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

Componente de software para la generación automática del cuadro de clasificación archivística

Universidad de las Ciencias Informáticas

Facultad 2

Autor: Luis Miguel Molina Betancourt

Tutor: Ing. Dasiel Cordero Morales

La Habana, junio de 2016

"Año 58 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor del trabajo de diploma “Solución Informática para organización de forma automática de los archivos digitales textuales” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ___ días del mes de junio del año 2016.

_____	_____
Luis Miguel Molina Betancourt	Ing. Dasiel Cordero Morales
Firma Autor	Firma Tutor

AGRADECIMIENTOS

Quiero agradecer a mis padres por toda la confianza depositada en mí y por todo el apoyo que me han dado siempre en los momentos más difíciles de mi carrera. Y a mi familia, por su apoyo incondicional durante todos estos años.

Quiero agradecer a mi familia por apoyarme siempre y servirme de guía procurando en todo momento mi superación educacional para ser en un futuro una persona independiente.

Agradezco especialmente a los que cursaron algún semestre conmigo, compañeros de desvelos, presiones y todo aquello que implica la vida de estudiante por los momentos tan lindos que hemos compartido, mis amigos todos, por confiar en mi amistad y brindarme su apoyo.

Agradecer al profesor Dasiel su incondicional apoyo cuando necesité su ayuda, por sus consejos y ser siempre una persona servicial y atenta conmigo.

Quiero agradecer a nuestra Revolución por darnos la oportunidad de estudiar en una universidad de excelencia y en especial a nuestro Comandante Fidel Castro.

Finalmente, un recuerdo a todas las personas que me quieren, a quienes seguro he abandonado demasiado tiempo y en algún caso decepcionado. Aunque a veces puedan dudarlo, siempre han estado, están y estarán conmigo.

DEDICATORIA

Dedico especialmente este trabajo de diploma a mis padres porque más que mío este es su sueño y por ser mis grandes amigos en todos los momentos.

RESUMEN

La clasificación y gestión de los archivos es tan antigua como la organización social de la humanidad. Con el auge de las Tecnologías de la Información y las Comunicaciones, estas actividades han cambiado en gran parte a archivos digitales. Los Sistemas de Gestión de Archivos permiten gestionar el contenido de los archivos estableciendo su estructura y funciones. Tales actividades corresponden a la clasificación archivística, constituyendo el cuadro de clasificación artefacto clave en estas acciones.

Para la creación del cuadro de clasificación es necesario procesar las fuentes archivísticas de la organización. Para ello deben digitalizarse los archivos mediante técnicas de reconocimiento óptico de caracteres. Luego debe realizarse una extracción y representación del contenido de los archivos, para agrupar los archivos según características de su contenido. Uno de los procesos para el procesamiento de la información es el agrupamiento automático, de la Inteligencia Artificial, encargada de agrupar un conjunto de objetos en subconjuntos para una mejor organización.

El objetivo de este trabajo es desarrollar un componente de software para generar automáticamente una primera aproximación del cuadro de clasificación archivística, para sistemas de gestión de archivos desarrollados en la plataforma Java, utilizando técnicas de agrupamiento jerárquico. Para ello fueron implementados los algoritmos los Single-Link, Complete-Link, UPGMA, WPGMA, UPGMC, WPGMC y Ward, utilizando las funciones de semejanza Dice, Jaccard y Coseno. Los resultados obtenidos fueron evaluados empleando las métricas Overall Similarity, Entropía y F-Measure; las cuales miden la calidad de agrupamiento tanto interna como externa. Esto fue posible mediante un caso de estudio, utilizando un conjunto de datos de prueba obtenidos de Reuters-21578; que permitió comparar los algoritmos utilizando la precisión como medida principal de la calidad de los agrupamientos. También fueron realizadas pruebas de software dividida en los niveles de unidad y sistema utilizándose pruebas unitarias y de aceptación como define la metodología XP, lográndose corregir todas las deficiencias detectadas en las iteraciones realizadas.

Palabras clave: algoritmo de clústering jerárquico, archivo digital textual, clasificación automática, cuadro clasificación archivística.

Índice

Introducción

El estudio de los archivos y su documentación es tan antiguo como la organización social de la humanidad (1). Los primeros archivos aparecen con los primeros imperios como herramienta de control de la población y la riqueza. Al finalizar el siglo XX con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), el cúmulo de archivos se ha trasladado en gran parte a archivos digitales. Casi toda la documentación que se genera actualmente es guardada digitalmente en centros de datos distribuidos por el mundo (2). Por tal motivo los archivos son necesarios para promover el conocimiento, impulsar la investigación histórica y científica, además de apoyar la gestión documental y archivística de cualquier organización. Lo anterior implica que los archivos son indispensables en el desarrollo de las organizaciones y de la sociedad, por lo que es estudio e informatización de las actividades de las Ciencias de la Información y la Archivística juegan un papel importante para la sociedad.

De acuerdo con el Diccionario de la Real Academia de la Lengua Española, un archivo se define como el local en el que se custodian documentos públicos o particulares. También puede definirse como el lugar donde se custodia el acervo documental, pero el más útil para la investigación lo constituye: fondo documental, como conjunto de documentos producidos o recibidos por una persona física o jurídica en el ejercicio de sus actividades (3). Es necesario destacar que la investigación estará orientada a los archivos de texto.

Asociado a este concepto se encuentra la acción de archivar que significa guardar de forma ordenada documentos útiles, haciéndolo de un modo lógico y eficaz que permita su posterior localización de la forma más rápida posible cuando sea necesario (3). Pero esta acción de archivar requiere de un proceso para su correcto funcionamiento de ahí que se necesite utilizar Sistemas de Gestión de Archivos o Gestión Documental.

Estos Sistemas de Gestión de Archivos son los encargados de gestionar y tratar en todos sus aspectos la información fijada en los documentos de archivo. En este sistema se encuentran un conjunto de normas técnicas y prácticas usadas para administrar el flujo de archivos de todo tipo en una organización (4). El establecimiento de un Sistema de Gestión de Archivos en una institución, asume como premisa que todas las instancias de archivo existentes en la organización operen en forma coordinada, siendo necesario establecer su estructura y funciones, cómo se van a relacionar sus diferentes componentes y cómo se gestionará el flujo de archivos entre ellos en consideración a su ciclo de vida (4). Tales sistemas son necesarios para cualquier organización puesto que posibilita una gestión y control efectiva con una mayor sencillez, logrando que la organización tenga acceso instantáneo a toda la documentación necesaria para su actividad de negocio, reducción de tiempo de consultas y tareas de archivo, ahorro de espacio físico y resolución del problema de localización de archivos. Un Sistema de Gestión de Archivo debe contar con un determinado componente que permite organizar los archivos a partir de determinadas características (4). De ahí se derivará su posterior consulta, almacenamiento, modificación, así como los permisos para su acceso. Esto permite que los documentos de archivo se encuentren en una determinada estructura que

se deriva de los intereses de la organización y responde a las necesidades de aquellos que lo consumen; constituyendo el cuadro de clasificación archivística como artefacto clave en la gestión de archivos.

Un cuadro de clasificación archivística se define como una estructura jerárquica y lógica que refleja las características similares de la clasificación de una colección de archivos. Tal colección refleja generalmente las funciones y las actividades de una organización (5). Esto permite que se organice intelectualmente la información y situar los archivos en sus relaciones. La organización de los documentos de archivo se realiza de acuerdo al mismo orden en que se dividen llegando hasta su grado inferior, respetando el principio de procedencia: cada documento debe estar colocado en el fondo[1] de que proviene y en ese fondo en su lugar de origen.

Contando con un cuadro de clasificación bien estructurado la institución tiene a su merced una poderosa herramienta que le posibilita organizar los documentos de archivo que gestiona dentro del espacio físico de conservación, temporal o definitiva. Esto le permitirá agrupar los elementos que representan su contenido para relacionarlo de manera lógica con otros documentos de archivo mediante criterios de clasificación (5). Lo anterior facilita la localización conceptual de la documentación o bien el acceso a la información contenida en el acervo documental, a la vez que facilita la localización física de cada documento o expediente para su eficaz consulta. El correcto funcionamiento de una entidad u organización determinada en gran medida depende de una adecuada gestión de documentos de archivo, mientras que esta última debe seguir ciertos requisitos dentro de los cuales se destaca la creación de cuadro de clasificación archivística.

Para la correcta formulación de un cuadro de clasificación, como actividad clave dentro la gestión de documentos de archivo, es necesario realizar un procesado de las fuentes archivísticas existentes en la organización. En una primera etapa se deben digitalizar todos los archivos mediante un proceso tecnológico que permite, mediante la aplicación de técnicas fotoeléctricas o de escáner, convertir la imagen contenida en un documento en papel en una imagen digital (6). En el proceso de digitalización se emplean diferentes tecnologías para convertir imágenes de caracteres en letra manuscrita o impresos, en caracteres capaces de ser interpretados o reconocidos por un ordenador. Tal actividad se denomina reconocimiento óptico de caracteres (OCR por sus siglas en inglés). Luego es necesario realizar una clasificación de la documentación en diferentes categorías estableciendo niveles de relación o coordinación, así como de semejanza y diferencia entre las categorías de agrupamiento identificadas.

Existen varias vías de clasificar documentos: por autor, año de edición, u organización que generó el documento. También pueden ser clasificados a través del análisis del contenido de los mismos, donde las técnicas más utilizadas son la minería de texto. Esta última se define como el proceso de extraer patrones

[1] Fondo: es el conjunto documental procedente de una institución o persona, conservado en el archivo de dicha institución o de una institución de archivo, es el resultado natural de la actividad que la institución realiza.

interesantes y nuevos conocimientos desde archivos de texto no estructurados, el cual tiene involucrado otras disciplinas como el clústering, la categorización, y la recuperación de información (7).

La minería de texto consta de dos etapas principales: la primera es llamada etapa de pre procesamiento, donde los textos se transforman en algún tipo de representación estructurada o semiestructurada que facilite su posterior análisis (8). El trabajo de diploma (9) desarrollado en la Universidad de las Ciencias Informáticas en el año 2015, permite la representación de archivos textuales mediante un vector de términos, donde cada término lleva asociado un coeficiente o peso que trata de expresar la importancia o grado de representatividad de ese término en ese vector a partir de la evaluación de la frecuencia de aparición de los términos[2]. La segunda es llamada etapa de descubrimiento enfocada en el aprendizaje automático, rama de la inteligencia artificial cuyo objetivo es desarrollar o emplear técnicas que permiten realizar mediante las representaciones intermedias de determinados objetos, descubrir patrones interesantes, nuevo conocimiento, o realizar la categorización de objetos.

En el campo de la Inteligencia Artificial (IA), el análisis de grupos (agrupamiento o clasificación), es el proceso de dividir un conjunto de objetos de datos en subconjuntos, dando en cierta medida una clasificación de los grupos de objetos analizados. Cada subconjunto es un grupo, de tal manera que los objetos en un grupo son similares entre ellos, sin embargo, muy diferente a los objetos de otros grupos (10).

Por lo general, cuando se habla de clasificación se distinguen dos escenarios diferentes que requieren soluciones distintas. Estos escenarios reciben diversos nombres, pero básicamente consisten en lo siguiente: de un lado, una situación en la que se parte de una serie de clases o categorías conceptuales prediseñadas a priori y en la que la labor del clasificador es asignar cada elemento a la clase o categoría que le corresponda. Esto es comúnmente nombrado como clasificación supervisada o semisupervisada. En el segundo escenario posible, no hay categorías previas ni esquemas o grupos de clasificación establecidos a priori. Los elementos se agrupan en función de ellos mismos, de su contenido; de alguna manera, se puede afirmar que se auto organizan. Lo anterior se conoce como clasificación no supervisada y se efectúa de forma automática, sin supervisión o asistencia manual (11).

El primer escenario se conoce como clasificación supervisada, no sólo porque requiere la elaboración manual o intelectual del cuadro o esquema de categorías, sino también porque requiere un proceso de aprendizaje o entrenamiento por parte del clasificador, que debe ser supervisado manualmente (11). La elaboración del cuadro o esquema de categorías tiene un alto costo en recursos humanos, que pueden concurrir en errores en la elaboración del mismo, además, existe la necesidad de un experto que supervise el proceso de aprendizaje o entrenamiento. En cambio, se puede tener la posibilidad de ante la necesidad de organizar los archivos, este proceso sea de forma automática sin contar con la participación de un

[2] Término: se considera como la raíz semántica de cada palabra

experto, ahorrando recursos humanos y brindando un valor agregado a aquellos productos comercializables sobre la línea de gestión de archivos.

Las etapas descritas anteriormente agilizan tareas en entornos no informatizados de gestión de archivos que por separado requieren la participación de personal capacitado. El desarrollo tecnológico guía los procesos de generación automática del cuadro de clasificación archivística a un continuo avance y con ello las organizaciones a nivel mundial se ajustan a tales condiciones. Actualmente existe una amplia gama de soluciones informáticas que gestionan la clasificación archivística automática empleando una gran cantidad de funcionalidades.

Por tal motivo fueron estudiadas soluciones actuales que son utilizadas en una gran cantidad de organizaciones que generan grandes cantidades de documentación y que necesitan sistemas de gestión como medio para el perfeccionamiento de sus procesos. Entre los principales softwares de gestión de archivos se encuentran soluciones de gran envergadura que permiten la administración de archivos mediante el uso del cuadro de clasificación archivístico como una de sus herramientas; permitiendo en su mayoría una eficiente gestión de archivos (12) (13).

Estos programas informáticos, aunque posibilitan la autogeneración del cuadro archivístico no emplean técnicas de inteligencia artificial que permitan completar la estructura que define el cuadro de clasificación archivística. Pocas añaden funciones de última generación a sistemas de gestión de contenidos empresariales que permitan la generación del cuadro de clasificación archivística incluso usando métodos automáticos basados en minería de texto. Algunas de estas soluciones informáticas están desarrolladas para la plataforma Windows, por tanto, imposibilitan su integración con otros sistemas. Otras no utilizan clasificación mediante el análisis del contenido de los textos y solo analizan frases, palabras principales y características generales de los archivos (12) (13). Las más afines a la investigación tratan el problema de la generación automática del cuadro de clasificación archivístico de forma bastante acertada, pero se encuentran bajo licencia que imposibilitan la modificación y redistribución del mismo para su comercialización.

Actualmente, Cuba desde hace varios años ha comenzado a dar pasos significativos preservación de los archivos históricos de las instituciones competentes. La Universidad de las Ciencias Informáticas (UCI), centro educacional vinculado a la producción de software, está inmerso en esta tarea. Uno de sus centros productivos: Centro de Informatización de la Gestión Documental (CIGED) se dedica al desarrollo de sistemas y servicios informáticos para la Gestión de Documentos Administrativos, Gestión de Archivos Históricos y Gestión Bibliotecaria y Centros de Documentación (14).

Sobre la línea de Gestión de Archivos Históricos se desarrolla, bajo la marca Xabal el sistema informático Xabal-Arkheia. Este se utiliza para la gestión de archivos históricos y está desarrollado bajo la plataforma java (14). Actualmente el sistema realiza la digitalización de los documentos y mediante el trabajo de diplomado (9) antes mencionado, permite la representación mediante un vector de términos los documentos digitalizados, pero no permite la clasificación automática así como la generación automática

del cuadro de clasificación archivística. Por tal motivo tareas como: indexación, extracción de metadatos, construcción del cuadro de clasificación archivística y organización de acuerdo al contenido se realizan de forma manual. Esto implica que se necesite de la intervención de un experto para la ejecución de estas tareas. Para las organizaciones esto trae consigo gastos en formación de personal, establecimiento de procedimientos que aseguren los criterios de categorización y auditorías de calidad lo que aumenta los costos y el tiempo de tratamiento de los documentos, sin que con ello se asegure la plena calidad de los procesos de obtención, recuperación, representación y utilización de la información.

Contar con un componente informático capaz de clasificar automáticamente los archivos por su contenido para generar una primera aproximación del cuadro de clasificación archivística, de manera que su representación se genere de forma automática es necesario para la etapa de organización y clasificación automática de los archivos digitales textuales.

De acuerdo a lo anteriormente expuesto se plantea el siguiente **problema a resolver**: ¿Cómo generar automáticamente una primera aproximación del cuadro de clasificación archivística para sistemas de gestión de archivos desarrollados en la plataforma Java?

El **objeto de estudio** del presente trabajo es la clasificación archivística en sistemas de gestión de archivos.

De acuerdo al problema a resolver planteado se define como **objetivo general**: Desarrollar un componente de software para generar automáticamente una primera aproximación del cuadro de clasificación archivística para sistemas de gestión de archivos desarrollados en la plataforma Java, utilizando técnicas de agrupamiento jerárquico.

El **campo de acción** de este trabajo está enmarcado en las técnicas de agrupamiento jerárquico para la generación del cuadro de clasificación archivística.

En aras de guiar la realización de la investigación, se definen las siguientes **preguntas de investigación**:

- ¿Cuáles son los principales conceptos asociados a la clasificación archivística asociados a la investigación?
- ¿Cuáles son los principales conceptos asociados a la clasificación archivística sobre los Sistemas de Gestión de Archivos?
- ¿Cuál es la situación actual sobre la utilización de la clasificación y generación automática del cuadro de clasificación archivística sobre sistemas de gestión de archivo?
- ¿Cuáles son las principales actividades que intervienen en el proceso de generación automática del cuadro de clasificación archivística?
- ¿Cuáles son los principales algoritmos que definen el agrupamiento a través de estructuras jerárquicas?
- ¿Cuáles son las principales funciones de similitud que pueden ser empleadas en el agrupamiento jerárquico?

- ¿Cuáles son las métricas que permiten estudiar el grado de acierto algoritmos de agrupamiento jerárquico?
- ¿Cuáles son las principales herramientas a utilizar en el desarrollar una solución informática para generar una primera aproximación del cuadro de clasificación archivística utilizando técnicas de agrupamiento jerárquico?
- ¿Qué procedimientos se deberían utilizar para comprobar el funcionamiento de los algoritmos de agrupamiento jerárquico?
- ¿Qué estrategia de prueba utilizar para comprobar el funcionamiento de la solución informática para generar una primera aproximación del cuadro de clasificación archivística utilizando técnicas de agrupamiento jerárquico?

Para dar solución al objetivo general se definen los siguientes **objetivos específicos**:

- Caracterizar la situación existente en el mundo, la región y el país en particular sobre la generación automática del cuadro de clasificación archivística en sistemas de gestión de archivos, con el empleo de técnicas de agrupamiento jerárquico.
- Desarrollar un componente de software, sobre la plataforma Java que permita la generación automática del cuadro de clasificación archivística mediante el uso de técnicas de agrupamiento jerárquico.
- Comprobar con la utilización de las métricas para evaluar la calidad de los agrupamientos junto a las pruebas de software, la validez y funcionamiento del componente de software desarrollado.

Para dar cumplimiento al objetivo antes descrito se definen las siguientes **tareas de investigación**:

- Descripción de los principales conceptos del cuadro de clasificación archivística y los sistemas de gestión de archivos.
- Descripción de los elementos asociados a la clasificación archivística sobre sistemas de gestión de archivos.
- Caracterización de la situación actual de la generación automática del cuadro de clasificación archivística sobre sistemas de gestión de archivos.
- Selección de las métricas usadas para la comparación de semejanza en los diferentes modelos que conforman los antecedentes de la investigación.
- Descripción de los diferentes algoritmos que permiten que definen el agrupamiento mediante técnicas de agrupamiento jerárquico.
- Selección de la metodología de desarrollo de software, herramientas, lenguajes y tecnologías a utilizar para el desarrollo de la solución informática que permita una primera aproximación del cuadro de clasificación archivística utilizando técnicas de agrupamiento jerárquico.
- Validación del funcionamiento de los diferentes algoritmos usados para demostrar el correcto agrupamiento de archivos digitales textuales.

- ❖ Definición de una estrategia de prueba para comprobar el funcionamiento del componente desarrollado.

En el desarrollo de la investigación se utilizan los siguientes **métodos de investigación científica**.

Métodos teóricos

Analítico-Sintético: este método fue utilizado para estudiar toda la teoría y documentación referentes a el cuadro de clasificación archivística, los sistemas de gestión de archivos, y la clasificación archivística que es realizada sobre estos sistemas. Además, permitió la caracterización de la situación actual de la generación automática del cuadro de clasificación archivística. Con el uso de este método, se seleccionó las funciones de similitud y algoritmos que pueden ser empleadas en el agrupamiento jerárquico. También, fue empleado en la selección de la metodología de desarrollo de software, herramientas, lenguajes y tecnologías a emplear para el desarrollo de la solución informática que permita el agrupamiento de archivos digitales textuales.

Histórico-Lógico: este método fue utilizado para estudiar los antecedentes y como se relacionan con la investigación actual mediante el estudio de los procesos relacionados con la creación del cuadro de clasificación archivística, las métricas y los algoritmos que permiten el agrupamiento jerárquico. Su utilización permitió adquirir conocimientos sobre la forma en que se realizan estas tareas para la definición de los antecedentes de la investigación.

Modelación: este método fue utilizado para la modelación de la solución desarrollada para la generación automática del cuadro de clasificación archivística. Además, su utilización permitió la modelación de la propuesta de solución, la creación de los diagramas de clases para representar los componentes y relaciones del sistema a desarrollar.

Métodos empíricos

Simulación: este método fue utilizado para la ejecución del algoritmo y componente implementados mediante el uso de datos artificiales de prueba, lo que permitió comprobar su funcionamiento para garantizar la calidad de ambos.

Análisis estático: este método fue utilizado para realizar un examen de la estructura del componente implementado. Su utilización permitió además la aplicación de pruebas unitarias y de aceptación al componente para detectar y corregir errores.

Caso de estudio: este método fue utilizado para realizar pruebas al algoritmo y componentes implementados mediante ejemplos reales donde se conoce el efecto final, a los cuales se le hacen comparaciones que determinan la calidad de la solución implementada.

El presente documento está compuesto por introducción, desarrollo, conclusiones y bibliografía. El contenido del desarrollo está estructurado en tres capítulos que desarrollan su contenido de la siguiente manera:

Capítulo 1. Caracterización dela generación automática del cuadro de clasificación archivística con el empleo de técnicas de agrupamiento jerárquico: Expone los elementos teóricos de la investigación, realizando un análisis del estado del arte del uso de técnicas de agrupamiento jerárquico. Se realiza el estudio e investigación de principales procesos relacionados con la creación del cuadro de clasificación archivística existentes en la actualidad. Se determinan la metodología de desarrollo, herramientas y tecnologías que se utilizan para desarrollar la solución propuesta.

Capítulo 2. Desarrollo de un componente de software para la generación automática del cuadro de clasificación archivística: Presenta la selección del proceso de clasificación archivística a implementar y se detalla el diseño y la concepción de la solución informática. Expone el modelo de análisis, el de diseño y el de implementación que responden directamente a la solución del problema.

Capítulo 3. Comprobación del funcionamiento de la propuesta de solución: Incluye las definiciones y los resultados de las pruebas realizadas al sistema, para comprobar la efectividad de la aplicación.

Capítulo 1. Caracterización de la generación automática del cuadro de clasificación archivística con el empleo de técnicas de agrupamiento jerárquico.

1.1 Clasificación archivística sobre sistemas de gestión de archivos.

Un sistema de gestión archivos es todo programa de ordenador creado para la gestión de grandes cantidades de documentos de archivo (15). La gestión de archivos es compleja y exige la correcta aplicación de una gran variedad de parámetros específicos como el cuadro de clasificación.

