, se pueden aplicar los patrones de diseño. Rational Rose ha provisto 20 de los patrones de diseño GOF para Java. Admite la integración con otras herramientas de desarrollo (IDEs). Soporte para Ciclo de Vida de un Proyecto en Rational Rose: Modelado de Negocio: Usando el modelo de casos de uso de negocio, Administración de Requisitos: Junto con RequisitePro. Análisis y Diseño: Diagramas UML de clases y de interacción. El asistente de frameworks provee una gran cantidad de plantillas para estructurar el modelo. Implementación: Soporta la mayoría de los lenguajes excepto .NET Control de Versiones: Integrado con la aplicación de control de versiones compatible con SCC (Nobrega, 2005).

Herramienta: MagicDraw

Descripción: Herramienta de modelado con completas características UML. Es desarrollada por No Magic, Inc. Implementada totalmente en JAVA. Diseñada para los analistas del negocio y del software, los programadores, los ingenieros de software, y los escritores de la documentación, esta herramienta facilita el análisis y el diseño de los sistemas y de las bases de datos orientados objeto. Posee un Generador automático de informes (Alonso, 2017).

Herramienta: Visual Paradigm

Descripción: Herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar

todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (López). Características principales:

Soporte de UML versión 2.1. Diagramas de Procesos de Negocio, Modelado colaborativo con CVS y Subversión (control de versiones). Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI. Ingeniería inversa - Código a modelo, código a diagrama. Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL. Editor de Detalles de Casos de Uso – Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso. Diagramas de flujo de datos. Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos. Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación. Generador de informes. Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML (Alonso, 2017).

Herramienta: Enterprise Architect

Descripción: Soporta ocho de los nueve diagramas estándares del UML: diagrama de casos de uso, de clases, de secuencia, de colaboración, de actividad, de estados, de implementación (componentes), de despliegue y varios perfiles del UML. Enterprise Architect tiene un mecanismo de perfil UML genérico para cargar y trabajar con diferentes perfiles UML (Architect, 2012). Los perfiles disponibles son:

Modelado de Procesos de Negocio: Soporta las extensiones de modelado de procesos de negocio de Eriksson-Penker, Modelado de Datos, Modelado de la Interfaz de Usuario. Modelado Web, Esquema XSD. Permite ingeniería de código (directa e inversa) para ANSI C++, Visual Basic 6, Java, C#, VB.NET, Delphi y Base de datos: Ingeniería directa desde el modelo de datos al script DDL. La ingeniería reversa usa la fuente de datos ODBC. Adicionalmente, se pueden aplicar los patrones de diseño, el usuario tiene que crear los patrones (Alonso, 2017).

Soporte del ciclo de vida del proyecto en Enterprise Architect, para las disciplinas:

Modelado de Negocio: perfiles de UML para el modelado de procesos de negocio.

Administración de Requisitos: Requisitos funcionales y no funcionales; matriz de trazabilidad de requisitos.

Análisis y Diseño: Diagramas UML de clases y de interacción.

Implementación: Es adecuada para proyectos C++, VB, C# y VB.NET

Administración de proyecto: Administración de riesgos, asignación de recursos y estimación del proyecto.

Herramienta: CASE Studio

Descripción: Herramienta con potente utilidad de modelado para varias bases de datos. CASE Studio es una herramienta profesional con la que pueden diseñarse bases de datos, incluye facilidades para la

creación de diagramas de relación, modelado de datos y gestión de estructuras. Tiene soporte para trabajar con una amplia variedad de formatos de base de datos (Oracle, SQL, MySQL, PostgreSQL, Access) y permite además generar xcripts SQL, aplicar procesos de ingeniería inversa, usar plantillas de diseño personalizables y crear informes en HTML y RTF (Alonso, 2017).

Se analizaron las herramientas anteriormente citadas y se evidenció la existencia de herramientas CASE que generan gran cantidad de documentación referente a las metodologías de desarrollo de software. Pero ninguna de las analizadas genera documentación específicamente con el modelo estándar de los productos de trabajos de la metodología AUP-UCI, que es la empleada actualmente en los proyectos de la Universidad de Ciencias Informáticas.

1.3. Tecnologías, lenguajes y herramientas utilizadas

1.3.1. Lenguajes de programación

PHP 5.5: Lenguaje diseñado especialmente para desarrollo de aplicaciones web, puede ser incrustado dentro del código HTML y desplegado en casi todos los sistemas operativos, plataformas y en la mayoría de los servidores web. Como principales ventajas se puede mencionar la capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL. Permite el manejo de excepciones, cuenta con una amplia documentación y soporte de la comunidad (Revista Científica Ingeniería y Desarrollo, 2014).

JavaScript: Lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (librosWeb, 2006-2017).

1.3.2. Sistema de base de datos

El sistema de base de datos a utilizar es MySQL, en su versión 5.6.12. Está escrito en C y C++, y se destaca por su adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación como PHP, Perl y Java, y su integración en distintos sistemas operativos. MySQL sigue políticas de código abierto, lo cual permite que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente, lo que ha favorecido su desarrollo y continuas actualizaciones(MySQL, 2015).

1.3.3. Marcos de trabajo

EXT.JS 6.2: Ext JS es un marco de JavaScript que proporciona una interfaz de usuario para crear aplicaciones web. Ext JS se utiliza básicamente para la creación de aplicaciones de escritorio. Es

compatible con todos los navegadores como Internet Explorer 6 +, FF, Chrome, Safari 6, ópera 12+ etc. Mientras que otro producto de sencha, touch sencha se utiliza para aplicaciones móviles. Ext JS se basa en la arquitectura MVC / MVVM. La última versión de Ext JS 6 es una única plataforma que puede ser utilizado tanto para escritorio y aplicaciones móviles sin tener código diferente para diferentes plataformas (ExtJS, 2016).

Symfony2: Es un marco de trabajo para desarrollar aplicaciones PHP. Symfony 2 ha sido ideado para aprovechar al máximo las nuevas características de PHP y es uno de los marcos de trabajo con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto (Equiluz, 2014).

El código de todos los componentes y librerías que incluye, se publican bajo la licencia MIT de software libre. La documentación del proyecto también es libre e incluye varios libros y decenas de tutoriales específicos. Cuenta con una comunidad que brinda el soporte necesario para desarrollar las aplicaciones. Incluye varias herramientas gráficas y de consola para depurar fácilmente los errores que se produzcan en las aplicaciones. Para evitar el uso de contraseñas en archivos de configuración, permite establecer los parámetros de configuración de las aplicaciones a través de variables de entorno del propio servidor. Utiliza la herramienta Composer para simplificar la instalación y gestión de las dependencias de las aplicaciones PHP. Dispone de un plan de lanzamientos predecible, con versiones estables mantenidas durante tres años.

Bosón: Es un marco de trabajo basado en una arquitectura de referencia haciendo uso de tecnologías libres, garantiza que todas las soluciones que se desarrollen sobre PHP, cumplan con la versatilidad necesaria para construir sistemas de distintos dominios y proporcione la capacidad de integración con un mínimo esfuerzo. Con el desarrollo de este proyecto se ahorra tiempo y recursos materiales en la construcción de soluciones informáticas, ya que provee un conjunto de componentes tecnológicos que son indispensables para la mayoría de los proyectos. Utiliza como marco base Symfony2 que garantiza la presencia de una comunidad mundial que brinda soporte y actualizaciones a diario. Permite a los equipos de desarrollo centrarse en la implementación solo de la lógica de su negocio abstrayéndose de elementos como la gestión de la caché, el manejo de estructuras de datos, el registro de trazas, la seguridad de los sistemas, la gestión de excepciones, la definición de componentes visuales, el manejo de conceptos globales, la validación de datos y la gestión de componentes. El marco de trabajo Bosón materializa los requisitos comunes de las arquitecturas base para sistemas de gestión web.

1.3.4. Mapeador de objeto relacional

Doctrine 0.9: Es un mapeador de objeto relacional Object Relation Mapper (ORM) para PHP 5.2.3+ que se encuentra en la parte superior de una capa poderosa de abstracción de base de datos (DBAL). Una de sus principales características es la opción de escribir las consultas de base de datos en un dialecto

orientado a objetos de propiedad SQL llamado Doctrine QueryLanguage (DQL), lo que proporciona a los desarrolladores una alternativa a SQL, que mantiene la flexibilidad sin necesidad de duplicar el código innecesario (Doctrine, 2016).

1.4. Métricas de validación del diseño

Medir la calidad del software es una tarea propensa al debate porque no existen métodos absolutos para lograrlo (Pressman, 2010). Con el objetivo de medir la calidad de los atributos de software se proponen un conjunto de listas de chequeo, como por ejemplo: facilidad de mantenimiento, facilidad de prueba y reusabilidad (McCall, 1977). Con respecto al diseño a nivel de componentes algunos autores han propuesto con similar objetivo un conjunto de métricas centrándose en las características internas de los componentes de software. El diseño de la solución propuesta se valida utilizando las métricas Tamaño operacional de clases (TOC por sus siglas) y Relaciones entre clases (RC por sus siglas). Estas son métricas de validación del diseño a nivel de componentes y evalúan gran parte de los atributos de software que se han mencionado, siendo la razón por la cual han sido seleccionadas (Equiluz, 2013). Los atributos medidos son:

Responsabilidad: Responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
Complejidad de implementación: Grado de complejidad que posee la implementación de un diseño de clases específico.

Reutilización: Grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

Acoplamiento: Grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.

Complejidad del mantenimiento: Grado de esfuerzo necesario para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

Cantidad de pruebas: Número o grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, entre otros) diseñado.

1.5. Arquitectura en Symfony 2. Ciclo de vida de la petición, controlador, respuesta

Cada petición manejada por un proyecto Symfony2 pasa por el mismo ciclo de vida básico. La plataforma se encarga de todas las tareas repetitivas iníciales y después, pasa la ejecución al controlador, que contiene el código personalizado de la aplicación. Cada petición es manejada por un único archivo controlador frontal. Un controlador frontal es un archivo PHP, normalmente pequeño, que se encuentra en el directorio web raíz del proyecto y a través del cual se atienden todas las peticiones del usuario. Una

aplicación típica tiene un controlador frontal de producción (por ejemplo, app.php) y un controlador frontal de desarrollo (por ejemplo, app_dev.php).

1. El sistema de enrutamiento (clase Routing) lee la información de la petición (por ejemplo, la URI), encuentra una ruta que coincida con esa información, y lee el parámetro_controller de la ruta.
2. Se ejecuta el controlador asignado a la ruta y este controlador crea y devuelve un objeto Response.
3. Las cabeceras HTTP y el contenido del objeto Response se envían de vuelta al cliente.

Conclusiones del capítulo

En este capítulo se trataron los principales elementos teóricos que sustentan la propuesta de solución del problema planteado.

Se concluye que el análisis de los principales conceptos asociados al problema y la recopilación de información permitió alcanzar mayor entendimiento de la investigación que se realiza. Existen herramientas CASE que generan documentación referente a las metodologías de desarrollo de software, pero no específicamente para los productos de trabajo de la metodología AUP-UCI, que es la que se emplea actualmente en la Universidad de Ciencias Informáticas. Se seleccionaron las herramientas, y tecnologías basadas en software libre, lo cual posibilita sustentarse en una base tecnológica enfocada en los proyectos productivos de la universidad que utilizan la metodología AUP-UCI.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA HERRAMIENTA PARA GENERAR PRODUCTOS DE TRABAJOS

Introducción

En este capítulo se presentan el análisis y diseño de la solución propuesta. Se realiza el modelo conceptual representando los objetos y conceptos significativos del dominio del problema. Son identificados y validados los requisitos funcionales y no funcionales, utilizando técnicas de obtención y validación de requisitos. Por último, se construyen los diagramas de clases del diseño que permiten modelar el código fuente de la solución propuesta.

2.1. Modelo Conceptual

Un modelo conceptual tiene como objetivo identificar y explicar los conceptos significativos en un dominio de problema, identificando los atributos y las asociaciones existentes entre ellos. Puede ser visto, también como una representación de las cosas, entidades, ideas, conceptos u objetos del “mundo real” o dominio de interés. También es considerado como un diccionario visual de abstracciones, conceptos, vocabulario e información del dominio de problema.

El modelo conceptual de la solución propuesta está representado por el concepto de Metodología de software, específicamente Variación AUP-UCI, como se muestra en la Figura 1 Modelo conceptual de la disciplina de modelado de negocio. La metodología tiene definida un conjunto de disciplinas, en la investigación se modela la disciplina de Modelado de negocio, que permite generar un conjunto de productos de trabajos. Los cuales son: Reglas de negocio, Descripción de procesos de negocio y Caso de Uso del Negocio, que tienen como atributos los elementos estructurales que componen cada uno de los productos de trabajos.

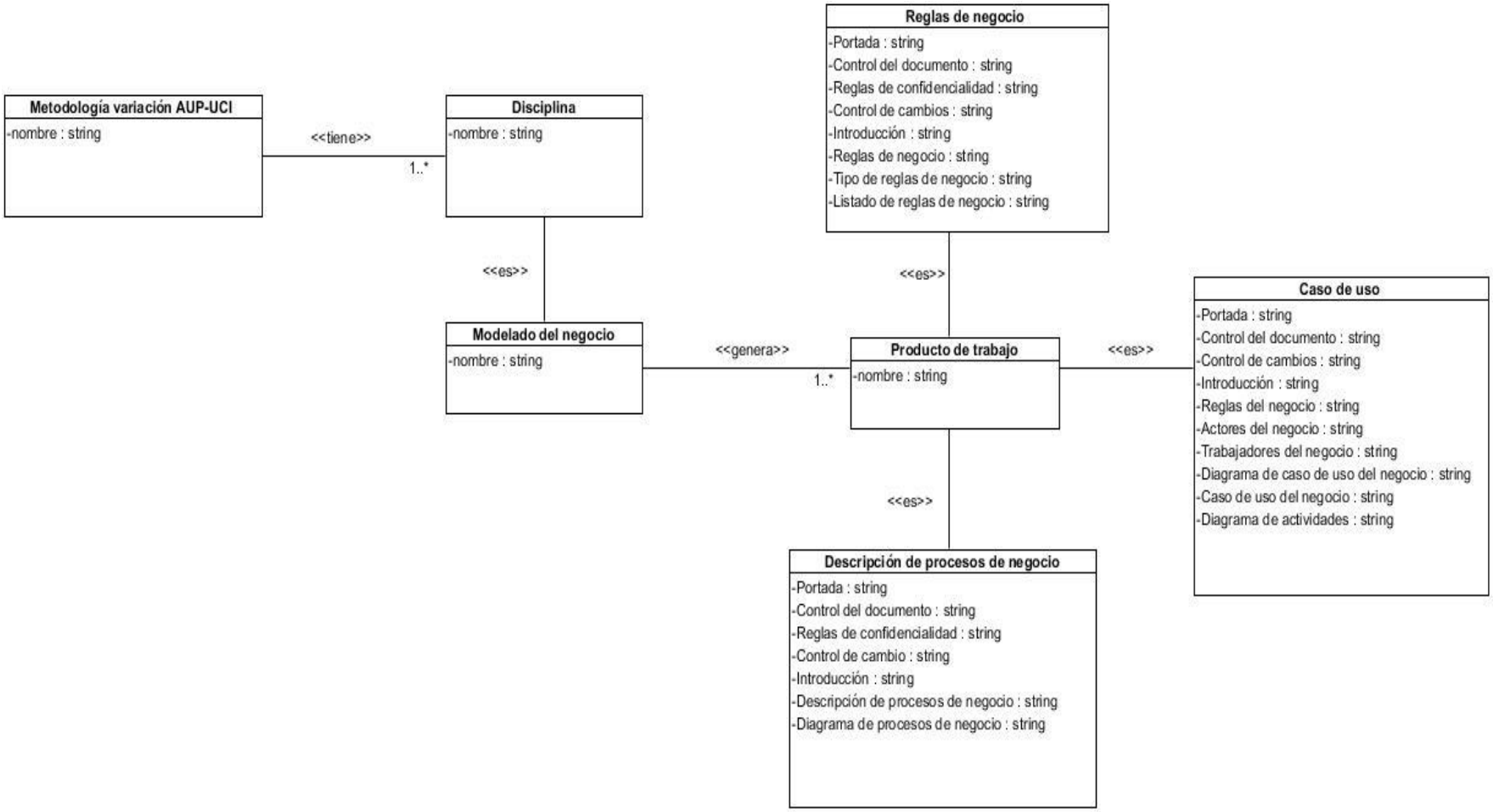


Figura 1 Modelo conceptual de la disciplina de modelado de negocio.

