Universidad de las Ciencias Informáticas

Facultad 6

Centro de Tecnología de Gestión de Datos

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Componente para extraer datos desde hojas de cálculo con matriz de datos compleja para Pentaho Data Integration

Autores: Carmen Bolivar Colomé

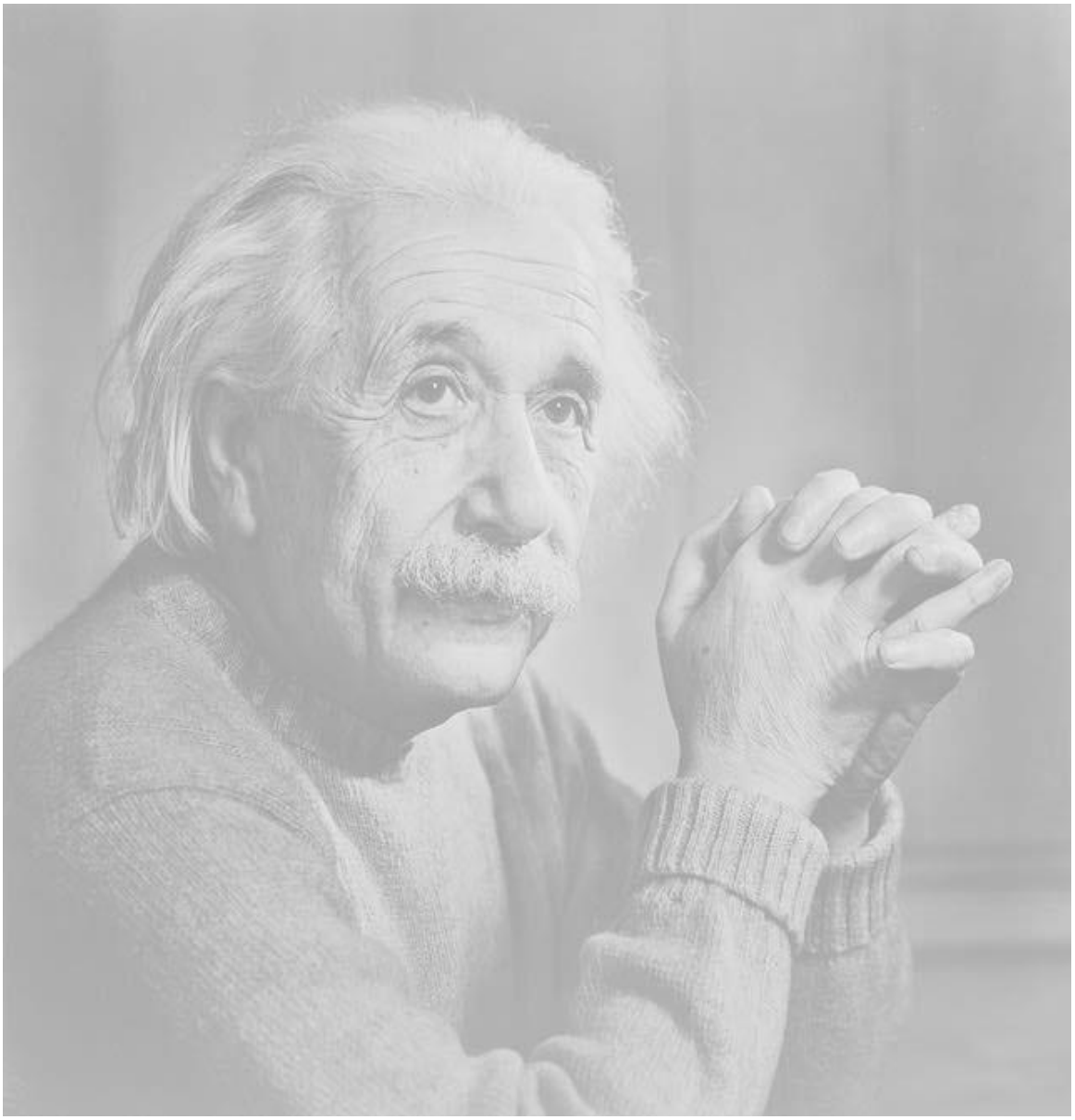
Ariel Linares Ylizastigui

Tutores: Msc. Yudelkis Abad Fuentes

Ing. Yordanka Hechavarría Melo

La Habana, 2014.

"Año 56 de la Revolución"



¨Todos somos muy ignorantes. Lo que ocurre es que no todos ignoramos las mismas cosas.¨

¨Si supiese lo que estoy haciendo, no le llamaría investigación, ¿Verdad?¨

Albert Einstein

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Carmen Bolivar Colomé

_____

Autor

Ariel Linares Ylizastigui

_____

Autor

Yudelkis Abad Fuentes

_____

Tutor

Yordanka Hechavarría Melo

_____

Tutor

Datos de contacto

TUTORES

Msc. Yudelkis Abad Fuentes

Universidad de las Ciencias Informáticas, La Habana, Cuba.

E-mail: yabad@uci.cu

Ing. Yordanka Hechavarría Melo

Universidad de las Ciencias Informáticas, La Habana, Cuba.

E-mail: yordankah@uci.cu

Agradecimientos

De ambos:

*A la **Revolución** y a su líder histórico **Fidel Castro Ruz** por habernos dado la oportunidad de estudiar en la UCI.*

*A los **profesores** que contribuyeron en nuestro desarrollo profesional, ayudándonos en todo momento, y a aquellos que no lo hicieron, gracias por enseñarnos a ser más fuertes, pero más que eso, gracias por enseñarnos que siempre debemos tener alternativas ante cualquier fallo, les agradecemos a todos por habernos preparado para el futuro.*

*A nuestras tutoras **Yudelkis y Yordanka** que nos han aclarado las dudas que han surgido a lo largo del camino, gracias por ser nuestros guías aportándonos su conocimiento y tiempo.*

*A mi **compañero/a de tesis** por tanto sacrificio y dedicación para la realización del trabajo.*

A todas las personas que hemos conocido en la UCI por haber compartido con nosotros esta etapa tan linda y difícil.

*A **todos** los que han contribuido a la realización de este trabajo de diploma, muchas gracias.*

De Carmen

Agradecer después de tantos años es difícil porque son muchas las personas involucradas en este logro. Tendría que escribir casi el número de hojas que utilicé en la investigación para agradecer a todas ellas, pero trataré de resumir de alguna forma que no se me escape ninguna de esta cuartilla.

Comenzaré agradeciendo a dos personas esenciales en mi vida:

***Mi mamá** que sé que aunque no hubiera una escuela como esta donde yo pudiera estudiar, ella la habría inventado, por ser más que una madre, mi amiga, mi hermana, apoyándome incondicionalmente en cada etapa de mi vida y confiando en mí en cada una de ellas.*

AGRADECIMIENTOS

Mi abuela *que ha sido otra madre para mí, la cual ha estado siempre presente en cada momento, llenándome de su experiencia y consejos.*

Agradezco también a ***Julio*** *que se ha comportado desde que me conoció con 8 años como un padre, escuchándome cuando lo he necesitado, y ayudando a mi mamá durante tanto tiempo en mi formación.*

A ***mi papá*** *por haber sido mi guía, y por haberme enseñado que para lograr las cosas hay que luchar.*

Le agradezco a ***Abelito*** *por ser mi apoyo desde que lo conocí, por quererme incondicionalmente y demostrármelo en cada momento que está a mi lado.*

Agradezco a todos los que de una forma u otra aportaron su granito de arena, mi familia, mis amigos, mis compañeros de aula durante estos años, y a mis profesores. Especialmente a ***La Mirix, Odalys, Darlys, Isenith, Anabel, Dary, El Migue, La Meli, Anita, Roxana, Angel, Ariel, Joel, Alberto (mentiroso), Darwis, Andrés, Tellez, Maikel, Lacoste, Ismael, Maurice, Sergio, Alcides, Julio César, Orisel, Ramón.***

Agradecer ***a mí****, porque dejando la modestia a un lado puedo decir que soy un eslabón importante en la realización de este sueño, después de tanto esfuerzo y sacrificio. Con cada caída he sabido reponerme, levantarme, he aprendido a ser fuerte y me he crecido ante las dificultades, a pesar de estar alejada de las personas que más quiero, las personas que siempre estuvieron ahí para aconsejarme y apoyarme. He tenido que aprender desde los 18 años que estoy en esta universidad a ser independiente y a tomar mis propias decisiones. Agradecerme porque después de tantos años de estudio, de tristezas y alegrías, salgo con una actitud y manera de pensar diferentes, y con muchas ganas de vivir y hacer.*

Dedicatoria

De Carmen

A mi mamá y a mi abuela, principales responsables de todo lo bueno que pasa en mi vida.

Resumen

Las herramientas de integración de datos facilitan la gestión de la información ya que permiten organizarla, integrarla y analizarla, convirtiéndolas en datos útiles para la empresa. Con el avance de las Tecnologías de la Información y las Comunicaciones (TIC) ha sido necesario mejorar estas herramientas. Cuba no está exenta de este desarrollo utilizando la herramienta *Pentaho Data Integration* la cual es usada durante la extracción, transformación y carga de datos de un sistema a otro. En la presente investigación se propone una nueva versión del componente para la extracción de datos, que permite realizar de forma correcta el proceso de carga de los datos desde hojas de cálculo con matriz de datos compleja[1]. Para el desarrollo del componente se utilizó XP como metodología, Eclipse como entorno de desarrollo y Java como lenguaje de programación. Se implementa e integra dicho componente a la herramienta y se planean un conjunto de pruebas para garantizar el correcto funcionamiento del componente. Con la aplicación de las pruebas de software se evidenció que el componente es superior al anterior, ya que reduce los tiempos de ejecución de las transformaciones realizadas. Además disminuye la cantidad de componentes a utilizar en dicho proceso.

Palabras claves: *Pentaho Data Integration*, componente, matriz de datos compleja

[1]Hoja de cálculo que posee varias celdas (fila, columna) combinadas.

Abstract

The data integration tools facilitate the management of information and organize, compose and analyze it, making them in useful information for the organization. With the advancement in the Information Technologies and Communications (ICT) has been necessary to develop them. Cuba is not exempt from this development using the Pentaho Data Integration tool that is used for the extraction, transformation and loading of data from one system to another. In the present investigation is proposed a new version of the component for data mining, which allows correctly the process of data mining since from spreadsheets with complex data matrix. For the development of this component is used the methodology XP, Eclipse as a development environment and Java as programming language. The component is implemented and integrated to the tool and a group of tests are planned for ensuring the correct function of the component. With the implementation of software testing showed that the component is higher than the previous, as it reduces the execution time of the changes made. It also reduces the number of components to be used in this process.

Keywords: Pentaho Data Integration, component, complex data matrix.

ÍNDICE

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS

ÍNDICE DE LAS ILUSTRACIONES

ÍNDICE DE LAS TABLAS

Introducción

Con la evolución de las Tecnologías de la Información y las Comunicaciones (TIC) se ha acelerado el crecimiento económico y social en el mundo actual, provocando que se generen grandes cúmulos de datos, difíciles de manejar por el hombre de forma manual. Esta disponibilidad de información requiere de técnicas y herramientas que permitan organizarla, integrarla y analizarla para una mejor gestión de la información y convertirla en datos útiles para la empresa.

En el proceso de organización juega un papel importante las bases de datos y conceptos asociados; en el proceso de integración los avances se han orientado a la extracción, transformación y carga (por sus siglas en inglés ETL) de distintas fuentes en uno o varios repositorios de información y para un correcto análisis de los datos se ha logrado el desarrollo e integración de técnicas de inteligencia artificial que garantizan una mejor toma de decisiones a los directivos de una empresa.

De estos procesos, el de extracción, transformación y carga se ha convertido en un paso inevitable en todo sistema de almacenamiento para el traslado de datos desde algunos orígenes hasta el almacenamiento destino. Por lo que se han desarrollado numerosas herramientas entre la que destaca el *Pentaho Data Integration* (PDI) perteneciente a una *suite* de Inteligencia de Negocios (BI por sus siglas en inglés) de código abierto orientada a la implementación de soluciones y basada en procesos llamada *Pentaho.*

Cuba no se encuentra exenta al desarrollo anteriormente expuesto, en numerosos proyectos desarrollados en la Universidad de las Ciencias Informáticas (UCI) se utiliza dicha herramienta para facilitar los procesos de integración de las aplicaciones y apoyar los procesos de negocio. Dentro de la universidad, el Centro de Tecnologías de Gestión de Datos (DATEC) de la facultad 6 cuenta con el departamento de almacenes de datos en donde radica un grupo de trabajo perteneciente al proceso de integración.

En la actualidad, este grupo de trabajo de integración está presentando dificultades al manipular las hojas de cálculo con matriz de datos compleja, ya que la herramienta *Pentaho Data Integration* presenta limitaciones tales como:

- No identifica correctamente los encabezados que se encuentran dentro de celdas combinadas en una misma columna.
- No identifica los encabezados que se encuentran en un intervalo de columnas determinado donde existan celdas combinadas.
- No permite extraer datos de hojas de cálculo bloqueadas o protegidas.
- Cuando se extrae desde distintas hojas que presentan la misma estructura y nombre de los encabezados, ubica los encabezados de la segunda hoja a continuación de la primera y no presenta la opción de adicionar los datos de la segunda hoja a continuación de la primera.
- No lee las funciones predefinidas dentro de los ficheros Excel.

Por los elementos antes expuestos se define como **problema de investigación:** ¿Cómo extraer datos desde hojas de cálculo que presenten una matriz de datos compleja en la herramienta *Pentaho Data Integration* para mejorar el trabajo con dichas fuentes de datos?

La presente investigación posee como **objeto de estudio:** Aplicaciones informáticas[2] basadas en componentes, enmarcado en el **campo de acción:** Desarrollo de componentes para la herramienta *Pentaho Data Integration*.

La herramienta PDI posee una arquitectura basada en componentes lo que hace del sistema una aplicación flexible, a la cual se le puede ir aumentando sus funcionalidades gradualmente. Por esta razón se define como **objetivo general:** Desarrollar un componente para extraer datos desde hojas de cálculo con matriz de datos compleja para *Pentaho Data Integration*, del cual se desglosan las siguientes **preguntas científicas:**

1. ¿Cuáles son los referentes teóricos relacionados con la implementación del componente de extracción de datos desde hojas de cálculo con matriz de datos compleja para la herramienta *Pentaho Data Integration*?

[2]Tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajos.

2. ¿Cuáles son los requisitos a implementar para el desarrollo del componente de extracción de datos desde hojas de cálculo con matriz de datos compleja para la herramienta *Pentaho Data Integration*?
3. ¿Cómo desarrollar el componente de extracción de datos desde hojas de cálculo con matriz de datos compleja para la herramienta *Pentaho Data Integration*?
4. ¿Cómo integrar el componente de extracción de datos desde hojas de cálculo con matriz de datos compleja para la herramienta *Pentaho Data Integration*?
5. ¿Qué resultado se obtendrá de la integración del componente a la herramienta *Pentaho Data Integration*?

Para dar cumplimiento a las preguntas planteadas se definen las siguientes **tareas de la investigación:**

1. Caracterización de la herramienta *Pentaho Data Integration* para lograr comprender mejor su funcionamiento.
2. Descripción del componente para extraer datos desde hojas de cálculo de la herramienta *Pentaho Data Integration* para comprender su funcionamiento.
3. Análisis y selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo del componente para guiar el desarrollo del proyecto.
4. Elaboración de las historias de usuario para un mejor entendimiento de las funcionalidades del componente.
5. Elaboración de las tarjetas CRC (Clase, Responsabilidad, Colaboración) para el diseño de las clases del componente.
6. Elaboración del diagrama de clase de diseño para representar las relaciones entre las clases del componente.
7. Implementación del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja.
8. Realización de pruebas unitarias y funcionales para verificar el correcto funcionamiento del componente diseñado.

Para el desarrollo de la solución se utilizaron los métodos teóricos y empíricos de la investigación científica. Los mismos permitieron tener sintetizada la información a utilizar, información recopilada

necesaria y fundamental para tener una base o punto de partida. Dentro de los métodos teóricos se usaron:

- Histórico-Lógico: Para recopilar la información existente de las herramientas, sistemas, aplicaciones y soluciones de integración de datos similares a la propuesta solución.
- Analítico-Sintético: Para analizar y sintetizar la información recopilada durante la investigación.
- Modelación: Con el objetivo de modelar los eventos y acciones de la vida cotidiana a abstracciones, para una mejor comprensión de la solución a implementar.

Y dentro de los métodos empíricos se tuvo en cuenta la observación, para identificar y resumir las principales limitaciones con que cuenta la herramienta *Pentaho Data Integration*, así como, las principales funcionalidades del componente a desarrollar durante la investigación.

Estructura de la Tesis

El contenido de esta investigación estará estructurado en tres capítulos, conclusiones generales, bibliografía utilizada y los anexos que complementan el cuerpo del trabajo y que son necesarios para su entendimiento. A continuación se realiza una breve descripción de los elementos principales contenidos en cada acápite.

Capítulo 1: Fundamentación teórica del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja.

En este capítulo se definen los principales pasos a seguir durante el desarrollo de la investigación ya que contiene los fundamentos teóricos y conceptos para entender el problema. También se describen metodologías, lenguajes, y herramientas existentes, posibilitando una buena selección de los elementos a utilizar en el transcurso de la investigación.

Capítulo 2: Planificación y diseño del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja.

En este capítulo se realiza la planificación y diseño del componente para la extracción de datos

desde hojas de cálculo con matriz de datos compleja. Para ello se especifican los requisitos funcionales (RF) del sistema a través de las historias de usuarios (HU), las propiedades o cualidades que el producto debe tener, los patrones de diseño, y las tarjetas CRC para el diseño de las clases del componente.

Capítulo 3: Implementación y prueba del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja.

En este capítulo se implementan los procesos definidos en el diseño y se verificar si el sistema cumple con los requisitos descritos por el cliente. Para ello se implementan la programación en pareja y la integración continua, como artefactos del proceso de codificación; y se realizan pruebas de unidad y aceptación al componente desarrollado.

Capítulo 1: Fundamentación teórica del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja.

1.1. Introducción

En el transcurso de este capítulo se realiza un estudio del estado del arte del proceso de integración de datos, así como, de las herramientas y la metodología de la investigación a utilizar durante el desarrollo del componente.

1.2. Estado del arte

1.2.1. Proceso de integración de datos

Hoy en día los datos crecen de manera exponencial, tanto en volumen como en diversidad de fuentes. Integrarlos entre las aplicaciones y procesos de negocios de manera eficiente es la clave para maximizar el uso de estos en la toma de decisiones. La integración de datos es el proceso que permite concentrar, analizar y acceder a la información que provienen de disímiles orígenes (aplicaciones, ficheros, bases de datos, hojas de cálculo) para posteriormente mostrar la unión de estos a un cliente o usuario final. Ayudándolos a mejorar la agilidad, flexibilidad ante el cambio y la innovación. Es también una forma de reducir costos de integración, desarrollo y mantenimiento.