El cuadro de clasificación constituye el elemento clave de cualquier sistema de gestión de archivos y por tanto el parámetro más importante sobre este tipo de sistemas. Este define el modo en que los documentos electrónicos de archivo se organizarán en expedientes, así como las relaciones entre dichos expedientes. Lo anterior define un sistema de clases según el cual se distribuyen los objetos de cualquier género, necesitando para la creación de este sistema de clases un proceso de clasificación archivística.

Tal proceso define tres actividades fundamentales en el ámbito actual de desarrollo tecnológico e informatización de procesos y actividades. Estas actividades son: digitalización, extracción de contenido y clasificación. Sobre esta última se centra la investigación, pero parte de una investigación precedente (poner la referencia) que asume que se cuenta con una colección de documentos digitalizados y procesa estos mediante técnicas de minería de texto y recuperación de información. Esto arroja como resultado una colección de vectores asociados a los archivos de texto que cuentan como relevancia o peso la relación de frecuencias de aparición de los términos. Tales términos son definidos por actividades de filtrado y lematización de palabras en idioma español, que constituye la reducción de las palabras a su raíz semántica.La actual investigación se centra en la actividad de generar el cuadro de clasificación de forma automática a partir de lo definido en los vectores de los archivos. Todo lo anterior prepara las bases para la automatización del proceso de clasificación archivística.

1.1.1 Clasificación archivística

La **clasificación archivística** puede definirse como un sistema de clases según el cual se distribuyen los objetos de cualquier género, sobre la base de rasgos diferenciales que les son inherentes; en este sistema cada clase ocupa un lugar determinado constante, con relación a las otras clases. Su objetivo es el de asignar archivos a un lugar exacto en un sistema de clasificación en el cual los distintos contenidos están agrupados de acuerdo con sus semejanzas o con las relaciones de uno con otros (15). La esencia de la clasificación en la archivística es sistematizar un complejo de archivos textuales y determinar el lugar de cada uno de ellos dentro de dicho complejo. Por tanto, la clasificación archivística es, en general, una operación que tiende a separar y reagrupar los grupos de archivos dentro de una estructura.

La clasificación archivística debe estar sujeta a un grupo de **características***,* entre las cuales se encuentran que la clasificación debe ser: procedente de lo general a lo particular, exhaustiva al alcanzar a todo el campo de cada materia, detallada expresando las ideas en todos sus grados, flexible permitiendo la combinación de ideas en todos sus grados, lógica respondiendo a la mecánica del pensamiento formal,

explícita y concisa, sencilla utilizando una notación fácil de escribir y recordar, expansiva siendo capaz de incorporar nuevos elementos, utilizar elementos complementarios como índices, tablas y otros, y debe ser sometida a revisiones periódicas (16).

Existen varios **tipos de sistemas** para lograr la clasificación de archivos, entre los cuales se definen: el sistema enumerativo, el pre coordinado y el post coordinado (17). Existen otros sistemas como el sistema jerárquico en el cual cada subclase va precedida inmediatamente de una sola clase, donde las materias se desarrollan de lo más general a lo más particular y de lo simple a lo complejo. Otro sistema es el facetico que posee tablas de características o aspectos únicos con un sistema de signos que permiten la clasificación multifacética de un archivo documental, o sea, permite combinar las características de un archivo dosificándola de una forma sintética y profunda con ayuda de una fórmula facetica, estos dos sistemas dan lugar al sistema semifacético, enumerativo y jerárquico que permite una indización multifacética con bastante profundidad, ya que posee todo un sistema de signos que permite combinar los términos de clasificación, así como utilizar una fórmula facetica (18). Este último sistema es el generalmente utilizado para la construcción de cuadros de clasificación archivística ya que combina los anteriores dándole la estructura jerárquica que necesita el cuadro de clasificación más la clasificación que requiere cada archivo.

Para lograr realizar una correcta descripción de la clasificación de archivos, se debe tener en cuenta al cuadro de clasificación archivística como instrumento clave para la representación de la clasificación realizada que refleja la estructura de un Sistema de gestión de archivos con base de las características y contenido de los archivos. La organización de los archivos se realiza de acuerdo al mismo orden en que se dividen y subdividen las diversas clasificaciones de los archivos, llegando hasta su grado inferior, respetando el principio de procedencia: cada documento debe estar colocado en el fondo de que proviene y, en ese fondo en su lugar de origen, es decir, conservar el orden establecido por la institución que creó los documentos.

La clasificación archivística presenta varias **etapas o fases** comenzando por el análisis temático o conceptual que determina el grupo de conceptos o temáticas a la cual está sujeta el contenido de los archivos. En la fase siguiente se encuentra la traducción a los términos controlados que permite un mayor conocimiento de la terminología del archivo, para así hacer una selección de la notación que será usada para mostrar determinadas características. Seguido a ello, es necesaria la elección de la clasificación a la que será sometida el documento y el establecimiento de políticas locales de clasificación (19).

Todas las características y etapas de la clasificación archivística han sido agrupadas en sistemas de gestión de archivos que brindan una mayor comodidad en el trabajo con documentos y que agilizan todo el proceso. Mediante las características de la clasificación se definen los requisitos que se deben perseguir ante el proceso de clasificación. Por otra parte, las etapas o fases logran proveer un marco teórico a seguir para el procedimiento de clasificación archivística. Mediante estos sistemas, la clasificación de documentos se potencia considerablemente estableciendo estándares en la clasificación archivística.

1.1.2 Sistemas de gestión de archivos

Los **sistemas de gestión de archivos** aglomeran un conjunto de normas, técnicas y prácticas usadas para administrar el flujo de archivos de todo tipo en una organización y permitir la recuperación de información desde ellos. Estos sistemas se ocupan del procesado, almacenamiento, búsqueda, recuperación y distribución de archivos, uso de tecnología y procedimientos que permiten la gestión y el acceso unificado a información generada en una organización (15). Muchos de los sistemas de gestión de archivos se han informatizado, creando softwares altamente potentes para la gestión documental.

Un **sistema informático de gestión archivos** es todo aquel programa de ordenador creado para la gestión de grandes cantidades de archivos. También suelen rastrear y almacenar archivos electrónicos o imágenes de documentos en papel. Estos documentos de archivo no tienen una organización clara de sus contenidos, al contrario de lo que suele suceder con la información almacenada en una base de datos. La combinación de este tipo de bibliotecas de archivos textuales con índices almacenados en una base de datos permite el acceso rápido mediante diversos métodos a la información contenida en los archivos (20). Los sistemas de gestión de archivo comúnmente proporcionan medios de almacenamiento, seguridad, así como capacidades de recuperación e indexación.

La gestión de documentos electrónicos de archivo es compleja y exige la correcta aplicación de una gran variedad de funciones. Es obvio que el sistema que colme tales necesidades precisa software especializado, que puede consistir en un módulo especializado, en varios módulos integrados, en software desarrollado a la medida del usuario o en una combinación de varios tipos de programas informáticos. En todos los casos, siempre tendrán que existir procedimientos y políticas que complementen la gestión de forma manual.

Existen un grupo de especificaciones que describen un Modelo de Requisitos (21) para la gestión de documentos electrónicos de archivo e incide especialmente en los requisitos funcionales de la gestión de documentos electrónicos de archivo mediante un sistema de gestión de este tipo. La especificación se ha concebido de forma que pueda aplicarse en todas las organizaciones públicas y privadas que deseen introducir un sistema de gestión de archivos o bien quieran evaluar la capacidad del que ya poseen.

Según MoReq, existen un grupo de **funcionalidades** que deben ser aplicadas sobre un sistema de gestión de archivo para su correcta implementación. Primeramente, se encuentra el cuadro de clasificación, que constituye el elemento clave de cualquier sistema de gestión de archivos. Define el modo en que los documentos electrónicos de archivo se organizarán en expedientes, así como las relaciones entre dichos expedientes (21). Por esta razón, la configuración del cuadro de clasificación, definición de clases y expedientes y mantenimiento del cuadro de clasificación, constituyen funcionalidades clave en un sistema de gestión de archivo.

Como el **cuadro de clasificación** constituye el elemento importante de cualquier sistema de gestión de archivos es necesario especificar los requisitos necesarios para su correcta implementación. Existe un gran número de **requisitos**, pero entre los requisitos más importantes se define que el sistema debe

soportar del cuadro de clasificación de la organización y ser compatible con él. Debe permitir la utilización de un cuadro de clasificación en el que los expedientes se puedan representar dispuestos en una jerarquía con un mínimo de tres niveles. Tres niveles se consideran el mínimo esencial; en ciertos entornos serán necesarios más (21).

También debe permitir a los usuarios añadir nuevas clases en cualquier posición dentro de una clase, cuando no existan expedientes almacenados en ese punto. Téngase en cuenta que esto puede suceder en cualquier nivel. Si posee una interfaz gráfica, ésta deberá permitir la navegación y la exploración, en un entorno visual, de los expedientes y de la estructura del cuadro de clasificación, así como la selección, la recuperación y la presentación de los expedientes electrónicos y su contenido por medio de tal mecanismo. El sistema debe soportar además los metadatos de expedientes y clases del cuadro de clasificación (21).

Otro requisito importante es permitir la reubicación de un expediente y sus volúmenes, o bien de una clase completa de la jerarquía, en un lugar distinto del cuadro de clasificación. Asimismo, debe garantizar que todos los documentos electrónicos de archivo ya colocados sigan vinculados a los expedientes y volúmenes reubicados. El sistema debe permitir que un documento electrónico de archivo se pueda volver a clasificar en otro volumen de expediente electrónico (21).

En los sistemas de gestión de archivo en general, la clasificación es dada manualmente mediante expertos sobre el tema. Los gestores solo tienen que organizar los archivos en el cuadro de clasificación, o bien los archivos son adicionados manualmente al nivel donde pertenecen. Con la creciente disponibilidad de documentos en formato electrónico y la necesidad de ser procesados de manera automática, surge la posibilidad de abordar la clasificación de archivos de manera automática. Actualmente el uso de técnicas más modernas no necesita de una clasificación previa, sino que existen varios algoritmos propuestos para este tipo de clasificación. La mayor parte de ellos no son específicos para clasificar archivos, sino que se han propuesto para clasificar todo tipo de objetos. Sucede que algunos de éstos han sido utilizados para la clasificación de archivos (22) (23).

1.1.3 Clasificación automática

Por **clasificación automática** se entiende una extensa colección de algoritmos, ideas y técnicas tendientes a resolver racionalmente el problema general de clasificación de objetos (24). Para poder llevar a cabo la clasificación automática de archivos digitales textuales es preciso contar con una serie de elementos previos.

Primeramente, se deben digitalizar todos los documentos físicos mediante un proceso tecnológico que permita, mediante la aplicación de técnicas fotoeléctricas o de escáner, convertir la imagen contenida en un documento en papel en una imagen digital. En el **proceso de digitalización** se emplean diferentes tecnologías para convertir imágenes de caracteres en letra manuscrita, en caracteres capaces de ser interpretados o reconocidos por un ordenador (2). El **reconocimiento óptico de caracteres** es un proceso dirigido a la digitalización de textos, los cuales identifican automáticamente a partir de una imagen símbolos

o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlos en forma de datos (2). Así se puede interactuar con estos mediante un programa de edición de texto o similar.

En el caso de la clasificación de archivos, lo más importante es contar con una forma consistente de representar cada archivo textual (su contenido). La **minería de texto**, también conocida como minería de datos textuales o descubrimiento de conocimiento desde bases de datos textuales, se define como el proceso de extraer patrones interesantes y nuevos conocimientos desde archivos de textos no estructurados. Esta constituye un campo multidisciplinario al tener involucrado otras disciplinas como: la recuperación de información, el análisis de los textos, la extracción de información, el clústering, la categorización, la visualización, el almacenaje de datos, y los sistemas de aprendizaje (8).

La minería de texto consta de dos etapas principales. La primera es llamada etapa de pre procesamiento, donde los textos se transforman en algún tipo de representación estructurada o semiestructurada que facilite su posterior análisis. La segunda es llamada etapa de descubrimiento, donde las representaciones intermedias se utilizan para descubrir los patrones interesantes y los nuevos conocimientos (7).

En la etapa de pre procesamiento, el mecanismo más utilizado es el del bien conocido modelo vectorial (7) (8). En este cada documento puede ser representado mediante un vector de términos. Cada término lleva asociado un coeficiente o peso que trata de expresar la importancia o grado de representatividad de ese término en ese vector o documento. Este peso puede calcularse de forma automática a partir de diversos elementos, basándose en las frecuencias de los términos, tanto en el documento como en toda la colección con que se trabaje (8). Por tal motivo, el tamaño o número de términos de cada documento también juega un papel importante.

En la etapa de descubrimiento se enfoca al **aprendizaje automático** o aprendizaje de máquinas (del inglés, "**Machine Learning**"), el cual es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras aprender (23). De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. En los problemas de clasificación existen un gran número de algoritmos, donde el sistema de aprendizaje trata de clasificar una serie de vectores utilizando de una a varias categorías (25).

Independientemente del algoritmo de clasificación que se utilice, en algún momento será preciso estimar la similitud entre dos archivos. Si es representado cada documento mediante un vector, entonces se puede aplicar alguna de las muchas funciones de similitud disponibles para estimar la similitud entre dos vectores. Existen multitud de funciones de similitud propuestas ya incluso desde los años 50 (26).En realidad, trabajos experimentales han mostrado que los resultados obtenibles son muy parecidos con cualquiera de dichas funciones; lo cual se explica porque en realidad todas usan los mismos elementos de información (27). La más conocida y utilizada de éstas es la llamada función del coseno, en referencia al coseno de ángulo que forman dos vectores que se representan en un espacio multisectorial (28).

Por lo general, cuando se habla de clasificación automática se distingue entre dos escenarios diferentes, que obviamente, requieren soluciones distintas. Estos escenarios reciben diversos nombres, pero básicamente consisten en lo siguiente: de un lado, una situación en la que se parte de una serie de clases o categorías conceptuales prediseñadas a priori, y en la que labor del clasificador (manual o automático) es asignar cada documento a la clase o categoría que le corresponda. En el segundo escenario posible, no hay categorías previas ni esquemas o cuadros de clasificación establecidos a priori. Los archivos se agrupan en función de ellos mismos, de su contenido; de alguna manera, se puede decir que se auto organizan. Es lo que se conoce como clasificación (automática) no supervisada o clústering; no supervisada porque se efectúa de forma totalmente automática, sin supervisión o asistencia manual (29).

El primer tipo o escenario se conoce como **clasificación supervisada**, no sólo porque requiere la elaboración manual o intelectual del cuadro o esquema de categorías, sino también porque requiere un proceso de aprendizaje o entrenamiento por parte del clasificador, que debe ser supervisado manualmente en mayor o menor medida (22). El segundo tipo o escenario llamado **clasificación no supervisada**, está basado en una estrategia ambiciosa y difícil puesto que su planteamiento original no presupone ningún conocimiento previo sobre lo que se quiere aprender, a diferencia de los algoritmos de aprendizaje supervisado que si exigen tener conocimiento previo para llevar a cabo sus objetivos. Ellos inducen la descripción de un concepto a partir de la presentación de diferentes instancias de este (22).

El uso de clasificación supervisada al necesitar de un cuadro o esquema de categorías creado, necesita de un grupo de recursos humanos necesarios para poder ser creado, además de que un experto que supervise el proceso de aprendizaje o entrenamiento, el cual puede concurrir en errores humanos. Con el objetivo de un mayor aprovechamiento de los recursos y un desarrollo más eficiente en la clasificación se debe hacer uso de la clasificación automática no supervisada. En este tipo de clasificación se encuentra el agrupamiento jerárquico el cual busca construir una jerarquía de grupos, dándole respuesta a la necesidad del cuadro de clasificación de poseer una estructura que pueda ser representada dispuestos a una jerarquía.

El cuadro de clasificación archivística necesita entonces hacer uso de algoritmos de agrupamiento jerárquico que posibilitan la creación de una estructura jerárquica correspondiente al mismo, utilizando algoritmos de aprendizaje automático referentes a la clasificación no supervisada para determinar las categorías. El uso de algoritmos de aprendizaje automático obliga a un estudio sobre medidas de similitud entre dos archivos textuales para asistir en el desarrollo de estos algoritmos. Además, se debe prever que los archivos a clasificar estén previamente digitalizados mediante procesos de reconocimiento óptico de caracteres para así poder ser analizadas utilizando técnicas de minería de texto.

Para realizar la clasificación automática mediante un algoritmo jerárquico se deben seguir un conjunto de requisitos que se encuentran en las especificaciones concebidas por MoReq, específicamente para la construcción de cuadro de clasificación que los expedientes se tengan que representar dispuestos a una jerarquía con un mínimo de tres niveles sin restringir el máximo de niveles.

1.2 Estudio de los antecedentes de la investigación

Una vez aclarados los conceptos fundamentales involucrados en la investigación, el siguiente paso consiste en investigar todo lo relacionado sobre la situación actual de los sistemas de gestión de archivos que permiten la construcción del cuadro de clasificación archivística, para entonces establecer los antecedentes de la investigación.

Se necesita también entender cuáles son los principales procesos de clasificación y agrupamiento jerárquico de archivos digitales textuales y como es su funcionamiento. Conocer las medidas de similitud entre archivos textuales es una característica distintiva de todos los procesos de clasificación y agrupamiento de archivos digitales textuales, por lo que su estudio se hace obligatorio.

1.2.1 Actualidad de los sistemas de gestión de archivo

Al analizar elementos a implementar en un **sistema de gestión archivos** en un entorno profesional existen **en la actualidad** una gran variedad de soluciones, tanto de software libre como propietarias. Todas estas soluciones aportan salidas a las necesidades primarias de gestión archivística de las empresas, permitiendo gestionar cómodamente archivos mediante control de versiones, flujos de trabajo asociados, publicación remota o búsqueda avanzada a texto completo, además de integración con suites ofimáticas habituales como Microsoft Office y Open Office (30). Sus implementaciones son sencillas si lo que se quieren cubrir son necesidades no muy específicas permitiendo fácil uso.

Los sistemas de clasificación de archivos en la actualidad ofrecen un grupo de **características** como colaboración, flujos de trabajo, búsquedas eficientes, flexibilidad, seguridad, velocidad, facilidad de uso, entorno web, soporte para Gestión de Contenidos Empresariales (incluidas Gestión Documental y Gestión de Activos Digitales), soporte para Gestión de Contenido Web, fácil despliegue y administración, escalabilidad, etiquetado automático de archivos, incorporación de tecnología semántica a los procesos de gestión documental, facilidad de integración de la solución con otras aplicaciones, modelo de negocio adaptable a cada situación en coste e implantación de soluciones concretas, búsquedas semánticas, clasificación automática de archivos, construcción de relaciones semánticas entre archivos que permiten resultados de búsqueda más cercanos a nuestros deseos (gracias al procesamiento del lenguaje natural), y extracción automática de metadatos de los archivos textuales (12). Aunque es un gran número de características son pocas las aplicaciones informáticas de gestión archivística que garantizan su totalidad.

Existe una variedad de opciones que, si se desea implementar en la organización un sistema de gestión de archivos, este permitirá gestionar cómodamente grandes conjuntos de archivos, los cuales utilizan clasificación archivística. Entre los más reconocidos se encuentran: Microsoft SharePoint, DocPatch Aspen, Lakaut, Smile, LogicalDOC, Abox document, Navegadoc, OpenKM, R2Docuo, Nuxeo, Alfresco y Athento (13) (12) (31). Todos constituyen software de gestión documental altamente potentes que permiten la gestión de grandes cantidades de archivos mediante cuadros de clasificación archivística. Cada uno de estos sistemas tienen fortalezas y debilidades, independientemente de que sistema gestor es mejor, y que herramientas brinda uno sobre otro, se puede agrupar en una característica generar de que son pocos los

sistemas de gestión documental que permiten la clasificación automática de documentos. Del grupo de sistemas de gestión mencionados, que son los más representativos, solo DocPatch Aspen, Lakaut y Athento (13) (32) (12), usan algún tipo de sistema de clasificación automática de documentos y de estos solo Athento usa tecnología semántica y clasificación por contenido (12). Aunque este software se encuentra bajo la licencia “ATHENTO SHARED SOURCE LICENSE” donde los términos de la licencia no son considerados como software libre (12). Por otra parte, otorgan beneficios frente a software licenciado bajo el modelo de “Software Propietario”, como son: el acceso al código y la capacidad de modificarlo. Aunque en el apartado de redistribución del software este especifica que no permite la copia o redistribución del software incluido su código fuente o su forma compilada.

La mayoría de los sistemas de gestión de archivos inteligente, que utilizan clasificación automática de archivos, son softwares propietarios, o pertenecen a organizaciones con fines altamente lucrativos que protegen sus sistemas bajo licencia. Los softwares de gestión de archivos que están de acuerdo con la iniciativa Open Source, sobre el grupo estudiado, no presentan herramientas que permitan clasificación automática de documentos, dejando como única opción softwares bajo licencias no libres.

Los softwares de gestión de archivos inteligente que puedan ser adquiridos para generación automática del cuadro de clasificación archivística, en muchos de los casos necesitan de reformas para responder a las funcionalidades del negocio, lo que incurre en gastos, o en el caso de que el propietario del software permita su modificación, pierde el soporte de software brindado. Tanto software propietario como software bajo licencia, no permiten la redistribución de su software, por lo que incluir un sistema de gestión de archivo a una herramienta con fines comerciales no es posible. La mejor opción es utilizar técnicas de minería de texto para recrear las funciones avanzadas de estos sistemas de gestión de archivo para la generación automática del cuadro de clasificación archivística (33).

El trabajo de diploma (9) desarrollado en la Universidad de las Ciencias Informáticas en el año 2015, responde a la primera etapa de la minería de texto, la etapa de pre procesamiento. Este trabajo se enfoca en la representación mediante un vector de términos, donde cada término lleva asociado un coeficiente o peso que trata de expresar la importancia o grado de representatividad de ese término en ese vector (9). Lo anterior brinda una solución informática para realizar dicha representación capaz de integrarse a sistemas informáticos de múltiples plataformas, que permita el procesamiento de archivos digitales textuales y que analice la mayor cantidad sin importar el tema que traten.

A este punto fueron expuestos el precedente de la investigación y las necesidades actuales, caracterizadas las actividades que intervienen en la clasificación archivística, así como los elementos para su automatización. Por tal motivo, se deriva la necesidad de crear un programa capaz de dar frente al problema de clasificación mediante el uso de algoritmos jerárquicos no supervisados, donde se obtendría una estructura jerárquica que serviría para generar una primera aproximación del cuadro de clasificación archivística. Esta estructura jerárquica homologaría el dendodrama generado por algoritmos de agrupamiento jerárquico a la estructura de árbol que presenta el cuadro de clasificación archivística.

1.2.2 Algoritmos de agrupamiento jerárquico

El agrupamiento jerárquico o clústering jerárquico se enfoca en intentar crear una jerarquía de descomposición de la colección de archivos. Tienen por objetivo agrupar clústeres para formar un nuevo clúster o bien separar un clúster ya existente para dar origen a otros dos clústeres, de tal forma que, si sucesivamente se va efectuando este proceso de aglomeración o división se maximice en alguna medida la similitud. Los métodos jerárquicos se subdividen en aglomerativos y divisivos (22). Cada una de estas categorías presenta una gran diversidad de variantes.

