Universidad de las Ciencias Informáticas

Facultad 6

Título: Componente para imputar datos en Pentaho Data Integration

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

Autor: Adrian Alberto Thondique Guzman

Tutores: Ing. Adalennis Buchillón Soris

Ing. Yanet Cardoso García

La Habana, Julio, 2015

"Año 57 de la Revolución"

Declaración de autoría

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

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

Adrian Alberto Thondique Guzman

_____

Firma del Autor

Adalennis Buchillón Soris

_____

Firma del Tutor

Yanet Cardoso García

_____

Firma del Tutor

Datos de contacto

Tutores:

Adalennis Buchillón Soris

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email:abuchillon@uci.cu

Yanet Cardoso García

Universidad de las Ciencias Informáticas, La Habana, Cuba

Email: ycardosog@uci.cu

Resumen

El estudio de la información es un proceso importante en el mundo de la informática. Diariamente se generan una gran cantidad de datos, la mayoría de los fenómenos que ocurren en el mundosearchivan para realizarle posteriormente un análisis con el fin de obtener nuevos conocimientos del entorno que nos rodea. La información que es estudiada generalmente no cuenta con la calidad requerida para obtener patrones que describan o generalicen el objeto del estudio.Mucha de esta información cuando se analiza es eliminada u obviada aunque presenten atributos que brinden un mejor entendimiento del problema.

Con el objetivo de aprovechar al máximo los datos, se desarrolló un*plugin*que permite limpiar la informaciónmediante la imputación de valores en campos nulos en laherramienta Pentaho Data Integration. Este *software* se utiliza en la Universidad de las Ciencias Informáticas para realizar integración de datos.El componente fue desarrollado usando algoritmos de *Machine Learning* para predecir el comportamiento de una variable. Se utiliza la librería *weka* para el análisis de datos y Java para desarrollar la aplicación por ser el lenguaje utilizado en la implementacióndel Pentaho Data Integration. Se utilizó OpenUP como metodología de desarrollo y Eclipse como entorno de desarrollo.

Palabras claves:análisis, calidad, imputación, información, *plugin*

Abstract

The studyof information isan important process inthecomputer world. Dailya large amount ofdatais generated, most of the phenomena occurring in the world, are archived for later do atest in orderto gain new knowledgeof the environmentaround us. The information is studied usually does not have the quality required to obtain patterns that describe or generalize the object of study. Much of this information when analyzed is eliminated or overlooked although they have attributes that provide a better understanding of the problem.

In order to maximize data, a plugin that allows you to clean the information by imputation null values in fields in the Pentaho Data Integration a *plugin* was developed. This software is used at the University of Informatics Sciences for data integration.The component was developed using Machine Learning algorithms to predict the behavior of a variable.The weka library for data analysis was used andJava to develop the application as the language used in the implementation of the Pentaho Data Integration. OpenUP was used as a development methodology and Eclipseas a development environment.

Keywords:analysis, imputation, information, quality, plugin

ÍNDICE

Introducción 1

Capítulo 1: Fundamentos teóricos de la investigación 6

1.1 Proceso de Extracción de Conocimiento 6

1.2 Inferencia estadística 7

1.2.1 Estadística descriptiva 8

1.2.2 Estadística inferencial 8

1.3 Imputación de valores 8

1.3.1 Imputación por regresión 9

1.3.2 Imputación por el vecino más cercano 9

1.3.3 Imputación por redes neuronales 9

1.4 Pentaho Data Integration 9

1.4.1 Características de la herramienta PDI: 9

1.4.2 Elementos 10

1.4.3 Herramientas y utilidades: 10

1.5 Herramientas existentes que realizan imputación de datos 11

Data Cleaner 11

Potter´s Wheel 12

AJAX 12

1.6 Metodología de desarrollo 13

1.7 Herramientas y tecnologías utilizadas 16

Conclusiones del capítulo 18

Capítulo 2: Análisis y diseño del componente Imputar Datos 19

Introducción 19

2.1 Modelo de dominio 19

2.2 Propuesta de solución a desarrollar. 20

2.3 Especificación de requisitos 20

2.3.1 Requisitos funcionales 20

2.3.2 Requisitos no funcionales 21

2.4 Definición de los casos de uso 22
2.4.1 Descripción general de los casos de uso 22
2.4.2 Descripción de los actores del sistema 23
2.4.3 Descripción del Caso de Uso Configurar Imputación. 23
2.5 Diagramas de clases de diseño 25
2.6 Patrón arquitectónico 27
2.7 Patrones utilizados 27
2.7.1 Patrones GRASP 27
2.7.2 Patrones GoF 29
Conclusiones del capítulo 30
Capítulo 3 Implementación y pruebas del componente Imputar Datos 31
3.1 Diagrama de componentes 31
3.1.1 Descripción del diagrama de componentes 31
3.2 Consideraciones sobre la implementación 32
3.2.1 Descripción de las funciones asociadas a los ficheros de un plugin para Pentaho Data Integration 32
3.3 Estándar de codificación 34
3.4 Pruebas 35
3.4.1 Formas de probar un software 35
3.4.2 Pruebas de integración 37
3.5 Interfaz del sistema 38
Conclusiones Parciales 39
Conclusiones Generales 40
Recomendaciones 41
Referencias Bibliográficas 42

ÍNDICE DE FIGURAS

Fig. 1 Fases del descubrimiento de conocimientos en bases de datos. 6
Fig. 2 Pentaho Data Integration herramientas y componentes. 11
Fig. 3 Fases de OpenUp. 15
Fig. 4 Diagrama de modelo de dominio 19
Fig. 5 Diagrama de casos de uso del sistema. 22
Fig. 6 Prototipo de interfaz para el componente Imputar Datos. 25
Fig. 7 Prototipo de interfaz de aviso cuando el algoritmo seleccionado no corresponde con el tipo del campo. 25
Fig. 8 Diagrama de clases del diseño del caso de uso Configurar Imputación. 26
Fig. 9 Diagrama de la clase CIDataData 28
Fig. 10 Fragmentos de la inicialización de un objeto. 28
Fig. 11 Patrón GRASP Controlador en el diseño de la clase CIDataMeta. 29
Fig. 12 Diagrama de componentes del caso de uso Configurar Imputación. 31
Fig. 13 Estructura del plugin en Eclipse. 32
Fig. 14 Código XML para la incorporación del plugin. 33
Fig. 15 Ejemplo del estándar de nombres de variables y clases. 34
Fig. 16 Ejemplo del estándar de espacio en blanco. 34
Fig. 17 Resultados de las pruebas de Caja Negra. 37
Fig. 18 Componente para imputar datos en funcionamiento a partir de una transformación. 38
Fig. 19 Estadísticas del flujo de datos a partir de una transformación. 38
Fig. 20 Interfaz del componente Inferir Datos 39

ÍNDICE DE TABLAS

Tabla. 1 Descripción de los conceptos del dominio 19
Tabla. 2 Descripción de los actores del sistema 23
Tabla. 3. Especificación formal de CU Configurar Imputación 23
Tabla. 4 Descripción de las variables del caso de prueba Configurar Imputación 35
Tabla. 5 Caso de prueba Configurar Imputación 36
Tabla. 6 Ejemplo de no conformidades encontradas 37

Introducción

Con el desarrollo de las ciencias y las tecnologías, la información ha adquirido gran valor cuando de ella depende la toma decisiones sobre un aspecto en específico. La información es considerada como el conocimiento que se descubre mediante la observación, la reflexión de manera accidental o mediante un esfuerzo de estudio o de investigación(1).Un uso consciente del conocimiento que puede ser extraído de una fuente de información ayuda a mejorar el desarrollo económico y social de las instituciones.

Como consecuencia del crecimiento exponencial del volumen de la información, se ha hecho indispensable cambiar las formas tradicionales de almacenamiento y análisis de esta. Surgen, como respuesta a esta situación, los almacenes de datos con el propósito de guardar una colección de datos orientada a un determinado ámbito y el proceso de extracción de conocimientos en bases de datos como soporte a la comprensión de la información contenida en los almacenes.

Cuba se encuentra inmersa en un proceso de informatización de los sectores económicos y socialesen el que se necesita adoptar nuevos métodos y tecnologías en aras de mejorar los niveles de productividad de las empresas del país. Para garantizareste proceso existen en la nación varias entidades como: ALBET, DESOFT y SOFTELque se dedican al estudio y desarrollo de aplicaciones y servicios informáticos para servir de soporte a la industria cubana de la informática,como ejemplo de lo anterior expresado se destaca también la Universidad de las Ciencias Informáticas (UCI).

Dentro de esta entidad de altos estudios se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC) que brinda soluciones integrales y servicios especializados en el análisis y almacenamiento de datos. Uno de los departamentos que lo conforma es el de Desarrollo de Componentes. Este a su vez presenta unalínea de producción centrada en el desarrollode proyectos de almacenes de datos y aplicación de técnicas de Inteligencia de Negocio.

En el departamento cuando se analiza una fuente de datos la información pasa por las siguientes fases del proceso de extracción de conocimiento en bases de datos (KDD): recopilación e integración de datos y selección, limpieza y transformación; para aplicarle los procedimientos asociadas a cada fase mencionadalos especialistas del centro se apoyan en la herramienta Pentaho Data Integration(PDI) un poderoso *software* que se encarga del proceso de Extracción transformación y carga (ETL).Como una de

las operaciones que se le realizan a los datos en la primera fase del proceso de KDDse encuentra el perfilado de datos que permite describir el contenido, estructura y calidad de los datos. Una vez que se cuenta con un perfil de la fuente, que debe incluir tipos de datos, clasificación de las variables, existencia de valores perdidos, entre otros elementos, a partir de este conocimiento de la estructura y composición de la fuente es posible aplicar transformaciones a los datos para corregir los problemas de calidad identificados. Específicamente el tratamiento de valores faltantesmediante el uso del PDI pueden ser aplicadas estrategias de limpieza tales como:

1. Ignorar el campo o fila en los casos en que las variables no contienen valor.
2. Usar una constante global para la sustitución como “desconocido”.
3. Rellenar un valor usando la media/desviación del resto de los campos pertenecientes a la misma clase.

Cada una de las estrategias mencionadas permite aplicar tratamiento al problema de la existencia de valores perdidos. Sin embargo se maximizan los errores en la fuente a la que se pretende aplicar técnicas de inteligencia de negocios, lo cual tributa a la obtención de conocimiento poco fidedigno, del cual depende la efectividad de las decisiones tomadas en las organizaciones.

Por los elementos antes expuestos se define como **problema de investigación**¿Cómo predecir losvaloresfaltantes en fuentes de datos procesadas en la herramientaPentaho Data Integration?, definiendo como **objeto de estudio**la imputación de valores faltantes en fuentes de datosprocesadasy como **campo de acción** los métodos de imputación de valores faltantes enfuentes de datosprocesadas. Para dar solución al problema se plantea como **objetivo general** desarrollar un componente para imputar valores faltantes en fuentesde datos en Pentaho Data Integration.

En la siguiente investigación se definen como **objetivos específicos**:

- Analizar los conceptos, tecnologías y herramientas empleados para la imputación de valores faltantes.
- Analizar y diseñar el componente Imputar Datos.
- Implementar el componente Imputar Datos.
- Verificar el correcto funcionamiento del componente Imputar Datos.

Para guiar la lógica de la investigación se formularon las siguientes **preguntas científicas**:

- ¿Cuáles son los fundamentos teóricos del proceso de imputación de datos?
- ¿Cuáles son las características que debe presentar un componente del Pentaho Data Integration para el proceso de imputar datos?
- ¿Cómo estructurar el proceso de desarrollo del componente del Pentaho Data Integration para el proceso de imputar datos?

Para resolver los objetivos planteados y dar respuesta a las preguntas antes planteadas se proponen las siguientes **tareas de la investigación**:

- Análisis de los conceptos asociados al problema de la investigación.
- Selección de las tecnologías, metodología y herramientas a utilizar en el desarrollo del componente Imputar Datos.
- Identificación de los requisitos funcionales y no funcionales para el correcto funcionamiento del componente Imputar Datos.
- Confección del modelo de casos de uso del componente Imputar Datos.
- Elaboración del modelo de diseño del componente Imputar Datos.
- Elaboración del modelo de implementación del componente Imputar Datos.
- Diseño de los casos de prueba del componente Imputar Datos.
- Identificación y resolución de las no conformidades identificadas en la ejecución de las pruebas.