Los datos de negocios se encuentran en varios lugares y en múltiples formas, por lo que es necesario recopilarlos. Para llevar a cabo este proceso son útiles las herramientas ETL. En la bibliografía se encontraron numerosas herramientas de integración de datos, tanto propietarias, como de código abierto. Entre las herramientas de integración de datos de código abierto, se destacan en la actualidad, *Kettle* conocido como PDI, *Talend* y *Clover*.

1.2.2. Herramientas de integración de datos

Pentaho Data Integration es un proyecto de código abierto adoptado por la suite de inteligencia de negocios *Pentaho*. Presenta la característica de ser multiplataforma, y posee una interfaz amigable,

en donde el usuario puede ver y probar las diferentes transformaciones que se van realizando. Esta herramienta se encuentra escrita en el lenguaje de programación *Java*, soporta varios servidores de base de datos y es capaz de distribuir las tareas de ETL a través de ellos. (1)

PDI incluye un conjunto de programas que le permiten realizar los procesos ETL:

- **SPOON:** para diseñar transformaciones ETL usando el entorno gráfico.
- **PAN:** para ejecutar transformaciones diseñadas con spoon.
- **CHEF:** para crear trabajos.
- **KITCHEN:** para ejecutar trabajos. (1)

Con dichos programas, la herramientas cuenta con numerosas funcionalidades para extraer datos desde distintas fuentes, entre ellas, la de manipulación de hojas de cálculo, pero presenta numerosas limitaciones a la hora de manipular hojas de cálculo con matriz de datos compleja., como por ejemplo, problemas al identificar los encabezados en una misma columna con celdas combinadas, problemas de lectura de datos en hojas bloqueadas o protegidas, mala ubicación de los datos cuando se leen varias hojas con la misma estructura, y al leer las funciones sobre datos calculados por las hojas de cálculo.

Talend Data Integration es una herramienta de código abierto compatible con la plataforma para la creación de soluciones empresariales, de transformaciones e integración de datos *Microsoft SQL Server Integration Services* (SSIS).Posee versiones disponibles para *Windows*, *Unix* y *Linux*, lo que lo convierte en una herramienta multiplataforma. Presenta una interfaz de diseño donde se arrastran y sueltan componentes, los cuales se relacionan mediante el uso de conectores. Puede ser conectado con varios servidores de base de datos y genera componentes en diversos lenguajes de programación dentro de los que se destaca *Java*. (2)

Esta herramienta al igual que *Pentaho Data Integration* cuenta con numerosas funcionalidades para extraer datos desde distintas fuentes. La entrada de datos a partir de las hojas de cálculo, se realizan a través del componente *tFileInputExcel*. (Ver Fig.1)

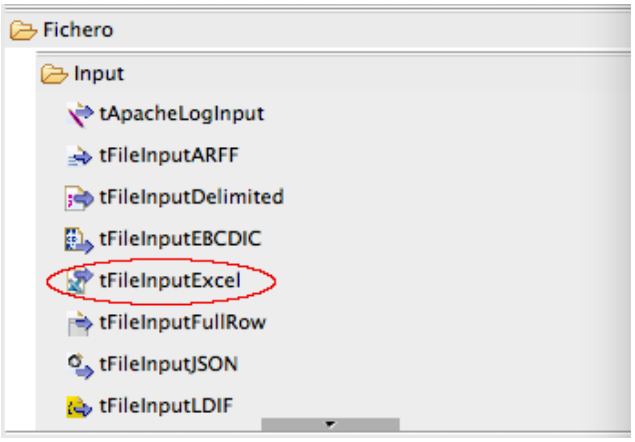


Fig. 1 Componentes de ficheros de entrada de Talend

Este componente permite realizar diversas funcionalidades como es el caso de *Read Excel 2007 file format* (xlsx) que permite marcar en caso de que el fichero a analizar pertenezca a una versión de Excel de 2007 o superior y *Check column to trim* que permite seleccionar que columnas del fichero serán excluidas del flujo de salida del componente. (3) Sin embargo, no cuenta con opciones o funcionalidades para la manipulación de hojas de cálculo con matriz de datos compleja.

Clover ETL es una herramienta de código abierto basada en *Java* que transforma datos estructurados. Puede estar incluida en otra aplicación, permite la conexión con bases de datos, realiza las transformaciones utilizando lenguaje de marcas extensible (por sus siglas en inglés XML) y puede ser ejecutado en varias máquinas. Está compuesto por componentes que actúan de forma independiente, los cuales pueden ser utilizados en otras aplicaciones, se distribuye bajo licencia pública y soporta varios tipos de datos y valores nulos. (4)

Clover cuenta con componentes de integración de datos desde hojas de cálculo con formato XLS y XLSX llamados *XLSDataReader* y *SpreadsheetDataReader*.

XLSDataReader solo realiza la lectura de aquellas hojas de cálculo con formato XLS mientras que *SpreadsheetDataReader* realiza las lecturas desde formatos XLS y XLSX, y presenta otras funcionalidades como Mapeo de datos complejos (formularios, tablas, registros de varias filas, entre otros). Todas las opciones de entrada estándar están disponibles como en otros lectores: los archivos locales y remotos, archivos zip, un puerto de entrada o un diccionario. También permite, si

los datos son encriptados en la hoja de cálculo de origen, escribir la contraseña para desbloquearlo. (4)

Esta herramienta solventa algunas de las limitaciones de las herramientas anteriores, pero no permite aislar dichas funcionalidades en un componente externo, que pueda adaptarse a otra herramienta de integración de datos.

Esto evidencia que no existen componentes para la manipulación de hojas de cálculo con matriz de datos compleja en las herramientas de código abierto más usadas en la actualidad, por lo que al incorporar dicha funcionalidad a la herramienta PDI, la convertiría en una herramienta de integración con características superiores en este aspecto, tema fundamental a desarrollar en la investigación.

1.2.3. Pentaho Data Integration

El *Pentaho Data Integration* fue creado por Matt Casters en el año 2001 para darle solución a los problemas presentados durante la construcción de almacenes de datos. Esta herramienta exhibe una colección de componentes en el submenú de entrada que se encuentran al lado izquierdo de su área de diseño, los cuales permiten la extracción de datos desde diferentes fuentes como se muestra a continuación:

- Entrada Access
- Entrada de Archivo CSV
- Des-Serialización desde Fichero
- Entrada Excel
- Entrada Fichero de Texto
- Entrada Tabla
- Entrada XBase
- Entrada XML
- Entrada XML Continuo
- Entrada de archive fijo
- Generar Filas
- Generar valor aleatorio
- Obtener Filas desde archivos Count
- Obtener datos desde XML
- Información del Sistema
- Entrada LDAP
- Entrada LDIF
- Lector ESRI Shapefile
- Obtener nombres de archivo
- Entrada desde Mondrian
- Entrada de la propiedad (5)

Entrada Excel que es el componente que servirá de guía durante la investigación para darle cumplimiento al desarrollo del nuevo componente, se encarga de extraer datos desde hojas de cálculo, para posteriormente ser transformados y cargados.

El PDI presenta una arquitectura basada en componentes, esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales y en la reutilización de código lo que permite probar el componente de forma individual antes de probar el producto completo y actualizar y agregar componentes sin afectar otras partes del sistema. El componente después de construido debe ser integrado al producto final mejorando la calidad de la aplicación con el paso del tiempo. Se decide escoger el PDI como herramienta a tratar por tener varias fucionalidades que facilitan el trabajo con diversas fuentes de datos y además porque es la herramienta utilizada en la UCI para realizar los procesos de integración de datos.

1.3. Entorno de desarrollo

La adecuada selección de las herramientas, tecnologías y metodología es esencial para lograr la calidad del componente a desarrollar. Por este motivo en este epígrafe se abordará sobre la metodología y herramientas necesarias, que permitan llevar el proceso sin dificultad.

1.3.1. Metodología de desarrollo de software: Extreme Programming (XP)

"Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar." Kent Beck. (6)

Ante la necesidad de guiar el desarrollo de software se definen las metodologías. Estas son las encargadas de definir que artefactos y actividades realizar en cada etapa del desarrollo, y que persona o rol queda asignado en la elaboración de cada una de estas actividades, según las características particulares del proyecto. Para proyectos grandes se pueden usar metodologías tradicionales o pesadas, y para proyectos pequeños metodologías ligeras o ágiles.

Las metodologías tradicionales mantienen un control riguroso de los procesos del proyecto, definiendo muy detalladamente las actividades a desarrollar, mientras que las otras se centran más en la interacción con el cliente y realizan un desarrollo diario, permitiendo la entrega en poco tiempo de pequeñas versiones. De esta forma el cliente estima el avance y decide si se están cumpliendo o no los objetivos iniciales para el cual fue concebido el proyecto. (6)

En el transcurso de la investigación se hará uso de la metodología ágil dado a que el componente a realizar es un producto pequeño que necesita ser supervisado en períodos cortos, permitiendo lograr la culminación satisfactoria en el tiempo establecido para ello. Algunas metodologías ágiles que existen son:

- *Crystal Clear*
- *SCRUM*
- *Extreme Programming (XP)*

Crystal Clear es una metodología cuya prioridad es mantener la seguridad del proyecto necesitando hasta ocho integrantes y ocho roles, de los cuales cuatro deben ser realizados por personas distintas. Sus estrategias, artefactos y técnicas no tienen que ser utilizados obligatoriamente, ya que permite hacer uso de las definidas por otras metodologías ágiles. Esto la convierte en una

metodología cambiante en dependencia del proyecto a realizar, permitiéndole al equipo de desarrollo crear su propia metodología. (7)

En la metodología SCRUM se pueden apreciar tres roles, uno de ellos, es el equipo de desarrollo que puede estar conformado por siete personas, en ocasiones por el gran número que integran el equipo es necesario hacer más de uno. En esta metodología al terminar una tarea se actualiza el tiempo de trabajo estimado para las tareas restantes y si una tarea no es necesaria, simplemente se elimina. Durante el desarrollo se realizan varias reuniones entre los integrantes del equipo para debatir los adelantos, los diversos cambios a realizar, la planificación de la entrega y las estimaciones de costos. Esto lleva consigo que no se cuente en esta metodología con ninguna práctica de desarrollo de *software*, dejando al equipo toda la potestad para hacer cambios y mejoras a las herramientas y prácticas a utilizar durante todas las iteraciones. (8)

Por su parte XP es una metodología que se centra en mantener las buenas relaciones entre los miembros del equipo, de tal forma, que todos directamente forman parte del desarrollo del proyecto (incluyendo al cliente o usuario). De esta forma, se logra la completa satisfacción del cliente, ya que es capaz de dar opiniones durante el proceso. XP se usa en proyectos donde los requisitos suelen ser muy cambiantes y se basa en la reutilización del código desarrollado. (9)

Esta metodología consta en su ciclo de vida de cuatro fases: Planeación, Diseño, Codificación y Pruebas. (Ver Fig.2)

Fig. 2 Fases de la metodología XP

Planeación: En la fase de planeación se crean las Historias de Usuarios (HU) que son escritas por el cliente. Las mismas estarán compuestas por las características y funcionalidades del sistema a implementar. Además se asigna la prioridad a las historias y se decide cuáles se implementan en cada iteración.

Diseño: En la fase de diseño como parte del proceso de XP, se crean las tarjetas CRC que identifican y organizan las clases orientadas a objetos que serán usadas.

Codificación: Después de realizadas las HU, culminada la fase de diseño y la planificación de las pruebas que se realizarán al código, para que el desarrollador se centre en lo que debe implementar, entonces se pasa a la fase de codificación. La misma se debe realizar en pareja, uno se centra en la codificación de una parte particular del diseño, mientras el otro se asegura de que se cumplan los estándares de codificación y que el código implementado, coincida con el diseño general especificado. Durante la codificación el equipo va realizando la integración continua del código, lo que facilita encontrar errores en etapas tempranas del desarrollo.

Pruebas: La metodología XP realiza pruebas al código, conocidas como pruebas unitarias. Estas pruebas se realizan durante todo el ciclo de desarrollo lo que indicaría el progreso del sistema y con ello poder darse cuenta a tiempo de los errores que se van generando. Se realizan además las pruebas de aceptación, también conocidas como pruebas del cliente, estas pruebas se derivan de la historias de usuarios que se han implementado y se enfocan en la funcionalidad del sistema. (10)

XP establece siete roles de trabajo, entre los cuales se destacan 3 en el avance del proyecto: el programador que es el encargado de crear el código del componente, el cliente que es el responsable de especificar las HU y el Plan de iteraciones, y el encargado de pruebas que lleva todo el proceso de pruebas del componente a implementar.

En la investigación se usará XP ya que en el proyecto a desarrollar el cliente forma parte del equipo de desarrollo, realizando cambios en el componente durante todo el proceso. Además porque esta metodología utiliza un número pequeño de artefactos lo que permite simplificar el tiempo, el costo y requiere de un pequeño grupo de desarrollo para la aplicación de la metodología. Es fácil de

implementar y de aprender por lo que puede ser utilizada por equipos de desarrollo con poca experiencia.

1.3.2. Lenguaje de modelado: Unified Modeling Language (UML)

El modelo de negocios es el proceso que permite la representación gráfica de los procesos de análisis de un problema determinado, ayudando a visualizar el diseño y a hacerlo más accesible para otros. Incluye lenguajes de modelado que facilitan la realización de la modelación del sistema, dentro de los que se destaca Lenguaje Unificado de Modelado (por sus siglas en inglés UML) que se utiliza en la representación y modelación de la información, principalmente en la fase de análisis y diseño de un proyecto. Sus funciones se resumen en visualizar, especificar, construir y documentar un sistema, para ello se apoya en la utilización de diagramas. (11)

Se decide utilizar UML como lenguaje de modelado ya que permite especificar la estructura y el comportamiento del sistema usando clases y las relaciones entre ellas. Además puede ser utilizado por cualquier metodología de análisis y diseño orientada a objetos[3].Es la forma utilizada para representar las actividades del negocio permite reducir los costos, aumenta la calidad, automatiza determinados procesos y permite generar código a partir de los modelos y viceversa. Además se utiliza ya que es el lenguaje de modelado de sistemas más conocido y utilizado en la actualidad, en el mundo y en la UCI.

1.3.3. Herramienta CASE para UML: Visual Paradigm for UML 8.0.

Las herramientas *CASE* para UML es la forma de gestionar el modelado de sistemas con diagramas. Dentro de las se puede mencionar el *Visual Paradigm*, la cual será usada durante todo el ciclo de vida del sistema, proporcionando herramientas que facilitan el modelado mediante la utilización de UML. Esta herramienta admite realizar reutilización de modelos y correcciones sintácticas, lo que permite hacer diagramas correctos, además aprueba el trabajo en grupo ya que proporciona herramientas que posibilitan la compartición e integración con otras herramientas, y generan códigos evitando así errores de codificación.

[3] Es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento.

1.3.4. Lenguaje de programación: Java

Un lenguaje de programación es un conjunto de reglas sintácticas y semánticas utilizadas para crear programas que controlen el comportamiento físico y lógico de una máquina, así como su estructura y el significado de sus elementos. Dentro de estos lenguajes se encuentra *Java*.

Java es un lenguaje de programación de propósito general que permite la realización de aplicaciones de diversos tipos. Es orientado a objetos ya que soporta las características encapsulación, herencia y polimorfismo. Es robusto ya que la comprobación de tipos de datos ayuda a detectar errores en el momento en que se producen, y seguro dado a que el código pasa a través de un verificador que comprueba el formato de los fragmentos de código y detecta fragmentos de código ilegal, que falsean punteros, violan derechos de acceso sobre objetos o intentan cambiar el tipo o clase de un objeto.

Todas las características expuestas lo convierten en el indicado para llevar el desarrollo del componente a realizar. Además se tiene en cuenta ya que es el lenguaje de programación usado por la herramienta PDI.

1.3.5. Entorno de desarrollo integrado (IDE): Eclipse Helio

Un IDE, es un sistema que facilita el trabajo del desarrollador del sistema el cual integra la edición de código, la compilación, la depuración y la construcción de interfaz gráfica (por sus siglas en inglés GUI). Entre los IDE que existen se encuentra *Eclipse*, utilizado para soportar la construcción e integración de herramientas de desarrollo para la creación de varias aplicaciones. Es multiplataforma y cuenta con una arquitectura abierta y extensible, que le permite integrar nuevas funcionalidades. Utiliza *Java* como lenguaje de programación. *Eclipse* permite además la programación en equipo poniendo los proyectos en un repositorio que pueden ser accedidos por otros desarrolladores.

Se escoge *Eclipse* para el desarrollo del componente por todas las características mencionadas anteriormente y por ser el lenguaje de creación de la herramienta PDI. Al añadir un componente es más adecuado utilizar el mismo IDE que usar otro nuevo, evitando así los problemas de compatibilidad y facilitando además el proceso de integración del componente a la herramienta.

Conclusiones del capítulo

A partir de la investigación realizada en el presente capítulo para dar a conocer los componentes de la herramienta PDI, las diferentes tecnologías, herramientas, entorno de programación y metodología para la realización del proceso de desarrollo del sistema se concluye que:

- Para guiar el desarrollo se seleccionó la metodología XP, por permitir una retroalimentación frecuente, entre el cliente y el equipo de desarrollo.
- Como lenguaje de modelado el UML, y *Visual Paradigm* 8.0 como herramienta CASE en su versión 8.0, por ser una herramienta muy completa y fácil de usar.
- Como lenguaje de programación *Java*, por ser la herramienta utilizada por PDI y *Eclipse* Helios como entorno de desarrollo, por ser multiplataforma y ofrecer servicios reutilizables comunes a las aplicaciones de escritorio.

Capítulo 2: Planificación y diseño del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja.

2.1. Introducción

En este capítulo se muestra todo lo relacionado con la planificación y diseño del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja, y se abordan una serie de elementos que hacen referencia a sus características y al levantamiento de requisitos tanto funcionales como no funcionales. Se definen las historias de usuarios y se realiza un diagrama de clases del diseño que se utiliza como complemento de la metodología definida. Se establece un modelo de dominio para comprender mejor el problema a tratar y se exponen las tarjetas CRC, las tareas de ingeniería, el plan de iteraciones y una estimación de la duración de las tareas de ingeniería, garantizando una correcta implementación. Además se específica la arquitectura del componente y los patrones de diseño que se usan para lograr una visión de las funcionalidades del sistema.

2.2. Descripción del negocio

La herramienta PDI es utilizada para el desarrollo de los procesos de integración de datos. La misma cuenta con componentes que se encargan de configurar la fuente de origen, de donde se obtendrán los datos que posteriormente pueden ser transformados y cargados en una amplia variedad de estructuras de datos. Los componentes que se encargan de extraer los datos desde hojas de cálculo tienen dificultades con el manejo de ficheros que presenten celdas combinadas y los datos en hojas bloqueadas.

2.3. Modelo de dominio

La metodología XP no establece una habilidad para definir el negocio, por lo que es necesario realizar un modelo de dominio para comprender el entorno organizacional que se estudia. A pesar

de no proponer artefactos en este sentido, la metodología es flexible y puede apoyarse en algunos que simplifiquen y apoyen el proceso.

