Universidad de las Ciencias Informáticas

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

Título:

Componente de clasificación de huellas dactilares.

Autor:

Alicia Delgado Hernández

Tutor:

Ing. Ramón Santana Fernández

Co-tutor:

Ing. Adrián Rivera Correa

La Habana, Junio de 2013

Declaración de autoría

Declaro ser autora del presente trabajo de diploma titulado “Componente de clasificación de huellas dactilares” y reconozco a la **Universidad de las Ciencias Informáticas** los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2013.

_____

Autora: Alicia Delgado Hernández

_____

Tutor: Ing. Ramón Santana Fernández

_____

Co-tutor: Ing. Adrián Rivera Correa

Datos de contacto

Tutor: Ing. Ramón Santana Fernández.

Ingeniero en Ciencias Informáticas. Profesor Instructor asociado al Centro de Identificación y Seguridad Digital (CISED), líder del proyecto Dactilab del Departamento de Biometría, graduado en el curso 2010-2011. Tiene 2 años de experiencia en el tema. Presentó un trabajo en FESI 2011 y en Uciencia 2012 en el cual obtuvo resultados, además de exponer un trabajo en el Fórum de Ciencia y Técnica en el cual obtuvo la categoría de destacado, todos ellos relacionados con los temas de reconstrucción, generación y extracción de características de huellas dactilares.

Correo electrónico: rsfernandez@uci.cu

Co-tutor: Ing. Adrián Rivera Correa.

Ingeniero en Ciencias Informáticas. Profesor asociado al Centro de Identificación y Seguridad Digital (CISED), pertenece al Departamento de Biometría, graduado en el curso 2011-2012. Tiene 1 año de experiencia en el tema.

Correo electrónico: acorrea@uci.cu

"A la memoria de mi padre va dedicado el resultado no solo de este trabajo de diploma, sino todas las metas alcanzadas durante estos cinco años desde que partió para siempre.

Gracias por ser el mejor padre del mundo, por confiar en mí, en cualquier lugar donde estés, estoy convencida de que vas a sentirte muy orgulloso de tu niña."

Agradecimientos

A la Revolución por permitirme formar parte de este grandioso "proyecto futuro".

A mi mamá y a mi hermano por acompañarme en mis tristezas y alegrías, y soportarme cuando se tornaba imposible hacerlo.

A toda mi familia, y en especial a mis abuelitos Hilda y Pablo, y mi tía Mary por todo el afecto y cariño que me brindan.

A Frankis Alberto por ser mi ángel de la guardia a cada instante, por su amor, su apoyo y comprensión incondicional.

A mis tutores y a Adrián Hernández por contribuir a la materialización de este trabajo.

A todos esos buenos profesores que conocí y durante estos cinco años aportaron su granito de arena para mi formación profesional.

A mis amigos, a mis compañeros de brigada y a todas esas personas especiales de mi vida, por darme fuerzas para continuar cuando el camino se hacía difícil.

A Reisel por ser para mí más que mi amigo, mi hermano, y al que tengo la certeza de que voy a extrañar mucho.

A todos realmente muchas gracias.

"No hacen falta alas para hacer un sueño, basta con las manos, basta con el pecho, basta con las piernas y con el empeño…"

Silvio Rodríguez

Resumen

Los sistemas biométricos son sistemas automatizados para el reconocimiento de personas que fundamentan sus decisiones en los rasgos conductuales o físicos intrínsecos de éstas. De manera particular la identificación dactilar, requiere la comparación de una huella con las almacenadas en una base de datos, que en aplicaciones a gran escala, tiende a ser un proceso engorroso, por lo que se utilizan técnicas de indexación que le proporcionen escalabilidad y modularidad a estos sistemas. La herramienta de identificación dactilar existente en el **Centro de Identificación y Seguridad Digital** (CISED) de la **Universidad de las Ciencias Informáticas** (UCI), realiza este proceso de forma centralizada.

Una estrategia común para reducir el número de comparaciones, y en consecuencia, disminuir el tiempo de respuesta de este proceso, es dividir la base de datos en clases predefinidas, de acuerdo a la uniformidad del recorrido de sus crestas papilares. Por ello el presente trabajo de diploma propone una solución cuyo objetivo es clasificar una huella dactilar de entrada a partir de sus puntos característicos y orientación de su imagen, que ofrezca un criterio válido para una posterior segmentación de la base de datos del sistema al que se integre.

El componente de clasificación de huellas dactilares creado, guiado por un enfoque ágil usando la metodología Programación Extrema (XP), se codificó con el lenguaje de programación C# sobre el entorno integrado de desarrollo Microsoft Visual Studio 2012, para que su integración a la herramienta existente en el CISED, creada en este mismo lenguaje, sea menos compleja.

Palabras claves: huellas dactilares, clasificación de patrón, Sistemas Automáticos de Identificación por Huellas Dactilares (AFIS), biometría, estrategias de búsqueda.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad la evolución de las tecnologías de la información y las comunicaciones ha permitido automatizar y perfeccionar los sistemas de identificación personal, de forma que poseen múltiples aplicaciones y finalidades. Los sistemas de reconocimiento biométrico surgen como complemento de seguridad respecto a los sistemas de autenticación e identificación por posesión[1] y por conocimiento[2], con el propósito de reducir las posibilidades de que un tercero suplante la identidad de un individuo para la comisión de algún delito.

La biometría (del griego *bios* vida y *métro* medida) es el estudio de métodos automáticos para el reconocimiento único de humanos basados en uno o más de sus rasgos conductuales o físicos intrínsecos. El proceso de identificación consiste en la comparación de dichas características físicas o conductuales con un patrón conocido y almacenado previamente en una base de datos (1).

Las huellas dactilares, la retina, el iris, los patrones faciales, los de las venas de la mano y la geometría de la palma de la mano, representan ejemplos de características físicas o estáticas, mientras que entre las de comportamiento o dinámicas se incluyen la firma manuscrita y el modo de andar.

De manera particular, el uso de la huella dactilar es la más antigua de las técnicas biométricas y el rasgo individual más utilizado para el proceso de autenticación e identificación personal por su facilidad de adquisición, de uso, fiabilidad y gran aceptación por parte de los usuarios (1). Sus orígenes se remontan al siglo XIX, cuando investigadores en criminología intentaron relacionar las características físicas de los individuos con tendencias criminales, sin embargo no es hasta finales de los años 60 que, con el desarrollo de las tecnologías de la computación, esta técnica comienza su transición a la automatización, marcando el surgimiento de los **Sistemas Automáticos de Identificación por Huellas Dactilares** (AFIS de sus siglas en inglés).

Un **AFIS** es un sistema informático compuesto de hardware y software integrados, que permite la captura, consulta y comparación automática de huellas dactilares (2). En el proceso de identificación de una persona la entrada de estas aplicaciones es la imagen de la huella dactilar y la salida es una lista con las identidades candidatas de toda la base de datos a coincidir con la huella de entrada, además de un índice que indica el grado de similitud.

La estrategia que se utilice en la etapa de búsqueda y cotejo de las huellas dactilares en estos sistemas, incide directamente en su rendimiento y tiempo de respuesta. Si este proceso se ejecuta de forma

[1]**Sistema de autenticación por posesión:** utiliza un pequeño dispositivo de hardware que los usuarios cargan consigo para autorizar el acceso a un servicio de red. El dispositivo puede ser en forma de una tarjeta inteligente o puede estar incorporado a un objeto utilizado comúnmente.

[2]**Modelo de autenticación por conocimiento:** consiste en decidir si un usuario es quien dice ser a partir de una prueba de conocimiento, que solo él pueda superar, esa prueba no es más que una contraseña que en principio es secreta y está compuesta por un código alfanumérico y en ocasiones solamente numérico.

centralizada, tiene como ventajas entre otras características, que el almacenamiento de todos los datos se hace en un único nodo, por lo que las normas de seguridad de la información solo deben ser aplicadas a éste, sin embargo el análisis lineal que se debe realizar, buscando la equivalencia de 1:N (uno a muchos) entre toda la información, requiere de un gran procesamiento para ser ejecutado por una unidad de cotejo, de aquí la necesidad de implementar variantes que brinden escalabilidad y modularidad a dicho proceso, que de una forma u otra disminuyan su campo de análisis y en consecuencia su tiempo de respuesta.

Una de las alternativas es desarrollar un sistema distribuido cuya arquitectura esté organizada en un grupo de unidades de cotejo. Esta metodología permite flexibilidad en el balanceo de la carga de trabajo y una alta fiabilidad y disponibilidad del sistema (3). Cada unidad de comparación desarrollaría comparaciones en una parte específica de la base de datos de minucias del AFIS (sub-base de datos). No obstante, uno de los aspectos negativos de esta técnica radica en el costo adicional que implica la adquisición y posterior mantenimiento de toda la infraestructura de equipos y software requeridos.

