



Universidad de las Ciencias
Informáticas

Facultad 5

Centro de Consultoría y Desarrollo de Arquitecturas Empresariales

**Componente para la visualización de trazas del SOICIM
utilizando técnicas de minería de procesos.**

**Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas.**

Autor: Ever Leandro Santiesteban Jiménez

Tutor: Ing. Yidier Romero Zaldivar

Co-Tutora: Ing. Susana Alba Silot

“Año 58 de la Revolución” La Habana, Cuba, julio 2016

Declaración de autoría

Por este medio declaro ser autor de este trabajo y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

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

Ever Leandro Santiesteban Jiménez.

Firma del Tutor

Firma de la Co-Tutora

Firma del Autor

Datos de contacto

Dedicatoria

Este trabajo va dedicado a mi familia por su cariño y ayuda durante mis 24 largos años de estudio, a cada uno de los amigos que de alguna manera contribuyeron a que llegara la fecha de mi graduación y muy especialmente a mi hermano a mi mamá y a mi papá.

Agradecimientos

Es mi más profundo deseo agradecer a cada una de las personas que de alguna manera me apoyaron sin tener que pronunciar una palabra. Es muy bonito llegar a la meta y decir: "Lo conseguí", pero siempre es importante recordar a aquellas personas que hicieron posible que lo lograra. Más de una vez he escuchado: "Lo conseguí solo", pero no es mi caso. Quiero aprovechar esta oportunidad única para hacer pública mi gratitud infinita hacia cada uno de los amigos que hice a lo largo de la universidad. Hoy siento orgullo de decir que no tengo un millón de amigos, pero los que tengo valen por un millón. Con mi familia las palabras sobran, porque mi gratitud hacia ella es absoluta. Gracias tías, primos, hermano, mami y papi por lograr hacer de mí la persona que, de niño, soñaba ser.

Resumen

Resumen

Las empresas de hoy presentan la necesidad de hacer más eficientes sus operaciones y mantenerse competitivas en un mundo cada vez más interconectado y globalizado. Estas necesidades pueden erradicarse haciendo uso de diferentes herramientas que le proporcionen agilidad a la organización, destacándose el uso de portales web (dirigidos a ofrecer al usuario, de forma fácil e integrada, además del acceso a una serie de servicios relacionados a un mismo tema: enlaces, buscadores, foros, documentos, aplicaciones, compra electrónica).

Liferay Portal es una de estas plataformas web, que tiene como objetivo ofrecer el máximo beneficio para sus clientes, cuando es utilizado como intranet de la organización, como portal de cara a los clientes u otra variante de aplicación compuesta. Es un producto que presenta más de 60 aplicaciones portlets, listas para desarrollar tareas básicas de un portal moderno a nivel de empresa.

En la presente investigación se argumenta el desarrollo de un portlet que permitirá a las entidades responsables la monitorización del proceso: Gestión de órdenes comerciales en el Centro de Inmunología Molecular (CIM). El ciclo de desarrollo de la solución estuvo regido por la Metodología Unificada Ágil (AUP), adaptada a los procesos productivos de la Universidad de las Ciencias Informáticas (UCI). Las tecnologías y herramientas empleadas para la aplicación son libres (Open Source), con excepción de Visual Paradigm for UML.

Palabras claves: *Portlet, Redes de Petri, Minería de procesos, ProM.*

Índice de contenido

Índice de contenido

Introducción	1
Capítulo 1. Fundamentación teórica.....	6
1.1 Introducción	6
1.2 Aplicaciones de gestión empresarial	6
1.3 Proceso de negocio:.....	7
1.4 Red de Petri:.....	7
1.5 Minería de procesos:.....	8
1.6 Minería de procesos en la gestión de procesos de negocio:.....	8
1.7 Registro de eventos:	9
1.8 Descubrimiento de procesos:	10
1.9 Desafíos en Minería de Procesos:	10
1.10 Algoritmos de descubrimiento de proceso.....	11
1.10.1 Algoritmo Alpha Miner AM	11
1.10.2 Algoritmo Heuristic Miner HM.....	12
1.10.3 Algoritmo Integer Linear Programming ILP	13
1.11 Herramientas para la minería de procesos: ProM.....	13
1.12 Tecnologías y el proceso de desarrollo de software utilizado	13
1.13 Lenguaje de programación	13
1.13.1 Java	13
1.13.2 JavaScript.....	14
1.13.3 Hojas de estilo en cascada (CSS).....	14
1.14 Herramientas de Modelado.....	15
1.14.1 Visual Paradigm for UML	15
1.15 Entorno de desarrollo integrado. Eclipse Luna	15
1.16 Portal empresarial	16
1.17 Plataforma web. Liferay Portal	17
1.18 Portlets	17
1.19 Metodología de desarrollo.....	18
1.20 Metodología Unificada Ágil (AUP):	20
1.21 Conclusiones parciales.....	22
Capítulo 2. Características y diseño del sistema.....	24
2.1 Introducción	24

Índice de contenido

2.2 Propuesta de solución.....	24
2.2.1 Interpretación de una red de Petri	25
2.3 Especificación de requisitos	26
2.3.1 Requisitos funcionales (RF).....	26
2.3.2 Requisitos no funcionales (RNF)	27
2.4 Historias de Usuario	28
2.4.1 Descripción de las Historias de Usuario	29
2.5 Plan de Iteraciones	36
2.6 Estilo arquitectónico:	37
2.6.1 Arquitectura de la aplicación:.....	37
2.8 Patrones de diseño:.....	38
2.8.1 Patrones GRASP.....	38
2.8.2 Patrones GOF	39
2.9 Modelo de diseño:	40
2.9.1 Algoritmo visualizador	40
2.9.2 Descripción de las clases del diseño:	41
2.10 Conclusiones parciales.....	43
Capítulo 3. Implementación y prueba del sistema	44
3.1 Introducción	44
3.2 Estándar de código.....	44
3.3 Pruebas:	46
3.3.1 Pruebas de caja negra.....	46
3.3.2 Estrategia de pruebas.....	46
3.3.2 Técnica de la Partición de Equivalencia:	49
3.3.3 Casos de pruebas	49
3.5 Validaciones	51
3.5.3 Pruebas internas.....	52
3.6 Lista de chequeo:.....	56
3.7 Resultados	57
3.8 Conclusiones parciales	57
Conclusiones generales:	58
Bibliografía.....	60

Índice de Ilustraciones:

Ilustración 1 Red de Petri (7).	8
Ilustración 2. El ciclo de vida de BPM (9).	9
Ilustración 3 Fases e Iteraciones (28).	21
Ilustración 4: Modelo de Ciclo de vida de proyectos de minería de procesos (L*) (9)..	25
Ilustración 5: Clásica red de Petri.....	25
Ilustración 6: Arquitectura de la aplicación.	37
Ilustración 7:Diagrama de clases.	40
Ilustración 8: Algoritmo DibujarCapas.	41
Ilustración 9: red generada por el visor del portlet.....	52
Ilustración 10: red generada por el visor de la herramienta ProM.	52
Ilustración 11: Método "LlenarListaCapas()".	53
Ilustración 12:" Grafo y Caminos"	54
Ilustración 13:Regiones del grafo.....	55

Índice de tablas:

Tabla 1 Diferencias entre metodologías ágiles y no ágiles.....	20
Tabla 2: Requisitos funcionales (RF)	27
Tabla 3: Requisitos no Funcionales.	28
Tabla 4: Historia de Usuario HU1.....	30
Tabla 5: Historia de Usuario HU2.....	31
Tabla 6: Historia de Usuario HU3.....	32
Tabla 7: Historia de Usuario HU4.....	33
Tabla 8: Historia de Usuario HU5.....	34
Tabla 9: Historia de Usuario HU6.....	35
Tabla 10: Plan de iteraciones.....	37
Tabla 11: Métodos auxiliares del módulo Petri_Net.js.....	41
Tabla 12: Descripción de las clases del diseño.....	42
Tabla 13:Estrategia de pruebas	49
Tabla 14:Caso de prueba EC1:Cargar Registro.	50
Tabla 15: Caso de prueba EC2: Eliminar Registro.....	50
Tabla 16: Caso de prueba EC2: Filtrar algoritmo.	50
Tabla 17:Aplicar algoritmo de descubrimiento.....	51
Tabla 18:Aplicar algoritmo de descubrimiento.....	51
Tabla 19:Lista de chequeo.....	56

Introducción

Es inevitable el creciente desarrollo de la humanidad gracias a todos los avances ocurridos en las diferentes esferas de la ciencia y la tecnología; por ejemplo, los avances tecnológicos en ramas como la informática y las comunicaciones, que hoy en día juegan un papel de vital importancia para el crecimiento y el éxito de cualquier institución. La industria del software ha alcanzado una posición privilegiada, precisamente por la gran variedad de aplicaciones que ofrece y la gran cantidad de soluciones que aportan dichas aplicaciones a problemas del tipo empresarial, educativo, económico, de la salud, etc. La industria de las tecnologías de la información y las comunicaciones (TIC) está fuertemente sujeta al desarrollo económico de cada nación, por lo que la industria del software, como apéndice de la industria de las nuevas tecnologías, se encuentra bajo las mismas condiciones.

Inmerso dentro de las grandes ventajas que trajo para el mundo el uso de las TIC se encuentra la aparición de aplicaciones empresariales que automatizan los procesos de cada organización y generan un gran cúmulo de datos en formato digital. Dichos datos pueden utilizarse en análisis posteriores para detectar anomalías u oportunidades de mejora en los procesos automatizados por las aplicaciones. La minería de procesos, una disciplina de investigación relativamente joven, tiene su aplicación en procesos reales y proporciona ventajas tales como la identificación de cuellos de botella, anticipación de problemas, recomendación de contramedidas, la simplificación de procesos, con el objetivo de mejorar el funcionamiento del negocio. A través del modelado de procesos puede lograrse un mejor entendimiento del negocio, incluso detectar dificultades que serían invisibles para análisis de poca profundidad. En las instituciones actuales se hace necesaria la informatización de los procesos que se desarrollan para así poder mantener un mayor control sobre estos.

Como parte del desarrollo actual, las instituciones cubanas no están ajenas a la informatización global que está viviendo la sociedad contemporánea. El sector de la biotecnología es uno de los más favorecidos con el avance de la informática, teniendo como ejemplo de ello el Sistema de Operaciones Integrales del Centro de Inmunología Molecular (SOICIM),

Introducción

que fue desarrollado por el Centro de Consultoría y Desarrollo de Arquitecturas Empresariales (CDAE) para informatizar la gestión de las órdenes comerciales del CIM. Como resultado se obtuvo un sistema que maneja tanto información de los clientes actuales de la institución, como los pedidos de los clientes y la conversión de dichos pedidos a órdenes de producción a ejecutar por las plantas del CIM. Actualmente, SOICIM ha logrado automatizar, en buena medida, la gestión de información referente a los clientes del CIM, a sus productos comercializables, a las solicitudes de producción de cada cliente y a las órdenes de producción derivadas de dichas solicitudes. SOICIM almacena gran cantidad de información digitalizada referente a las operaciones comerciales del CIM, la cual podría ser útil a los directivos de la institución en el análisis del comportamiento del negocio y para tomar una decisión eficaz posteriormente. Dentro de los posibles análisis está el monitoreo de los procesos a través de las técnicas de minería de procesos. El sistema mencionado almacena las trazas de las actividades del negocio, datos potenciales para aplicárseles las técnicas de minería de procesos. Se realiza un monitoreo de variables aisladas, pero pudiera monitorizarse el proceso y de hacerse correctamente, ayudaría en la identificación de desvíos y congestiones del mismo.

Debido a lo antes expuesto queda formulado el **problema de investigación** de la siguiente forma:

¿Cómo facilitar la detección de anomalías en el proceso de gestión de órdenes comerciales del CIM?

Objeto de estudio:

La monitorización automatizada de procesos.

Se define como **objetivo general**:

Desarrollar un componente que monitorice el proceso de órdenes comerciales en el CIM basado en técnicas de minería de procesos para apoyar la toma de decisiones por la alta gerencia.

Introducción

Campo de acción:

La monitorización automatizada de procesos utilizando técnicas de minería de procesos

Para dar cumplimiento al objetivo propuesto se presentan las siguientes **tareas de investigación:**

- Elaboración del marco conceptual para precisar los principales conceptos que se emplean en los componentes para el monitoreo de procesos.
- Selección de la metodología para definir los métodos y técnicas necesarias que guiarán el desarrollo.
- Descripción de las herramientas, tecnologías y lenguajes a emplear para definir el ambiente de desarrollo.

Métodos de investigación

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. Los métodos científicos empleados se pueden clasificar en teóricos o empíricos, los cuales están dialécticamente relacionados. Son utilizados por el investigador de acuerdo a las circunstancias y el escenario donde se desarrolla la investigación. (1)

Métodos teóricos

Dentro de las principales características que podemos mencionar de estos métodos se encuentra, que permiten estudiar las características del objeto, que no son observables directamente, su uso facilita la construcción de modelos e hipótesis de investigación y crean las condiciones para ir más allá de las características fenomenológicas y superficiales de la realidad, además de posibilitar el conocimiento del estado del arte del fenómeno, su evolución en una etapa determinada y su relación con otros fenómenos. Es muy importante señalar que se apoyan en el proceso de análisis-síntesis. (1)

A continuación, se describen los métodos teóricos tenidos en cuenta durante el proceso investigativo:

Introducción

El **Análisis-Síntesis** para formular conclusiones valiéndose del análisis del comportamiento de los resultados obtenidos al aplicarle a los registros de eventos los distintos algoritmos de descubrimiento que posee la herramienta ProM.

La Modelación es considerada como uno de los métodos lógicos que consistirá en esquemas, diagramas o representaciones. Se utiliza en la confección de todos los diagramas y representaciones de la propuesta de solución. Además, se utiliza en la representación de los procesos en forma de redes de Petri partiendo del registro de eventos y utilizando la herramienta ProM.

Métodos empíricos

Estos métodos describen y explican las características fenomenológicas del objeto. Representan un nivel de la investigación cuyo contenido procede de la experiencia y es sometido a cierta elaboración racional. (1)

A continuación se menciona el método empírico utilizado durante el proceso investigativo:

El método de **Consulta de la Información** es utilizado para elaborar el marco teórico o el estado del arte de la investigación, permitiendo la apropiación del conocimiento y el asentamiento de las referencias bibliográficas de los múltiples criterios que han sido citados. Se utiliza para comprender mejor el problema de investigación planteado y decidir si es posible tanto científicamente como económicamente su ejecución.

Idea a defender

Con el desarrollo de un portlet que utilice técnicas de minería de procesos para el descubrimiento de procesos, a partir de las trazas del sistema SOICIM, se apoyará la monitorización automatizada del proceso de la gestión de órdenes comerciales del CIM, logrando que sea más fácil el control estructural del mismo.

El presente trabajo, está estructurado en tres capítulos, distribuidos de la siguiente forma:

Introducción

Capítulo 1. Fundamentación teórica: Se describe el marco teórico de la investigación, así como las tecnologías y herramientas actuales enmarcadas en el proceso, y la metodología de software seleccionada para el cumplimiento del proyecto.

Capítulo 2. Características del sistema: se identifican y describen los conceptos asociados al dominio del problema y los procesos relacionados con el negocio. Se definen cuáles son los requerimientos funcionales, no funcionales y el modelo de datos.