Los **métodos aglomerativos**, también conocidos como ascendentes, comienzan el análisis con tantos grupos como individuos haya. En primer lugar, se crea una partición donde todos los clústeres están compuestos por un único objeto. Posteriormente y de forma iterativa, se van uniendo los dos clústeres más cercanos hasta que la partición obtenida tenga un único clúster, con todos los objetos (34). Como consecuencia se crea una jerarquía con tantas particiones como objetos existen, donde en cada partición se unen exactamente dos clústeres de la partición anterior.

Existen diferentes medidas utilizadas en el clústering para calcular la similitud entre clúster y dependiendo de la medida utilizada este método permite emplear distintos algoritmos. Por ejemplo, el algoritmo single-link mostrado en la figura 1.1 considera que la similitud entre clústers es la similitud entre los objetos más cercanos de cada clúster.

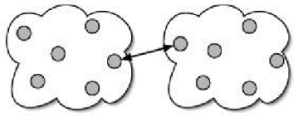

Figura 1.1 Single-Link

Sin embargo, el complete-link de la figura 1.2 define la similitud entre los clústeres como la similitud entre sus dos casos más lejanos (22).

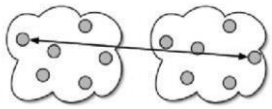

Figura 1.2 Complete-Link

Aparte de estos dos algoritmos, existen otras medidas de proximidad entre clústers y, por lo tanto, algoritmos basados en clúster jerárquico. Uno de ellos es el algoritmo Ward que trata de minimizar el error cuadrático medio. Otros cuatro algoritmos comunes denominados UPGMA (o average-link) (Figura 1.3), WPGMA, UPGMC y WPGMC (DUBES. 1988; SOKAL 1973) surgen de la combinación de dos criterios. El prefijo “U” (Unweighted) significa que todos los casos pesan igual, mientras que “W” (Weighted) significa que todos los clústeres pesan igual, de forma que los casos en los clústeres pequeños pesan más. El sufijo “A” (Average) significa que la similitud se calcula mediante la media entre todos los casos de cada clúster, mientras que “C” (Centroid) significa que la similitud entre clúster se define como la similitud entre centroides (22).

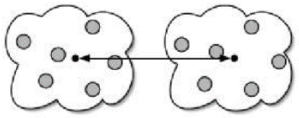

Figura 1.3 Average-Link

En (35) se propone una fórmula con cuatro coeficientes que sirven para actualizar la matriz de similitudes y se adecúa a la mayoría de medidas de similaridad conocidas. Todas estas funciones de similaridad, que emplean los métodos jerárquicos anteriormente mencionados, se pueden ver como una selección de parámetros en la fórmula que se muestra a continuación.

Suponiendo que se ha unido los clúster C_i y C_j para crear el nuevo clúster C_{ij}. Entonces la similitud entre el nuevo clúster C_{ij} y un clúster C_l se puede calcular como:

$$D\left(C_l,\left(C_i\,,C_j\right)\right)=\alpha_i\,D(C_l,C_i)+\alpha_j\,D\left(C_l,C_j\right)+\beta D\left(C_i,C_j\right)+\gamma|D(C_l,C_i)-D\left(C_l,C_j\right)|$$

Donde D es la función de proximidad entre los clúster.

En la tabla 1.1 se indican los valores de los coeficientes para las medidas de proximidad mencionadas anteriormente.

Algoritmos	α_i	α_j	β	γ
Single-Link	$1/2$	$1/2$	0	$-1/2$
Complete-Link	$1/2$	$1/2$	0	$1/2$
UPGMA	$\frac{\lvert C_i\rvert}{\lvert C_i\rvert+\lvert C_j\rvert}$	$\frac{\lvert C_j\rvert}{\lvert C_i\rvert+\lvert C_j\rvert}$	0	0
WPGMA	$1/2$	$1/2$	0	0
UPGMC	$\frac{\lvert C_i\rvert}{\lvert C_i\rvert+\lvert C_j\rvert}$	$\frac{\lvert C_j\rvert}{\lvert C_i\rvert+\lvert C_j\rvert}$	$\frac{-\lvert C_i\rvert\lvert C_j\rvert}{(\lvert C_i\rvert+\lvert C_j\rvert)^2}$	0
WPGMC	$1/2$	$1/2$	$-1/4$	0
Ward	$\frac{\lvert C_i\rvert+\lvert C_l\rvert}{\lvert C_i\rvert+\lvert C_j\rvert+\lvert C_l\rvert}$	$\frac{\lvert C_j\rvert+\lvert C_l\rvert}{\lvert C_i\rvert+\lvert C_j\rvert+\lvert C_l\rvert}$	$\frac{-\lvert C_l\rvert}{\lvert C_i\rvert+\lvert C_j\rvert+\lvert C_l\rvert}$	0

Tabla 1.1 Coeficientes para medidas inter-clúster

Esta es una forma de simplificar los métodos combinatorios en una única fórmula, pero también permite definir con sencillez, nuevas medidas de proximidad. Además, permite utilizar técnicas de optimización para instanciar los cuatro coeficientes de forma que se pueda definir una medida de proximidad adaptada a cada problema.

Los **métodos divisivos**, constituyen el proceso inverso a los aglomerativos. Esta clase de procedimientos es bastante menos popular que los aglomerativos por lo que la literatura sobre ellos no es muy extensa. Comienzan con un conglomerado que engloba a todos los casos tratados y a partir de este grupo inicial, a través de sucesivas divisiones, se van formando grupos cada vez menores. Al final del proceso se tienen tantas agrupaciones como casos han sido tratados (36).

En cuanto a la clasificación de estos métodos se puede reutilizar la filosofía de los métodos aglomerativos puede mantenerse los procedimientos enfocados en calcular la similitud entre los grupos y partir de un grupo único que hay que subdividir. Se sigue la estrategia de maximizar las similitudes, o minimizar la similaridades, puesto que se busca ahora los individuos menos similares para separarlos del resto del conglomerado.

Una cuestión importante que puede surgir en su desarrollo es el hecho de cuándo un clúster determinado debe dejar de dividirse para proceder con la división de otro conglomerado distinto. Dicha cuestión puede resolverse con la variante concebida para aquellas medidas de asociación que sean positivas pero aumenta ligeramente el costo computacional del algoritmo (37). Así se tiene un clúster unitario y otro formado por los restantes individuos.

A continuación, se añadirá al clúster unitario a aquel elemento cuya similitud total al resto de los elementos que componen su actual clúster menos la similitud al clúster anteriormente formado sea máxima (mínima). Cuando esta diferencia sea negativa dicho elemento no se añade y se repite el proceso sobre los dos subgrupos (35).

En el análisis de los algoritmos de agrupamiento jerárquico aglomerativos y divisivos, se demuestra una estrecha relación entre ambos y un procedimiento común en el uso de métricas inter-clúster. Cabe destacar una preferencia por parte de la literatura (10) (38) (39) en el uso de algoritmos aglomerativos. Otro punto a destacar es la necesidad de aplicar procedimientos en los algoritmos divisivos en posibles bloqueos durante el desarrollo de la aplicación que pueden aumentar el tiempo de ejecución. Tomando como principal indicador la amplia documentación existente sobre los algoritmos aglomerativos, es recomendable implementar el grupo de estos algoritmos para obtener así la estructura jerárquica necesaria.

Existe **otro grupo de algoritmos jerárquicos** como Chameleon el cual basa su funcionamiento en una primera fase, en construir el grafo de los k vecinos más cercanos y usa un algoritmo de particionamiento de grafo para agrupar los puntos en subgrupos. En una segunda fase, se usa un algoritmo jerárquico aglomerativo para encontrar los clústeres genuinos combinando repetidamente estos subgrupos. En esta última fase determina el par de subgrupos más similares tomando en cuenta su interconectividad y cercanía, éstas expresan las características internas de los subgrupos, el modelo no es estático, sino que es capaz de adaptarse a las características internas de los subgrupos según estos van cambiando (22).

Este algoritmo es diferente de los algoritmos jerárquicos anteriormente estudiados porque permite unir varios pares de subgrupos en la misma iteración. Esto logra una disminución considerable en los tiempos de ejecución, pero en ningún momento garantiza una mejor calidad del agrupamiento, ni establece una jerarquía entre los grupos bases, lo cual evita el desarrollo jerárquico en su totalidad, el cual puede ser necesario en algún punto de esta investigación.

El algoritmo CURE constituye un algoritmo híbrido entre los dos enfoques jerárquico y particional, que trata de emplear las ventajas de ambos y de eliminar las limitaciones. En éste, en lugar de usar un solo punto como representante de un grupo se emplea un número c de puntos representativos del grupo. La similaridad entre dos grupos se mide por la similaridad del par de puntos representativos más cercanos, uno de cada grupo. Para tomar los puntos representativos selecciona los c puntos más dispersos del grupo y los atrae hacia el centro del mismo por un factor de contracción α, en cada paso se unen los dos grupos más cercanos y una vez unidos se vuelve a calcular para éste su centro y los c puntos representativos (22).

Con este algoritmo se encuentran grupos de diferentes tamaños. Con el método de sacar c puntos representativos y atraerlos hacia el centro del grupo CURE maneja el ruido y puntos lejanos presentes. Pero a su vez contiene los mismo limitantes del algoritmo Chamaleon por lo cual tampoco es recomendable usarlo en la actual investigación.

1.2.3 Métricas de similitud

Cuando se agrupan objetos es necesario conocer el grado de asociación que poseen para poder determinar a cuál clúster serán asociados. Esta puede estar dada por una medida de similitud o una medida de similitud o disimilaridad. Algunos algoritmos de clústering exigen una medida específica, pero otros dejan a elección del desarrollador cuál usar (16).

Existen multitud de funciones de similitud propuestas desde los años 50 (26). Trabajos experimentales han mostrado que los resultados a obtener son muy parecidos con cualquiera de dichas funciones; lo cual se explica porque en realidad todas usan los mismos elementos de información (27). En la comparación de vectores generados a partir del análisis del contenido de documentos de archivo, los coeficientes Dice, Jaccard y el Coseno han mostrado los mejores resultados (27). El Coeficiente Dice plantea:

$$S(D_i, D_j) = \frac{2\sum_{k=1}^{L}(\omega_{ik} * \omega)}{\sum_{k=1}^{L}\omega^2{}_{ik} + \sum_{k=1}^{L}\omega^2{}_{jk}}$$

Donde D_i y D_j son los documentos de archivo que se están comparando, y ω es la importancia del termino k en i o j. Si el peso de los términos es binario la ecuación se reduce a:

$$S(D_i, D_j) = \frac{2C}{A + B}$$

Donde C es el número de términos que $D_i \; y \; D_j$ tienen en común, y A y B son el número de términos de $D_i \; y \; D_j$ respectivamente. El coeficiente de Dice siempre está en un rango [0, 1], donde 1 es similaridad máxima, por tanto, igualdad absoluta y 0 viceversa (26).

El coeficiente Jaccard se define como:

$$S(D_i, D_j) = \frac{2\sum_{k=1}^{L}(\omega_{ik} * \omega_{jk})}{\sum_{k=1}^{L}\omega^2{}_{ik} + \sum_{k=1}^{L}\omega^2{}_{jk} - \sum_{k=1}^{L}(\omega^2{}_{ik} - \omega^2{}_{jk})}$$

Siempre toma valores entre 0 y 1, correspondiente este último a la igualdad total entre ambos conjuntos (26).

El Coeficiente Coseno se define como sigue:

$$S(D_i, D_j) = \frac{\sum_{k=1}^{L}(\omega_{ik} * \omega_{jk})}{\sqrt{\sum_{k=1}^{L} \omega^2{}_{ik}} * \sqrt{\sum_{k=1}^{L} \omega^2{}_{jk}}}$$

Esta medida es ampliamente usada (11) (37) (8) (33) (22) y basa su funcionamiento en el coseno del ángulo entre dos objetos. Mide la desviación o semejanza entre dos vectores por el número de grados del ángulo que forman. Si el ángulo es cercano a cero la semejanza se aproximará a uno, entonces estos documentos son semejantes, lo contrario denotaría diferencia. Esto es posible porque crean una estructura triangular, que se puede observar en la figura 1.4, a la que se aplica el cálculo del ángulo que forma la hipotenusa (en este caso el vector del documento de archivo D_i) y el adyacente (el vector del documento de archivo D_j) que resulta ser el coseno del triángulo (26).

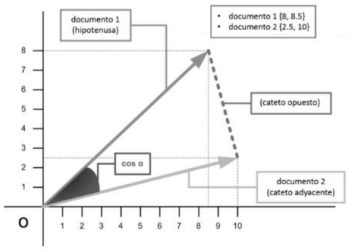

Figura 1.4. Cálculo del coseno

Según la bibliografía utilizada la similitud coseno es la más utilizada para comparar documentos de archivos, pero no definen resultados variantes entre una y otra métrica. En cambio, la variación del valor de similitud, aunque sea pequeña evaluada en una gran cantidad de archivos, puede generar resultados completamente diferentes. Al no presentar un parámetro de recomendación en la actual bibliografía estudiada, la mejor opción es aplicar los diferentes coeficientes de similitud para comprobar el comportamiento de cada algoritmo de agrupamiento ante los diferentes coeficientes.

1.2.4 Análisis de criterios de poda

Con frecuencia, cuando se emplean técnicas clústering jerárquicas, el investigador no está interesado en la jerarquía completa sino en un subconjunto de particiones obtenidas a partir de ella. Las particiones se obtienen podando el dendograma o seleccionando una de las soluciones en la sucesión encajada de clúster que comprende la jerarquía (34).

Desafortunadamente este paso fundamental esta entre los problemas que todavía no están totalmente resueltos. Entre las razones más importantes que se pueden citar para que dicho problema siga siendo un campo abierto están la inexistencia de una hipótesis nula apropiada que radica en la falta de una definición

clara y comprensiva de lo que significa no estructurar en un conjunto de datos y en la naturaleza compleja de las distribuciones muéstrales multivariables (35).

Las soluciones propuestas a estas cuestiones han sido múltiples (35). Algunas soluciones se basan en múltiples reglas, algunas de estas reglas son simples métodos heurísticos (34), otras están basadas en contrastes de hipótesis formales, los cuales han sido desarrollados al amparo de la hipótesis de la existencia de una determinada distribución muestral (35), mientras que otros son procedimientos heurísticos pero que extraen la filosofía de los contrastes existentes en poblaciones normales (39). Es necesario que mencionar que son variados los procedimientos que en los últimos años han sido desarrollados, con frecuencia orientados a técnicas particulares.

La primera técnica que puede ser citada se basa simplemente en cortar el dendodrama de forma subjetiva tras visualizarlo. Obviamente este procedimiento no es nada satisfactorio puesto que esta generalmente sesgado por la opinión que el investigador posee sobre sus datos e imposibilita que la poda se realice automáticamente.

Un método más formal, se basa en representar en una gráfica el número de clústers que se observan en los distintos niveles del dendrograma frente a los niveles de fusión a los que los clústeres se unen en cada nivel. La presencia de una pendiente poco pronunciada sugiere que la siguiente unión de clúster no aporta información adicional sobre el nivel anterior (35). Este método, por lo tanto, se basa en la existencia de pequeños saltos o discontinuidades en los niveles de fusión.

Mojena (39) siguió con la idea de estudiar los saltos relativos en los valores de fusión y sugirió otro procedimiento heurístico bastante divulgado. En su método se compara el valor de fusión de cada etapa con el promedio de los valores de fusión sumado con el producto de una cierta constante por la cuasi desviación típica de los valores de fusión. Cuando un valor de fusión supera dicha cantidad se concluye que el nivel precedente es el que origina la solución óptima. Mojena (39)propuso además que el valor de la constante debía de estar comprendido en el rango de 2.75 a 3.50 aunque en (39), tras una detallada investigación de valores en función del número de clústers, establece que el valor óptimo para dicha constante debe ser 1.25.

La poda del dendrograma consiste en quitar los grupos demasiado específicos para que la búsqueda sea más eficiente. La idea es que, si dos grupos hermanos o un grupo padre y su hijo son muy similares, pueden formar un solo grupo. Un método centrado en grupos es FIHC, siglas de Frequent Itemset-based Hierarchical Clústering. El mismo permite la creación de grupos descendientes a varios niveles. A niveles intermedios pueden existir grupos que contengan pocos o ningún objeto. La poda en profundidad elimina estos grupos (38). Para ello obtiene la similitud entre un grupo y el grupo que lo antecede de la forma que se detalla a continuación.

Sean G_i y G_j dos grupos, $doc(G_i)$ es la combinación de todos los documentos del subárbol G_j como un único documento X representa la frecuencia global de los términos en el documento G_j que son objetos

frecuentes en G_i y X' representa los términos globales frecuentes en el ($doc(G_j)$ que no son términos frecuentes en G_i. La similitud del grupo G_i con el grupo G_j, se calcula mediante:

$$S(G_i \leftarrow G_j) = \frac{P(G_i \rightarrow doc(G_j))}{\sum_x n(x) + \sum_{x'} n(x')} + 1$$

donde $n(x)$es el peso de la frecuencia de X en el vector característico de $doc(G_j)$; $n(X')$ es el peso de la frecuencia de X' en el vector característico $doc(G_j)$ (38).

Sea G_i el grupo i, d_j el documento j, x el conjunto de términos frecuentes globales que pertenecen al grupo y x' el conjunto de términos frecuentes globales que no pertenecen al grupo. Para determinar a qué grupo G_i pertenece el documento d_j, se aplica la fórmula siguiente:

$$P(G_i \leftarrow d_j) = \left[\sum_x n(x) * G_r(x)\right] - \left[\sum_{x'} n(x') * G_l(x')\right]$$

Donde $n(x)$ y $n(x')$ es la suma de las frecuencias (TFxIDF) de x y x' respectivamente en el vector característico del documento j (38). Existen dos parámetros de clasificación el porcentaje mínimo de aparición de un término en la colección para que sea considerado término frecuente. Calculado mediante el promedio de todas las frecuencias globales de los términos que son inferior a la mediana de las frecuencias globales. Y otro parámetro es el porcentaje mínimo de aparición de un conjunto de términos frecuentes, obtenido por la menor cantidad de elementos de un grupo considerados frecuentes en el proceso de agrupamiento entre la aparición total de todos los términos frecuentes. El primero se conoce como frecuencia mínima global o soporte global y el segundo frecuencia mínima grupal o soporte de grupo (38). La similitud que comparten ambos grupos se calcula como:

$$S_{IC}(G_i \leftrightarrow G_j) = \sqrt{S(G_i \leftarrow G_j) * S(G_j \leftarrow G_i)}$$

El valor de la similitud es un número en el rango [0, 2]. Si el valor de similitud que comparten ambos grupos es mayor que 1, entonces éstos se unen (38).

Una vez analizados los distintos procedimientos para agrupar clústeres mediante la poda del dendrograma, en esta investigación se optó por FIHC, el cual no requiere la supervisión de un experto, su eficiencia no depende del conocimiento previo del usuario tal como el número de grupos y produce una estructura jerárquica que facilita la búsqueda de documentos relevantes. Se adecua además a la actual investigación porque utiliza como elemento esencial para la función de la poda la frecuencia de aparición de términos, siendo el peso de los objetos a agrupar frecuencias de igual forma.

1.2.5 Métricas para evaluar el agrupamiento

Con el objetivo de poder comprobar la validez de la implementación realizada respecto a lo abordado en la literatura, se ha propuesto tratar métricas de evaluación que permitan validar la calidad de los agrupamientos y así demostrar que los resultados pueden ser aplicados. Existen dos tipos de métricas que son ampliamente usadas (40). Un tipo de métrica es la que permite comparar diferentes conjuntos de grupos sin referenciar a conocimiento externo, llamada métrica de calidad interna. Un ejemplo de métrica

interna es la métrica overall similarity basada en la similitud de pares de objetos en un clúster. El otro tipo de métricas es la que permiten evaluar cuán bueno es el agrupamiento y se basan en la comparación de los grupos producidos por las técnicas de agrupamiento, donde se conocen a partir del conjunto de datos de prueba las clases de la colección. Este tipo de métrica es llamada métrica de calidad externa. Ejemplos de métricas externas son la entropía y F-Measure (40). En la ausencia de información externa, tales como las etiquetas de clases, el uso de una métrica interna es necesaria.

La métrica **overall similarity** basada en la similitud de pares de objetos en un clúster, permite usar la cohesión de los clústeres como una métrica de similitud de clústeres (41). Un método para calcular la cohesión de un clúster es usar la similitud pesada de la similitud interna del mismo según la métrica siguiente:

$$O_s = \frac{1}{|S|^2} \sum_{\substack{i \epsilon S \\ j \epsilon S}} S(Di, Dj)$$

donde |S| es el número de documentos que pertenecen al clúster a evaluar y $S(Di, Dj)$ la similitud entre dos documentos de archivo i y j respectivamente (42). En (41) utilizan el cociente Coseno para calcular la similitud entre los documentos de archivo.

El otro tipo de métricas es la que permiten evaluar cuán bueno es el agrupamiento es la **entropía** que puede utilizarse como métrica de calidad de los clústeres (con la advertencia de que la mejor entropía es obtenida cuando cada clúster contiene exactamente un documento) (43). Sea CS un resultado de agrupamiento, para cada clúster es calculada primero la distribución de las clases (por ejemplo para un clúster j se calcula p_{ij}, la probabilidad que un miembro del clústers j pertenezca a la clase i)

$$p_{ij} = \frac{n_j^i}{n_j}$$

donde n_j^i es el número de documentos de la clase i que están asignados al clúster j. Entonces, usando esa distribución de la clase, la entropía de cada clúster j es calculada usando la fórmula estándar:

$$E_j = -\sum_i p_{ij} \log(p_{ij})$$

donde la sumatoria es aplicada sobre todas las clases. La entropía total para el conjunto de clústeres i es calculada como la suma de las entropías de cada clúster y han sido pesadas por el tamaño de cada uno:

$$E_{cs} = \sum_{j+1}^{m} \frac{n_j * E_j}{n}$$

Donde n_j es el tamaño del clúster j, m es el número de clústers y n es el números total de documentos.

La segunda métrica de calidad externa es **F-Measure** (41). Es una métrica que combina las ideas de precision y recall de la recuperación de información. Se trata cada clúster como si este fuera el resultado de una consulta y cada clase como si esta fuera el conjunto de documentos deseados para una consulta.

Así, se calcula precision y recall de los clústeres para cada clase dada, más específicamente para el clúster j y la clase i

$$precision(i,j) = {}^{n_{ij}}/_{n_j} \qquad recall(i,j) = {}^{n_{ij}}/_{n_i}$$

donde n_{ij} es el número de miembros de la clase i en el clúster j, n_j es el número de miembros del clúster j y n_i es el número de miembros de la clase i (41). Entonces, según (41) F-Measure, del clúster j y la clase i es entonces dada por

$$F(i,j) = \frac{2 * recall(i,j) * precision(i,j)}{recall(i,j) + presicion(i,j)}$$

Un valor de F-Measure es calculado para toda la colección obteniendo un promedio pesado de todos los valores de las métricas F-Measure según la siguiente expresión

$$F = \sum_i \frac{n_i}{n} * \max\{F - Measure(i,j)\}$$

En la práctica diaria de la recuperación de información los niveles más altos de precisión generalmente se obtienen con valores bajos de recall. La siguiente expresión permite calcular el valor F-Measure, proporcionando un parámetro β que permite ponderar precision y recall.

$$F\beta(i,j) = \frac{(1+\beta^2) * recall(i,j) * precision(i,j)}{\beta^2 * recall(i,j) + presicion(i,j)}$$

En esta fórmula α puede verse como el grado relativo de importancia atribuida para precision y recall. Si β = 0 entonces $F\beta(i,j)$ coincide con precision, mientras que para mayor β entonces recall obtiene mayor relevancia. El valor β = 1 es retoma la fórmula original donde se considera igual importancia para precision y recall.