Desde otra perspectiva, el análisis del recorrido de las crestas papilares de una huella dactilar concluye en que las mismas no describen formas aleatorias, sino que se agrupan hasta llegar a formar sistemas definidos por la uniformidad de su orientación y figura. La mayoría de los sistemas de identificación utilizando las huellas dactilares reconocen como clases, las propuestas por el sistema de Henry: arco, arco tendido, lazo izquierdo, lazo derecho y espiral, cuyas proporciones reales en huellas dactilares son 3.7%, 2.9%, 33.8%, 31.7% y 27.9% respectivamente (4) .

La Universidad de las Ciencias Informáticas **(UCI)**, como elemento de su infraestructura productiva cuenta con el Centro de Identificación y Seguridad Digital **(CISED).** En el departamento de Biometría de dicho centro, existe una pequeña versión de un AFIS llamada SourceAFIS, que entre sus principales funcionalidades permite el enrolamiento de los usuarios en el sistema biométrico, la verificación de una identidad, así como la identificación de una huella anónima, proceso que se realiza utilizando una estrategia centralizada que resulta en aproximadamente 10 000 comparaciones por segundo, estadísticas que en consonancia con los datos obtenidos por otros sistemas homólogos existentes en el mundo, indican que la herramienta no es lo suficientemente rápida como se requiere, constituyendo ésta la **situación problemática** de la presente investigación.

El análisis de esta situación problemática da paso a la formulación del siguiente **problema de la investigación**: ¿Cómo reducir el conjunto de datos a analizar durante el proceso de búsqueda en la base de datos de un AFIS para contribuir a la disminución en sus tiempos de respuesta?

Ante el problema definido anteriormente, se determina como **objeto de estudio** los procesos de clasificación de huellas dactilares.

El **objetivo general** de la investigación es desarrollar un módulo que realice el proceso de clasificación de huellas dactilares mediante sus singularidades, lográndose con esto un componente del sistema de control de acceso con verificación biométrica del CISED.

Se definen como objetivos específicos:

- Analizar los aspectos teórico-prácticos referentes a los sistemas de identificación biométrica haciendo uso de las huellas dactilares.
- Definir los algoritmos, tecnologías, metodologías y herramientas a utilizar para el desarrollo del componente.
- Implementar un componente de clasificación de huellas dactilares que aporte un criterio para la posterior segmentación de la base de datos del AFIS al que se integre.
- Realizar pruebas de software del componente desarrollado para comprobar su correcto funcionamiento.

Para dar cumplimiento a dichos objetivos se plantean como **tareas de la investigación:**

1) Definición de los aspectos teóricos referentes al proceso de identificación haciendo uso de las huellas dactilares.
2) Análisis del estado del arte respecto a las diferentes alternativas aplicadas durante el proceso de búsqueda y comparación de un AFIS.
3) Definición de las tecnologías, metodología y herramientas a utilizar para el desarrollo del componente.
4) Especificación de los requisitos del software.
5) Definición de la arquitectura de software del componente para la clasificación de huellas dactilares.
6) Diseño de las interfaces gráficas de prueba del componente.
7) Implementación del algoritmo para la lectura de las imágenes de huellas dactilares.
8) Realización de pruebas funcionales continuas.
9) Implementación del algoritmo de pre-procesamiento de imágenes de huellas dactilares.
10) Implementación del algoritmo para la extracción de los puntos característicos de las huellas dactilares.
11) Implementación del algoritmo para la clasificación de las huellas dactilares.
12) Realización de pruebas de aceptación del componente.

Los **Métodos Científicos** empleados son:

Métodos Teóricos:

- Analítico-Sintético: se recurre para analizar las diversas teorías de huellas dactilares, definir sus características generales, además de examinar el tema de clasificación de imágenes de huellas dactilares.
- Análisis Histórico-Lógico: se utiliza en el estudio de los antecedentes, la evolución y el desarrollo que han tenido los sistemas biométricos de huellas dactilares, así como para valorar las tendencias actuales de los algoritmos de lectura, pre-procesamiento, extracción de características y clasificación de huellas dactilares.
- Modelación: permite crear los diagramas correspondientes al diseño del software, de manera que contribuyan a comprender el funcionamiento del mismo.

Justificación de la Investigación:

La determinación y compromiso de la industria de los sistemas biométricos automatizados y las necesidades de los gobiernos a nivel mundial, llevan a una concepción en reconocimiento de huellas dactilares, que promete y requiere de dispositivos y programas de alta calidad, rápidos y con un elevado costo de adquisición, para producir una mayor exactitud y confiabilidad.

Por ello Cuba, país subdesarrollado, precisa la creación de soluciones biométricas propias, que cuenten con la calidad y rendimiento que demandan este tipo de productos internacionalmente. Por tanto el aporte práctico de la presente investigación va dirigido a la obtención de:

- Componente para la clasificación de huellas dactilares, que aporte un criterio para una posterior fragmentación de la base de datos de cualquier AFIS al que se integre, con el propósito de disminuir su campo de análisis y tiempo de respuesta en el proceso de búsqueda y comparación.

El presente trabajo de diploma se divide en tres capítulos estructurados de la siguiente forma:

- **Capítulo I: Fundamentación teórica.** En este capítulo se puntualizan los principales conceptos relacionados con el tema, se realiza el estudio del estado del arte a nivel internacional y nacional, sobre las estrategias utilizadas por diferentes AFIS en el proceso de búsqueda y cotejo de una huella dactilar, se caracterizan los principales algoritmos de clasificación de huellas dactilares, además de definirse las tecnologías, metodología y herramientas a utilizar durante el desarrollo del módulo.
- **Capítulo II: Características de la solución propuesta.** En este capítulo además de realizarse una descripción general de la solución propuesta y su funcionamiento, se detallan los principales aspectos relacionados con su diseño. Se capturan los requisitos funcionales y no funcionales. Se define la arquitectura que organice la lógica del módulo, además de especificarse los patrones del

diseño que se van a aplicar y los artefactos derivados de la metodología de desarrollo de software que se seleccione.

- **Capítulo III: Implementación y Pruebas.** En este capítulo se abordan aspectos relacionados con la implementación del componente. Se realizan por los desarrolladores, las principales pruebas unitarias al módulo dirigidas al código, para verificar que responda a un correcto funcionamiento. Se construyen los casos de prueba de aceptación en base a los requisitos definidos inicialmente para el desarrollo de la aplicación.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA.

Introducción

En el presente capítulo se presentan los conceptos fundamentales que constituyen la base teórica de los sistemas de reconocimiento biométrico basados en huellas dactilares. Se realiza un estudio del estado del arte sobre las diferentes alternativas aplicadas durante el proceso de búsqueda y comparación en los AFIS, existentes a nivel internacional y nacional, además de exponerse las principales características, ventajas y desventajas de los algoritmos usados para realizar el proceso de clasificación de huellas dactilares. Se analizan las metodologías, tecnologías, y herramientas actuales que guían el proceso de desarrollo de software, y se seleccionan las más adecuadas a utilizar durante el ciclo de desarrollo de la solución que se propone.

1.1. Reconocimiento biométrico

La biometría es un método de reconocimiento de personas basado en sus características físicas o de comportamiento. El reconocimiento biométrico a partir de las características físicas del individuo se determina por considerar parámetros derivados de la medición directa de algún rasgo estrictamente físico del cuerpo humano a la hora de identificarlo, ya sea la huella dactilar, rostro, iris, entre otros, mientras que las técnicas de comportamiento se definen analizando en el proceso de identificación rasgos derivados de una acción realizada por la persona, tales como la voz, la firma manuscrita, escritura en el teclado, entre otros (5).

1.2. Definición de huellas dactilares

Una huella dactilar es la impresión visible o moldeada que produce el contacto de las crestas papilares de un dedo de la mano (generalmente se usa el dedo pulgar o el dedo índice) sobre una superficie, que como característica individual distingue a todos los seres humanos (6).

1.2.1. Propiedades de las huellas dactilares

El reconocimiento biométrico haciendo uso de las huellas dactilares se basa en tres principios fundamentales que cumplen éstas:

- Perennidad: desde que se forman en el sexto mes de la vida intrauterina, permanecen indefectiblemente invariables en número, situación, forma y dirección hasta que la putrefacción del cadáver destruye la piel.
- Inmutabilidad: las crestas papilares no pueden modificarse fisiológicamente, de existir un traumatismo poco profundo, se regeneran, y si es profundo, no reaparecen con forma distinta a la que tenían, sino que la parte afectada por el traumatismo resulta en una cicatriz.
- Diversidad Infinita: son únicas e irrepetibles, por lo que identifican unívocamente a cada individuo.

1.2.2. Características de las huellas dactilares

- **Valles y crestas:** las huellas dactilares están constituidas por rugosidades que forman salientes y depresiones. Las salientes se denominan crestas papilares y las depresiones surcos inter-papilares o valles. Una cresta está definida como un segmento de curva y un valle es la región entre dos crestas adyacentes (7). Las crestas, en una imagen de una huella dactilar usualmente aparecen como una serie de líneas oscuras que representan los relieves, mientras que los valles entre estas crestas aparecen como espacio en blanco y están en bajo relieve.