Un modelo de dominio es una forma de representar gráficamente los conceptos de los que el futuro sistema debe de hacer seguimiento, llevando un orden lógico en dicha representación. Es el mecanismo fundamental para comprender el dominio del problema y se describe mediante diagramas, específicamente mediante diagramas de clases. Es una representación de las clases conceptuales o entidades del mundo real que ayuda a los usuarios, clientes y desarrolladores a utilizar un lenguaje común para poder entender el contexto en que se desarrolla el sistema. (13)

En el modelo de dominio mostrado se representan los pasos a partir de que el usuario accede a la herramienta con que se trabaja, la cual cuenta con varios componentes dentro de los que se encuentra el de entrada excel, el cual permite obtener y configurar los datos desde diversas fuentes, para finalmente transformados y cargados, ser mostrados al usuario. (Ver Fig. 3)

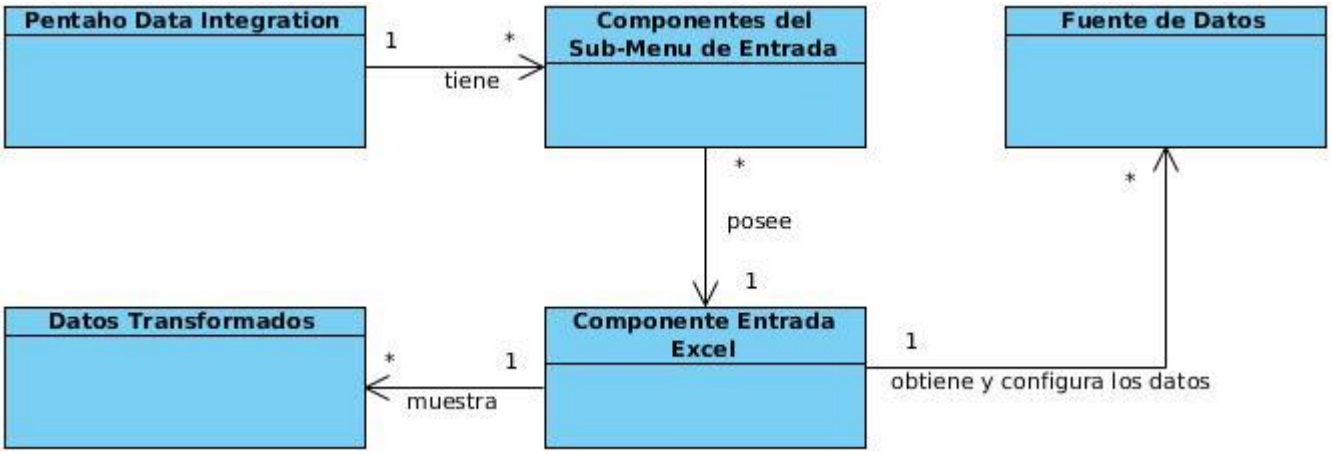


Fig. 3 Diagrama Modelo de dominio

Conceptos del Diagrama Modelo de Dominio

- **Pentaho Data Integration:** Herramienta de integración de datos PDI.
- **Componentes del Sub-Menú de entrada:** Funcionalidades con que cuenta la herramienta.
- **Componentes Entrada Excel:** Componente para la extracción de datos desde excel.
- **Fuente de Datos:** Fichero donde se encuentran los datos con los cuales se trabajará.

- **Datos Transformados:** Datos mostrados después de ser transformados y cargados.

2.4. Especificación de requisitos

Los requisitos de *software* son las características que el sistema a desarrollar debe poseer. Son definidos teniendo en cuenta las necesidades de los usuarios y de su área de trabajo específica. Estos requisitos pueden clasificarse en varios tipos, pero se destacan en la construcción del componente los requisitos funcionales y no funcionales. (14)

En la metodología utilizada se especifican los requisitos funcionales haciendo uso de las HU, pero para mayor organización de la investigación se decide hacer este proceso en 2 etapas. En la primera etapa, se definen los requisitos definidos para el componente con una breve descripción de sus funciones; y en la segunda etapa, se describen cada uno de los requisitos definidos a través de las HU.

2.4.1. Requisitos funcionales (RF)

Los requisitos funcionales describen las funcionalidades que el sistema debe permitir, y la manera que este reacciona ante determinadas entradas. (15) A continuación se muestran los requisitos funcionales identificados:

RF1. Examinar fichero excel: Permitir buscar un fichero o directorio.

RF2. Añadir fichero excel: Añadir el fichero a la lista de ficheros o directorios.

RF3. Eliminar fichero excel: Eliminar el fichero seleccionado de la lista de ficheros.

RF4. Seleccionar el formato del fichero excel: Permitir seleccionar el tipo de extensión que tendrán los ficheros que serán cargados.

RF5. Desbloquear fichero excel: Permitir seleccionar la opción de excel protegido e introducir la contraseña.

RF6. Obtener hoja(s) del fichero excel: Permitir seleccionar las hojas del fichero con el cual se trabaja.

RF7. Listar hojas a leer del fichero excel: Permitir especificar la fila inicial, columna inicial y número de filas del encabezado.

RF8. Eliminar filas vacías del fichero excel: Permitir seleccionar la opción de eliminar filas vacías a la hora de la salida o previsualización de los datos.

RF9. Detener al encontrar fila vacía dentro del fichero excel: Permitir seleccionar la opción de detener el proceso cuando se llega a una fila vacía.

RF10. Identificar celdas combinadas dentro del fichero excel: Identificar cuando una celda es combinada.

RF11. Identificar encabezados por columna dentro del fichero excel: Identificar los encabezados que se encuentran dentro de celdas combinadas en una misma columna.

RF12. Identificar encabezados por intervalo dentro del fichero excel: Identificar los encabezados que se encuentran en un intervalo de columnas determinado donde existan celdas combinadas.

RF13. Extraer datos de ficheros excel bloqueados o protegidos**:** Extraer datos de hojas de cálculo bloqueadas o protegidas.

RF14. Obtener campos de cabecera del fichero excel: Mostrar los encabezados de la hoja seleccionada a partir de la fila inicial, la columna inicial y el número de filas del encabezado especificada en la Lista de hojas a leer del fichero excel.

RF15. Adicionar datos de varias hojas: Adicionar los datos de la segunda hoja a continuación de la primera y sucesivamente.

RF16. Obtener funciones: Mostrar funciones utilizadas dentro del fichero excel.

RF17. Previsualizar los datos de las filas: Mostrar los datos de la(s) fila(s) que se encuentren en el

rango especificado por el usuario.

2.4.2. Requisitos no funcionales (RNF)

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema y a menudo se aplican al sistema en su totalidad. (15) Son propiedades o cualidades que el sistema debe cumplir para lograr el correcto funcionamiento de la aplicación.

A continuación se muestran los requisitos no funcionales identificados en el sistema.

Software

RNF1. El componente puede ser integrado a cualquier PDI a partir de la versión 4.2.0.

RNF2. Es necesaria la máquina virtual de *Java* en su versión 1.5 o superior.

RNF3. La herramienta PDI se puede utilizar en versiones de los sistemas operativos que soporten la máquina virtual de *Java*; como: *Windows XP, Windows 7, Windows 8, Ubuntu 12.04 LTS*.

Hardware

RNF4. La herramienta debe usarse en un procesador: *Celeron*, 2.0GHz o versiones superiores.

RNF5. Se debe usar Memoria RAM de 1 GB o superior.

RNF6. Debe haber un espacio libre en disco duro de al menos 200 Mb.

Usabilidad

RNF7. El componente debe presentar opciones de pestañas permitiendo ser usado por cualquier usuario final independientemente del nivel de experiencia que tengan.

RNF8. Se debe lograr un diseño adaptable, con la capacidad de soportar nuevas funcionalidades o modificar las funcionalidades existentes.

RNF9. El componente no debe permitir la obtención de datos de ficheros excel cuyas funciones han sido modificadas en herramientas ofimáticas[4] de diferentes sistemas operativos.

RNF10. El componente funciona correctamente en los archivos realizados en versiones de *Microsoft Office Excel* inferior a la 2011.

[4] Conjunto de técnicas, aplicaciones y herramientas informáticas utilizadas para optimizar, automatizar y mejorar los procedimientos o tareas relacionadas con funciones de oficina.

2.5. Historias de Usuarios

Las HU son descripciones cortas de los requisitos funcionales del sistema, utilizadas con el objetivo de que los programadores puedan hacer una estimación del riesgo y el tiempo que conllevará la implementación de dicha HU (Ver Tabla 1). También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la HU. Tienen el mismo propósito que los casos de uso y las escriben los propios clientes en dependencia de las necesidades del sistema. (9) Durante la investigación se definieron HU por cada requisito funcional identificado, quedando un total de 17 HU (Ver Expediente de Proyecto).

Tabla 1 Historia de usuario Extraer datos.

Historia de Usuario	
Usuario: Carmen Bolivar Colomé **Ariel Linares Ylizastigui**	
Nombre historia: Extraer datos de ficheros excel bloqueados o protegidos.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1 semana	**Iteración asignada: 3**
Programador responsable: Ariel Linares Ylizastigui	
Descripción: Permite a la herramienta PDI extraer datos desde hojas de cálculo bloqueadas o protegidas, una vez introducida la contraseña del fichero.	

2.6. Lista de reserva del producto

En la lista de reserva del producto se ordenan los requisitos funcionales que el sistema debe cumplir según la prioridad de implementación que tengan. Además se definen los no funcionales a tener en cuenta para lograr el desarrollo y culminación del componente. En el caso de la investigación se tendrán 3 tipos de prioridad: Alta, Media y Baja. (Ver Tabla 2)

Tabla 2 Lista de reserva del producto

Orden de Prioridad	Requisito	Estimación	Estimado por
Prioridad Alta			

1	Permitir seleccionar la opción de excel protegido e introducir la contraseña.	1 semana	Programador
2	Permitir especificar la fila inicial, columna inicial y número de filas del encabezado.	1 semana	Programador
3	Identificar cuando una celda es combinada.	1 semana	Programador
4	Identificar los encabezados que se encuentran dentro de celdas combinadas en una misma columna.	1 semana	Programador
5	Identificar los encabezados que se encuentran en un intervalo de columnas determinado donde existan celdas combinadas.	1 semana	Programador
6	Extraer datos de hojas de cálculo bloqueadas o protegidas.	1 semana	Programador
7	Mostrar los encabezados de la hoja seleccionada a partir de la fila inicial, la columna inicial y el número de filas del encabezado especificada en la Lista de hojas a leer del fichero excel.	1 semana	Programador
8	Adicionar los datos de la segunda hoja a continuación de la primera y sucesivamente.	1 semana	Programador
9	Mostrar funciones utilizadas dentro del fichero excel.	1 semana	Programador

10	Mostrar los datos de la(s) fila(s) que se encuentren en el rango especificado por el usuario.	1 semana	Programador
Prioridad Media			
11	Permitir buscar un fichero o directorio.	1 semana	Programador
12	Añadir el fichero a la lista de ficheros o directorios.	1 semana	Programador
13	Eliminar el fichero seleccionado de la lista de ficheros.	1 semana	Programador
14	Permitir seleccionar el tipo de extensión que tendrán los ficheros que serán cargados.	1 semana	Programador
15	Permitir seleccionar las hojas del fichero con el cual se trabaja.	1 semana	Programador
16	Permitir seleccionar la opción de eliminar filas vacías a la hora de la salida o previsualización de los datos.	1 semana	Programador
17	Permitir seleccionar la opción de detener el proceso cuando se llega a una fila vacía.	1 semana	Programador
Prioridad Baja			

18	**Software:** La herramienta PDI se puede utilizar en las versiones 4.2.0, 4.2.1, 4.4.0 y 5.0.1; es necesaria la máquina virtual de *Java* en su versión 1.5 o superior y la herramienta PDI se puede utilizar en versiones de los sistemas operativos que soporten la máquina virtual de *java*; como: *Windows XP, Windows 7, Windows 8, Ubuntu 12.04LTS.*		
19	**Hardware:** Es necesario la utilización de un Procesador: *Celeron*, 2.0GHz o versiones superiores, se debe usar memoria RAM de 1Gb o superior y debe haber un espacio libre en disco duro de al menos 200 Mb.		
20	**Usabilidad:** El componente debe detallar las pestañas y opciones a utilizar permitiendo ser usado por cualquier usuario final independientemente del nivel de experiencia que tengan, se debe lograr un diseño adaptable, con la capacidad de poder soportar funcionalidades adicionales o modificar las funcionalidades existentes. El componente no permite obtener los datos de fichero excel que han soportado modificaciones (en cuanto a las		

	funciones utilizadas en el fichero) en versiones de *Microsoft Office* de diferentes sistemas operativos y el componente realiza todas sus funcionalidades en los archivos realizados en cualquier versión de *Microsoft Office Excel* inferior a la 2011.		
21	**Interfaz:** La interfaz estará compuesta por pestañas con un contenido claro y bien estructurado permitiendo la interpretación correcta de la información, lo cual facilita y acelera la utilización del sistema por parte del usuario.		

2.7. Plan de iteraciones

En un plan de iteraciones los desarrolladores y clientes establecen cuales HU serán seleccionadas, implementadas y probadas en cada iteración en un orden preestablecido. Además establecen los tiempos que tarda la implementación de las HU.

La cantidad de iteraciones (IN) a realizar para el desarrollo de las HU está determinada por la suma de los puntos de esfuerzo (PE) para cada una de ellas dividida por la velocidad de iteración del equipo (VIE).

IN=PE / VIE

PE= 84 días (11 semanas)

IN= 11/2.5

IN =4.4

La velocidad de iteración del equipo (VIE) se obtiene dividiendo la cantidad de desarrolladores (CD) entre el factor de dedicación (FD) al proyecto (en el caso de la presente investigación es de 4 [100%]) y multiplicado por el tiempo de duración máximo de una iteración (DMI) (en el caso de la presente investigación es de 10 días máximo).

VIE= (CD / FD)*DMI

CD=1, FD =100% (4),

DMI =10.

VIE= 0.25*10

VIE=2.5

Por lo anterior expuesto, se definen en el desarrollo de la aplicación cuatro iteraciones, las cuales se detallan en la siguiente tabla:

Tabla 3 Plan de iteraciones

Iteración	Descripción de la iteración	Orden de la HU a implementar	Duración total
1	En esta iteración se realizarán 4 HU de prioridad alta, estas HU se encargarán de identificar: cuando una celda es combinada, los encabezados que se encuentran dentro de celdas combinadas en una misma columna, los encabezados que se encuentran en un intervalo de columnas determinado donde existan celdas combinadas y extraer datos de hojas de cálculo bloqueadas o protegidas.	10-11-12-13	3 semanas
2	En esta iteración se realizarán 4 HU de prioridad alta, estas HU se encargarán de: mostrar los encabezados de la hoja seleccionada a partir de la fila inicial, la columna inicial y el número de filas del	14-15-16-17	3 semanas

	encabezado especificada en la lista de hojas a leer del fichero excel, adicionar los datos de la segunda hoja a continuación de la primera y sucesivamente, visualizar funciones utilizadas dentro del fichero excel y mostrar los datos de la(s) fila(s) que se encuentren en el rango especificado por el usuario.		
3	En esta iteración se realizarán 5 HU, 2 de prioridad alta y 3 de prioridad media. Estas HU se encargarán de: permitir seleccionar la opción de excel protegido e introducir la contraseña, seleccionar las hojas del fichero con el cual se trabaja, especificar la fila inicial, columna inicial y número de filas del encabezado, seleccionar la opción de eliminar filas vacías a la hora de la salida o previsualización de los datos y seleccionar la opción de detener el proceso cuando se llega a una fila vacía.	5-6-7-8-9	3 semanas
4	En esta iteración se realizarán 4 HU de prioridad media. Estas HU se encargarán de: buscar las fuentes de datos, añadir el fichero a la lista de ficheros o directorios, eliminarlo de la lista y permitir seleccionar el tipo de extensión que tendrán los ficheros que serán cargados.	1-2-3-4	3 semanas

2.8. Diagrama de clases del diseño

Un diagrama de clases del diseño es la estructura estática de un sistema que permite describir gráficamente las especificaciones de las clases del *software* y las interfaces en una aplicación. Además visualiza las relaciones entre las clases de dicho sistema. (Ver Fig.4)

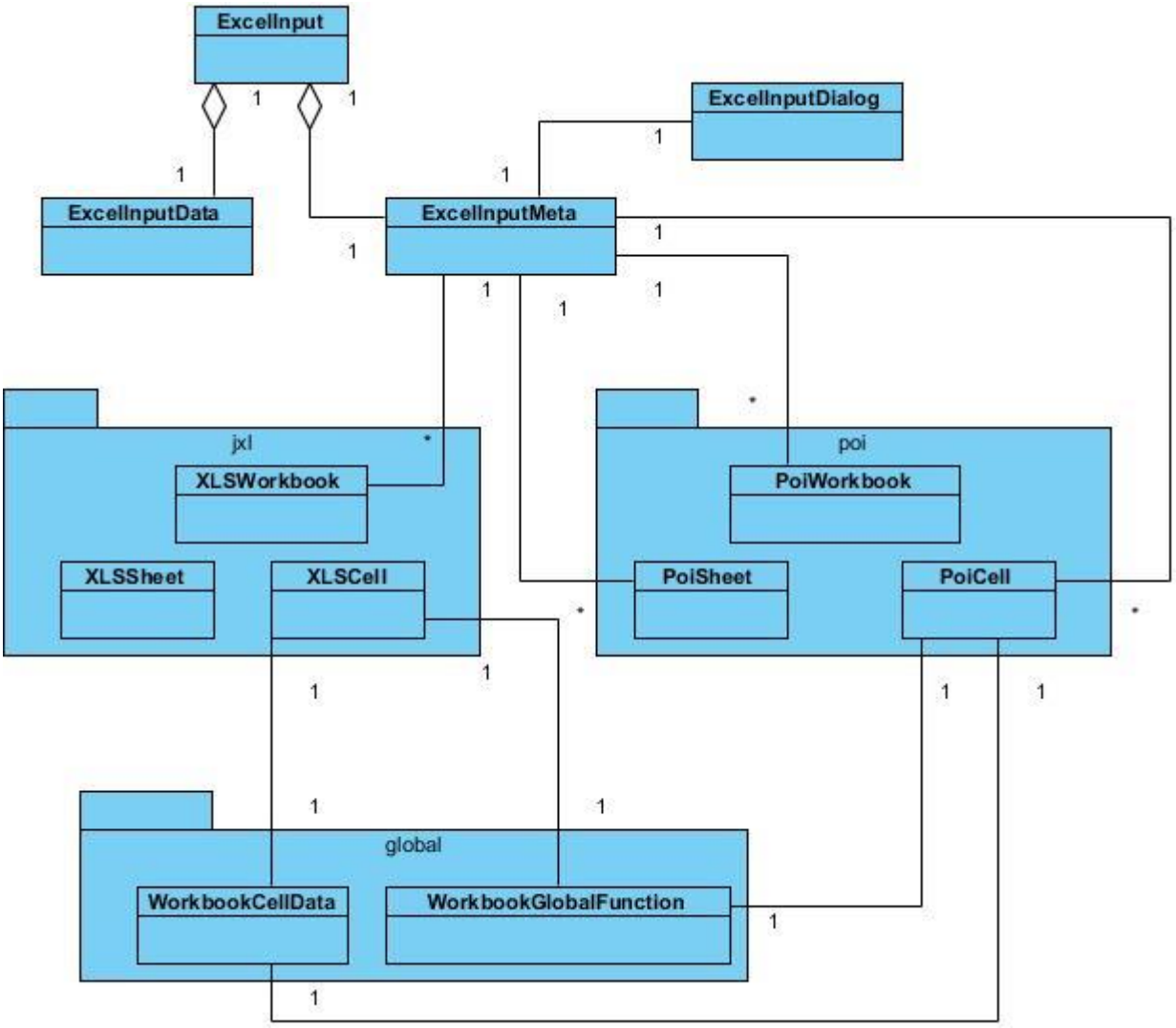


Fig. 4 Diagrama de clases del diseño.