Para la evaluación de la calidad serán empleadas las tres métricas de calidad con el objetivo de obtener la mayor calidad posible para cada una de las métricas. Con la evaluación de cada métrica se puede asegurar una mayor seguridad en la selección de nuestro algoritmo y mayor variedad al seleccionar sobre que parámetros se desea mejorar esta calidad.

1.3 Metodología, Lenguajes y Herramientas de Desarrollo

En aras de preparar el ambiente de desarrollo del software, se realiza la identificación y el análisis de la metodología de desarrollo, lenguaje de programación, entorno de desarrollo, lenguaje y herramienta de modelado. En esta sección se introduce el análisis de las principales características y ventajas de cada uno de los elementos antes mencionados que justifican su elección como vía para dar solución a la problemática planteada en cumplimiento de los objetivos definidos.

1.3.1 Metodología de Desarrollo

Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo de software. Por una parte, se tienen aquellas más tradicionales que se centran especialmente

en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, así como las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en otros muchos, ya sea debido a la numerosa documentación que generan, al tamaño de los equipos que le utilizan, así como otras múltiples causas.

Por otra parte, están las metodologías ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas y generando poca documentación. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo calidad. Las metodologías ágiles están revolucionando la manera de producir software, pues de forma efectiva logran que bajo su uso la mayoría de los proyectos lleguen a feliz término con buenos resultados (44). Para guiar el desarrollo del software a implementar se decide el empleo de una metodología ágil. Dicha selección se basa en que el equipo de desarrollo involucrado es pequeño, existe una estrecha relación con el cliente, llegando este a formar parte del equipo y se necesitan constantes entregas del avance en el proceso de desarrollo. También se tiene en cuenta que durante este proceso el software se puede ver sometido a constantes cambios. A continuación, se describen cuatro metodologías de desarrollo ágil y se selecciona una de ellas.

Feature Driven Development

Feature Driven Development (FDD por sus siglas en inglés) es una metodología ágil diseñada para el desarrollo de software, basada en la calidad y el monitoreo constante del proyecto. Esta se enfoca en iteraciones cortas, que permiten entregas tangibles del producto en un período corto de tiempo, de como máximo dos semanas. Se preocupa por la calidad, por lo que incluye un monitoreo constante del proyecto y propone tener etapas de cierres cada dos semanas por lo que se obtienen resultados periódicos y tangibles. Además, se basa en un proceso iterativo con iteraciones cortas que producen un software funcional que el cliente puede ver y monitorear. Define claramente entregas tangibles y formas de evaluación del progreso del proyecto. No hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción (45).

La metodología FDD define cinco procesos desarrollados de forma iterativa: desarrollar el modelo global, construir una lista de características, planificar, diseñar y construir por características. El modelo global se construye teniendo en cuenta la visión, el contexto y los requisitos que debe tener el sistema a construir. Luego se elabora una lista que resume las funcionalidades que debe tener el sistema, la cual es evaluada por el cliente. Como tercera actividad se procede a ordenar los conjuntos de funcionalidades conforme a su prioridad y dependencia y se asigna a los programadores jefes. Posteriormente en la etapa de diseño se selecciona un conjunto de funcionalidades de la lista y se procede a diseñar y construir la funcionalidad mediante un proceso iterativo, decidiendo que funcionalidad se van a realizar en cada iteración. Este proceso iterativo incluye inspección de diseño, codificación, pruebas unitarias, integración e inspección de código. Luego se pasa a la última actividad que es la construcción del proyecto (44).

Es necesario resaltar que si bien el diseño del modelo global al inicio del desarrollo elimina confusiones y ayuda a comprender mejor las características que deben ser implementadas, se necesita tener un conocimiento amplio sobre los procesos del negocio y contar con un profesional de alta experiencia para el diseño del mismo. Además, el código no puede tomarse como una documentación, esto significa que es necesario generar documentación adicional para explicar el mismo.

SCRUM

Es una metodología especialmente desarrollada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir de la siguiente forma: el desarrollo de software se realiza mediante iteraciones denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Al finalizar un sprint, las tareas que se han realizado y el cliente ha quedado conforme con ellas, no se vuelven a modificar. El equipo de desarrollo trata de seguir el orden de prioridad que marca el cliente, pero puede modificarlo si determina que es mejor hacerlo y cada miembro del equipo trabaja de forma individual (44). Esta metodología se basa más en la administración del proyecto que en la propia programación o creación del producto.

Crystal Methodologies

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo, Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) (44).

Programación Extrema

La Programación Extrema (XP por sus siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como adecuada para proyectos con requisitos imprecisos y muy cambiantes y donde existe un alto riesgo técnico (44). Esta metodología propone un desarrollo iterativo e incremental que se basa en realizar pequeñas mejoras, unas tras otras. Las tareas de desarrollo se deben llevar a cabo por dos personas en un mismo puesto, asume que la calidad del código escrito de esta manera es más importante que la posible pérdida de productividad inmediata. Recomienda que un representante del cliente trabaje junto al equipo de desarrollo, así como la corrección de todos los errores antes de añadir una nueva

funcionalidad, haciendo entregas frecuentes. Para la investigación se decide emplear la metodología de desarrollo XP. Para ello se tienen en cuenta tres factores fundamentales: equipo de desarrollo, proyecto a desarrollar y cliente.

Equipo de desarrollo

El equipo de desarrollo está formado por un solo programador. Las ventajas de este estilo de programación son: muchos errores son detectados conforme son introducidos en el código (inspecciones de código continuas), por consiguiente, la tasa de errores del producto final es más baja, los diseños son mejores, no existen problemas de compatibilidad en la programación y el tamaño del código menor. Los problemas de programación se resuelven más rápido, la programación en sí, es mucho más rápida, se evita la necesidad de invertir tiempo compartiendo conocimiento o usarlo poniéndose de acuerdo entre los miembros del equipo. Menor consumo de recursos y mayor carga de trabajo para los recursos destinados, y disponibilidad en tiempo completo de las mismas. Conocimiento centralizado, la información brindada por el equipo de desarrollo no tiene contradicciones, ni enfoques variados. La dinámica del equipo es muy fluida, claramente ante la imposibilidad de contradicciones internas, ni puntos de vistas diferentes, u opiniones divergentes.

Proyecto a desarrollar

El proyecto a desarrollar se debe entregar en menos de un año, lo que se considera un período de tiempo corto. Se necesitan ciclos cortos tras los cuales se muestren resultados del proceso de desarrollo. Esto permite minimizar el tener que rehacer partes que no cumplen con los requisitos y ayuda a el programador a centrarse en lo que es más importante, en este caso la entrega del producto de software.

Cliente

El cliente forma parte del desarrollo del software, su opinión sobre el estado del proyecto se conoce en tiempo real y la comunicación entre este y el equipo de desarrollo es fluida. Esta integración posibilita que el cliente conozca en todo momento el estado de desarrollo del software. Además de que le permite al equipo de desarrollo conocer inmediatamente las inconformidades o cambios necesarios a medida que el cliente prueba el software.

Lenguaje de Modelado

El lenguaje unificado de modelado (UML por sus siglas en inglés) es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software (46). Este lenguaje permite a los analistas y desarrolladores del software representar todos los elementos necesarios para el proceso de desarrollo de software. En general UML representa en un lenguaje común y formal el desarrollo de cualquier producto de software. A pesar de que la metodología XP no define el uso UML, este se escoge para el modelado de los artefactos diagrama de dominio y diagrama de clases, que permiten establecer las relaciones entre los objetos para la comprensión de la solución.

1.3.2 Herramientas CASE

La Ingeniería de Sistemas Asistida por Computadoras (CASE por sus siglas en inglés) es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo, su objetivo es acelerar el proceso para el que han sido diseñadas, en el caso de CASE para automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas (47).

Visual Paradigm Suite 5.0

Visual Paradigm es una herramienta para UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y documentación. Presenta licencia gratuita y comercial (48). Además de brindar soporte UML, permite utilizar técnicas de ingeniería inversa para llevar de código a diagramas de clases, manteniendo de esta manera la sincronización entre el modelo y el código. Posibilita la generación de código a partir de un modelo o diagrama y cuenta con un generador de informes. También realiza la distribución automática de diagramas, así como la reorganización de las figuras y conectores de los diagramas UML.

1.3.3 Lenguaje de programación y Entorno de desarrollo

Java es un lenguaje desarrollado por Sun Microsystems, está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Es un lenguaje simple, orientado a objetos, distribuido, robusto, seguro, de arquitectura neutra, portátil, de alto desempeño, dinámico y que tiene soporte para hilos múltiples, es libre y de código abierto (49). El tiempo estimado para desarrollar la solución es corto, por lo que no es aconsejable el aprendizaje de otro lenguaje, ya que el equipo tiene experiencia utilizando Java. Para el desarrollo de la solución se utilizará Java en su versión 8.0.

Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código de programación, un compilador, un depurador y un constructor de interfaz gráfica (50).

NetBeans 8.1

Es un entorno de desarrollo integrado hecho principalmente para el lenguaje de programación Java, desarrollado por la compañía Sun Microsystems en el año 2000. Desde su lanzamiento NetBeans IDE es un producto libre y gratuito sin restricciones de uso, de código abierto, escrito completamente en Java usando la plataforma NetBeans. Soporta el desarrollo de todos los tipos de aplicación Java, siendo el IDE oficial de Java 8. Cuenta con un constantemente mejorado Editor de Java, muchas características enriquecidas y una amplia gama de herramientas, plantillas y muestras. Su editor permite re factorizar código fácilmente, también proporciona plantillas de código, consejos de codificación y generadores de código (51).

Capítulo 2. Desarrollo de un componente de software para la generación automática del cuadro de clasificación archivística.

2.1 Modelo general de propuesta de solución

A partir del estudio de los antecedentes y consecuentemente con el objetivo de la investigación, se concluye que es necesario diseñar e implementar un componente informático que construya una estructura de datos que permita la representación del cuadro de clasificación archivística. La solución debe implementar algoritmos de agrupamiento jerárquico para dar solución al problema. Tales algoritmos proporcionan la estructura de datos necesaria para representar mediante un dendrograma el agrupamiento de los archivos. La figura 2.1 presenta un esquema general que aclara el proceso de construcción de un árbol de agrupamiento jerárquico de archivos de acuerdo a las especificaciones que se realizan en la actual investigación. La misma se compone por elementos generales que definen la propuesta a desarrollar y sus posibles componentes.

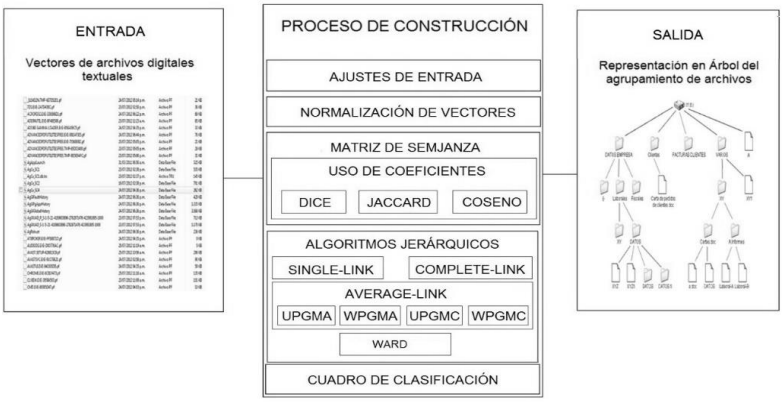

Figura 2.1. Modelo de la propuesta de solución

Como se observa en la figura 2.1 el modelo de propuesta de solución cuenta con tres partes fundamentales: entrada, proceso de construcción y salida. El flujo de entrada consiste en una colección de vectores construidos a partir de archivos digitales textuales que serán analizados por el proceso de construcción. Este, primeramente, realiza un ajuste de la entrada de los vectores para luego aplicar una transformación de los datos mediante la normalización de los vectores, actividad necesaria para que puedan ser procesados. Luego se encuentra la creación de la matriz semejanza. Este proceso cuenta con el cálculo de la similitud entre los vectores mediante tres diferentes coeficientes: Dice, Jaccard y Coseno. Cada uno de estos coeficientes construirá una matriz de semejanza diferente sobre las cuales se aplicarán los diferentes algoritmos. Después aplicar sobre las matrices de semejanza los algoritmos Single-Link, Complete-Link, Ward, y Average-Link (este último a su vez comprende los algoritmos UPGMA, WPGMA, UPGMC y WPGMC) estos retornan una estructura de matrices (52) que puede ser transformadas en un cuadro de clasificación. Finalmente, solo se necesita transformar los datos obtenidos por cada algoritmo para construir el dendrograma que ayudara en la elaboración del cuadro de clasificación archivística.

Como etapa posterior al diseño general del modelo de la propuesta de solución, se pretende mostrar un modelo conceptual de carácter técnico que ilustre los conceptos que dan origen a la solución propuesta y sus relaciones. El modelo seleccionado es el de dominio, artefacto UML que permite la representación general de los conceptos a desarrollar y sus relaciones específicas. Tal modelo, aunque no es contemplado como necesario por la metodología de desarrollo empleada, permite representar de forma más adecuada la situación inicial ante el potencial desarrollo de software y a partir de él, definir las posibles funcionalidades como historias de usuario.

2.1.1 Modelo de dominio

Es una representación visual de clases conceptuales o de objetos reales en un dominio de interés. No constituyen clases de software, son representaciones de los conceptos involucrados para la comprensión de la solución y las asociaciones establecidas describen las relaciones entre los conceptos (53). Por tanto, un modelo de dominio constituye una representación conceptual de las relaciones entre los objetos involucrados en el proceso de construcción de software. La figura 2.2 muestra una representación estática del entorno real del proyecto mediante el modelo de dominio.

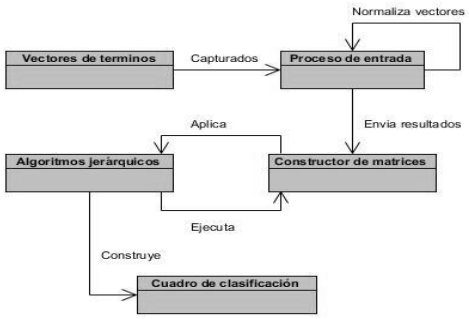

Figura 2.2. Modelo de dominio

En el modelo de dominio mostrado se aprecia que los vectores de términos son capturados mediante el proceso de entrada que necesita normalizar los vectores para que puedan ser analizados. Esto se debe a que los vectores de términos contienen un numero diferente de elementos (cardinalidad) y para el cálculo de semejanza entre vectores estos deben encontrarse normalizados (poseer la misma cardinalidad).

Luego del proceso de entrada donde fueron normalizados los vectores es necesario construir la matriz de semejanza. En este punto, el constructor de matrices obtiene los vectores normalizados por parte del proceso de entrada y encargarse en una primera etapa de crear una matriz con la semejanza de todos los vectores. Acto seguido, se realizan los cálculos de semejanza entre dos vectores mediante la similitud Dice, Jaccard y Coseno, respectivamente, devolviendo en cada caso un valor de similitud que será insertado en matrices diferentes correspondiendo con cada coeficiente. Cada una de las matrices generadas será guardada para formar parte de la estructura de datos jerárquicos basado en matrices. Una vez construida la primera matriz de semejanza se prosigue a aplicar los algoritmos.

Todos los algoritmos a ejecutar parten de una primera matriz de semejanza y logran realizar un agrupamiento jerárquico de los vectores ejecutando en cada iteración el constructor de matrices que genera una nueva matriz, creando una lista de matrices que formará la estructura jerárquica. Este algoritmo es iterativo y tiene como condición de parada la terminación de análisis de los vectores en la matriz de semejanza. Finalmente se obtiene de cada algoritmo que se aplicó a cada matriz obtenida por los diferentes coeficientes de similitud, una estructura de datos basada en matrices que construye el cuadro de clasificación. Luego de establecer los elementos del modelo de propuesta general de solución y representar seguidamente los conceptos específicos del dominio del problema, se prosigue a la etapa de planeación según la metodología seleccionada. Dicha etapa es el punto inicial del desarrollo de la solución.

2.2 Planeación

La planeación es la etapa inicial en todo desarrollo de software bajo la metodología XP. En este punto comienza la interacción con el cliente para identificar los requerimientos del sistema y se identifican las iteraciones y ajustes necesarios a la metodología según las características de la solución (44). En esta sección se describen dos elementos fundamentales, los cuales son: Historia de Usuario y Ejecución del Proyecto. Dichos elementos resumen la etapa inicial planteada por la metodología de desarrollo de software seleccionada (54) (55) (56). En ellos se recogen las características principales del sistema a construir y se concreta el tiempo que se necesitará para implementarlas. Las especificidades de la planeación y definición del tiempo de ejecución del desarrollo en cuestión son mostradas a continuación.

2.2.1 Historias de Usuario

Las historias de usuario son un instrumento propuesto por la metodología seleccionada para el levantamiento de requisitos en el desarrollo de software. Constituyen tarjetas donde el cliente describe brevemente (con el fin de que sean dinámicas y flexibles) las características que el sistema debe poseer, sean requisitos funcionales o no funcionales (57). Cada historia de usuario debe ser lo suficientemente comprensible y no muy extensa en cuanto a la cantidad de requisitos a incluir en ella, con el fin de que se puedan implementar en poco tiempo. No existe un estándar determinado para redactar una historia de usuario, según la metodología seleccionada, el equipo de desarrollo determina que es lo importante que debe quedar escrito y como llevar el control de cada una. A partir de un estudio realizado de los elementos más importantes recomendados por varios autores (58) (59) se concluye en la selección de los siguientes campos para conformar una historia de usuario:

- ✓ Número de historia de usuario: permite que la historia sea rápidamente identificada en los pasos posteriores que se llevarán a cabo en la etapa de planificación. La idea es que posean un número consecutivo, que solo proporciona información respecto al orden en el que fueron redactadas las historias.
- ✓ Usuario: persona involucrada en el desarrollo de la historia de usuario.
- ✓ Fecha: corresponde con la fecha en la cual la historia de usuario es redactada.
- ✓ Título: corresponde con el nombre que se le otorga a la historia de usuario por parte del cliente.

- ✓ Descripción: lugar donde el cliente describe que se debe hacer de forma comprensible y no muy extensa, evitando el uso de terminología y la acumulación de varias funcionalidades en una misma historia de usuario.
- ✓ Observaciones: corresponden a aquellos detalles relevantes que serán resueltos tras la conversación del equipo desarrollador con el cliente.
- ✓ Puntos Estimados: tiempo que se asigna o se supone al desarrollo de la historia de usuario.
- ✓ Puntos Reales: tiempo real que demora el desarrollo de la historia de usuario.

De acuerdo con los principios ágiles del desarrollo de software y las técnicas propuestas por la metodología XP, se describen en esta sección las principales historias de usuario del sistema, el resto se encuentra en el anexo número 1. Tales descripciones tienen el objetivo de ilustrar, los elementos relevantes de las funcionalidades más importantes a desarrollar para dar solución al problema planteado. Las tablas 2.1, 2.2 y 2.3 muestran las historias de usuario Normalizar vectores, Construir matriz inicial y Aplicar algoritmos jerárquicos respectivamente. Los campos que contienen son los definidos anteriormente.

Historia de Usuario					
Numero de Historia:	HU-3	Usuario:	Programador	Fecha:	13/01/2016
Titulo:	Normalizar conjunto de vectores				
Descripción:	El componente debe normalizar el conjunto de vectores. Para aplicar la normalización se recibe dos vectores. El resultado es dos vectores con la misma cantidad de componentes				
Observaciones:	Los vectores de términos contienen un numero diferente de elementos, y para el cálculo de semejanza entre vectores estos deben encontrarse normalizados (tener la misma cardinalidad).				
Puntos Estimados:	1		Puntos Reales:	1	

Tabla 2.1. HU-3. Normalizar conjunto de vectores

Historia de Usuario					
Numero de Historia:	HU-4	Usuario:	Programador	Fecha:	13/01/2016
Titulo:	Construir matriz de semejanza inicial				
Descripción:	El componente permite la conformación de la primera matriz de semejanza. Para la construcción de la matriz se debe recibir una colección de vectores normalizados. El resultado es una matriz de semejanza entre todos los vectores de la colección.				
Observaciones:	La matriz es completada mediante el cálculo de similitud entre los diferentes vectores utilizando los coeficientes Dice, Jaccard, y Coseno.				
Puntos Estimados:	4		Puntos Reales:	4	

Tabla 2.2. HU-4. Construir matriz de semejanza inicial

Historia de Usuario

Numero de Historia:	HU-5	Usuario:	Programador	Fecha:	13/01/2016
Titulo:	Ejecutar algoritmos jerárquicos				
Descripción:	El componente permite la conformación de clústeres y la jerarquización entre ellos. Para la ejecución de los algoritmos debe recibir una matriz de semejanza. El resultado es un estructura de datos basada en matrices que permite crear un cuadro de clasificación.				
Observaciones:	Se utilizan los algoritmos Single-Link, Complete-Link, UPGMA, WPGMA, UPGMC, WPGMC y Ward.				
Puntos Estimados:	6		Puntos Reales:	6	

Tabla 2.3. HU-5. Ejecutar algoritmos jerárquicos

2.2.2 Tiempo de ejecución del proyecto

El tiempo de ejecución del proyecto es una medida que tiene el equipo de desarrollo para completar las historias de usuario en una determinada iteración. Dicha medida se calcula totalizando los puntos estimados de las historias de usuario realizadas en una iteración. En este sentido un punto, equivale a una semana ideal de programación donde se trabaje sin interrupciones (58) (59). La planificación se realiza basándose en el tiempo de implementación de una historia de usuario. El tiempo de ejecución se utiliza para establecer cuántas historias de usuario se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. La tabla 2.4 presenta la estimación de esfuerzo por historia de usuario propuesta para el desarrollo de la aplicación:

Número	Título de Historia de Usuario	Estimación en Puntos
HU-1	Recibir colección de entrada	1
HU-2	Crear diccionario de términos	1
HU-3	Normalizar conjunto de vectores	1
HU-4	Construir matriz de semejanza inicial	4
HU-5	Ejecutar algoritmos jerárquicos	6
HU-6	Realizar poda de dendrograma	2
HU-7	Construir cuadro de clasificación	4
HU-8	Exportar cuadro de clasificación	1

Tabla 2.4. Estimación por historia de usuario

2.2.3 Iteraciones

En la metodología XP, el desarrollo del software se divide en etapas para facilitar su realización, dichas etapas se nombran iteraciones. Para cada iteración se define un número determinado de historias de usuario a implementar y al final de cada iteración se obtiene como resultado la entrega de las funcionalidades correspondientes (44). Esto ocurre conjuntamente con la aceptación por parte del cliente de los requisitos solucionados. De acuerdo con lo planteado por la metodología y en consecuencia con el proceso de desarrollo se divide el mismo en cuatro iteraciones. El número de iteraciones, así como la

cantidad de historias de usuario de cada una, constituyen el comienzo y fin de cada fase del proceso de construcción del árbol. La tabla 2.5 muestra el plan de iteraciones propuesto por el equipo de desarrollo.