2.2. Requisitos del sistema

La ingeniería de requisitos tiene como objetivo garantizar la correcta descripción de los requisitos para evitar futuros errores y lograr la reducción de tiempo en la construcción de un software. En ella se identifican dos aspectos muy importantes: el primero es el propósito del sistema que se va a desarrollar y el segundo, el contexto en el que será usado. La ingeniería de requisitos se define como el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo que satisfaga las necesidades del usuario (Pressman, 2010).

2.2.1. Pasos para realizar la Ingeniería de Requisitos.

1. **Captura de requisitos**: el equipo de desarrollo define los requisitos, teniendo en cuenta la información brindada por el cliente.
2. **Descripción de requisitos**: a partir de dicha información, se elaboran los documentos de especificación de requisitos.
3. **Valoración y validación de los requisitos**: se buscan inconsistencias, errores o faltas en los requisitos detectados, se analiza nuevamente la información para verificar que no se omita ningún requerimiento del cliente.

2.2.2. Técnicas de obtención de Requisitos

Existen un número de técnicas para llevar a cabo el proceso de captura de requisitos, pero depende del equipo de desarrollo determinar cuál o cuáles son las más indicadas a usar en su caso particular. Las principales son:

Entrevistas y cuestionarios: La entrevista es un método muy efectivo que permite conocer los problemas de los clientes y encontrar requisitos generales (Ganesh, 2008). Permite al equipo de desarrollo interpretar las necesidades del cliente. Pueden ser lo mismo provechosos o no ya que dependen de las habilidades del entrevistador y los entrevistados para obtener información con la mayor calidad posible. Se aplicó en las entrevistas realizadas a la analista del proyecto Bosón y especialistas de calidad de la universidad.

Mapas conceptuales: Es otra técnica bastante común, usada para la representación gráfica de las ideas y sus relaciones. Se aplicó en la representación de la estructura de los productos de trabajos, que se generan en la disciplina de Modelado del negocio.

Tormenta de ideas: Consiste en la acumulación de ideas sin prejuicios y valoraciones que puedan descartarlas y aunque no evidencia los detalles concretos que se pueden necesitar del sistema, es muy común en los comienzos del proceso de ingeniería de requisitos. Se aplicó en las reuniones con la tutora de la tesis de investigación, sobre la definición de requisitos según la característica de la metodología.

2.2.3. Técnicas de Validación de Requisitos

Las técnicas de validación de requisitos descritas a continuación son las que se encuentran en el libro “Ingeniería de Software” v.7 de Ian Somerville: (Somerville, 2005)

Revisiones: Los requisitos son analizados sistemáticamente por un equipo de revisores, en busca de anomalías y/u omisiones. Se aplicó cuando se generan los productos de trabajos Descripción de requisitos por procesos, donde son revisados por la analista del proyecto, corrigiendo errores conceptuales y ortográficos.

Prototipos: Se muestra un modelo ejecutable del sistema a los usuarios finales y los clientes, para que puedan ver si dicho modelo cumple con sus necesidades reales. Se aplicó cuando se realizan prototipos de interfaces de usuarios en ExtJS.

Generación de casos de prueba: La elaboración de casos de prueba puede revelar los problemas con que puede contar un requisito y es parte fundamental de la programación externa. Se aplicó cuando se identifican valores de entrada y salida para cada uno de los requisitos funcionales.

2.3. Requisitos Funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer.

En el proceso de captura de los requisitos se tuvo en cuenta cada uno de los productos de trabajos que se deben generar por cada una de las disciplinas. Luego, de obtener los requisitos funcionales se agruparon siguiendo los criterios:

Agrupación funcional: Se agrupan los requisitos que tienen una relación funcional directa según el tipo de datos o según los procesos de negocio que van a cubrir. Se aplica agrupando por relación funcional directa ejemplo control del documento para adicionar, eliminar, listar y modificar. Como resultado se obtiene gestionar control del documento.

En la disciplina de modelado de negocio se deben generar tres productos de trabajos principales: Modelo de Negocio con Caso de Uso, Descripción de proceso de negocio y Reglas del Negocio quedando definido los siguientes requisitos funcionales:

Tabla 1.Requisitos funcionales del producto de trabajo Descripción de proceso de negocio.

Producto de trabajo: **Descripción de proceso de negocio**		
Estructura del documento	**Técnica de Agrupación funcional**	**Requisitos funcionales**

Portada	RF1. Gestionar elementos de la portada.	RF1. 1 Adicionar elementos de la portada RF1. 2 Modificar elementos de la portada RF 1.3 Eliminar elementos de la portada RF1.4 Listar elementos de la portada
Control del documento	RF2. Gestionar control del documento.	RF2. 1 Adicionar elementos del control del documento. RF2. 2 Modificar elementos del control del documento. RF 2.3 Eliminar elementos del control del documento. RF2.4 Listar elementos del control del documento.
Control de cambios	RF3. Gestionar control de cambios.	RF3. 1 Adicionar elementos del control de cambios. RF3. 2 Modificar elementos del control de cambios. RF 3.3 Eliminar elementos del control de cambios. RF3.4 Listar elementos del control de cambios.
Introducción	RF4. Gestionar elementos de la introducción.	RF4. 1 Adicionar elementos de la introducción. RF4. 2 Modificar elementos de la introducción. RF 4.3 Eliminar elementos de la introducción. RF4.4 Listar elementos de la introducción.
Descripción del proceso del negocio	RF5. Gestionar elementos del proceso de negocio.	RF5. 1 Adicionar elementos del proceso de negocio. RF5. 2 Modificar elementos del proceso de negocio. RF 5.3 Eliminar elementos del proceso de negocio. RF5.4 Listar elementos del proceso de negocio.
Diagrama del proceso del negocio	RF6. Gestionar diagrama del proceso de negocio.	RF6. 1 Cargar imagen del diagrama del proceso negocio. RF6. 2 Modificar imagen del diagrama del proceso de negocio. RF 6.3 Eliminar imagen del diagrama del proceso de negocio.
RF7. Exportar a formato pdf el producto de trabajo Descripción de proceso de negocio. RF8. Exportar a formato doc el producto de trabajo Descripción de proceso de negocio.		

.

Tabla 2 Requisitos funcionales del producto de trabajo Modelado de negocio.

Producto de trabajo: **Modelo de Negocio con Caso de Uso**

Estructura del documento	**Técnica de Agrupación funcional**	**Requisitos funcionales**
Portada	RF9. Gestionar elementos de la portada.	RF 9. 1 Adicionar elementos de la portada RF 9. 2 Modificar elementos de la portada RF 9.3 Eliminar elementos de la portada RF 9.4 Listar elementos de la portada
Control del documento	RF 10. Gestionar control del documento.	RF 10. 1 Adicionar elementos del control del documento. RF 10. 2 Modificar elementos del control del documento. RF 10.3 Eliminar elementos del control del documento. RF 10.4 Listar elementos del control del documento.
Control de cambios	RF11. Gestionar control de cambios.	RF11. 1 Adicionar elementos del control de cambios. RF11. 2 Modificar elementos del control de cambios. RF11.3 Eliminar elementos del control de cambios. RF11.4 Listar elementos del control de cambios.
Introducción	RF12. Gestionar elementos de la introducción.	RF12.1 Adicionar elementos de la introducción. RF12.2 Modificar elementos de la introducción. RF12.3 Eliminar elementos de la introducción. RF12.4 Listar elementos de la introducción.
Reglas del negocio	RF13. Gestionar elementos de la regla de negocio.	RF13.1 Adicionar elementos de la regla de negocio. RF13. 2 Modificar elementos de la regla de negocio. RF13.3 Eliminar elementos de la regla de negocio. RF13.4 Listar elementos de la regla de negocio.
Actores del negocio	RF14. Gestionar actores del negocio.	RF14.1 Adicionar actores del negocio. RF14. 2 Modificar actores del negocio. RF14.3 Eliminar actores del negocio. RF14.4 Listar actores del negocio.
Trabajadores del negocio	RF15. Gestionar trabajadores del negocio.	RF15.1 Adicionar trabajadores del negocio. RF15. 2 Modificar trabajadores del negocio. RF15.3 Eliminar trabajadores del negocio. RF15.4 Listar trabajadores del negocio.

Diagrama de casos de usos del negocio	RF16. Gestionar diagrama de casos de usos del negocio.	RF16. 1 Cargar imagen del diagrama de casos de usos del negocio. RF16. 2 Modificar imagen del diagrama de casos de usos del negocio. RF16.3 Eliminar imagen del diagrama de casos de usos del negocio.
Caso de uso del negocio	RF17. Gestionar caso de uso del negocio.	RF17.1 Adicionar caso de uso del negocio. RF17. 2 Modificar caso de uso del negocio. RF17.3 Eliminar caso de uso del negocio. RF17.4 Listar caso de uso del negocio.
Diagrama de actividad	RF18. Gestionar diagrama de actividades.	RF18. 1 Cargar imagen del diagrama de actividades. RF18. 2 Modificar imagen del diagrama de actividades. RF18.3 Eliminar imagen del diagrama de actividades.
RF19. Exportar a formato pdf el producto de trabajo Modelo de Negocio con Caso de Uso. RF20. Exportar a formato doc el producto de trabajo Modelo de Negocio con Caso de Uso.		

Tabla 3 Requisitos funcionales del producto de trabajo Reglas de negocio.

Producto de trabajo: **Reglas de negocio**		
Estructura del documento	**Técnica de Agrupación funcional**	**Requisitos funcionales**
Portada	RF20. Gestionar elementos de la portada.	RF20. 1 Adicionar elementos de la portada RF20. 2 Modificar elementos de la portada RF20.3 Eliminar elementos de la portada RF20.4 Listar elementos de la portada
Control del documento	RF21. Gestionar control del documento.	RF21. 1 Adicionar elementos del control del documento. RF21. 2 Modificar elementos del control del documento. RF21.3 Eliminar elementos del control del documento. RF21.4 Listar elementos del control del documento.

Control de cambios	RF22. Gestionar control de cambios.	RF22. 1 Adicionar elementos del control de cambios. RF22. 2 Modificar elementos del control de cambios. RF22.3 Eliminar elementos del control de cambios. RF22.4 Listar elementos del control de cambios.
Introducción	RF23. Gestionar elementos de la introducción.	RF23. 1 Adicionar elementos de la introducción. RF23. 2 Modificar elementos de la introducción. RF23.3 Eliminar elementos de la introducción. RF23.4 Listar elementos de la introducción.
Tipos de Reglas del negocio	RF24. Gestionar tipos de regla de negocio.	RF24.1 Adicionar tipos de regla de negocio. RF24. 2 Modificar tipos de regla de negocio. RF24.3 Eliminar tipos de regla de negocio. RF24.4 Listar tipos de regla de negocio.
Listado de reglas del negocio	RF25. Gestionar listado de reglas de negocio.	RF25.1 Adicionar listado de reglas de negocio. RF25. 2 Modificar listado de reglas de negocio. RF25.3 Eliminar listado de reglas de negocio. RF25.4 Listar listado de reglas de negocio.
RF26. Exportar a formato pdf el producto de trabajo Reglas de negocio. RF27. Exportar a formato doc el producto de trabajo Reglas de negocio.		

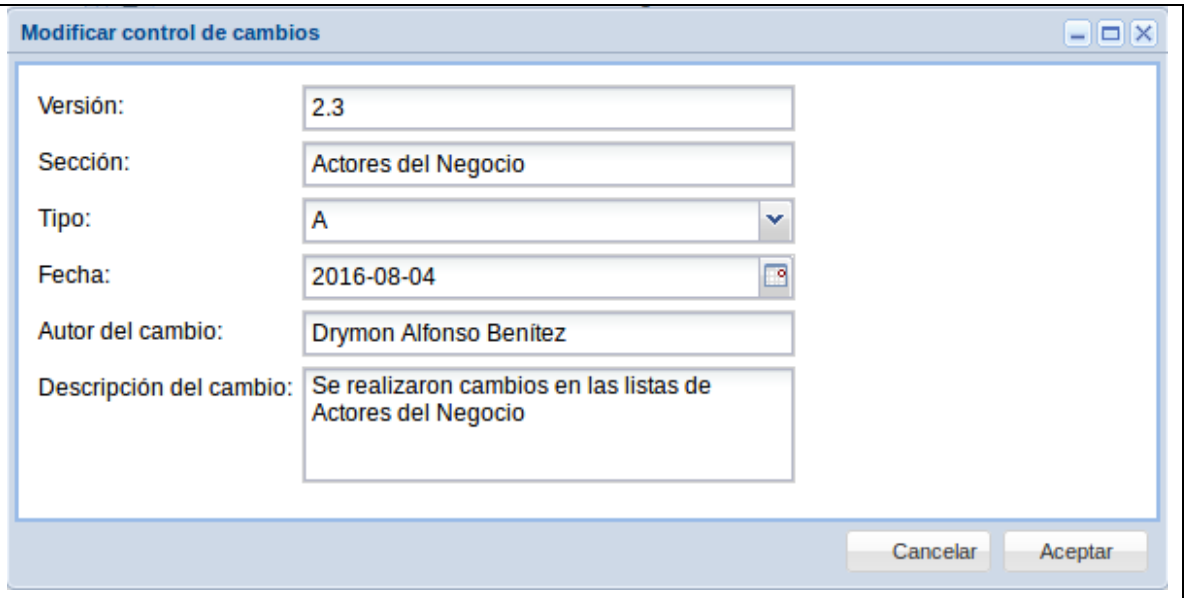
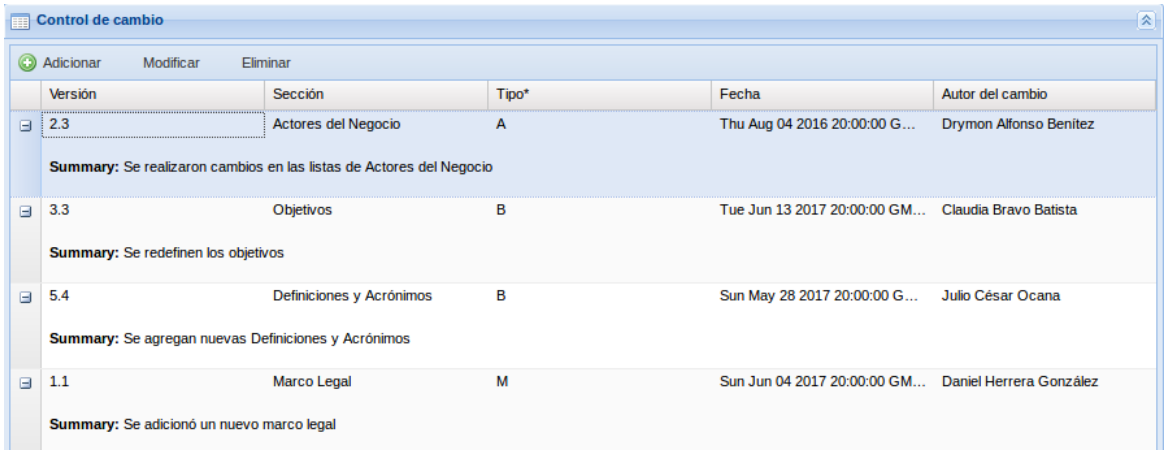
A continuación, se muestra la descripción del Requisito funcional **Modificar control de cambio**.