2.7.1. Descripción de las clases del diseño.

En el diagrama de clases del diseño se ven evidenciadas las clases esenciales del componente realizado. Todos los componentes que se generen para ser integrados a la herramienta PDI deben contar al menos con 4 clases fundamentales.

ExcelInput: Clase encargada de asignarle responsabilidades a las clases *ExcelInputData* y *ExcelInputMeta*, mediante la creación de instancias de las mismas.

Tabla 4 Descripción de la clase ExcelInput

Nombre: ExcelInput
Tipo de clase: Controladora
Atributo
<<private>> meta : ExcelInputMeta <<private>> data : ExcelInputData
Operación
<< public >> fillRow(arg0,arg1:ExcelInputRow):Object << public >> checkType (): void << public >> processRow (): boolean << public >> handleMissingFile (): void << public >> gestRowFormWorkbooks (): Object << public >> isLineEmpty (): boolean << public >> jumpToNextFile (): void << public >> initErrorHandling (): void << public >> initReplayFactory (): void << public >> init (): boolean << public >> dispose (): void

ExcelInputData: trabaja con cada uno de los parámetros fundamentales del fichero excel como son: el nombre de la hoja, la cantidad de hojas, nombre del libro de trabajo.

Tabla 5 Descripción de la clase ExcelInputData

Nombre: ExcelInputData
Atributo

<<public>> previousRow: Object << public >> maxfilelenght: int << public >> maxsheetlength: int << public >> files << public >> filenr: int << public >> file << public >> workbook << public >> sheetnr: int << public >> sheet << public >> rownr: int << public >> colnr: int
Operación
<<>> ExcelInputData()

ExcelInputMeta: encargada de trabajar con cada uno de los metadatos del fichero introducidos por parámetros en la clase *ExcelInputDialog*, es decir con los datos obtenidos dentro del excel.

Tabla 6 Descripción de la clase ExcelInputMeta

Nombre: ExcelInputMeta
Atributo

<<private>> fileName : String <<private>> fileMask: String <<private>> fileField : String <<private>> sheetName: String <<private>> startRow: int <<private>>startColumn: int <<private>>sheetField: String <<private>>startsWithHeader: boolean <<private>>ignoreEmptyRows: boolean <<private>>rowNumberField: String <<private>>rowLimit: long <<private>>field: ExcelInputField
Operación
<<public>> readData(arg0:Node):void << public >> getFields (): void << public >> getXML (): String << public >> getFilePaths ():String << public >> getFileList () << public >> getEmptyFields () << public >> getUsedLibraries ():String << public >> readAllSheets ():boolean << public >> exportResources ():String << public >> getStepMetaInjectionInterface ()

ExcelInputDialog: realiza todas las interfaces visuales.

Tabla 7 Descripción de la clase ExcelInputDialog

Nombre: ExcelInputDialog
Tipo de clase: Interfaz
Atributo

<<private>> wSheetTab
<<private>> wFieldsTab
<<private>> wSheetComp
<<private>> wFieldsComp
<<private>> fdFileComp
<<private>> fdSheetComp
<<private>> fdContentComp
<<private>> fdFieldsComp
<<private>> wlFilenameList
<<private>> wbGetSheets
<<private>> fdHeader
<<private>> wlNoempty
<<private>> wlInclRownumField
<<private>> wlInclSheetsRownumField
<<private>> wlLimit
<<private>> wSpreadSheetType
<<private>> wbGetFields
<<private>> margin:int
<<private>> fdlAddResult
<<private>> wSizeFieldName
<<private>> fieldSelected: List
Operación
<<public>> open():String
<<public>> setFlags():void
<<public>> getData(arg0:ExcelInputMeta):void
<<private>> cancel():void
<<private>> ok():void
<<private>> getInfo(arg0:ExcelInputMeta):void
<<private>> addErrorTab ():void
<<private>> preview():void
<<public>> getSheets():void
<<public>> getFields():void
<<private>> clearfieldSelected():void

<<private>> showFiles():void
<<private>> setEncodings():void
<<private>> checkAlerts():void
<<private>> tagTab():void
<<private>> addAdditionalFieldsTab():void

Los paquetes JXL y POI contienen 3 clases *Workbook, Sheet y Cell,* en esta última es donde se realiza el tratamiento a cada uno de los tipos de ficheros excel, y a su vez utilizando las clases *WorkbookCellData* y *WorkbookGlobalFunction* del paquete Global.

La clase *WorkbookCellData* contiene cada uno de los datos de la celda como son: columna inicial, fila inicial, posición inicial, posición final, que posteriormente le servirán de guía a la clase *WorkbookGlobalFunction*, esta última clase construye una cadena donde se tendrán las funciones utilizadas en el fichero excel.

2.9. Tareas de ingeniería

Las tareas de ingeniería son tablas donde se especifican las tareas a realizar por cada historia de usuario descrita. Por cada HU de usuario descrita se realizó una Tarea de ingeniería, quedando un total de 17 Tareas de ingenierías.

En la tabla mostrada se describe la tarea de ingeniería cinco nombrada Implementar la funcionalidad *isXlsPasswordProtected* correspondiente a la HU con el mismo número. Además se muestra el tipo de tarea, la fecha de inicio y fin, los puntos estimados, el responsable de realizar la tarea, así como una breve descripción de la misma. El resto de las tareas se encuentran descritas en el Expediente de Proyecto.

Tabla 8 Tarea de ingeniería Implementar *ExcelInput.*

Tarea de ingeniería	
Número de tarea: 5	**Número de Historia de Usuario**: 5
Nombre tarea: Implementar la funcionalidad *isXlsPasswordProtected*	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1 semanas
Fecha inicio: 24/02/2014	**Fecha fin:** 09/03/2014
Programador responsable: Ariel Linares Ylizastigui	
Descripción: Se implementa esta funcionalidad en la clase *PoiWorkbook* para permitir al usuario poder desbloquear un Excel desde la propia herramienta.	

2.10. Tarjetas CRC

Para realizar el diseño del sistema a implementar se usan las tarjetas CRC. En ellas se especifican las clases y funcionalidades que deben ser implementadas durante el desarrollo de la aplicación para de esta forma realizar el proceso con la mayor calidad posible logrando una mayor satisfacción del cliente. Durante la investigación se realizan un total de 12 tarjetas CRC, una por cada clase utilizada.

Las tarjetas CRC están divididas en 3 partes el nombre de la clase que se encuentra descrita en la sección Clase, en la sección Responsabilidades se describen las responsabilidades o funciones que debe realizar la misma y en la sección Colaboraciones donde se describen las clases con las que trabaja una clase determinada, con el objetivo de saber la relación que existe entre las clases, de tal forma que le permitan llevar a cabo sus responsabilidades.

En la siguiente tabla se muestra la tarjeta CRC *ExcelInput* en la cual se encuentran descritas sus responsabilidades y las colaboraciones que recibe de otras tarjetas CRC como *BaseStep* entre otras. El resto de las Tarjetas CRC se encuentran descritas en el Expediente de Proyecto.

Tabla 9 Tarjeta CRC ExcelInput.

Tarjeta CRC	
Clase: *ExcelInput*	
Responsabilidades	**Colaboraciones**
Esta clase lee los datos desde uno o más ficheros Excel. Las funcionalidades son: *fillRow*: Llenar fila para obtener los campos de cabecera. *checkType*: Comprueba el tipo de dato que contiene la celda. *processRow*: Verifica si una fila posee datos o no. *handleMissingFiles*: Maneja los archivos que faltan. *getRowFromWorkbooks*: Obtiene las filas de libros excel. *isLineEmpty*: Verifica si una celda del excel es vacía. *jumpToNextFile*: Pasa a procesar o analizar la siguiente hoja del excel. *initErrorHandling*: Adiciona nuevos tipos de errores (de tipo número y de tipo archivo) a una lista. *initReplayFactory*: Obtiene los datos del fichero replicados. *dispose*: Verifica si el excel posee datos para ser procesado.	*BaseStep* *StepInterface* *ExcelInputMeta***(Asociación)** *ExcelInputData***(Asociación)**

2.11. Arquitectura de *software*

La arquitectura de *software* es el diseño anticipado de la aplicación que permite a los desarrolladores guiar la construcción de la misma. Establece la estructura, funcionamiento e

interacción entre las partes del sistema y cubre todos los objetivos y restricciones de la aplicación. Además satisface los atributos de calidad del sistema, como las opciones de modificación, mantenimiento, seguridad y rendimiento.

2.11.1. Arquitectura basada en componentes

La arquitectura basada en componentes, es una arquitectura que se enfoca en la descomposición del diseño, en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. De esta forma, los programas pueden usar su funcionalidad sin revelar detalles del proceso, pueden ser más extensibles (se pueden agregar o reemplazar componentes) sin tener que compilar nuevamente todo el código, en caso de errores en un componente específico, no se afecta al resto de la aplicación y los componentes implementados pueden ser reusados por diferentes aplicaciones y tareas específicas. (16)

El elemento a desarrollar durante la investigación es una aplicación que va a hacer uso de las funcionalidades del componente Entrada Excel de la herramienta PDI para de esta forma realizar el nuevo componente con una modificación de las funcionalidades ya existentes y la agregación de nuevas que mejoren el trabajo con la herramienta. Este componente se ejecuta en la aplicación principal e interactúa por medio de la interfaz gráfica de la aplicación. Por lo que se define como arquitectura a utilizar la arquitectura basada en componentes. La misma soporta varios componentes permitiendo reducir los tiempos de ejecución lo cual ofrece rapidez, poco costo y mantenimiento. Además los componentes implementan un interface bien definida para proveer la funcionalidad definida permitiendo el desarrollo sin impactar otras partes del sistema.

En la imagen mostrada se representa la arquitectura del PDI la cual integrar varios componentes como son Entrada Excel, Entrada Tabla, Des-Serialización desde Fichero y Entrada Excel Complejo.

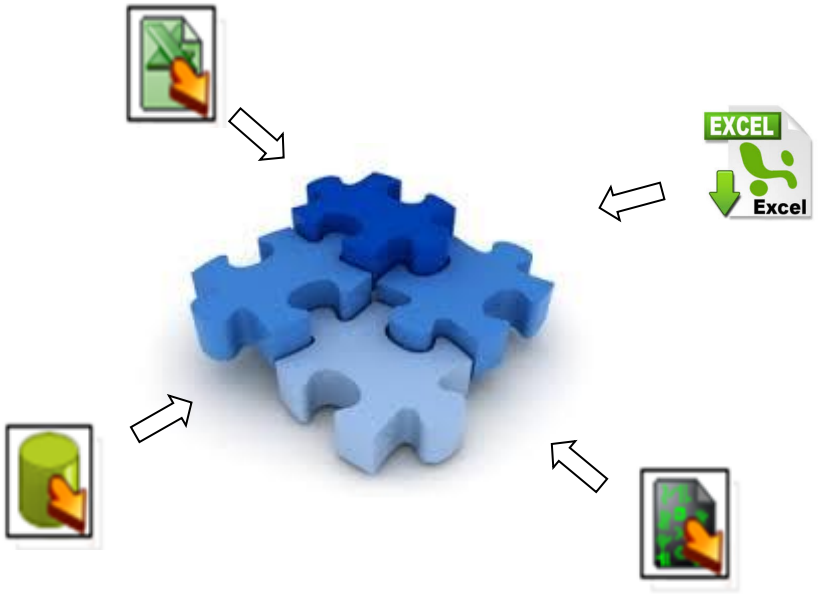


Fig. 6 Arquitectura basada en componentes del PDI.

2.11.2. Patrones de diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema de diseño particular dentro de un contexto específico. Existen patrones de diseño como es el caso del patrón GRASP y GOF. Con el uso de patrones de diseño se evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Permite formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño.

Patrones GRASP

Los patrones GRASP (Patrones Generales de Asignación de Responsabilidad) representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. (17) En el desarrollo de la investigación se destacan los siguientes patrones:

Experto: Es aquella clase que cuenta con la información necesaria para cumplir la responsabilidad de asignar trabajos al experto en información.

El uso de este patrón se pone en evidencia ya que cada conector tiene la responsabilidad de conectar con fuentes de datos diferentes. (Ver Fig.5)

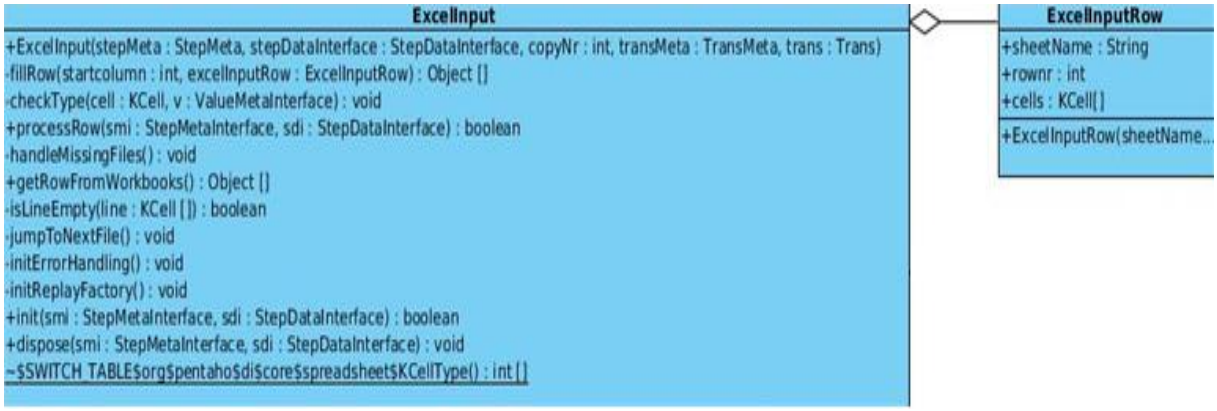


Fig. 7 Fragmento del diagrama de clases donde se evidencia el uso del patrón GRASP: Experto.

Creador: Dada la clase A y B, siendo B creador de los objetos A, este patrón es el responsable de asignarle a la clase B la responsabilidad de crear una instancia de clase A. (17)

El uso de este patrón se pone en evidencia entre las clases *ExcelInput*, *ExcelInputData* y *ExcelInputMeta*, ya que la clase *ExcelInput* tiene instancias creadas de las otras 2 clases. (Ver Fig.6)

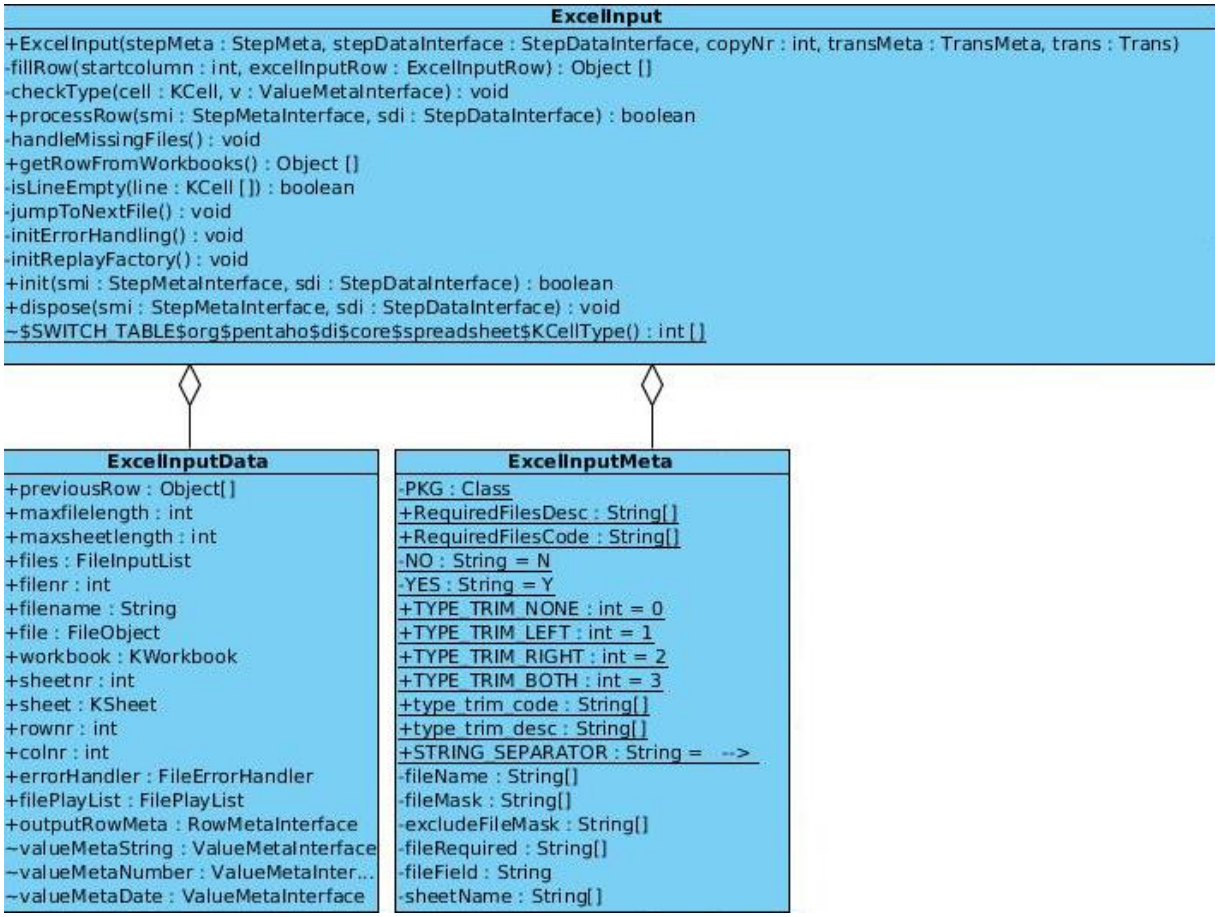


Fig. 8 Fragmento del diagrama de clases donde se evidencia el uso del patrón GRASP: Creador.