Iteración	Título de Historia de Usuario	Duración Total
1	Recibir colección de entrada, Normalizar conjunto de vectores, Crear diccionario de términos	3
2	Construir matriz de semejanza inicial	4
3	Ejecutar algoritmos jerárquicos	6
4	Realizar poda de dendrograma, Construir cuadro de clasificación, Exportar cuadro de clasificación	7
Total		20

Tabla 2.5. Plan estimado de duración de las iteraciones

El plan de iteraciones mostrado en la tabla 2.5 se confecciona teniendo en cuenta la complejidad de las funcionalidades involucradas, las pruebas y la necesidad de producir rápidamente versiones del sistema que sean operativas. Para la investigación se define una versión operativa del sistema como una unidad atómica del componente desarrollado que se comporta de forma independiente y delimita el final de una iteración. Objetivamente cada iteración constituye una parte fundamental del sistema a desarrollar, posee una entrada y salida determinada consecuente con la iteración contigua. Como se observa en la figura 2.1 el modelo de la propuesta de solución cuenta con cinco etapas principales que culminan en una versión operativa del mismo. Conjuntamente las pruebas planificadas durante el proceso de desarrollo necesitan de partes individuales completamente funcionales para verificar cada una de las etapas fundamentales. Además, el tiempo estimado para cada iteración no supera el máximo establecido por la metodología XP de 3 meses (44) y la complejidad de las tareas permite agruparlas en el mismo orden del flujo de datos.

2.2.4 Plan de entrega

El plan de entrega consiste en establecer conjuntamente el equipo de desarrollo y el cliente la duración y fecha de entrega de cada iteración hasta lograr el producto final. En este plan se detalla la fecha fin de cada iteración, mostrando una versión desarrollada del producto en ese momento, hasta lograr el producto final en la fecha establecida (44). A partir del plan de iteraciones y la fecha de inicio del proceso de desarrollo se conforma el plan de entrega coincidiendo cada entrega con el fin de una iteración. La tabla 2.6 muestra el plan de entrega definido por el equipo de desarrollo.

Iteraciones	Fecha de entrega
Iteración 1	1 de febrero del 2016
Iteración 2	29 de febrero del 2016
Iteración 3	4 de abril del 2016
Iteración 4	23 de mayo del 2016

Tabla 2.6. Plan de entrega de las iteraciones

El plan de entrega mostrado en la tabla 2.6 se confecciona teniendo en cuenta la necesidad de realizar entregas operativas del sistema, cada entrega es una versión incremental del mismo. Para la primera fecha el sistema debe ser capaz de recibir el conjunto de vectores, crear el diccionario de términos y lograr normalizarlos. Una vez culminado este proceso, para la segunda entrega el sistema debe ser capaz de construir la matriz de semejanza inicial utilizando los coeficientes de similitud Dice, Jaccard, y Coseno entre dos vectores. Para la tercera entrega se deben poder ejecutar los grupos de algoritmos Single-Link, Complete-Link, UPGMA, WPGMA, UPGMC, WPGMC y Ward. En la entrega final el producto debe ser completamente funcional, permitiendo construir la estructura de matrices de la cual se podrá generar el cuadro de clasificación, para ser exportado a un archivo bin.

2.3 Diseño de la propuesta de solución

El diseño se encuentra en el núcleo técnico de la ingeniería de software. Una vez que se analizan y especifican los requisitos del software, esta es la primera de las tres actividades técnicas que se requieren para construir y verificar el software (60). En esta sección se realiza la descripción de la arquitectura de software y los patrones de diseño aplicados a la propuesta de solución. Se presenta el diagrama de clases persistentes del componente a desarrollar y se expone brevemente en qué consisten las tarjetas clase-responsabilidad-creador, así como la descripción mediante el uso de dichas tarjetas de las principales clases del sistema. Estos elementos muestran una vista más técnica y lógica del software a desarrollar, describen la arquitectura utilizada basada en las necesidades de la solución, así como las clases y sus relaciones. Además, permiten la definición de una visión más específica sobre las funcionalidades a desarrollar y son el punto de partida para la posterior codificación.

2.3.1 Arquitectura de software

De acuerdo al Instituto de Ingeniería de Software (SEI por sus siglas en inglés), la arquitectura de software se refiere a las estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos. Los elementos son entidades que existen en tiempo de ejecución (objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios). Las relaciones entre elementos dependen de sus propiedades visibles, quedando ocultos los detalles de implementación (61). Por lo cual implementar una arquitectura de software permite representar el diseño a alto nivel del sistema a desarrollar y sirve como guía para el proceso de desarrollo de software. Además, permite dividir el sistema en estructuras arquitectónicas relacionadas, posibilitando la reutilización y el acoplamiento entre las mismas.

Arquitectura N capas

El estilo arquitectónico en n capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan. La programación por capas tiene como objetivo principal separar la lógica de negocios de la de diseño, esto se aprecia al separar la capa de datos de la de negocio y la de negocio de la de presentación (46).

El diseño empleado en la presente investigación es el diseño en tres capas. Aunque la solución implementada no define las capas clásicas de presentación, control y negocio, si cuenta con una capa dedicada a la captura de datos para obtener la colección de vectores a procesar y normalizarlas para su procesamiento. Las otras dos capas se encargan de la ejecución del algoritmo y de los cálculos que son necesarios para realizar la matriz de semejanza y transformar la estructura de datos que es resultado de los algoritmos a el cuadro de clasificación. En la figura 2.3 se ilustra la arquitectura de la propuesta de solución.

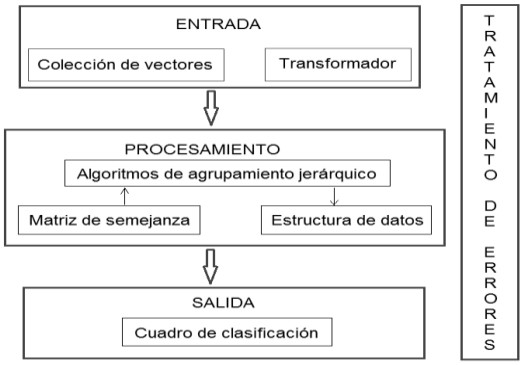

Figura 2.3. Arquitectura de la propuesta de solución

Como se muestra en la figura 2.3, el diseño de la arquitectura está divido en tres capas: entrada, procesamiento y salida. En la primera capa se captura la colección de vectores y términos, para posteriormente ser normalizados. La normalización de vectores los que busca es convertir los vectores que contienen un numero diferente de elementos, y así poder realizar cálculos de semejanza sobre estos vectores. La capa de procesamiento es la responsable de las operaciones de la aplicación, en ella se realiza la construcción de la matriz de semejanza a partir del cálculo de las medidas de similitud Dice, Jaccard y Coseno entre los diferentes vectores, con la matriz creada se puede aplicar los algoritmos jerárquicos que generaran un listado de matrices de semejanza que es la parte fundamental de la estructura de datos. Dicha capa es la encargada de devolver una estructura de datos basada en matrices que puede ser convertida en el cuadro de clasificación como resultado final del proceso. El diseño presentado explota las ventajas del uso de los patrones de asignación de responsabilidades. El bajo acoplamiento se evidencia al mantener la independencia entre las capas de la aplicación, esto permite que los cambios que ocurran en una capa no repercutan en el resto. Cada componente de las capas se encarga de mantener las responsabilidades asignadas lógicamente evidenciando la alta cohesión.

2.3.2 Patrones de diseño

Los patrones de diseño son soluciones para problemas típicos y recurrentes que enfrentan los programadores al desarrollar una aplicación. Son soluciones probadas y documentadas que explican cómo resolver un determinado problema bajo determinadas circunstancias. Por lo que constituyen una guía para el rápido desarrollo de software (62). Por tanto, un patrón de diseño es una solución demostrada, eficiente y reutilizable en diferentes contextos y escenarios de desarrollo de software.

Patrones de Asignación de Responsabilidades (GRASP)

Indican que la responsabilidad de la creación de un objeto o la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para crearlo (46). De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). Los patrones analizados en esta sección son: Experto, Creador, Alta Cohesión y Bajo Acoplamiento.

Experto

El experto en información establece el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo (46). Por tanto, la utilización de este patrón conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida. La figura 2.5 muestra el uso de este patrón.

```
@Override
public void run() {
    for (int i = 0; i < v2.getListTerm().size(); i++) {
        boolean enc = false;
        for (int j = 0; j < v1.getListTerm().size(); j++) {
            if (v2.getListTerm().get(i).getStem().equals(v1.getListTerm().get(j).getStem())) {
                enc = true;
                break;
            }
        }
        if (!enc) {
            v1.add(v2.getListTerm().get(i).getStem());
        }
    }
}
```

Figura 2.5. Implementación del patrón Experto

Como se observa en la figura 2.5 la clase NormalizarVectores es la clase transformadora que permitirá normalizar los datos de entrada por la clase Matriz a datos de salida en MatrizSemejanza. Esta cuenta con toda la información necesaria para la implementación del método run() encargada de la normalización de los vectores de entrada.

Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental es encontrar un creador que se debe conectar con el objeto producido en cualquier evento (46). Por tanto, este patrón pretende lograr un bajo acoplamiento (patrón que se describe más adelante) lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. La figura 2.6 muestra el uso de este patrón.

```
@Override
public void run() {
    for (int i = 0; i < vectores.size(); i++) {
        for (int j = 0; j < i; j++) {
            if (i != j) {
                Semejanzas s = new Semejanzas(vectores.get(i), vectores.get(j));
                switch (coeficiente) {
                    case "Dice":
                        semejanzas[i][j] = s.CoeficienteDice();
                        break;
                    case "Jaccard":
                        semejanzas[i][j] = s.CoeficienteJaccard();
                        break;
                    case "Coseno":
                        semejanzas[i][j] = s.CoeficienteCoseno();
                        break;
                }
            }
        }
        analizados = i+1;
    }
    CompletarMatriz();
}
```

Figura 2.6. Implementación del patrón Creador

Como se observa en la figura 2.6 la clase MatrizSemejanza es la encargada de la creación de los objetos tipo Semejanza mediante una asociación por agregación con la clase Semejanza (figura 2.4).

Alta Cohesión

La alta cohesión indica que los datos y responsabilidades de una entidad están fuertemente ligados a la misma en un sentido lógico. La información que maneja una entidad de software tiene que estar conectada lógicamente con esta, no deben existir entidades con atributos que describan comportamientos que en realidad no le corresponden (46). Este patrón permite la obtención de clases fácilmente mantenibles y reutilizables durante el proceso de desarrollo de software. La figura 2.7 muestra el uso del mismo.

```
public Cluster UnificarClusters() {
    Cluster c = new Cluster();
    for (int i = 0; i < matrices.get(pos).getClusters().get(iSim).getKeys().size(); i++) {
        c.add(matrices.get(pos).getClusters().get(iSim).getKeys().get(i));
    }
    for (int i = 0; i < matrices.get(pos).getClusters().get(jSim).getKeys().size(); i++) {
        c.add(matrices.get(pos).getClusters().get(jSim).getKeys().get(i));
    }
    return c;
}
```

Figura 2.7. Implementación del patrón Alta Cohesión

En la figura 2.7 se muestra la clase Algoritmos encargada de la ejecución de los diferentes algoritmos de agrupamiento jerárquico, en donde en cada iteración necesita unificar los clústeres seleccionados creando un nuevo clúster al que se le agregan los objetos de los anteriores clústeres usando el método add(). La responsabilidad de implementar este método le corresponde a la clase Cluster teniendo en cuenta lo que plantea el patrón Experto, manifestándose además la alta cohesión entre las clases ya que la entidad Algoritmos no lo implementa, solo hace uso del método.

Bajo Acoplamiento

El bajo acoplamiento sostiene la idea de mantener las entidades y clases lo menos ligadas entre sí posible, de tal forma que, en caso de producirse modificaciones en alguna de ellas, la repercusión sea la mínima posible en el resto de las entidades o clases (46). Por tanto, este patrón potencia la reutilización, y disminuye la dependencia entre clases. Soporta un diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad. El ejemplo analizado en la figura 2.7 soporta tanto la alta cohesión como el bajo acoplamiento ya que la responsabilidad de añadir el objeto al clúster recae sobre la clase Cluster por lo que una modificación en este proceso no repercute en el resto de las entidades.

Patrones de Concurrencia

Los patrones de esta categoría permiten coordinar las operaciones concurrentes. Estos patrones se dirigen principalmente a dos tipos diferentes de problemas:

- ✓ Recursos compartidos: Cuando las operaciones concurrentes acceden a los mismos datos u otros tipos de recursos compartidos, puede darse la posibilidad de que las operaciones interfirieran unas con otras si acceden a los recursos al mismo tiempo. Para garantizar que cada operación se ejecuta correctamente, la operación debe ser protegida para acceder a los recursos compartidos en solitario (63).
- ✓ Secuencia de operaciones: Si las operaciones son protegidas para acceder a un recurso compartido una cada vez, entonces podría ser necesario garantizar que ellas acceden a los recursos compartidos en un orden particular. Por ejemplo, un objeto nunca será borrado de una estructura de datos antes de que este sea añadido a la estructura de datos (63).

Por tal motivo aplicar patrones de concurrencia dentro de la programación permite que el sistema gestione de manera eficiente los recursos de hardware disponibles, microprocesador, memoria y disco duro. La utilización de cada uno de ellos corresponde con el límite físico de sus capacidades. Permite que procesadores multinúcleos ejecuten de forma paralela una secuencia de tareas determinadas e influye considerablemente en el tiempo de ejecución y respuesta de los sistemas informáticos.

Single Threaded Execution

Algunos métodos acceden a datos u otros recursos compartidos de tal forma, que producen resultados incorrectos si hay llamadas concurrentes y acceden a los datos o recursos compartidos al mismo tiempo. Este patrón soluciona este problema impidiendo las llamadas concurrentes a un método que provocan ejecuciones concurrentes del mismo (63). Esto permite que el acceso a los recursos compartidos se realice de forma organizada, lo que evita la pérdida de datos y posibles errores en tiempo de ejecución.

Scheduler

Controla el orden en el cual los hilos son organizados para ejecutar el código. Potencia el empleo de un hilo único, utilizando un objeto que sigue explícitamente en la sucesión de hilos que esperan. El patrón Scheduler provee un mecanismo para implementar una política de organización. Esto es independiente de

cada una de las políticas de organización específicas (63). El uso de tales patrones será detallado en el subepígrafe 2.4.1.

2.3.4 Tarjetas Clase-Responsabilidad-Creador

La utilización de tarjetas clase-responsabilidad-creador (CRC) es una técnica de diseño propuesta por la metodología XP. Consiste en hacer, mediante determinadas tarjetas, un inventario de las clases que se necesitan para implementar el sistema y la forma en que interactúan (44). De este modo se pretende facilitar el análisis y discusión de las mismas por parte del equipo de proyecto con el objetivo que el diseño sea lo más simple posible verificando las especificaciones del sistema. Las tarjetas CRC contienen una serie de campos que deben ser llenados por el equipo de desarrollo, ellos son (59):

- ✓ Nombre de la Clase: especifica el nombre de la clase en el sistema.
- ✓ Responsabilidades de la Clase: describen a alto nivel el propósito de la existencia de la clase. Una clase no debe tener más de tres o cuatro responsabilidades.
- ✓ Colaboradores de la Clase: describe las clases con las que va a colaborar para ejecutar las responsabilidades.

Las tablas 2.7, 2.8 y 2.9 describen mediante tarjetas CRC las clases más importantes que intervienen en el desarrollo de la aplicación:

Nombre de la Clase: Algoritmos
Responsabilidades de la Clase: La clase Algoritmos es la responsable de la ejecución de los algoritmos de clustering jerárquico Single-Link, Complete-Link, UPGMA, WPGMA, UPGMC, WPGMC y Ward, que se encargará de todo el proceso de construcción de la dendrograma que permitirá crear el cuadro de clasificación. Utiliza una lista de MatrizNumerica para cada iteración de los algoritmos.
Colaboradores de Clase
MatrizNum, Poda, Prinipal, CuadroClasificacion

Tabla 2.7. Tarjeta CRC. Clase Algoritmos

Nombre de la Clase: MatrizSemejanza
Responsabilidades de la Clase: La clase MatrizSemejanza es la encargada calcular dada una colección de vectores de entrada a una matriz de semejanza. Invoca a la clase encargada de determinar la semejanza entre dos vectores mediante los coeficientes Dice, Jaccard y Coseno.
Colaboradores de Clase
NormalizarVectores, MatrizNumerica, Principal, Semejanza

Tabla 2.8. Tarjeta CRC. Clase MatrizSemejanza

Nombre de la Clase: Poda
Responsabilidades de la Clase:

La clase Poda es la encargada podar el dendrograma que se va generando por la clase Algoritmos en cada iteración, aplicando un determinado criterio de poda para sintetizar los clústeres.
Colaboradores de Clase
Algoritmos, Diccionario

Tabla 2.9. Tarjeta CRC. Clase Poda

2.4 Implementación de la Propuesta de Solución

En esta sección se describe implementación de la propuesta de solución. Para ello se detalla los algoritmos implementados para la representación del cuadro de clasificación, así como las estructuras de datos utilizados durante el proceso de construcción. Tales algoritmos tienen como base colocar cada vector perteneciente a un documento en un grupo propio e ir uniendo los dos grupos más similares en uno solo, pero utiliza una mejora potencial en el uso de matrices de semejanza para representar cada iteración y evitar sobrecarga de cálculos para cada vector, mejorando el uso en espacio de memoria y agilizando la rapidez del algoritmo. La tabla 2.10 muestra el pseudocódigo del procedimiento para la construcción de la matriz de semejanza mediante el cálculo de semejanza coseno entre cada vector. Se asume que comienza su ejecución a partir de la matriz de vectores importada. Dicha matriz constituye la entrada al proceso de transformación.

Procedimiento de construcción de matrices	
1	Inicio
2	Vectores ← **Entrada** (Matriz de vectores)
3	Semejanzas ← {Matriz de semejanzas, inicializar}
4	Para cada V1 de Vectores hacer
5	Para cada V2 de Vectores hacer
6	Semejanzas ← **CalcularSemejanza(V1, V2)**
7	Fin Para
8	Fin Para
9	Fin

Tabla 2.10. Pseudocódigo. Procedimiento de construcción de matrices

En las líneas 2 y 3 del pseudocódigo mostrado se inicializan las variables y estructuras de datos necesarias para el procedimiento de transformación. Inicialmente Vectores toma todos los vectores pertenecientes a la matriz de vectores que es importada al programa (9), para a partir de aquí inicializar una matriz de semejanza numérica correspondiente a la cantidad de vectores que fueron importados. En la línea 4 y 5 se define que cada vector V1 de la lista de Vectores debe ser procesado con cada vector V2 de la igual lista de Vectores. Para finalmente en la línea 6 llenar la matriz de semejanza con el resultado de método **CalcularSemejanza(V1, V2)**.

CalcularSemejanza(V1, V2) → Es el grupo de funciones matemáticas correspondientes a la similitud Dice, Jaccard y Coseno aplicadas diferentemente para cada proceso de construcción de matrices donde toma los vectores V1 y V2 y luego de aplicárseles una ecuación matemática este retorna un valor numérico que representa cuan semejante es un vector con otro.

La tabla 2.11 muestra el pseudocódigo del proceso de normalización de vectores necesario para el cálculo de semejanza. Se asume que comienza su ejecución a partir de dos vectores de términos a los cuales se le realizara el proceso de normalizado. Dichos vectores constituyen la entrada del proceso de normalización.

Proceso de normalización	
1	Inicio
2	Vector1 ← **Entrada** (Primer vector)
3	Vector2 ← **Entrada** (Segundo vector)
4	Para cada Termino1 de Vector1 hacer
5	Existe ← **Falso**
6	Para cada Termino2 de Vector2 hacer
7	Si Termino1 = Termino2 hacer
8	Existe ← **Verdadero**
9	Fin Si
10	Fin Para
11	Si **No** Existe hacer
12	Vector2 ← Vector2 U Termino1 (Añadir el término del vector 1 al vector 2)
13	Fin Si
14	Fin Para
15	**NormalizarVector2(V2,V1)**
16	Fin

Tabla 2.11. Pseudocódigo. Proceso de normalización

En las líneas 2 y 3 del pseudocódigo mostrado se inicializan las variables necesarias para el procedimiento de normalización: Vector1 y Vector2 que capturan los dos vectores que serán normalizados. En las líneas 4 y 5 se define que para cada Termino1 de Vector1 se creará una compuerta lógica *Existe* que permite saber si el término que se analiza se encuentra en ambos vectores. Luego las líneas 6, 7 y 8 definen que para cada Termino2 de Vector2 será comparada con el Termino1 que se está analizando y así determinar si el término se encuentra en ambos vectores. Para finalmente en las líneas 11 y 12 si la compuerta lógica *Existe* es negativa entonces colocar el Termino1 en el Vector2 sin valor de peso o frecuencia de ocurrencia. En la línea 15 se llama nuevamente al proceso de normalización que repite el proceso invirtiendo los vectores, obteniendo la simetría para ambos vectores. La tabla 2.12 muestra el pseudocódigo para la ejecución en general de algoritmos de agrupamiento jerárquicos desarrollados para la construcción del dendodrama. Se asume que comienza su ejecución a partir de una matriz de semejanza numérica que

contengan los valores de semejanza entre cada vector y una lista correspondiente al identificador de cada vector. Dicha matriz constituye la entrada al algoritmo propuesto.

Procedimiento general de agrupamiento jerárquicos	
1	Inicio
2	Matrices ← {Lista de matrices, inicializar}
3	Matrices ← Matrices U **Entrada** (Añadir la matriz de semejanza)
4	Pos ← {matriz que se está analizando según la iteración del algoritmo, inicia en 0}
5	Mientras |Matrices| > 1 hacer
6	M ← **AlgoritmoJerarquico()**
7	Matrices ← Matrices U M
8	Pos ← Pos +1
9	Fin Mientras
10	Fin

Tabla 2.11. Pseudocódigo. Algoritmos de agrupamiento jerárquicos

Entre las líneas 2 y 4 del pseudocódigo mostrado se inicializan las variables y estructuras de datos necesarias para la corrida del algoritmo. Inicialmente Matrices es inicializado para poder recibir la primera matriz que recibirá para el inicio del algoritmo, mientras que Pos define en que iteración del algoritmo se encuentra y por tanto sobre qué matriz de la lista de matrices se estará trabajando. A partir de la línea 5 comienza la corrida del algoritmo, donde en cada iteración se toma la matriz con la que se está trabajando y se ejecuta uno de los algoritmos de agrupamiento mediante la función AlgortimoJerarquico(), el cual devolverá la próxima matriz a analizar.

AlgoritmoJerarquico() → Este método ejecuta los algoritmos Single-Link, Complete-Link, UPGMA, WPGMA, UPGMC, WPGMC y Ward de manera independientes los cuales retornan en cada ejecución una nueva matriz correspondiente con la próxima iteración.

Una vez ejecutado uno de los algoritmos, en la línea 7 ya se tiene la nueva matriz que formara parte de la lista de matrices correspondiente a la estructura de datos del algoritmo. Finalmente, en la línea 8 se actualiza la iteración del algoritmo, actualizando a su vez la matriz con la que se trabajara en la siguiente iteración. El algoritmo representado supone una mejora en cuanto al uso de los recursos de hardware con respecto al diseño teórico de los algoritmos originales los que usaban modelos vectoriales en el cálculo de la semejanza desperdiciando espacio en datos innecesarios. Esto se debe a la implementación de matrices numéricas durante todo el proceso de construcción del cuadro de clasificación, que indican un menor uso de espacio en memoria por parte del algoritmo.