En el proceso de desarrollo de la siguiente investigación se emplean los siguientes **métodos de investigación:**

Métodos teóricos:

Analítico-Sintético: se utiliza para comprenderel funcionamiento de los métodos de imputación y las características generales que estos presentan, también se evidencia al estudiar las diferentes técnicas y

herramientas para el desarrollo de componentes para Pentaho Data Integration y la integración que debe existir entre sus componentes para un correcto funcionamiento de la solución.

Histórico-Lógico: se emplea para obtener conocimiento de las distintas etapas principales de la evolución de un componente de Pentaho Data Integration, esto condujo a la compresión de su lógica, estructura interna e historia de su desarrollo.

Métodos empíricos:

Investigación-Acción: se realizan observaciones con respecto a la situación problemática para luego darle solución según el conocimiento adquirido en la investigación. De esta forma surge la necesidad de realizar un componente para el Pentaho Data Integration que impute datos.

Análisis documental: se realizan un conjunto de operaciones intelectuales con el fin de describir y representar los documentos consultados incluyendo la descripción bibliográfica y general de la fuente; de forma organizada para facilitar su búsqueda cuando sea necesario.

El presente trabajo de diploma está estructurado de la siguiente manera:

- **CAPÍTULO 1. Fundamentos teóricos de la investigación:** En este capítulo se presentan conceptos asociados a los procesos que intervienen en la imputación. Se realiza un estudio de los componentes y el funcionamiento de la herramienta Pentaho Data Integration. Se explica también el porqué de la selección de las herramientas, metodología y tecnologías que se desarrollan en el desarrollo del componente.
- **CAPÍTULO 2. Análisis y diseño del componente Imputar Datos:** En este capítulo se realiza el modelo de dominio para describir los elementos que interviene en el problema. Se describen los requisitos funcionales y no funcionales los que son agrupados posteriormente en casos de uso. Además se definen las clases del diseño que intervienen en el componente así como el patrón arquitectónico y los patrones de diseño aplicados.
- **Capítulo 3. Implementación y prueba del componente Imputar Datos:** En este capítulose describeel proceso de implementación de la solución, así como las características del estándar de

codificación empleado. Se aborda también el proceso de pruebas comprobando que se resolviera lo planteado por los requisitos funcionales.

Posteriormente se presentan las conclusiones generales, recomendaciones, y referencias bibliográficas.

Capítulo 1: Fundamentos teóricos de la investigación

En este capítulo se realizará unestudio del arte de la herramienta Pentaho Data Integration, se describen los principales conceptos referentes al proceso deimputación de datos, además, se explican las herramientas, tecnologías, técnicas y metodología que se van a utilizar para desarrollar la solución.

1.1 Proceso de Extracción de Conocimiento

El proceso de extracción de conocimientos en bases de datos se define como: la extracción no trivial de conocimiento implícito, previamente desconocido y potencialmente útil, a partir de una base de datos (2).El conocimiento que se adquiere es de suma importancia para decidir si es necesario cambiar la proyección que tenga la organización sobre el objetivo que se estudió.Para arribar a cualquier conclusión mediante el KDD la información recién recopilada necesita transitar por una serie de etapas como se muestraen la Fig. 1.

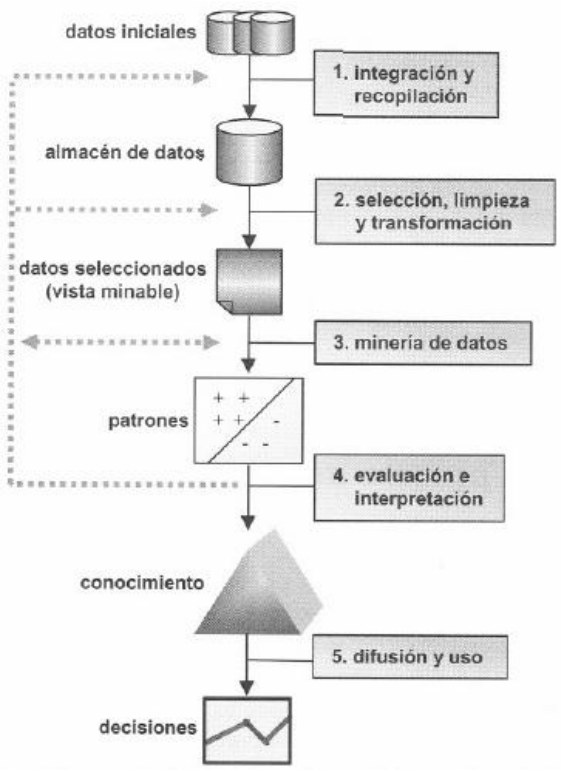

Fig. 1 Fases del descubrimiento de conocimientos en bases de datos.

Integración y recopilación de datos:se determinan las fuentes de información que pueden ser útiles y dónde conseguirlas. Seguidamente se transforman todos los datos a un formato común, frecuentemente mediante un almacén de datos que consiga unificar de manera operativa toda la información recogida, detectando y resolviendo inconsistencias. Este almacén de datos facilita la navegación y visualización previa de los datos.

Fase de selección, limpieza y transformación: en esta fase se eliminan o corrigen los datos incorrectos y se decide la estrategia a seguir con los datos incompletos. Además, se proyectan los datos para considerar únicamente aquellas variables o atributos que van a ser relevantes, con el objetivo de hacer más fácil la tarea propia de minería de datos y para que los resultados de la misma sean útiles. La selección incluye tanto una fusión horizontal (filas/registros) como vertical (columnas/atributos).

Minería de datos: la minería de datos permite establecer procesos de forma automatizada para la obtención de información a partir de grandes cantidades de datos. Esta es la etapa de descubrimiento en el proceso de KDD mediante el uso de algoritmos concretos que generan una enumeración de patrones a partir de los datos preprocesados(3).

Interpretación y evaluación: se evalúan los patrones y se analizan por los expertos y si es necesario se vuelve a las fases anteriores para una nueva iteración. Esto posibilita resolver posibles conflictos con el conocimiento que se disponía anteriormente.

Difusión: por último se hace uso del nuevo conocimiento y se hace partícipe de él a todos los posibles usuarios.(4)

Después de analizar las fases que forman parte del proceso de extracción de conocimientos se concluyó que el *plugin* a desarrollar estaría enmarcado en la fase de selección, limpieza y transformación teniendo en cuenta su objetivo de modificar el flujo de datos con el fin de mejorar la calidad de estos.

1.2 Inferencia estadística

El estudio de la estadística es empleado en el mundo real en disímiles campos como la salud, la industria, los negocios entre otros; esta disciplina ayuda a diseñar esquemas y registrar información para describirla y analizarla con facilidad. Como resultado del estudio se obtienen conclusiones que mejoran el conocimiento de la realidad. Cuando se habla de Estadística se refiere al "*estudio científico relativo a*

unconjunto de métodos encaminados a la obtención, representación y análisis de observaciones numéricas, con el fin de describir la colección de datos obtenidos, así como inferir generalizaciones acerca de las características de todas las observaciones y tomar las decisiones más acertadas en el campo de su aplicación". La estadística como disciplina incluye la observación y el tratamiento de datos numéricos así como el empleo de los datos estadísticos con fines inferenciales dividiéndose en dos categorías fundamentales la estadística descriptiva y la estadística inferencial. (5)

1.2.1 Estadística descriptiva

"*La estadística descriptiva es el estudio que incluye la obtención, organización, presentación y descripción de la información*". Como su concepto lo indica esta categoría de la estadística está centrada en la calidad del manejo de la información para que sea fácil su lectura e interpretación (5).

1.2.2 Estadística inferencial

"*La inferencia estadística es una técnica mediante la cual se obtienen generalizaciones o se toman decisiones en base a una información parcial o completa obtenida mediante técnicas descriptivas*". Este tipo de estadística tiene como objetivo la obtención de patrones de los datos lo que trae consigo que se pueda predecir un comportamiento futuro de estos. (5)

De las dos clasificaciones antes planteadas se utilizó la estadística inferencial como base para dar solución al problema planteado. El uso de esta categoría ayudará a entender cómo se comportan las características del fenómeno a estudiar.

1.3 Imputación de valores

Se denomina imputación al procedimiento que utiliza la información contenida en la muestra para asignar un valor a aquellas variables cuyo valor no está registrado, el objetivo que persigue este proceso es la obtención de un conjunto de información completa y consistente a la que se le puedan aplicar técnicas estadísticas.Atendiendo a la estrategia de asignación de valores que presentan los distintos métodos de imputación, se pueden clasificar en: simple cuando se asigna un valor por cada valor faltante basándose en el valor de la propia variable o de otras variables para generar una base de datos completa o múltiple si se asignan a cada valor faltante varios valores (m), generando *m* conjuntos de datos completos. (6)

Para la confección del *plugin* se utilizaron tres métodos de imputación simple porque estos ofrecen un mejor trabajo con los datos, ya que se predice un valor paracada valor perdido. Este tipo de imputación no

necesita de mucho requerimiento de cómputo o almacenamiento para crear la base de datos imputada y produce una única respuesta por lo que el proceso no necesita de la supervisión de un investigador.

1.3.1 Imputación por regresión

Se emplean modelos de regresión para imputar información en la variable *Y*, a partir de covariables relacionadas con *Y*. Este procedimiento consiste en eliminar las observaciones con datos incompletos y ajustar la ecuación de la regresión para predecir los valores faltantes. Dependiendo del comportamiento de la variable que se va a imputar se obtiene un modelo de regresión u otro. (6)

1.3.2 Imputación por el vecino más cercano

En este método se basa en la suposición de que los individuos cercanos en un mismo espacio tienen características similares. Se identifica la distancia entre la variable a imputar y cada una de las unidades restantes de esta forma se usa el valor que presenta un vecino para imputar el dato faltante. (6)

1.3.3 Imputación por redes neuronales

Son sistemas de información que reconocen patrones de los datos sin algún valor perdido para aplicarlos a los datos a imputar. Se entrena el sistema proporcionándoles datos que no presenten anomalías cuando esta fase concluye, el sistema al recibir un valor faltante es capaz de predecir un valor apoyándose en su base de conocimientos. (6)

1.4 Pentaho Data Integration

Pentaho Data Integration es una herramienta que se encarga de realizar el proceso de ETL son las siglas en inglés de extraer, transformar y cargar. Es un proceso que permite mover datos desde múltiples fuentes, reformatearlos, limpiarlos y cargarlos en otra base de datos o almacén de datos para un posterior análisis. A continuación se describen las características de la herramienta PDI con el objetivo de entender su funcionamiento. (7)

1.4.1 Características de la herramienta PDI:

- Instalación fácil, interfaz gráfica sencilla.
- El sistema es multiplataforma Windows, Macintosh, GNU/Linux.
- Le brinda al usuario la flexibilidad de llegar a una solución utilizando diferentes estrategias.
- Arquitectura adaptable para entender su funcionalidad.

- Desarrollado en java.
- Uso de lenguajes como XML, JavaScript, SQL.
- Permite agregarle nuevas funcionalidades.
- Los pasos como los trabajos son implementados por separado del sistema.
- Los *plugins* se pueden cargar de forma dinámica sin tener que recompilar el núcleo del sistema.

Pentaho Data Integration está formado por un motor de integración de datos que es capaz de interpretar y ejecutar trabajos y transformaciones. Este *software* también ofrece varias herramientas y utilidades para crear, manejar y cargar transformaciones y trabajos. (7)

1.4.2 Elementos

El Pentaho Data Integration está construido sobre dos tipos elementos: las transformaciones y los trabajos a continuación una breve explicación de estos:

Transformación: se compone de varios pasos, que están enlazados entre sí a través de saltos. Los pasos son el elemento más pequeño dentro de las transformaciones. Los saltos constituyen el elemento a través del cual fluye la información entre los diferentes pasos (siempre es la salida de un paso y la entrada de otro). La herramienta cuenta con varios tipos de pasos implementados que permiten satisfacer necesidades en el diseño del proceso de integración de datos.