Capítulo 3. Implementación y prueba: en este capítulo se describen los estándares de códigos usados durante el desarrollo de la solución. Además, se diseña una serie de casos de pruebas con el propósito de validar la solución propuesta, la evaluación de su ejecución y los resultados obtenidos.

Capítulo 1. Fundamentación teórica

Capítulo 1. Fundamentación teórica

1.1 Introducción

En el presente capítulo se pretende dar una visión global de los fundamentos teóricos, plasmando conceptos tales como proceso de negocio, minería de procesos y descubrimiento de procesos, que están estrechamente relacionados con el estudio de la información recopilada a lo largo de la investigación, con el fin de lograr un mejor entendimiento de los términos frecuentados en el documento. Por último, se confecciona un análisis de las diferentes tecnologías, lenguajes y herramientas utilizadas para el desarrollo de la aplicación.

1.2 Aplicaciones de gestión empresarial

La gestión empresarial conocida también como administración de empresas o ciencia administrativa es una ciencia social que estudia la organización de las empresas y la manera como se gestionan los recursos, procesos y resultados de sus actividades. Son ciencias administrativas o ciencias económicas y financieras, la contabilidad, las finanzas corporativas y la mercadotecnia, la administración, la dirección estratégica, etc (2). Entre los tipos de aplicaciones de gestión empresarial se encuentran los ERP (Planificación de Recursos Empresariales), los CRM (Gestión de Clientes), RMS (Sistemas de Gestión de Ventas), RMA (Sistemas de Gestión de Materiales), SCMS (Sistema de Gestión de la Cadena de Suministros), por nombrar algunos. En el caso de los ERP, disponen de un conjunto de módulos que ofrecen distintas funcionalidades según el área de la empresa: ejemplo, poseen un módulo de Gestión de Inventarios útil en almacenes, poseen un módulo de contabilidad y finanzas, poseen módulos para la Gestión de Recursos Humanos útil en la gestión del personal de la empresa, y podrán tener más módulos en dependencia del software. Los CRM proporcionan a la empresa la capacidad de atender personalmente a los clientes, con vista a retenerlos y fortalecer las relaciones comerciales. Los RMA proporcionan a la empresa la capacidad de gestionar los ingresos de todos los productos que se ofertan y concentra información importante para análisis posteriores (3).

Capítulo 1. Fundamentación teórica

1.3 Proceso de negocio:

Se define como un proceso de negocio a un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio específico. Cada uno de los procesos de negocio tiene sus entradas, funciones y salidas. Las entradas son requisitos que deben tenerse antes de que una función pueda ser aplicada (4). Es una colección de actividades estructurales relacionadas que producen un valor para la organización, sus inversores o sus clientes. Es, por ejemplo, el proceso a través del que una organización ofrece sus servicios a sus clientes. Un proceso de negocio puede ser parte de un proceso mayor que lo abarque o bien puede incluir otros procesos de negocio que deban ser incluidos en su función. En este contexto un proceso de negocio puede ser visto a varios niveles de granularidad. El enlace entre procesos de negocio y generación de valor lleva a algunos practicantes a ver los procesos de negocio como los flujos de trabajo que efectúan las tareas de una organización (5).

1.4 Red de Petri:

Las redes de Petri son una de las técnicas más conocidas para la especificación de los procesos de negocio de una manera formal y abstracta, son una importante base para el lenguaje de proceso. Pueden ser utilizadas para modelar sistemas dinámicos con una estructura estática; dicha estructura estática está representado por una red y el comportamiento dinámico es capturado por el juego simbólico de la red de Petri. Están conformadas por lugares, transiciones y arcos dirigidos; son grafos bipartitos, de modo que nunca se conectan dos transiciones o dos lugares. En notaciones gráficas los lugares están representados por círculos y las transiciones están representados por rectángulos que están conectados por arcos dirigidos o flechas. Las redes de Petri describen la estructura de un sistema, representa un modelo de procesos de negocio, y sus transiciones representan modelos de actividad. Los niveles de instancia son capturados por fichas. Esto significa que el disparo de una transición representa una instancia de actividad. Cada instancia de proceso está representada por al menos una ficha. Puesto que puede haber varias instancias de proceso de un modelo de proceso, las fichas en una red de Petri pueden pertenecer a diferentes instancias de proceso (6).

Capítulo 1. Fundamentación teórica

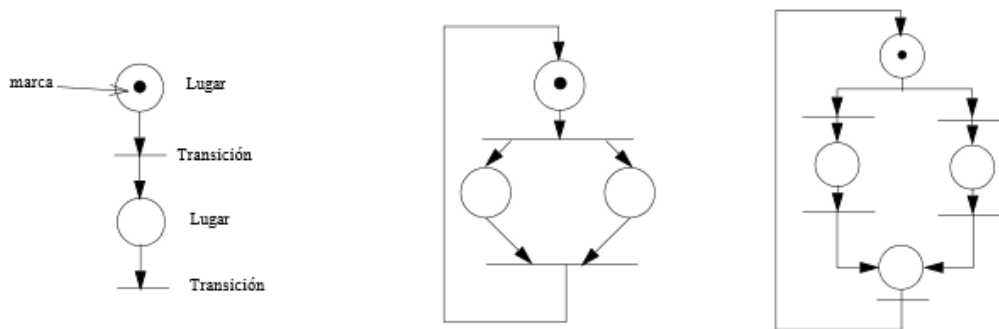


Ilustración 1: Red de Petri (7).

1.5 Minería de procesos:

La ejecución de los procesos de negocio genera un gran volumen de datos, de los que se puede extraer información que ayude a comprender fenómenos o a tomar decisiones aplicando técnicas y herramientas. La minería de procesos es una técnica de administración de los mismos que permite analizar los procesos de negocios de acuerdo con un registro de eventos. A través de esta actividad se desea extraer conocimiento desde los registros de evento de los procesos almacenados por los sistemas. Este conocimiento implica lograr realizar la traza de los procesos en estudio, incluyendo información de los actores que lo realizan, los tiempos involucrados, entre otras cosas. Uno de los objetivos es llevar el control de los procesos, pero además tiene como objetivo permitir el descubrimiento de controles, información y estructuras organizacionales partiendo de la base de los registros de eventos (8).

1.6 Minería de procesos en la gestión de procesos de negocio:

Como es conocido el ciclo de vida de la Gestión de Procesos de Negocio (BPM6) abarca las siete fases de un proceso de negocio y los sistemas de información asociados al mismo.

Capítulo 1. Fundamentación teórica

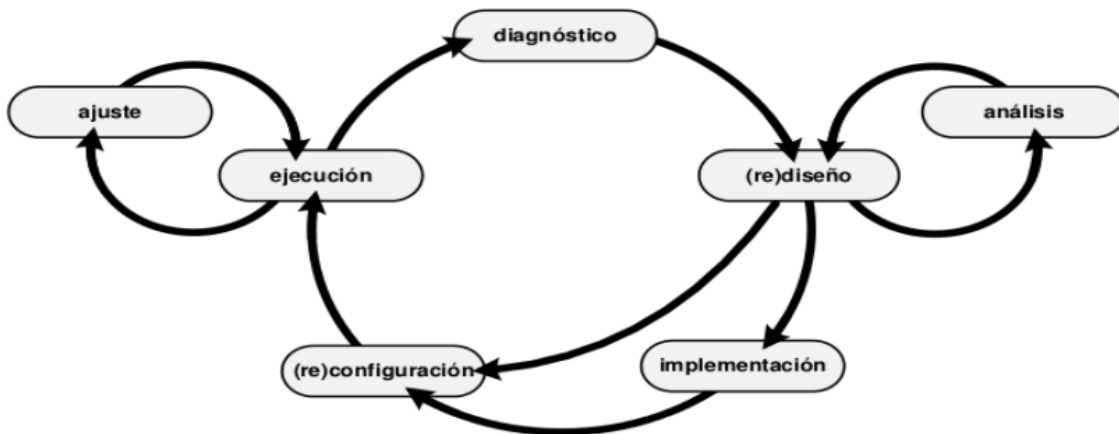


Ilustración 2. El ciclo de vida de BPM (9).

Primeramente, se diseña un proceso, el cual es convertido en un sistema ejecutable en la fase de implementación. En la fase de (re)diseño se crea un nuevo modelo de proceso o se adapta un modelo de proceso existente. En la fase de análisis se analiza un modelo candidato y sus alternativas. Después de la fase de (re)diseño, se implementa el modelo (fase de implementación) o se (re)configura un sistema existente (fase de (re)configuración). En la fase de ejecución se ejecuta el modelo diseñado. Durante la fase de ejecución el proceso es monitoreado. Además, se podrán realizar pequeños ajustes sin rediseñar el proceso (fase de ajuste). En la fase de diagnóstico se analiza el proceso ejecutado y la salida de esta fase podría iniciar una nueva fase de rediseño del proceso (9).

1.7 Registro de eventos:

Las instituciones del sector de la salud utilizan sistemas automatizados para gestionar la ejecución de sus procesos. Estos sistemas registran en forma de trazas las acciones realizadas al ejecutar las instancias o casos de los procesos de negocio. Las trazas de ejecución al transcurrir por un proceso de transformación son convertidas en registros de eventos. El punto de partida para aplicar técnicas de Minería de Procesos es un registro de eventos, asumiendo que es posible registrar eventos secuencialmente. Para dar formato a un log de Registro de eventos, se define el formato XES, este es un estándar basado en experiencias prácticas de MXML que su principal propósito es ofrecer un formato

Capítulo 1. Fundamentación teórica

de intercambio de Registros de eventos entre herramientas y dominios de aplicaciones (8).

1.8 Descubrimiento de procesos:

El descubrimiento de procesos abarca el conjunto de técnicas para la obtención de un modelo de procesos partiendo de un registro de eventos. Para muchas organizaciones resulta sorprendente la capacidad de los algoritmos de descubrimiento para identificar los procesos reales partiendo únicamente de los registros de eventos asociados a sus ejecuciones. Los modelos descubiertos pueden presentar la información de la ejecución del proceso desde diferentes perspectivas (9) .

1.9 Desafíos en Minería de Procesos:

Los desafíos de los que se quiere ocupar la Minería de Procesos se resumen en (Vander Aalst et al., 2011). (...) La minería de procesos es una herramienta importante para las organizaciones modernas que necesitan gestionar procesos operacionales no triviales. Por un lado, hay un increíble crecimiento de la cantidad de datos de eventos. Por otro lado, los procesos operacionales y la información necesitan estar perfectamente alineados para cumplir requerimientos relacionados con el cumplimiento de normas, eficiencia y servicio al cliente. A pesar de la aplicabilidad de la minería de procesos, aún hay desafíos importantes que necesitan ser abordados; estos ilustran que la minería de procesos es una disciplina emergente. A continuación, son nombrados algunos de estos desafíos. Se debe tener en cuenta que, con el paso del tiempo, podrían aparecer nuevos desafíos o podrían desaparecer desafíos existentes debido a los avances en minería de procesos (10):

- Encontrar, Fusionar y Limpiar Datos de Eventos: Todavía toma esfuerzos considerables extraer datos de eventos apropiados para la minería de procesos. Típicamente se necesita superar varios obstáculos (10).
- Lidar con Registros de Eventos Complejos con Diversas Características: Algunos registros de eventos pueden ser extremadamente grandes lo cual hace difícil manipularlos mientras otros registros de eventos son tan pequeños que no tienen suficientes datos para obtener conclusiones

Capítulo 1. Fundamentación teórica

confiables (10).

- Crear Puntos de Referencia Representativos: La falta de buenos puntos de referencia es explicable, esto pasa porque la minería de procesos es una disciplina de investigación relativamente joven (10).
- Balancear Criterios de Calidad tales como Ajuste, Simplicidad, Precisión y Generalización: Los registros de eventos están a menudo lejos de ser completos, es decir, sólo se cuenta con un comportamiento de ejemplo (10).
- Mejorar la Usabilidad para los No Expertos: Uno de los objetivos de la minería de procesos es crear “Modelos de procesos reales” que sean utilizados a diario, más que modelos estáticos que terminan en algún archivo (10).
- Mejorar el Entendimiento para los No Expertos: El usuario puede tener problemas para entender la salida, o es tentado a inferir conclusiones incorrectas (10).

1.10 Algoritmos de descubrimiento de proceso

Existen variados algoritmos para el descubrimiento de procesos entre los que se pueden mencionar: Alpha Miner, Heuristic Miner, Integer Linear Programming y Fuzzy Miner. Para la solución propuesta se tomaron en cuenta los tres primeros algoritmos de los mencionados, debido a que generan como salida una red de Petri en formato pnml. Este formato es el que hasta ahora es reconocido por el portlet desarrollado para graficar redes. A continuación, se describen algunas de las principales características de los algoritmos escogidos:

1.10.1 Algoritmo Alpha Miner AM

Este procedimiento consta de tres fases: la fase de abstracción, la fase de inducción y la fase de construcción. En la fase de abstracción, cada evento en cada traza de eventos en el registro de eventos se escanea y se registran las relaciones básicas entre las tareas de ordenación. Estas relaciones contienen información sobre el número de sucesiones directas entre tareas. En la fase de inducción, las relaciones de pedidos avanzados son inducidas a partir de las

Capítulo 1. Fundamentación teórica

básicas y nuevas tareas (por ejemplo, tareas invisibles y tareas duplicadas) pueden ser creados a partir de cero. Las relaciones de pedidos avanzados muestran si las tareas son causalmente dependientes o en paralelo. En la fase de construcción, el modelo final se construye a partir de las relaciones de pedidos avanzados de acuerdo con algunas reglas heurísticas. Todos los algoritmos de la serie *alpha* hacen algunas suposiciones sobre el registro dado. Hay un total de tres suposiciones importantes. La primera de ellas es que no debe haber datos con ruido en el registro. En otras palabras, ningún evento falta o es redundante o está en un lugar equivocado. La segunda es que el registro debe estar completo en función de las relaciones básicas de ordenación (como la siguiente relación). El último de ellos es que el modelo de proceso a generar del registro (llamándole el modelo potencial) debe expresarse en redes de Petri correctamente y no debe contener algunas construcciones especiales (es decir, bucles cortos, construcciones de elección obligatorias, tareas invisibles, tareas duplicadas o construcciones unión/división). (11)

1.10.2 Algoritmo Heuristic Miner HM

Este algoritmo puede lidiar con el ruido y puede ser utilizado para expresar el comportamiento principal (es decir, no todos los detalles y excepciones) registrados en un registro de eventos. Es compatible con la minería de todas las construcciones comunes en modelos de proceso (es decir, la secuencia, la elección, el paralelismo, bucles, tareas invisibles y algunos tipos de elecciones obligatorias), a excepción de las tareas duplicadas. El algoritmo HM tiene dos fases principales. En el primer paso, se construye un gráfico de dependencias. Este gráfico de dependencias contiene las dependencias causales que van a ser mantenidas en la construcción del modelo de red de Petri. A diferencia de los algoritmos basados en la abstracción, el HM toma en cuenta la frecuencia de las relaciones básicas de pedidos durante el cálculo de la fuerza de las relaciones causales. Por defecto el algoritmo crea una dependencia del mejor sucesor y predecesor causal de una tarea determinada. En una segunda etapa se establece la semántica de los puntos de división/unión en el gráfico de la dependencia. (12)

Capítulo 1. Fundamentación teórica

1.10.3 Algoritmo Integer Linear Programming ILP

Utilizando este algoritmo el número de plazas en la red de Petri obtenida es como máximo de segundo grado, hay que tener en cuenta que no se hacen suposiciones acerca de la integridad del registro (es decir, las dependencias causales sólo orienten la búsqueda de lugares), pero si no se encuentran dependencias causales, entonces se obtiene un modelo desajustado.