2.4.1 Programación Concurrente

La programación concurrente es la simultaneidad en la ejecución de múltiples tareas interactivas. Estas tareas pueden ser un conjunto de procesos o hilos de ejecución creados por un único programa. Los programas de software concurrentes tienen más de una línea lógica de ejecución y permiten que varias

partes del código se ejecuten simultáneamente (64). Esto permite que arquitecturas multiprocesadores ejecuten al menos tantas tareas como cantidad de procesadores posean.

La concepción del software como programa concurrente puede conducir a una reducción del tiempo de ejecución. Cuando se trata de un entorno monoprocesador, permite solapar los tiempos de entrada/salida o de acceso al disco de unos procesos con los tiempos de ejecución de procesador de otros procesos. Cuando el entorno es multiprocesador, la ejecución de los procesos es realmente simultánea en el tiempo, y esto reduce el tiempo de ejecución del programa. También permite mayor flexibilidad de planificación, y con ella, los procesos con plazos límites de ejecución más urgentes pueden ser ejecutados antes de otros de menor prioridad (64).

Implementación de la concurrencia

Los algoritmos descritos anteriormente en la sección 2.4 son analizados independientemente con respecto al factor tiempo de ejecución de cada uno. La figura 2.8 muestra el tiempo de ejecución media de los coeficientes de similitud, la figura 2.9 el tiempo de ejecución del proceso de normalización y la figura 2.10 el tiempo de ejecución media de los algoritmos de agrupamiento jerárquico, en cada caso se realizan dos corridas de los algoritmos. La colección de entrada varía desde 10 hasta 100 archivos para cada par de corridas, y cada una es realizada de forma secuencial y concurrente.

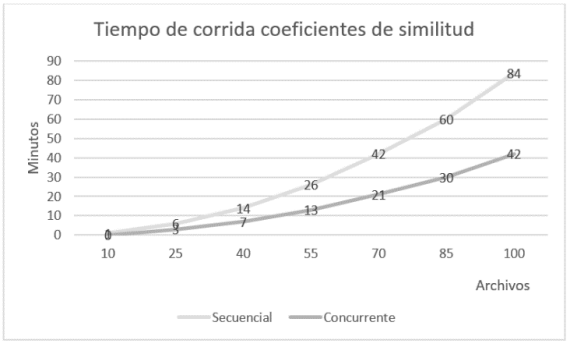

Figura 2.8. Tiempo de corrida media de los coeficientes de similitud

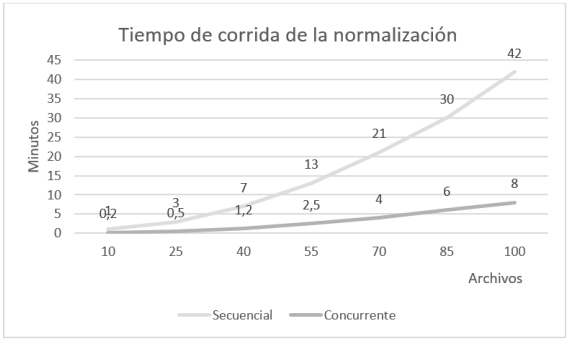

Figura 2.9. Tiempo de corrida del proceso de normalización

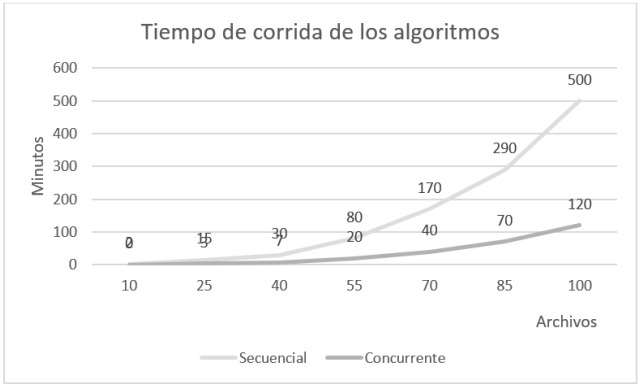

Figura 2.10. Tiempo de corrida media de los algoritmos

Como se observa en las gráficas, el tiempo de corrida secuencial es mayor en todos los casos que la corrida concurrente. Esto se debe a que a medida que aumenta el tamaño de la colección aumenta el número de operaciones secuenciales que debe realizar el componente. La concurrencia se fija en la implementación de los métodos que pueden correr en paralelo, en caso de los coeficientes de similitud ejecutando los métodos getDistorcion() y getSimilitud() en diferentes hilos. En la ejecución de la normalización se procede a ejecutar la normalización de cada vector en un hilo por separado. Mientas que en la ejecución de los algoritmos las funciones de poda son paralelizadas ejecutando concurrentemente las diferentes funciones para cada clúster que se desee podar. Esto se realiza de manera que una arquitectura multiprocesador permite ejecutar un máximo de hilos simultáneamente por vez. Aun así, la evidencia demuestra que a vectores más extensos se puede presumir tiempos de corrida elevados. Para el control de la concurrencia se emplean los patrones: Single Threaded Execution y Scheduler. Las figuras 2.11 y 2.12 muestran su implementación respectivamente.

```
public class Algoritmos extends Thread implements Runnable {

    private List<MatrizNumerica> matrices;
    private MatrizSemejanza matrizIni;
    private Diccionario diccionario;
    private String algoritmo;
    private double maxSim;
    private int iSim;
    private int jSim;
    private int pos;
    private long ini;
    private long fin;

    public Algoritmos(MatrizSemejanza matriz, String algoritmo, Diccionario diccionario) {
        matrizIni = matriz;
        matrices = Collections.synchronizedList(new ArrayList<MatrizNumerica>());
        matrices.add(new MatrizNumerica(matriz));
        this.algoritmo = algoritmo;
        this.diccionario = diccionario;
    }
```

Figura 2.11. Implementación del patrón Single Threaded Execution

La clase Algoritmos que se muestra en la figura 2.11 es la encargada de almacenar las listas con todas las matrices numéricas que van generando. El constructor de la clase es el encargado de establecer la sincronización en el acceso a la lista. El método synchronizedList() restringe el acceso a los objetos almacenados en tiempo de ejecución cuando más de un hilo intenta realizar modificaciones sobre ellos, para esto el primer hilo en apoderarse del objeto establece un bloqueo hasta que termina sus operaciones.