Alta cohesión: Asigna una responsabilidad de forma tal que la cohesión siga siendo alta. (17)

Bajo acoplamiento: Es el encargado de asignar una responsabilidad para conservar bajo acoplamiento. (17)

Los patrones alta cohesión y bajo acoplamiento son complementarios. Entre *ExcelInput*, ExcelInputRow y *KettleCellValuesException* existe alta cohesión y entre *ExcellInput* y *ExcelInputRow* Bajo acoplamiento. (Ver Fig. 7)

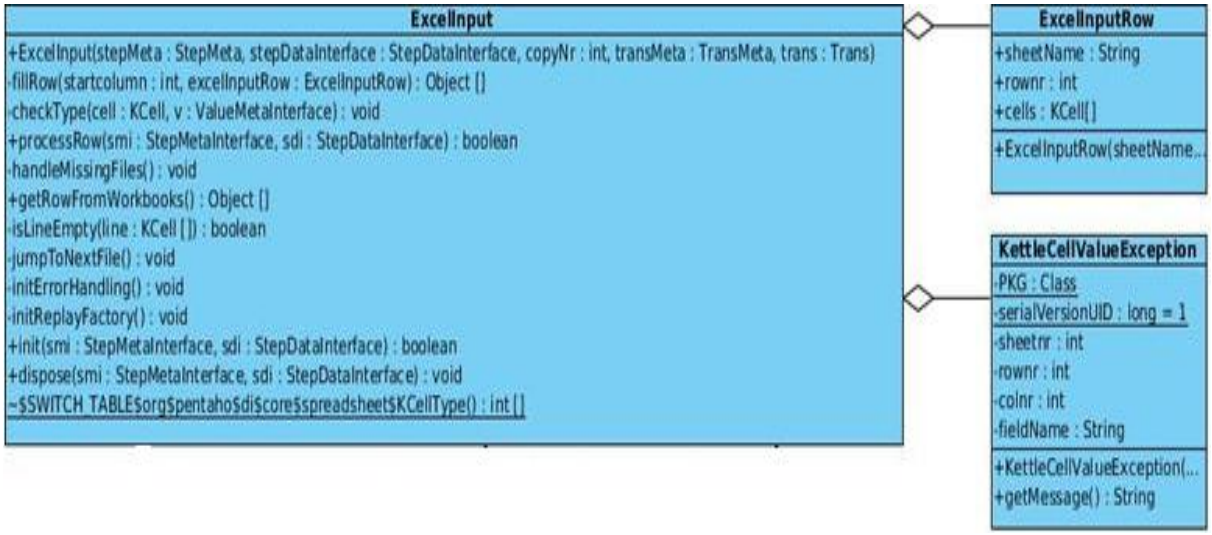


Fig. 9 Fragmento del diagrama de clases donde se evidencia el uso de los patrones GRASP: Alta cohesión y Bajo acoplamiento.

Controlador: Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. (17)

El patrón controlador se pone en evidencia en *ExcelInput* ya que esta clase asigna responsabilidades al resto de las clases. (Ver Fig. 8)

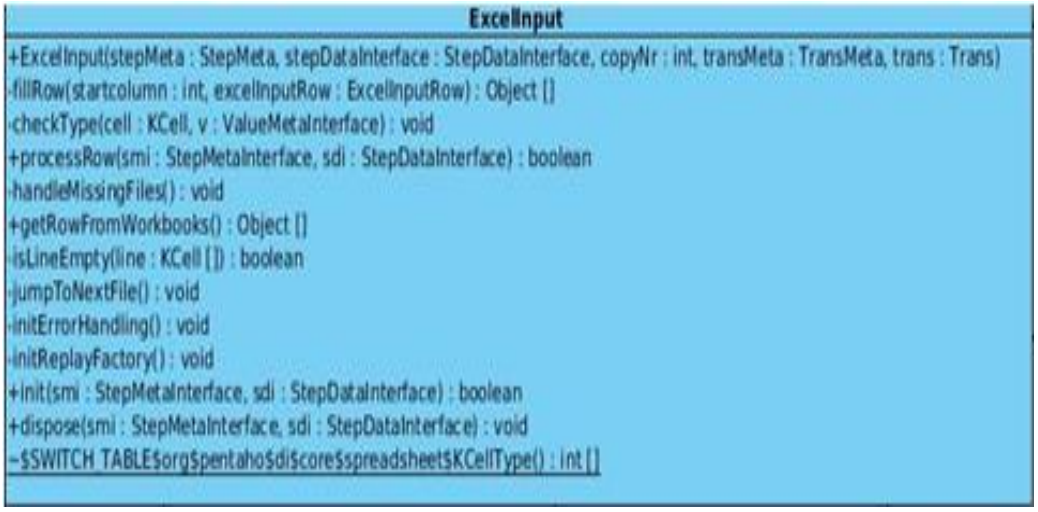


Fig. 10 Fragmento del diagrama de clases donde se evidencia el uso de los patrones GRASP: Controlador.

Patrones GOF

Los patrones de diseño el grupo de GOF (Pandilla de los Cuatro) se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. (18)

Creacionales: tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. (18) Dentro de esta clasificación existen muchos patrones, pero se destacan en el código del componente los siguientes:

- ***Object Pool*** (Conjunto de Objetos): Se obtienen objetos nuevos a través de la clonación. Utilizado cuando el costo de crear una clase es mayor que el de clonarla. Especialmente con objetos muy complejos. Se especifica un tipo de objeto a crear y se utiliza una interfaz del prototipo para crear un nuevo objeto por clonación. El proceso de clonación se inicia instanciando un tipo de objeto de la clase que se quiere clonar. (Ver Fig.9)

```
public Object clone() {
    ExcelInputMeta retval = (ExcelInputMeta) super.clone();

    int nrfiles = fileName.length;
    int nrsheets = sheetName.length;
    int nrfields = field.length;

    retval.allocate(nrfiles, nrsheets, nrfields);

    for (int i = 0; i < nrfields; i++) {
        retval.field[i] = (ExcelInputField) field[i].clone();
    }
```

Fig. 11 Fragmento del código donde se evidencia el uso de los patrones GOF: Conjunto de Objetos.

- ***Abstract Factory*** (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. (Ver Fig. 10)

```
public class ExcelInput extends BaseStep implements StepInterface
{
    private static Class<?> PKG = ExcelInputMeta.class;

    private ExcelInputMeta meta;

    private ExcelInputData data;
```

Fig. 12 Fragmento del código donde se evidencia el uso de los patrones GOF: Fábrica abstracta.

Estructurales: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos. (18)

Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema. (Ver Fig.11)

```
public ExcelInputDialog(Shell parent, Object in, TransMeta transMeta,
        String sname) {
    super(parent, (BaseStepMeta) in, transMeta, sname);
    input = (ExcelInputMeta) in;
}
```

Fig. 13 Fragmento del código donde se evidencia el uso de los patrones GOF: Fachada.

Proxy: Mantiene un representante de un objeto. (Ver Fig. 12)

```
public Object clone() {
    ExcelInputMeta retval = (ExcelInputMeta) super.clone();
```

Fig. 14 Fragmento del código donde se evidencia el uso de los patrones GOF: Proxy.

Comportamiento: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos. (18)

Command (Orden): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma. (Ver Fig.13)

```
public Object clone() {
    ExcelInputMeta retval = (ExcelInputMeta) super.clone();

    int nrfiles = fileName.length;
    int nrsheets = sheetName.length;
    int nrfields = field.length;

    retval.allocate(nrfiles, nrsheets, nrfields);
```

Fig. 15 Fragmento del código donde se evidencia el uso de los patrones GOF: Orden.

Memento (Recuerdo): Permite volver a estados anteriores del sistema. (Ver Fig.14)

```
public void getFields() {

    clearfieldSelected();

    RowMetaInterface fields = new RowMeta();
    ExcelInputMeta info = new ExcelInputMeta();
    getInfo(info);

    int clearFields = SWT.YES;
    if (wFields.nrNonEmpty() > 0) {
        MessageBox messageBox = new MessageBox(shell, SWT.YES | SWT.NO
                | SWT.CANCEL | SWT.ICON_QUESTION);
        messageBox.setMessage(BaseMessages.getString(PKG,
                "ExcelInputDialog.ClearFieldList.DialogMessage"));
        messageBox.setText(BaseMessages.getString(PKG,
                "ExcelInputDialog.ClearFieldList.DialogTitle"));
        clearFields = messageBox.open();
        if (clearFields == SWT.CANCEL) {
            return;
        }
    }
```

Fig. 16 Fragmento del código donde se evidencia el uso de los patrones GOF: Recuerdo.

Conclusiones del capítulo

En este capítulo se arriban a las siguientes conclusiones parciales:

- La especificación de los requisitos funcionales y no funcionales facilitaron el mejor entendimiento de las acciones a realizar
- Se hizo uso de las HU para la descripción de las funcionalidades y tarjetas CRC permitiendo de esta forma describir las clases.
- Se utilizaron cinco patrones de diseño GRASP y seis GOF, y se utilizó la arquitectura basada en componentes, decisión que posibilitó una mejor asignación de responsabilidades entre las clases y estructurar correctamente los componentes a implementar.
- Se realizó el proceso de planeación y diseño permitiendo crear las bases para el posterior desarrollo del componente.

Capítulo 3: Implementación y prueba del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja.

3.1. Introducción

En este capítulo se realiza la descripción de los elementos esenciales referidos implementación del componente a tratar y se realizarán las pruebas unitarias y las pruebas de aceptación, para ello se usarán técnicas de caja blanca y caja negra que permitirán darle cumplimiento o aplicación a las pruebas que se le desarrollarán al componente.

3.2. Implementación del componente para la extracción de datos desde hojas de cálculo con matriz de datos compleja.

Después de concluir la planificación y el diseño del sistema se pasa a la fase de codificación con el objetivo de tener una entrega real del producto, e ir avanzando de forma iterativa e incremental hacia la solución definitiva. En esta fase la metodología XP se centra en la importancia de la programación en pareja para llevar todo el proceso de implementación de las HU. Esta implementación tiene como objetivo principal desarrollar la arquitectura y el sistema como un todo, así como definir la organización del código.

El componente a desarrollar contendrá un total de cinco nuevas funcionalidades y será integrado al *Pentaho* a través de 4 pasos fundamentales.

1. Utilizar cuatro clases fundamentales *ExcelInput, ExcelInputData, ExcelInputMeta* y *ExcelInputDialog*.
2. Exportar el proyecto como .jar, de lo que se generan los archivos .jar , .xml y .png
3. Después de generado esos 3 archivos, se copian en una carpeta nueva con el nombre del proyecto. Esta carpeta será copiada luego dentro de la herramienta PDI en la dirección /plugins/step/.

4. Abrir el PDI, y se mostrará el proyecto integrado.

Cada interfaz del componente implementado contiene las funcionalidades fundamentales especificadas por el usuario en los requisitos funcionales .Además cuenta con una apariencia agradable y fácil de entender dispersa en un total de seis interfaces para cada una de las pestañas que complementan el componente (Ver Anexos), las que se describen a continuación:

Ficheros: En esta interfaz se ven los botones que permiten buscar un fichero en una dirección determinada, para posteriormente añadirlo a la lista de ficheros o directorios, y una vez añadido se puede eliminar de la lista ya que permite seleccionar esta opción.

Hoja(s): En esta interfaz el usuario obtiene las hojas del fichero excel con las cuales trabajará posteriormente, especificándoles la fila inicial, la columna inicial y el número de filas del encabezado. En caso de que el fichero esté protegido por contraseña, el usuario tendrá la opción de marcar el *checkbox* e introducir la contraseña para desbloquear el excel.

Contenido: En esta interfaz es donde se especifica la librería con la cual se trabajará según el tipo de fichero excel que haya sido cargado en la herramienta .Además se selecciona la opción eliminar filas vacías para eliminar filas vacías de la salida, y la opción de detener al encontrar una fila vacía para detener el proceso al encontrarse con filas con esta característica.

Manejador de Error: En esta interfaz es donde se selecciona la opción de saber si existen tipos estrictos de datos, además de contar con la opción de ignorar o no los errores encontrados para de esta forma poder proseguir con el trabajo.

Campos: En esta interfaz es donde se obtienen cada uno de los campos de cabecera y se detallan cada una de sus características: nombre, tipo, longitud, precisión, tipo de poda, repetir, formato, moneda, decimal y agrupamiento.

ExcelInputDialog.AdditionalFieldsTab.TabTitle: En esta interfaz se añaden a la previsualización de los datos los nuevos campos que se deseen conocer, como es el caso de: campo con el nombre del fichero, campo con el nombre de la hoja, campo con el número de fila de la hoja, campo con número de filas escritas, entre otros.

3.3. Estándar de codificación

Un estándar de codificación es la forma en que se escribe el código de un programa para facilitar su entendimiento, lectura y modificación por otros desarrolladores o miembros del equipo. (19) En el caso del componente a implementar es importante definir un estándar de codificación ya que la herramienta PDI está escrita en el lenguaje de programación *Java* y cuenta con componentes incorporados por varios desarrolladores. *XP* enfatiza la comunicación de estos desarrolladores a través del código, por lo cual es necesario seguir los estándares de programación que se especifican a continuación.

3.3.1. Asignación de nombres

En el lenguaje de programación utilizado se determinan reglas para nombrar los elementos del programa. Estas reglas definen que los nombres de las clases deben iniciar en mayúsculas y las variables en minúsculas y que para referirse a los nombres no se usarán diminutivos. (18) (Ver Fig.15)

```
public class Arbol {
    private Coordenada valor;

    private List<Arbol> subarbol;

    public Arbol(Coordenada valor) {
        this.valor = valor;
        subarbol = new LinkedList<Arbol>();

    }
```

Fig. 17 Bloque de código donde se evidencia el uso del estándar de codificación: Asignación de nombre

3.3.2. Sangría

La sangría muestra la estructura lógica del programa, y ayuda a identificar la información de los elementos de programación haciendo el código más fácil de leer y entender. Los saltos de líneas ocurrirán después de las comas y antes de los operadores. (18) (Ver Fig. 16)

```
public void preorden(List<Coordenada> lista) {

    lista.add(this.valor);

    for (int i = 0; i < this.subarbol.size(); i++) {
        subarbol.get(i).preorden(lista);
    }
}
```

Fig. 18 Bloque de código donde se evidencia el uso del estándar de codificación: Sangría

3.3.3. Comentarios

El uso de los comentarios hace que todo el código sea más fácil de trabaja. Estos comentarios deben ser incluidos antes de declaraciones de variables clave, los métodos, las clases y las estructuras largas, así como dentro de todos estos. Los comentarios en bloque abrirán con /* y cerrarán con */, una sola línea de comentario comenzará con //. (18) (Ver Fig.17)

```
/*
  Lee los datos del objeto ExcelInputMeta y los muestra en este diálogo.

              Para obtener los datos del objeto ExcelInputMeta.
 */


    int greater = greaterRange(coord); // Busca el mayor rango de columnas de cada fila combinada
```

Fig. 19 Bloques de código donde se evidencia el uso del estándar de codificación: Comentario

3.4. Pruebas del software

Las pruebas representan una revisión final de las especificaciones, del diseño y de la codificación del sistema convirtiéndose en el elemento fundamental para garantizar la calidad del mismo. En esta fase se detectan los problemas del sistema y su correcto funcionamiento para asegurar que el sistema cumpla con los requisitos y necesidades del cliente. Para validar el sistema se realizaron las pruebas propuestas por la metodología *XP*, las pruebas unitarias y las de aceptación. También se desarrollarán las pruebas funcionales y de integración con el objetivo de ir creando pequeñas versiones del producto y al mismo tiempo ir aplicando pruebas para descubrir errores asociados con la interfaz. (20)

3.4.1. Técnicas de pruebas del software

Las técnicas de pruebas del software son técnicas que facilitan una guía sistemática para diseñar pruebas que: comprueben la lógica interna de los componentes del sistema y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y el rendimiento.

Pruebas estructurales o Caja blanca

La prueba de caja blanca comprueba la lógica interna del programa, para ello debe acceder al código fuente (datos y lógica) del sistema. Trabaja con entradas, salidas y el conocimiento interno.

Para implementar este método se utilizan varias técnicas como es el caso del camino básico. Esta técnica permite obtener la complejidad lógica de un diseño para guiar la definición de un conjunto

básico de caminos de ejecución. Los casos de prueba obtenidos de este conjunto básico permiten que durante la prueba se ejecute al menos una vez cada sentencia del programa. (21) La complejidad lógica de un programa se calcula de la siguiente forma:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
2. La complejidad ciclomática V (G), de un grafo de flujo G, se define como: V (G)=A-N+2, donde A es el número de aristas del grafo y N es el número de nodos del mismo grafo.
3. La complejidad ciclomática V (G), de un grafo de flujo G, también se define como: V (G)=P+1, donde P es el número de nodos predicado9 contenidos en el grafo de flujo G.

Pruebas de funcionalidad o Caja Negra

El método de caja negra describe las pruebas que se aplican sobre la interfaz del software utilizando los casos de prueba. Con estos, se pretende demostrar que las funciones del software son operativas, se detectan funcionalidades incorrectas o ausentes, errores en la interfaz, se definen las entradas al sistema y los resultados esperados de estas. Son diseñados para validar los requisitos funcionales sin detenerse en el funcionamiento interno del programa. (22)

Existen varias técnicas para implementar el método de caja negra, una de ellas es la Técnica de la Partición de equivalencia, que es la encargada de dividir el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Se selecciona esta técnica ya que intenta dividir el dominio de entrada de un programa en un número finito de variables de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada variable es equivalente a una prueba realizada con cualquier otro valor de dicha variable.

Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Éstas se identifican examinando cada condición de entrada y dividiéndola en dos o más grupos. Se definen dos tipos de variables de equivalencia, las válidas (entradas válidas al programa) y las no válidas (valores de entrada erróneos). (23)

3.4.2. Pruebas unitarias

Las pruebas de unidad verifican el funcionamiento de partes del software, las cuales se prueban de forma independiente, detectan errores en los datos, lógica, algoritmos. Estas partes podrían ser funcionalidades individuales o un componente más grande formado por unidades muy relacionadas. Las pruebas de unidad se realizan con acceso al código fuente y con el soporte de herramientas de depuración, pudiendo implicar a los programadores que escribieron el código.

Aplicando la técnica de prueba caja blanca se comprueba el código fuente del sistema. Para ello se probaron las funcionalidades implementadas haciendo uso de ficheros de entradas con diversas estructuras, dando como resultado las salidas esperadas. Este proceso permitió probar que el código no presentaba errores.