1.11 Herramientas para la minería de procesos: ProM

ProM es una de las herramientas más usada en el campo de la Minería de Procesos. Es de software libre, multiplataforma y está diseñada para que en ella se puedan desarrollar y ejecutar algoritmos de Minería de Procesos. Es un marco extensible que es compatible con una amplia variedad de técnicas de Minería de Procesos en forma de plugin. Es independiente de la plataforma, ya que es implementado en Java y puede ser descargado sin ningún costo. Está publicado bajo una licencia de código abierto (9). La herramienta ProM fue seleccionada para inspeccionar los registros de eventos obtenidos, con el fin de realizar análisis de minería de procesos sobre los mismos y luego mostrar los resultados a través de la web.

1.12 Tecnologías y el proceso de desarrollo de software utilizado

Se hace una breve descripción del ambiente de trabajo como las herramientas, metodología y tecnologías seleccionadas que ayudarán con la solución de la aplicación.

1.13 Lenguaje de programación

Los lenguajes de programación pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. (13)

1.13.1 Java:

Java es un lenguaje de programación orientado a objetos. Su diseño, robustez, el respaldo de la industria y su fácil portabilidad han hecho del mismo uno de los

Capítulo 1. Fundamentación teórica

lenguajes con un mayor crecimiento y amplitud de uso en distintos ámbitos de la industria de la informática, a través de él es posible crear cualquier tipo de programa particularmente para aplicaciones de cliente-servidor en la web (13). Entre sus Múltiples ventajas destacan:

- Se trata de un lenguaje independiente de la plataforma, es decir, cualquier programa creado a través de Java podrá funcionar correctamente en ordenadores de todo tipo y con sistemas operativos distintos (14).
- Con este lenguaje se facilita la creación de atractivas páginas web dinámicas mediante XML, ofreciendo un diseño mucho más atractivo que una página estática. Además permite incluir sonido y objetos multimedia así como bases de datos y otras funcionalidades (14).
- Cualquier dispositivo que sea compatible con este lenguaje de programación ofrece la posibilidad de ejecutar un programa creado en Java sin tener que instalar plugins frecuentemente (14).

1.13.2 JavaScript

JavaScript se ha convertido en un lenguaje de programación indispensable para la creación de páginas web dinámicas, el mismo es interpretado, por lo que no es necesario compilar los programas para ejecutarlo. Maneja objetos dentro de la página web y sobre ese objeto se pueden definir diferentes eventos. Dichos objetos facilitan la programación de páginas interactivas. Con esto se puede cambiar totalmente el aspecto de la página al gusto del usuario, evitándose tener en el servidor una página para cada gusto, hacer cálculos en base a variables cuyo valor es determinado por el usuario. Por medio de JavaScript y a través del DOM se puede acceder a todos los elementos que pueda tener dentro la propia página, como párrafos, divisiones, tablas, formularios y sus campos, o sea, se pueden modificar, suprimir, crear nuevos elementos y colocarlos en la página (15).

1.13.3 Hojas de estilo en cascada (CSS).

Hojas de Estilo en Cascada es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo

Capítulo 1. Fundamentación teórica

va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en el CSS afectará a todas las páginas vinculadas a ese CSS en las que aparezca ese elemento. CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos (16).

1.14 Herramientas de Modelado

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación (17).

1.14.1 Visual Paradigm for UML

Como herramienta CASE de modelado con UML se utilizó Visual Paradigm for UML en su versión 6.4. Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Constituye una herramienta de software libre de gran utilidad para el analista (18).

1.15 Entorno de desarrollo integrado. Eclipse Luna

Es una plataforma de desarrollo, diseñada para ser extendida de forma indefinida a través de plugins. Fue concebida desde sus orígenes para convertirse en una plataforma de integración de herramientas de desarrollo. No tiene en mente un

Capítulo 1. Fundamentación teórica

lenguaje específico, sino que es un IDE genérico, goza de mucha popularidad entre la comunidad de desarrolladores del lenguaje Java; entre sus principales características se encuentra:

El desarrollo sobre Eclipse se basa en los proyectos, que son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios. El IDE proporcionará asistentes y ayudas para la creación de proyectos. Por ejemplo, cuando se crea uno, se abre la perspectiva adecuada al tipo de proyecto que se está creando, con la colección de vistas, editores y ventanas pre configuradas por defecto.

Se incluye un potente depurador, de uso fácil e intuitivo, y que visualmente ayuda a mejorar el código. Para ello sólo se debe ejecutar el programa en modo depuración (con un simple botón). De nuevo, se tiene una perspectiva específica para la depuración de código, la perspectiva depuración, donde se muestra de forma ordenada toda la información necesaria para realizar dicha tarea.

Están disponibles en una gran cantidad de *plugins*, unos publicados por Eclipse, otros por terceros. Los hay gratuitos, de pago, bajo distintas licencias, pero casi para cualquier cosa que se imagine se tendrá el *plugin* adecuado (19).

1.16 Portal empresarial

Existen muchas definiciones de portal empresarial. Gartner (empresa consultora y de investigación de las tecnologías de la información), por ejemplo, lo explica de este modo:

Un portal es una infraestructura de software de Web que provee acceso e interacción con activos de información (información/contenido, aplicaciones y procesos de negocio), activos de conocimiento y recursos humanos, mediante la selección de audiencia identificada de una manera relacionada.

Por su parte, Heidi Collins proporciona esta definición:

Aplicación basada en la web que permite al trabajador del conocimiento el acceso a una amplia variedad de información relacionada con el negocio, con la que puede tomar decisiones y realizar acciones, independientemente de su ubicación y del formato en el que la información esté almacenada. A continuación se mencionan algunas de sus características:

Capítulo 1. Fundamentación teórica

- Permite el acceso fácil y eficiente a contenidos relevantes de una compañía, lo que ayuda a los usuarios a tomar decisiones sobre fundamentos más sólidos.
- Facilita la colaboración entre las personas, sin importar su ubicación física.
- Ofrece acceso a aplicaciones de negocio, necesarias para el trabajo diario de los empleados.
- Presenta su contenido de manera personalizada a los usuarios, al mostrarles lo que le interesa de acuerdo con su rol, ubicación física o área de trabajo (20).

1.17 Plataforma web. Liferay Portal

Liferay Portal es una fuente abierta rica de funcionalidad basada en Java que permite la creación de portales Web de una manera muy sencilla y rápida. Desde la perspectiva del usuario es un producto muy completo ya que cuenta con más de 60 portlets que ya están listas para realizar todas las tareas comunes de un portal moderno a nivel de empresa (21). Con Liferay, se puede configurar un sitio Web en muy poco tiempo sin tener que escribir código, dada la abundante disponibilidad de portlets y temas listos para su uso. Lo que es más interesante desde el punto de vista del desarrollo de funciones personalizadas, es que Liferay es también una plataforma de desarrollo completa, un marco real para el desarrollo del portal en Java. Este segundo aspecto, menos evidente para el usuario final, por lo que es una de las mejores plataformas para el desarrollo de portales y aplicaciones Web normalmente disponibles en la actualidad en el mundo del código abierto. Liferay Portal es un sistema de gestión de contenidos, un agregado de aplicación y una herramienta de colaboración de gran alcance (22).

1.18 Portlets

Los portlets son componentes modulares de interfaz de usuario gestionada y visualizada en un portal web. Producen fragmentos de código de marcado (html, xhtml) que se agregan en una página de un portal (23). Además, una página de

Capítulo 1. Fundamentación teórica

un portal se visualiza como una colección de ventanas de portlet que no se solapan, donde cada una de estas muestran un portlet. Por lo tanto, un portlet (o colección de portlets) se asemeja a una aplicación web que está hospedada en un portal.

Los portlets son módulos Web reutilizables que se ejecutan en un servidor del portal y proporcionan acceso al contenido, a las aplicaciones y a otros recursos basados en la web. Además, se ensamblan en una página más grande, con varias instancias del mismo portlet que muestran diferentes datos para cada usuario (24).

Características principales:

- Los portlets son manejados por un contenedor especializado, que controla su ciclo de vida.
- Los portlets generan contenido dinámico e interactúan con el cliente web mediante el uso del paradigma request/response (solicitud/respuesta).
- Los portlets son únicamente generados como parte de una página web y no como documentos completos, no están asociados directamente a una URL y no pueden generar contenido arbitrario, es por estas razones que son diferentes a los servlets.
- Los portlets proporcionan un objeto Portlet Preferences para almacenar las preferencias del usuario (24).

1.19 Metodología de desarrollo

El éxito de un software depende en gran medida de la metodología elegida por el equipo de desarrollo, pues de ella depende que se maximice el potencial del equipo o no en cuanto a los recursos y el tiempo empleado.

Existen dos grandes enfoques, tanto metodologías tradicionales y metodologías ágiles, las primeras están pensadas para el uso exhaustivo de documentación durante todo el ciclo del proyecto mientras que las segundas ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente (25).

Capítulo 1. Fundamentación teórica

Entre las principales metodologías tradicionales se pueden mencionar dos de las más conocidas: RUP (Rational Unified Process) y MSF (Microsoft Solution Framework) entre otros, que centran su atención en llevar una documentación exhaustiva de todo el proyecto y centran su atención en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto (25).

Entre los principales métodos ágiles están: XP (eXtreme Programming), Scrum, Iconix, Cristal Methods, AUP (Agil Unified Process) entre otras.

Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Nos lo proponen porque para muchos clientes esta flexibilidad será una ventaja competitiva y porque estar preparados para el cambio significar reducir su coste (25).

Metodologías Ágiles.	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.

Capítulo 1. Fundamentación teórica

Pocos roles.	Más roles.
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla 1: Diferencias entre metodologías ágiles y no ágiles.

Para el desarrollo del portlet que permita la monitorización del proceso de la gestión de órdenes comerciales en el CIM se utilizan las herramientas y tecnologías definidas por el Centro de Informatización Universitaria (CENIA) para el desarrollo de sistemas de gestión de información basados en tecnologías web. Para guiar el desarrollo del módulo se utiliza el proceso de desarrollo con enfoque ágil basado en el nivel 2 de CMMI (en español, Modelo de Madurez de Capacidad Integrado). A continuación, se realiza una descripción del mismo:

1.20 Metodología Unificada Ágil (AUP):

Una metodología de desarrollo de software se refiere al entorno que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información (26). Para guiar el proceso de desarrollo de la solución propuesta se escogió la Metodología Unificada Ágil (AUP), haciendo una variación de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. AUP es una forma simplificada del Proceso Racional Unificado (RUP, Rational Unified Process) desarrollada por Scott Ambler. Describe un enfoque simple del desarrollo del software usando técnicas y conceptos ágiles. Algunas técnicas usadas por AUP incluyen el desarrollo orientado a pruebas, modelado y gestión de cambios ágiles y refactorización de base de datos para mejorar la productividad (27).

AUP establece cuatro fases:

1. Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.

Capítulo 1. Fundamentación teórica

3. Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.

4. Transición: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción (28).

La UCI decide para el ciclo de vida de los proyectos mantener la fase de Inicio, en la cual se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto. Se unifican las restantes tres fases de AUP en una sola, a la que se denomina Ejecución que sería aquella fase en la que se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software, y se agrega una fase de cierre. En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

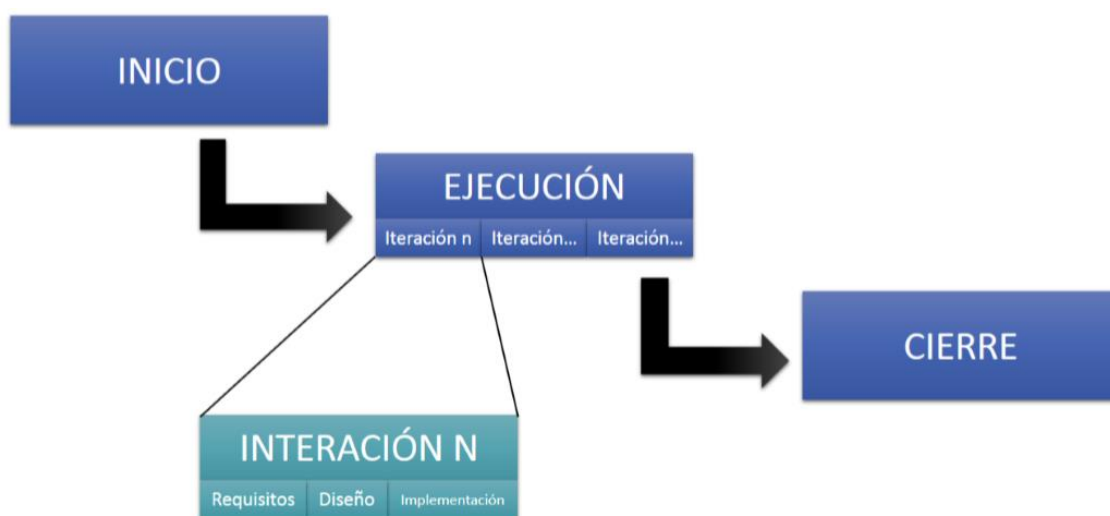


Ilustración 3: Fases e Iteraciones (28).

Capítulo 1. Fundamentación teórica

Esta metodología 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 tener ocho disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y 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 y la disciplina Despliegue se considera opcional. Las restantes tres 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 dos, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto).

Ventajas:

- El personal sabe lo que está haciendo: no obliga a conocer detalles.
- Simplicidad: apuntes concisos.
- Agilidad: procesos simplificados del RUP.
- Centrarse en actividades de alto valor: esenciales para el desarrollo.
- Herramientas independientes: a disposición del usuario.
- Fácil adaptación de este producto: de fácil acomodo (HTML).

Desventajas:

Como es un proceso simplificado, muchos desarrolladores eligen trabajar con RUP, por tener a disposición más detalles en el proceso (29).

1.21 Conclusiones parciales

A partir del estudio realizado como parte del marco teórico se dio a conocer un conjunto de elementos que definen el punto de partida sobre el cual se apoyará la investigación para iniciar la construcción de un software que cumpla con los objetivos propuestos. Se realizó un estudio sobre las diferentes herramientas

Capítulo 1. Fundamentación teórica

utilizadas en la actualidad en la minería de procesos para desarrollar un software que sea capaz de satisfacer las necesidades del cliente. Se han expuesto las herramientas y metodologías necesarias para el análisis y desarrollo de la solución.

Capítulo 2. Características y diseño del sistema

Capítulo 2. Características y diseño del sistema

2.1 Introducción

En el presente capítulo se dará a conocer la propuesta del sistema y los diagramas para apoyar la comprensión del funcionamiento del mismo. Tomándose como base la metodología AUP para guiar el proceso de desarrollo. También se definirá la arquitectura que se usará para la implementación del portlets para la monitorización de procesos para gestión de órdenes comerciales en el CIM.

2.2 Propuesta de solución

Actualmente el SOICIM no cuenta con un sistema que permita la monitorización de los procesos de una forma sencilla y práctica accesible para usuarios no tan avanzados en los sistemas informáticos.

Para resolver este problema se propone desarrollar un portlet que permita la visualización de los procesos de forma gráfica haciendo más fácil el control de los mismos.

A partir de las restricciones de negocio existentes, con el desarrollo de la propuesta de solución planteada, los administradores de sistemas podrán obtener una idea clara de cuanto se asemejan los procesos ideales de la empresa con lo que realmente ocurre.