Sin embargo, la lectura puede realizarse de forma simultánea sin afectar la integridad o pérdida de objetos en la colección.

```
@Override
public void run() {
    boolean e = false;
    while (!e) {
        if (!nv.isAlive()) {
            ini = System.currentTimeMillis();
            ms1 = new MatrizSemejanza(nv.getVectores(), "Dice");
            ms2 = new MatrizSemejanza(nv.getVectores(), "Jaccard");
            ms3 = new MatrizSemejanza(nv.getVectores(), "Coseno");
            ActualizarSemejanzaBar sb1 = new ActualizarSemejanzaBar(ms1, diceBar);
            ActualizarSemejanzaBar sb2 = new ActualizarSemejanzaBar(ms2, jaccardBar);
            ActualizarSemejanzaBar sb3 = new ActualizarSemejanzaBar(ms3, cosenoBar);
            EjecutarAlgoritmos ea = new EjecutarAlgoritmos(ms1, ms2, ms3, d);
            ms1.start();
            ms2.start();
            ms3.start();
            sb1.start();
            sb2.start();
            sb3.start();
            ea.start();
            e = true;
        }
        try {
            sleep(1000);
        } catch (InterruptedException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

Figura 2.12. Implementación del patrón Scheduler

La clase Principal, de la cual se muestra un fragmento del código en la figura 2.12, es el hilo principal del programa. Esta controla el orden en el cual los hilos hijos son organizados para ejecutar el código del hilo único, utilizando un objeto que sigue explícitamente en la sucesión de hilos que esperan. El método start() inicia la ejecución del hilo que lo invoca y el método join() le indica al hilo padre que debe esperar a que termine el hilo hijo para continuar su ejecución.

2.4.2 Estructuras para la representación del cuadro de clasificación

La matriz de semejanza es una estructura de datos en forma de matriz vector–vector que contiene el vector de la colección de referencia y la semejanza existente para el resto de los vectores. En la intersección entre dos vectores se almacena un valor numérico de similitud de estos vectores; tal valor representa cuán semejante o diferente es un vector con otro, dado que un vector representa un documento, por lo que tal valor representa cuan semejante o diferente es un documento con otro. Para calcular la semejanza entre los vectores que representan los documentos se utilizan los coeficientes de similitud del Dice, Jaccard y Coseno (27). A partir de la matriz de semejanza es necesario el uso de otra estructura homóloga a esta, las matrices numéricas, que son utilizadas ampliamente en la computación, por su facilidad y para manipular información. En este contexto, son una buena forma para representar las similitudes entre los vectores sin necesidad de cargar los términos de ellos y solo utilizar la información útil para los algoritmos (65). Este tipo de matrices son un arreglo bidimensional de números con una lista de identificadores que determinan en la fila sobre que vector o clúster se está trabajando.

Para mejorar la rapidez de cálculo y el consumo de espacio en memoria se utilizan matrices de tipo triangular inferior. Una matriz se dice que es triangular si cuyos elementos por encima o por debajo de su diagonal principal son cero (65). La matriz numérica original se puede escalonar ya que es una matriz cuadrada, la cual tiene la característica de ser igual a su traspuesta, por lo que no es necesario el uso de

la matriz completa para conocer el valor de semejanza entre un determinado clúster o vector con otro. Al aplicar los algoritmos de agrupamiento jerárquicos las matrices numéricas van a ir reduciéndose uniendo los clústeres y los vectores más semejantes, hasta obtener una matriz de un solo elemento. El grupo de matrices generadas van a guardarse en una lista de matrices donde cada matriz tiene una lista de identificadores que representa en cada iteración que clústeres o vectores fueron unificados. Esta lista de vectores se transforma a un dendrograma que muestra la jerarquía de los elementos analizados por los algoritmos.

Un dendrograma es un tipo de representación gráfica o diagrama de datos en forma de árbol que organiza los datos en subcategorías que se van dividiendo en otros hasta llegar al nivel de detalle deseado. Este tipo de representación permite apreciar claramente las relaciones de agrupación entre los objetos e incluso entre grupos de ellos, aunque no las relaciones de similaridad o cercanía entre categorías. Observando las sucesivas subdivisiones se puede observar los criterios de agrupación de los mismos y la similitud entre los datos según las relaciones establecidas (23). En este caso se puede definir el dendrograma como la ilustración de las agrupaciones derivadas de la aplicación de un algoritmo de clustering jerárquico. La transformación de la lista de identificadores a un dendrograma se realiza recorriendo la lista inversamente tomando el primer identificador como cabeza. Posteriormente cada identificador de las restantes lista es tomado como hijo comparando si el grupo de elementos del hijo se encuentra en la cabeza realizando una estructura jerárquica. Mediante la figura 2.8 se puede analizar cómo se realiza este proceso mediante un ejemplo:

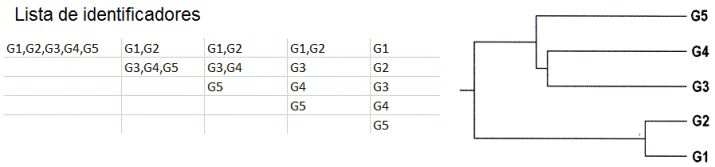

Figura 2.8. Proceso de creación del dendrograma

2.4.3 Resultados de la propuesta de solución

El actual componente de software permite la generación automática del cuadro de clasificación mediante el uso de siete algoritmos de agrupamiento jerárquicos que son evaluados distintamente con tres medidas de similitud, por lo que en este momento el software genera un total de 21 cuadros de clasificación. Cada cuadro de clasificación tiene marcadas características por las que pueden ser analizadas como el nivel del cuadro que corresponde con el nivel del dendograma y la cantidad de subfondos generados que corresponde con el número de clúster del algoritmo que lo construyó. Otro punto importante a destacar es el tiempo que demora la creación de estos cuadros de clasificación que sería el tiempo agregado del cálculo de la similitud con el respectivo algoritmo de agrupamiento jerárquico. Para comprobar el comportamiento del componente de software se toma una muestra de 100 archivos aleatorios que serán el objeto de prueba. Los resultados obtenidos están agrupados por los tres coeficientes de similitud analizados y destacando las características antes descritas.

Para el coeficiente Dice se generaron siete cuadros de clasificación con las siguientes características: el nivel de los cuadros en todos los casos pertenece a un nivel tres, excepto en el cuadro generado por el algoritmo Complete-Link que llega a un nivel 16. El nivel de subfondos de cada cuadro varía entre 68 y 97 siendo el algoritmo Complete-Link y el algoritmo WPGMC los extremos correspondientes. Con respecto al tiempo de creación de los cuadros de clasificación en una primera instancia la demora del cálculo de la matriz de semejanza por el coeficiente Dice fue de 1 hora 16 min y 31 seg. En el desarrollo de los algoritmos los tiempos variaron entre 1 hora 11 min y 11 seg a 2 horas 58 min y 3 seg correspondientes a cada extremo al algoritmo Single-Link como más rápido y al algoritmo Complete-Link como más lento, para hacer un total de tiempo de ejecución de 4 horas 14 min y 34 seg.

En otro caso, para el coeficiente Jaccard los cuadros de clasificación generados presentan las siguientes características: el nivel de los cuadros en todos los casos pertenece a un nivel tres, excepto en el cuadro generado por el algoritmo Complete-Link que llega a un nivel 4. El nivel de subfondos de cada cuadro varía entre 95 y 98 siendo los algoritmos UPGMA, UPGMC y WPGMC lo de menor cantidad de subfondos y el algoritmo Single-Link el de mayor cantidad. Con respecto al tiempo de creación de los cuadros de clasificación en una primera instancia la demora del cálculo de la matriz de semejanza por el coeficiente Jaccard fue de 1 hora 48 min y 28 seg. En el desarrollo de los algoritmos los tiempos variaron entre 59 min y 45 seg a 2 horas 30 min y 24 seg correspondientes a cada extremo al algoritmo Single-Link como más rápido y al algoritmo Complete-Link como más lento, para hacer un total de tiempo de ejecución de un máximo de 4 horas 18 min y 52 seg.

Para el coeficiente Coseno los cuadros de clasificación generados presentan las siguientes características: el nivel de los cuadros en todos los casos pertenece a un nivel tres, excepto en el cuadro generado por el algoritmo Complete-Link que llega a un nivel 9. El nivel de subfondos de cada cuadro varía entre 92 y 97 siendo el algoritmo Complete-Link y el algoritmo Single-Link los extremos correspondientes. Con respecto al tiempo de creación de los cuadros de clasificación en una primera instancia la demora del cálculo de la matriz de semejanza por el coeficiente Coseno fue de 1 hora 16 min y 29 seg. En el desarrollo de los algoritmos los tiempos variaron entre 1 hora 11 min y 24 seg a 2 horas 58 min y 13 seg correspondientes a cada extremo al algoritmo Single-Link como más rápido y al algoritmo Complete-Link como más lento, para hacer un total de tiempo de ejecución de un máximo de 4 horas 14 min y 42 seg.

En modo resumen se puede determinar mediante la observación de los resultados que se mantiene un nivel tres, estándar para los cuadros de clasificación variando únicamente en los cuadros creados por el algoritmo Comlete-Link. En el número de subfondos sobre una observación simple no se puede determinar ningún patrón en los resultados. Con respecto a los tiempos de ejecución se observa una elevada demora en el cálculo del coeficiente Jaccard, pero que a su vez es compensada por la reducción del tiempo de ejecución de los algoritmos, manteniendo un tiempo de ejecución máxima muy similar al resto de los métodos.

Capítulo 3. Comprobación del funcionamiento de la propuesta de solución.

3.1 Comparación del funcionamiento de los algoritmos implementados

En este epígrafe se pretende realizar una comparación de los algoritmos implementados evaluando la calidad de los agrupamientos obtenidos en la construcción del cuadro de clasificación. Se han de llevar a cabo un conjunto de pruebas para demostrar que los resultados obtenidos presentan diferentes grados de precesión que permiten su comparación. Destacar que las pruebas serán aplicadas al primer nivel del cuadro de clasificación por contener los grupos generales de los cuales se deriva la clasificación general de todos los archivos. Para ello se definió un caso de estudio con una colección de documentos previamente etiquetados empleando métricas de evaluación.

3.1.1 Métricas para evaluar la calidad de los agrupamientos

Las métricas de calidad expuestas en el apartado 1.2.5 destinadas a evaluar la calidad de los agrupamientos pueden ser utilizadas para determinar la calidad de un determinado algoritmo sobre otro. La métrica interna Overall Similarity se basa en la similitud de pares de objetos en un clúster, por lo que puede ser usada como métrica de similitud interna. La misma establece que mientras mayor sea el valor obtenido por la función de similaridad mayor es el nivel de cohesión del clúster, determinando que si un algoritmo presenta un elevado número de clústeres con una alta cohesión el algoritmo aumenta su nivel de acierto. Para determinar cuán preciso es un algoritmo se puede usar una media estadística del cálculo del Overall Similarity de cada archivo del clúster (42).

La entropía y F-Measure son otras métricas ampliamente usadas en la evaluación de la calidad de los clústeres. Estas métricas externas usan una colección previamente clasificada a la cual se le aplica el algoritmo de agrupamiento y luego permiten comprobar cuán exacto fue el resultado del algoritmo sobre los objetos ya clasificados. La entropía como métrica puede ser comparable entre varios algoritmos al utilizar como base para la comparación la entropía total calculada para el conjunto de clústeres devuelto por cada algoritmo, definiendo que a mayor entropía existe un alto nivel de precisión del algoritmo. En otro caso, F-Measure igualmente puede ser utilizada para comparar algoritmos (41). Un factor importante al aplicar F-Measure es determinar qué grado de importancia es atribuida para precision y recall, determinando un $\beta = 0.5$ recomendado por (66) que le otorga una mayor importancia a la precisión.

3.1.2 Definición del caso de estudio

Para la ejecución de la solución informática se definió un caso de estudio, el cual debía estar formado por una colección de documentos, donde estos estuvieran previamente clasificados en categorías, con el objetivo de realizar comparaciones entre los resultados. Con el fin de seleccionar esta colección fue necesario revisar las colecciones disponibles que existen en Internet y están reconocidas por los investigadores que pueden ser usadas para realizar este tipo de tareas.

Selección de la colección para realizar los agrupamientos

Existen publicados en Internet varias colecciones de documentos de texto que son recomendadas para realizar pruebas a los algoritmos en las diversas áreas de la minería de texto. Entre las más recomendadas se encuentran:

- The 20 Newsgroups data set (67)
- TDT2 Multilanguage Text Version 4.0 (68)
- Reuters-21578 (69)

The 20 Newsgroups data set, es una colección de alrededor de 20 000 textos etiquetados en 20 categorías. Posee un formato sencillo, pero provienen de listas de discusión, lo cual la hace contar con bastante información no útil y texto de poca longitud. Algunas de las categorías están estrechamente relacionadas unas con otras, mientras otras están muy separadas. La colección TDT2 Multilanguage Text Version 4.0 solo está clasificada en dos tipos de clases: news story y miscellaneous text. En su formato usan etiquetas del Lenguaje de Marcado Estándar Generalizado (SGML) por lo que sería fácil el procesamiento. Esta colección solo presenta dos clasificaciones por lo que el proceso de clasificación jerárquico no es aplicable sobre este tipo de colección.

Por último, la colección de Reuters-21578 está compuesta por 22 ficheros de datos que contienen 21 578 documentos multi-clasificados en 135 tópicos y 6 ficheros que describen las categorías y estos ficheros siguen el formato SGML. Después de analizar los textos y los inconvenientes presentes en cada colección se ha decidido la colección de Reuters-21578 porque cumple la mayoría de los requerimientos predefinidos por (9) para el procesamiento de texto y que son necesarios para evaluar la calidad de los agrupamientos.

Caso de estudio: Grupos de documentos de la Agencia de noticias Reuters

Con el objetivo de conformar la muestra para realizar la evaluación. Se tomaron de la colección de Reuters-21578 un total de 1000 documentos pertenecientes a 10 categorías, con una distribución de 100 documentos por categoría.

Los documentos fueron obtenidos realizando un procesamiento de la colección y tomando solo aquellas noticias que estaban clasificadas en una sola categoría y que estuvieran recomendadas para realizar entrenamientos. Cada uno de los documentos fue almacenado en un fichero texto mediante la decodificación del dataset mediante una aplicación externa que captura la información almacenada en el archivo reut2.sgm. Este componente externo extrae solo los datos necesarios guardados en las etiquetas <Title> que contiene el nombre del archivo, la etiqueta <Topics> que contiene las categorías a la que pertenece el archivo, y la etiqueta <Text> contenedora del contenido del archivo (70). Posteriormente se llevó a cabo el proceso de representación de los documentos al modelo vectorial apoyándonos en la solución informática (9).

3.1.3 Experimento de pruebas

Con el siguiente experimento se pretende llegar a la conclusión cuál algoritmo implementado tiene un mayor nivel de acierto en sus resultados. Para ello se han realizado un conjunto de pruebas

correspondientes a las métricas descritas: Overall Similarity, entropía y F-Measure. Para realizar el experimento se ha tomado una muestra aleatoria simple de 100 documentos para cada corrida del programa de la población de los 1000 documentos del caso de estudio descrito. Se realizan 5 corridas sobre los algoritmos evaluando en cada caso los 7 algoritmos en los 3 coeficientes de similitud definidos en el epígrafe 1.2.2 y 1.2.3 por lo que se obtiene como resultado no solo el algoritmo con el mayor nivel de acierto para las métricas de prueba seleccionada, sino igualmente el comportamiento del algoritmo en el coeficiente de similitud correspondiente. En la tabla 3.1, tabla 3.2 y tabla 3.3 descritas a continuación los valores expuestos son los valores medios obtenidos por las 5 corridas de los diferentes algoritmos.

Algoritmos	*Overall Similarity*	*Entropía*	*F-Measure*
Single-Link	0,848721	0,798692	0,378541
Complete-Link	0,724973	0,875552	0,353766
UPGMA	0,861873	0,865024	0,376732
WPGMA	0,797315	0,764955	0,341311
UPGMC	0,813597	0,807506	0,314745
WPGMC	0,795634	0,816584	0,348516
Ward	0,845215	0,789423	0,313564
Media	0,812475	0,816819	0,346739

Tabla 3.1 Métricas calculadas para los algoritmos sobre Coeficiente Dice

Algoritmos	*Overall Similarity*	*Entropía*	*F-Measure*
Single-Link	0,815763	0,812674	0,321651
Complete-Link	0,726871	0,715692	0,354824
UPGMA	0,893547	0,862542	0,384851
WPGMA	0,836972	0,712582	0,371517
UPGMC	0,848236	0,782349	0,348871
WPGMC	0,787359	0,846284	0,345988
Ward	0,721862	0,824921	0,354989
Media	0,804372	0,79386343	0,35467014

Tabla 3.2 Métricas calculadas para los algoritmos sobre Coeficiente Jaccard

Algoritmos	*Overall Similarity*	*Entropía*	*F-Measure*
Single-Link	0,848312	0,824562	0,315742
Complete-Link	0,812582	0,842364	0,346148
UPGMA	0,914117	0,895214	0,391324
WPGMA	0,821892	0,728758	0,348521
UPGMC	0,848267	0,842391	0,345989
WPGMC	0,835871	0,842347	0,348475
Ward	0,892587	0,841257	0,394244

Media	0,851946	0,83098471	0,35577757

Tabla 3.3 Métricas calculadas para los algoritmos sobre Coeficiente Coseno

En el análisis de las medias obtenidas por la ejecución de los algoritmos a partir de las tres tablas en que son evaluados los distintos coeficientes de similitud, la primera conclusión a la que se puede arribar es la absoluta superioridad en los resultados por parte del Coeficiente Coseno, donde para cada grupo de pruebas se registró el máximo absoluto de medida de similitud. Entonces se puede concluir que para el grupo de prueba analizado el Coeficiente Coseno responde de mejor manera para el procesado de los vectores. En el desarrollo de los algoritmos no fue necesario la construcción de ningún estadígrafo que delimite el algoritmo que respondió con mayor precisión a las pruebas, ya que al igual que en análisis de las medidas de similitud existe un algoritmo que resalta por tener en la mayoría de sus casos el mejor resultado, sin importar que coeficiente fue utilizado. El algoritmo UPGMA respondió mejor sobre el grupo de pruebas que se estaba analizando, destacándose en la prueba Overall Similarity donde los resultados fueron destacados. Esto demostró una cohesión mucho mayor en los clústeres obtenidos que para el resto de los algoritmos.

En resumen, para el caso de estudio sobre el que se aplicaron los algoritmos, la similitud Coseno conjuntamente con el algoritmo UPGMA muestran un desarrollo mucho más satisfactorio en los resultados. Se recomienda de esta forma el uso para la construcción del cuadro de clasificación utilizar Coeficiente Coseno conjuntamente con el algoritmo UPGMA.

3.2 Comprobación del funcionamiento del software desarrollado

Uno de los pilares de XP es el proceso de pruebas (71). Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados al realizar modificaciones y refactorizaciones. La metodología XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñadas por los programadores y pruebas de aceptación destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida por el cliente (71). Para comprender como realizar estas pruebas es necesario definir en qué consisten las pruebas del sistema.

Las pruebas del sistema tienen como objetivo verificar las funcionalidades comprobando que su resultado esté en función de los requisitos del sistema (72); en este caso historias de usuario. Generalmente estas pruebas son desarrolladas por los programadores para verificar que su solución se comporta de la manera esperada, por lo que se podrían acoplar dentro de la definición de pruebas unitarias que propone XP. Sin embargo, las pruebas tienen como objetivo verificar que el sistema cumple los requisitos establecidos por el usuario por lo que también pueden conectar dentro de la categoría de pruebas de aceptación. Ambas pruebas se ejecutarán mediante la definición de una estrategia de prueba que especificará los niveles y métodos de prueba necesarios para la aplicación de las mismas.

3.2.1 Estrategia de Prueba.

La estrategia de prueba describe el enfoque y los objetivos generales de las actividades de prueba. Incluye además los niveles de prueba a ser diseccionados y el tipo de prueba a ser ejecutada (73). Por tanto, la estrategia de prueba permite definir las técnicas y criterios a tener en cuenta para realizar las pruebas al componente desarrollado.

La estrategia de prueba define:

- ✓ Técnicas de pruebas (manual o automática) y herramientas a ser usadas.
- ✓ Qué criterios de éxitos y culminación de la prueba serán usados.
- ✓ Consideraciones especiales afectadas por requerimientos de recursos o que tengan implicaciones en la planificación.

Los niveles de pruebas que se distinguen son:

- ✓ Prueba de unidad.
- ✓ Prueba de aceptación.

Ambos niveles son los definidos por la metodología de desarrollo seleccionada para la realización de las pruebas.

Los métodos de pruebas definidos son:

- ✓ Prueba de caja negra.
- ✓ Prueba de caja blanca.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Por otro lado, la prueba de la caja blanca comprueba los caminos lógicos del software proponiendo casos de prueba que se ejerciten con conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.

La estrategia seguida para la realización de las pruebas al componente desarrollado contempla dos niveles: el nivel de pruebas de unidad y el nivel de prueba de aceptación. En el primer nivel, se realizan pruebas automáticas haciendo uso de tecnologías que permiten la comprobación del código. En el segundo incluye la técnica manual mediante las pruebas de aceptación con el diseño de casos de prueba utilizando el método de caja negra. Las secciones Pruebas unitarias y Pruebas de aceptación describen detalladamente las pruebas realizadas según la estrategia planteada, así como la descripción de la herramienta utilizada.

3.2.2 Pruebas unitarias

Una prueba unitaria es un método que prueba una unidad estructural de código, por lo general son simples y rápidas de codificar. Las pruebas unitarias son una forma para verificar la estructura y el buen funcionamiento de una parte del código, están dirigidas a probar clases aisladamente y relacionadas con el código y la responsabilidad de cada clase, así como sus fragmentos de código más críticos (74). Las mismas son diseñadas y realizadas por los programadores para verificar constantemente las funcionalidades implementadas. Para el desarrollo de la solución estas pruebas se realizan de forma automática utilizando el framework de pruebas JUnit 4.12.

JUnit 4.12

Es un framework para automatizar las pruebas unitarias de aplicaciones Java en la fase de desarrollo. Este permite controlar la ejecución de las clases y de esta forma evaluar si el funcionamiento de cada uno de los métodos se comporta como se espera (75). Para realizar esta evaluación se compara el resultado obtenido durante la ejecución del método con el resultado esperado, partiendo de los datos de entrada necesarios para realizar la prueba. Si ambos resultados son iguales JUnit retorna que el método pasó la prueba correctamente, de lo contrario devuelve un fallo en la misma. Las funcionalidades con que cuenta el sistema fueron comprobadas independientemente una vez implementadas. A continuación, se muestran mediante iteraciones las pruebas realizadas a las principales funcionalidades del sistema, las restantes se encuentran en el anexo número 2.

Pruebas unitarias. Iteración 1

La primera iteración de pruebas unitarias fue realizada sobre las historias de usuario siguientes: HU-1 Recibir colección de entrada, HU-2 Crear diccionario de términos y HU-3 Normalizar conjunto de vectores.

Las tablas 3.4 y 3.5 muestran el resultado de la ejecución de las baterías de prueba realizadas a las clases ReadWriteFile y Diccionario, generadas para la resolución de las historias de usuario HU-1 Recibir colección de entrada y HU-2 Crear diccionario de términos. En ellas se muestran el método u operación a probar, la entrada proporcionada, los resultados esperados y obtenidos y si funciona correctamente.

Pruebas Unitarias				
Método a probar	**Entrada**	**Resultado esperado**	**Resultado obtenido**	**Funciona correctamente**
Read(String url)	El método recibe como entrada una dirección hacia el archivo Matriz.bin.	El método debe retornar una clase Matriz que contenga una colección de vectores.	Clase Matriz creada con colección de vectores.	Si

Tabla 3.4. HU-1 Recibir colección de entrada. Clase ReadFile

Pruebas Unitarias

Método a probar	Entrada	Resultado esperado	Resultado obtenido	Funciona correctamente
FrecuenciaGlobalTF()	El método recibe como entrada los pesos de cada término del diccionario.	El método debe retornar la frecuencia global de todos los términos del diccionario.	Retorna la frecuencia global de todos los términos del diccionario.	Si
Ordenar()	El método recibe como entrada la lista de términos del diccionario y sus pesos.	El método debe ordenar la lista de términos del diccionario según el peso ascendentemente.	Ordena la lista de términos del diccionario según el peso ascendentemente.	Si
NormalizarTerminos()	El método recibe como entrada todos los pesos de cada término del diccionario.	El método debe normalizar cada peso a un intervalo de 0 a 1.	Normalización de cada peso a un intervalo de 0 a 1.	Si
add(Term t)	El método recibe como entrada un objeto Term correspondiente a un Vector de la clase Matriz.	El método debe extraer del objeto su término correspondiente y el peso, agregándolo al diccionario en caso de no encontrarse o sino aumentando su peso global.	Extrae del objeto su término correspondiente y el peso, agregándolo al diccionario en caso de no encontrarse o sino aumentando su peso global.	Si
CalcularMin()	El método recibe como entrada todos los pesos de cada término del diccionario.	El método debe obtener el índice mínimo de frecuencia para determinar si un término es frecuente.	Obtiene el índice mínimo de frecuencia para determinar si un término es frecuente.	Si
CalcularMax()	El método recibe como entrada todos los pesos de cada término del diccionario.	El método debe obtener el máximo valor de frecuencia global.	Obtiene el máximo valor de frecuencia global.	Si

Tabla 3.5. HU-2 Crear diccionario de términos. Clase Diccionario

Durante la primera iteración de pruebas realizada no se detectaron errores en la codificación de los métodos probados, obteniéndose el resultado esperado en cada una de las pruebas realizadas. Por tanto, al no existir error alguno en le Iteración 1, entonces se puede permitir dar paso a la próxima iteración.

Pruebas unitarias. Iteración 2

La segunda iteración de pruebas unitarias se realizó sobre las historias de usuario siguientes: HU-4 Construir matriz de semejanza inicial, HU-5 Ejecutar algoritmos jerárquicos y HU-6 Realizar poda de dendrograma. Las tablas 3.6 y 3.7 muestran el resultado de la ejecución de las baterías de prueba realizadas a las clases Semejanza y MatrizSemejanza, generadas para la resolución de las historias de usuario HU-4 Construir matriz de semejanza inicial. También se encuentran en los anexos las tablas correspondientes a las historias de usuario HU-5 Ejecutar algoritmos jerárquicos y HU-6 Realizar poda de dendrograma. En ellas se muestran los mismos campos descritos anteriormente.

Pruebas Unitarias				
Método a probar	**Entrada**	**Resultado esperado**	**Resultado obtenido**	**Funciona correctamente**
CalcularSemejanza()	El método recibe como entrada dos vectores.	El método debe retornar un valor numérico correspondiente a la semejanza entre dos vectores según un coeficiente de similitud.	Valor numérico correspondiente a la semejanza entre dos vectores según un coeficiente de similitud.	Si
getSemejanza()	El método recibe como entrada dos vectores.	El método debe retornar un valor numérico correspondiente a una parte de la fórmula de los coeficientes de similitud.	Valor numérico correspondiente a una parte de la fórmula de los coeficientes de similitud.	Si
getDistorcion()	El método recibe como entrada dos vectores.	El método debe retornar un valor numérico correspondiente a una parte de la fórmula de los coeficientes de similitud.	Valor numérico correspondiente a una parte de la fórmula de los coeficientes de similitud.	Si

Tabla 3.6. HU-4 Construir matriz de semejanza inicial. Clase Semejanza

Pruebas Unitarias				
Método a probar	**Entrada**	**Resultado esperado**	**Resultado obtenido**	**Funciona correctamente**

CalcularSemejanzas()	El método recibe como entrada una colección de vectores.	El método debe retornar una matriz numérica triangular que contenga la semejanza entre todos los vectores de la colección.	Matriz numérica triangular que contiene la semejanza entre todos los vectores de la colección.	Si
CompletarMatriz()	El método recibe como entrada una matriz numérica triangular.	El método debe actualizar las posiciones correspondientes a la matriz para poder ser consultada en cualquier orden.	Las posiciones correspondientes a la matriz no sufrieron ningún cambio.	No

Tabla 3.7. HU-4 Construir matriz de semejanza inicial. Clase MatrizSemejanza

En la segunda iteración se detectaron errores de codificación en el método CompletarMatriz() de la clase MatrizSemejanza al actualizar las posiciones. Tras aplicar el método se identificó que las posiciones no sufrieron cambios, evitando de esta forma que la matriz pueda ser consultada en cualquier orden. También se detectaron errores de codificación en los métodos UnificarClusters(), ActualizarMatriz(), WPGMA(), WPGMC() y Ward() de la clase Algoritmos principalmente en las posiciones de los clústeres que se estaban analizando. Esto provocó en los primeros dos métodos clústeres mal construidos y matrices vacías, y en los otros métodos resultados erróneos. Además, se detectaron errores igualmente de codificación en los métodos SoporteGrupo(), AparicionCluster1(), AparicionCluster2() y AparicionTotal() de la clase Poda a causa del mal diseño de la concurrencia sobre la lista de MatricesNumericas que causaba la alteración del listado de matrices y por tanto la desestabilización de todo el sistema. Se dio solución a cada uno de los errores encontrados garantizando el correcto funcionamiento del método y se pasó a la siguiente iteración.

Pruebas unitarias. Iteración 3