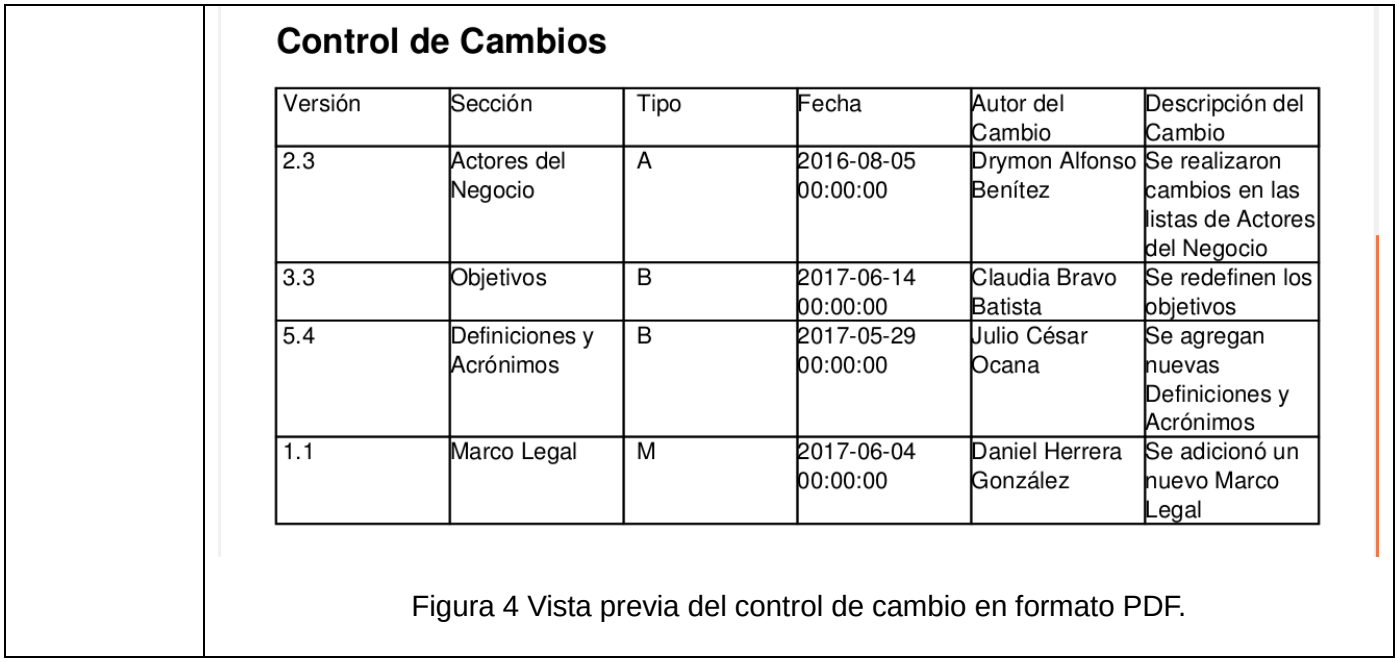
Tabla 4 Descripción del requisito funcional Modificar control de cambio.

Precondiciones	El usuario se ha autenticado en el sistema y cuenta con los permisos necesarios para ejecutar esta acción. El usuario selecciona la opción Descripción de procesos de negocio. Se ha registrado al menos un control de cambio.
1.	El usuario selecciona un control de cambio de los listados (Ver Interfaz de usuario Listado de control de cambios Figura 3)
2.	El usuario presiona el botón Modificar.
3.	El usuario introduce los valores a modificar (Ver Interfaz de usuario Modificar control de cambios Figura 2)
4.	El usuario presiona el botón Aceptar.
5.	El sistema verifica que los datos sean correctos.
6.	El sistema confirma que se modificaron los datos.

7.	El sistema actualiza el listado de control de cambio (Ver Interfaz de usuario Listado de control de cambios Figura 3)
8.	Concluye el requisito.
Pos-condiciones	
1.	Se modificó la solicitud.
Flujos alternativos	
Flujo alternativo 4.a El usuario cancela la modificación del control de cambio.	
4.	5. El sistema no modifica el control de cambio.
Pos-condiciones	
6.	7. N/A
Validaciones	
1	Una solicitud sobre un control de cambio con valores incorrectos no se realiza la modificación.

Conceptos	**Control de cambio**	**Utilizados internamente:** Versión Sección Tipo Fecha Autor del Cambio Descripción del cambio

Prototipo de Interfaz de usuario	 Figura 2 Modificar control de cambios.  Figura 3 Listado de control de cambio.



Control de Cambios

Versión	Sección	Tipo	Fecha	Autor del Cambio	Descripción del Cambio
2.3	Actores del Negocio	A	2016-08-05 00:00:00	Drymon Alfonso Benítez	Se realizaron cambios en las listas de Actores del Negocio
3.3	Objetivos	B	2017-06-14 00:00:00	Claudia Bravo Batista	Se redefinen los objetivos
5.4	Definiciones y Acrónimos	B	2017-05-29 00:00:00	Julio César Ocana	Se agregan nuevas Definiciones y Acrónimos
1.1	Marco Legal	M	2017-06-04 00:00:00	Daniel Herrera González	Se adicionó un nuevo Marco Legal

Figura 4 Vista previa del control de cambio en formato PDF.

2.4. Requisitos no Funcionales

2.4.1. Concepto de requisitos no Funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. Estos requisitos especifican o restringen las propiedades del sistema.

Requisitos no Funcionales

Software

RNF1. Navegador web Mozilla Firefox, en su versión 42.0 o superior.

RNF2. Gestor de Bases de datos: El sistema de base de datos a utilizar es MySQL, en su versión 5.6.12 o superior.

RNF3. Para el servidor el Sistema Operativo será: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior.

RNF4. Servidor web Apache en su versión 2.0 o superior, con los módulos de PHP 5.5 disponibles.

Hardware

Para el servidor:

RNF5. Procesador Pentium III a 1 GHz de velocidad de procesamiento y 1Gb de memoria RAM.

RNF6. Procesador Pentium II a 1 GHz de velocidad de procesamiento y 128 MB de memoria RAM.

Seguridad

RNF7. Autenticación (Contraseña de acceso). Protección contra acciones no autorizadas o que puedan afectar la integridad de los datos.

El marco de trabajo Bosón provee un componente SeguridadBundle que es el encargado de proveer las funcionalidades básicas de autenticación y autorización. En el componente de Seguridad, el paquete de autorización es el encargado de validar si las credenciales mostradas por el usuario conectado son suficientes para acceder a determinado recurso del proyecto. El modelo utilizado es basado en roles, los que son almacenados en base de datos y asociados a cada uno de los usuarios en una relación de muchos a muchos. A continuación se muestra la interfaz principal de autenticación del Marco de trabajo Bosón.

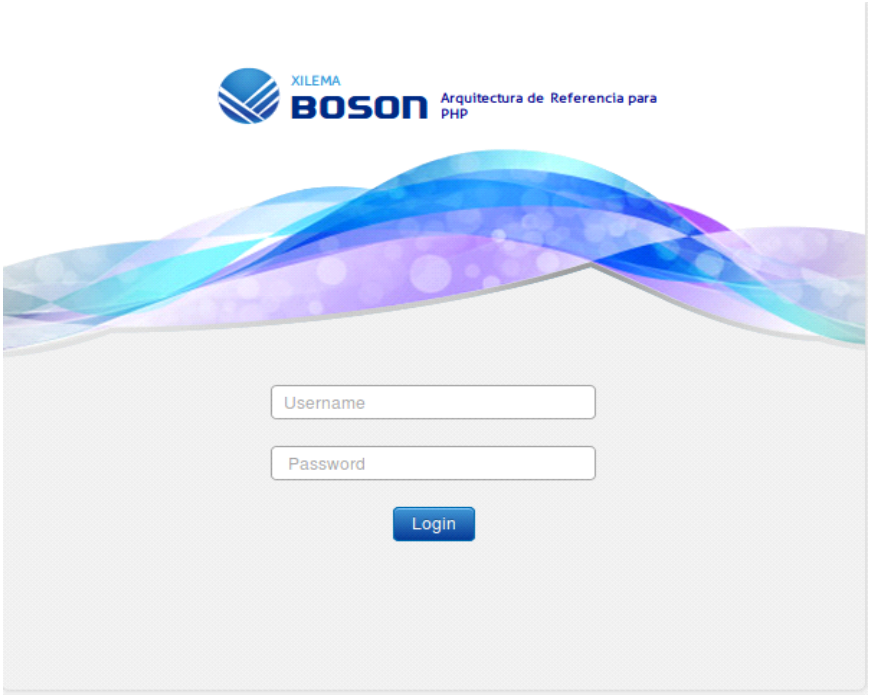


Figura 5 Interfaz principal de autenticación

2.5. Modelado del diseño

El modelo de diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es utilizado como entrada esencial en las actividades relacionadas con la implementación. El modelo de diseño puede contener: diagramas, clases, paquetes, subsistemas, interfaces, relaciones y los atributos (Pressman, 2010)

En la Figura 6 se modela el Diagrama de clases del diseño para el producto de trabajo Caso de Uso del negocio, donde se representan las clases y las relaciones que existen entre las mismas. En la clase app.php que trae por defecto el marco de trabajo Symfony 2, que es el controlador frontal que se

encuentra en el directorio web raíz del proyecto y a través del cual se atienden todas las peticiones del usuario, se construye la clase AppKernel, que es la clase de Symfony que registra todos los Bundles de la aplicación. Las clases de presentación se encuentran representadas por los ficheros principales del marco de trabajo ExtJS en su versión 6.0, el ext-all.js que contiene la implementación de los componentes y theme-classic-all.css que tiene los temas y estilos del framework. Las clases del negocio se encuentran representadas por los controladores que se encuentran en el directorio Controller y que implementa las funcionalidades de la aplicación. Por último, las clases entidades que se encuentran en el directorio Entity, contenedoras de la información de cada uno de los productos de trabajos que se deben generar, relacionadas con cada una de las clases controladoras que las crean.

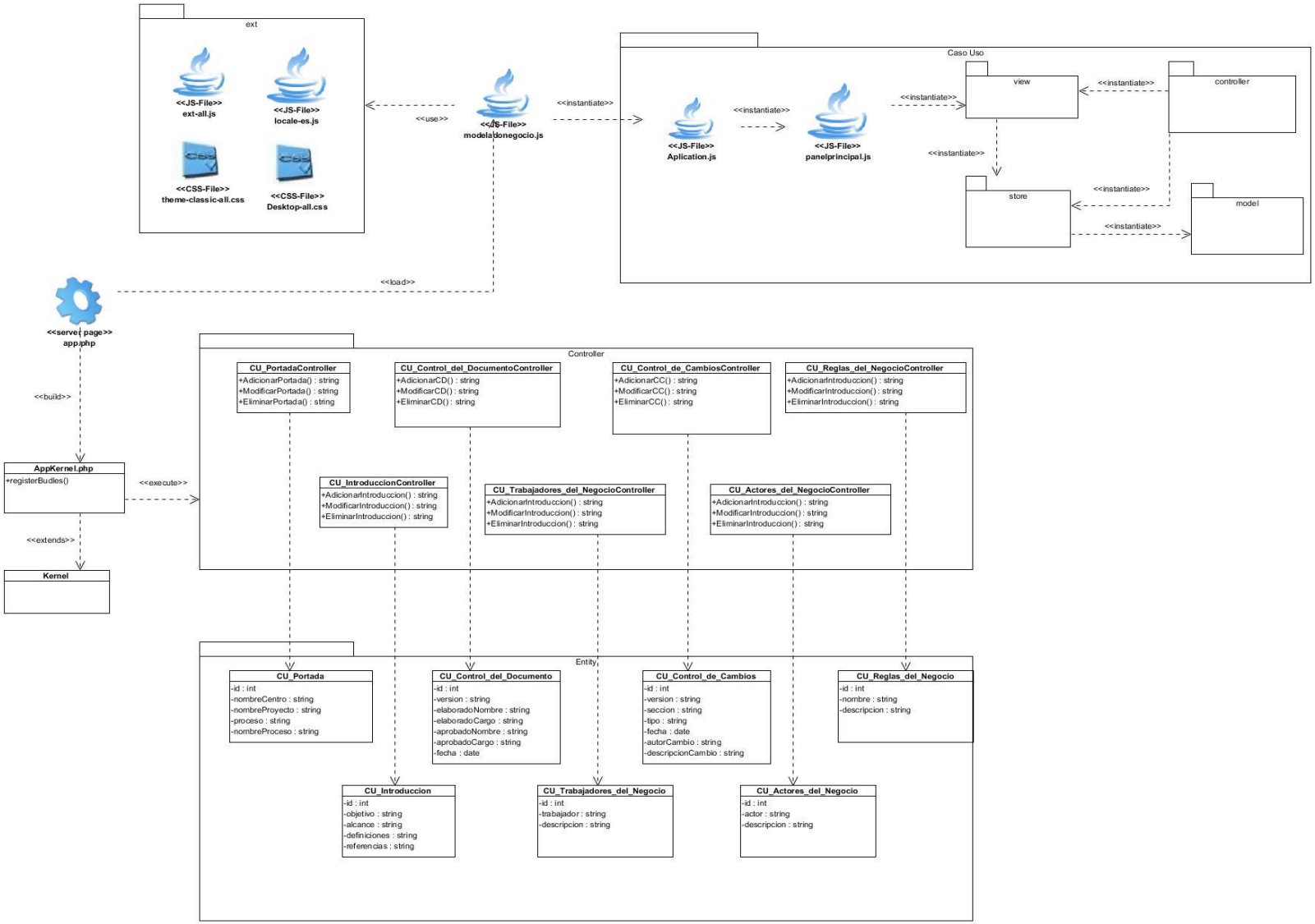


Figura 6 Diagrama de clases del diseño para el producto de trabajo Caso de Uso

2.6. Arquitectura de la herramienta para generar productos de trabajos

2.6.1 Estructura de directorios

La estructura de directorios se caracteriza por ser una aplicación de Symfony 2.

app/ La configuración de la aplicación

src/ El código PHP del proyecto.

vendor/ Las dependencias de terceros

web/ El directorio raíz

El kernel requiere primero al archivo *bootstrap.php*, el cual inicia el framework y registra al *autoloader*. El controlador frontal app.php usa una clase Kernel, AppKernel para reiniciar la aplicación.