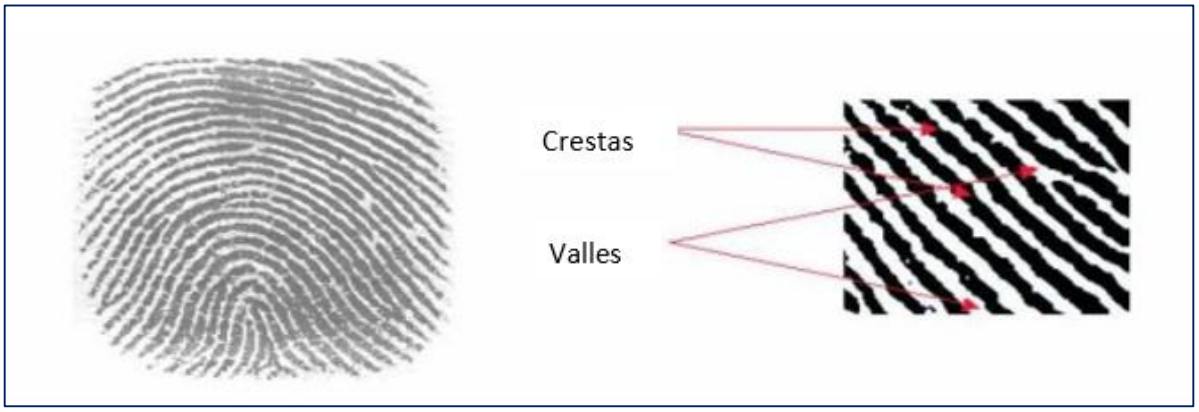

Figura 1.1 Valles y crestas de una huella dactilar.

- **Minucia:** punto de interés de la huella dactilar, que se representa como: $\boldsymbol{minucia} = \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}\}$ donde $\boldsymbol{x}$ y $\boldsymbol{y}$ son la posición en la imagen de la huella, y $\boldsymbol{\theta}$ es el ángulo en radianes de dicha minucia (7). Estas líneas se pueden clasificar en diferentes tipos (ver figura 1.2), no obstante, las dos clases que aportan mayor información de una huella son las terminaciones y las bifurcaciones puesto que el resto de las clasificaciones puede verse como combinaciones de éstas.

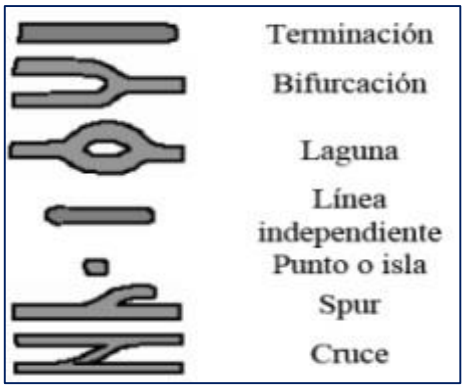

Figura 1.2 Tipos de minucias.

  - **Minucia de Terminación:** es donde la cresta termina.
  - **Minucia de Bifurcación:** es donde la cresta se divide.
- **Núcleo o core:** punto de máxima curvatura de la cresta más interna. Centro de la huella dactilar, donde se genera el inicio del lazo en la huella o el inicio del círculo (8).

- **Delta:** centro de un conjunto de crestas inferiores al núcleo, que dibujan varios triángulos concéntricos (8).

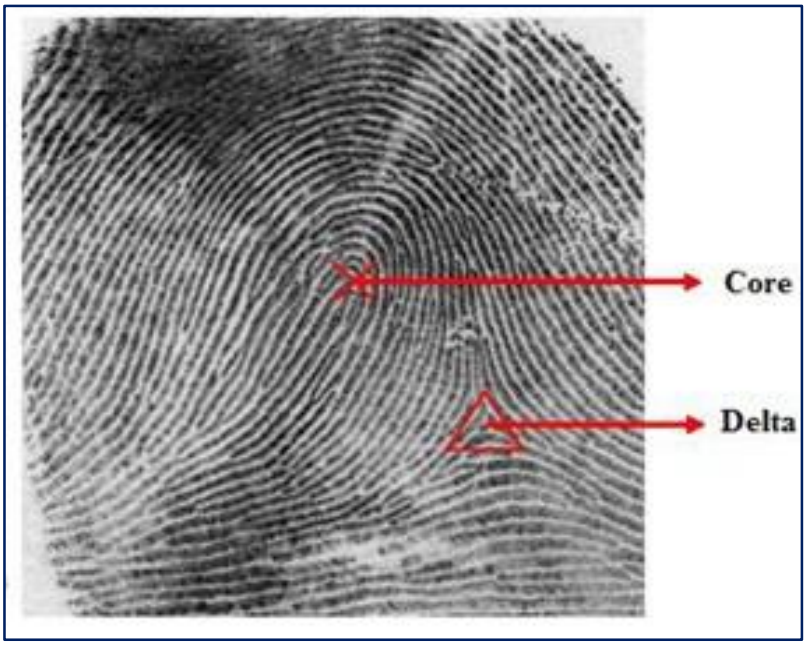

Figura 1.3 Núcleo y delta.

1.3. Clasificación de las huellas dactilares

La clasificación de huellas dactilares consiste en asignar de modo consistente y fiable cada huella a una clase, de manera tal que en la identificación de una huella desconocida, solo se necesite que ésta sea comparada con el subconjunto de huellas dactilares en la base de datos que pertenece a su misma clase (9).

1.3.1. Clasificación exclusiva o de patrón de huellas dactilares

De acuerdo a la forma que describen los dibujos papilares de una huella dactilar, se pueden distinguir cinco grupos o clases distintas de configuraciones dérmicas según la denominada Clasificación de Henry:

a) **Arco:** este dactilograma carece de puntos delta y de núcleo. Las crestas fluyen de un lado al otro de la imagen, curvándose ligeramente en el centro de la misma.

b) **Arco Tendido:** presenta un punto delta y un núcleo. Semeja a un arco, pero con la diferencia de que es una línea que rápidamente nace y se pierde en un paso de ángulo.

c) **Lazo Izquierdo:** las crestas entran por el lado izquierdo de la imagen, fluyen hacia arriba rodeando el núcleo y luego dan vuelta sobre sí mismas para regresar al mismo punto de partida. Este dactilograma cuenta con un núcleo y un punto delta ubicado del lado derecho del observador.

d) **Lazo Derecho:** las crestas entran por el lado derecho de la imagen, fluyen hacia arriba rodeando el núcleo y luego dan vuelta sobre sí mismas y regresan al mismo punto de partida. Existe un núcleo y un punto delta que aparece del lado izquierdo del observador.

e) **Espiral:** la característica más importante de este tipo de huella dactilar es que cuenta con dos puntos delta entre los cuales fluyen las crestas curvándose alrededor de un núcleo que puede adoptar formas circulares, elípticas y espirales.

Figura 1.4 Clasificación de las huellas dactilares.

1.3.2. Clasificación continua o topológica de huellas dactilares

En la clasificación continua las huellas dactilares no son distribuidas en clases, se asocian con vectores numéricos que resumen sus principales características, creados en un espacio multidimensional a través de una transformación, preservando las semejanzas de manera que las huellas similares tienden a cerrar puntos. Durante el proceso de búsqueda, se consideran las huellas cuyos puntos están dentro de un radio dado a partir de la consulta de la huella dactilar de entrada (10).

1.4. Estado del arte

Los sistemas biométricos haciendo uso de las huellas dactilares tuvieron sus orígenes en el siglo XIX. En 1856, sir William Herschel fue el primero en implantar la huella dactilar como método de identificación en documentos para personas analfabetas. El 28 de octubre de 1880 Henry Faulds, médico escocés que trabajaba en Tokio, publicó un artículo en la revista Nature sobre cómo identificar criminales a partir de sus huellas dactilares llamado “On the Skin-Furrows of the Hand” (1).

En 1892 Sir Francis Galton, publicó el libro "Finger Prints", que contenía un estudio detallado de las huellas dactilares y en donde además presentó un novedoso método de clasificación usando las impresiones dactilares de los 10 dedos de las manos. Con este método, cuyo uso trasciende hasta la actualidad, se identifican las características por las que las huellas dactilares pueden ser clasificadas.

Sin embargo, no es hasta más de medio siglo después, que esta técnica de reconocimiento de humanos comienza su transición a la automatización, tomando un subconjunto de los puntos de Galton, “minucias o rasgos específicos”, como base de dichos sistemas. En 1969, hubo un empuje mayor por parte del **Buró Federal de Investigaciones** (FBI) para desarrollar un sistema que automatizara sus procesos manuales de identificación por huellas dactilares, los cuales requerían muchas horas hombre (11).