La tercera iteración de pruebas unitarias se realiza sobre las historias de usuario siguientes: HU-7 Construir cuadro de clasificación y HU-8 Exportar cuadro de clasificación.

Las tablas 3.8 y 3.9 muestran el resultado de la ejecución de las baterías de prueba realizadas a las clases CuadroClasificacion y ReadWriteFile, generadas para la resolución de la historia de usuario HU-7 Construir cuadro de clasificación, HU-8 Exportar cuadro de clasificación respectivamente. Las mismas mantienen los campos que las tablas anteriores.

Pruebas Unitarias				
Método a probar	**Entrada**	**Resultado esperado**	**Resultado obtenido**	**Funciona correctamente**

ConstruirCuadro ()	El método recibe como entrada la lista de matrices de semejanza generadas por la clase Algoritmos.	El método debe construir el cuadro de clasificación y generar una lista de fondos correspondientes al cuadro de clasificación.	Construye el cuadro de clasificación eliminando varios documentos durante el proceso	No
Completada(List<String> enc, List<String> keys)	El método recibe como entrada dos listas de identificadores correspondientes a los clústeres encontrados y al clúster que se está analizando.	El método debe responder de manera lógica si ya el clúster que se está analizando se encuentra completado.	Responde de manera correcta si ya el clúster que se está analizando se encuentra completado.	Si
Existe(String key, Cluster c2)	El método recibe como entrada un identificador y un clúster.	El método debe responder de manera lógica si el identificador se encuentra dentro del clúster.	Responde de manera correcta si el identificador se encuentra dentro del clúster.	Si
Repetido(String key, List<String> enc)	El método recibe como entrada un identificador y la listas de identificadores correspondientes a los clústeres encontrados	El método debe responder de manera correcta si el identificador se encuentra sobre la listas de identificadores correspondientes a los clústeres encontrados.	Responde de manera correcta si el identificador se encuentra sobre la listas de identificadores correspondientes a los clústeres encontrados.	Si
CrearFondos(List<String> keys, int id)	El método recibe como entrada la listas de identificadores correspondientes a los clústeres encontrados y un identificador numérico que dará	El método debe crear todos los fondos que no fueron encontrados durante la ejecución del método ConstruirCuadro().	crear por cada llave un archivo dentro del fondo correspondiente.	No

	nombre a los fondos a crear.			
CrearArchivos(List<String> keys)	El método recibe como entrada un conjunto de llaves clústeres.	El método debe crear por cada llave un archivo dentro del fondo correspondiente.	Crea por cada llave un archivo dentro del fondo correspondiente.	Si

Tabla 3.8. HU-7 Construir cuadro de clasificación. Clase CuadroClasificacion

Pruebas Unitarias				
Método a probar	**Entrada**	**Resultado esperado**	**Resultado obtenido**	**Funciona correctamente**
Write()	El método recibe como entrada una clase CuadroClasificacion.	El método debe crear un archivo Cuadro.bin que contendrá los datos del cuadro de clasificación.	Crea un archivo Cuadro.bin que contendrá los datos del cuadro de clasificación.	Si

Tabla 3.9. HU-8 Exportar cuadro de clasificación. Clase ReadWriteFile

Durante la tercera iteración de pruebas realizada se detectaron errores de codificación en el método ConstruirCuadro() y CrearFondos(List<String> keys, int id) de la clase CuadroClasificacion. Los mismos fueron solucionados por el equipo de desarrollo de esta forma concluye el proceso de pruebas unitarias realizado a las funcionalidades seleccionadas.

Como se observa en la figura 3.1 durante el proceso de pruebas unitarias se verificaron un total de 33 operaciones del componente implementado. Durante la primera iteración no se encontraron errores en ninguna de las 9 operaciones comprobadas y en la segunda iteración de 17 operaciones solo 6 presentaron problemas. En la tercera iteración se comprobaron 7 operaciones y se obtuvieron errores en 2 de ellas, por tanto, solo se necesitó corregir errores en 8 de las operaciones involucradas en el proceso de pruebas.

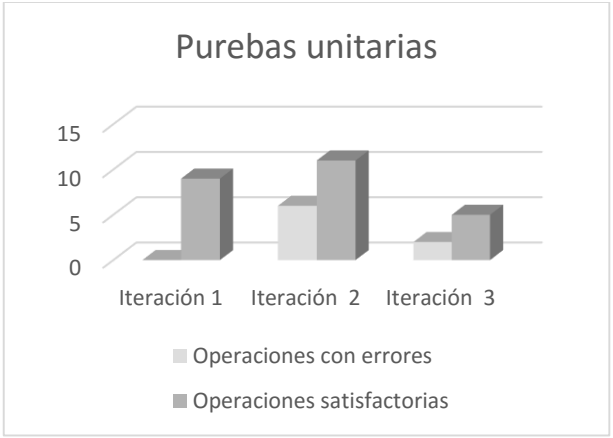

Figura 3.1. Pruebas unitarias

3.2.3 Pruebas de aceptación

Las pruebas de aceptación son una parte integral del desarrollo incremental definido por XP. Todas las historias de usuario son apoyadas por pruebas de aceptación, que se definen por el propio cliente. Las pruebas de aceptación son encaminadas a la satisfacción del cliente lo que permite el fin de una iteración y comienzo de otra (76). Las mismas obligan al cliente a un profundo conocimiento del estado de la aplicación y lo que debe hacer en circunstancias específicas (71). La realización de las pruebas de aceptación en cada iteración del proceso de desarrollo de software está asociada a cada historia de usuario. Para ello se realiza un modelo que involucra: acción a probar, datos de prueba, resultado esperado, resultado obtenido y evaluación de la prueba.

Pruebas de aceptación. Iteración 1

La tabla 3.8 muestra los datos obtenidos a partir de la aplicación de las pruebas de aceptación a las historias de usuario HU-1 Recibir colección de entrada, HU-2 Crear diccionario de términos y HU-3 Normalizar conjunto de vectores.

Pruebas de Aceptación				
Acción a probar	**Datos de prueba**	**Resultado esperado**	**Resultado obtenido**	**Evaluación de la prueba**
HU-1: el componente debe ser capaz de importar el archivo Matriz.bin que contiene la colección de vectores.	La prueba da inicio con el archivo Matriz.bin que contiene aproximadamente 1000 vectores.	Todos los vectores deben ser importados a la clase Matriz.	El componente procesó correctamente todos los datos de prueba.	**Aceptada**
HU-3: el componente debe ser capaz de crear un diccionario de términos que contenga el peso global de cada termino.	La prueba da inicio con la clase Matriz que contiene aproximadamente 1000 vectores.	Todos los términos de cada vector debe ser agregado a la clase Diccionario, calculando su respectivo peso global.	Se agregaron correctamente todos los términos, calculando su respectivo peso global.	**Aceptada**
HU-2: el componente debe ser capaz de transformar todos los vectores a una misma cardinalidad.	La prueba da inicio con 100 vectores.	Todos los vectores deben transformarse a un mismo espacio vectorial.	El tiempo de normalización demoró un total de 1 hora.	**No conformidad:** Tiempo de ejecución demasiado alto.

Tabla 3.10. Pruebas de aceptación. Iteración 1

Para la primera iteración se detectó una no conformidad en el proceso de Normalizar conjunto de vectores. Para esta iteración se solucionó la no conformidad mediante la aplicación de patrones de concurrencia. Las no conformidades son corregidas por el equipo de desarrollo lo que permite dar paso a la siguiente iteración.

Pruebas de aceptación. Iteración 2

La tabla 3.9 muestra los datos obtenidos a partir de la aplicación de las pruebas de aceptación a las historias de usuario HU-4 Construir matriz de semejanza inicial, HU-5 Ejecutar algoritmos jerárquicos y HU-6 Realizar poda de dendrograma.

Pruebas de Aceptación				
Acción a probar	**Datos de prueba**	**Resultado esperado**	**Resultado obtenido**	**Evaluación de la prueba**
HU-4: el componente debe ser capaz de dado una colección de vectores determinar la similitud existente entre ellos.	La prueba da inicio con 100 vectores.	El componente debe retornar una matriz de semejanza con el valor numérico correspondiente con la similitud entre los vectores.	El tiempo de construcción de la matriz demoró un total de 2 horas.	**No conformidad:** Tiempo de ejecución demasiado alto.
HU-5: el componente debe ser capaz ejecutar el algoritmo y retornar una estructura de datos capaz de construir el cuadro de clasificación.	La prueba da inicio con una matriz de semejanza de 100 vectores.	El componente debe retornar una estructura de datos basada en matrices.	El tiempo de ejecución del algoritmo demoró más de 5 horas.	**No conformidad:** Tiempo de ejecución demasiado alto.
HU-6: el componente debe ser capaz de podar el dendrograma en cada iteración del algoritmo de agrupamiento.	La prueba da inicio con la ejecución del algoritmo de agrupamiento sobre una matriz de semejanza de 100 vectores.	El componente debe determinar si en la iteración del algoritmo debe fusionar los clústeres creados	Realiza la fusión de los clústeres creados en cada iteración del algoritmo.	**Pedido de cambio:** La poda del dendrograma debe tender a un tercer nivel.

Tabla 3.11. Pruebas de aceptación. Iteración 2

En la segunda iteración se detectaron dos no conformidades asociadas a las historias de usuario: HU-4 y HU-5, además de un pedido de cambio en la HU-6. Para esta iteración se solucionan las no conformidades mediante la aplicación de patrones de concurrencia. Los pedidos de cambios son aceptados y mediante el ajuste del criterio de poda se logra una tendencia a podar el dendrograma en un nivel tres. Con las no conformidades resueltas y los pedidos de cambio realizados entonces se puede dar paso a la siguiente iteración.

Pruebas de aceptación. Iteración 3

La tabla 3.10 muestra los datos obtenidos a partir de la aplicación de las pruebas de aceptación a las historias de usuario HU-7 Construir cuadro de clasificación y HU-8 Exportar cuadro de clasificación.

Pruebas de Aceptación				
Acción a probar	**Datos de prueba**	**Resultado esperado**	**Resultado obtenido**	**Evaluación de la prueba**
HU-7: el componente debe crear un objeto que contenga el cuadro de clasificación.	La prueba da inicio con el conjunto de matrices obtenido de la aplicación del algoritmo de agrupamiento a 100 vectores.	Debe construir un objeto que contenga el cuadro de clasificación.	Construye un objeto que contiene el cuadro de clasificación.	**Pedido de cambio:** Los archivos dentro del cuadro de clasificación deben estar representados por el nombre del archivo.
HU-8: el componente debe ser capaz de exportar a un archivo Cuadro.bin un cuadro de clasificación.	La prueba da inicio con el objeto que contiene el cuadro de clasificación.	Debe crear un archivo Cuadro.bin que guardará la información correspondiente al cuadro de clasificación.	Crea un archivo Cuadro.bin que guardará la información correspondiente al cuadro de clasificación.	**Aceptada**

Tabla 3.12. Pruebas de aceptación. Iteración 3

En la tercera iteración se detecta una no conformidad correspondiente con el proceso de construcción del cuadro de clasificación. Para esta iteración se solucionan las no conformidades sustituyendo el identificador del documento por el nombre del mismo. Finalmente se corrige la no conformidad encontrada en las pruebas de aceptación y culmina el proceso de desarrollo del software.

Como se muestra en la figura 3.4 el proceso de pruebas de aceptación se realizó en tres iteraciones a un total de ocho acciones. Durante la primera iteración se obtuvo solamente una no conformidad sobre la

solución desarrollada, la cual fue corregida satisfactoriamente. En la segunda iteración se detectaron dos no conformidades y un pedido de cambio que fueron corregidos forma correcta. Luego se realizó la tercera iteración detectando otro pedido de cambio, el cual también fue solucionado. En total se obtuvieron tres no conformidades, dos pedidos de cambio durante el proceso de pruebas y se aprobaron tres de las acciones.

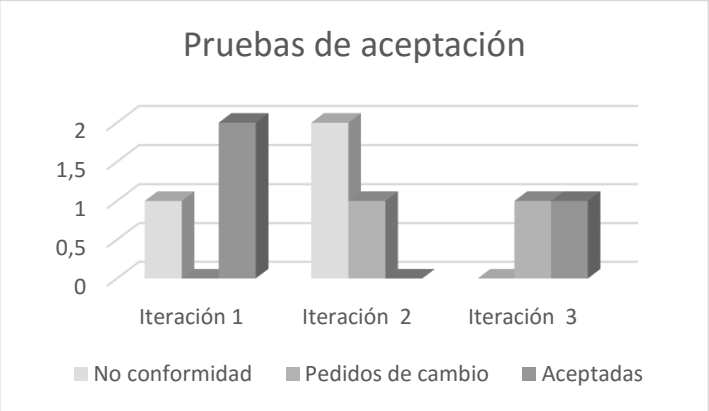

Figura 3.2. Pruebas de aceptación

Conclusiones

El presente documento describe el proceso de investigación realizado para dar solución al problema de cómo generar automáticamente una primera aproximación del cuadro de clasificación archivística para sistemas de gestión de archivos desarrollados en la plataforma Java. Para ello se identificaron los conceptos fundamentales que permitieron la comprensión de la archivística, los sistemas de gestión de archivos y el cuadro de clasificación archivística dentro de éstos. Lo anterior, ilustra la necesidad de generar el cuadro de clasificación archivística empleando la clasificación automática. Para ello, se realizó un estudio de los antecedentes referente a los sistemas de gestión de archivos que aplican clasificación archivística y de alguna manera introducen conceptos de clasificación automática. Además, se caracterizaron algoritmos de agrupamiento jerárquico tales como: Simple-Link, Complete-Link, UPGMA, WPGMA, UPGMC, WPGMC y Ward. Debido a que permiten definir una estructura de capaz de representar un cuadro de clasificación archivística. A ello se le sumó la selección de un criterio de poda basado en el algoritmo FIHC que trabaja específicamente con la frecuencia de términos.

Para preparar el ambiente de desarrollo se realizó el estudio de las metodologías de desarrollo de software FDD, SCRUM y Crystal Methodologies y se seleccionó la metodología XP para guiar el desarrollo de la solución, por las características del equipo de desarrollo, la cercanía del cliente y el tiempo de desarrollo. En apoyo a la metodología se seleccionó el lenguaje de modelado UML y Visual Paradigm Suite 5.0 para el modelado de artefactos de la propuesta de solución. También se caracterizó el lenguaje de programación Java y se seleccionó como entorno de desarrollo NetBeans 8.1 para la codificación e implementación de la solución. Todo ello permitió la caracterización de la situación existente en el mundo, la región y el país en particular sobre la generación automática del cuadro de clasificación archivística en sistemas de gestión de archivos, con el empleo de técnicas de agrupamiento jerárquico.

Se expuso además la propuesta de solución y a partir de ello se ilustraron los elementos fundamentales de la misma. Se mostró el modelo general de propuesta de solución seguido del modelo de dominio; que permitieron especificar los conceptos básicos de la problemática y sus relaciones. Fueron diseñadas un total de 8 historias de usuario, planificadas en 4 iteraciones además de crearse el plan de entregas de las funcionalidades. Se diseñó y mostró la arquitectura de software para el desarrollo de la solución escogiendo para ello una arquitectura en 3 capas lógicas: Entrada, Procesamiento y Salida. Fueron descritos los patrones de diseño utilizados tales como: Experto, Creador, Alta Cohesión, Bajo Acoplamiento y se mostró un ejemplo de la utilización de cada uno. Se expuso el algoritmo implementado, las estructuras de datos utilizadas para la construcción del cuadro de clasificación, así como el análisis de los resultados obtenidos de los algoritmos y el resultado de las mejoras en tiempo de ejecución gracias al uso de la programación concurrente. Lo anterior da cumplimiento al desarrollo de un componente informático para generar automáticamente una primera aproximación del cuadro de clasificación archivística para sistemas de gestión de archivos.

Fue comprobado también el funcionamiento de los distintos algoritmos implementados correspondientes con los coeficientes de similitud realizando las pruebas necesarias utilizando un caso de estudio

empleando un conjunto de datos de prueba provenientes de Reuters-21578. Los mismos arrojaron que el algoritmo más preciso para la resolución de este problema en específico es el algoritmo de agrupamiento jerárquico UPGMA y la función de similitud recomendada es el Coeficiente Coseno. Para verificar el correcto funcionamiento de la solución implementada se definió una estrategia de prueba, en la que se decidió la aplicación de pruebas unitarias y de aceptación. Tales pruebas fueron realizadas en tres iteraciones y arrojaron tres no conformidades y dos pedidos de cambio que fueron resueltas y realizados respectivamente. Estas pruebas permitieron detectar y corregir errores de codificación en las funcionalidades durante la implementación, aportando un nivel de calidad aceptable al producto final. Por tanto, se logró comprobar con la utilización de las métricas para evaluar la calidad de los agrupamientos junto a las pruebas de software, la validez y funcionamiento del componente de software desarrollado.

Por lo expuesto anteriormente, se puede concluir que se logró desarrollar de forma efectiva un componente de software para generar automáticamente una primera aproximación del cuadro de clasificación archivística para sistemas de gestión de archivos desarrollados en la plataforma Java, utilizando técnicas de agrupamiento jerárquico.

Recomendaciones

Luego de concluido el trabajo “Componente de software para la generación automática del cuadro de clasificación archivística” y el cumplimiento de los objetivos planteados se propone a modo de recomendación lo siguiente:

- ✓ Emplear el framework de Java para el manejo de la concurrencia (java.util.concurrent.*) para aprovechar al máximo los procesos y núcleos disponibles en la arquitectura multiprocesador; en aras de mejorar la utilización de hilos, procesos y trabajos, empleando memoria compartida, semáforos y sincronización.
- ✓ Aplicar y comparar los resultados de otros algoritmos y métodos de poda para aumentar el espectro de criterios a evaluar para la efectividad tanto en la fusión de grupos como en la generación del dendrograma homologo al cuadro de clasificación.
- ✓ Utilizar el test de Student para la definición según los métodos estadísticos que este plantea del algoritmo que mejores resultados muestra según los casos de estudio.
- ✓ Emplear otros patrones de diseño tales como el Facade para disminuir el acoplamiento entre las capas de la arquitectura de la solución.
- ✓ Exportar la definición del dendrograma generado a un fichero .xml como estándar para su comparación y uso en otras aplicaciones.

Referencias

1. Villanueva Bazán, Gustavo. Teoría y práctica archivistica - Volumen 2. [En línea] 2000. [Citado el: 2 de junio de 2016.] https://books.google.com.cu/books?isbn=9683682545.

2. Rossini, Daniel. LOS ARCHIVOS Y LAS NUEVAS TECNOLOGIAS DE LA INFORMACION. [En línea] 2003. [Citado el: 2 de junio de 2016.] http://eprints.rclis.org/4651/1/lapaz11.pdf.

3. Diccionario de la lengua española. [En línea] Real Academia Española, 2016. [Citado el: 2 de junio de 2016.] http://dle.rae.es/?id=3SrKnVZ.

4. Vasquez Lema, Marcelo R. *Gestión documental e implementación de sistemas de gestión de calidad-ISO 9001.* 2014.

5. Jaén García, Luis Fernando. *El Sistema Nacional de Información Archivística: como modelo de unificación de archivos.* s.l. : Universidad de Costa Rica, 2006. 996893626X.

6. Vitoria-Gasteiz. Metodología de Digitalización de Documentos. [En línea] 2008. [Citado el: 3 de junio de 2016.] http://www.zuzenean.euskadi.eus/s68-contay/eu/contenidos/informacion/modelo_gestion_documental/eu_modgesdo/adjuntos/Metodolog%C3%ADa%20de%20Digitalizaci%C3%B3n%20de%20Documentos.pdf.

7. Vielra Braga, Luis Paulo, Ortiz Valencia, Luis Iván y Ramirez Carvajal, Santiago Segundo. *Introducción a la Mineria de Datos.* 2009. 8576502313.

8. Eíto Brun, Ricardo y Senso, José Antonio. Mineria textual. [En línea] 2004. [Citado el: 21 de abril de 2016.] http://eprints.rclis.org/11491/1/Artmineriapdf.pdf.

9. Yero Guevara, Andis Eloy y Reyes Estévez, Pável. *Solución Informática para la representación del corpus textual de archivos digitales .* Habana : s.n., 2015.

10. García Serrano, Alberto. *Inteligencia artificial : fundamentos, práctica y aplicaciones.* 2012. 8493945021.

11. Carlos G. Figuerola, José L. Alonso Berrocal, Angel F. Zazo Rodríguez, Emilio Rodríguez. Algunas Técnicas de Clasificación Automática de Documentos. [En línea] 2014. [Citado el: 29 de abril de 2016.] http://reina.usal.es.

12. Athento. Gestión Documental Inteligente. [En línea] [Citado el: 13 de febrero de 2016.] http://www.athento.com/gestion-documental-inteligente/.

13. Aspen. Solución de software para la gestión documental. [En línea] [Citado el: 13 de febrero de 2016.] http://www.docpath.com/es/document-management-software-solution-aspen.aspx.

14. UCI. gespro. Suite de Gestión de Proyectos. [En línea] [Citado el: 26 de febrero de 2016.] http://gespro.ciged.prod.uci.cu. .

15. Heredia Herrera, Antonia. *Archivística general: teoría y práctica.* s.l. : UNAM, 1991. 9683683746.

16. Herrera Acosta, Reina Estrella. Clasificación de documentos. [En línea] 2009. [Citado el: 6 de junio de 2016.] http://catalogo.bnjm.cu/cgi-bin/koha/opac-detail.pl?biblionumber=86996.

17. Martín Gavilán, César . Principios generales de organización de fondos archivísticos. Clasificación y ordenación de documentos. Cuadro de clasificación. [En línea] 2009. [Citado el: 15 de marzo de 2016.] http://eprints.rclis.org/14526/1/principios.pdf.

18. Guinchat, Claire y Menou, Michel. Introducción general a las ciencias y técnicas de la información y de la documentación. [En línea] 1983. [Citado el: 22 de febrero de 2016.] http://unesdoc.unesco.org/images/0018/001842/184243so.pdf.

19. Cruz Mundet, José Ramón. *La gestión de documentos en las organizaciones.* Madrid : s.n., 2006. 8436820657.

20. Russo Gallo, Patricia. *Gestión documental en las organizaciones.* s.l. : UOC, 2011. 8497882938.

21. plc., Cornwell Affiliates. MODELO DE REQUISITOS PARA LA GESTIÓN DE DOCUMENTOS ELECTRÓNICOS DE ARCHIVO . [En línea] 2001. [Citado el: 22 de mayo de 2016.] http://ec.europa.eu/archival-policy/moreq/doc/moreq_es.pdf.

22. Rui Xu, Don Wunsch. *Clustering.* s.l. : IEEE, 2008. 0470382783.

23. J. Kearns, Michael. *The Computational Complexity of Machine Learning.* s.l. : Massachusetts Institute of Technology, 1990. 0262111527.

24. Piza Volio, Eduardo. Métodos de Clasificación Automática. [En línea] 2010. [Citado el: 12 de abril de 2016.] http://www.stat.rice.edu/~jrojo/PASI/lectures/Costa%20rica/5_Clasificacion%20Automatica.pdf.

25. Sebastiani, Fabrizio. Machine learning in automated text categorization. [En línea] 2002. [Citado el: 8 de mayo de 2016.] http://dl.acm.org/citation.cfm?id=505283.

26. Rodrígez Salazar, Maria Elena, Ávlarez Hernández, Sergio y Bravo Núñes, Ernesto. *Coeficientes de Asociación.* 2000. 9688569011.

27. García Monsalve, Luz Stella . Experimento de recuperación de información usando las medidas de similitud coseno, jaccard y dice. [En línea] 2012. [Citado el: 30 de abril de 2016.] https://dialnet.unirioja.es/descarga/articulo/5113370.pdf.

28. Singhal, Amit . Modern Information Retrieval: A Brief Overview. [En línea] 2001. [Citado el: 12 de febrero de 2016.] http://singhal.info/ieee2001.pdf.

29. JENSEN, RICHARD y QIANG, SHEN. COMPUTATIONAL INTELLIGENCE AND FEATURE SELECTION Rough and Fuzzy Approaches. [En línea] 2008. [Citado el: 3 de abril de 2016.] https://books.google.com.cu/books?isbn=0470377917.

30. García Caballero, Ricardo . HERRAMIENTAS PARA LA GESTIÓN DE LOS DOCUMENTOS ELECTRÓNICOS EN LOS NUEVOS SERVICIOS DE INFORMACIÓN Y DOCUMENTACIÓN. [En línea] [Citado el: 12 de abril de 2016.] http://www.cobdc.org/jornades/7JCD/27.pdf.

31. OpenKM. Software de Gestión Documental | OpenKM. [En línea] [Citado el: 12 de junio de 2016.] https://www.openkm.com/es.html.

32. Lakaut. Lakaut | Gestión Documental Inteligente. [En línea] [Citado el: 29 de mayo de 2016.] http://www.lakaut.com.ar/.

33. Eibe , Frank y H. Witten, Ian. Data mining : practical machine learning tools and techniques. [En línea] 2005. [Citado el: 15 de junio de 2016.] https://books.google.com.cu/books?isbn=008047702X.

34. Yaari, Yaakov. Segmentation of Expository Texts by Hierarchical Agglomerative Clustering. [En línea] 1997. [Citado el: 12 de abril de 2016.] http://arxiv.org/abs/cmp-lg/9709015.

35. S. Everitt, Brian , y otros. Cluster Analysis. [En línea] 2011. [Citado el: 21 de mayo de 2016.] https://books.google.com.cu/books?isbn=0470978449 .

36. S. Dhillon, Inderjit, Mallela, Subramanyam y Kumar, Rahul. A Divisive Information-Theoretic Feature Clustering Algorithm for Text Classification. [En línea] 2003. [Citado el: 26 de abril de 2016.] http://www.jmlr.org/papers/v3/dhillon03a.html.

37. Clarke, Bertrand , Fokoue, Ernest y Helen Zhang, Hao. Principles and Theory for Data Mining and Machine Learning. [En línea] 2009. [Citado el: 11 de marzo de 2016.] https://books.google.com.cu/books?isbn=0387981357.

38. C.M. Fung, Benjamin, Wang, Ke y Ester, Martin . Hierarchical Document Clustering Using Frequent Itemsets. [En línea] 2002. [Citado el: 7 de junio de 2016.] https://www.cs.sfu.ca/~ester/papers/FWE03Camera.pdf.

39. Mojena, R. Hierachical grouping methods and stopping rules: An avaluation. [En línea] 1975. [Citado el: 6 de junio de 2016.] http://comjnl.oxfordjournals.org/content/20/4/359.full.pdf.

40. A. Ingaramo, Diego y L. Errecalde , Marcelo . Medidas internas y externas en el agrupamiento de res´umenes cient´ıficos de dominios reducidos. [En línea] 2002. [Citado el: 1 de junio de 2016.] http://users.dsic.upv.es/~prosso/resources/IngaramoEtAl%20_SEPLN07.pdf.

41. STEINBACH, M. y KARYPIS, G. A Comparison of Document Clustering Techniques. [En línea] 2002. [Citado el: 5 de junio de 2016.] http://citeseer.ist.psu.edu/steinbach00comparison.html .

42. Longmore, Chris , J Wills, Andy y Milton, Fraser. Processes of Overall Similarity Sorting in Free Classification. [En línea] 2008. [Citado el: 12 de junio de 2016.] https://www.researchgate.net/publication/51393913_Processes_of_Overall_Similarity_Sorting_in_Free_Classification.

43. Fernández González, Rodolfo. Técnica de Inteligencia Artificial en Minería de Datos. [En línea] [Citado el: 22 de abril de 2016.] http://www.ucm.es/info/pslogica/rodolfo2.pdf.

44. LETELIER, P. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 2006. [Citado el: 4 de abril de 2016.] http://www.cyta.com.ar/ta0502/v5n2a1.htm.

45. AMBLER, S. W. Feature Driven Development (FDD) and Agile Modeling. Agile Modeling. [En línea] 2014. [Citado el: 20 de febrero de 2016.] http://www.agilemodeling.com/essays/fdd.htm.

46. LARMAN, C. *UML y Patrones.* New York : s.n., 1999.

47. VILLAGÓMEZ, E. Herramienta Case. Prezi. [En línea] 2013. [Citado el: 21 de febrero de 2016.] https://prezi.com/pdojfgbqbwlz/herramientas-case.

48. VÁZQUEZ ORIHUELA, A. Visual Paradigm. Scribd. . [En línea] [Citado el: 24 de febrero de 2016.] http://es.scribd.com/doc/166415572/Visual-Paradigm#scribd.

49. EXES. Curso de Introducción a Java. El lenguaje java. . [En línea] [Citado el: 23 de febrero de 2016.] http://www.mundojava.net/caracteristicas-del-lenguaje.html?Pg=java_inicial_4_1.html.

50. PROGRAMACIÓNDESARROLLO. Qué es un entorno de desarrollo integrado, IDE. [En línea] 2011. [Citado el: 25 de febrero de 2016.] http://programaciondesarrollo.es/que-es-un-entorno-dedesarrollo-integrado-ide.

51. NETBEANS. Oracle Corporation and/or its affiliates. . [En línea] [Citado el: 26 de febrero de 2016.] http://netbeans.org.

52. Hastie, Trevor , Tibshirani, Robert y Friedman, Jerome . The Elements of Statistical Learning: Data Mining, Inference, and Prediction. [En línea] 2009. [Citado el: 26 de mayo de 2016.] http://statweb.stanford.edu/~tibs/ElemStatLearn/.

53. CAROLINA MARTÍNEZ, I. *Modelo Conceptual / Modelo de Dominio.* Caracas : Universidad Simón Bolívar : s.n., 2010.

54. WELLS, D. The Rules of Extreme Programming. Extreme Programing. . [En línea] [Citado el: 12 de junio de 2016.] http://www.extremeprogramming.org/rules.html.

55. AMBLER, S. W. Agile Modeling and eXtreme Programming (XP). Agile Modeling. . [En línea] 2012. [Citado el: 16 de junio de 2016.] http://agilemodeling.com/essays/agileModelingXP.html.

56. BOLIVARIANA, U. Programación extrema XP. Ingenieria de Software. . [En línea] [Citado el: 16 de junio de 2016.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

57. PMO. ¿Qué son las historias de usuario? PMOinformatica.com. [En línea] 2013. [Citado el: 12 de junio de 2016.] http://www.pmoinformatica.com/.

58. A. SÁNCHEZ, E., LETELIER, P. y CANÓS, J. H. *Mejorando la gestión de historias de usuario en eXtreme Programming.* Valencia : s.n., 2004.

59. ECHEVERRY TOBÓN, L. M. y DELGADO CARMONA, L. E. *Caso práctico de la metodología ágil XP al desarrollo de software.* Pereira : s.n., 2007.

60. PRESSMAN, R. S. *Ingeniería de Software. Un enfoque práctico. Quinta.* s.l. : Mc Graw Hill, 2002.

61. L. BASS, P. y CLEMENTS, R. K. *Software Architecture in Practice. Segunda. .* New York : Addison Wesley, 2003.

62. INNOCUO. Patrones de diseño, la importancia de aprenderlos. innocuo. . [En línea] 2006. [Citado el: 3 de mayo de 2016.] http://blog.innocuo.com/2006/09/patrones-de-diseno-la-importanciade-aprenderlos.

63. MARTÍNEZ JUAN, F. J. *Guía de construcción de software en java con patrones de diseño. .* Oviedo : s.n.

64. DRAKE, J.M. Programación Concurrente. CTR - Computadores y Tiempo Real. . [En línea] [Citado el: 8 de junio de 2016.] http://www.ctr.unican.es/asignaturas/procodis_3_ii/doc/procodis_1_01.pdf. .

65. Casteleiro Villalba, José Manuel. Las matrices son fáciles: Manual autodidáctico. [En línea] 2010. [Citado el: 3 de abril de 2016.] https://books.google.com.cu/books?isbn=8473566815.

66. Maharam-Stone, Dorothy , y otros. *Measure and Measurable Dynamics.* s.l. : American Mathematical Soc., 1989.

67. LANG, K. The 20 Newsgroups data set. [En línea] 2005. [Citado el: 7 de junio de 2016.] http://people.csail.mit.edu/jrennie/20Newsgroups/ .

68. WAYNE, C. y DODDINGTON, G. DT2 Multilanguage Text Version 4.0. [En línea] 2001. [Citado el: 7 de junio de 2016.] http://projects.ldc.upenn.edu/TDT2/ .

69. LEWIS, D. D. Reuters-21578. [En línea] 1999. [Citado el: 7 de junio de 2016.] http://www.daviddlewis.com/resources/testcollections/reuters21578.

70. Lewis, David D. Reuters-21578 text categorization test collection. [En línea] AT&T Labs - Research , 1997. [Citado el: 4 de junio de 2016.] https://archive.ics.uci.edu/ml/machine-learning-databases/reuters21578-mld/README.txt.

71. BECK, K. *Extreme Programming Explained. .* New York : Addison-Wesley Professional, 1999.

72. MOORE, J. W. *SWEBOOK: Guide to the Software Engineering Body of Knowledge.* California : IEEE Computer Society, 2004.

73. UNAD. Capítulo 9. Estrategia de prueba de software. Universidad Nacional Abierta y a Distancia. . [En línea] [Citado el: 9 de junio de 2016.] http://datateca.unad.edu.co/contenidos/301404/301404_ContenidoEnLinea/captulo_9_estrategias_de_pru eba_del_software.html. .

74. SOMMARIVA, A. Pruebas Unitarias, Parte 1: Introducción y utilización de objetos simulados (Mock). MicroGestion. [En línea] 2014. [Citado el: 9 de junio de 2016.] http://www.microgestion.com/index.php/mg-developers/articulos/74-unit-test-part1-mock. .

75. MAVEN. JUnit. [En línea] 2014. [Citado el: 10 de junio de 2016.] http://junit.org/..

76. KEN AUER, R. M. *Extreme Programming Applied.* New York : Addison-Wesley Professional, 2001.

77. México, Archivo General de la Nación. Cuadro general de clasificación archivística. [En línea] 2012. [Citado el: 2 de junio de 2016.] http://www.agn.gob.mx/menuprincipal/archivistica/pdf/instructivoCuadroClasificacion06072012.pdf.

78. FLORES CUETO, J. J. y BERTOLOTTI ZUÑIGA, C. Diagrama de clases en UML. Diagrama de clases en UML. [En línea] [Citado el: 19 de marzo de 2016.] http://es.scribd.com/doc/31096724/Diagrama-de-Clasesen-UML#scribd.