La clase AppKernel es el punto de ingreso principal a la configuración de la aplicación y como tal, se ubica en el directorio app/. Esta clase debe implementar dos métodos:

- registerBundles(): Debe devolver un array de todos los bundles necesarios para correr la aplicación.
- registerContainerConfiguration(): Carga la configuración.

La auto-carga de PHP puede ser configurada por medio de app/autoload.php, quedando implementada de la siguiente forma:

```
<?php
use Doctrine\Common\Annotations\AnnotationRegistry;
use Composer\Autoload\ClassLoader;
/**
 * @var ClassLoader $loader
 */
$loader = require __DIR__.'/../vendor/autoload.php';
AnnotationRegistry::registerLoader(array($loader, 'loadClass'));
return $loader;
```

La clase Symfony\Component\ClassLoader\UniversalClassLoader se utiliza para autocargar archivos. Las dependencias están ubicadas dentro del directorio *vendor/*.

2.6.2 Patrón arquitectónico

El patrón Modelo Vista Controlador (MVC) consiste en tres tipos de objetos: el modelo que es el objeto de la aplicación, la vista que es su representación y el controlador que define el modo en que la interfaz reacciona a la entrada del usuario. El mismo permite mejorar la organización del código, la posibilidad de representar de diferentes formas la misma información, como se describe en la tabla 5. Descripción de los patrones arquitectónicos.

Tabla 5 Descripción de los patrones arquitectónicos.

Controlador	El Controlador frontal, los filtros, las acciones y los objetos request, response y session. En el bundle MetodologiasBundle se evidencia el patrón en las clases controladoras que se encuentran dentro del directorio Controller.
Vista	El Layout de la aplicación, las plantillas de los módulos, los slots, los partials, los componentes y los helpers. En el bundle MetodologiasBundle se evidencia el patrón en las clases js que se encuentran dentro del directorio Resources, views.
Modelo	El ORM, los formularios, las extensiones propias que el programador realice de las clases del ORM y las clases y funciones propias que el programador construya para implementar la lógica de negocio. En el bundle MetodologiasBundle se evidencia el patrón en las clases entidades que se encuentran dentro del directorio Entity.

2.7. Patrones de diseño utilizados

Patrones Generales de Asignación de Responsabilidades GRASP (Acrónimo de General Responsibility Assignment Software Patterns), describen los principios fundamentales para asignar responsabilidades a los objetos. A continuación, se describen los utilizados en el diseño de la solución.

Experto: Es uno de los patrones que más se utiliza en Symfony, con la inclusión de la librería Propel para mapear la base de datos. Symfony utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan. Por lo cual, se evidencia el patrón en las clases entidades ControlDocumento y ControlCambios, expertas en la información del control y los cambios del documento.

Creador: En la clase Actions se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase Actions es “creador” de dichas entidades. Por lo cual, se evidencia el patrón en las clases ControlDocumentoController.php y ControlCambiosController.php encargadas de crear las entidades ControlDocumento y ControlCambios.

Controlador: la clase controladora se encarga de llevar el control de todos los eventos relacionados con el negocio. Implementa las funcionalidades que dan respuesta a las peticiones del usuario. Por lo cual, se evidencia el patrón en las clases ControlDocumentoController.php y ControlCambiosController.php.

Alta Cohesión: Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase Actions, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las properties.

Bajo Acoplamiento: La clase Actions hereda únicamente de sfActions para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

El uso de estos patrones garantizó asignar a cada clase la responsabilidad que le corresponde, obtener el menor número de relaciones y dependencias entre clases y aumentar las posibilidades de reusabilidad de las mismas.

Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación, a raíz del libro “Design Patterns-Elements of Reusable Software” de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides, son los patrones de la pandilla de los cuatro (GoF, gang of four). A continuación, se describen los utilizados en el diseño de la solución.

Patrón Singleton: Clase sfRouting – método getInstance Esta clase la utiliza el controlador frontal (sfWebFrontController) y se encarga de enrutar todas las peticiones que se hagan a la aplicación. El singleton sfRouting precisa otros métodos muy útiles para la gestión manual de las rutas: ClearRoutes (), hasRoutes (), getRoutesByName ().

Patrón Command: Este patrón se observa en la clase sfWebFrontController, en el método dispatch(). Esta clase está por defecto y es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. Este patrón se aplica además en la clase sfRouting, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el framework, la cual se puede activar o desactivar. En este método es parseada la URL con el objetivo de precisar los parámetros de la misma y de esta forma saber el Actions que debe responder a la petición.

Patrón Decorator: Este método pertenece a la clase abstracta sfView, padre de todas las vistas, que contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo nombrado layout.php es el que contiene el Layout de la página. Este archivo, conocido también como plantilla global, guarda el código HTML que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página.

Conclusiones del capítulo

En este capítulo fueron diseñadas las interfaces gráficas de usuarios de los componentes, representando las clases y las relaciones entre las mismas para una mejor definición de la

implementación. Se elaboró el modelo conceptual de la solución. Se realizó la descripción de los requisitos funcionales y no funcionales identificados durante el proceso de Ingeniería de requisitos. La definición de la arquitectura del sistema mediante el patrón arquitectónico MVC permitió delimitar la vista, en la cual se construye las interfaces gráficas de usuarios, la lógica del negocio para la definición de los controladores que gestionan las operaciones de los componentes, y el modelo para la persistencia de la información. Se elaboró y describió el diagrama clases, los modelos de datos con que contará la solución y las métricas de validación del diseño asociadas al mismo.

CAPÍTULO 3: IMPLEMENTACIÓN, PRUEBA Y VALIDACIÓN DE LA HERRAMIENTA PARA GENERAR PRODUCTOS DE TRABAJOS

3.1. Introducción

En el presente capítulo se describen los estándares de codificación empleados durante la implementación de las interfaces gráficas de los componentes para garantizar el entendimiento y legibilidad del código, y los componentes necesarios para realizar la implementación y despliegue de la solución. Se muestran los resultados de las pruebas efectuadas al software y se presenta la estrategia que se sigue para la validación de la investigación.

3.2. Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizados; y cómo dependen los componentes unos de otros.

3.3. Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. (Alfonso, 2008).

Los marcos de trabajo Bosón y Symfony establecen el uso de los estándares de codificación PSR-0, PSR-1 y PSR-2. Algunos de los elementos generales contemplados en estos estándares son:

- Los archivos deben utilizar solamente las etiquetas <?php y <?=.
- Los archivos deben emplear solamente la codificación UTF-8 para el código PHP.
- Las constantes de las clases deben declararse en mayúsculas con guiones bajos como separadores.
- El código debe usar cuatro espacios como indentación, no tabuladores.
- La visibilidad debe estar declarada en todas las propiedades y métodos; abstracto y final deben estar declaradas antes de la visibilidad; estático debe estar declarada después de la visibilidad.

Estándar para clases

Después de incluir los **namespace** y los **use**, se añade la información de la clase de la siguiente manera:

```
/**
* Descripción
*
```

* @author nombre <usuario@dominio.dom>

*/

La información debe estar contenida entre los signos /** */. Se declaran 2 bloques, separados y diferenciados entre sí por un caracter de espacio *. En el primer bloque se ofrece una breve descripción del propósito de la clase. En el segundo bloque se ofrecen los datos del autor de la clase. Esta información del autor es opcional.

Estándar para métodos

La estructura que debe añadirse cuando se declara un método o función:

/**

* Descripción

*

* @**param** tipo $variable

* @**param** tipo $variable2

*

* @**return** tipo

*/

function miFuncion ($variable, $variable2) {...}

La información está contenida entre los signos /** */. Se declaran 3 bloques de datos, cada uno con sus funciones específicas. Los bloques se diferencian y separan entre sí por un caracter de espacio *. En el primer bloque se ofrece una breve descripción del propósito del método o función. En el segundo bloque se especifican los parámetros que recibe, especificando para cada uno su tipo y su nombre. En el tercer bloque se especifica el o los resultados que devuelve el método o función. Los nombres de los métodos deben ser declarados utilizando la notación camelCase.

3.4. Métricas de Validación del Diseño

La métrica TOC está dada por la cantidad de operaciones o métodos asignados a una clase y evalúa los atributos Responsabilidad, Complejidad de Implementación y Reutilización. Para el caso del componente DPN se analizaron un conjunto de 4 clases, obteniendo el valor 3.5 como promedio en cantidad de procedimientos. Luego de aplicar los criterios de medición correspondientes a la métrica se obtienen los siguientes resultados:

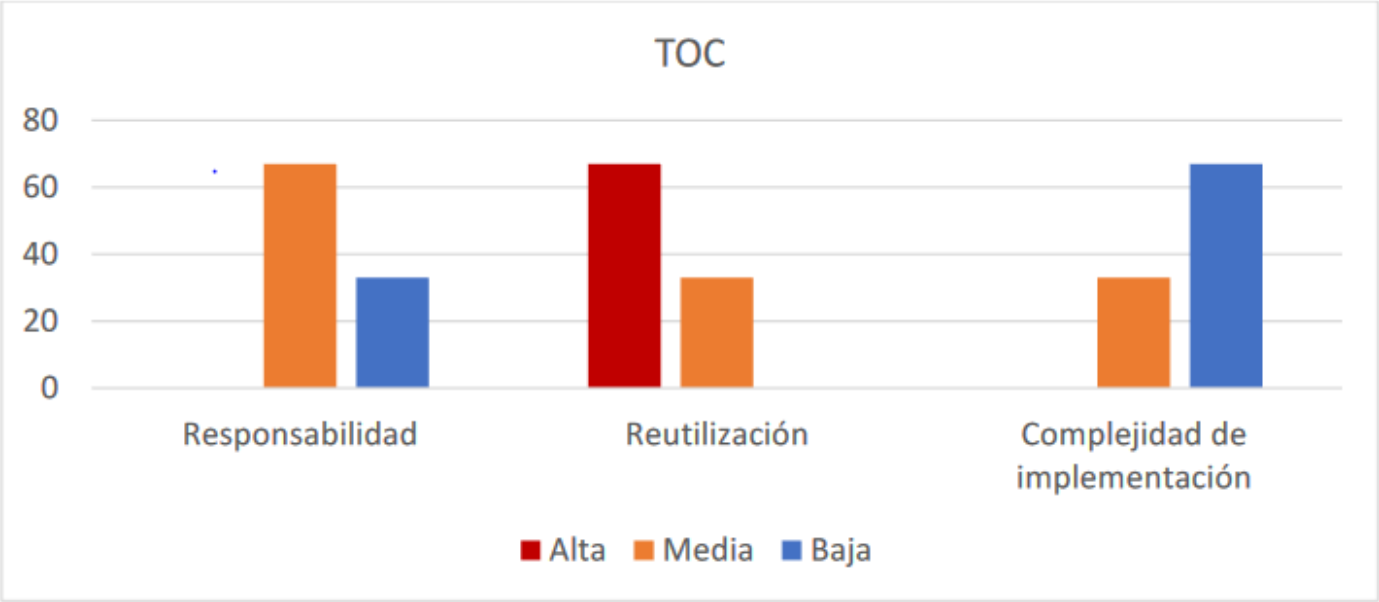


Figura 7 Resultado de las métricas de validación del diseño TOC

La métrica RC está dada por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas. Para el caso del componente DPN se analizaron un conjunto de 4 clases, obteniendo el valor 0.2 como promedio de relaciones de uso entre las mismas. Luego de aplicar los criterios de medición correspondientes a la métrica se obtienen los siguientes resultados:

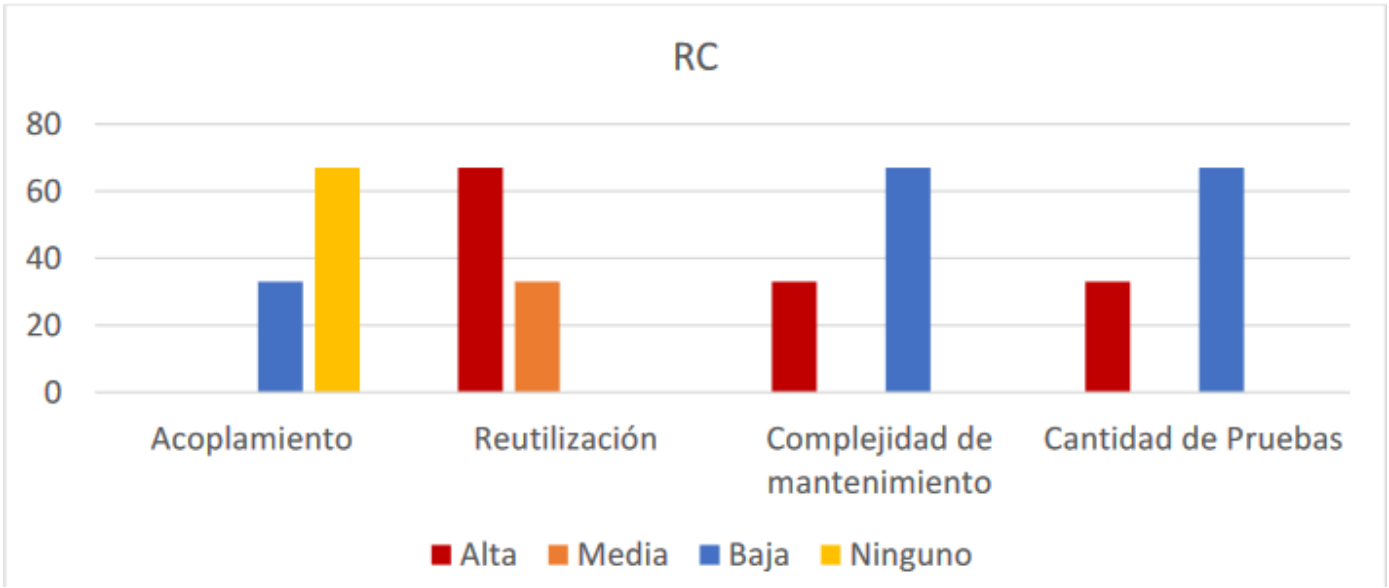


Figura 8 Resultado de las métricas de validación del diseño RC

Analizando los resultados obtenidos, luego de aplicadas las métricas es posible observar que el diseño ofrece una responsabilidad media, baja complejidad de implementación, alta reutilización, bajo acoplamiento, baja complejidad de mantenimiento y un bajo indicador en cantidad de pruebas. Los resultados obtenidos indican que aproximadamente el 67% de las clases del componente se encuentran

desacopladas, lo cual se valora de positivo para el diseño de la solución pues existen muy pocas relaciones de uso entre los nuevos elementos que se adicionan. Esto trae como consecuencia que al existir cambios en la solución la repercusión en el resto de los elementos es mínima. El bajo acoplamiento de elementos favorece a la reutilización, la cual es alta en un 67% de las clases del componente. Por otra parte, la complejidad de mantenimiento, de implementación, así como la cantidad de pruebas arrojan valores mayormente bajos. Esto quiere decir que se requiere de poco esfuerzo en estas tareas. Teniendo en cuenta los resultados obtenidos y lo que representan cada uno para la solución, es posible afirmar que el diseño propuesto tributa al desarrollo de una aplicación con calidad.

3.5. Modelo de Componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre los mismos. Los componentes físicos incluyen archivos, módulos, ejecutables, paquetes, entre otros. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema (Somerville, 2005). El modelo de componentes del Modelado de negocio, se encuentra representado por los componentes Modelo del Negocio con CU, Descripción de procesos de Negocio y Reglas de Negocio, como se muestra en la Figura 9.