La investigación realizada propone el desarrollo de un portlet que realice todas las etapas de la minería de procesos que propone el Modelo de Ciclo de vida L* para proyectos de minería de procesos. Lo anterior será el producto de varios resultados parciales. Con la implementación de este recurso se logrará visualizar los procesos descubiertos con un algoritmo de descubrimiento de procesos que genere una red de Petri. Para ello cuenta con el registro de eventos extraído de la base de datos del sistema SOICIM y deberá visualizar la red de Petri correspondiente. La propuesta se enmarca en la Etapa 2 del modelo de ciclo de vida de los proyectos de minería de datos L* y, específicamente, en la actividad de creación del modelo de control de flujo a partir de un registro de eventos.

Capítulo 3. Implementación y prueba del sistema

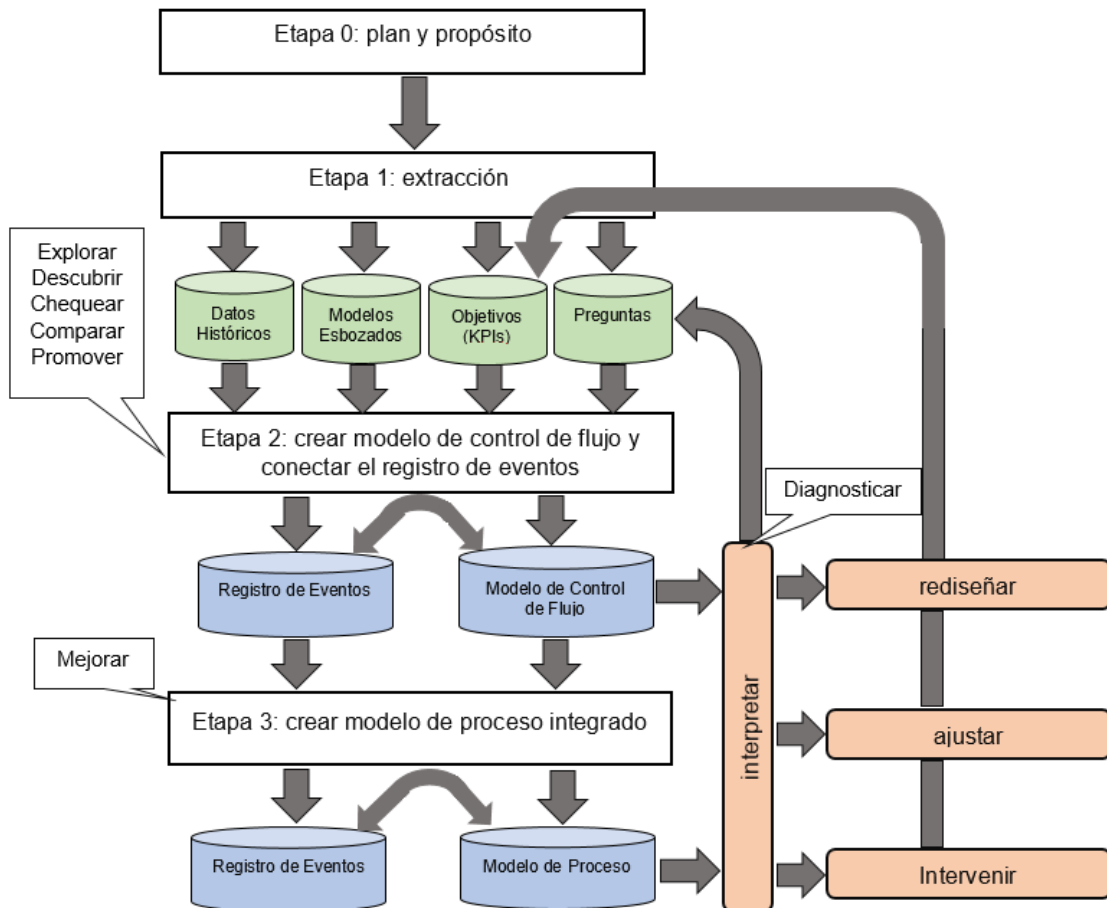


Ilustración 4: Modelo de Ciclo de vida de proyectos de minería de procesos (L*) (9).

2.2.1 Interpretación de una red de Petri

Las redes de Petri son ideales para la definición y análisis de los procesos complejos debido a que hacen las definiciones fáciles de entender para los no expertos. Esto facilita la comunicación entre diseñadores y usuarios. En la ilustración se muestra una red de Petri clásica:

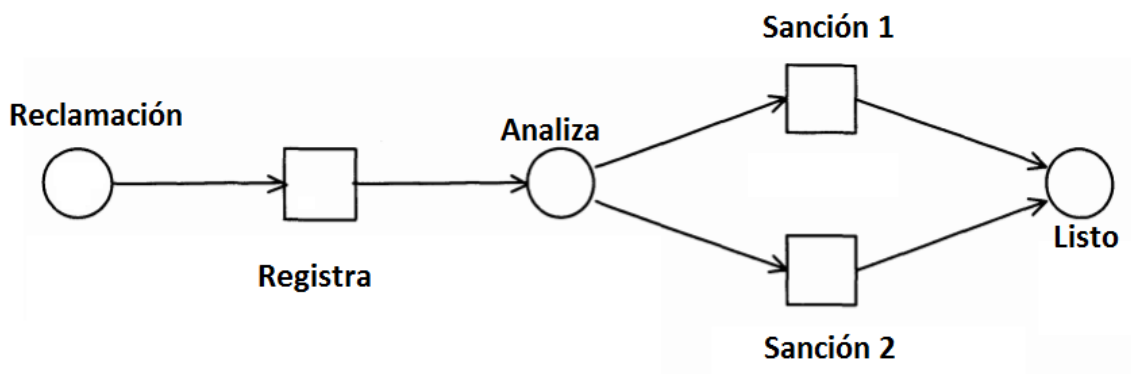


Ilustración 5: Clásica red de Petri.

Capítulo 3. Implementación y prueba del sistema

Como se explicó en epígrafes anteriores los lugares y las transiciones son representados gráficamente por círculos y rectángulos respectivamente. Un rectángulo representa las actividades a realizar dentro de un proceso y los círculos podrían interpretarse como un estado donde se toma la decisión de qué actividad será la siguiente dentro de la actual instancia del proceso. En el ejemplo que muestra la ilustración 4, el proceso se dispara cuando un cliente realiza una reclamación, la misma es registrada en los archivos de la institución y posterior a su análisis dependiendo de la gravedad de la queja se decide qué tipo de sanción emplear sobre los implicados, luego de ejecutarse la sanción escogida se llega a un estado final. Es importante conocer que las salidas que generan los lugares y las transiciones son interpretadas como el operador lógico OR y AND respectivamente.

2.3 Especificación de requisitos

La especificación de requisitos de software es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de casos de uso que describe todas las interacciones que tendrán los usuarios con el software. Los casos de uso también son conocidos como requisitos funcionales. Además de los casos de uso, la especificación de requisitos de software también contiene requisitos no funcionales o complementarios (30).

2.3.1 Requisitos funcionales (RF).

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer (31).

No	Nombre	Descripción	Complejidad
[RF1.]	Cargar registro de eventos.	El usuario debe ser capaz de cargar el registro de eventos desde cualquier ubicación en su estación de trabajo.	Baja

Capítulo 3. Implementación y prueba del sistema

[RF2.]	Mostrar listado de registros de eventos cargados.	El usuario debe ser capaz de visualizar en una lista los registros de eventos disponibles.	Baja
[RF3.]	Seleccionar registro de eventos	El usuario debe ser capaz de seleccionar un registro de eventos de un listado, ya sea para aplicarle un algoritmo de descubrimiento de procesos como para eliminarlo del listado de registros de eventos disponibles.	Baja
[RF4.]	Filtrar algoritmos de descubrimiento de procesos.	El usuario debe ser capaz de filtrar el listado de algoritmos de descubrimiento de procesos por el nombre del algoritmo a aplicar.	Baja
[RF5.]	Aplicar algoritmo de descubrimiento de procesos.	El usuario debe ser capaz de aplicar un algoritmo de descubrimiento de procesos, previamente seleccionado.	Alta
[RF6.]	Visualizar Red de Petri.	El usuario debe ser capaz de visualizar la red de petri generada por el algoritmo de descubrimiento de procesos aplicado previamente.	Alta

Tabla 2: Requisitos funcionales (RF).

2.3.2 Requisitos no funcionales (RNF)

Un requisito no funcional o atributo de calidad es un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales, son todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento (32).

Con el objetivo de estandarizar la redacción de los requisitos no funcionales (RNF) de Atributos de calidad se propone una nueva Taxonomía partiendo de la

Capítulo 3. Implementación y prueba del sistema

ISO 25010. Las partes que conforman la taxonomía pueden o no aplicar en dependencia del RNF que se esté evaluando.

No	Nombre	Descripción
[RNF1.]	Portabilidad	El usuario debe ser capaz de agregar o retirar el portlet del portal, según lo requiera, sin afectarse el funcionamiento del portlet.
[RNF2.]	Usabilidad	El usuario debe ser capaz de interactuar con las pantallas del portlet y con las opciones disponibles de un modo fácil e intuitivo.
[RNF3.]	Mantenibilidad	El portlet podrá modificarse sin afectar el funcionamiento del portal.
[RNF4.]	Soporte	El portlet podrá desplegarse en servidores Apache Tomcat 6.0 o superior.
[RNF5.]	Soporte	El SO del servidor debe tener instalada la Máquina Virtual de Java (JVM) versión 1.7 o superior.
[RNF6.]	Soporte	El servidor debe utilizar como sistema operativo (SO) GNU/Linux Ubuntu u otro basado en Debian.

Tabla 3: Requisitos no Funcionales.

2.4 Historias de Usuario

Definición: Una historia de usuario (HU, o user story en inglés) describe una funcionalidad que, por sí misma, aporta valor al usuario (33).

Las HU son representadas por tablas que contienen los siguientes campos o secciones:

- Número: Número que representa el identificador de la historia de usuario.
- Nombre de Historia: Nombre que identifica la HU.
- Prioridad: El valor de este campo es Baja, Media o Alta, definido por el cliente dependiendo de la importancia y el orden para el proceso de desarrollo.

Capítulo 3. Implementación y prueba del sistema

- Iteración Asignada: Número de la iteración en la cual se desarrollará la HU.
- Actor: Personajes o entidades que participarán en un caso de uso.
- Tiempo Estimado: Tiempo estimado en horas que se le asignará.
- Tiempo Real: Tiempo en horas dedicado al cumplimiento del requisito.
- Descripción: Breve descripción del proceso que define la historia.
- Observaciones: Algunas aclaraciones que es importante señalar acerca de la historia.
- Prototipo elemental de interfaz gráfica de usuario: Descrito el requisito mediante una breve imagen.

2.4.1 Descripción de las Historias de Usuario

Historia de Usuario	
Número: HU1	Nombre del requisito: Cargar el registro de eventos.
Programador: Ever Leandro Santiesteban Jiménez	Iteración Asignada: 1
Prioridad: Baja	Tiempo Estimado: 21 horas
Riesgo en Desarrollo: <ul style="list-style-type: none"> ○ Problemas eléctricos. ○ Problemas técnicos de los servicios de redes. 	Tiempo Real: 28 horas
Descripción: El sistema dará al usuario la opción de cargar un fichero en formato XES conocido como registro de eventos, a través del botón "Subir". Una vez seleccionada esta opción se le podrá aplicar a dicho registro los diferentes algoritmos de descubrimiento disponibles.	
Observaciones:	

Capítulo 3. Implementación y prueba del sistema

Prototipo de interfaz:

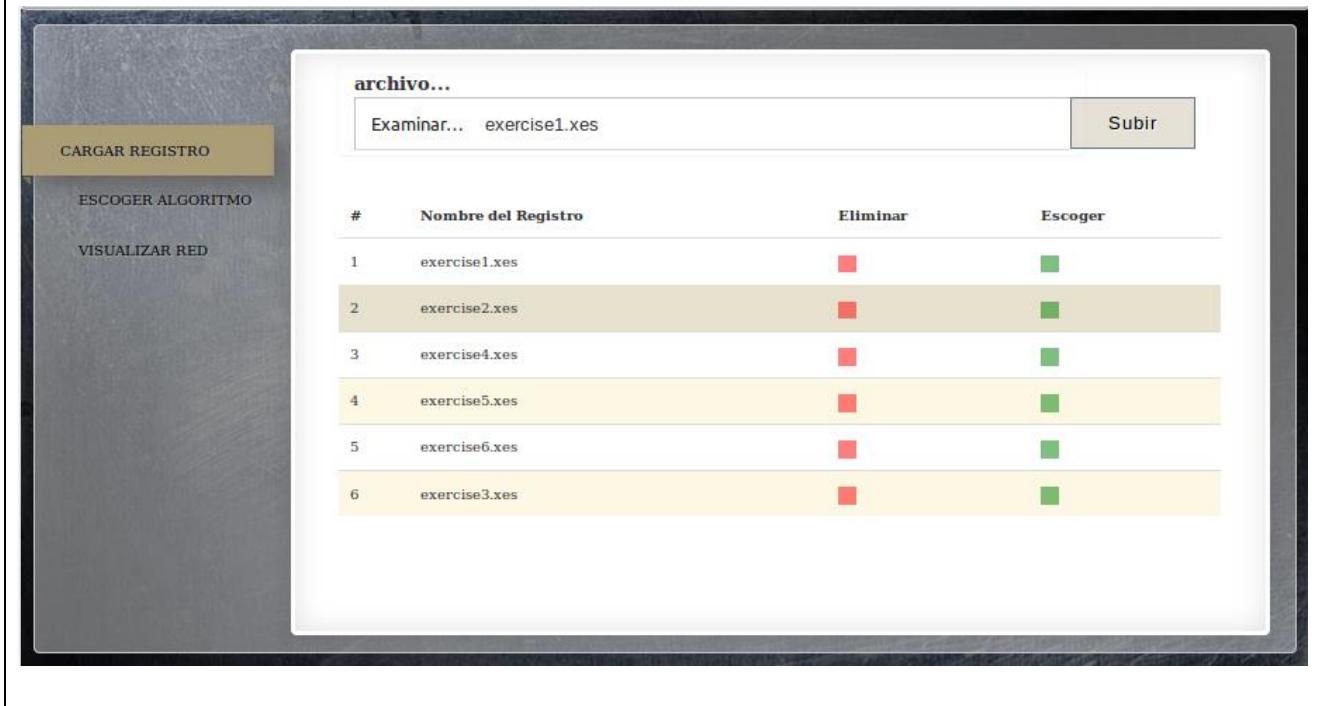


Tabla 4: Historia de Usuario HU1.