Trabajos:Un trabajo es un componente que crea una secuencia de actividades que brinda un orden de ejecución (no empieza la ejecución del elemento siguiente hasta que el anterior no ha concluido) por lo que se usan para controlar el flujo y generalmente consisten en una serie de transformaciones.

1.4.3 Herramientas y utilidades:

- *Spoon*: Es un entorno de desarrollo integrado gráfico de integración de datos que sirve para crear transformaciones y trabajos.
- *Kitchen*: Es una herramienta de consola para ejecutar trabajos.
- *Pan*: Es una herramienta de consola para ejecutar transformaciones.
- *Carte*: Es un servidor ligero que permite ejecutar trabajos y transformaciones en un servidor remoto.

En la Fig. 2 se muestra la interacción entre los componentes, herramientas y utilidades del PDI.

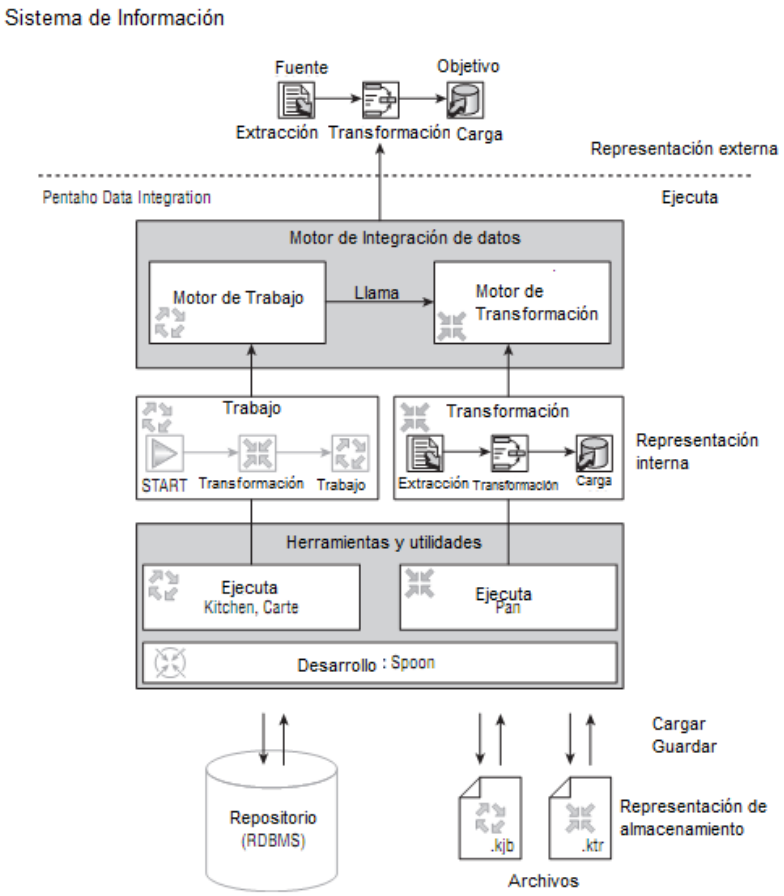

Fig. 2 Pentaho Data Integration herramientas y componentes.

1.5 Herramientas existentes que realizan imputación de datos

En la actualidad existen programas que tienen como fin común la limpieza de datos utilizando técnicas de inferencia e imputación de valores, en la presente investigación se estudiaron los más significativos con el fin de identificar las principales características como los tipos de errores que estas corrigen, la compatibilidad con los gestores de base de datos y el tipo de licencia con que está registrado.Ejemplo de herramientas utilizadas para garantizar la calidad de los datos que se estudiaron son las siguientes:

Data Cleaner: su principal función es realizar perfiles de datos para luego descubrir y analizar la calidad de los mismos.Encuentra patrones, valores perdidos, juegos de caracteres y otras características de los valores de los datos, además,se le introdujo la detección de duplicados de última generación construido basándose en los principios de aprendizaje automático e igualdad inferencial, se pueden trabajar condatos

procedentes de hojas de cálculos, archivos CSV (*TheCommaSeparatedValue*) y bases de datos no relacionales con el fin de estandarizarlo para crear reglas de limpieza de datos dependiendo del escenario en que se desarrolle el estudio.Data Cleaner es una herramienta que se puede integrar al Pentaho Data Integration pero presenta el inconveniente que no le brinda al usuario la opción de cambiar la información del flujo de datos. (8)

Potter´s Wheel: es un sistema interactivo de limpieza de datos que integra la transformación de datos y la detección de errores usando una hoja de cálculo como interfaz. Los efectos de las operaciones desarrolladas son mostrados inmediatamente en campos visibles en pantalla. La detección de errores es hecha para la colección de datos completa de forma automática como un proceso de fondo. Un grupo de operaciones son especificadas y soportan los esquemas de transformaciones comunes sin una programación explícita. Las especificaciones para el proceso de limpieza de datos son hechas de forma interactiva, además de que la retroalimentación inmediata de las transformaciones llevadas a cabo y las detecciones de errores permiten a los usuarios un desarrollo gradual y pulir el proceso.(9)

***AJAX*:**es un *framework* flexible y extensible que intenta separar los niveles lógicos y físicos de la limpieza de datos. El nivel lógico sustenta el diseño del flujo de trabajo de la limpieza de datos y la especificación de las operaciones de limpieza desarrolladas, mientras que en el nivel físico solo está definida su implementación. El mayor interés de AJAX es transformar los datos existentes de una o más colecciones de datos en un esquema destino y eliminar los duplicados dentro de este proceso. Para este propósito, se define un lenguaje declarativo basado en un grupo de operaciones de transformación, las cuales son: asignación (*mapping*), visión (*view*), correspondencia (*matching*), agrupación (*clustering*) y fusión (*merging*).(10)

Las herramientas AJAX y Potter´s Wheel no cuentan con versiones que se puedan integrar al PDI por lo que los especialistas estarían obligados exportar los datos de las trasformaciones fuera de PDI para realizarle el proceso de imputación y luegoimpórtalos de nuevo a la herramienta de integración utilizada, lo que ralentizaría el proceso de ETL. En el caso de DataCleaner no tiene implementado la modificación de los valores perdidos. Finalmente se optó por el desarrollo de un componente para imputar datos que se adapte a las necesidades del departamento de Desarrollo de Componentes.

1.6 Metodología de desarrollo

Para asegurar un buen desarrollo de *software,* altas probabilidades de éxito y una mejor organización del trabajo a realizar, es necesario el uso de un conjunto de técnicas y métodos que permitan describir de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto.Según la envergadura del *plugin* que se va a desarrollar y la filosofía de desarrollo.

Metodologías ágiles

Este concepto surge a principios de la década de los 90 como un enfoque revolucionario para su momento ya que se oponía a toda creencia de que mediante procesos altamente definidos se iba a lograr obtener un programa informático en tiempo, costo y con la requerida calidad. Dentro del desarrollo ágil se encuentran un grupo de metodologías de desarrollo de *software* que promueven generalmente un proceso de gestión de proyectos que fomenta el trabajo en equipo, la organización y responsabilidad propia, que permiten la entrega rápida de una solución de alta calidad según las necesidades del cliente y los objetivos de la compañía asociada a este.

Las metodologías ágiles se rigen por los siguientes principios:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de *software* que le aporte un valor.
2. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
3. Entregar frecuentemente *software* que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4. Las personas que intervienen en el negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El *software* que funciona es la medida principal de progreso.

8. Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento. (11)

OpenUp

Es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. Por lo tanto no provee directrices para gestionar todos los elementos que se manejan en un proyecto, pero tiene los componentes básicos que pueden servir de base a procesos específicos. La mayoría de los elementos están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto atendiendo a sus objetivos, alcance y avances; mantiene las características de *Rational Unified Process*(RUP)como es el desarrollo incremental,la utilización de casos de uso y escenarios así, como presentar un diseño basado en la arquitectura. (12)

Principios de la metodología OpenUp

- Colaborar para sincronizar intereses y compartir conocimiento.
- Equilibrar las prioridades para maximizar el beneficio obtenido por los interesados en el proyecto.
- Centrar la arquitectura tempranamente para minimizar riesgos y organizar el desarrollo.
- Mantener un desarrollo evolutivo para obtener retroalimentación y mejoramiento continuo.

Fases de la metodología OpenUp

El ciclo de vida de un proyecto en el que se use la metodología OpenUp cuenta con cuatro fases y cada una de estas fases se divide a su vez en iteraciones como se muestra en la Fig. 3.

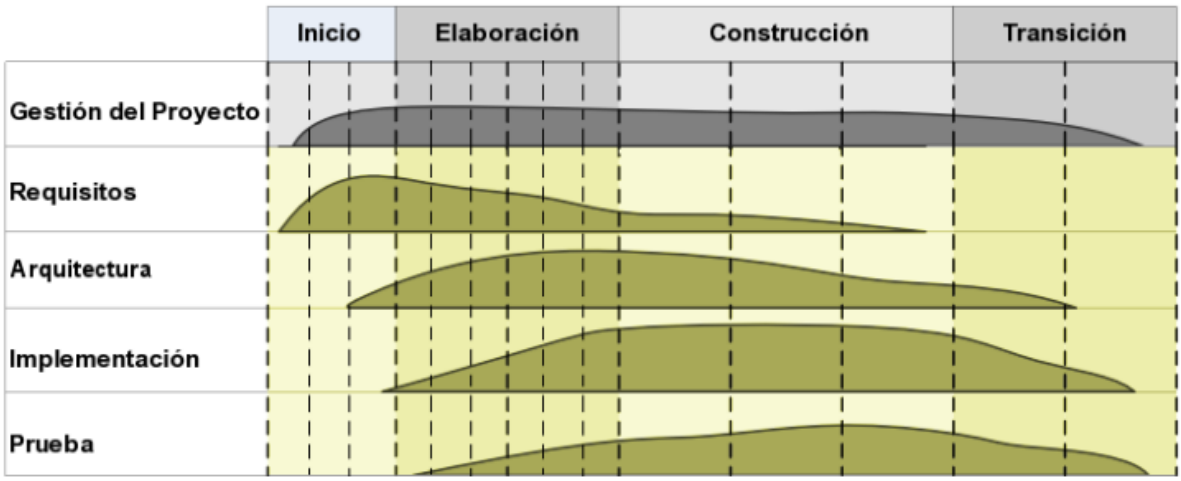

Fig. 3Fases de OpenUp.

Fase de inicio: En esta faselas necesidades de cada participante del proyecto son tomadas en cuenta y plasmadas en los objetivos del proyecto. Se definen para el proyecto: el ámbito, los límites, el criterio de aceptación, los casos de uso críticos, una estimación inicial del coste y un boceto de la planificación.

Fase de elaboración: En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Se debe elaborar un plan de proyecto, estableciendo los requisitos y una arquitectura estables. Por otro lado, el proceso de desarrollo, las herramientas, la infraestructura a utilizar y el entorno de desarrollo también se especifican en detalle en esta fase. Al final de la fase se debe tener una definición clara y precisa de los casos de uso, los actores, la arquitectura del sistema y un prototipo ejecutable de la misma.

Fase de construcción: Todos los componentes y funcionalidades del sistema que falten por implementarson realizados, probados e integrados en esta fase. Los resultados obtenidos en forma deincrementos ejecutables deben ser desarrollados de la forma más rápida posible sin dejar de lado lacalidad de lo desarrollado.

Fase de transición: Esta fase corresponde a la introducción del producto en la comunidad de usuarios,cuando el producto está lo suficientemente maduro. La fase de transición consta de las subfasesde pruebas de versiones beta, pilotaje y capacitación de los usuarios finales y de los

encargadosdel mantenimiento del sistema. En función de la respuesta obtenida por los usuarios puede sernecesario realizar cambios en las entregas finales o implementar alguna funcionalidad más. (13)

OpenUpes apropiado para proyectos pequeños y de bajos recursos el uso de esta metodología permite disminuir las probabilidades de fracaso e incrementar las probabilidades de éxito.Otro aspecto importante es que permite detectar errores tempranos en el desarrollo del *software*, otra característica es que evita la elaboración de documentación, diagramas e iteraciones innecesarias y al ser una metodología ágil tiene un enfoque dirigido al cliente por lo que se decidió su uso para el desarrollo del *plugin* para Pentaho Data Integration.