En el año 1975, el FBI inició el desarrollo de escáneres de huella dactilar para clasificadores automatizados y tecnología de extracción de minucias, lo cual condujo a la creación de un lector prototipo. En ese momento solo los datos biográficos de los individuos, la clasificación de las huellas dactilares y las minucias eran archivados, a causa de que el almacenamiento de las imágenes digitales de las huellas dactilares era prohibido. La tecnología de huellas dactilares disponible continuó mejorando y para el año 1981, cinco sistemas automatizados de identificación por huella dactilar fueron desplegados (12).

Así, la organización de impresiones dactilares en archivos manuales utilizando sistemas decadactilares, cuya infraestructura requiere grandes espacios físicos, paulatinamente con el paso del tiempo va quedando obsoleta, dando cobertura a la implementación de los AFIS.

El funcionamiento de un AFIS posibilita guardar las imágenes de las huellas dactilares en una base de datos informática en formato aceptado por las organizaciones policiales del mundo, acceso rápido y automático a éstas, y determinación automática de sus puntos característicos, proceso que en forma manual demanda mucho tiempo y recursos.

Además estos sistemas proponen herramientas para la modificación y mejoramiento de las imágenes con el propósito de recuperar las de mala calidad o latentes, permiten ejecutar búsquedas automáticas por mono dactilar, latente o decadactilar, basado en las ciencias biométricas, la matemática, los cálculos de la transformada de Fourier, la coherencia y la correlación, a partir de la lectura de una imagen alineada de rasgos integrales paralelos, con bifurcaciones aleatorias, pero que establecen una figura integrada por “puntos”, que en el caso del registro electrónico se denominan “píxeles” (2).

1.4.1. Nivel Internacional

En el mundo, el número de participantes en la industria biométrica crece gradualmente, llevando a que el mercado sea cada vez más complejo con el desarrollo de tecnologías de alta calidad, a la medida de los intereses de los usuarios finales. Los diferentes productos se distinguen en términos de aplicación respecto a los patrones biométricos que utilizan.

Entre las instituciones más reconocidas que trabajan en la industria de la tecnología biométrica de huellas dactilares se pueden citar: el grupo SAFRAN surgido en el año 2005 a partir de la fusión de Sagem Morpho Inc. y SNECMA, Neurotechnology, Cogent System Inc., Cross Match Technologies Inc., ZKsotfware Inc., Identix Inc. y SecuGen Corporation.

- **MegaMatcher**

La compañía de Tecnologías Biométricas e Inteligencia Artificial “Neurotechnology” proporciona algoritmos y productos de desarrollo de software para huella dactilar, rostro, iris, voz y reconocimiento de la impresión de la palma de la mano, visión por computadora y reconocimiento de objetos, a empresas de seguridad, integradores de sistemas y fabricantes de hardware (13).

Entre los productos de esta compañía está la tecnología MegaMatcher, disponible como Kit de Desarrollo de Software (SDK de sus siglas en inglés) que permite crear productos de identificación de huellas dactilares, rostros, iris o multi-biométricos de gran escala para Microsoft Windows y Linux (14).

El motor de identificación de huellas dactilares MegaMatcher se caracteriza por:

- Completa certificación MINEX: el Instituto Nacional de Estándares y Tecnología (NIST de sus siglas en inglés) ha certificado el algoritmo de huellas MegaMatcher para su uso en programas de verificación de identidad personal.
- Comparación de huellas planas y roladas entre sí.
- Incorpora una determinación de la calidad de la imagen, que puede usarse durante la captura para asegurar que sólo las plantillas de mejor calidad lleguen a la base de datos.
- Generalización de plantillas: se usa para generar un registro de mejor calidad a partir de varias huellas.
- Es tolerante a la traslación, rotación y deformación de huellas.
- Comparación rápida pre-ordenando la base de datos: para algunas tareas de identificación MegaMatcher puede incrementar la velocidad de comparación hasta 960.000 huellas por segundo pre-organizando la base de datos utilizando ciertas características.
- El algoritmo de filtrado de adaptación de imágenes elimina ruidos, ruptura de trazos y obstrucciones para extraer las minucias de forma segura, inclusive a partir de imágenes de calidad pobre, en menos de 1 segundo.

Las siguientes arquitecturas y motores de comparación se utilizan dependiendo de la velocidad requerida, el tamaño de la base de datos y el alcance del sistema:

- Servidor Unitario MegaMatcher: compara 160.000 huellas o 1.400.000 rostros o 800.000 iris por segundo. Requiere MegaMatcher Estándar SDK.
- Clúster MegaMatcher: compara hasta varios millones de huellas, rostros o iris por segundo. Requiere MegaMatcher Extendido.
- MegaMatcher Accelerator 3.0 Estándar o Extendido.
- Clúster de unidades MegaMatcher Accelerator 3.0 Estándar o Extendido.

Entre los módulos de soporte de componentes incluidos para el cliente MegaMatcher están:

- **Módulo de Extracción MegaMatcher:** realiza el procesamiento de imágenes de huellas y rostros extrayendo sus características biométricas únicas, que se envían al Servidor o Clúster MegaMatcher para la identificación.
- **Módulo de Comunicación Cliente:** permite enviar tareas al Servidor o Clúster MegaMatcher o al MegaMatcher Accelerator, preguntar el estado de las tareas, obtener los resultados y eliminar tareas del servidor. Este componente oculta todas las comunicaciones de bajo nivel y proporciona

una Interfaz de Programación de Aplicaciones (API de sus siglas en inglés) de alto nivel para el desarrollador.

- **Componente Visor de Huellas (.NET):** muestra en pantalla la imagen de la huella capturada y es capaz de mostrar también los puntos característicos extraídos.
- **Módulo de Segmentación de Huellas:** separa individualmente las huellas si una imagen contiene más de una impresión dactilar. Permite procesar por separado las huellas de una tarjeta decadactilar o capturadas utilizando dispositivos para dos o más dedos.
- **Módulo de Clasificación de Patrones Dactilares:** permite determinar el patrón o clase de una huella. Típicamente utilizado en medicina forense pero puede ser usado para incrementar la velocidad de comparación.

- **Sagem Défense Sécurité**

Sagem Défense Sécurité es una empresa francesa de alta tecnología perteneciente al grupo SAFRAN. Entre sus productos se encuentra el sistema biométrico AFIS implementado a petición del Ministerio del Interior y Justicia (MIJ) de la República Bolivariana de Venezuela, con el objetivo de garantizar la integridad del proceso de solicitud y emisión de tarjetas de identidad y de pasaportes del país.

La función principal del AFIS Civil es incrementar la seguridad y confiabilidad del proceso en general, al detectar de forma automatizada los intentos de usurpación de identidad y/o de múltiple registro de una misma persona bajo diferentes identidades (15).

El módulo MetaMatcher propuesto por dicho sistema no hace uso ni depende de ningún hardware propietario, utiliza una clasificación topológica de las huellas dactilares. Su arquitectura está distribuida alrededor de un grupo de unidades de cotejo, en las que la comparación consiste en dos etapas que proveen velocidad y alta precisión:

- **1ra etapa:** un proceso de pre-filtrado, desarrollado a muy alta velocidad, calcula la rotación y la traslación de las huellas, después de haber seleccionado en la base de datos únicamente los registros que potencialmente puedan corresponder, a través del uso de sus clasificaciones.
- **2da etapa:** es realizada por otro módulo de software para refinar la lista de candidatos, por lo que se ejecuta otra comparación con una precisión más alta, obtenida espacialmente por que el algoritmo tiene en cuenta cualquier distorsión que se haya podido presentar durante la etapa de adquisición.

Este cotejo en varias etapas reporta al sistema una velocidad de 500 000 comparaciones por segundo y una precisión mayor del 98 % (15).

- **IAFIS**

El Sistema Integrado Automático de Identificación por Huellas Dactilares (IAFIS de sus siglas en inglés), constituye una herramienta nacional automatizada del gobierno de los Estados Unidos de identificación de

huellas dactilares y antecedentes penales, mantenida por el Buró Federal de Investigaciones (FBI). IAFIS proporciona capacidades de búsqueda automática de huellas digitales, capacidad de búsqueda de huellas latentes, almacenamiento electrónico de imágenes e intercambio electrónico de huellas dactilares, además de ser la mayor base de datos biométrica del mundo, que alberga las huellas dactilares y la historia criminal de 70 millones de sujetos en un archivo criminal maestro, 31 millones de impresiones de civiles y las huellas dactilares de 73.000 terroristas conocidos y sospechosos que han sido procesados por el gobierno de los Estados Unidos o por organismos internacionales de aplicación de la ley (16).

Aunque esté diseñado para responder a las transacciones electrónicas de criminales en un intervalo de 2 horas y transacciones civiles dentro de las 24 horas, IAFIS ha superado estas exigencias, proporcionando a menudo peticiones de búsqueda de criminales en menos de 20 minutos y de controles civiles en menos de 3 horas.