A continuación se presenta la técnica de camino básico para la función **"***isXlsXPasswordProtected (String fileName)***"**.

```
public boolean isXlsXPasswordProtected(String fileName) {
    try {
        new XSSFWorkbook(new FileInputStream(fileName));  1
    } catch (EncryptedDocumentException e) {  2
        return true;  3
    } catch (Exception e) {  4
        return true;  5
    }
    return false;  6
}
```

Fig. 20 Función isXlsXPasswordProtected (String fileName)

Se identifica 1 nodo predicado, del cual se derivan más de un camino a seguir como se muestra a continuación:

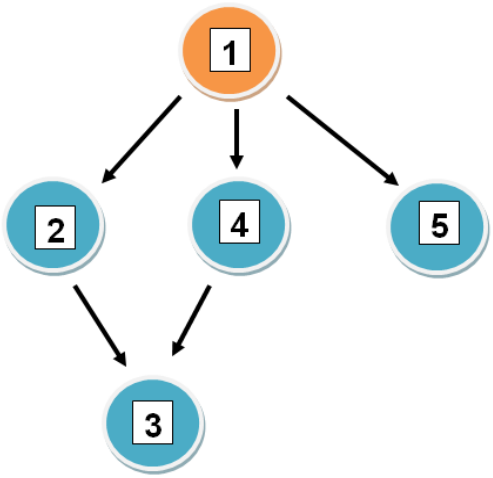


Fig. 21 Camino básico de la función isXlsXPasswordProtected

En el camino básico determinado, se aplica una de las tres formas para calcular la complejidad ciclomática. Se utilizó la fórmula V (G)= A-N+2. En la aplicación de dicha fórmula se obtuvo 5 aristas

y 5 nodos, por lo tanto: V (G)=5-5+2, quedando V (G)=2. Se puede comprobar que las variantes antes mencionadas pueden arrojar el mismo resultado.

Resultados de las pruebas de caja blanca

Como resultado de la prueba de caja blanca se tiene, que como complejidad ciclomática es 2, por tanto se deberán diseñar 2 casos de prueba. Los caminos elegidos son los que a continuación se muestran:

Camino1: 1, 2, 3.

Camino2: 1, 5.

Caso de prueba para el Camino1

Descripción

Este método pretende obtener un fichero xls y responder si está protegido por contraseña, para esto es cargado dicho fichero por el usuario. Una vez que arroja la excepción, devuelve la respuesta indicando que está protegido por contraseña.

Resultado esperado

Al ejecutarse este proceso se conocerá si dicho fichero está protegido por contraseña o no.

Caso de prueba para el Camino2

Descripción

Este método pretende obtener un fichero con formato xls y responder si está protegido por contraseña, para esto es cargado dicho fichero por el usuario. Una vez ejecutado dicho proceso se devuelve el resultado indicando que no está protegido por contraseña.

Resultado esperado

Al ejecutarse este proceso se conocerá si dicho fichero está protegido por contraseña o no.

3.4.3. Pruebas funcionales

En las pruebas funcionales el cliente realiza comparaciones con el sistema para comprobar que se satisfacen sus requisitos. Esta actividad de pruebas puede incluir o no a los desarrolladores del sistema.

Para la metodología utilizada al finalizar cada iteración de desarrollo se hacen iteraciones de pruebas las cuales pueden llegar hasta 3 iteraciones. Por cada una de las iteraciones de desarrollo especificadas Si en la tercera iteración de pruebas se siguen encontrando no conformidades se abortan las pruebas.

Luego de concluida la implementación de las 17 HU centralizadas en las cuatro iteraciones definidas y realizar las pruebas de caja negra mediante los casos de prueba asociados a cada HU, se comprobó el correcto funcionamiento del componente. Detectándose un total de 5 no conformidades en el desarrollo del componente a las que se les proporcionó seguimiento y fueron eliminadas a medida que se fue avanzando en el proceso de pruebas. (Ver Fig. 20)

No conformidades arrojadas por las pruebas

1. Al abrirse un excel protegido desde la herramienta se guarda la contraseña y posteriormente sin haberse guardado los cambios, se desea abrir este mismo excel y no es necesario introducir nuevamente la contraseña.
2. Cuando en una misma hoja del fichero existen varios campos con el mismo nombre, la herramienta elimina estos campos repetidos.
3. Cuando se abre el excel en vez de mostrar las funciones utilizadas en este y el resultado, muestra solamente los resultados de dichas funciones.
4. Cuando se ejecuta una transformación guardada, la cual utiliza excel protegidos; vuelve a solicitar la contraseña.
5. Cuando se desea previsualizar a partir de un encabezado determinado, esto no se ejecuta satisfactoriamente.

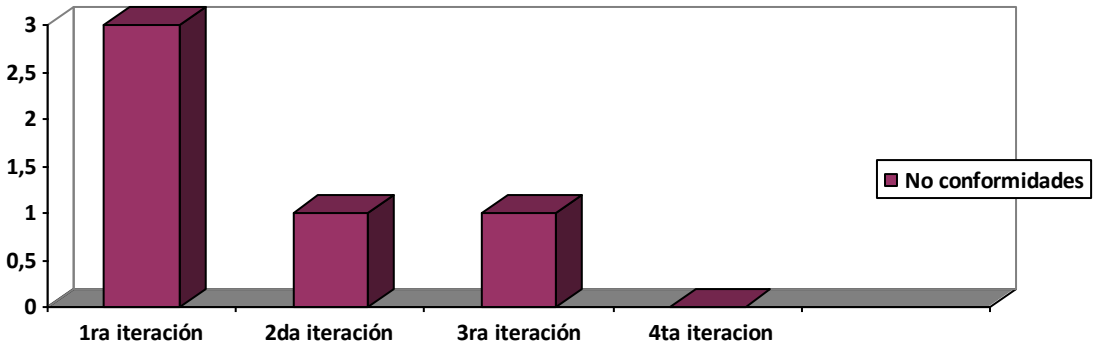


Fig. 22 No conformidades detectadas por cada iteración en las pruebas funcionales

3.4.4. Pruebas de integración

En las pruebas de integración se detectan errores de interfaces y relaciones entre componentes. Son realizadas por el implementador del sistema.

Para llevar a cabo las pruebas de integración se unieron en un solo conjunto todas las funcionalidades que le dan cumplimiento a los requisitos funcionales. Posteriormente se comprobó que este conjunto integrado con otros componentes podía ser capaz de realizar transformaciones con calidad y con un nivel superior a las transformaciones que se realizaban con el componente anterior. Con el nuevo componente se demostró una mejoría en cuanto a la cantidad de componentes a usar, velocidad de respuesta y tiempo.

A continuación se muestra el proceso de integración y los resultados arrojados.

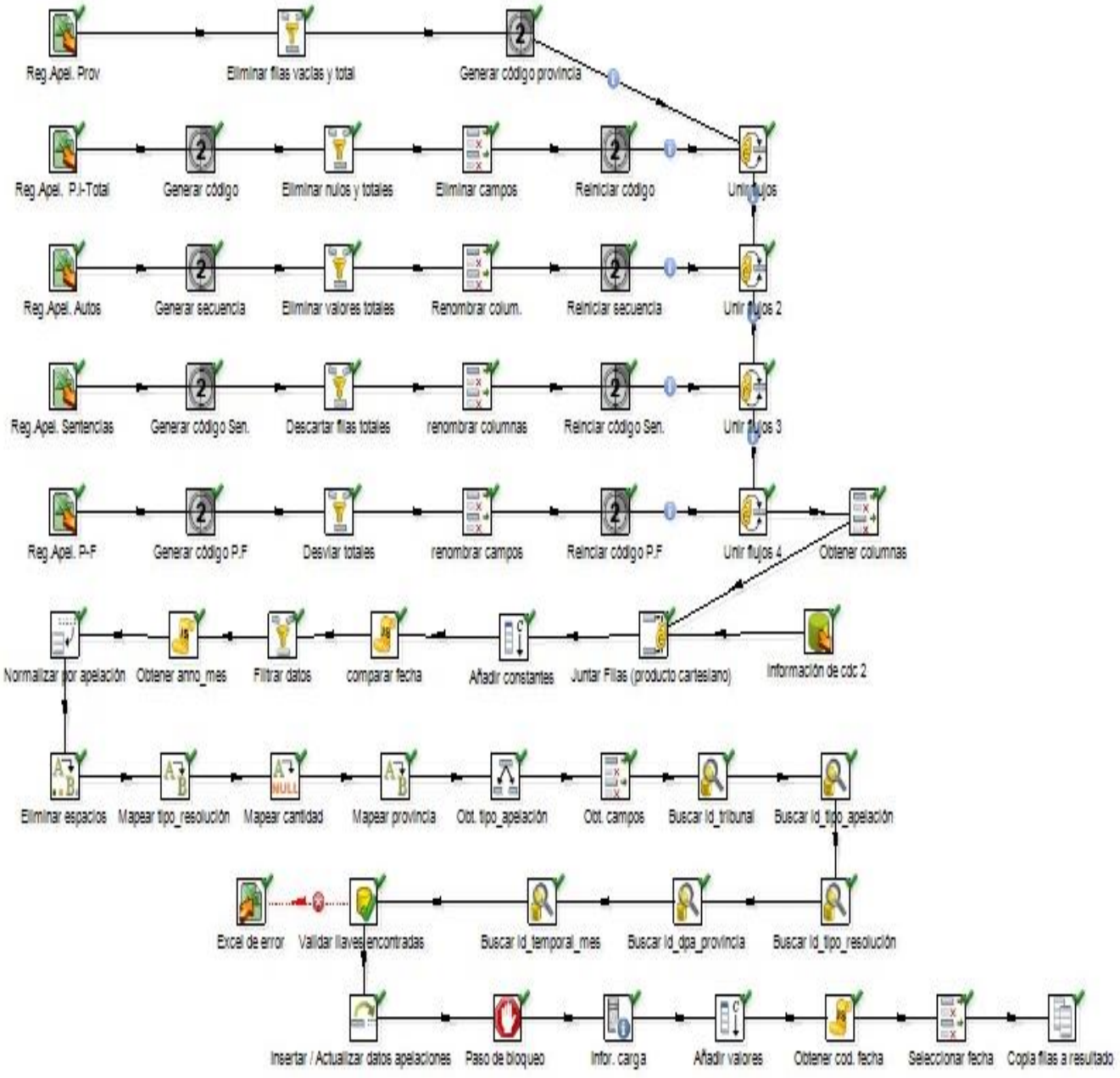


Fig. 23 Transformación donde se evidencia el uso del componente Entrada Excel

Execution Results

Execution History | Logging | Step Metrics | Performance Graph

#	Nombre paso	Leído	Escrito	Entrada	Salida	Activo	Tiempo	Velocidad (r/s)	Pri/E/S
7	Renombrar colum.	48	48	0	0	Finalizado	5.1s	9	-
16	Reiniciar secuencia	48	48	0	0	Finalizado	5.1s	9	-
15	Reiniciar código	48	48	0	0	Finalizado	4.7s	10	-
17	Reinciar código Sen.	48	48	0	0	Finalizado	5.1s	9	-
38	Reinciar código P.F	48	48	0	0	Finalizado	5.0s	10	-
8	Reg.Apel. Sentencias	0	51	51	0	Finalizado	5.1s	10	-
14	Reg.Apel. Prov	0	60	60	0	Finalizado	3.3s	18	-
37	Reg.Apel. P-F	0	51	51	0	Finalizado	4.0s	13	-
4	Reg.Apel. Autos	0	51	51	0	Finalizado	4.9s	10	-
21	Reg.Apel. PJ-Total	0	51	51	0	Finalizado	3.3s	15	-
47	Paso de bloqueo	768	1	0	0	Finalizado	6.4s	121	-
40	Obtener columnas	48	48	0	0	Finalizado	6.1s	8	-
45	Obtener cod. fecha	1	1	0	0	Finalizado	6.4s	0	-
51	Obtener anno_mes	48	48	0	0	Finalizado	6.3s	8	-
33	Obt. tipo_apelación	768	768	0	0	Finalizado	6.3s	121	-
43	Obt. campos	768	768	0	0	Finalizado	6.3s	121	-
42	Normalizar por apelación	48	768	0	0	Finalizado	6.3s	122	-
23	Mapear tipo_resolución	768	768	0	0	Finalizado	6.3s	122	-
32	Mapear provincia	768	768	0	0	Finalizado	6.3s	121	-
41	Mapear cantidad	768	768	0	0	Finalizado	6.3s	122	-
55	Juntar Filas (producto cartesiano)	49	48	0	0	Finalizado	6.3s	8	-
25	Insertar / Actualizar datos apelaciones	768	768	768	768	Finalizado	6.4s	121	-
46	Infor. carga	1	1	0	0	Finalizado	6.4s	0	-
5	Generar secuencia	51	51	0	0	Finalizado	5.1s	10	-
13	Generar código provincia	48	48	0	0	Finalizado	4.0s	12	-

Fig. 24 Resultados obtenidos con el uso del componente Entrada Excel

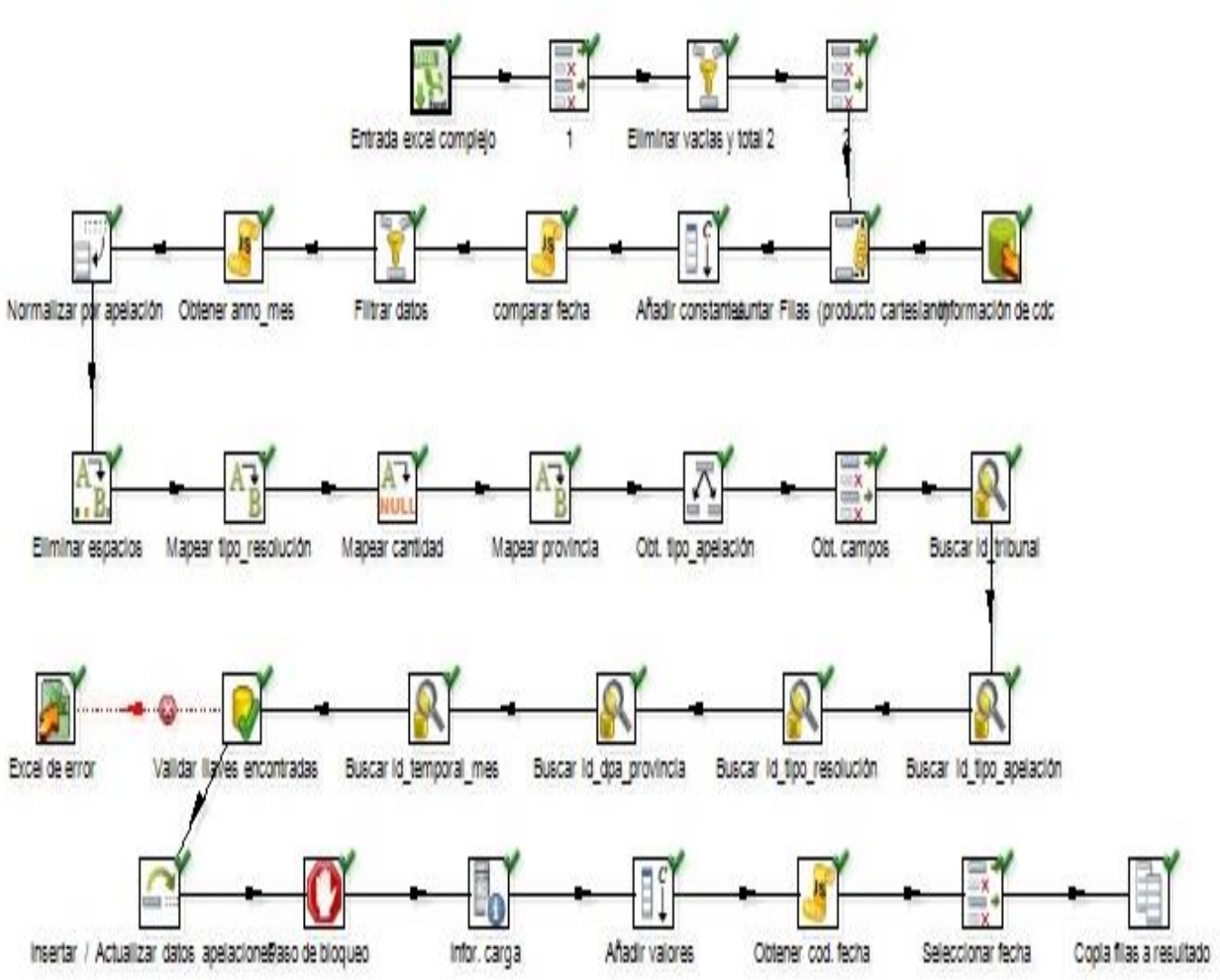


Fig. 25 Transformación donde se evidencia el uso del componente Entrada Excel Complejo

Execution Results

Execution History | Logging | Step Metrics | Performance Graph

#	Nombre paso	Leído	Escrito	Entrada	Salida	Activo	Tiempo	Velocidad (r/s)	Pri/E/S
28	1	51	51	0	0	Finalizado	1.0s	49	-
26	2	48	48	0	0	Finalizado	1.0s	46	-
29	Añadir constantes	48	48	0	0	Finalizado	1.2s	40	-
16	Añadir valores	1	1	0	0	Finalizado	1.5s	1	-
8	Buscar id_dpa_provincia	768	768	768	0	Finalizado	1.4s	554	-
9	Buscar id_temporal_mes	768	768	768	0	Finalizado	1.4s	547	-
6	Buscar id_tipo_apelación	768	768	768	0	Finalizado	1.3s	570	-
7	Buscar id_tipo_resolución	768	768	768	0	Finalizado	1.4s	561	-
5	Buscar id_tribunal	768	768	768	0	Finalizado	1.3s	581	-
21	Copia filas a resultado	1	1	0	0	Finalizado	1.5s	1	-
1	Eliminar espacios	768	768	0	0	Finalizado	1.2s	635	-
25	Eliminar vacías y total 2	51	48	0	0	Finalizado	1.0s	49	-
27	Entrada excel complejo	0	51	51	0	Finalizado	1.0s	50	-
10	Excel de error	0	0	0	0	Finalizado	1.4s	0	-
22	Filtrar datos	48	48	0	0	Finalizado	1.2s	40	-
18	Infor. carga	1	1	0	0	Finalizado	1.5s	1	-
4	Insertar / Actualizar datos apelaciones	768	768	768	0	Finalizado	1.4s	537	-
30	Juntar Filas (producto cartesiano)	49	48	0	0	Finalizado	1.2s	41	-
13	Mapear cantidad	768	768	0	0	Finalizado	1.2s	630	-
11	Mapear provincia	768	768	0	0	Finalizado	1.2s	627	-
2	Mapear tipo_resolución	768	768	0	0	Finalizado	1.2s	633	-
14	Normalizar por apelación	48	768	0	0	Finalizado	1.2s	636	-
15	Obt. campos	768	768	0	0	Finalizado	1.2s	623	-
12	Obt. tipo_apelación	768	768	0	0	Finalizado	1.2s	625	-
32	Obtener Variables	1	1	0	0	Finalizado	0.0s	200	

Fig. 26 Resultados obtenidos con el uso del componente Entrada Excel Complejo

3.4.5. Pruebas de aceptación

Las pruebas de aceptación en *XP* son pruebas de caja negra definidas por el cliente para cada HU, con el objetivo de indicar cuando las funcionalidades de una iteración han sido completadas exitosamente. En *XP* las pruebas de aceptación son responsabilidad del cliente ya que deben reflejar los requisitos y las funcionalidades que se quieren obtener en cada iteración.