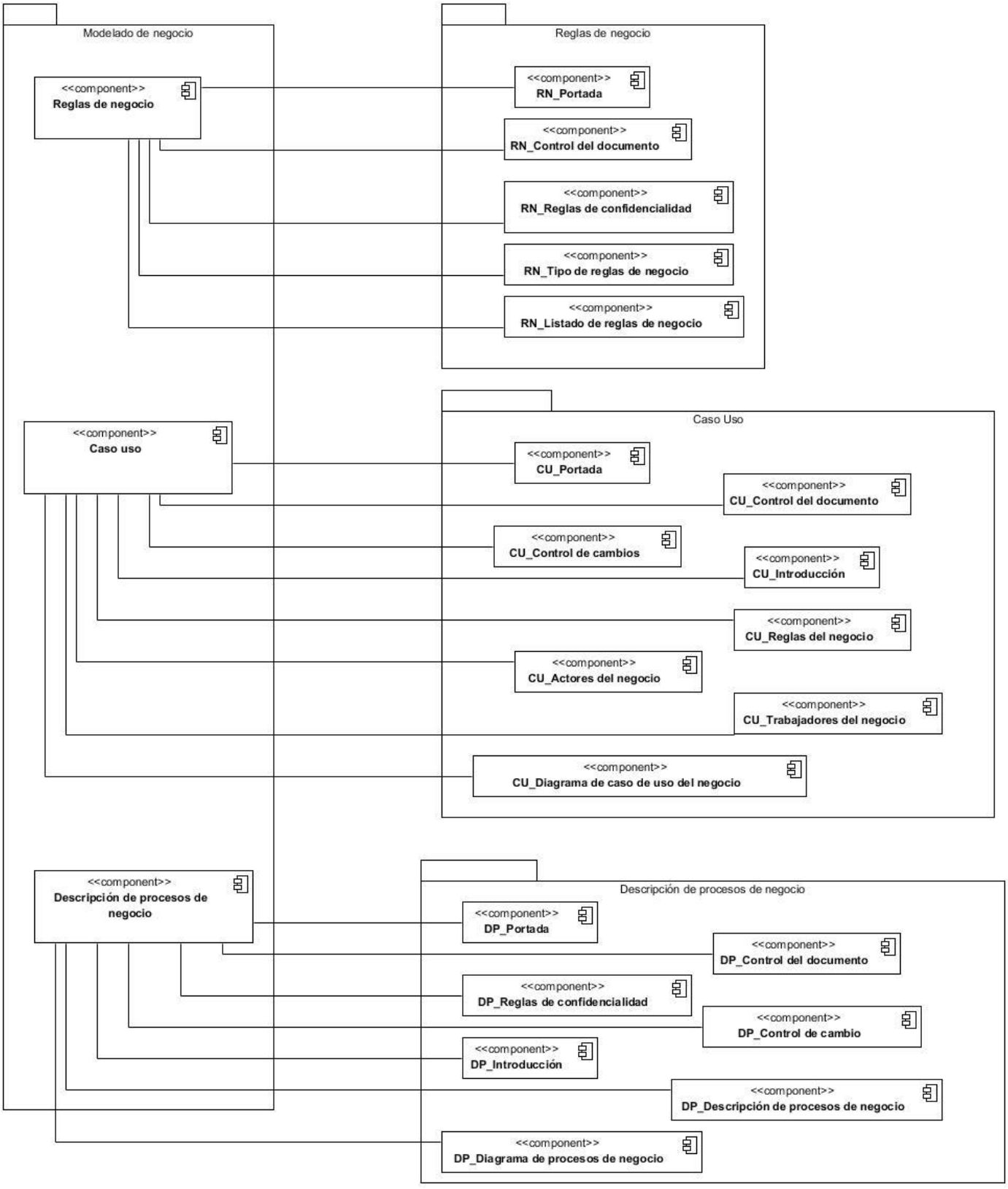


Figura 9 Modelo de componentes del Modelado de negocio.

3.6. Modelo de datos

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, permite describir el tipo de datos que incluye la base y la forma en que se relacionan, las restricciones de integridad y las operaciones de manipulación de los datos. En la Figura 10 se muestra el Modelo de datos para el componente Caso de uso del Negocio.

El modelo está representado por las entidades CU_Portada, CU_Control_del_Documento, CU_Control_de_Cambios, CU_Trabajadores_del_Negocio, CU_Casos_de_Uso_del_Negocio, CU_Actores_del_Negocio, CU_Introduccion, y CU_Reglas_del_Negocio.

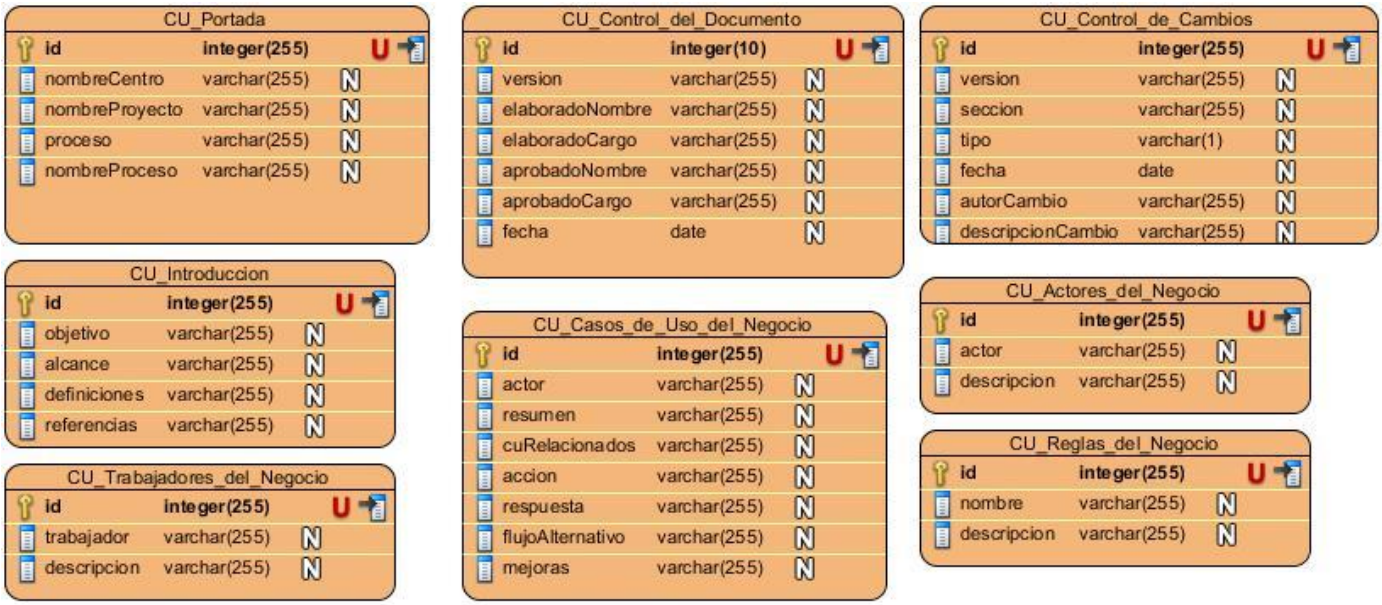


Figura 10 Diagrama entidad relación del producto de trabajo Caso de uso del negocio.

3.7. Modelo de despliegue

El diagrama de despliegue permite modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes (nodos). Los nodos son objetos físicos que existen en tiempo de ejecución y que representan algún tipo de recurso computacional, también pueden ser dispositivos del sistema. El diagrama de despliegue de la solución propuesta se representa por una PC cliente que solicita peticiones mediante el protocolo HTTPS al servidor de aplicaciones: Apache, el cual se conecta al servidor de base de datos MySQL para extraer la información, como se muestra en la Figura 11 Diagrama de despliegue.

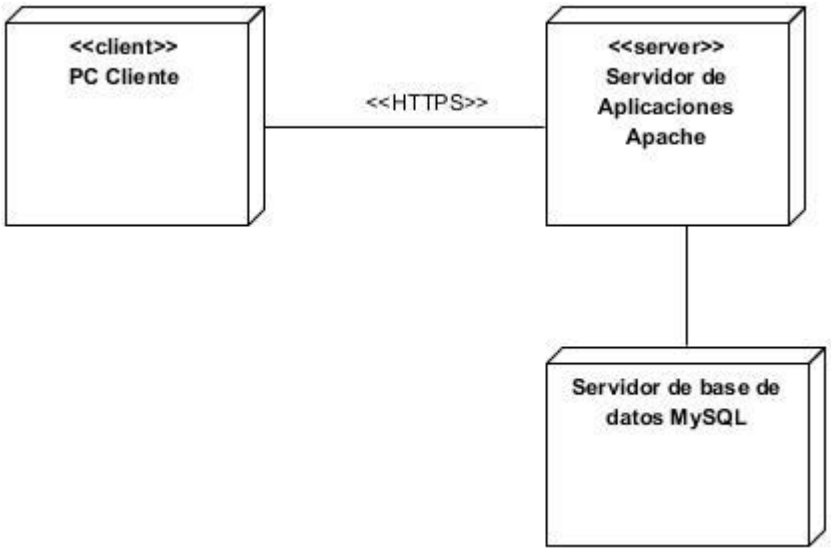


Figura 11 Diagrama de despliegue.

3.8. Automatización de pruebas unitarias de códigos PHP

Las pruebas unitarias de software permiten evaluar por separado el correcto funcionamiento de los códigos que lo componen. Existen dos enfoques principales para el diseño de pruebas unitarias. Un enfoque de caja blanca orientado a la estructura del código y otro enfoque de caja negra orientado al correcto funcionamiento de éste a partir del análisis de entradas y salidas que posee y verificando que el resultado es el esperado.

3.8.1. Estrategia de diseño de pruebas

La estrategia se fomenta en un procedimiento dividido en pasos bajo los cuales el desarrollador o encargado de pruebas podrá seleccionar sus archivos de código PHP o módulos de clase, archivos que contienen funciones que ejecutan tareas específicas entregando resultados a partir de una serie de posibles entradas de datos.

La prueba unitaria busca responder si las funciones que componen el código cumplen o no con el comportamiento esperado. A través de esta estrategia de automatización el proceso será automatizado y se apoyará en el framework de pruebas unitarias PHPUnit.

El proceso además llevará a cabo el diseño de los casos de prueba de la función y la ejecución de éstos con el fin de obtener resultados que serán organizados y presentados en forma de reportes al encargado de las pruebas para apoyarlo a tomar decisiones basadas en estos reportes. La estrategia de diseño se compone de los pasos que se detallan a continuación.

Paso 1: Seleccionar Módulos de Clase: Un módulo de clase es la unidad de código más pequeña a la cual puede realizarse una prueba unitaria, como se muestra en la Figura 12.

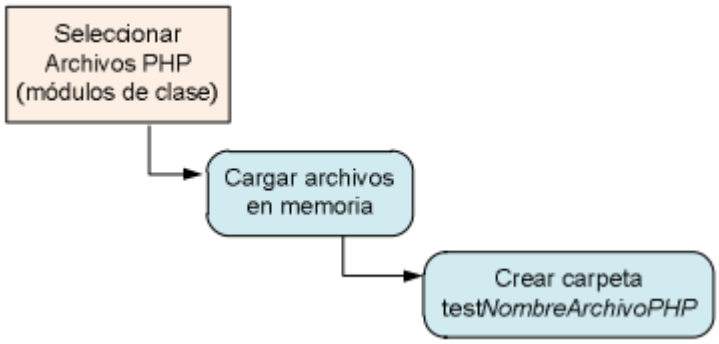


Figura 12 Seleccionar módulos de clase

Paso 2: Validar estándar de codificación: Después de realizar la selección de los módulos de clase PHP comienza un ciclo iterativo por cada uno de los archivos de código PHP seleccionados. El archivo pasa por una validación del estándar de codificación propuesto, como se muestra en la Figura 13.

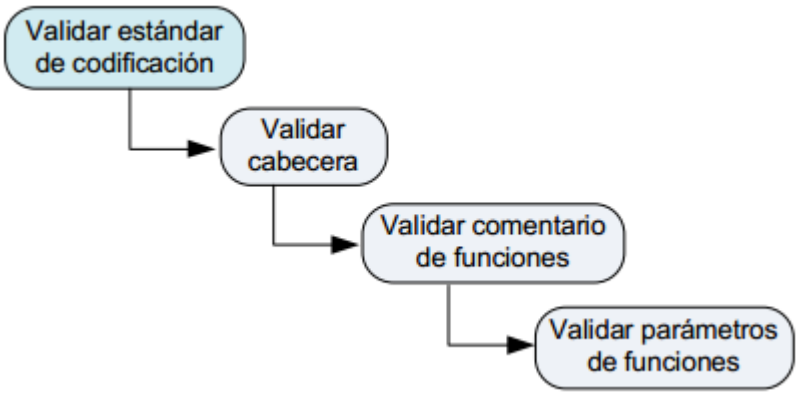


Figura 13 Validar estándar de codificación

Paso 3: Identificar tipos de parámetros: Después de validar el estándar de codificación del archivo seleccionado, se detectan las funciones o métodos que lo componen y por cada parámetro de cada función o método se identifica el tipo de parámetro, como se muestra en la Figura 14.

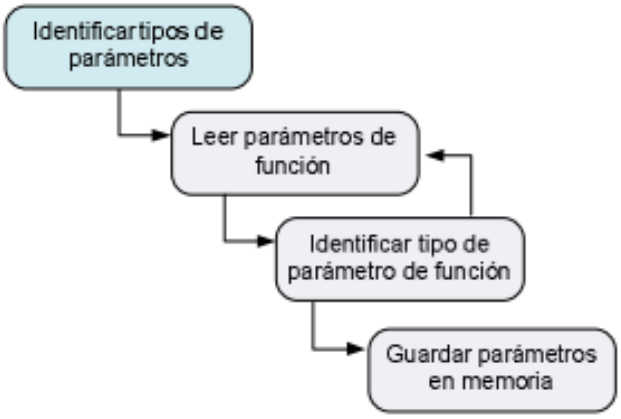


Figura 14 Identificar tipos de parámetros.

Paso 4: Identificar Asserts: Luego de identificar los tipos de parámetros de las funciones se identifican los posibles asserts a partir de estos. Un assert es la salida o resultado esperado de cierta entrada en la función o método a probar. En PHPUnit son una colección de métodos estáticos para verificar los valores actuales con los valores esperados, como se muestra en la Figura 15.

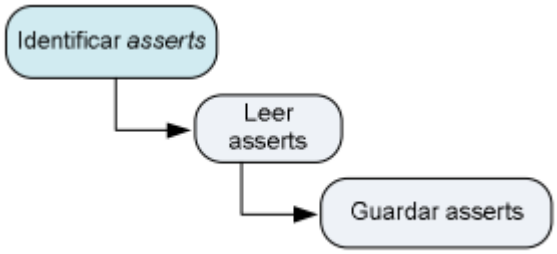


Figura 15 Identificar Asserts.

Paso 5: Generar casos de Prueba: Se genera o diseña una clase con los casos de prueba del archivo PHP a partir del uso de la librería PHPUnit y su componente PHPUnit Framework TestCase, como se muestra en la Figura 16.