El sistema IAFIS de manera general está conformado por tres módulos o componentes entre los que se encuentra el **AFIS**. Su función es escanear la huella digital y generar su clasificación de patrón correspondiente (17).

1.4.2. Nivel nacional

- **DATYS, Tecnologías y Sistemas**

La empresa **"DATYS, Tecnologías y Sistemas"**, cuenta con una división destinada al desarrollo de software biométrico (Biomesys SUITE), que desde el año 2006 implementa el Biomesys AFIS, diseñado para reducir los tiempos y aumentar la efectividad, en la identificación y autenticación de personas, a partir de las impresiones dactilares con una efectividad del 99.5%, manteniendo las tasas de falsa aceptación y falso rechazo por debajo del 1%.

El sistema en su versión Civil permite la identificación y autenticación de personas de forma rápida y segura detectando los intentos de suplantación de identidad. También viabiliza la entrega de documentos de identidad con un grado de confianza acorde con la relevancia de la información que se procesa (18).

Está basado en una arquitectura de servidores entre los que se encuentra el Clúster de Búsqueda y Comparación que trabaja basado en el procesamiento distribuido para lo cual la base de datos de características de impresiones dactilares (minucias) del AFIS se encuentra particionada en los n servidores de comparación. La solución a las solicitudes de búsqueda se realiza de forma paralela en cada servidor de comparación, posteriormente se consolidan los resultados y se conforma la lista de candidatos que entrega el sistema como respuesta al usuario.

.

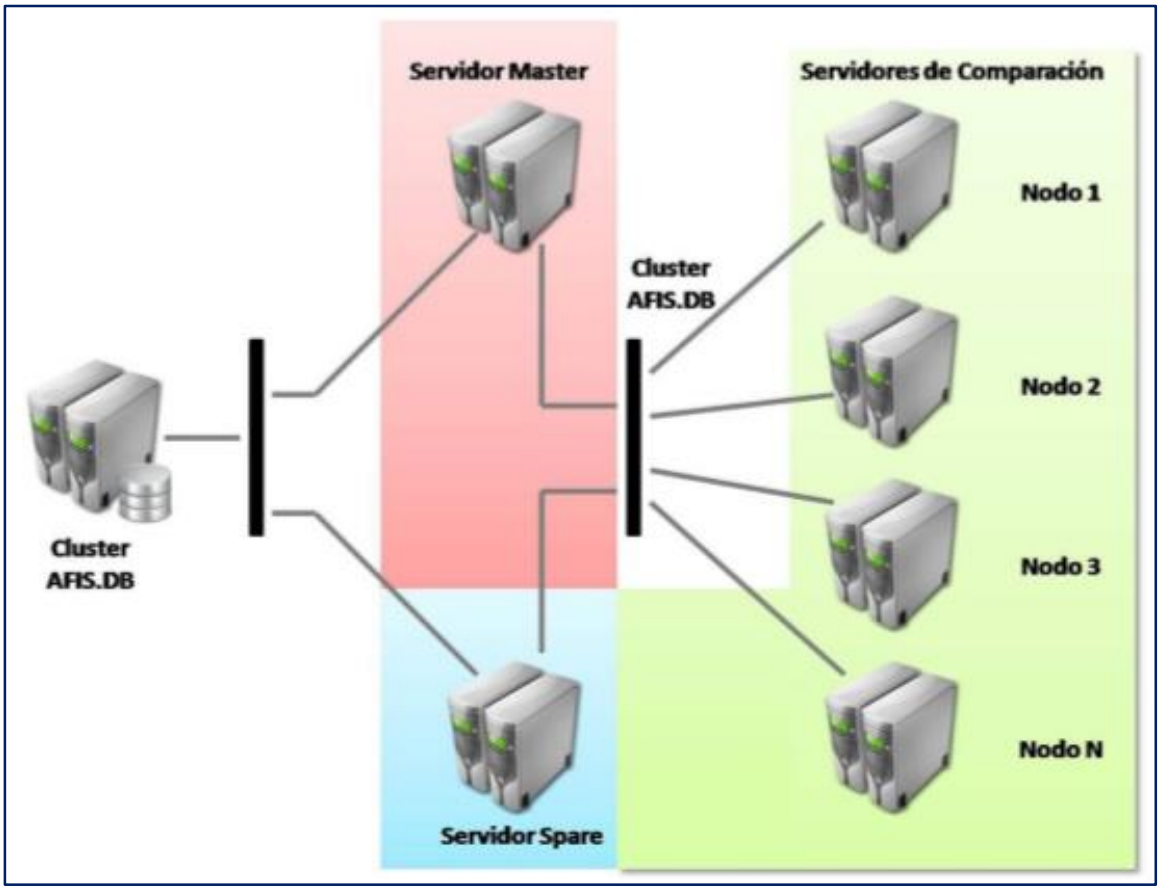

Figura 1.5 Arquitectura de búsqueda del Biomesys AFIS.

- **Universidad de las Ciencias Informáticas**

Uno de los principales propósitos de la infraestructura productiva de la UCI va dirigido al desarrollo de soluciones biométricas propias con la calidad requerida, para su uso nacional e internacional. Como elemento de su red de centros, la universidad cuenta con el CISED, cuya estructura organizativa se basa en la existencia de cinco departamentos que tienen una estrecha vinculación entre sí y responden a líneas investigativas-productivas específicas.

El Departamento de Biometría de dicho Centro desarrolla soluciones y productos en las líneas biométricas de huellas dactilares, imágenes faciales y firmas manuscritas online y offline, en relación a componentes de captura en vivo sobre dispositivos biométricos, herramientas para el procesamiento, visualización y análisis de muestras biométricas, y utilitarios para el desarrollo y prueba de algoritmos biométricos.

La variante de AFIS existente en el departamento llamada SourceAFIS, no posee todas las funcionalidades que caracterizan a estos sistemas, solo abarca la extracción de plantillas de minucias en un tiempo aproximado de 180 milisegundos, y los procesos de verificación e identificación de un usuario, con una velocidad de comparación de 10 000 huellas dactilares por segundo (19). La estrategia de

búsqueda centralizada que emplea este sistema resulta en un gran procesamiento a ejecutarse por un solo nodo, que resulta en tiempos de respuesta insatisfactorios en relación con estadísticas de sistemas similares existentes a nivel mundial.

Después de analizar sobre algunos de los AFIS existentes, comparar sus características, estrategias de comparación que utilizan y sus resultados, se considera que:

- En ninguna de estas soluciones, el proceso de búsqueda se realiza de forma centralizada, al contrario, se utilizan vías de indexación que lo distribuyen de una forma u otra. En la mayoría de los casos, además de utilizar el procesamiento distribuido en diferentes sub-bases de datos, se efectúan alguno de los dos tipos de clasificaciones de huellas dactilares, lo que contribuye a menores tiempos de respuesta y una mayor optimización del proceso.
- Los sistemas internacionales tienen como principal inconveniente que son software privativo, por lo que además del costo de adquisición que implican, tienen ciertas limitaciones para su uso, como es el caso de tener inaccesible el código fuente para hacerle modificaciones o redistribuirlo, por lo que en el caso particular de Cuba sería más factible utilizar un producto propio que cumpla la calidad y precisión adecuada, pero sobre todo que consuma la menor cantidad posible de recursos financieros.
- En Cuba la herramienta creada por DATYS, se basa en un procesamiento distribuido, sin embargo no se tiene conocimiento de que posea algún módulo que clasifique las huellas dactilares. En el caso de la UCI, el SourceAFIS ejecuta una búsqueda centralizada, y sus resultados no son competentes respecto al resto de los sistemas mostrados. Por ello la presente investigación tiene como objetivo crear un componente que ejecute la clasificación de patrón de las huellas dactilares, cuyo resultado pueda ser utilizado para fraccionar el banco de datos del AFIS al que se integre.

1.5. Principales algoritmos de clasificación de huellas dactilares

La clasificación de huellas dactilares ha atraído un significativo interés en la comunidad científica debido a su importancia y dificultad intrínseca. Aunque se hayan desarrollado una variada gama de algoritmos para dar solución a este problema, un número relativamente pequeño de características extraídas de las imágenes de huellas dactilares ha sido utilizado por la mayoría de los autores. En particular, casi todos los métodos se fundamentan en el flujo de líneas de crestas, la orientación de la imagen, los puntos característicos y/o la respuesta del filtro de Gabor (9).