Historia de Usuario	
Número: HU2	Nombre del requisito: Mostrar listado de registros de eventos cargados.
Programador: Ever Leandro Santiesteban Jiménez.	Iteración Asignada: 1
Prioridad: Baja	Tiempo Estimado: 21 horas
Riesgo en Desarrollo: <ul style="list-style-type: none"> ○ Problemas eléctricos. ○ Problemas técnicos de los servicios de redes. 	Tiempo Real: 28 horas
Descripción: El usuario debe ser capaz de visualizar en una lista los registros de eventos disponibles.	
Observaciones:	

Capítulo 3. Implementación y prueba del sistema

Prototipo de interfaz:

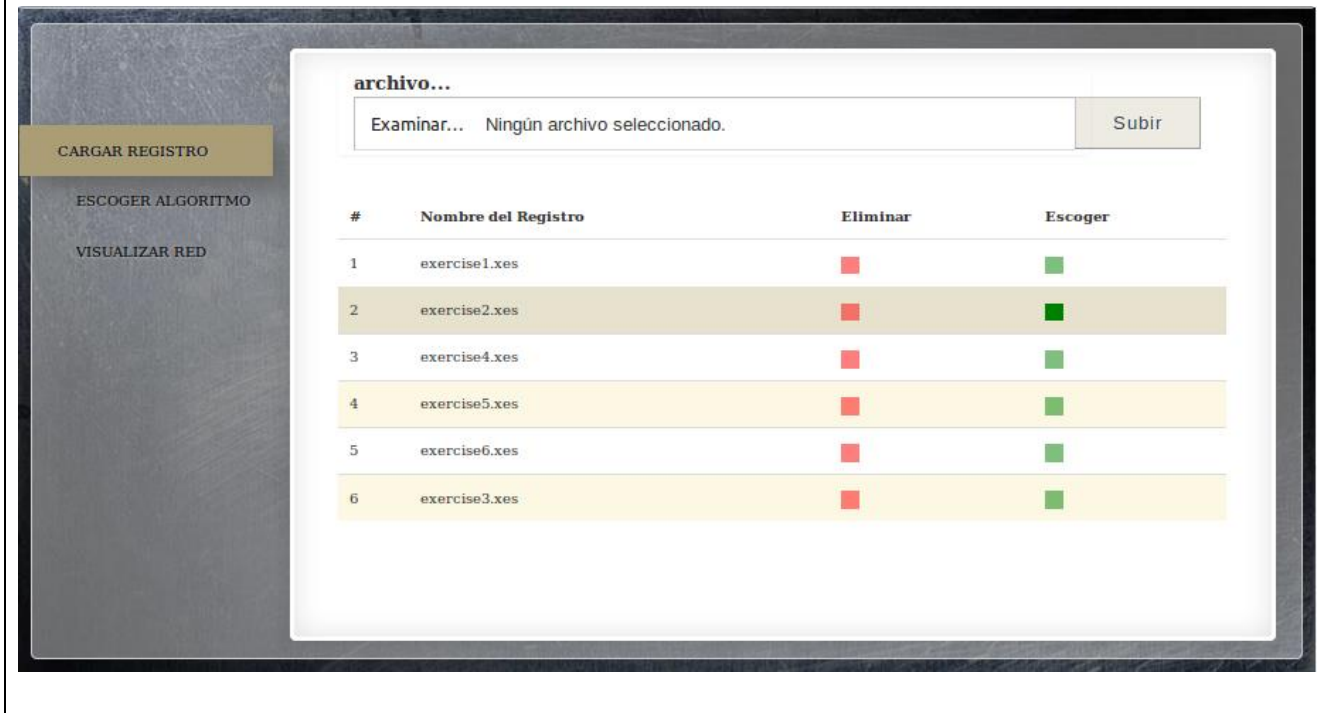


Tabla 5: Historia de Usuario HU2.

Historia de Usuario	
Número: HU3	Nombre del requisito: Seleccionar registro de eventos.
Programador: Ever Leandro Santiesteban Jiménez.	Iteración Asignada: 1
Prioridad: Baja	Tiempo Estimado: 7 horas.
Riesgo en Desarrollo: <ul style="list-style-type: none"> ○ Problemas eléctricos. ○ Problemas técnicos de los servicios de redes. ○ Problemas técnicos de la computadora. 	Tiempo Real: 14 horas.
Descripción: Una vez cargado el registro de eventos, el sistema debe mostrar al usuario una tabla con las siguientes columnas: #, Nombre del registro, Eliminar y Escoger; de este	

Capítulo 3. Implementación y prueba del sistema

listado el usuario debe ser capaz de seleccionar un registro de eventos ya sea para aplicarle un algoritmo de descubrimiento de procesos como para eliminarlo del listado de registros de eventos disponibles.

Observaciones:

Prototipo de interfaz:

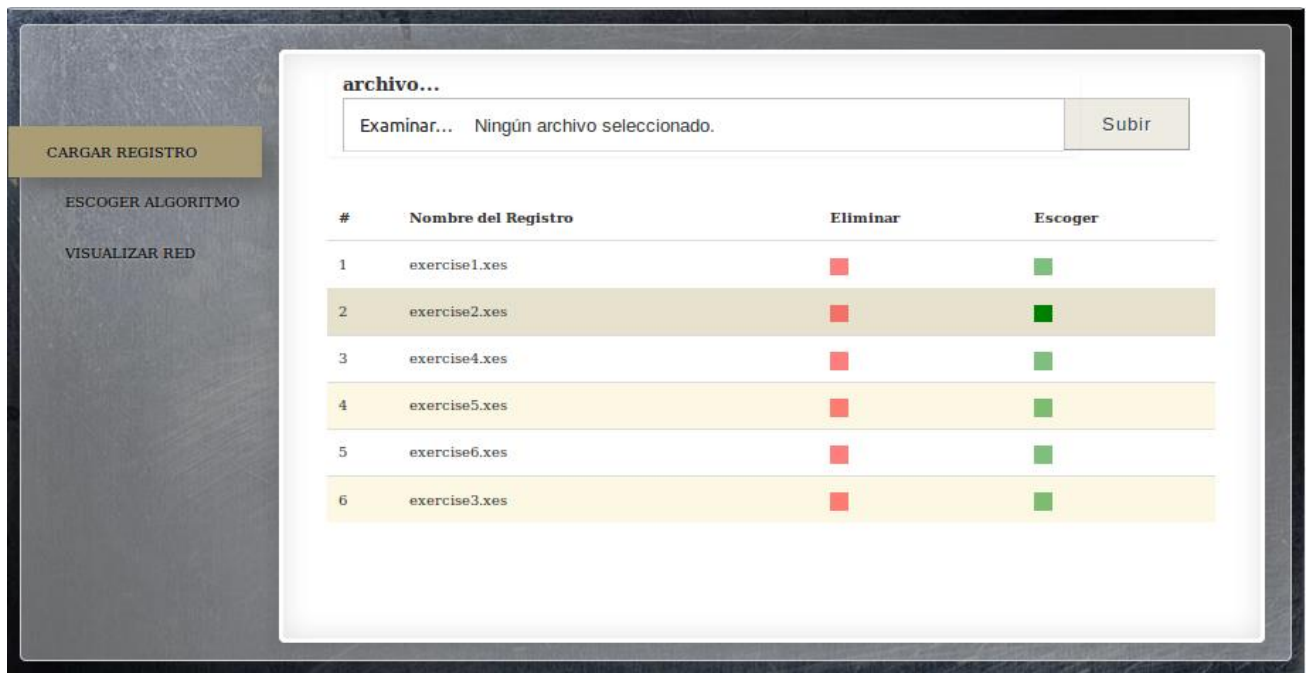


Tabla 6: Historia de Usuario HU3.

Historia de Usuario	
Número: HU4	Nombre del requisito: Filtrar algoritmos de descubrimiento de procesos.
Programador: Ever Leandro Santiesteban Jiménez	Iteración Asignada: 1
Prioridad: Baja	Tiempo Estimado: 28 horas.
Riesgo en Desarrollo: <ul style="list-style-type: none"> ○ Problemas eléctricos. ○ Problemas técnicos de los servicios de 	Tiempo Real: 14 horas.

Capítulo 3. Implementación y prueba del sistema

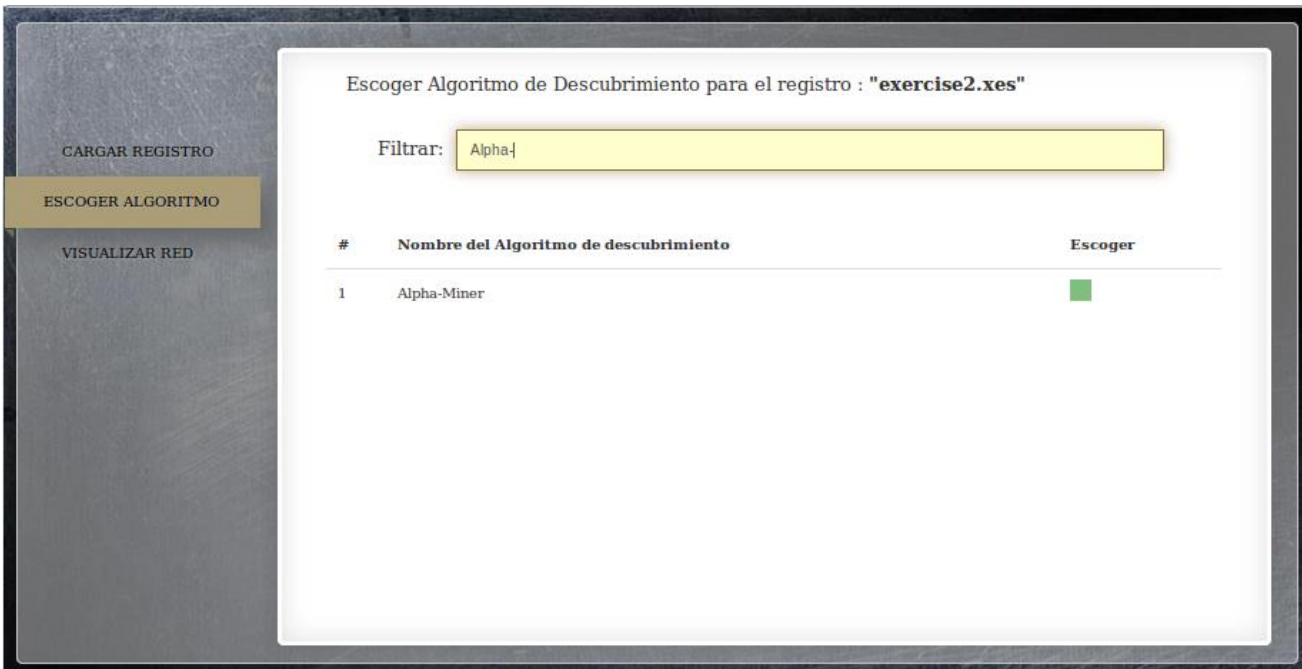
<p>redes.</p> <ul style="list-style-type: none"> ○ Problemas técnicos de la computadora. 	
<p>Descripción: Una vez cargado el registro de eventos, estará disponible en el sistema la opción: “ESCOGER ALGORITMO”. En esta vista se muestra un campo que permite el filtrado de los algoritmos de descubrimiento por el nombre del algoritmo a aplicar.</p>	
<p>Observaciones:</p>	
<p>Prototipo de interfaz:</p> 	

Tabla 7: Historia de Usuario HU4.

Historia de Usuario	
<p>Número: HU5</p>	<p>Nombre del requisito: Aplicar algoritmo de descubrimiento de procesos.</p>
<p>Programador: Ever Leandro Santiesteban Jiménez</p>	<p>Iteración Asignada: 2</p>
<p>Prioridad: Alta</p>	<p>Tiempo Estimado: 140 horas.</p>

Capítulo 3. Implementación y prueba del sistema

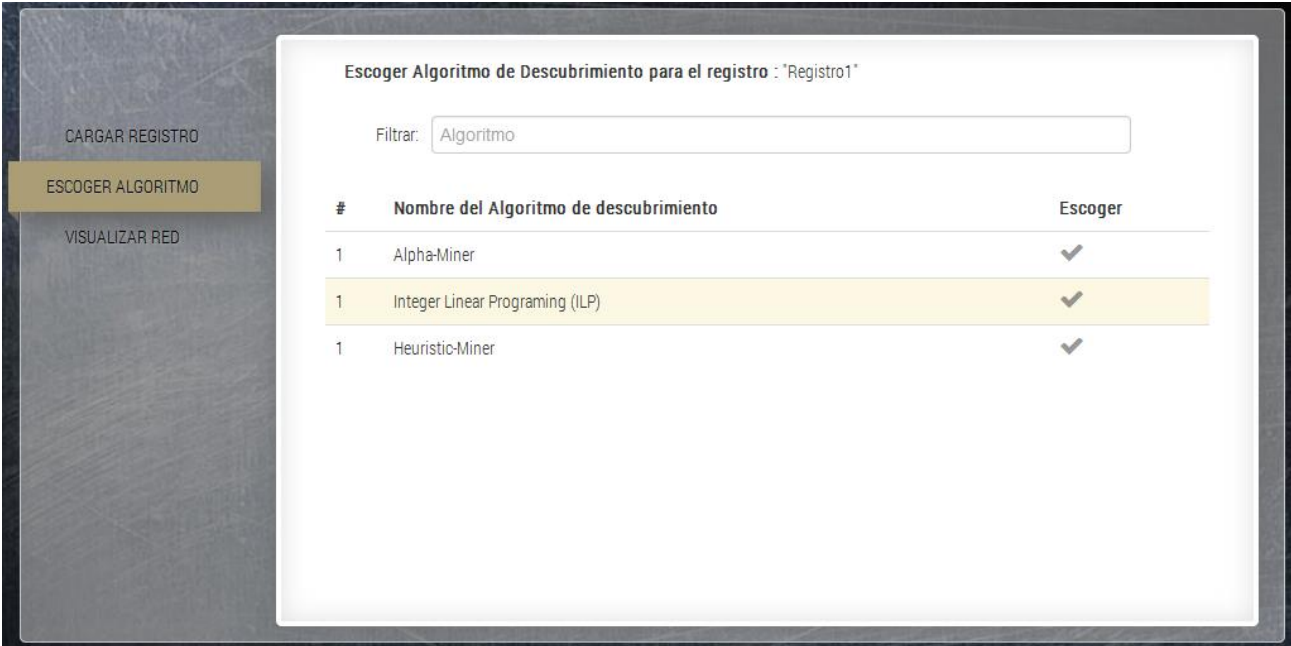
<p>Riesgo en Desarrollo:</p> <ul style="list-style-type: none"> ○ Problemas eléctricos. ○ Problemas técnicos de los servicios de redes. ○ Problemas técnicos de la computadora. 	<p>Tiempo Real: 154 horas.</p>
<p>Descripción: Una vez cargado y seleccionado el registro de eventos, el sistema habilitará la opción: “ESCOGER ALGORITMO”, mostrando al usuario una tabla que permita escoger uno de los algoritmos disponibles en el sistema para aplicárselo al registro de eventos previamente seleccionado.</p>	
<p>Observaciones:</p>	
<p>Prototipo de interfaz:</p> 	

Tabla 8: Historia de Usuario HU5.