1.7 Herramientas y tecnologías utilizadas

Lenguaje Unificado de Modelado v2.0 (UML)

El lenguaje unificado de modelado permite visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir el *software* incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación,esquemas de bases de datos. Se puede aplicar en el desarrollo de *software* para dar soporte a la metodología escogida para el desarrollo, no es un lenguaje de programación sino que sirve para el modelado completo de sistemas complejos tanto en el diseño de los sistemas de *software* como para la arquitectura de *hardware* donde se ejecuten.Este lenguaje es utilizado en la fase de análisis y diseño del componente para realizar modelos conceptuales, diagramas de paquetes, diagramas de clases, diseño de interfaces y modelos de diseño de la base de datos. (14)

Herramienta CASE

Las herramientas*Computer Aided Software Engineering*(CASE) tienen como objetivo ser el soporte para el ingeniero de *software* en el ciclo de desarrollo de este.Su uso le facilita la verificación y mantenimiento de la consistencia de la información del proyecto medianteel establecimiento de estándares en el proceso de desarrollo y documentación.Ofrece tambiénvarias funciones automatizadas que brindan una visión de cómo quedará el producto. (15)

Visual Paradigm para UML v8.0

Visual Paradigm para UML (VP-UML) es una herramienta CASE que tiene como objetivofacilitar a los diseñadores de *software* la creación de modelos conceptuales de una aplicación. Ofrece un conjunto de herramientas de los equipos de desarrollode *software* necesario para la captura de requisitos, la planificación de controles, el modelado de clases y el modelado de los datos. Es multiplataforma y está concebida para soportar el ciclo de vida completode una aplicación informática.

Considerando todas las facilidades brindadas porel Visual Paradigm para UML se decide escoger esta herramienta para el modelar el ciclo de desarrollo de *software*aparte de contar con una licencia gratuita y otra comercial, además de ser utilizada en los centros de desarrollo de la UCI y contar con bastante información sobre ella. (16)

Lenguaje de programación Java v1.8

Java es un lenguaje de programación orientado a objetos, su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo;de manera que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de Java son generalmente compiladas a *bytecode* que puede ejecutarse en cualquier máquina virtual java sin importar la arquitectura de la computadora subyacente. Con la utilización de este lenguaje se pueden crear *applets*, aplicaciones, manipuladores de protocolo, manipuladores de contenido entre otros productos. (17)

Entorno de desarrollo integrado

Un entorno de desarrollo integrado llamado también *IntegratedDevelopmentEnviroment* (IDE), es un programa informático compuesto por un conjunto de herramientas de programación. Un IDE es un entorno de programación que contiene un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). (18)

Plataforma de desarrollo Eclipse Luna *Service-Release* 1 v4.4.1

Es una herramienta de desarrollo multiplataforma, diseñada para ser extendida de forma indefinida a través de *plugins*. Eclipse cuenta con una plataforma de cliente enriquecida por lo que cuenta con una

estructura principal donde se ejecutan los *plugins*que posee, también cuenta con un *widget* para Java llamado (SWT)*StandardWidgetToolkit* que ayuda con la interfaz visual. Permite el desarrollo de *software* utilizando otros lenguajes de programación como C/C++ y Python y lenguaje para procesado de texto como Latex. En cuanto a las aplicaciones clientes le brinda al programador marcos de trabajos útiles para el trabajo con aplicaciones gráficas con el uso de GEF (*GraphicEditingFramework*). Fue liberado bajo la licencia Eclipse Public License (Licencia Pública de Eclipse), los receptores de esta herramienta pueden utilizar, modificar, copiar y distribuir el trabajo y las versiones modificadas.(19)

WEKA v3.6.10

Waikato Enviroment for Knowledge(WEKA) es una herramienta escrita en Java que permite la experimentación de estudio de los datos mediante la aplicación, análisis y evaluación de las técnicas más relevantes de análisis de datos, principalmente las provenientes del aprendizaje automático sobre cualquier conjunto de información. El *software* contiene varias herramientas de visualización y algoritmos unidos a interfaz gráfica que provee 4 flujos de trabajo*SimpleCLI*, *Explorer*, *Experimenter* y *KnowledgeFlow*. Se distribuye bajo la licencia GNU *GeneralPublicLicence*(GNU-GPL) por lo que es un *software* libre. WEKA también puede incorporarse como librería a proyectos desarrollados en *Java*ya que todos sus componentes se encuentran dentro de un archivo con extensión *jar*. (20)

Conclusiones del capítulo

A partir de las necesidades que presenta el departamento de Componentes y del estudio realizado de la herramienta Pentaho Data Integration y los conceptos asociados al proceso de imputación de datos, se determinó la creación de un *plugin* que impute datos. Para la creación del *plugin* se definió como metodología de desarrollo OpenUp porque sus principios pueden ser aplicados al entorno de elaboración del componente. Se analizaron las herramientas y tecnologías se decidió el uso del Visual Paradigm para el modelado del lenguaje UML y como IDE Eclipse por su estrecho vínculo con el lenguaje de programación Java.

Capítulo 2: Análisis y diseño del componente ImputarDatos

Introducción

En el presente capítulo se analiza y diseña la solución propuesta. Se define el modelo de dominio al que pertenece la aplicación, se identifican los requisitos funcionales y no funcionales, los casos de uso con sus descripciones textuales. Se abordará también el uso de los patrones tanto arquitectónicos como de diseño.

2.1 Modelo de dominio

El modelo de dominio se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos de *software.* Presenta a los modeladores las clases conceptuales significativas en el contexto donde se desarrolla el problema; considerándose uno de los artefactos de gran importancia, creado durante el análisis orientado a objetos (21).

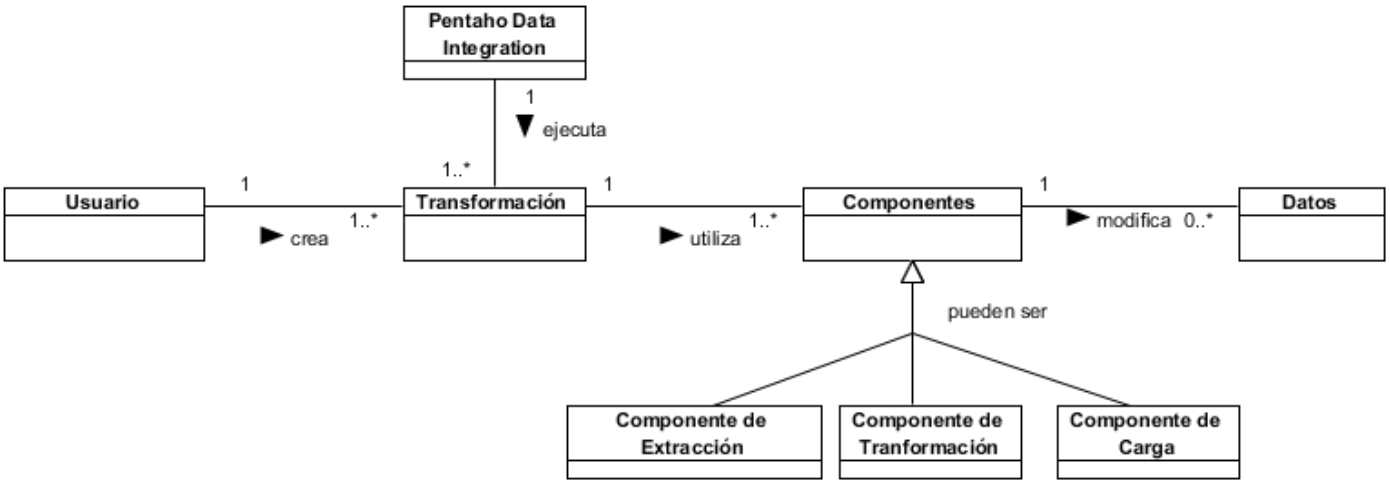

Fig. 4Diagrama de modelo de dominio

Los conceptos representados en el modelo de dominio de la Fig. 4 se describen a continuación:

Tabla.1 Descripción de los conceptos del dominio

Concepto del dominio	Descripción
Usuario	Persona encargada de realizar el proceso de ETL
Pentaho Data Integration	Herramienta encargada de realizar el proceso de ETL
Componentes	Conjunto de funcionalidades enfocadas a obtener un resultado específico

Transformación	Flujo de trabajo del PDI que contiene un conjunto de componentes
Datos	Información con la que interactúa el componente
Componente de Extracción	Generalización de un componente que se encarga de la entrada de datos al PDI
Componente de Transformación	Generalización de un componente que se encarga de realizar modificaciones en los datos en el PDI
Componente de Carga	Generalización de un componente que se encarga de la salida de los datos del PDI

2.2 Propuesta de solución a desarrollar.

Una vez realizada la descripción del modelo de domino del negocio y analizada la situación problemática existente; se propone la confección de un componente para la herramienta Pentaho Data Integration que sea capaz de detectar un campo vacío y modificar este valor con una aproximación calculada mediante el uso de algoritmos de imputación.

2.3 Especificación de requisitos

Los requisitos son capacidades y condiciones con las cuales debe ser conforme el sistema a desarrollar. El primer reto del trabajo de los requisitos es encontrar, comunicar y recordar lo que se necesita realmente, de manera que tenga un significado claro para el cliente y los miembros del equipo de desarrollo. (21)

2.3.1 Requisitos funcionales

Los requisitos funcionales (RF) son los que definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas. Su principal objetivo es que se describa el ¿Qué? y no el ¿Cómo? se deben hacer las transformaciones. Con el avance del tiempo estos requerimientos se convierten en los algoritmos, la lógica y gran parte del código del sistema a desarrollar (22).

Para el desarrollo de la solución se determinaron seisRF, que se definen a continuación:

RF1.Obtener campos de una tabla a partir de componentes que tengan asociados una entrada.

El sistema debe permitir al usuarioobtener los campos que forman parte del flujo de datos.

RF2. Listar algoritmos de imputación de datos.

El sistema debe permitir al usuario visualizar la lista de algoritmos de imputación disponibles.

RF3.Seleccionar algoritmos de imputación de datos.

El sistema debe permitir al usuario elegir un algoritmo de imputación.

RF4.Seleccionar imputación de datos por cada columna.

El sistema debe permitir al usuario elegir si se imputan los datos en una columna.

RF 5: Guardar configuración.

El sistema debe permitir al usuario guardar la configuración de las opciones del *plugin*.

RF 6: Imputar datos por algoritmo seleccionado.

El sistema debe permitir al usuarioimputar datos utilizando el algoritmo seleccionado.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no se refieren directamente a las funcionesespecíficas que proporciona el sistema, sino a las propiedades que deberá poseer o a las restricciones del mismo (23).

Para el desarrollo de la solución se determinaron nueveRNF, estos se describen a continuación:

Apariencia o interfaz externa

RNF 1.La interfaz externa debe estar diseñada de acuerdo con la estructura que presentan todos los *plugins* del Pentaho Data Integration.

Usabilidad

RNF 2. Los elementos que componen la interfaz deben poseer mensajes contextuales que muestren información de su función dentro del *plugin*.

RNF 3. Facilidad de uso por parte de los usuarios:La interfaz debe ser descriptiva, permitiendo que el usuario puede entender las operaciones que realiza dentro del *plugin*.

RNF 4. Ante la ocurrencia de un error, el *plugin* le brinda información al usuario del origen de este.

Software

RNF 5. Se requiere para el funcionamiento del Pentaho Data Integration disponer de la Máquina Virtual de Javav1.6 o superior.

RNF 6. Se requiere para el funcionamiento del*plugin* agregarle a la herramienta Pentaho Data Integration la librería weka.jar en su versión v3.6.

Hardware

RNF 7. Para la ejecución de la aplicación se necesitará: un procesador Celeron 2.0GHz o superior, 1 Gygabytes (Gb) de memoria RAM como mínimo y 1 Gb al menos de espacio libre en el disco duro.