El cliente realiza pruebas de aceptación no críticas con el objetivo de no afectar los tiempos estimados para cada HU. Si estas pruebas se culminan exitosamente generan una confianza en el cliente, de lo contrario se repiten dichas pruebas al final de la siguiente iteración aumentando el grado de criticidad de las mismas.

Con el objetivo de validar que el componente cumple las características planteadas por el cliente, se creó una comisión conformada por especialistas del departamento de almacenes de datos del centro DATEC encargada de someter la solución a una revisión técnica. De esta revisión se detectó una NC con un bajo impacto para la liberación, la cual fue resuelta en un breve período de tiempo. De esta forma se pudo probar que el componente está estable y listo para su uso. La comisión emitió el acta de liberación correspondiente, evidenciando que la herramienta cumple con todas las expectativas planteadas, los requisitos funcionales especificados y por la interfaz del componente.

3.4.6. ***Casos de pruebas basados en historias de usuario***

La metodología *XP* al concluir cada iteración realiza pruebas a las HU implementadas, con el objetivo de minimizar el número de errores del componente al concluir el desarrollo. Estas pruebas son aplicadas a los casos de prueba correspondientes por cada una de las HU definidas permitiendo comprobar que las funcionalidades del componente desarrollado se adaptan a las especificaciones funcionales y requisitos definidos por el cliente. Para llevar este proceso se le fueron realizando más de una prueba a aquellas iteraciones a las que inicialmente se le encontraron errores, con el fin de que al concluir esa iteración esta no tuviera fallas y poder enseñarle al cliente versiones cortas del producto.

Con la utilización de la técnica de caja negra los casos de prueba demuestran que las funciones del sistema son realizables y que la entrada y salida de los datos se realiza de forma correcta. (Ver Tabla 6)

Tabla 10 Caso de prueba para la HU 5 Extraer datos de ficheros Excel bloqueados o protegidos

Escenario	Descripción	V 1	V 2	V 3	V 4	Respuesta del sistema	Flujo central
EC 1.1 Extraer datos de ficheros Excel bloqueados o protegidos.	En este escenario se extraen datos de ficheros tipo excel que se encuentren bloqueados o protegidos correctamente.	V	V	V	V	El sistema muestra el resultado de la acción realizada.	1. El usuario selecciona el fichero que desea abrir. 2. El usuario pone la contraseña si el Excel es protegido 3. El sistema guarda la contraseña. 4. El

							sistema abre el Excel protegido.
EC 1.2 Extraer datos de *ficheros* bloqueados o protegidos que no sean excel.	En este escenario se extraen datos de ficheros tipo excel que se encuentren bloqueados o protegidos correctamente.	I	I	I	I	El sistema muestra un mensaje de error.	1. El usuario selecciona el fichero que desea abrir. 2. El sistema retorna errores y no abre el fichero seleccionado.

Conclusiones del capítulo

En este capítulo fueron probadas todas las funcionalidades definidas por el cliente lo que permitió arrojar a las siguientes conclusiones:

- Se implementó e integró el componente diseñado a través de los artefactos de la metodología *XP*, HU, Lista de reserva del producto, Tarjeta CRC, Plan de iteraciones.
- Se realizaron las pruebas unitarias, funcionales, de integración y de aceptación utilizando para ello la técnica de caja blanca camino básico y la técnica de caja negra partición de equivalencia.
- Se detectaron un total de 5 no conformidades en la primera iteración de pruebas, a las cuales se les fue dando solución en un corto período de tiempo.
- Las pruebas realizadas permitieron obtener un componente funcional y con calidad, integrado a la herramienta de integración PDI, que facilita el trabajo con matrices dinámicas en hojas de cálculo

Conclusiones Generales

La realización del presente trabajo posibilitó cumplir con los objetivos y tareas propuestas para su desarrollo, arribándose a las siguientes conclusiones:

1. La bibliografía consultada permitió seleccionar la metodología XP para guiar el desarrollo del componente, *Visual Paradigm* 8.0 para el modelado de los datos, *Eclipse Helio* como IDE de desarrollo, las tecnologías UML como lenguaje de modelado y *Java* como lenguaje de programación.
2. Se definieron 17 Requisitos Funcionales y 10 Requisitos No Funcionales que definieron las bases para el desarrollo del sistema
3. El componente desarrollado permitió a la herramienta *Pentaho Data Integration* manipular hojas de cálculo con matriz de datos compleja, eliminando las limitaciones que este tenía.
4. Las pruebas realizadas demostraron el correcto funcionamiento del componente, el cual fue aceptado por el cliente.
5. Durante las pruebas de integración, se constató que el componente desarrollado reduce los tiempos de ejecución de las transformaciones realizadas en comparación con el anterior, de un valor máximo de 6.4 segundos a 1.4 segundos. La cantidad de componentes a utilizar también disminuyó de 6 a 2 y la velocidad de respuesta de la transformación realizada fue más rápida con el componente desarrollado.

Referencias

1. **Pentaho Corporation.** Pentaho. [Online] 2005-2014. http://www.pentaho.com/product/data-integration.

2. **Talend.** Talend. [Online] 2006-2014. http://www.talend.com/resources/documentation.php.

3. —. Talend . [Online] 2006-2014. http://www.talendforge.org/components/.

4. **Clover.** Clover. [Online] https://www.clover.com/.

5. **Grupo ETL.** *Guía de componentes de Pentaho Data Integration.* La Habana : s.n.

6. **Beck, Kent.** *Metodologías XP y RUP. .*

7. **CrystalClear.** CrystalClear . [Online] http://www.crystalclear.com.ph/.

8. **Palacio, Juan.** *El modelo Scrum.* 2006.

9. **Letelier, Patricio y Penadés , Mª Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).*

10. **Pressman, Roger S.** Cap_04_Desarrollo_Agil_Parte_1. *Ingeniería de Software, un enfoque práctico. Quinta edición .* s.l. : McGraw-Hill Companies, 2002.

11. **Hernández Orallo, Enrique.** *El Lenguaje Unificado de Modelado (UML). .*

12. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico. Quinta edición .* s.l. : McGraw-Hill Companies, 2002. ISBN: 8448132149.

13. **Delgado Dapena, Martha D.** *Definición del modelo del negocio y del dominio utilizando UML.*

14. **RUMBAUGH, I.J.G.B.J ,Addison Wesley.** *El Proceso Unificado de Desarrollo de Software.* 2000.

Referencias

15. **Sommerville, Ian.** Sommerville. Parte_II_Requerimientos. [book auth.] Sommerville. *SOFTWARE ENGINEERING .* s.l. : Pearson Educación SBN-13: 978-0-13-703515-1., 2011.

16. **Lidia Fuentes, José M. Troya y Antonio Vallecillo.** *Desarrollo de Software Basado en Componentes. Departamento Lenguajes y Ciencias de la Computación. .* Universidad de Málaga : s.n.

17. **LARMAN, CRAIG.** *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos. .* s.l. : PRENTICE HAL, 1999. 970-17-0261-1.

18. **GAMMA, ERICH, y otros.** *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995.

19. **Serrano Cuayahuitl, Victor Hugo.** *Pruebas de unidad, Estándares de codificación y generación automática de código. .* s.l. : Facultad de Ciencias básicas, ingeniería y tecnología. Universidad Autónoma de Tlaxcala.

20. **Gutiérrez, J.J, Escalona, M. Mejías M.J, Torres J.** *Pruebas del sistema en Programación Extrema.* Sevilla : s.n.

21. **Roque, Linet Lores Sanchez y Diana Monne.** *Aplicacion de las pruebas de liberacion al Sistema Informatico de Genetica Medica.* La Habana : s.n., 2009.

22. Metodo de Caja Negra . [Online] 2010. http://www.innosupport.net/index.php?id=2080&L=6.

23. Articulos Educativos Tecnicas de Prueba . [Online] 2010. http://centrodeartigos.com/articulos-educativos/article_11744.html.

Bibliografía

1. **Pentaho Corporation.** Pentaho. [Online] 2005-2014. http://www.pentaho.com/product/data-integration.

2. **Talend.** Talend. [Online] 2006-2014. http://www.talend.com/resources/documentation.php.

3. —. Talend . [Online] 2006-2014. http://www.talendforge.org/components/.

4. **Octopus.** Octopus . [Online] http://www.octopus.com.co/.

5. **Clover.** Clover. [Online] https://www.clover.com/.

6. **Grupo ETL.** *Guía de componentes de Pentaho Data Integration.* La Habana : s.n.

7. **Beck, Kent.** *Metodologías XP y RUP. .*

8. **CrystalClear.** CrystalClear . [Online] http://www.crystalclear.com.ph/.

9. **Palacio, Juan.** *El modelo Scrum.* 2006.

10. **Letelier, Patricio y Penadés, María del Carmen.** *Metodologías ágiles para el desarrollo de software, eXtreme Programing (XP).* Valencia : s.n., 2006. Vol. 05. 1666-1680.

11. **Pressman, Roger S.** Cap_04_Desarrollo_Agil_Parte_1. *Ingeniería de Software, un enfoque práctico. Quinta edición .* s.l. : McGraw-Hill Companies, 2002.

12. **Hernández Orallo, Enrique.** *El Lenguaje Unificado de Modelado (UML). .*

13. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico. Quinta edición .* s.l. : McGraw-Hill Companies, 2002. ISBN: 8448132149.

14. **Delgado Dapena, Martha D.** *Definición del modelo del negocio y del dominio utilizando UML.*

15. **RUMBAUGH, I.J.G.B.J ,Addison Wesley.** *El Proceso Unificado de Desarrollo de Software.* 2000.

16. **Lidia Fuentes, José M. Troya y Antonio Vallecillo.** *Desarrollo de Software Basado en Componentes. Departamento Lenguajes y Ciencias de la Computación. .* Universidad de Málaga : s.n.

17. **LARMAN, CRAIG.** *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos. .* s.l. : PRENTICE HAL, 1999. 970-17-0261-1.

18. **GAMMA, ERICH, y otros.** *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995.

19. **Serrano Cuayahuitl, Victor Hugo.** *Pruebas de unidad, Estándares de codificación y generación automática de código. .* s.l. : Facultad de Ciencias básicas, ingeniería y tecnología. Universidad Autónoma de Tlaxcala.

20. **Gutiérrez, J.J, Escalona, M. Mejías M.J, Torres J.** *Pruebas del sistema en Programación Extrema.* Sevilla : s.n.

21. **Roque, Linet Lores Sanchez y Diana Monne.** *Aplicacion de las pruebas de liberacion al Sistema Informatico de Genetica Medica.* La Habana : s.n., 2009.

22. Metodo de Caja Negra . [Online] 2010. http://www.innosupport.net/index.php?id=2080&L=6.

23. Articulos Educativos Tecnicas de Prueba . [Online] 2010. http://centrodeartigos.com/articulos-educativos/article_11744.html.

24. **Sommerville, Ian.** Sommerville. Parte_II_Requerimientos. [book auth.] Sommerville. *SOFTWARE ENGINEERING .* s.l. : Pearson Educación SBN-13: 978-0-13-703515-1., 2011.

25. **Rodríguez, Dr. Justo Luís Pereda, Hernánde, MSc. Rolando Díaz y Domínguez, Ing. Idalmys Cruz.** *Las Tecnologías de la Información y las Comunicaciones en función del Desarrollo Social. Proyecciones de Cuba .* La Habana : Revista Digital Sociedad de la Información.

26. Ingeniería del Software . *Ingeniería del Software.* [Online] 2011. www.is.ls.fi.upm.es/docencia/is2/documentacion/ModeloDominio. .

27. **Schmuller, Joseph.** *Aprendiendo UML en 24 horas.* México : Prentice Hall, 2000. ISBN: 968-444-463-X.

28. **Letelier, Patricio y Penadés , Mª Carmen.** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).*

29. **Fernández Escribano, Gerardo.** *Ingeniería del Software II. Introducción a Extreme Programming.* 2002.

30. **Hernández León, Rolando Alfredo y Coello Gon, Sayda.** *El proceso de investigación cientifica.*

31. **Echeverry Tobón, Luis Miguel y Delgado Carmona, Luz Elena.** *Caso práctico de la metodologia ágil XP al desarrollo de software.*

32. **Hommel, Scott.** *Convenciones de Código para el lenguaje de programación.* s.l. : Sun Microsystems Inc, 1999.

33. **Cabrera González, Lianet y Pompa Torre, Enrique Roberto.** *Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información.*

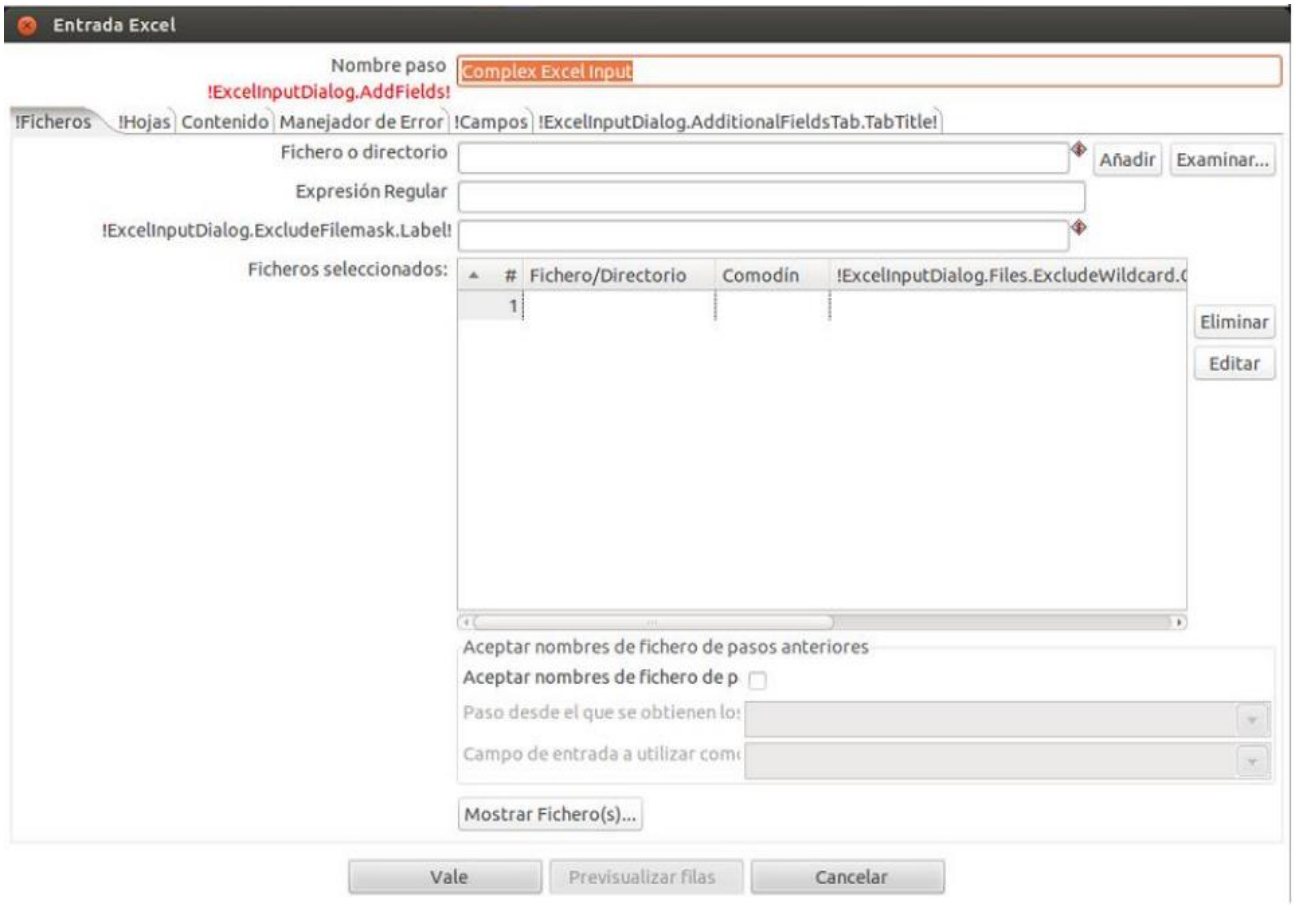
34. Octopus. [Online] http://octopus.objectweb.org ..

35. **Kniberg, Henrik.** *SCRUM Y XP DESDE LAS TRINCHERAS. Cómo hacemos Scrum.*

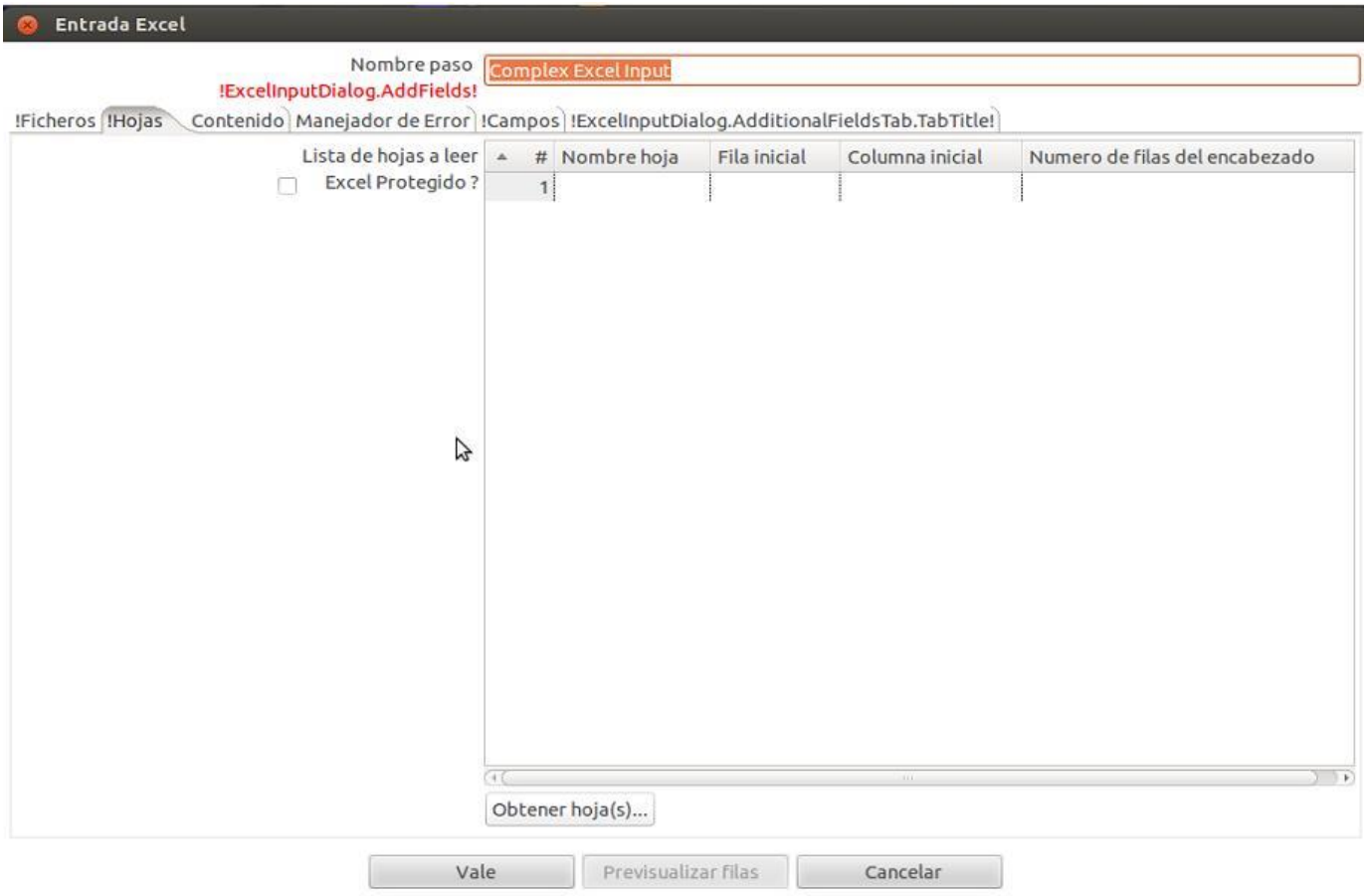
36. **Peláez, Juan.** *Arquitectura basada en Componentes.* 2009.