<p>Historia de Usuario</p>	
<p>Número: HU6</p>	<p>Nombre del requisito: Visualizar red de Petri.</p>

Capítulo 3. Implementación y prueba del sistema

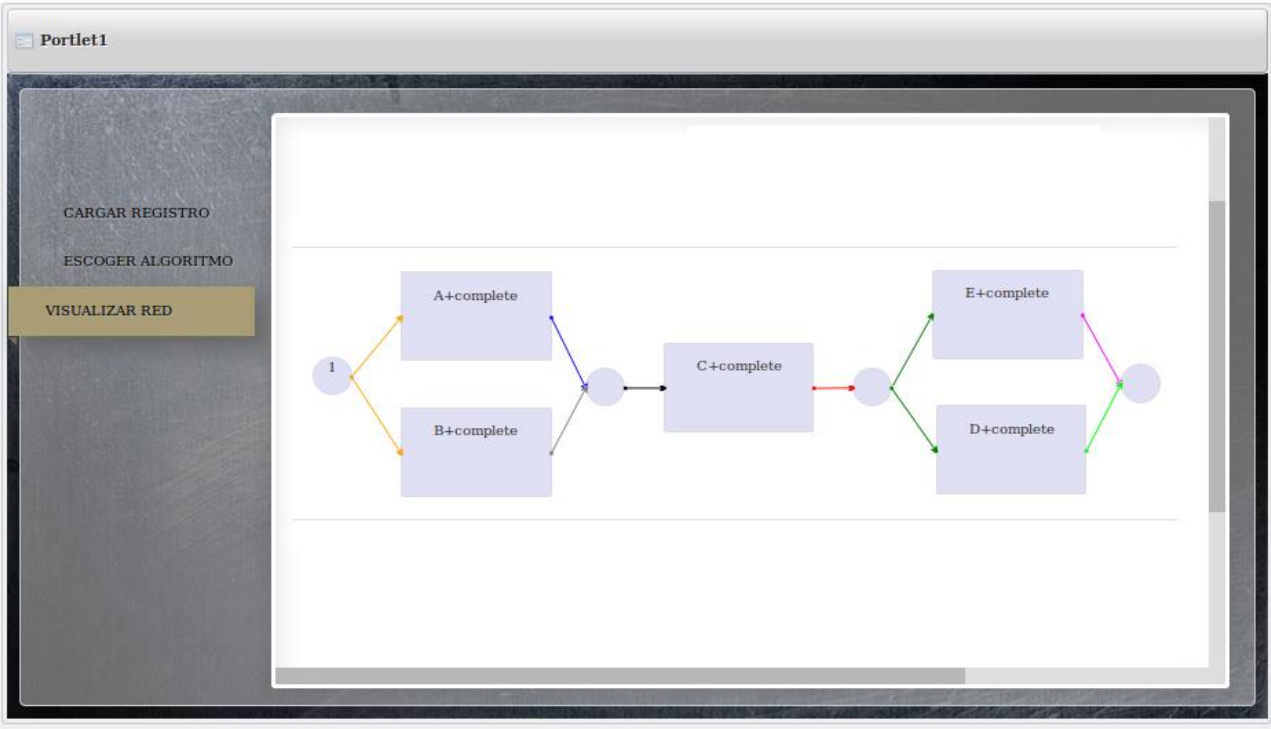
Programador: Ever Leandro Santiesteban Jiménez	Iteración Asignada: 2
Prioridad: Alta	Tiempo Estimado: 210 horas
Riesgo en Desarrollo: <ul style="list-style-type: none">○ Problemas eléctricos.○ Problemas técnicos de los servicios de redes.○ Problemas con el rendimiento de la computadora.	Tiempo Real: 224 horas
Descripción: Una vez confirmado el algoritmo de descubrimiento de procesos que se desea aplicar, el sistema mostrará una red de Petri correspondiente a la aplicación de dicho algoritmo sobre el registro de eventos escogido.	
Observaciones:	
Prototipo de interfaz:	
	

Tabla 9: Historia de Usuario HU6

Capítulo 3. Implementación y prueba del sistema

2.5 Plan de Iteraciones

Después de identificar y elaborar cada una de las HU, y determinar la estimación del esfuerzo, continúa la confección del plan de iteraciones. Para cada iteración son seleccionas las HU de acuerdo al orden preestablecido.

Se definieron dos iteraciones para el desarrollo de las partes funcionales de la aplicación. A continuación, se explican y justifican cada una de las iteraciones:

- Iteración 1: Para esta iteración se desarrollan las HU asociadas a los requisitos funcionales de complejidad baja.
- Iteración 2: En esta iteración se desarrollan las HU asociadas a los requisitos funcionales de prioridad alta. Para la implementación de los requisitos de esta iteración es tomado como apoyo la iteración anterior.

Para aproximar el tiempo de ejecución de las iteraciones se tomó como medida de tiempo las horas, constando que una semana tiene 5 días (lunes, martes, miércoles, jueves, viernes), de los cuales se trabaja aproximadamente 7 horas diarias. A continuación, se muestra en la tabla 6 con el plan de iteraciones con el tiempo estimado.

Iteración	Historias de usuario	Puntos de estimación (horas).
1	Cargar el registro de eventos.	77
	Mostrar listado de registros de eventos cargados.	
	Seleccionar registro de eventos.	

Capítulo 3. Implementación y prueba del sistema

	Filtrar algoritmos de descubrimiento de procesos.	
2	Aplicar algoritmo de descubrimiento de procesos.	350
	Visualizar red de Petri.	

Tabla 10: Plan de iteraciones.

Después de realizar un análisis se pudo estimar que el costo estimado para el desarrollo es de 427 horas, equivalente a 12 semanas.

2.6 Estilo arquitectónico:

Sistemas basados en llamado y retorno: Este estilo arquitectónico permite al arquitecto del sistema construir una estructura de programa relativamente fácil de modificar y ajustar a escala. Persigue obtener cualidades de escalabilidad y procedimiento remoto, modificabilidad (34).

2.6.1 Arquitectura de la aplicación:

Para el desarrollo de la solución se decidió utilizar una arquitectura en 2 Capas, lo que permitirá que las distintas partes de la aplicación se puedan modificar sin que esto afecte las demás capas, es decir, separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación y la lógica de negocio.

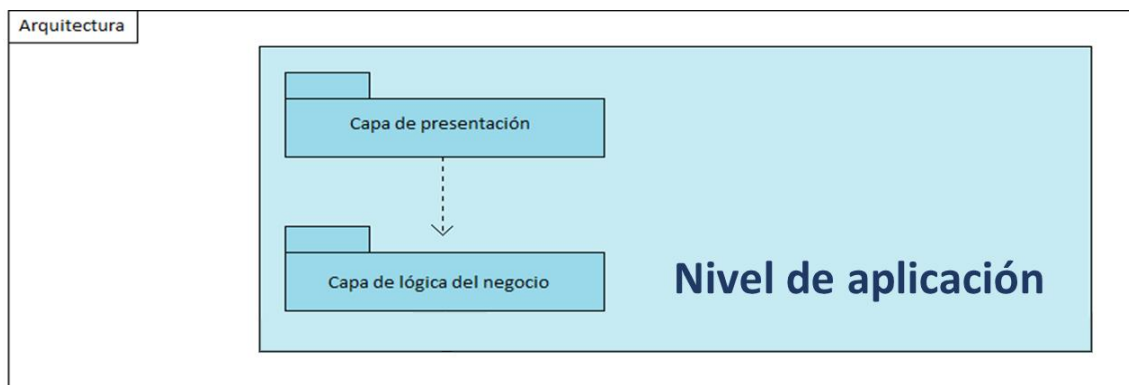


Ilustración 6: Arquitectura de la aplicación.

Capítulo 3. Implementación y prueba del sistema

2.6.1.1 Capa de presentación: Se refiere a la presentación del programa frente al usuario, esta presentación debe cumplir su propósito con el usuario final, una presentación fácil de usar y amigable. También las interfaces deben ser consistentes con la información dentro del software (Por ejemplo; en los formularios no debe haber más que lo necesario), tomar en cuenta los requerimientos del usuario, la capa de presentación va de la mano con la capa de la lógica de negocio (35).

2.6.1.2 Capa de lógica de negocios: En esta capa es donde se encuentran los programas que son ejecutados, recibe las peticiones del usuario y posteriormente envía las respuestas. Esta capa es muy importante pues es donde se establecen todas aquellas reglas que se tendrán que cumplir (35).

2.8 Patrones de diseño:

Los patrones de diseño son la piedra angular para la búsqueda de soluciones a problemas de origen común en el desarrollo e implementación del software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño no es más que una solución a un problema de diseño. Una de las características que debe poseer una solución para ser considerada un patrón es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (36).

2.8.1 Patrones GRASP

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es el controlador quien recibe los datos del usuario y quien los envía a las distintas clases según el método llamado (37).

Queda evidenciado el uso del patrón a través de la clase ControladorPortlet, que maneja las peticiones realizadas por los usuarios y que tendrán algún impacto o requieran de una respuesta de la aplicación.

Capítulo 3. Implementación y prueba del sistema

Bajo Acoplamiento: Es una medida de la fuerza con que una clase se relaciona con otras; es decir, no posee numerosas relaciones (37).

Queda evidenciado a través de la arquitectura en capas, es decir, que si existe algún cambio en la capa inferior no afectaría la capa superior o viceversa.

Alta cohesión: El patrón Alta Cohesión es una medida que determina cuán relacionadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo colosal; una clase con baja cohesión realiza un trabajo excesivo, haciéndola difícil de comprender, reutilizar y conservar (37).

Queda evidenciado el uso del patrón a través de las clases: SalidaProM, EntradaProM, que serían aquellas que controlan el manejo de datos que se le proporciona y se le pide a la herramienta.

2.8.2 Patrones GOF

Facade: Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar (38).

Sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas (38).

El uso del patrón mencionado anteriormente, queda evidenciado a través de la clase EntradaProM que sirve de intermediaria entre la herramienta ProM y las demás clases que necesiten de las funcionalidades de esta última.

Singleton: Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella (38).

El patrón de diseño singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto (38).

El uso del patrón mencionado anteriormente, queda evidenciado a través de la clase Variable, que proporciona un punto de acceso global a ella.

Capítulo 3. Implementación y prueba del sistema

2.9 Modelo de diseño:

Diagrama de clases:

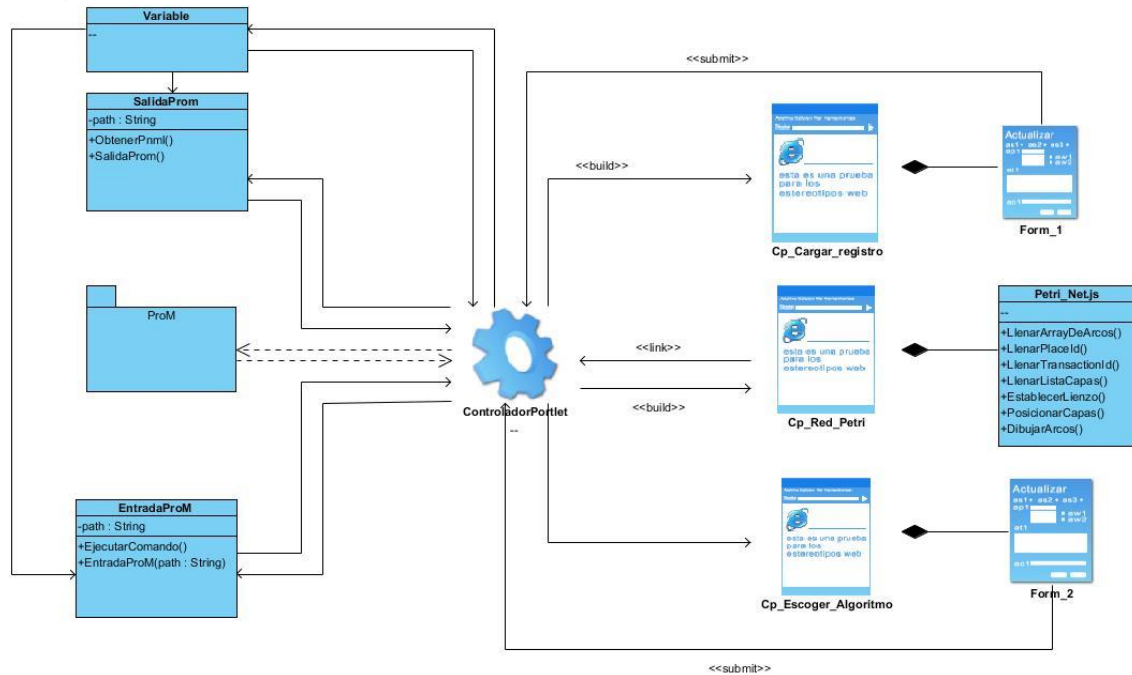


Ilustración 7: Diagrama de clases.

2.9.1 Algoritmo visualizador

Durante el proceso de desarrollo fue necesaria la implementación de un algoritmo que permitiera mostrar de forma gráfica, usando tecnologías web, la red de Petri generada por la herramienta ProM. Como resultado se obtuvo un módulo **Petri_Net.js**, desarrollado en javascript que es capaz de estructurar un modelo de procesos y mostrar en pantalla una red. A continuación se muestra el pseudocódigo del algoritmo mencionado:

Capítulo 3. Implementación y prueba del sistema

```
DibujarCapas( ){
    iterador=0;
    mientras(iterador<listaDeCapas.longitud){
        coordenadas= obtenerPosicionDeCapa(iterador);
        iterador2=0;
        Mientras(iterador2<listaDeCapas[iterador].longitud){
            posicionarEtiqueta(coordenadas.X,coordenadas.Y,
                                listaDeCapas[iterador][iterador2]);
            iterador2=iterador2+1;
        }
        iterador=iterador+1;
    }
}
```

Ilustración 8: Algoritmo DibujarCapas.

Métodos auxiliares y atributos utilizados:

Método o atributo	Descripción
listaDeCapas	Es una lista que almacena las capas organizadas por orden de pintado y a su vez cada capa va a estar compuesta por las etiquetas que conforman la red.
obtenerPosiciónDeCapa(int:index)	Devuelve un punto(x,y) con la posición exacta que debe tener la capa que se desea posicionar;
PosicionarEtiqueta(int:x,int:y,String:Etiqueta)	Posiciona la Etiqueta pasada por parámetro en la posición especificada.

Tabla 11: Métodos auxiliares del módulo Petri_Net.js.

2.9.2 Descripción de las clases del diseño:

Nombre	Descripción
ControladorPortlet	Esta clase es la encargada de manejar las peticiones realizadas por el usuario y proporcionar a cada una de las

Capítulo 3. Implementación y prueba del sistema

	restantes clases del sistema la información que necesite en dependencia de la acción realizada, es además la responsable de construir cada una de las páginas clientes dentro del portlet.
EntradaProM	Esta clase implementa todas las funcionalidades necesarias para comunicarse con la herramienta ProM; es la encargada de enviarle por medio de un script, a dicha herramienta, la información proporcionada por el usuario para obtener una red de petri.
SalidaProM	Esta clase es la encargada de cargar la salida generada por la herramienta ProM y extraer de ella toda la información referente a la red de Petri que espera visualizar el usuario.
Variable	Esta clase se encarga de almacenar en memoria variables globales, que sean accesibles para cada una de las clases implementadas por el sistema y conserven el mismo valor independientemente del usuario que se encuentre trabajando en el momento.
Cp_Red_Petri	Es la interfaz que mostrará al usuario la red de Petri correspondiente a la ejecución del algoritmo de descubrimiento de procesos seleccionado sobre el registro de eventos cargado con anterioridad.
Cp_Seleccionar_Algoritmo	Esta interfaz permitirá al usuario escoger, a través de una tabla, uno de los algoritmos de descubrimiento disponibles para aplicárselo al registro de eventos precargado.
Cp_Cargar_Fichero	Esta es la interfaz encargada de proveer al usuario la opción de cargar un nuevo registro de eventos en formato XES, además de dar la posibilidad de eliminar alguno que haya sido cargado con anterioridad y ya no se necesite su uso.

Tabla 12: Descripción de las clases del diseño.

Capítulo 3. Implementación y prueba del sistema

2.10 Conclusiones parciales

- Para un correcto trabajo en el cumplimiento de los requisitos definidos, se estableció la arquitectura que debe tener la aplicación a desarrollar, así como los patrones de diseño que permitirán obtener ventajas significativas en su utilización.
- Se expusieron y explicaron todos los artefactos generados por la metodología AUP verificándose la factibilidad de su uso para dar cumplimiento a las tareas que impuso la solución propuesta.