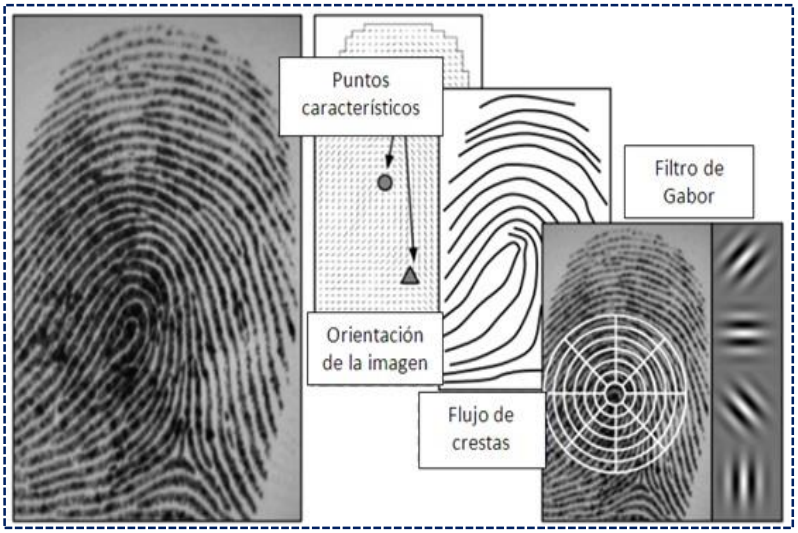

Figura 1.6 Características más usadas para la clasificación de huellas dactilares.

1.5.1. Algoritmos basados en reglas

Estos métodos realizan el proceso de clasificación de acuerdo al número y posición de las singularidades, es el patrón utilizado habitualmente por los expertos humanos para la clasificación manual.

M. Kawagoe y A. Tojo proponen:

- En un primer momento obtener como resultado una clasificación gruesa utilizando el tipo y posición de los puntos singulares.
- Luego hallar una clasificación refinada mediante el trazado de las líneas del flujo de crestas (20).

K. Karu y A.K. Jain plantean:

- Calcular la imagen direccional (orientación de la imagen) de la huella dactilar.
- Suavizar la imagen orientada.
- Localizar y detectar los puntos característicos.
- Realizar una regularización iterativa, suavizando la imagen orientada hasta que un número válido de puntos característicos sean detectados, lo que reduce el ruido y por lo tanto mejora la precisión del proceso de clasificación.
- Clasificar la imagen de la huella atendiendo al número y posición relativa de los puntos singulares. Para diferenciar entre arco tendido y lazos se conectan las dos singularidades con una línea recta y se mide la diferencia del promedio entre las orientaciones locales a lo largo de la línea y su pendiente (21).

L. Hong y A.K Jain:

- Proponen una técnica que introduce un algoritmo de clasificación basado en reglas que utiliza el número de singularidades junto con el número de crestas recurrentes encontradas en la imagen.

- La combinación de estas dos características conduce a un mejor rendimiento que el encontrado en el método anterior (22).

Desventajas del enfoque basado en reglas:

- Aunque simples, surgen algunos problemas en presencia de huellas dactilares con ruido o parciales, donde la detección de las singularidades puede ser extremadamente difícil.
- Puede funcionar bien en imágenes de impresiones de huellas dactilares escaneadas de tarjetas, pero no son adecuados para las procedentes de escáneres, puesto que los puntos delta faltan a menudo en este tipo de imágenes.

1.5.2. Algoritmos con un enfoque sintáctico

Un método sintáctico describe los patrones por medio de símbolos terminales y normas de producción. Los símbolos terminales se asocian a pequeños grupos de elementos direccionales dentro de la imagen orientada representando una clase.

K. Rao y K. Balck plantean:

- Analizar una línea de cresta y representarla por un conjunto de líneas conectadas.
- Estas líneas son marcadas de acuerdo con los cambios de dirección, obteniendo así un conjunto de cadenas que se procesan a través de gramáticas punto a punto o técnicas de cadenas de coincidencia para deducir la clasificación final (23).

Desventajas del enfoque basado en reglas:

- Debido a la gran diversidad de patrones de huellas dactilares, el enfoque sintáctico requiere gramáticas muy complejas cuya inferencia demanda de enfoques complejos e inestables.

1.5.3. Algoritmos con un enfoque estructural

Se basa en la organización relacional de características de bajo nivel en estructuras de nivel superior. Esta organización relacional se representa por medio de estructuras de datos simbólicos, que permite una organización jerárquica de la información.

D. Maio y D Maltoni presentan un método donde:

- La imagen direccional se divide en varias regiones homogéneas de forma regular, que se utilizan para construir un gráfico relacional que resume las características de la huella dactilar.
- La imagen direccional se calcula, sobre una matriz de 32 x 32 píxeles, utilizando una técnica robusta propuesta por Donahue y Rokhlin en 1993.
- Luego se adopta un algoritmo de agrupamiento dinámico de los propios autores, para segmentar la imagen direccional.
- Posteriormente se construye un gráfico relacional mediante la creación de un nodo para cada región y un arco para cada par de regiones adyacentes.

- Por último se utiliza una técnica inexacta de comparación gráfica, derivada de Bunke y Allermann, para calcular un vector "distancia" entre el gráfico de la huella y cada gráfico de la clase prototipo (24).

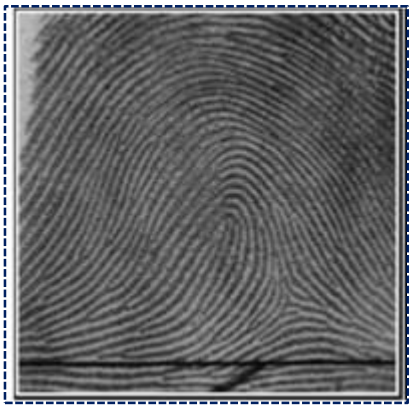

Figura 1.7 Imagen de la huella dactilar.

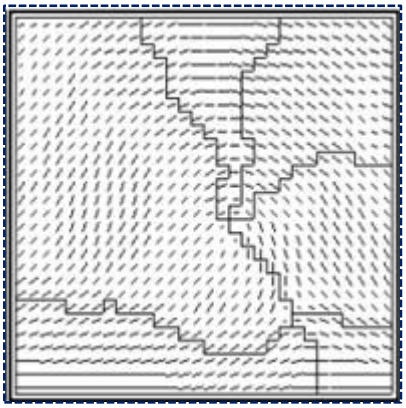

Figura 1.8 Partición de la imagen orientada.

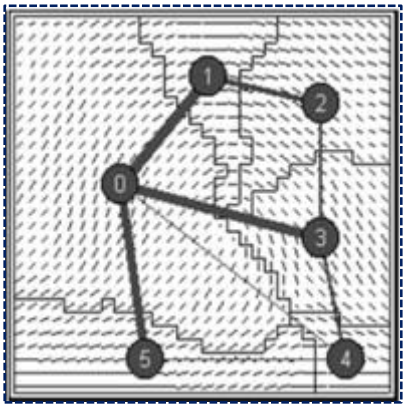

Figura 1.9 Gráfico relacional correspondiente.

Ventajas de los algoritmos que utilizan el enfoque estructural:

- Los gráficos relacionales son invariantes con respecto al desplazamiento y rotación de la imagen.
- La técnica no requiere ninguna alineación de posición ni ninguna normalización.
- En principio, se puede utilizar directamente para la clasificación de las huellas dactilares parciales.

Desventajas:

- No es sencillo obtener una partición robusta de la imagen orientada en regiones homogéneas, especialmente en las huellas digitales de baja calidad.

1.5.4. Algoritmos basados en múltiples clasificadores

Clasificadores diferentes ofrecen información complementaria sobre los patrones que van a ser clasificados lo cual puede ser explotado para mejorar el rendimiento. Esto motiva la combinación de diferentes enfoques para lograr la clasificación de las huellas dactilares.

Jain, Prabhakar y Hong proponen:

Una estrategia de clasificación en dos etapas basada en los enfoques estadísticos y red neuronal:

- Etapa 1: Un clasificador vecino k, más cercano, se utiliza para encontrar las dos clases más probables a partir de un vector de características FingerCode.
- Etapa 2: Una red neuronal específica, entrenada para distinguir entre las dos clases, se utiliza para obtener la decisión final. Un total de 10 redes neuronales son entrenadas para distinguir entre cada par posible de clases (25).

Tras el estudio de las ventajas y desventajas de los algoritmos y enfoques expuestos anteriormente, y su costo computacional, se decidió implementar el algoritmo propuesto por Kalle Karu y Anil K. Jain.

1.6. Metodologías de desarrollo de software

Una metodología de desarrollo de software, en ingeniería de software, es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información, a través de un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. (Ver diferencias entre metodologías: Anexo 1)

1.6.1. Metodologías tradicionales

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo de software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total del trabajo a realizar y una vez que esté todo detallado, comienza el ciclo de desarrollo del producto. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada (26).

- **RUP**

El **Proceso Unificado de Desarrollo** (RUP) es un proceso de software guiado por casos de uso, centrado en la arquitectura, e iterativo e incremental. Propone un levantamiento exhaustivo de requerimientos, con el propósito de detectar defectos en las fases iniciales del desarrollo del software. Intenta reducir el número de cambios realizando un análisis y diseño tan completo como sea posible, buscando anticiparse a futuras necesidades, ya que un cambio en etapas posteriores provocaría un gran incremento del costo de producción del software. Las necesidades de los clientes no son fáciles de discernir, por lo que existe un contrato prefijado con los mismos, los cuales interactúan con el equipo de desarrollo mediante reuniones (27).