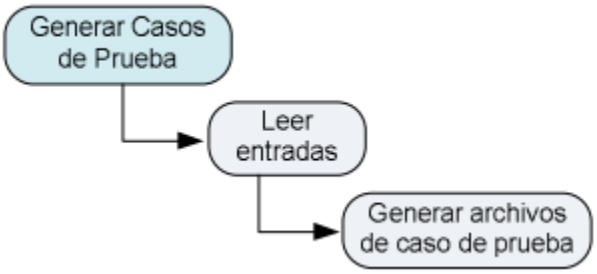


Figura 16 Generar casos de Prueba.

Paso 6: Ejecutar Prueba con PHPUnit: Se ejecuta la prueba mediante el Framework de pruebas PHPUnit. Se valida e inspeccionan los archivos de casos de prueba definidos, verificando que se han

contemplado todas las pruebas necesarias, que no existen pruebas duplicadas o implícitas en otras y que el tiempo estimado no sobrepasa la fecha límite de ejecución.

Se invoca el ejecutable phpunit.bat y se envía la ruta en que se encuentra el archivo a probar. PHPUnit generará un log que puede ser capturado y enviado a un archivo temporal para preparar los resultados en el siguiente paso. Este proceso se muestra a continuación en la Figura 17.

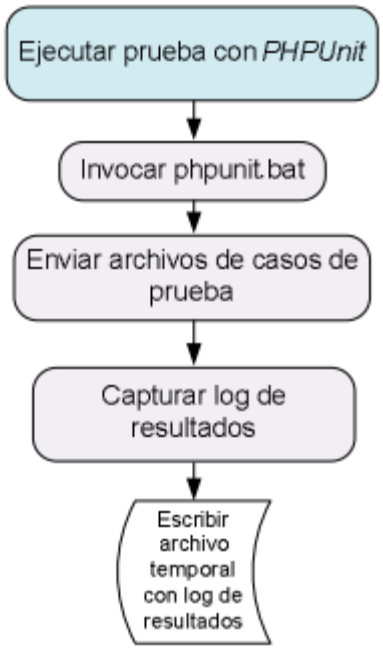


Figura 17 Preparar Resultado.

Paso 7: Preparar Resultados: Por cada una de las pruebas ejecutadas se produce un log de resultados. Dicho log indicará:

Número de pruebas: El número de pruebas que han sido llevadas a cabo mediante la invocación de PHPUnit y los archivos de casos de prueba en PHP.

Número de aciertos: Es el número de resultados esperados a partir de los datos de entrada ingresados después de una ejecución de prueba mediante PHPUnit.

Número de fallos: Es el número de resultados no esperados a partir de los datos de entrada ingresados después de una ejecución de prueba mediante PHPUnit.

3.9. Resultado de las pruebas

Para la ejecución de las pruebas se realizaron los siguientes pasos.

1. Descargar e instalar el phpunit-3.1.3. phar.
2. Configurar el phpunit-skelgen-2.0.1. phar en Settings > PHP > PHPUnit.
3. Cargar el autoload de las librerías.

4. Instalar y configurar php-xdebug.
5. Programar los tests como se muestra en el siguiente ejemplo.

En la ejecución de las pruebas unitarias se parte del hecho de que cada unidad funciona correctamente y eficientemente por separado. Que el código hace lo que tiene que hacer, se verifica que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve y que si el estado inicial es válido entonces el estado final es válido.

Nombre del test: ReglasDelNegocioControllerTest.php

```
<?php
namespace Metodologias\MetodologiasBundle\Tests\Controller;
use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;
class ReglasDelNegocioControllerTest extends WebTestCase
{
    public function testCompleteScenario()
    {
        // Se crea un nuevo cliente para examinar la aplicación
        $client = static::createClient();
        // Se crea una nueva entrada en la Base de Datos
        $crawler = $client->request('GET', '/reglasdelnegocio/');
        $this->assertEquals(200, $client->getResponse()->getStatusCode(), "Unexpected HTTP status code for GET /reglasdelnegocio/");
        $crawler = $client->click($crawler->selectLink('Create a new entry')->link());
        // Se llena el formulario con datos y se envía
        $form = $crawler->selectButton('Create')->form(array(
        'metodologias_metodologiasbundle_reglasdelnegocio[nombre]'  => 'Test',
        'metodologias_metodologiasbundle_reglasdelnegocio[descripcion]'  => 'Test',
        ));
        $client->submit($form);
        $crawler = $client->followRedirect();
        // Se revisan los datos en las vistas
```

```
        $this->assertGreaterThan(0, $crawler->filter('td:contains("Test")')->count(), 'Missing element td:contains("Test")');

        // Se prueba modificar el contenido introduciendo otros datos
        $crawler = $client->click($crawler->selectLink('Edit')->link());

        $form = $crawler->selectButton('Update')->form(array(
        'metodologias_metodologiasbundle_reglasdelnegocio[nombre]'  => 'Foo',
        'metodologias_metodologiasbundle_reglasdelnegocio[descripcion]'  => 'Foo',
        ));
        $client->submit($form);
        $crawler = $client->followRedirect();

        // Se comprueba que el elemento contiene un atributo con valor igual a
        // los datos introducidos
        $this->assertGreaterThan(0, $crawler->filter('[value="Foo"]')->count(), 'Missing element [value="Foo"]');

        // Se prueba la funcionalidad de borrar una entidad
        $client->submit($crawler->selectButton('Delete')->form());
        $crawler = $client->followRedirect();

        // Se revisa si la entidad fue borrada de forma exitosa
        $this->assertNotRegExp('/Foo/', $client->getResponse()->getContent());
    }
}
```

6. Ejecutar los tests, presentando como resultado la siguiente.

Se muestra que se ha probado exitosamente todo el código de las clases controladoras, las entidades y los formularos. Así como sus correspondientes funcionalidades y métodos y las clases que utilizan. Obteniendo como resultado una tasa elevada de efectividad.

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	689 / 689		100.00%	172 / 172		100.00%	24 / 24
Controller		100.00%	521 / 521		100.00%	80 / 80		100.00%	8 / 8
Entity		100.00%	98 / 98		100.00%	68 / 68		100.00%	8 / 8
Form		100.00%	70 / 70		100.00%	24 / 24		100.00%	8 / 8
MetodologiasBundle.php		n/a	0 / 0		n/a	0 / 0		n/a	0 / 0

Legend

Low: 0% to 50% **Medium**: 50% to 90% **High**: 90% to 100%

Generated by php-code-coverage 5.2.1 using PHP 7.0.18-0ubuntu0.16.04.1 with Xdebug 2.4.0 and PHPUnit 6.1.3 at Thu Jun 8 16:07:16 CDT 2017.

Se muestra que se ha probado exitosamente el 100% de las líneas de código: funcionalidades, y clases que emplea específicamente las clases controladoras de la aplicación. Obteniendo como resultado una tasa elevada de efectividad.

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	521 / 521		100.00%	80 / 80		100.00%	8 / 8
ActoresDeNegocioController.php		100.00%	59 / 59		100.00%	10 / 10		100.00%	1 / 1
CasosDeUsoDelNegocioController.php		100.00%	66 / 66		100.00%	10 / 10		100.00%	1 / 1
ControlDeCambiosController.php		100.00%	66 / 66		100.00%	10 / 10		100.00%	1 / 1
ControlDelDocumentoController.php		100.00%	66 / 66		100.00%	10 / 10		100.00%	1 / 1
IntroduccionController.php		100.00%	66 / 66		100.00%	10 / 10		100.00%	1 / 1
PortadaController.php		100.00%	66 / 66		100.00%	10 / 10		100.00%	1 / 1
ReglasDelNegocioController.php		100.00%	66 / 66		100.00%	10 / 10		100.00%	1 / 1
TrabajadoresDelNegocioController.php		100.00%	66 / 66		100.00%	10 / 10		100.00%	1 / 1

Legend

Low: 0% to 50% **Medium**: 50% to 90% **High**: 90% to 100%

Generated by php-code-coverage 5.2.1 using PHP 7.0.18-0ubuntu0.16.04.1 with Xdebug 2.4.0 and PHPUnit 6.1.3 at Thu Jun 8 16:07:16 CDT 2017.

Se muestra que se ha probado exitosamente el 100% de las líneas de código: funcionalidades, y clases que emplea específicamente las entidades de la aplicación. Obteniendo como resultado una tasa elevada de efectividad.

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	98 / 98		100.00%	68 / 68		100.00%	8 / 8
ActoresDeNegocio.php		100.00%	7 / 7		100.00%	5 / 5		100.00%	1 / 1
CasosDeUsoDelNegocio.php		100.00%	16 / 16		100.00%	11 / 11		100.00%	1 / 1
ControlDeCambios.php		100.00%	19 / 19		100.00%	13 / 13		100.00%	1 / 1
ControlDelDocumento.php		100.00%	16 / 16		100.00%	11 / 11		100.00%	1 / 1
Introduccion.php		100.00%	13 / 13		100.00%	9 / 9		100.00%	1 / 1
Portada.php		100.00%	13 / 13		100.00%	9 / 9		100.00%	1 / 1
ReglasDelNegocio.php		100.00%	7 / 7		100.00%	5 / 5		100.00%	1 / 1
TrabajadoresDelNegocio.php		100.00%	7 / 7		100.00%	5 / 5		100.00%	1 / 1

Legend

Low: 0% to 50% **Medium**: 50% to 90% **High**: 90% to 100%

Generated by php-code-coverage 5.2.1 using PHP 7.0.18-0ubuntu0.16.04.1 with Xdebug 2.4.0 and PHPUnit 6.1.3 at Thu Jun 8 16:07:16 CDT 2017.

Se muestra que se ha probado exitosamente el 100% de las líneas de código: funcionalidades, y clases que emplea específicamente los formularios de la aplicación. Obteniendo como resultado una tasa elevada de efectividad.

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	70 / 70		100.00%	24 / 24		100.00%	8 / 8
ActoresDeNegocioType.php		100.00%	7 / 7		100.00%	3 / 3		100.00%	1 / 1
CasosDeUsoDelNegocioType.php		100.00%	10 / 10		100.00%	3 / 3		100.00%	1 / 1
ControlDeCambiosType.php		100.00%	11 / 11		100.00%	3 / 3		100.00%	1 / 1
ControlDelDocumentoType.php		100.00%	10 / 10		100.00%	3 / 3		100.00%	1 / 1
IntroduccionType.php		100.00%	9 / 9		100.00%	3 / 3		100.00%	1 / 1
PortadaType.php		100.00%	9 / 9		100.00%	3 / 3		100.00%	1 / 1
ReglasDelNegocioType.php		100.00%	7 / 7		100.00%	3 / 3		100.00%	1 / 1
TrabajadoresDelNegocioType.php		100.00%	7 / 7		100.00%	3 / 3		100.00%	1 / 1

Legend

Low: 0% to 50% **Medium**: 50% to 90% **High**: 90% to 100%

Generated by php-code-coverage 5.2.1 using PHP 7.0.18-0ubuntu0.16.04.1 with Xdebug 2.4.0 and PHPUnit 6.1.3 at Thu Jun 8 16:07:16 CDT 2017.

3.10. Validación de la investigación

Como estrategia de la validación de la solución propuesta se ha realizado un Cuasiexperimento para realizar un estudio correlacional. La asignación de participantes a los grupos no se hace en forma aleatoria, ni por emparejamiento. Los grupos están previamente confeccionados (Proyectos productivos de la Universidad de Ciencias Informáticas CESIM y CESOL). El chequeo explícito de la equivalencia inicial de los grupos es imprescindible para medir la validez interna del experimento.

Se llevó a cabo un Cuasiexperimento de tipo 2; Pre y Post prueba con grupo de control. Se definieron como Grupo 1 y Grupo 2 a los proyectos mencionados anteriormente y las observaciones iniciales permiten evidenciar su equivalencia inicial. El instrumento utilizado es la entrevista y las observaciones son los resultados de las mismas.

Se mide la variable dependiente estandarizar, referida a la acción de ajustar a los analistas y especialistas en general, que crean, revisan y definen los productos de trabajo que se generan durante el proceso de desarrollo de software. Para la realización del cuasiexperimento se definen diseños para pre-prueba. El diseño de pre-prueba permite obtener una referencia inicial y conocer el nivel de la variable dependiente antes de ser estimulada. El diseño pos-prueba permite establecer una comparación con los datos obtenidos anteriormente y el comportamiento de la variable dependiente luego de recibir el estímulo (Guido Elmer, 2014).

Para la aplicación del diseño pre-prueba se utilizó los resultados arrojados en la primera entrevista realizada a los especialistas de diferentes centros de producción de la Universidad de las Ciencias Informáticas, teniendo en cuenta los siguientes resultados arrojados según la siguiente tabla de indicadores, para la estandarización.

Tabla 6 Resultado de la pre-prueba, elaboración propia.

Indicadores	Proyectos	Cantidad de especialistas	Resultados
Fuentes estilos y tamaño de letras.	Proyecto CESIM. Centro de Informática Médica	1 analista	Deben ajustar las fuentes, estilos y tamaños de letra.
	Proyecto Nova Servidores. Centro de Soluciones Libres.	1 jefe de proyecto	
Formato de las imágenes	Proyecto CESIM. Centro de Informática Médica	1 analista	Deben ajustar en el documento el tamaño y posicionamiento de las imágenes.
	Proyecto Nova Servidores. Centro de Soluciones Libres.	1 jefe de proyecto	
Estructura del documento	Proyecto CESIM. Centro de Informática Médica	1 analista	Deben insertar las filas y columnas de las tablas, siendo engorroso el trabajo. Deben corregir los márgenes de la plantilla.
	Proyecto Nova Servidores. Centro de Soluciones Libres.	1 jefe de proyecto	

Para la aplicación del diseño pos-prueba se utilizaron los resultados arrojados en la primera entrevista realizada a los especialistas de diferentes centros de producción de la Universidad de las Ciencias Informáticas, estableciendo comparaciones y evaluando los mismos indicadores de la pre-prueba. En este caso, con los resultados de la herramienta para la generación de los productos de trabajo de la metodología AUP-UCI.

Tabla 7 Resultado de la pos-prueba, elaboración propia.

Indicadores	Herramienta	Resultados
Fuentes estilos y tamaño de letras.	Generación de los productos de trabajo de la disciplina de modelado de negocio de la metodología AUP-UCI.	Se ajustan las fuentes, estilos y tamaños de letra en los documentos.
Formato de las imágenes		Se ajustan las imágenes en los documentos a un mismo tamaño y posicionamiento.
Estructura del documento		Se pueden insertar las filas y columnas de las tablas automáticamente. Se ajustan los márgenes de la plantilla.

Conclusiones del capítulo

Para darle cumplimiento a los objetivos trazados en este capítulo se realizó el modelo de implementación y como parte de él se obtuvieron el Diagrama de Componentes y de Despliegue de la solución y se documentaron los estándares de codificación por los que se guió la implementación del componente. Se realizó el Modelo de Datos para la disciplina Caso de Uso del Negocio lo cual permitió tener constancia de una forma más clara de la estructura, tipo de datos que incluye la base y la forma en que se relacionan, las restricciones de integridad y las operaciones de manipulación de los datos.

Conclusiones generales

- Se definen los conceptos principales de la investigación asociado al marco teórico de la investigación. En la cual, se identifica la metodología variación AUP-UCI como guía de todo el proceso de desarrollo de software. Se describe la disciplina de modelado de negocio para comprender los procesos de negocio de una organización y los productos de trabajos que se ejecutan en la disciplina. Se describe la estructura de los productos de trabajos como basamento para su estandarización.

- Se define la metodología AUP-UCI como guía del proceso de desarrollo de software para la construcción de las herramientas. Se propone un estudio de las metodologías de desarrollo de software, y en mayor profundidad la metodología AUP y AUP-UCI, las disciplinas que componen las mismas, y de forma más específica los productos de trabajos de la disciplina Modelado de Negocio.