Capítulo 3. Implementación y prueba del sistema

Capítulo 3. Implementación y prueba del sistema

3.1 Introducción

En el presente capítulo se describe el proceso de implementación de la solución propuesta. Se plasman los casos de pruebas realizados a las historias de usuario correspondientes demostrando de esta forma el cumplimiento de los requerimientos definidos.

3.2 Estándar de código

Un Estándar de programación es: Una forma de "normalizar" la programación de forma tal que al trabajar en un proyecto, cualquier persona involucrada en el mismo tenga acceso y comprenda el código (39).

Variables:

Para estos identificadores se hace uso de la variante lowerCamelCase. Empiezan con minúsculas y si estos identificadores están compuestos por varias palabras las siguientes comenzarán con mayúscula.

Métodos:

Los nombres de métodos deben iniciar con un verbo.

Ejemplo:

```
public void crearPnml (...) {...}
```

Los métodos para obtener campos privados en las clases tienen el prefijo "get".

Ejemplo:

```
public String getPnml () {...}
```

Los modificadores de campos privados en las clases tienen el prefijo "set".

Ejemplo:

```
public void setAlgoritmo (...) {...}
```

Los obtenedores con el resultado de booleano tienen como nombre un verbo.

Capítulo 3. Implementación y prueba del sistema

Ejemplo:

```
public Boolean get(...) {...}
```

Constantes:

Las constantes o campos finales son escritos en letras minúsculas.

Ejemplo:

```
portlet = new ArrayList<Portlet>();
```

Clases:

Para estos identificadores se hace uso de la variante UpperCamelCase. Todas las palabras que componen a dichos identificadores empezarán con mayúscula.

Ejemplo:

```
public class PortletMVC implements MVCPortlet{...}
```

Comentarios de las clases:

Al inicio de cada clase se debe describir con un comentario de bloque el propósito de la clase e instrucciones de uso.

Ejemplo:

```
/**
 *
 * El método permite obtener el archivo de tipo pnml
 * generado por el PROM
 *
 * @parámetro String url
 * el primer valor
 * @retorna cadenaString
 *
 *
 */
```

Capítulo 3. Implementación y prueba del sistema

3.3 Pruebas:

Las pruebas de software, son los procesos que permiten verificar y revelar la calidad de un sistema. Son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas.

Las pruebas de software se integran dentro de las diferentes fases del ciclo del software que se ubican dentro de la Ingeniería de Software. Para determinar el nivel de factibilidad y calidad se le realizan pruebas a la aplicación mediante técnicas experimentales usando pruebas de caja negra.

3.3.1 Pruebas de caja negra

- Pruebas que se llevan a cabo sobre la interfaz del software.
- El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene (no se ve el código).

3.3.2 Estrategia de pruebas

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: Cargar el registro de eventos.	EC 1.1: Se obtuvo correctamente el registro de eventos.	El sistema debe permitir cargar un nuevo registro de eventos en formato XES al servidor.	1- En el portlet, accionar el input: "Cargar Registro". 2-Se procederá a la selección del archivo que se desea subir al servidor. 2-Luego de Accionar el botón "Subir", en caso de que el archivo no

Capítulo 3. Implementación y prueba del sistema

			tenga el formato adecuado el sistema mostrará el mensaje “No se ha podido cargar el archivo por la siguiente razón: no es un archivo válido”, si todo ocurrió correctamente el sistema mostrará en una tabla el archivo cargado.
SC 2: Eliminar registro de eventos cargado.	EC 2.1: Se obtuvo correctamente el registro de eventos.	Luego de haber cargado uno o varios registros de eventos el sistema brinda la opción de borrarlos para evitar la acumulación de registros innecesarios en el servidor.	1-Dentro de la tabla que muestra los registros que han sido cargados previamente, accionar la opción eliminar que le corresponde. 2-El sistema muestra nuevamente la tabla esta vez sin el archivo eliminado.
SC 3: Filtrar algoritmos de descubrimiento de procesos.	EC 3.1: Cada uno de los algoritmos aparecen en la “Tabla de algoritmos” luego de poner su nombre o parte de él en el filtro.	El sistema debe Filtrar por la columna: “Nombre del algoritmo” cada uno de los algoritmos disponibles.	1-Introducir un nombre válido dentro del input Filtrar. 2-El sistema muestra una tabla con los algoritmos que el nombre o parte de él corresponden al valor filtrado.

Capítulo 3. Implementación y prueba del sistema

<p>SC 4: Aplicar algoritmo de descubrimiento de procesos a un registro de eventos precargado.</p>	<p>EC 4.1: El algoritmo Alpha-Miner fue ejecutado satisfactoriamente al registro precargado.</p>	<p>El sistema debe permitir aplicarle el algoritmo de descubrimiento Alpha-Miner a un registro de eventos proporcionado por el usuario.</p>	<p>1-Una vez seleccionado el algoritmo Alpha-Miner el sistema muestra un mensaje de confirmación. 2-Luego de cargar los datos necesarios, se redirecciona a la vista "VISUALIZAR RED".</p>
	<p>EC 4.2: El algoritmo Huristic-Miner fue ejecutado satisfactoriamente al registro precargado.</p>	<p>El sistema debe permitir aplicarle el algoritmo de descubrimiento Huristic-Miner a un registro de eventos proporcionado por el usuario.</p>	<p>1-Una vez seleccionado el algoritmo Heuristic-Miner el sistema muestra un mensaje de confirmación. 2-Luego de cargar los datos necesarios, se redirecciona a la vista "VISUALIZAR RED".</p>
	<p>EC 4.3: El algoritmo Flower-Miner fue ejecutado satisfactoriamente al registro precargado.</p>	<p>El sistema debe permitir aplicarle el algoritmo de descubrimiento Flower-Miner a un registro de eventos proporcionado por el usuario.</p>	<p>1-Una vez seleccionado el algoritmo Flower-Miner el sistema muestra un mensaje de confirmación. 2-Luego de cargar los datos necesarios, se redirecciona a la vista "VISUALIZAR RED".</p>

Capítulo 3. Implementación y prueba del sistema

SC 5: Mostrar red de Petri	EC 5.1: La red de Petri se muestra correctamente.	El sistema debe mostrar al usuario una red de Petri derivada de la aplicación de un algoritmo de descubrimiento al registro de eventos precargado.	1-Una vez cargado el registro de eventos y aplicado el algoritmo de descubrimiento deseado, accionar en el link "VISUALIZAR RED". 2-El sistema mostrará la red de Petri correspondiente.
----------------------------	---	--	---

Tabla 13: Estrategia de pruebas.

3.3.2 Técnica de la Partición de Equivalencia:

La partición de equivalencia es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. En otras palabras, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realizada con cualquier otro valor de dicha clase. Esto quiere decir que, si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error. Y viceversa, si un caso de prueba no ha detectado ningún error, es de esperar que ninguno de los casos de prueba correspondientes a la misma clase de equivalencia encuentre ningún error (40).

3.3.3 Casos de pruebas

Las celdas de la tabla contienen **V**, **I**, o **N/A**. **V** indica válido, **I** indica inválido, y **N/A** que no es necesario proporcionar un valor del dato en este caso, dado que es irrelevante.

Capítulo 3. Implementación y prueba del sistema

Id del escenario	Escenario	Variable 1 Nombre	Respuesta del Sistema	Resultado de la Prueba
EC 1.	Cargar registro de evento.	V CargaRegistro.	Muestra en una lista el nuevo registro cargado.	Se agregó satisfactoriamente el registro de eventos al sistema.
		I Nulo o archivo incorrecto	Notificación de error.	El sistema notifica al cliente que ha habido un problema mientras se intentaba cargar el registro.

Tabla 14: Caso de prueba EC1: Cargar Registro.

Id del escenario	Escenario	Variable 1 Nombre	Respuesta del Sistema	Resultado de la Prueba
EC 2.	Eliminar registro.	N/A	Muestra nuevamente la lista de registros, esta vez sin el registro eliminado.	Se eliminó correctamente el archivo del servidor.

Tabla 15: Caso de prueba EC2: Eliminar Registro.

Id del escenario	Escenario	Variable 1 Nombre	Respuesta del Sistema	Resultado de la Prueba
EC 3.	Filtrar algoritmos de descubrimiento de procesos.	N/A	Muestra los algoritmos que tienen coincidencias, en el nombre, con el valor filtrado	Se eliminó correctamente el archivo del servidor.

Tabla 16: Caso de prueba EC2: Filtrar algoritmo.

Capítulo 3. Implementación y prueba del sistema

Id del escenario	Escenario	Variable 1 Nombre	Respuesta del Sistema	Resultado de la Prueba
EC 4.	Aplicar Algoritmo de descubrimiento.	N/A	Muestra un mensaje de “Cargando” y posteriormente redirecciona a la vista “Mostrar Red de Petri”.	Cada uno de los algoritmos disponibles fueron aplicados correctamente.

Tabla 17: Aplicar algoritmo de descubrimiento.

Id del escenario	Escenario	Variable 1 Nombre	Respuesta del Sistema	Resultado de la Prueba
EC 5.	Mostrar red de Petri.	N/A	Muestra una vista donde se visualiza una red de Petri y ofrece al usuario la opción de ampliarse por medio del botón “Pantalla Completa”.	La red fue mostrada satisfactoriamente.

Tabla 18: Aplicar algoritmo de descubrimiento.

3.5 Validaciones

Para verificar la correcta estructuración de las redes generadas por el visor implementado en el portlet, sirvió de apoyo el visor que trae la herramienta ProM. Con cada uno de los registros sometidos a prueba, ambos visores generaron

Capítulo 3. Implementación y prueba del sistema

redes bastante parecidas y en ocasiones exactamente iguales.

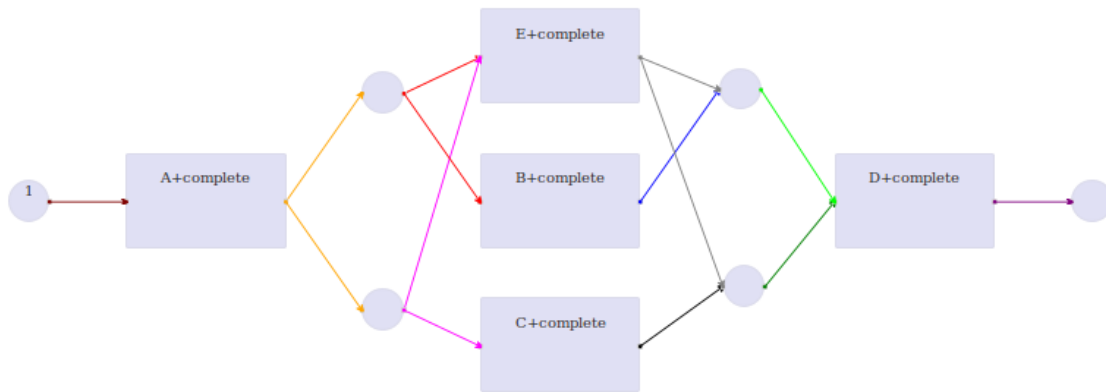


Ilustración 9: red generada por el visor del portlet.

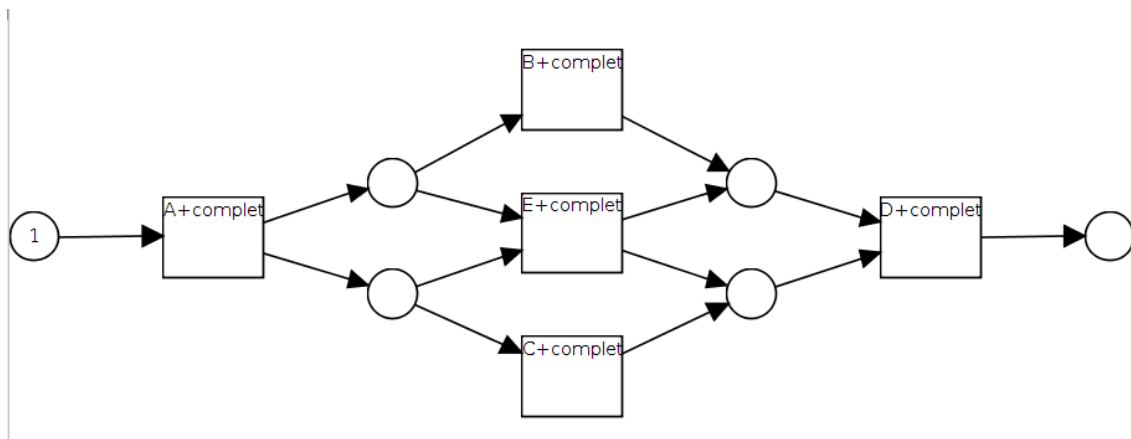


Ilustración 10: red generada por el visor de la herramienta ProM.

3.5.3 Pruebas internas

Las pruebas internas o de caja blanca presenta dos tipos de pruebas principales denominados Prueba de la Ruta Básica y Pruebas de la estructura de control, esta última consta de tres tipos como es la Prueba de condición, Prueba del flujo de datos y Prueba de bucles.

3.5.3.1 Prueba de la Ruta Básica

La prueba de la ruta básica es una Técnica propuesta por McCabe, que permite conocer una medida de la complejidad lógica de un diseño procedural y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Estas garantizan que se ejecute cada instrucción del programa por lo menos una vez durante la prueba (41).

Componentes de la gráfica de flujo:

Capítulo 3. Implementación y prueba del sistema

- **Aristas (E):** caminos a recorrer entre nodos o flujo de control.
- **Nodos (N):** representan una o más sentencias procedimentales.
- **Nodo predicado (P):** nodo del que emanan dos aristas.
- **Región (R):** área que se limitan por aristas y nodos

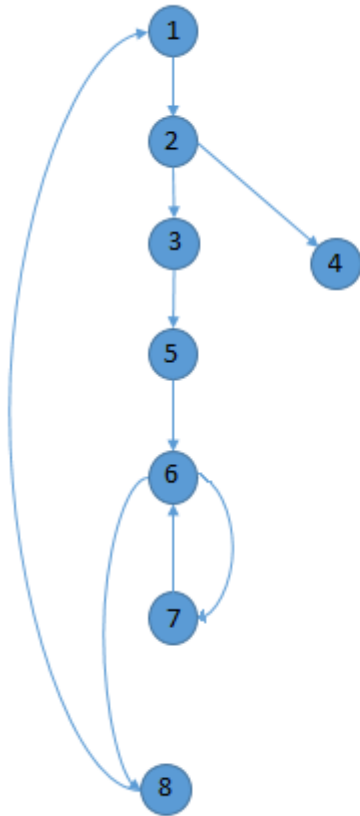
Para la realización del cálculo de la complejidad ciclomática es tomado como muestra un fragmento del código del componente PetriNet específicamente el método “*LlenarListaCapas()*”.

```
215     LlenarListaCapas: function(listCapas) {
216         if (listCapas[listCapas.length - 1][0] === this.finRed()) {
217
218             return listCapas;
219         }
220
221
222         var capaSiguienteTemp = [];
223         var long = listCapas[listCapas.length - 1].length;
224
225         for (var i = 0; i < long; i++) {
226             capaSiguienteTemp.push(this.BuscarTarget(listCapas[listCapas.length - 1][i]));
227         }
228         var capaSiguiente = this.LimpiarLista(capaSiguienteTemp);
229
230         listCapas.push(capaSiguiente);
231         return this.LlenarListaCapas(listCapas);

```

Ilustración 11: Método "LlenarListaCapas()".