Divide el proceso de desarrollo en cuatro fases:

- **Fase de inicio:** abarca la comunicación con el cliente y las actividades de planeación y destaca el desarrollo y refinamiento de casos de uso como un modelo primario.
- **Fase de elaboración**: incluye la comunicación con el cliente y las actividades de modelado con un enfoque en la creación de modelos de análisis y diseño, con énfasis en las definiciones de clases y representaciones arquitectónicas.
- **Fase de construcción:** se refina y después se traduce el modelo de diseño en componentes de software ya implementados.
- **Fase de transición:** se transfiere el software del desarrollador al usuario final para realizar las pruebas betas y obtener la aceptación (28).

Estas fases se realizan durante varias iteraciones en número variable según el proyecto, haciendo un mayor o menor hincapié en distintas actividades durante las que se pueden distinguir nueve flujos o disciplinas a realizar en mayor o menor medida en dependencia de cada etapa.

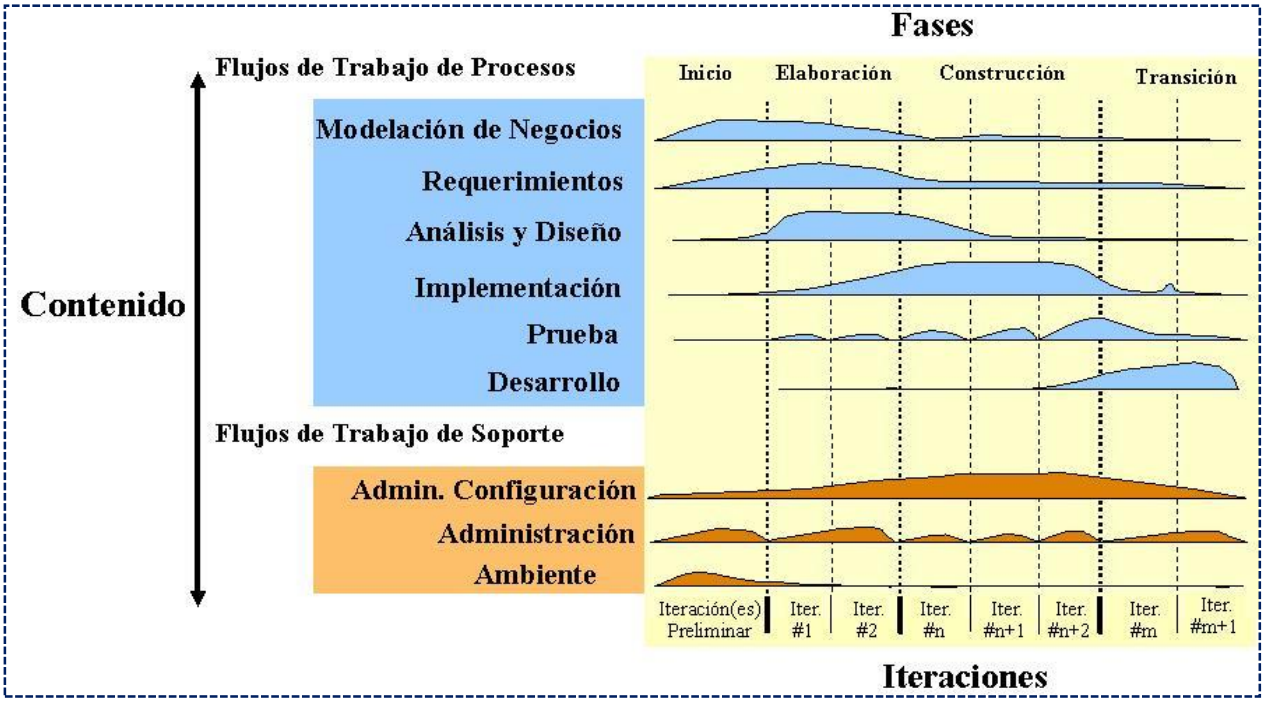

Figura 1.10 Fases y flujos de trabajo de RUP.

1.6.2. Metodologías ágiles

Las metodologías ágiles proporcionan una serie de pautas y principios junto a técnicas pragmáticas que harán la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega. Según el Manifiesto Ágil esta metodología valora (29):

- Al individuo y las interacciones del equipo de desarrollo, sobre el proceso y las herramientas.
- Desarrollar software que funcione, más que conseguir una buena documentación.
- La colaboración con el cliente, más que la negociación de un contrato.
- Responder a los cambios, más que seguir estrictamente un plan.

- **XP**

Programación Extrema (XP) es una disciplina de desarrollo de software cuyos objetivos fundamentales son alcanzar la satisfacción del cliente, intentando darle el software que él necesite y cuando lo necesite, debiendo responder muy rápido a sus necesidades, incluso cuando los cambios sean al final del ciclo de programación, y potenciar al máximo el trabajo en grupo teniendo como premisa que tanto los jefes de proyecto, los clientes y los desarrolladores son parte del equipo y están involucrados en el desarrollo del software. El ciclo de vida propuesto por la metodología se divide en las siguientes fases:

- **Fase de exploración:** los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

- **Fase de planificación:** el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, además de las entregas relacionadas con éstas. El resultado de esta fase es un Plan Entregas.
- **Fase de iteraciones:** esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entregas está compuesto por las diferentes iteraciones. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto, escogiendo las historias de usuario que fuercen la creación de esta arquitectura, sin embargo, no siempre es posible ya que es el cliente quien decide que historias se implementarán en cada iteración. Al final de la última iteración el producto estará listo para entrar en producción.
- **Fase de puesta en producción:** en esta fase se entregan módulos funcionales y sin errores, y según los intereses del cliente el sistema puede ponerse en producción o no. No se realizan desarrollos funcionales, pero pueden ser necesarias tareas de ajuste.
- **Fase de mantenimiento:** mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para esto se requiere de tareas de soporte para el cliente. La velocidad de desarrollo puede disminuir después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.
- **Fase de muerte del proyecto:** el cliente no tiene más historias para ser incluidas en el sistema, haciendo necesario que se satisfagan sus necesidades en otros aspectos tales como rendimiento y confiabilidad del mismo. Se genera la documentación final y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Durante el desarrollo de estas fases XP propone el uso de las siguientes prácticas:

- El juego de la planificación: el alcance de la siguiente versión está definido por las consideraciones del negocio (prioridad de los módulos, fechas de entrega) y estimaciones técnicas (estimaciones de funciones, consecuencias). El objetivo del juego es maximizar el valor del software producido.
- Versiones pequeñas: un sistema simple se pone rápidamente en producción. Periódicamente se producen nuevas versiones agregando en cada iteración aquellas funciones consideradas valiosas para el cliente.
- Metáfora del sistema: cada proyecto es guiado por una historia simple que explica el funcionamiento del sistema en general, reemplaza a la arquitectura y debe estar en lenguaje común, entendible para todos, cliente y desarrolladores.

- Diseño simple: el sistema se diseña con la máxima simplicidad posible. Se plasma el diseño en las tarjetas CRC (Clase - Responsabilidad - Colaboración), con lo que las clases descubiertas durante el análisis pueden ser filtradas para determinar las que son realmente necesarias para el sistema.
- Pruebas continuas: los casos de prueba se escriben antes que el código. Los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.
- Refactorización: es posible reestructurar el sistema sin cambiar su comportamiento, por ejemplo eliminando código duplicado, simplificando funciones.
- Programación por parejas: el código es escrito por dos personas trabajando en la misma computadora.
- Posesión colectiva del código: nadie es dueño de un módulo, cualquier programador puede cambiar cualquier parte del sistema en cualquier momento, por ello siempre se deben utilizar estándares de codificación.
- Integración continua: los cambios se integran en el código base varias veces por día. Todos los casos de prueba se deben pasar antes y después de la integración, se dispone de una máquina para la integración y se realizan pruebas funcionales donde participa el cliente.
- Cliente en el sitio: el equipo de desarrollo tiene acceso todo el tiempo al cliente, el cual está disponible para responder preguntas y fijar prioridades.
- Estándares de codificación: todo el código debe estar escrito de acuerdo a un estándar de codificación (29).

- **Scrum**

Scrum define un proceso empírico, iterativo e incremental de desarrollo que intenta obtener ventajas mediante la aceptación de la naturaleza caótica del desarrollo de software, y la utilización de prácticas tendientes a manejar el riesgo a niveles aceptables. Surgió como modelo para el desarrollo de productos tecnológicos, sin embargo actualmente se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad, situaciones frecuentes en el desarrollo de determinados sistemas de software (30).

El proceso de desarrollo guiado por Scrum se divide en tres fases:

Planificación del Sprint[3]: en esta fase se define el Product Backlog[4] que contiene todos los requerimientos funcionales y no funcionales que deberá satisfacer el sistema a construir. En cada iteración es revisado por el equipo. También se desarrolla la planificación del primer sprint. La planificación de cualquier sprint es la jornada de trabajo previa a su inicio y en ese momento se determinan cuáles son los objetivos y el trabajo que debe cubrirse.