Anexos

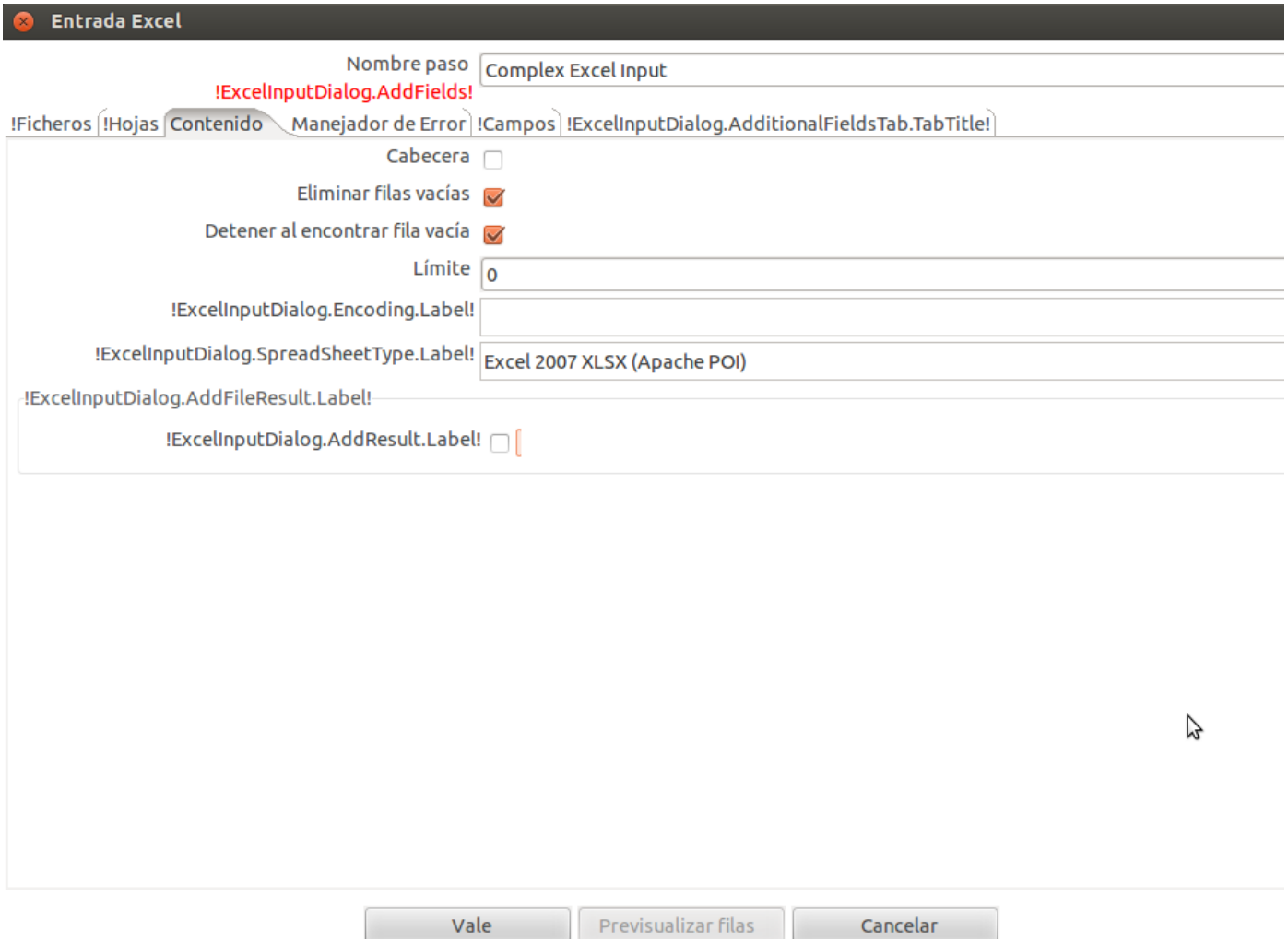
Anexo 1 Logo diseñado para el componente Complex Excel Input (Entrada Excel Complejo)



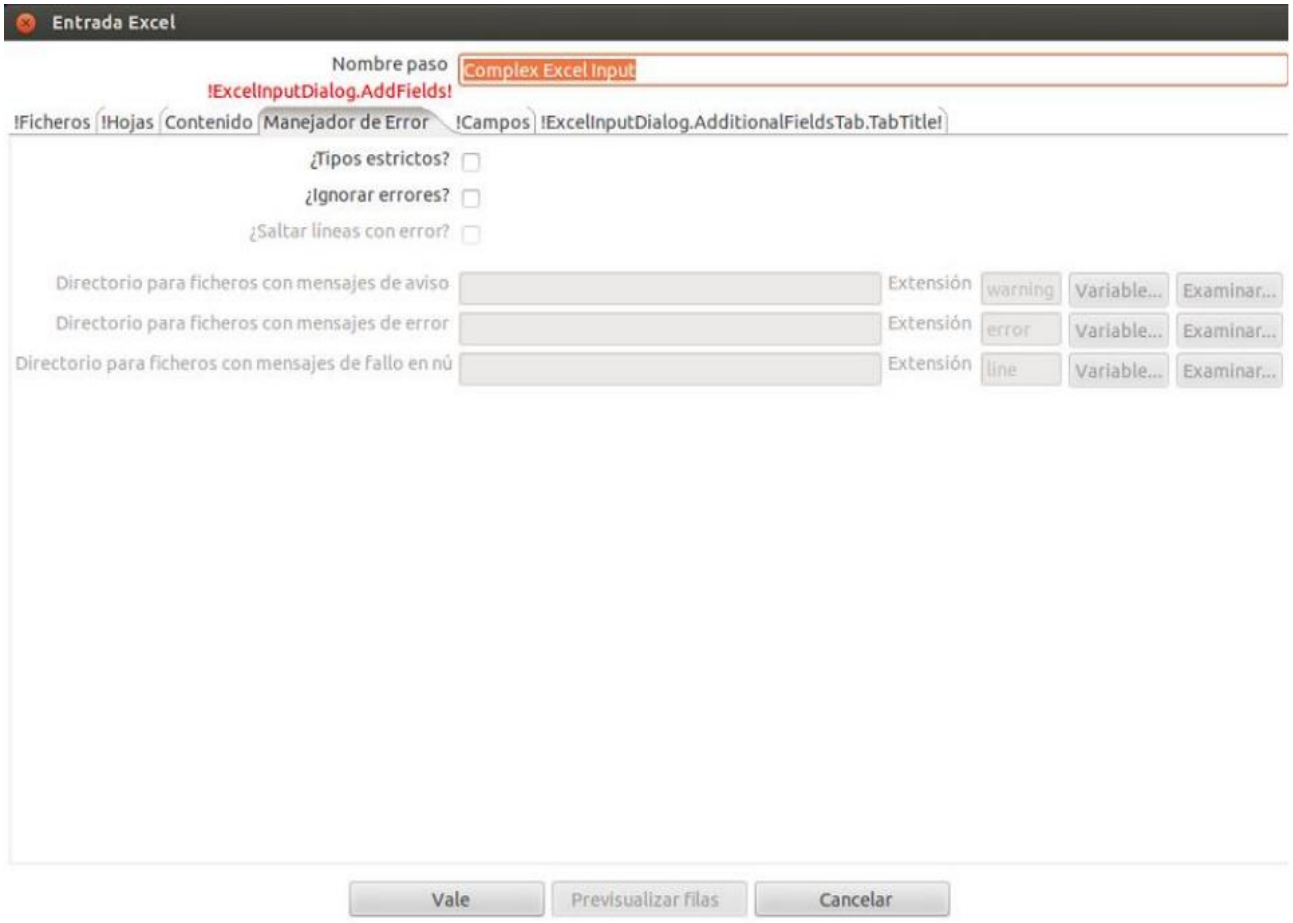
Anexo 2 Interfaz de la pestaña ficheros



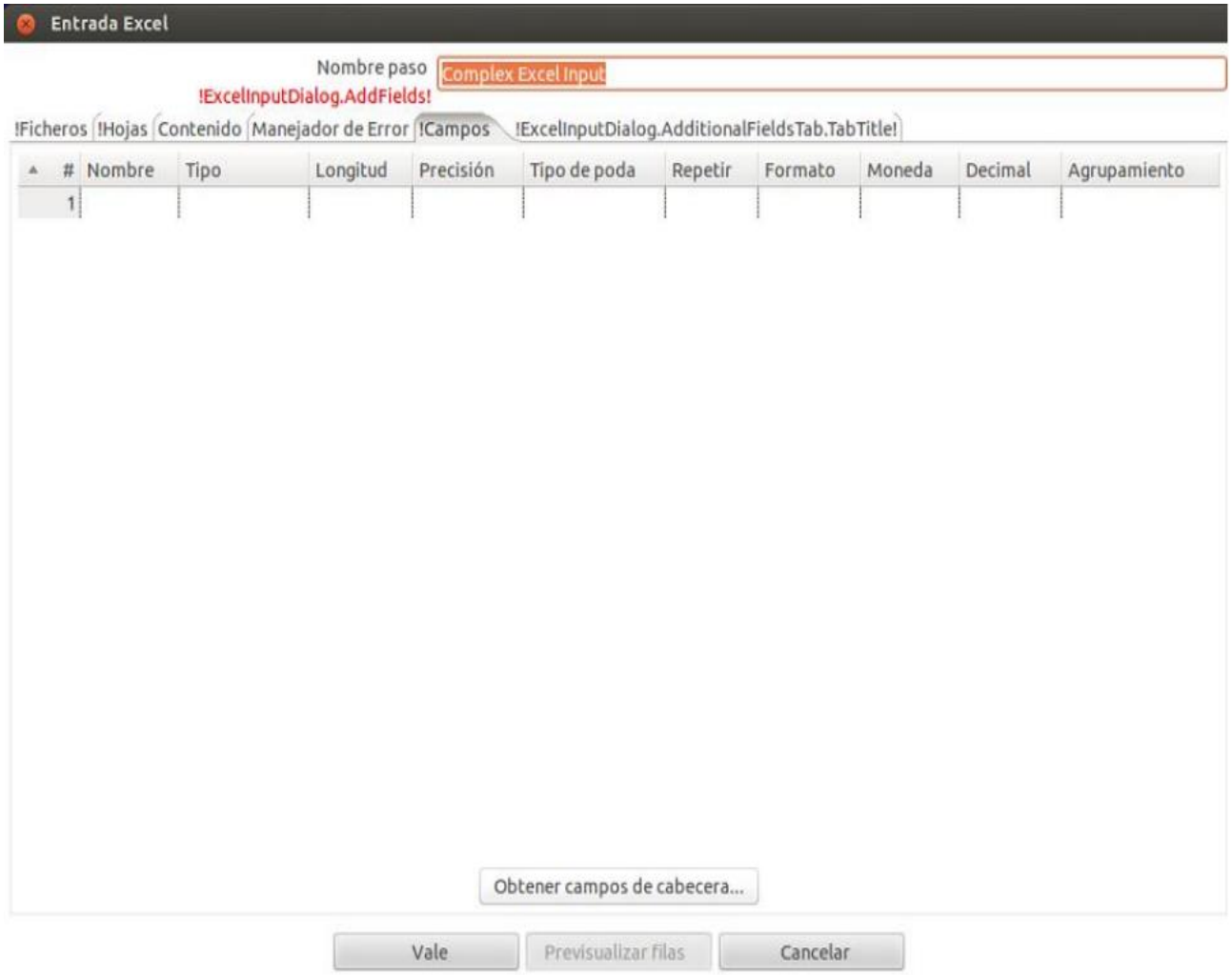
Anexo 3 Interfaz de la pestaña hojas



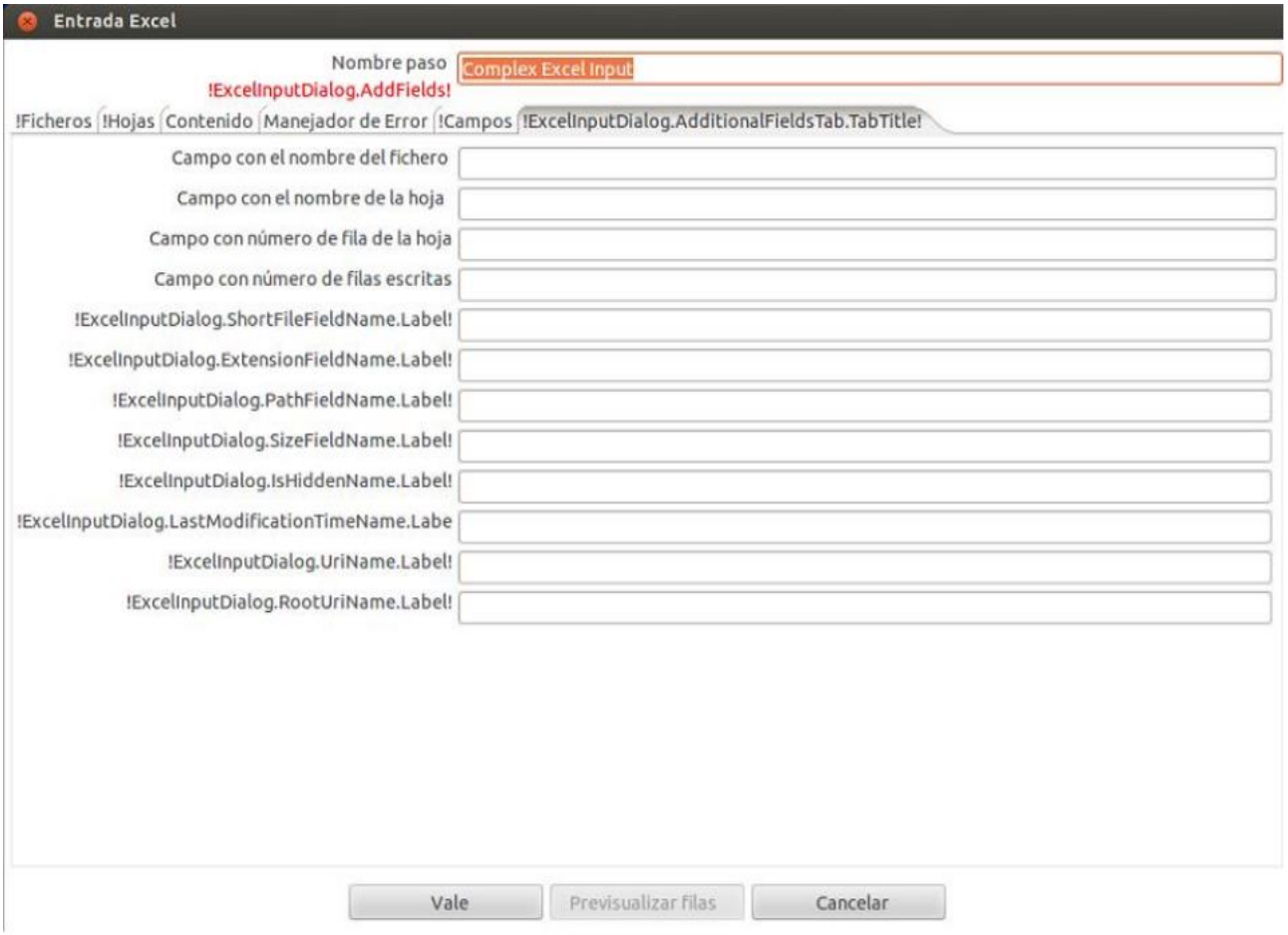
Anexo 4 Interfaz de la pestaña Contenido



Anexo 5 Interfaz de la pestaña manejador de error



Anexo 6 Interfaz de la pestaña Campos



Anexo 7 Interfaz de la pestaña ExcelInputDialog.AdditionalFieldsTab.TabTitle

Glosario de Términos

Archivo CSV: Los ficheros **CSV** (del inglés *comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal: España, Francia, Italia...) y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles. El formato CSV es muy sencillo y no indica un juego de caracteres concreto, ni cómo van situados los bytes, ni el formato para el salto de línea. Estos puntos deben indicarse muchas veces al abrir el fichero, por ejemplo, con una hoja de cálculo.

XBase: es el término genérico para todos los lenguajes de programación que derivan del lenguaje de programación dBase, originalmente publicado por *Ashton-Tate*. Existen indicativos de que existió un predecesor no comercial. La 'x' significa que existen diversos intérpretes y compiladores para este lenguaje.

Archivos Count: Los documentos COUNT son Archivos varios asociados con *Gen Counters File*.

LDAP: son las siglas *de Lightweight Directory Access Protocol* (en español Protocolo Ligero de Acceso a Directorios) que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. LDAP también se considera una base de datos (aunque su sistema de almacenamiento puede ser diferente) a la que pueden realizarse consultas.

LDIF: representa LDAP Data Interchange Format. Un archivo LDIF es un servicio de directorio de archivo que se puede transferir entre servidores. Cada vez que la información se transfiere automáticamente entre computadoras, el emisor y el receptor tienen que tener un formato común definido por la estructura de registro. No tiene sentido enviar un mensaje o un archivo por escrito a un nivel secreto, ya que el receptor de dicho archivo no sería capaz de romper el contenido en datos significativos. Esta es la razón por las normas internacionales que definen los formatos neutros se publican libremente. El formato LDIF está definido por un protocolo abiertamente disponible.

Lector ESRI Shapefile: El formato *ESRI Shapefile* (SHP) es un formato de archivo informático propietario de datos espaciales desarrollado por la compañía ESRI, quien crea y comercializa

software para Sistemas de Información Geográfica. Originalmente se creó para la utilización con su producto *ArcView GIS*, pero actualmente se ha convertido en formato estándar para el intercambio de información geográfica entre Sistemas de Información Geográfica.