Portabilidad

RNF 8. El *plugin* una vez integrado a la herramienta Pentaho Data Integration, se podrá utilizar en diferentes sistemas operativos por ser Pentaho Data Integration multiplataforma.

Eficiencia

RNF 9. El tiempo de ejecución de los análisis será proporcional al volumen de los datos a analizar.

2.4 Definición de los casos de uso

Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto (24).

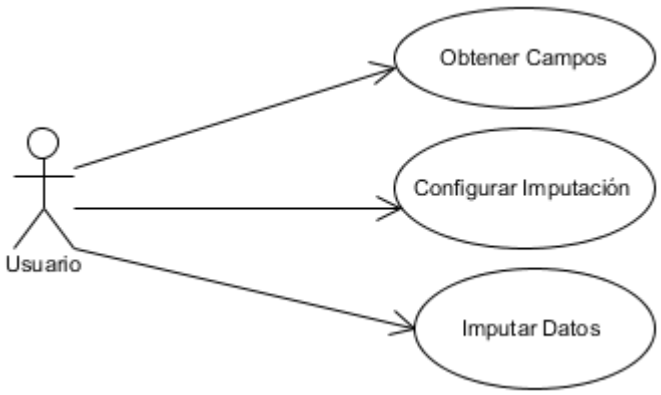

Fig. 5Diagrama de casos de uso del sistema.

2.4.1 Descripción general de los casos de uso

CU Obtener Campos

El caso de uso Obtener Campos cuenta con la funcionalidad que permite conocer los campos que llegan al componente por el flujo de datos. Este caso de uso satisface al requisito funcional siguiente:

- RF 1.Obtener campos de una tabla a partir de componentes que tengan asociados una entrada.

CU Configurar Imputación

Configurar Imputación constituye el caso de uso crítico del *plugin.* Contiene las funcionalidades que permiten la selección de las filas que serán modificadas y el algoritmo asociado a cada una de ellasademás de guardar las opciones con las que se realizará la imputación de valores. A continuación se enumeran los requisitos funcionales que engloba este caso de uso.

- RF 2.Listar algoritmos de imputación de datos.
- RF 3.Seleccionar algoritmos de imputación de datos.
- RF 4.Seleccionar imputación de datos por cada columna.
- RF 5.Guardar configuración.

CU Imputar Datos

El caso de uso Imputar Datos presenta la funcionalidad que permite realizar la imputación de valores en Pentaho Data Integration. Ese caso de uso está asociado al requisito funcional siguiente:

- RF 6.Imputar datos por algoritmo seleccionado.

2.4.2 Descripción de los actores del sistema

Tabla.2 Descripción de los actores del sistema

Actor	Descripción
Usuario	Persona que interactúa con el componente

2.4.3 Descripción del Caso de UsoConfigurar Imputación.

Tabla.3. Especificación formal de CU Configurar Imputación

Caso de Uso	Configurar Imputación
Objetivo	Establecer las opciones que utilizará el *plugin*para realizar la imputación de datos
Actores	Usuario
Resumen	El caso de uso se inicia cuando el usuario abre las opciones del componente mediante doble clic sobre la imagen del componente en el área de trabajo del Pentaho
Referencias	RF 2,RF 3, RF 4,RF 5
Complejidad	Alta
Prioridad	Alta

Precondiciones	El componente debe encontrarse en el área de trabajo del Pentaho Data Integration
Postcondiciones	El componente deberá guardar la configuración seleccionada por el usuario
Flujo Normal de eventos	
Flujo Básico "Imputar datos".	
Acción Actor	**Respuesta del Sistema**
1. El usuario abre las opciones del componente	2. El sistema muestra la interfaz con las opciones que tiene por defecto el componente
3. El usuario selecciona la opción "Obtener campos"	4. El sistema muestra los campos que llegan al componente por el flujo de datos
5. El usuario selecciona el algoritmo de imputaciónpara cada campo y selecciona la opción "OK"	6. El sistema verifica que el algoritmo esté de acuerdo con el tipo de dato del campo.
	7. El sistema guarda la configuración seleccionada por el usuario
Requisitos no funcionales	RNF 2, RNF 3, RNF 4

Prototipo de Interfaz

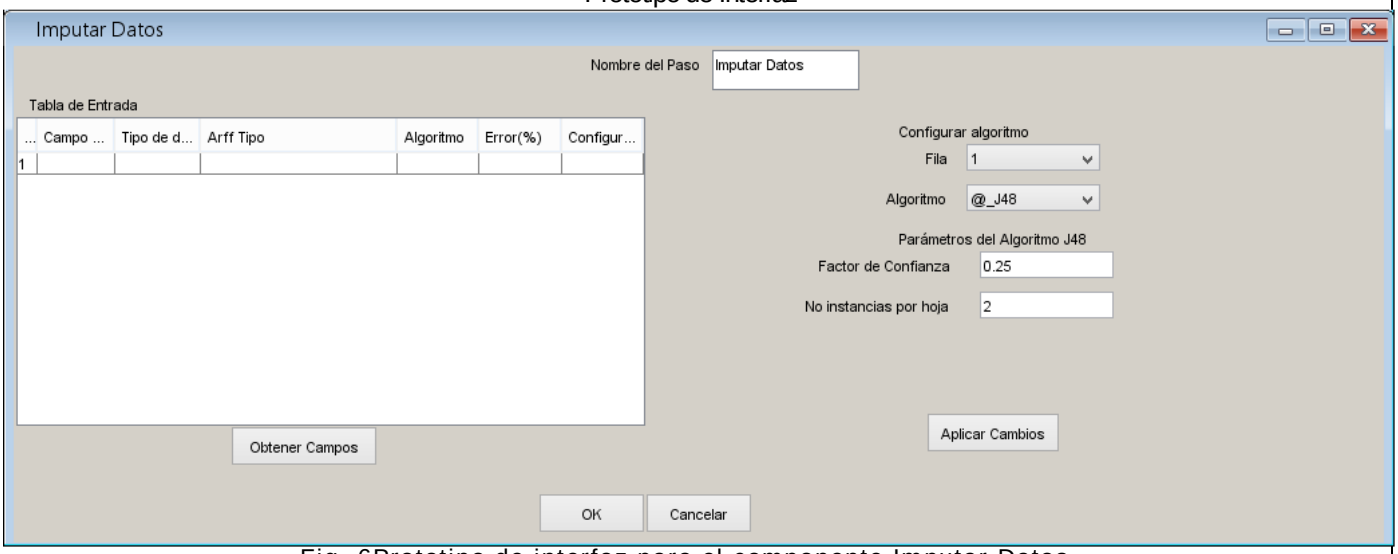

Fig. 6Prototipo de interfaz para el componente Imputar Datos.

Flujos Alternos	
Flujo alterno al paso 3 "Configuración cancelada"	
3.a El usuario selecciona la opción "Cancelar"	3.b El sistema cierra la interfaz de configuración
Flujo alterno al paso 5 "Configuración candelada"	
5.a El usuario selecciona la opción "Cancelar"	5.b El sistema cierra la interfaz de configuración
Flujo alterno al paso 6 "Algoritmo incorrecto"	
	6.a El sistema muestra un mensaje informando que el algoritmo seleccionado no corresponde con el tipo de dato del campo, regresando asi al paso 4 del flujo básico "Configurar Imputación"

Prototipo de Interfaz

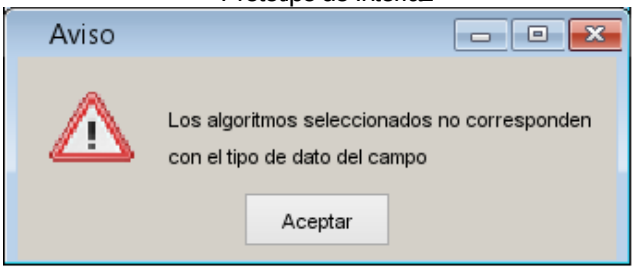

Fig. 7Prototipo de interfaz de aviso cuando el algoritmo seleccionado no corresponde con el tipo del campo.

2.5 Diagramas de clases de diseño

Un diagrama de clases de diseño (DCD) representa las especificaciones de las clases e interfaces del *software*. A continuación se muestra el diagrama de clases de diseño del caso de usoConfigurar imputación.

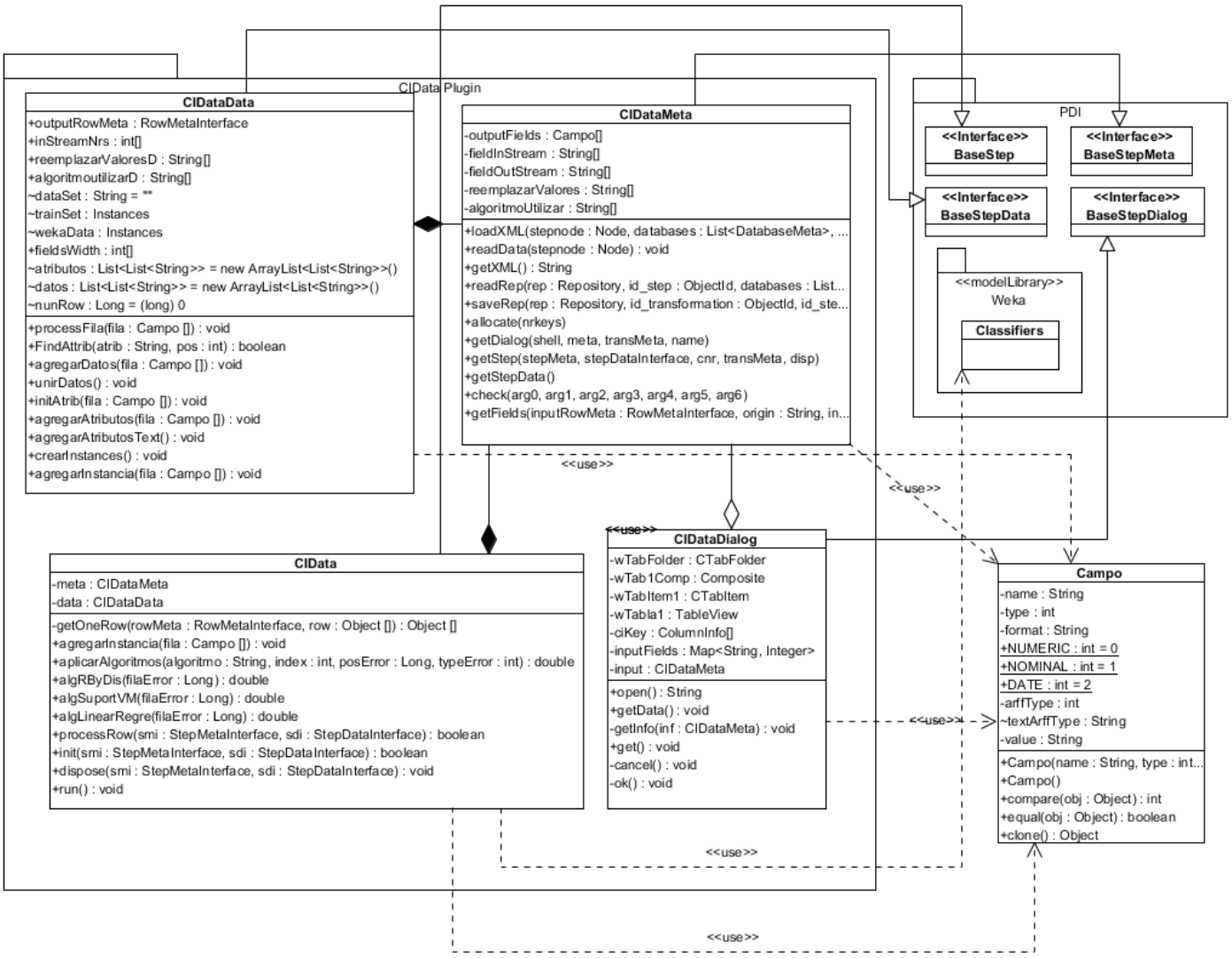

Fig. 8Diagrama de clases del diseño del caso de uso Configurar Imputación.