- Se propone el modelo conceptual que permitió representar los conceptos de la disciplina de modelado de negocio y los productos de trabajos que se generan en la misma. Se definen los requisitos funcionales y no funcionales, que se dan cumplimento al objetivo general de la investigación. Se propone los diagramas de clases del diseño de los productos de trabajos: Caso de uso, Descripción de procesos de negocio y Reglas del negocio que permitió definir las clases de la implementación y las relaciones entre las clases, teniendo en cuenta la arquitectura que propone Symfony 2.

- Se describe la arquitectura de la aplicación, utilizando la propuesta en Symfony 2, que permitió construir el esqueleto de la herramienta, quedando estructurado mediante el patrón arquitectónico Modelo-Vista-Controlador, el código fuente de la aplicación.

- Se desarrolló la herramienta para generar productos de trabajos de la variante de la metodología AUP-UCI, que permitió estandarizar los productos de trabajos de la disciplina de Modelado de negocio.

- Se validó la solución propuesta mediante la automatización de pruebas unitarias de códigos PHP, que permitió verificar que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve y que si el estado inicial es válido entonces el estado final es válido, de las clases definida en la implementación.

Recomendaciones

Se recomienda que se pueda dar continuidad a la investigación, generando los productos de trabajos de las restantes disciplinas de la metodología AUP-UCI, que producto al alcance de la tesis, solamente se generaron los productos de trabajos de la disciplina Modelado de negocio.

Que se pueda integrar la herramienta para generar productos de trabajos de la metodología variación AUP-UCI, con las herramientas CASE, de manera que se pueda obtener de la misma los diagramas que se generan por cada una de las disciplinas de la metodología.

Bibliografía

A Calás, Torres. 2012. Definición de pautas para la creación de un modelo de componentes para el marco de trabajo Sauxe. La Habana : Serie Científica, 2012, Vol. V.

—. 2014. Documentación de Boson. [Online] 2014. http://docs.prod.uci.cu/online/Boson.docset/Contents/Resources/Documents/docs/index.html..

—. 2014. Modelo de componentes para el marco de trabajo Sauxe. 2014.

Alfonso, Damián Pérez. 2008. *Normas y estándares de codificación. Universidad de las Ciencias Informáticas. Normas y estándares de Codificación del ERP.* 2008.

Alonso, Evelyn Menéndez. 2017. Monografías.com. *Herramientas CASE para el proceso de desarrollo de Software.* [Online] 2017. http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml.

AngularJS. 2015. AngularJS. [Online] 2015. http://docs.angularjs.org..

Architect, Herramienta CASE - Enterprise. 2012. Herramienta CASE - Enterprise Architect. [Online] agosto 17, 2012. https://es.slideshare.net/iFhernd/herramienta-case-enterprise-architect.

Architect, Introducción a Herramientas CASE y System. Metodología y Tecnología de la Programación. *Introducción a Herramientas CASE y System Architect.* [Online] http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

Brito Acuña, Karenny. eumed.net. [Online] http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm..

Designer, Oracle. 2017. Oracle Tecnology Network. *Oracle Designer 10g Release 2.* [Online] 2017. http://www.oracle.com/technetwork/developer-tools/designer/overview/index-082236.html.

Doctrine. 2016. Doctrine, Web. [Online] 2016. http://www.doctrine-project.org/projects/orm/1.2/docs/manual/introduction/en.

Equiluz, J. 2013. Desarrollo Web Ágil con Symfony 2. Madrid : s.n., 2013.

—. 2014. Manual de Symfony 2. *Estándares de codificacion.* [Online] 2014. http://symfony.es..

—. 2015. Synfony 2. [Online] 2015. http://symfony.es..

Excriba. Excriba Gestor de Documentos Administrativos. *Centro CEIGE.* [Online] https://excriba.prod.uci.cu/page/site/centro-ceige/document-details?nodeRef=workspace://SpacesStore/dd42effe-f51a-4118-ad8d-6569befe3fe3.

ExtJS, Comunidad. 2016. ExtJS Nuestra comunidad en español. Qué es ExtJS? [Online] 2016. http://www.extjs.mx/2011/11/post-con-cursos-generales/..

Guido Elmer, Jacinto. 2014. *Diseños experimentales de investigación: pre-experimentos y cuasi-experimentos.* 2014.

Inc, S. 2015. Inc, S. [Online] 2015. https://www.sencha.com..

José Manuel González Peña, Coral Almansa Domínguez, Valerio Jurado Sánchez, Estefanía Moreno Pamos. 2017. Herramienta CASE. *Easy CASE.* [Online] 2017. https://docs.google.com/document/d/10NVjs72lOUusbWd5TNIpEIEkiQvaVZZTsLaAow-Vh8g/edit.

librosWeb. 2006-2017. LibrosWeb. [Online] 2006-2017. http://librosweb.es/libro/javascript/capitulo_1.html.

López, Patricia. Herramienta CASE Visual Paradigm. *INGENIERÍA DEL SOFTWARE I.* [Online] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/practicas-1/is1-p01-trans.pdf.

Mara, Ruvalcaba. 2017. Procesos de Software. [Online] 2017. https://sg.com.mx/revista/1/procesos-software..

Martínez Iglesias, E. 2007. *Adaptación de la metodología RUP a los nuevos módulos del Proyecto Registros y Notarías.* La Habana, Universidad de las Ciencias Informáticas : Informática, 2007.

McCall, J, Richards, P and Walters, G. 1977. Factors in Software Quality. *Métricas de Software.* [Online] 1977. http://www.desarrollow.o/rtls-copyleft/articulo-metricas-de-software.html.

Mena López, Grette Leydi y Tenrrero Cabrera, Marianela. 2008. *Arquitectura ERP.* 2008.

MySQL, Sitio web oficial. 2015. Sitio web oficial MySQL. [Online] 2015. http://www.mysql.com/.

Nobrega, Maria De. 2005. Curso de Sistemas de Información II. *Herramientas CASE: Rational Rose. .* [Online] junio 6, 2005. https://curso_sin2.blogia.com/2005/060401-herramientas-case-rational-rose.-por-maria-de-nobrega.php.

PhalconPHP. PhalconPHP. [Online] https://phalconphp.com/es/.

PHPnet. 2016. PHPnet. [Online] 2016. php.net/manual/en/intro-whatis.php..

Pressman, Roger S. 2010. *Software Engineering, a Practitioner's Approach. 7th Edition.* 2010.

Revista Científica Ingeniería y Desarrollo. **Rojas Morales, Carmenza, Alcocer Olaciregui, Adalgisa and Jabba Molinares, Daladier. 2014.** 2014.

Significado, Que. Que Significado. [Online] http://quesignificado.com/estandarizacion/.

Somerville, Ian. 2005. *Ingeniería de Software. 7ma.* 2005.

Symfony. 2016. SensioLabs. [Online] 2016. http://symfony.com/doc/current/contributing/code/standards.html.

Tamara, Rodríguez Sánchez. 2015. *Metodología de Desarrollo para la actividad productiva de la UCI.* 2015.

Tutorialspoint. 2014. Learn AngularJS, web application framework. [Online] 2014. http://www.tutorialspoint.com.

Visual Paradigm, Sitio web oficial del producto. 2014. Sitio web oficial del producto Visual Paradigm. [Online] 2014. http://www.visual-paradigm.com.

—. Sitio web oficial del producto Visual Paradigm. [Online] http://www.visual-paradigm.com/product/vpuml/.

WordReference.com. 2017. Online Language Dictionaries. [Online] 2017. http://www.wordreference.com/definicion/estandarizar.

Anexos

Anexo 1. Escenarios que aplican los proyectos que utilizan la metodología AUP-UCI.

A continuación se propone un análisis realizado sobre una selección de proyectos de la universidad que utilizan la metodología variación AUP-UCI y los escenarios que aplican.

Tabla 8 Escenarios que aplican los proyectos que utilizan la metodología AUP-UCI.

No.	Centro	Proyecto	Modelo de negocio	Variante de Requisitos
1	CESOL	Nova Escritorio 5.0	No	HU
2	CESOL	Nova Ligero 5.0	No	HU
3	CIDI	Portal Web de la Empresa DESEQUIP	MC	HU
4	CIDI	Intranet para IMECO	MC	HU
5	CIDI	Portal de Archivos para OAHCE	MC	HU
6	CISED	Desarrollo del sistema para la gestión y otorgamiento de licencias de software	No	HU
7	CISED	Sistema Multibiométrico	No	HU
8	TLM	Sistema de Gestión de Inventario de Hardware y Software para la UCI (XILEMA-GRHS-UCI 1.0)	DPN	ERP
9	CIGED	ABCD 3.0	DPN	CU
10	CIGED	Arkhaeia 3.0	DPN	CU
11	CIGED	Excriba 3.1	MC	CU
12	CEIGE	Marco de trabajo Bosón	No	HU
13	CEIGE	Sistema de Gestión de Importación y Exportación BK Import-Export	DPN	ERP
14	FORTES	Repositorio de Recursos Educativos (Xauce-Rhoda 3.0)	MC	ERP
15	FORTES	Herramienta de autor para la creación de recursos educativos CRODA	MC	ERP

16	CDAE	Replicador de Datos Reko v4.0	MC	HU
17	CEDIN	Sistema Integral de Perforación de Pozos de Petróleo (SIPP) Fase 2	MC	HU
18	DATEC	Sistema Automatizado de Gestión Integral, SAGI 1.0	MD	CU
19	DATEC	Servidor Dinámico de Reportes v1.0 (SDR v1.0)	MD	CU
20	GEYSED	Sistema para la automatización del proceso de producción de series y telenovelas	CUN	CU
21	GEYSED	Sistema de Gestión Audiovisual para el Centro de Información del Comité Central del PCC	CUN	CU
22	CESIM	Desarrollo a la medida de un sistema para el servicio de Hemodinámica del Cardiocentro CIMEQ	DPN	CU
23	CESIM	Desarrollo de la solución desktop para la conducción de ensayos clínicos del sistema A las Clínicas para el Centro de Inmunología Molecular	MC	CU
24	CESIM	Desarrollo del Sistema de Gestión de Ensayos Clínicos (XAVIA SIGEC 3.0.0)	MC	CU
25	CESIM	Desarrollo del Sistema de Telemedicina (XAVIA Telemedicina 1.0.0)	MC	CU

Anexo 2. Entrevistas realizadas a los especialistas de diferentes centros de producción de la UCI.

Se realizaron entrevistas a especialistas de centros que actualmente existen en la Universidad de Ciencias Informáticas que utilizan la Metodología AUP-UCI.

Entrevista No.1

Especialista: Laura Migdalia Sotolongo Pack

Analista del proyecto CESIM. Centro de Informática Médica. Departamento de Desarrollo de Componentes

Preguntas realizadas durante la entrevista

Pregunta 1 - ¿Cuáles son los productos de trabajos que se generan en el proyecto por cada una de las disciplinas?

En el Centro CESIM, el Modelado del Negocio se realiza mediante la variante del Escenario 3, Descripción de Proceso de Negocio (DPN) y Modelo Conceptual (MC). No se realizan Caso de Uso del Negocio (CUN). En el flujo de trabajo correspondiente a Requisitos se encapsulan por Casos de Usos del Sistema (CUS) y Descripción de Requisitos por Proceso (DRP). En el flujo de trabajo Análisis y Diseño se genera como producto de trabajo el Modelo del Diseño. Por último, en la disciplina de Implementación, el Modelo de Componentes.

Pregunta 2 - ¿Qué tiempo se demora en generar cada uno de los productos de trabajos por las Disciplinas y cuáles de ellos requieren mayor tiempo para su elaboración?

El tiempo que se demoran en generar estos productos de trabajos es variable según el proyecto, se debe tener en cuenta la complejidad y tamaño de la aplicación informática que se construye, pero en general el producto de trabajo que más tiempo requiere su elaboración es la Descripción de requisitos por proceso.

Pregunta 3 - ¿Qué parte del proceso se hace más engorroso?

Específicamente en la definición y las pautas a seguir para la generación de los productos de trabajos y en general establecer un formato estándar.

Pregunta 4 - ¿Qué beneficios tiene desarrollar una aplicación informática capaz de generar la documentación de los productos de trabajos para la metodología AUP-UCI?

Con la aplicación se lograría generar plantillas con dicho formato previamente definido. Ayudaría a mejorar en gran medida la uniformidad y organización de la documentación generada y con esto optimizar el tiempo que se tardan los artefactos en ser generados por los analistas del Centro.

Entrevista No.2

Especialista: Ivaniet Díaz Romeu

Jefe de Proyecto Nova Servidores. Centro de Soluciones Libres. Departamento de Sistema Operativo y Desarrollo de Tecnologías Libres.

Pregunta 1 - ¿Cuáles son los productos de trabajos que se generan en el proyecto por cada una de las Disciplinas?

En el Centro CESOL, en el Modelado de Negocio, se generan los siguientes productos de trabajos: Descripción de Procesos del Negocio (DPN), Mapa de Procesos, Modelo del Negocio con Casos de Uso y Reglas del Negocio. La disciplina Requisitos se divide en dos, Requisitos y Administración. En requisitos

se generan Descripción de Requisitos por Proceso (DRP), Especificación de Casos de Uso, Especificación de Requisitos del Software, Historias de Usuario (HU) y Modelo Conceptual (MC). Y en Administración, Criterios para Validar Requisitos del Cliente, Criterios para Validar requisitos del Producto, Evaluación de Requisitos, Reportes de Trazabilidad. De acuerdo a los elementos del proyecto de Matrices de Trazabilidad puede haber: Matriz de EntidadesBD con Conceptos, Matriz de EntidadesBD con Modelo de Clases del Diseño, matriz de Requisitos con Artefactos, Matriz de Requisitos con Conceptos, Matriz de Requisitos con Caso de Uso del Negocio, Matriz de Requisitos con Caso de Uso del Sistema, Matriz de Requisitos con Diagrama de Clases del Proceso, Matriz de Requisitos con Diagrama de Clases del Diseño, Matriz de Requisitos con Proceso de Negocio, Matriz de Requisito con Requisito y Matriz de Requisito con Producto de Trabajo.

Los productos de trabajos que se generan son Historia de Usuario (HU), Especificación de requisitos, Diagrama de Caso de Prueba y Criterios para validar requisitos Producto y Cliente. Si este Criterio se somete a cambios se genera un Reporte de Trazabilidad, correspondiente a Administración. Y si se rechazan los requisitos se generan Requisitos Rechazados.

En Análisis y Diseño, el Diagrama de Caso de Prueba se usa para las pruebas. Se generan también las Vistas Arquitectónicas que se dividen en Arquitectura de integración y Arquitectura de Seguridad. También las Vistas de Entorno de Desarrollo Tecnológico y la Arquitectura de Vistas Infraestructura. Y por último la Definición de arquitectura del Software.

Pregunta 2 -¿Qué tiempo se demora en generar cada uno de los productos de trabajos por las Disciplinas y cuáles de ellos requieren mayor tiempo para su elaboración?

El tiempo que se demoran en generar estos artefactos es variable de un proyecto a otro teniendo en cuenta su complejidad y tamaño del proyecto. Por ejemplo, un artefacto de complejidad baja de un proyecto pequeño puede tardarse alrededor hora para ser generado. Esto depende de forma proporcional a lo planteado anteriormente y a la capacidad de los analistas en general. En uno de los proyectos que se están desarrollando actualmente en nuestro centro para generar 340 requisitos con cuatro analistas se tardó cuatro meses, y ninguno era de complejidad alta.