Capítulo 3. Implementación y prueba del sistema



Posible conjunto de caminos:
1-2-4.
1-2-3-5-6-8-1-2.
1-2-3-5-6-7-6-8-1-2.

Ilustración 12: "Grafo y Caminos"

La complejidad ciclomática se basa en la teoría gráfica y se calcula de tres maneras:

Capítulo 3. Implementación y prueba del sistema

1. Complejidad ciclomática es igual al número de regiones.

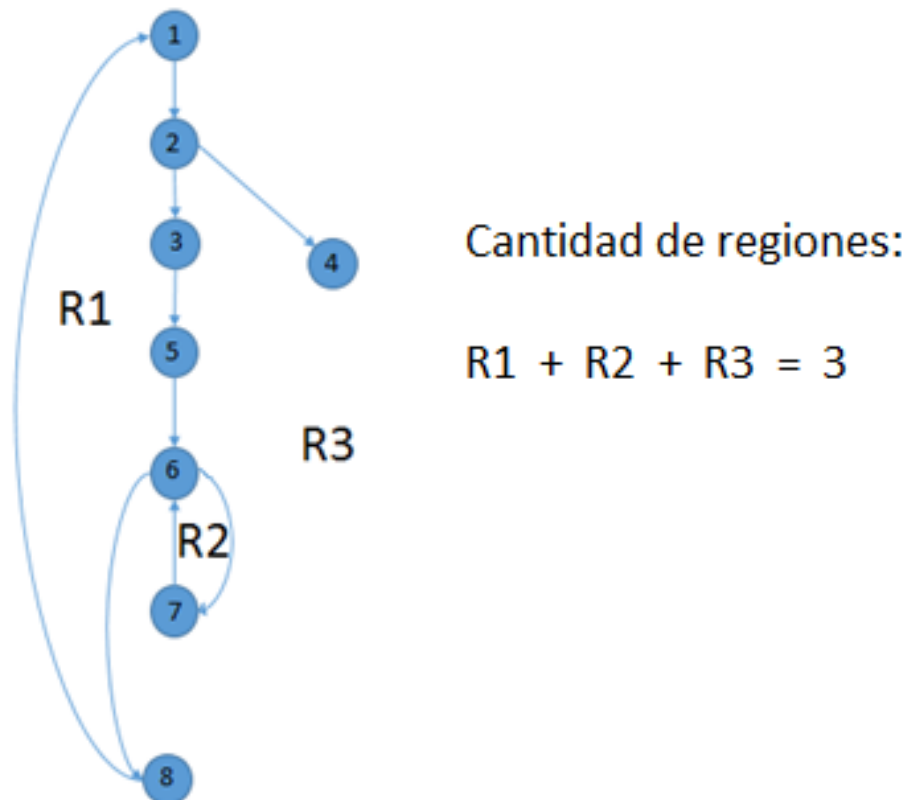


Ilustración 13: Regiones del grafo.

2. Complejidad ciclomática es igual a número de aristas, menos el número de nodos más 2: $V(G) = E - N + 2$.

Solución:

$$V(G) = 9 - 8 + 2 = 3.$$

3. Complejidad ciclomática es igual al número de nodos predicado más uno:

$$V(G) = P + 1.$$

Solución:

$$V(G) = 2 + 1 = 3.$$

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa (42), que permite apreciar la calidad del diseño de software, de una manera rápida y con independencia del tamaño de la aplicación. Esta es utilizada para planificar proyectos de mejora de grandes productos de software y para priorizar las partes del diseño a mejorar (43).

Capítulo 3. Implementación y prueba del sistema

3.6 Lista de chequeo:

Característica de soporte por comprobar.	Ponderación de importancia 1(poco), 5 (fundamental)	Nivel de cumplimiento (0-100)	Justificación
¿El usuario puede agregar o retirar el portlet del portal, según lo requiera, sin afectarse el funcionamiento del portlet?	5	100	Durante el proceso de desarrollo de la propuesta de solución el portlet en cuestión estuvo agregado al portal que ofrece Liferay, en más de una ocasión fue eliminado y vuelto a agregar sin afectar en ningún momento el correcto funcionamiento del portal.
¿El portlet podrá modificarse sin afectar el funcionamiento del portal?	5	100	Esta es precisamente una de las características de un portlet.
El portlet podrá desplegarse en servidores Apache Tomcat 6.0 o superiores?	5	100	El sistema fue probado exitosamente en el Apache Tomcat 6.0 por lo que puede ser desplegado en esta versión o en cualquiera de las superiores.

Tabla 19: Lista de chequeo.

Capítulo 3. Implementación y prueba del sistema

3.7 Resultados

Se realizaron dos iteraciones de las pruebas de caja negra, específicamente con la utilización de casos de pruebas y técnica de partición de equivalencia, para comprobar que la solución funcionara correctamente y si satisfacía los requerimientos del cliente. Durante la primera iteración se encontraron cinco no conformidades, tres de prioridad Baja y dos de prioridad Alta. La primera no conformidad de prioridad Alta fue con respecto al mal funcionamiento del componente en javascript Petri_Net.js, cuando debía visualizar redes de Petri con más de dos finales, el algoritmo recursivo encargado de organizar las capas de la red no llegaba nunca a la condición de parada. La segunda no conformidad de prioridad Alta fue de igual manera con el mismo componente Petri_Net.js que no era capaz de visualizar redes de Petri con más de un inicio, esto ocurría porque la condición de parada del algoritmo que conecta las aristas de la red no fue escogida apropiadamente y las Bajas estuvieron relacionadas con mal funcionamiento de mensajes de confirmación, todas fueron resueltas por el desarrollador de la aplicación. Se realizó una segunda iteración, en la cual no se encontraron no conformidades, evaluándose de esta manera todos los casos de prueba como satisfactorios.

3.8 Conclusiones parciales

- En el capítulo, para concretar la implementación y prueba del sistema, se presentaron los casos de pruebas.
- Las pruebas que se le realizaron al sistema trajeron consigo mejoras en la implementación de las capas de la lógica del negocio y presentación, logrando que el cliente tenga la mejor experiencia posible cuando interactúe con la aplicación.

Conclusiones generales:

- Las herramientas y tecnologías puestas en práctica permitieron el desarrollo de un sistema construido sobre bases sólidas y un entorno bien definido.
- Como resultado del proceso investigativo se obtuvo un portlet que es capaz de ejecutar correctamente tres tipos de algoritmos de descubrimiento de procesos diferentes y mostrar la salida que generan los mismos en forma de una red de Petri.
- Se validó la solución mediante el uso de pruebas de aceptación, a través de prueba de caja negra arrojando como resultado el buen funcionamiento del portlet desarrollado, quedando de esta forma resuelto el problema investigativo.

Recomendaciones

Se propone continuar con el proceso de desarrollo con el objetivo de implementar nuevas funcionalidades que permitan ampliar el repertorio de algoritmos disponibles.

Bibliografía

1. Hernández León, R.A y Coello González, S. El paradigma cuantitativo de la investigación científica. La Habana : s.n., 2012.
2. Idalberto, Chiavenato. Introducci'on a la Teoría General De la Administración, Séptima Edición. s.l. : mac-Graw-Hill Interamericana, 2004.
3. Kenneth Laudon, Jane Laudon. Sistemas de Información Gerencial. México : Pearson Educación, 2008.
4. Robledo, Pedro , y otros. El Libro del BPM. MADRID : Plaza edición, 2014.
5. Antonio, José. Prezi. Prezi. [En línea] 14 de 9 de 2013. [Citado el: 7 de 6 de 2016.] <https://prezi.com/h5a4v88k8iie/proceso-de-negocio/>.
6. Weske, Mathias. Business Process management. 2007.
7. Dpto. de Ingenieria Electrónica, de Sistemas Informáticos y Automática. www.uhu.es. [En línea] Universidad de Huelva, 2014. [Citado el: 17 de Mayo de 2016.] http://www.uhu.es/diego.lopez/AI/auto_trans-tema3.PDF.
8. Sepúlveda, Marcos, Seguel, Ricardo y Seguel, Hugo . Manifiesto sobre Minería de Procesos - versión Español. 2012.
9. Aalst, Wil van der. Process Mining. 2011.
10. Aalst, Wil van der. Process Mining. 2011.
11. van der Aalst, L. Maruster. Workflow Minign: Discovering Proce Model from Events Logs. 2004.
12. van der Alst, A.K Alves de Medeiros. Process Mining with HeuristicsMiner Algorithm. 2006.
13. Lutz, Mark. Learning Python, Fourth Edition. 2010.
14. Terrero , Henry y Paredes, José . Desarrollo de Aplicaciones con Java. s.l. : Fundación de Código Libre Dominicano, 2010.
15. Eckel, Bruce. Thinking in Java. s.l. : Prentice Hall, 2009.
16. Haverbeke, Marijn . Eloquent JavaScript: A Modern Introduction to Programming. 2013.
17. Gauchat, Juan Diego. El gran libro de HTML5, CSS3 y Javascript. 2011.
18. Kendall, Kenneth E. y Kendall, Julie E. Análisis y diseño de sistemas, 8va Edición. 2011.
19. González, LC. publicaciones.uci.cu. publicaciones.uci.cu. [En línea] 15 de 10 de 2012. [Citado el: 10 de 1 de 1016.] publicaciones.uci.cu/index.php/SC/article/download/1032/581.
20. Burd, Barry. Eclipse for Dummies. 2010.

21. Alva, Gustavo Cárdenas. Neoris. [En línea] 6 de 9 de 2010. [Citado el: 17 de 6 de 2016.] http://www.neoris.com/files/White_Paper_SPA_Corporate%20Portals_Gustavo%20Cardenas.pdf.
22. Jonas X., Yuan. Liferay Portal 6 Enterprise Intranets. 2010.
23. Rich, Szov. The Official Guide to Liferay Portal Development . 2011.
24. C., Hernández I. Plataforma de Aprendizaje a Distancia Basada en tecnología de Portlets y Web 2.0 para una Enseñanza Participativa. Madrid : s.n., 2008.
25. IBM. [En línea] 2010. [Citado el: 2 de 5 de 2016.] http://www-01.ibm.com/support/knowledgecenter/SSYJ99_6.1.0/com.ibm.wp.exp.doc_v6101/dev/wpsbpc.html?lang=es..
26. Roberth G. Figueroa, Camilo J. Solís, Armando A. Cabrera. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. 2010.
27. Romero, H. slideShare. slideShare. [En línea] 2012. [Citado el: 30 de 5 de 2016.] es.slideshare.net.
28. Universidad Unión Bolivariana. ingenieriadesoftware.mex. ingenieriadesoftware.mex. [En línea] <http://ingenieriadesoftware.mex.tl/images/18149/METODOLOGIAS%20AGILES.pdf>.
29. Sánchez, Tamara Rodríguez. Metodología de desarrollo para la Actividad productiva de la UCI. 2014.
30. Club Ensayos. Club Ensayos. [En línea] 19 de 10 de 2013. [Citado el: 5 de 4 de 2016.] <https://www.clubensayos.com/Tecnolog%C3%ADa/Metodologia-Aup/1156088.html>.
31. Especificaciones de los requisitos del Software. s.l. : IEEE STD 830-1998.
32. Sommerville, Ian. Software Engineering. 2006.
33. Pressman. Software Engineering . 2001.
34. Jose Manuel Beas. Historias de Usuario. Historias de Usuario. [En línea] 2011. [Citado el: 3 de 3 de 2016.] <http://jmbeas.es/guias/historias-de-usuario/>.
35. Fatima Cham. Introducción al diseño arquitectónico. Introducción al diseño arquitectónico. [En línea] 2013. [Citado el: 10 de 5 de 2016.] <http://es.slideshare.net/lucero6/diseo-arquitectonico-5800290>.
36. de la Torre Llorente, César, y otros. Guía de Arquitectura N-Capas orientada al Dominio con .Net 4.0. España : s.n., 2010.
37. Points, Tutorials. Java Design Patterns. 2015.
38. García Carmona, Juan. GRASP: Controlador. GRASP: Controlador. [En línea] 7 de 9 de 2012. [Citado el: 26 de 11 de 2015.] <http://juan-garcia-carmona.blogspot.com/2012/09/grasp-controlador.html>.
39. ecured. ecured. [En línea] 2008. [Citado el: 6 de 3 de 2016.] http://www.ecured.cu/Reutilizaci%C3%B3n_de_requisitos.
40. es.slideshare.net. es.slideshare.net. [En línea] 10 de 9 de 2013. [Citado el: 7 de 2 de 2016.] <http://es.slideshare.net/PiXeL16/estandares-de-codigo-emanuel>.

41. Juristo, Natalia, Moreno, Ana M. y Vegas, Sira. DOCPLAYER. DOCPLAYER. [En línea] 17 de 10 de 2006. [Citado el: 10 de 1 de 2016.] <http://docplayer.es/6006991-Tecnicas-de-evaluacion-de-software-version-12-0-fecha-17-de-octubre-de-2006-autoras-natalia-juristo-ana-m-moreno-sira-vegas.html>.
42. Blanco Bueno, Carlos. Construcción y Pruebas de Software. Cantabria : s.n.
43. EcuRed. EcuRed. [En línea] [Citado el: 2 de mayo de 2016.] http://www.ecured.cu/Pruebas_de_caja_blanca.
44. Ocampo Acosta, Alejandro y Correa Tapasco, Luisa Marcela. IMPACTO DE LAS PRUEBAS NO FUNCIONALES EN LA MEDICIÓN DE LACALIDAD DEL PRODUCTO SOFTWARE DESARROLLADO. 2011.
45. GARIMELLA, KIRAN y LEES, MICHAEL. BPM (GERENCIA DE PROCESOS). 2014.
46. Grad, Booch y James, Rumbaugh. Unified Modeling Language User Guide. 2005.
47. Modelos Conceptuales. Modelos Conceptuales. [En línea] 2014. [Citado el: 10 de 4 de 2016.] <http://www.buenastareas.com/ensayos/Modelos-Conceptuales/23455.html>.
49. Banerjee, Atanu. Microsoft. Microsoft. [En línea] 12 de 2006. [Citado el: 3 de 2 de 2016.] <https://msdn.microsoft.com/en-us/library/bb220803.aspx>.
50. Ogrinz, Michael. Mashup Patterns: Designs and Examples for the Modern Enterprise. 2009.
51. Alvarez, Miguel Angel. desarrolloweb.com. desarrolloweb.com. [En línea] 2 de 2 de 2014. [Citado el: 10 de 12 de 2015.] <http://www.desarrolloweb.com/articulos/que-es-mvc.html>.
52. Patrón Modelo-Vista-Controlador. Fernandez Romero, Yenisleidy y Días González, Yanette. La Habana : s.n., 2012.
53. Jones, Teresa, Roy Schulte, W. y Cantara, Michele. Magic Quadrant for Intelligent business Process Management Suites. 2014.
54. [En línea] [Citado el: 16 de enero de 2016.] <http://xml.ier.unam.mx/xml/Linux/glinux-2.html>.
55. [En línea] [Citado el: 16 de enero de 2016.] <http://www.monografias.com/trabajos6/sisop/sisop.shtml>.
56. Debian. Debian. [En línea] [Citado el: 16 de enero de 2016.] <https://www.debian.org/releases/stable/armhf/ch01s03.html.es>.
57. EcuRed. EcuRed. [En línea] [Citado el: 2 de mayo de 2016.] http://www.ecured.cu/Niveles_de_prueba_de_software.