El diagrama de clase del diseño de caso de uso Configurar imputación, mostrado anteriormente está compuesto por tres paquetes, el paquete CIData *plugin* que cuenta con cuatro clases que forman parte de la estructura básica de un *plugin* en la herramienta Pentaho Data Integration. Cada clase muestra los principales datos y métodos que intervienen en el proceso de imputación. En el paquete PDI se

encuentran las clases padres que contienen las funcionalidades básicaspara la ejecución de un *plugin.* Porúltimo está el paquete Weka el cual contiene algoritmos que clasifican y predicen variables de un modelo de datos.

2.6 Patrón arquitectónico

Los patrones arquitectónicos ayudan a definir las características básicas y el comportamiento de una aplicación. Mediante el uso de estos se estandariza y organiza la forma de desarrollar un sistema informático. (23)

Arquitectura N capas

Este modelo se encarga de organizar el sistema a desarrollar en varias capas, asignándole a cada una un conjunto de responsabilidades para la realización de una actividad. Este patrón soporta el desarrollo incremental de sistemas pues brinda la facilidad de que a medida que se desarrolla una capa algunos de los servicios proporcionados por esa capa pueden estar disponibles para los usuarios. Este tipo de arquitectura también soporta los cambios que ocurren en el sistema, esta característica se evidencia ya que si la interfaz permanece sin cambios una capa puede reemplazarse por otra capa equivalente.(23)

En la construcción del *plugin*se definieron 2 niveles de abstracción (capas) Presentación y Lógica. En la capa de Presentación está la clase que contiene el código que le brinda al usuario una interfaz con las opciones a las que puede acceder para la configuración del proceso de imputación. La capa Lógica en su interior posee las clases y procedimientos necesarios para dar cumplimiento a los requisitos funcionales.

2.7 Patrones utilizados

2.7.1 Patrones GRASP

Los patrones GRASP (acrónimo de General Responsibility Assingment Software Patterns)constituyen un apoyo para entender los principios fundamentales del diseño de objetos y la asignación de responsabilidades (21).

- **Experto**: El objetivo de este patrón es asignar la responsabilidad de realizar una labor específica a la clase que contenga la información necesaria para ello. Este diseño se evidencia en la clase

CIDataData que se encarga de crear la base de conocimientos con los datos que contiene para la predicción de un valor.

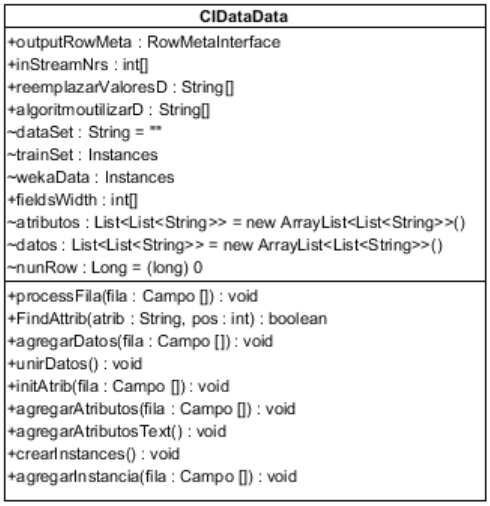

Fig. 9Diagrama de la clase CIDataData

- **Creador**: Se utiliza el patrón Creador para guiar la asignación de responsabilidades relacionadas con la creación de objetos. La actividad principal de este patrón es encontrar un creador que necesite conectarse al objeto creado en alguna situación. Se refleja en la clase CIDataMeta ya que se encarga de crear las clases CIDataDialog y CIData.

```
public class CIDataMeta extends BaseStepMeta implements StepMetaInterface {

public StepDialogInterface getDialog(Shell shell, StepMetaInterface meta,
        TransMeta transMeta, String name) {
    return new CIDataDialog(shell, meta, transMeta, name);
}

public StepInterface getStep(StepMeta stepMeta,
        StepDataInterface stepDataInterface, int cnr, TransMeta transMeta,
        Trans disp) {
    return new CIData(stepMeta, stepDataInterface, cnr, transMeta, disp);
}
}
```

Fig. 10Fragmentos de la inicialización de un objeto.

- **Controlador**: Patrón que establece el uso de una clase controladora, que es responsable del manejo de los eventos del sistema, que no pertenece a la interfaz del usuario, el controlador recibe la solicitud del servicio desde la interfaz y coordina su realización delegando esta actividad a otras

clases con las que mantiene un modelo de alta cohesión en la Fig. 11 se muestra la relación entre las clases en cuestión.

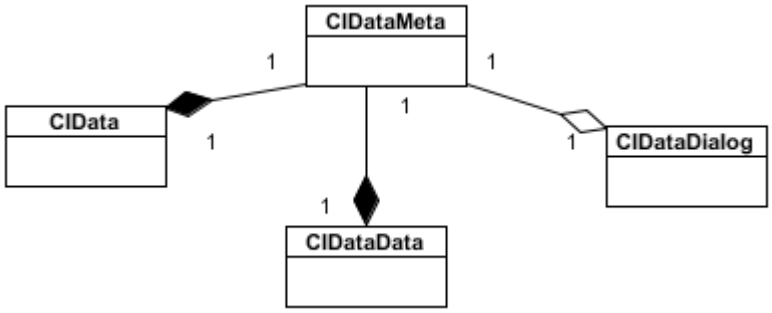

Fig. 11Patrón GRASP Controlador en el diseño de la clase CIDataMeta.

2.7.2 Patrones GoF

Los patrones Gang of Four (GoF) son patrones de diseño que describen soluciones a problemas específicos en el diseño del *software* orientado a objetos. Estos patrones forman en total un conjunto de 23 patrones los cuales se agrupan en tres grupos: patrones creacionales, estructurales y de comportamiento.

- **Prototipo**: Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo. Es utilizado en ciertos escenarios donde es preciso abstraer la lógica que decide que tipos de objetos utilizará una aplicación, de la lógica que luego usarán esos objetos en su ejecución.
- **Comportamiento**: En este patrón los objetos que manejan tipos particulares de acciones dentro de un programa. Éstos encapsulan procesos que se quieren ejecutar, permiten interpretar un lenguaje, completar una petición, moverse a través de una secuencia o implementar un algoritmo.
- **Mediador**: Este patrón define un objeto que contiene la forma en que interactúan un grupo de objetos asegurando así un acoplamiento débil al evitar las referencias explícitas entre los objetos y permitiendo, por tanto que su interacción se modifique de forma independiente.
- **Creación**: El objetivo que persigue este patrón es separar la construcción y la representación de un objeto complejo, para permitir que el mismo proceso de construcción pueda crear diferentes representaciones del mismo.

- **Estructura**: Esto afecta a la manera en que los objetos se conectan con otros objetos para asegurar que los cambios del sistema no requieren cambiar esas conexiones. Los patrones estructurales suelen imponer las restricciones del proyecto.

Conclusiones del capítulo

En este capítulo se realizó el modelo del dominio y el levantamiento de requisitos para definir las funcionalidades que presentará el *plugin*. Se identificaron seis requisitos funcionales los cuales quedaron agrupados en tres casos de uso del sistema lo que proporcionó un mayor entendimiento de la lógica de la aplicación. Se realizó el levantamiento de los requisitos no funcionales para poder detectar las características que se deben tener en cuenta a la hora deimplementar el *plugin*. Se analizaron y describieron los principales patrones asociados a la investigación; mediante generación de estos artefactos se logró comprender el funcionamiento del sistema lo que posibilita realizar de forma efectiva la fase de implementación.

Capítulo 3 Implementación y pruebas del componenteImputar Datos

En este capítulose describen los elementos principales del proceso de implementación del componente realizado con el objetivo de darle solución a los requisitos funcionales. Se describen los diagramas de componentes y los principales elementos del estándar de codificación que se emplean. También se muestran los resultados de las pruebas realizadas al *software* para chequear la correcta implementación de las funcionalidades definidas.

3.1 Diagrama de componentes

Un diagrama de componentes tiene como objetivo mostrar las partes modulares que encapsulan la implementación y el despliegue en un sistema. Se reflejan las relaciones y dependencias entre las partes (componentes), así como las entidades para dar cumplimiento a los requisitos. No es necesario que un diagrama incluya todos los componentes del sistema, estos normalmente se realizan por partes, en la Fig.12 se muestra el diagrama de componentes y para un mayor entendimiento a continuación se relaciona cada componente con las clases que representa.

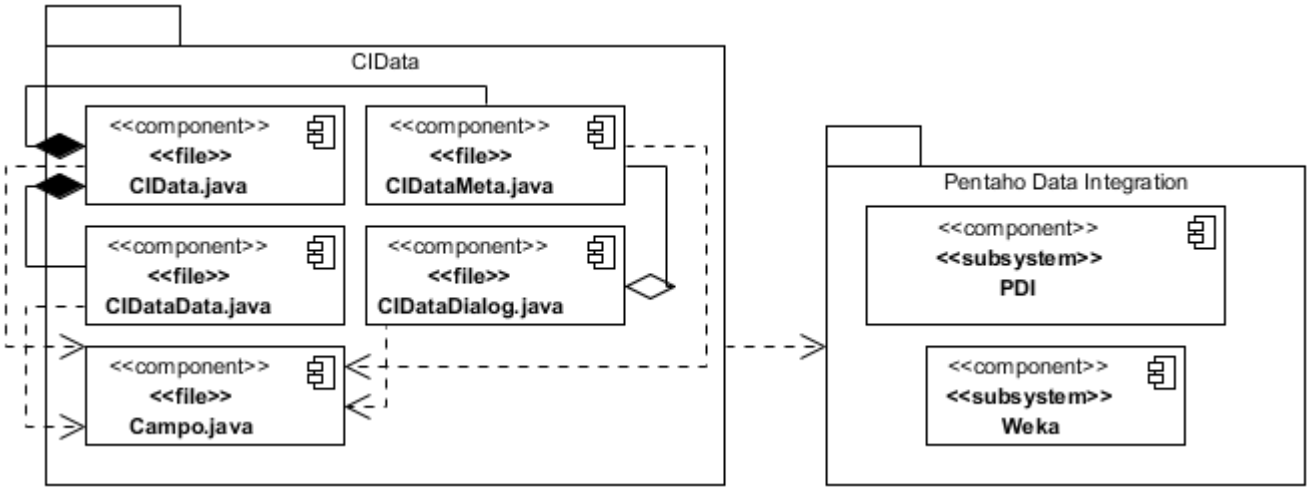

Fig. 12Diagrama de componentes del caso de uso Configurar Imputación.

3.1.1 Descripción del diagrama de componentes

CIData: Este paquete contiene los componentes CIData.java, CIDataMeta.java, CIDataData.java, CIDataDialog.java y Campo.java y estos a su vez las clases CIData, CIDataMeta, CIDataData,

CIDataDialog y Campo respectivamente las que componen la estructura básica de un componente para Pentaho Data Integration.

Pentaho Data Integration: Este paquete contiene dos componentes el PDI que engloba las funciones principales de la herramienta Pentaho Data Integration y Weka que cuenta con un conjunto de clases que ayudan al análisis estadístico de datos. El PDI es el motor de integración de datos con que cuenta el Pentaho Data Integration y el Weka se encarga de realizar la clasificación y predicción de los datos.

3.2 Consideraciones sobre la implementación

Cuando se implementa un *plugin* para la herramienta Pentaho Data Integration es necesario conocer la estructura que presentan las clases que lo conforman. Conociendo el formato de las clases y la configuración que debe tener el archivo plugin.xml se puede realizar la integración de un nuevo componente al PDI que le aporte nuevas funcionalidades para el trabajo los datos. El proyecto está compuesto por un paquete que en su interior alberga cinco clases las cuales tienen distribuidas las funcionales para la correcta ejecución del *plugin*. En la Fig. 13 se muestra lo antes expuesto.

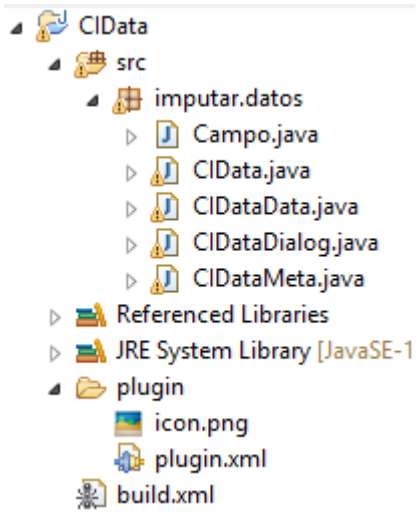

Fig. 13Estructura del plugin en Eclipse.