Pregunta 3 - ¿Qué parte del proceso se hace más engorroso?

En general lo que más tiempo necesita para su elaboración en el centro es definir los estándares y velar por que se cumplan los mismos.

Pregunta 4 - ¿Qué beneficios tiene desarrollar una aplicación informática capaz de generar la documentación de los productos de trabajo para la metodología AUP-UCI?

Si se generaran plantillas automatizadas de estos procesos contribuiría de forma positiva a la organización de la documentación generada y con esto optimizar el tiempo que se tardan los artefactos en ser generados por los analistas, así como a lograr que se cumpla con un formato estándar y el control pudiera ser más fácil de llevar a cabo.

Anexo 3. Diagramas de clases del diseño

Diagrama de clases del diseño del producto de trabajo Reglas de negocio

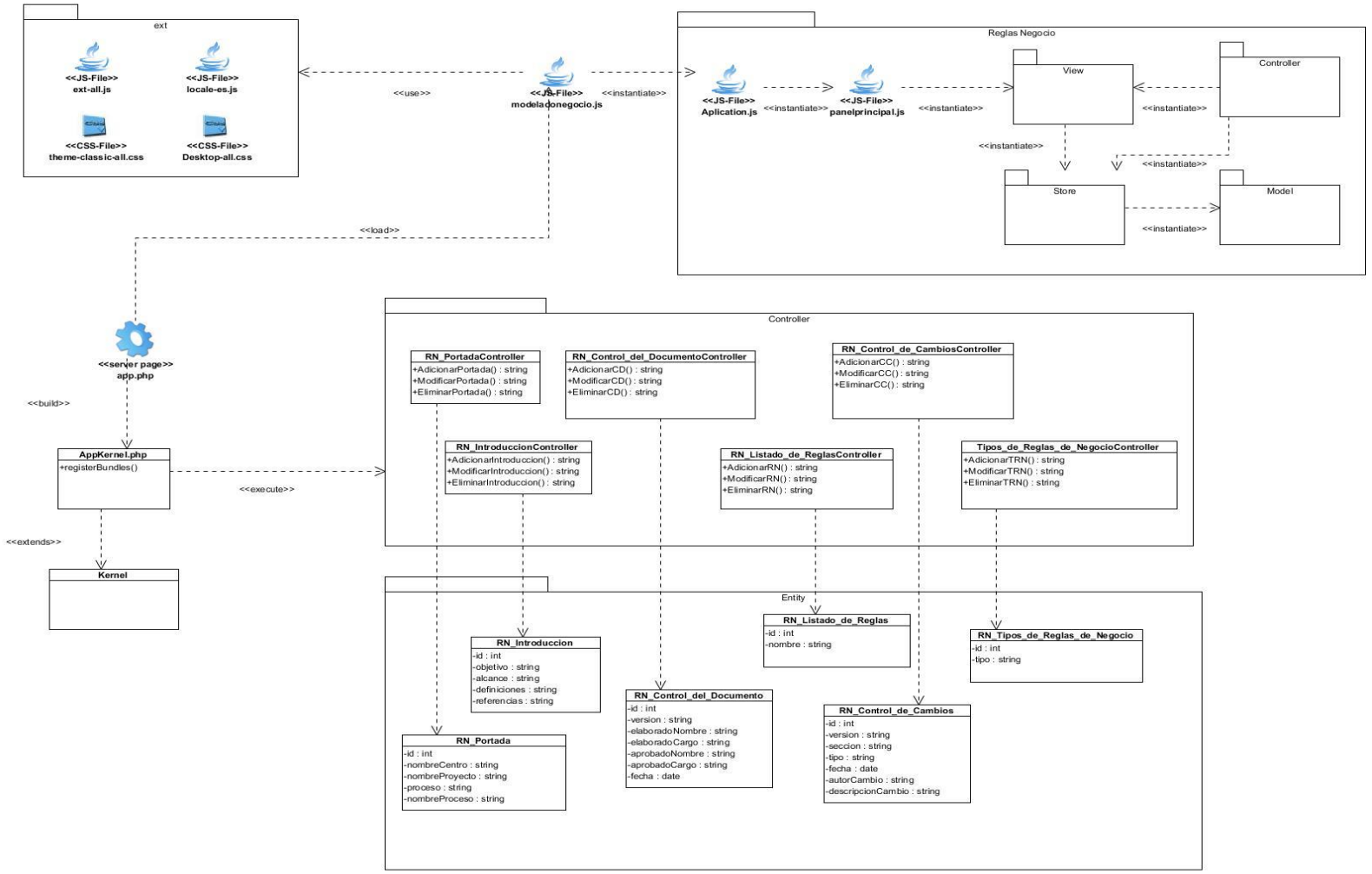


Figura 18 Diagrama de clases del diseño del producto de trabajo Reglas de negocio.

Diagrama de clases del diseño del producto de trabajo Descripción de procesos de negocio

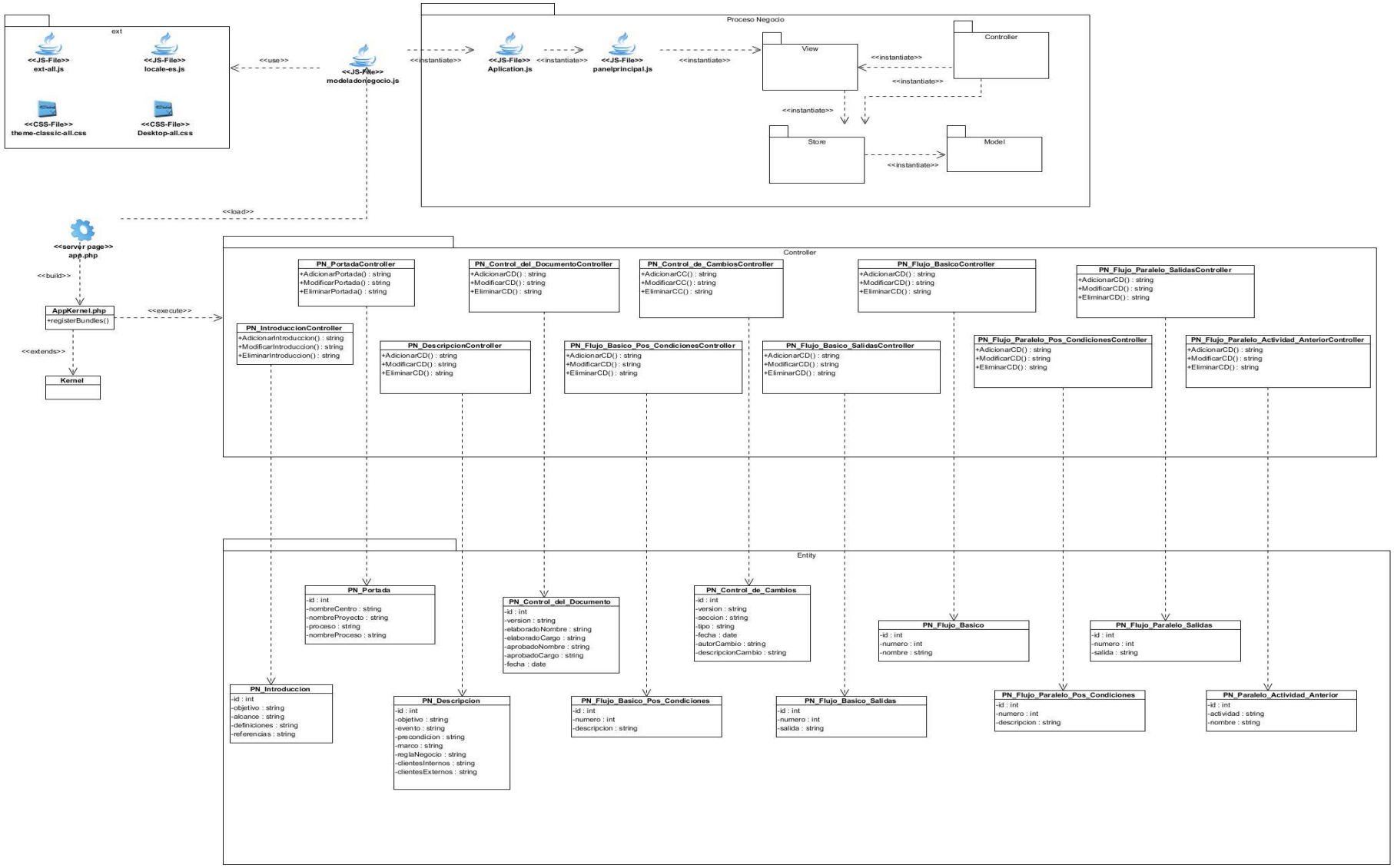


Figura 19 Diagrama de clases del diseño del producto de trabajo Descripción de procesos de negocio.

.

Anexo 4. Diagramas Entidad Relación

Diagrama Entidad Relación del producto de trabajo Reglas de negocio

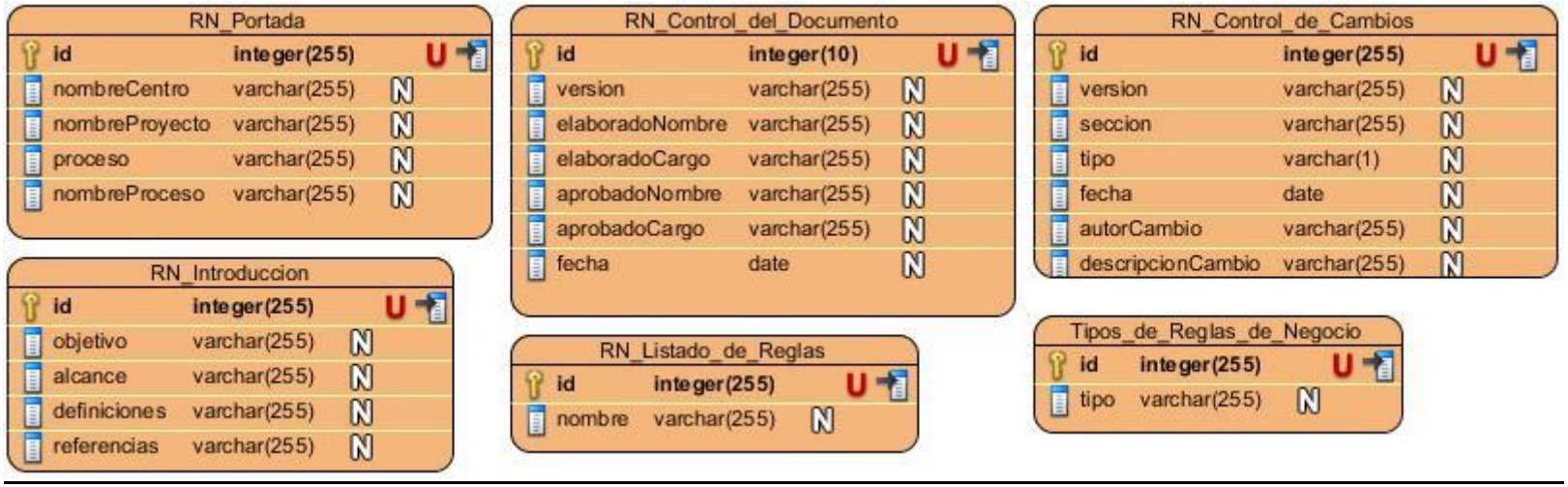


Figura 20 Diagrama Entidad Relación del producto de trabajo Reglas de negocio.

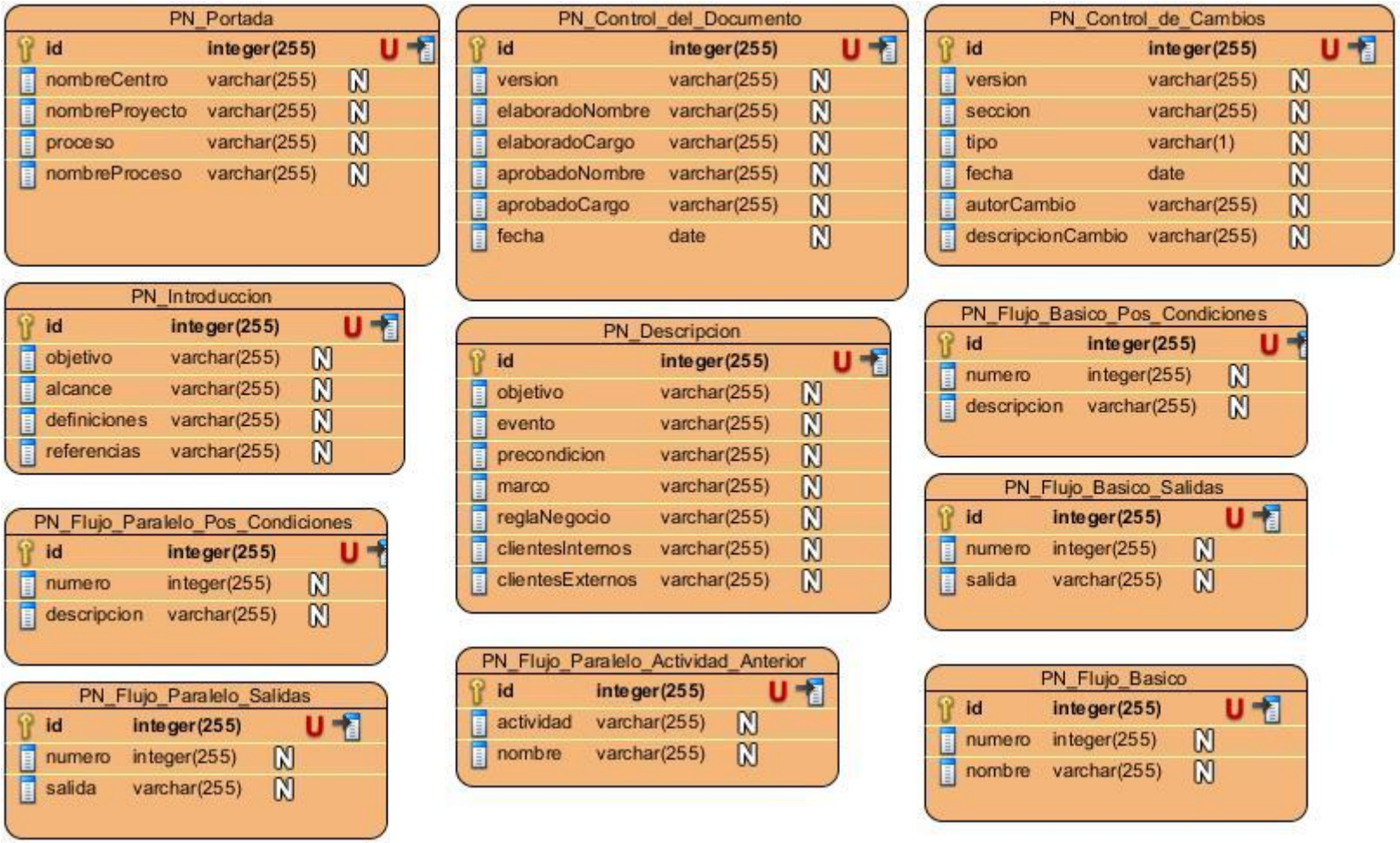


Figura 21 Diagrama Entidad Relación del producto de trabajo Descripción de procesos de negocio.