3.2.1 Descripción de las funciones asociadas a los ficheros de un plugin para Pentaho Data Integration

CIData: implementa la clase StepInterface que se encarga del procesamiento de las filas cuando la transformación es ejecutada en el Pentaho Data Integration. Cada hilo de ejecución es representado por

una instancia de esta clase. También se encarga de obtener una instancia de la clase Meta y la clase Data cuando es ejecutada la transformación para realizar el procesamiento de las filas.

CIDataData: implementa la clase StepDataInterface que se encarga del almacenamiento de los datos importantes asociados a un hilo de ejecución cuando la transformación comienza. En esta clase es donde se almacena en cache los datos asociados a las filas que sufrirán cambios durante la ejecución del componente.

CIDataMeta: implementa la clase StepMetaInterface, que es la responsable de obtener y almacenar las configuraciones escogidas para una instancia del paso en particular. En el caso de estudio es la clase que guarda el tipo de algoritmo, tipo de dato y nombre de columna de la tabla donde se realizará el proceso de imputación.

CIDataDialog: implementa la clase StepDialogInterface es la encargada de brindar la interfaz de usuario asociada al paso. Muestra al usuario un panel que le permite configurar el comportamiento del paso, esta clase está estrechamente relacionada con la clase Meta ya que es la que le brinda la configuración para que esta la almacene.

***plugin.xml**:* La función principal de este archivo es brindarle la localización de la clase Meta al Pentaho Data Integration además, del nombre y la descripción del *plugin.* A continuación una representación del código asociado al componente Imputar Datos.

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin

   id="CIData"
   iconfile="icon.png"
   description="Imputar Datos"
   tooltip="Componente para imputar valores vacios mediante varios métodos de predicción estadistica"
   category="Transform"
   classname="imputar.datos.CIDataMeta">

    <libraries>
            <library name="CIData.jar"/>
    </libraries>
    <localized_category>
    <category locale='en_ES'>Transformar</category>
    </localized_category>
    <localized_category>
    <category locale='en_US'>Transform</category>
    </localized_category>
</plugin>
```

Fig. 14Código XML para la incorporación del plugin.

3.3 Estándar de codificación

Los estándares de codificación establecen un conjunto de reglas que los desarrolladores deben seguir para escribir el código de fuente de un *software*. La aplicación de un estándar aporta una mayor organización y efectividad durante la fase de implementación. Son pautas de programación que no están enfocadas a la lógica del programa, sino a la estructura y apariencia física haciéndolo más reutilizable.

- Los nombres de las clases deben estar asociados a la funcionalidad que presente la misma, cuando están formados por más de una palabra cada una debe comenzar con mayúscula.
- Los nombres de las variables y métodos tienen que ser lo más descriptivos posibles procurando que sean palabras en minúsculas con un significado claro. De contener más de una palabra a partir de la segunda deben comenzar en mayúscula.
- Las líneas en blanco facilitan la lectura separando secciones de código que están lógicamente relacionadas por lo que se debe usar siempre una línea en blanco entre métodos.
- Evitar las líneas de más de 80 caracteres.

Ejemplos de código que cumplen con las normas establecidas:

```
public class CIDataDialog extends BaseStepDialog implements StepDialogInterface

    private static Class<?> PKG = CIDataMeta.class; // for i18n purposes

    private CTabFolder wTabFolder;
    private FormData fdTabFolder;
    private Composite wTab1Comp, wTab2Comp;
    private FormData fdTab1Comp, fdTab2Comp;
```

Fig. 15Ejemplo del estándar de nombres de variables y clases.

```
}

// buscar atributo
public boolean FindAttrib(String atrib, int pos) {
     int temp = atributos.get(pos).size();
     for (int i = 0; i < temp; i++) {
          if (atrib.equalsIgnoreCase(atributos.get(pos).get(i))) {
               return true;
          }
     }
     return false;
}

// Agregar datos
public void agregarDatos(Campo[] fila) {
     int cant = fila.length;
```

Fig. 16Ejemplo del estándar de espacio en blanco.

3.4 Pruebas

"*Las pruebas son un conjunto de actividades que se plantean con anticipación y se realizan de manera sistemática para evaluar la calidad de un producto de software y descubrir errores en él*" (25). Las pruebas cuentan con un conjunto de reglas u objetivos a seguir para conseguir el éxito de las mismas.

- Las pruebas consisten en un proceso que se ejecuta a un programa con la intención de encontrar fallas en su funcionamiento.
- Un buen caso de prueba es aquel en el que hay una gran probabilidad de encontrar un error.
- Una prueba exitosa es aquella que encuentra un error que no se sospecha su ocurrencia.

3.4.1 Formas de probar un software

Para evaluar la calidad de un producto de *software* son utilizadas las pruebas de Caja Negra estas se centran en el chequeo de los requisitos funcionales detectados en la fase de análisis y diseño. Permiten derivar un conjunto de condiciones de entrada que ayudarán a detectar errores en la interfaz, en la estructura de datos o el acceso a bases de datos externas, en el comportamiento o desempeño de la aplicación y por último errores de inicialización. Como parte de las pruebas de caja negra se empleó la técnica Partición de equivalente, que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Se elaboraron casos de pruebas a partir de las funcionalidades descritas en los casos de uso del sistema. En cada caso de prueba se refleja la especificación de un caso de uso, divido en secciones y escenarios donde se detallan las funcionalidades descritas en él, además se describen las variables utilizadas.

Para realizar el caso de pruebas correspondiente al caso de uso Configurar Imputación se realizó una tabla con las descripciones de las variables que representan una entrada de datos. En la Tabla. 4 se muestran las variables V1 y V2 representando a los campos "Nombre del paso" y "Algoritmo" respectivamente y en la Tabla. 5 se describe el caso de prueba antes mencionado.

Tabla.4 Descripción de las variables del caso de prueba Configurar Imputación

No	Nombre del campo	Clasificación	Valor nulo	Descripción
V1	Nombre del paso	Campo de texto	No	Campo que admite caracteres de cualquier tipo inicialmente contiene el nombre del paso y no puede ser nulo
V2	Algoritmo	Cuadro combinado	No	Cuadro que permite seleccionar el algoritmo de imputación a utilizar para un campo del flujo de datos.

Tabla.5 Caso de prueba Configurar Imputación

Escenario	Descripción	V1	V2	Respuesta del sistema	Flujo Central
EC 1.1 El actor le cambia el nombre al paso	En este escenario el actor le cambia el nombre al paso y selecciona la opción OK	V (Caso 1)	NA	El sistema cambia y muestra el nombre	1. El usuario abre la interfaz del componente 2. El usuario cambia el nombre del paso 3.El usuario selecciona la opción OK
EC 1.2 El actor selecciona el algoritmo de imputación	En este escenario el actor selecciona el algoritmo de imputación de acuerdo al tipo de dato del campo y selecciona la opción de OK	NA	V ($_J48)	El sistema cambia y guarda el valor del algoritmo	1. El usuario abre la interfaz del componente 2. El usuario selecciona el algoritmo 3. El usuario selecciona la opción OK

Después de realizadas las pruebas funcionales, los errores encontrados fueron redactados como no conformidades (ver Tabla. 6) para ser solucionados por el equipo de desarrollo. En total se realizaron tres iteraciones y se encontraron ocho no conformidades, las cuales fueron resueltas en cada iteración correspondiente, como se muestra en la Fig. 17.

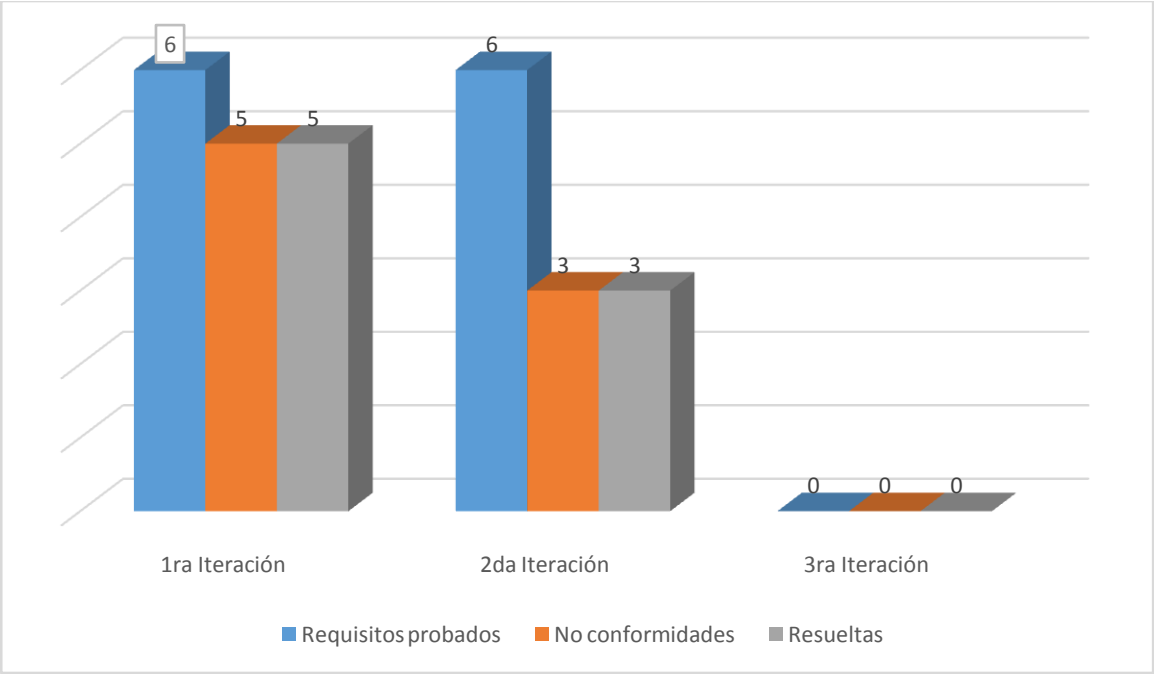

Fig. 17Resultados de las pruebas de Caja Negra.

Tabla.6 Ejemplo de no conformidades encontradas

Elemento	No	No conformidad	Ubicación	Etapa de detección	Clasificación
Aplicación	1	El campo “Nombre del paso” acepta un valor nulo	Interfaz del componente	Pruebas de funcionalidad	Significativa
Aplicación	2	El campo “Algoritmo” acepta un valor nulo	Interfaz del componente	Pruebas de funcionalidad	Significativa
Aplicación	3	El sistema no guarda la configuración	Interfaz del componente	Pruebas de funcionalidad	Significativa
Aplicación	4	El sistema no obtiene los campos del paso anterior	Interfaz del componente	Pruebas de funcionalidad	Significativa
Aplicación	5	El sistema no válida el tipo de algoritmo con respecto al tipo de dato	Interfaz del componente	Pruebas	Significativa

3.4.2 Pruebas de integración

Las Pruebas de Integración son aquellas que permiten probar en conjunto distintos subsistemas funcionales o componentes del proyecto para verificar que interactúan de manera correcta y que se ajustan a los requisitos especificados.

Este tipo de pruebas deberán ejecutarse una vez se haya asegurado el funcionamiento correcto de cada una de las funcionalidades por separado. Una vez que se ejecute sin errores el *plugin*se procederá a integrar el componente dentro del PDI, siguiendo una estrategia incremental ascendente. A medida que se integren las funcionalidades, se aplicarán pruebas funcionales para favorecer la detección de errores y no conformidades. (26)

Fig. 18Componente para imputar datos en funcionamiento a partir de una transformación.

Execution Results

Execution History | Logging | Step Metrics | Performance Graph

#	Stepname	Copynr	Read	Written	Input	Output	Updated	Rejected	Errors	Active	Time	Speed (r/s)	input/output
1	pred imputar	0	510	510	0	0	0	0	0	Finished	4.5s	114	-
2	pred salida	0	510	510	0	510	0	0	0	Finished	4.6s	112	-
3	pred entrada	0	0	510	510	0	0	0	0	Finished	0.5s	1.090	-

Fig. 19Estadísticas del flujo de datos a partir de una transformación.

Luego de realizada la prueba de integración se concluye que el *plugin*Imputar Datos fue aceptado por la herramienta Pentaho Data Integration ya que es capaz de acoplarse a la estructura de la herramienta y continuar con el flujo de datos del proceso de ETL.

3.5 Interfaz del sistema

Como resultado del proceso de implementación se muestra la interfaz del componente Imputar Datos. En la Fig. 20 se presenta la vista que se muestra al usuario donde se seleccionan las columnas a las cuales se le va a realizar la imputación de valores perdidos, también los algoritmos asociados a cada una de ellas.

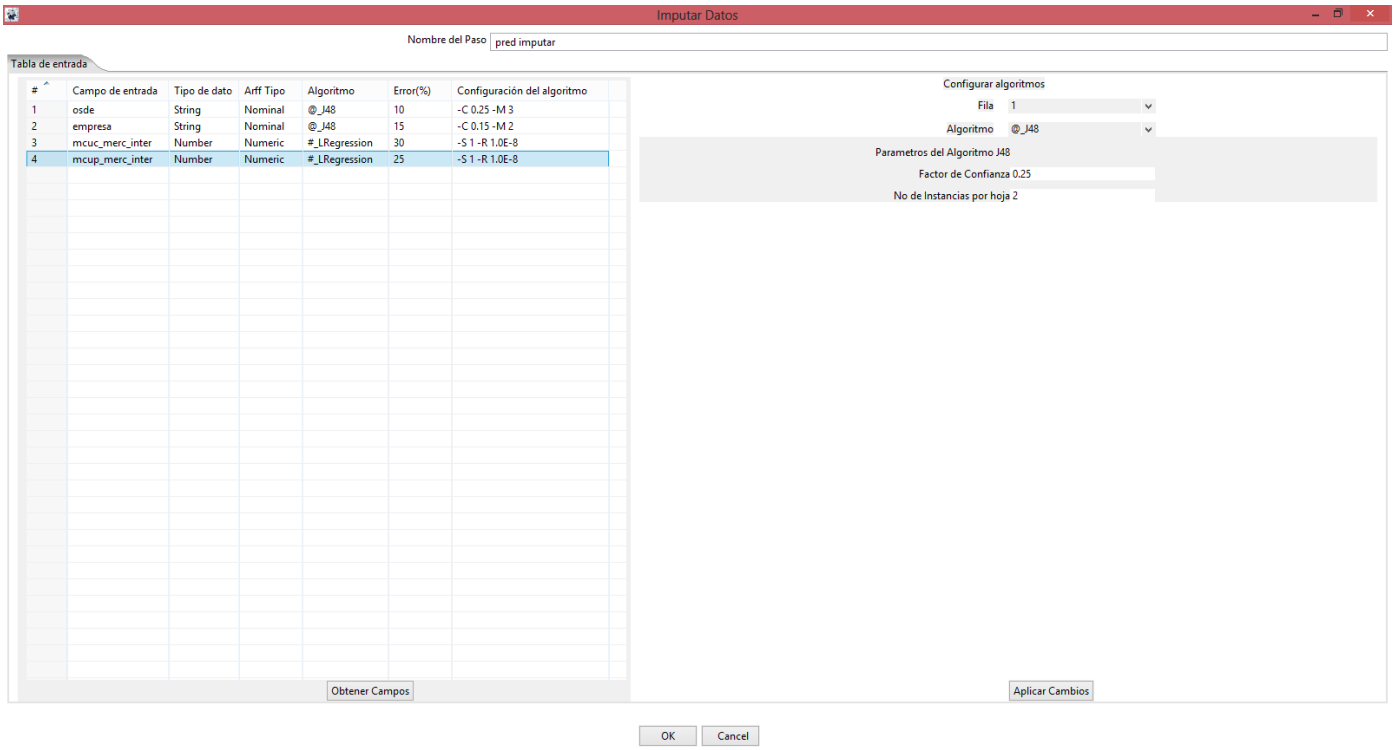

Fig. 20 Interfaz del componente Inferir Datos

Conclusiones Parciales

Con el desarrollo del presente capítulo se realizó el modelo de implementación del componente Imputar Datos con el propósito de mostrar los elementos que componen el *plugin* y su relación con el PDI. Se describieron de forma breve las clases que se implementaron. Se especificó el uso de los estándares de codificación lo cual contribuyó a la obtención de un código fácil de comprender. Se realizaron las pruebas a través del método de caja negra que permitieron detectar errores para posteriormente corregirlos ademásde comprobar la integración del componente con el Pentaho Data Integration.

Conclusiones Generales

Después de realizada la investigación y de haberse construido el *plugin*Imputar Datos, recorriendo por las fases del proceso de desarrollo de *software* se puede arribar a las siguientes conclusiones:

- A través del estudio de los fundamentos teóricos del proceso de inferencia e imputación se logró seleccionar una metodología de desarrollo, las herramientas y tecnologías necesarias para la construcción del componente para imputar datos en la herramienta Pentaho Data Integration.
- El estudio y análisis de la herramienta Pentaho Data Integration además del problema de imputación de valores perdidos permitió, una correcta identificación de los requisitos del sistema.
- Mediante el diseño del componente se logró definir su organización lógica, haciendo uso de los patrones de diseño.
- Las pruebas diseñadas y aplicadas garantizaron el correcto funcionamiento del*plugin* desarrollado, detectando no conformidades que fueron solucionadas posteriormente.
- El desarrollo del componente Imputar Datos para la herramienta Pentaho Data Integration le brinda a los especialistas en almacenes de datos del centro DATEC una nueva forma de tratar los valores perdidos, aumentando el conocimiento que se puede extraer mediante el proceso de KDD.

Recomendaciones

Se recomienda incrementar las funcionalidades del componente Imputar Datos de manera que se puedan realizar otras tareas de la limpieza de datos incrementando las posibles formas de usar el componente. Para la implementación de las nuevas prestaciones que tendrá el componente se proponen las siguientes tareas:

- Implementar un mecanismo que permita la detección y corrección de valores erróneos.
- Implementar un mecanismo que permita la detección y corrección de filas duplicadas

Referencias Bibliográficas

1. **Guevara, M.***La información.* 1995.

2. **Frawley, William.**Association for the Advancement of Artificial Intelligence! [En línea] 2014. http://aaaipress.org/ojs/index.php/aimagazine/article/viewFile/1011/929.

3. **Fayyad, Usama, Piatetsky-Shapiro, Gregory y Smyth, Padhraic .** From Data Mining to KDD. *Western University.* [En línea] 2015. [Citado el: 4 de Febrero de 2015.] http://www.csd.uwo.ca/faculty/ling/cs435/fayyad.pdf.

4. **Ramirez Quintana, José, Hernández Orallo, José y Ferri Ramírez, Cesar.***Introducción a la mineria de datos.* Madrid : Pearson Educación, S.A, 2004. ISBN 84-205-4091-9.

5. **Garcia Mancilla, Hugo y Matus Parra, Juan.** Estadistica descriptiva e inferencial I. *Colegio de Bachilleres.* [En línea] [Citado el: 2 de Junio de 2015.] http://www.biblioises.com.ar/Contenido/300/310/Estadistica%20general%20.pdf.

6. **Medina, Fernando y Galván, Marcos.** Imputación de datos. teoría y prática. [En línea] [Citado el: 2 de Junio de 2015.] http://archivo.cepal.org/pdfs/2007/S0700590.pdf.

7. **Bouman, Roland y Dongen, Jos van.***Pentaho Solutions Business Intelligence and Data Warehousing with Pentaho and Mysql.* Indianapolis, Indiana : Wiley Publishing Inc., 2009. ISBN:978-0-470-48432-6.

8. **NEOPOST.** Datacleaner. [En línea] 2014. [Citado el: 27 de Mayo de 2015.] http://datacleaner.org/resources/docs/4.0.7/pdf/datacleaner-reference.pdf.

9. **Raman, Vijayshankar y Hellerstein, Joseph M. .** Continuous Output and Navigation Technology with Refinement On-Line. *Potter'sWheel: An Interactive Data Cleaning System.* [En línea] [Citado el: 12 de Mayo de 2015.] http://control.cs.berkeley.edu/pwheel-vldb.pdf.

10. **Galhardas, Helena, y otros.** Very Large Data Base Endowment Inc. *VLDB Endowment.* [En línea] [Citado el: 12 de Mayo de 2015.] http://www.vldb.org/conf/2001/P371.pdf.

11. **Beck, Kent y y otros.** Manifiesto Ágil. *Ward Cunningham.* [En línea] 2001. [Citado el: 20 de Enero de 2015.] http://www.agilemanifesto.org/iso/es/principles.html.

12. **Eclipse Fundation.***EPF Wiki.* [En línea] 2014. [Citado el: 2 de Febrero de 2015.] http://epf.eclipse.org/wikis/openup/.

13. **Gimson, Loraine, Gil, Gustavo y Rossi, Gustavo.** [En línea] Junio de 2012. [Citado el: 5 de Febrero de 2015.] http://sedici.unlp.edu.ar/bitstream/handle/10915/24942/Documento_completo__.pdf?sequence=1.

14. **Hernández Orallo, Enrique.** El Lenguaje unificado de modelado (UML). *Universidad Politécnica de Valencia.* [En línea] [Citado el: 07 de Julio de 2015.] http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

15. **Universidad Politécnica de Valencia.** Introducción a Herramientas CASE y System Architect. *Universidad Politécnica de Valencia.* [En línea] [Citado el: 7 de Julio de 2015.] http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

16. **Departamento de Informática.** Guión Visual Paradigm for UML. *Universidad Carlos III de Madrid.* [En línea] [Citado el: 7 de Julio de 2015.] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.

17. **Martínez Ladrón de Guevara, Jorge.** Fundamentos de programación en Java . *Universidad Complutence de Madrid.* [En línea] [Citado el: 7 de Julio de 2015.] http://pendientedemigracion.ucm.es/info/tecnomovil/documentos/fjava.pdf. ISBN 978-84-96285-36-2 .

18. **Alonzo Velazquez, José Luis.***Centro de Investigación en Matématicas.* [En línea] [Citado el: 7 de Julio de 2015.] http://www.cimat.mx/~pepe/cursos/lenguaje_2010/slides/slide_17.pdf.

19. **Vogel, Lars.** Eclipse IDE - Tutorial. *Vogel/a.* [En línea] [Citado el: 7 de Julio de 2015.] http://www.vogella.com/tutorials/Eclipse/article.html.

20. **García Morate, Diego.** Manual de Weka. *MetaEmotion.* [En línea] [Citado el: 7 de Julio de 2015.] http://www.metaemotion.com/diego.garcia.morate/download/weka.pdf.

21. **Larman, Craig.***UML y Patrones.* 2003.

22. **Arias Chaves, Michael.** Revista Electrónica de las Sedes Regionales de la Universidad de Costa Rica. *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* [En línea] 26 de Julio de 2007. [Citado el: 15 de Mayo de 2015.] http://www.intersedes.ucr.ac.cr/ojs/index.php/intersedes/article/viewFile/119/118. ISSN 1409-4746.

23. **Sommerville, Ian.***Ingeniería del Software (7ma edición).* Madrid : Pearson Educación S.A, 2005. ISBN 84-7829-074-5.

24. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.***El proceso unificado de desarrollo de software.* Madrid : PEARSON EDUCACIÓN S.A, 2000. ISBN 84-7829-036-2.

25. **Pressman, Roger S.***Ingenieria del Software Un enfoque práctico.*

26. **Alfaro Medina, Jazmín y y otros.** Pruebas de software y Pruebas de Aceptación. *DocSlide.* [En línea] 2013. [Citado el: 2 de Junio de 2015.] http://myslide.es/education/pruebas-de-sistemas-y-aceptacion.html.

27. **Richards, Mark.***Software Architecture Patterns "Understanding Common Architecture".* Sebastopol : O´Relly Media, 2015.