[3] Sprint: Término utilizado por la metodología para referirse a una iteración del proceso de desarrollo de software.

[4] Product Backlog: documento de alto nivel para todo el proyecto que contiene las descripciones genéricas de todos los requerimientos y funcionalidades deseables, priorizadas según su retorno sobre la inversión.

Seguimiento del Sprint: a lo largo de esta fase se efectúan breves reuniones diarias, para conocer el avance de las tareas y el trabajo que está previsto para la jornada.

Revisión del Sprint: una vez finalizado el sprint se realiza un análisis y revisión del incremento generado.

1.6.3. Selección de la metodología de desarrollo de software a utilizar

Después del análisis de las diferentes características, ventajas y desventajas de diferentes metodologías de desarrollo de software, y en concordancia con las particularidades de la solución a desarrollar en la presente investigación se elige la metodología XP puesto que:

- Aunque RUP sea una de las metodologías más robustas para el análisis, diseño, implementación, soporte y documentación de sistemas informáticos orientados a objetos, el levantamiento exhaustivo de requerimientos que presupone con la premisa de detectar defectos en las fases iniciales, la realización del análisis y diseño tan completo como sea posible, y la serie de artefactos y roles que propone, indican que está concebida para grandes grupos de desarrollo donde el cliente interactúe con el equipo de desarrollo mediante reuniones, debiendo existir un contrato prefijado entre éstos que especifique el alcance del proyecto, todo lo cual no se ajusta a las particularidades del componente a desarrollar, puesto que, además de que el equipo de trabajo es pequeño, no se cuenta con una definición detallada y exacta de todos los requisitos que debe cumplir, lo que implica una alta probabilidad de ocurrencia de cambios de forma gradual, a medida que se vayan obteniendo las pequeñas versiones. Por esto es más conveniente el uso de una de las metodologías ágiles, diseñadas para pequeños equipos de desarrollo y preparadas para enfrentar cambios durante la creación del software.
- Si bien Scrum es una metodología ágil especialmente indicada para proyectos con un rápido cambio de requisitos, su poca documentación puede provocar efectos negativos en el modo de confeccionar el producto final.
- Esta metodología ágil está definida para equipos pequeños de desarrollo, tiene prioridad de satisfacer al cliente mediante un desarrollo iterativo e incremental, abierto a los cambios y caracterizado por un código simple, al mismo tiempo que aplica un conjunto de prácticas que harán la entrega del componente menos complicada y más satisfactoria tanto para los clientes como para el equipo de entrega. En el caso particular de esta investigación, el cliente, que es el jefe del Departamento, es parte del equipo de desarrollo, por lo que va a estar en condiciones de contestar rápida y correctamente a cualquier pregunta por parte del resto del equipo de desarrollo, de forma que no se atrase la toma de decisiones.

1.7. Herramientas y tecnologías

1.7.1. Lenguaje de modelado

En el proceso de desarrollo de un software, el modelado del mismo es de vital importancia y tiene una notable repercusión en su etapa de diseño e implementación, por proveer al ingeniero de un conjunto de

notaciones, herramientas y prácticas, que le permiten "visualizar" el sistema a construir, logrando un nivel de abstracción que organice la lógica del mismo.

- **UML**

El Lenguaje Unificado de Modelado (LUM o UML por sus siglas en inglés, Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de los sistemas de software (31). Ofrece un estándar para describir un "plano" del sistema, incluyendo aspectos conceptuales tales como procesos del negocio, funciones y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es fundamental aclarar que UML no es un lenguaje de programación, sino un lenguaje de modelado de propósito general, que ha demostrado su efectividad sobretodo en el área del análisis y diseño de sistemas de cómputo (32).

1.7.2. Herramientas para el diseño de la aplicación

Una CASE (de sus siglas en inglés Computer Aided Software Engineering, o Ingeniería de Software Asistida por Computadora) esencialmente, es una herramientas individual para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del ciclo de desarrollo del software (o mantenimiento) (33).

Sus objetivos van dirigidos a mejorar la productividad en el desarrollo y mantenimiento de éste, aumentar su calidad, reducir su tiempo y costo de desarrollo y mantenimiento, mejorar la planificación de un proyecto y automatizar la documentación, la generación de código, las pruebas de errores y su gestión, contribuyendo a la reutilización del software, portabilidad y estandarización de la documentación y facilitando el uso de las distintas metodologías propias de la ingeniería de software.

- **Rational Rose Enterprise Edition**

Es una herramienta CASE desarrollada por Rational Corporation basada en UML, que permite crear los diagramas que se generan durante el proceso de ingeniería en el desarrollo del software. Arquitectos, analistas, diseñadores de software y bases de datos, y desarrolladores de sistemas pueden usar este producto para producir modelos visuales que sirvan de apoyo y guía para el desarrollo del software, centrándose en los casos de uso y enfocándose hacia un producto de mayor calidad, empleando un lenguaje estándar común que facilite la comunicación.

El desarrollo es un proceso iterativo, que comienza con una aproximación del análisis, diseño e implementación para identificar los riesgos y probar el sistema, cuando la implementación pasa todas las pruebas que se determinan, se añaden los elementos modificados al modelo y una vez actualizado éste, se realiza la siguiente iteración (34).

Las ventajas de esta herramienta residen en su utilidad en aplicaciones grandes y complejas, reduce el tiempo de desarrollo de manera automática para pasar de un esquema a otro y al código, además de que

los nombres y los datos se mantienen de manera consistente proporcionando una sincronización para diferentes desarrolladores. Sin embargo tiene como desventajas su enfoque fijo de desarrollo, limitación en la flexibilidad de la documentación y los costos en software, manuales y capacitación.

- **Visual Paradigm 8.0**

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite construir todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (35).

Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de clases. Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros.

Visual Paradigm cumple con las políticas actuales de migración a software libre, siendo multiplataforma, de forma tal que facilita la modelación del software independientemente del sistema operativo que se emplee (36).

1.7.3. Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser ejecutados por máquinas computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (37).

- **C++**

Bjarne Stroustrup crea la primera versión experimental de este lenguaje hacia 1979, con la intención de proporcionar una herramienta de desarrollo para el núcleo Unix en ambientes distribuidos. En 1983 el lenguaje se rebautiza como C++ y en 1985 Stroustrup publica la primera edición del libro "The C++ Programming Language" que sirvió de estándar informal y texto de referencia. Desde sus inicios, C++ intentó ser un lenguaje que incluye completamente al lenguaje C, pero al mismo tiempo incorpora muchas características sofisticadas no incluidas en él, tales como programación orientada a objetos, excepciones, sobrecarga de operadores y plantillas (38). En ese sentido, desde el punto de vista de los lenguajes

orientados a objetos, C++ es un lenguaje híbrido. Una de sus particularidades es la posibilidad de redefinir los operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales.

- **Java**

Los inicios de Java datan de 1991 cuando James Gosling en Sun Microsystems, encabezó un proyecto cuyo objetivo original era implementar una máquina virtual ampliamente portable y un lenguaje de programación, ambos orientados a procesadores incorporados en diversos dispositivos de consumo masivo. La sintaxis del lenguaje heredó características de C y C++, explícitamente eliminando aquellas que para muchos programadores resultan excesivamente complejas e inseguras. En realidad, Java hace referencia a un conjunto de tecnologías entre las cuales el lenguaje Java es sólo una de ellas. Por tal motivo muchas veces se habla de la "plataforma Java", la cual es indesligable del lenguaje (38).

El lenguaje se caracteriza por ser:

- Simple
- Orientado a Objetos
- Familiar
- Robusto
- Seguro
- Portable.
- Independiente a la arquitectura
- Multi-hilo
- Interpretado

- **C#**

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos, que permite a los desarrolladores crear una amplia gama de aplicaciones sólidas. Su sintaxis es muy expresiva, sencilla y fácil de aprender, está basada en signos de llave, por lo que puede ser reconocida inmediatamente por cualquier persona familiarizada con C, C++ o Java. Los desarrolladores que conocen cualquiera de estos lenguajes pueden empezar a trabajar de forma productiva en C# en un plazo muy breve.

La sintaxis de C# simplifica muchas de las complejidades de C++ y, a la vez, ofrece funciones tales como tipos de valores que aceptan valores NULL, enumeraciones, delegados, métodos anónimos y acceso directo a memoria, que no se encuentran en Java. C# también admite métodos y tipos genéricos, que proporcionan mayor rendimiento y seguridad de tipos, e iteradores, que permiten a los implementadores de clases de colección definir comportamientos de iteración personalizados que en el código se pueden utilizar fácilmente (39).

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. No hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado. Un archivo de código fuente de C# puede definir cualquier número de clases, estructuras, interfaces y eventos.

Sus características quedan resumidas en:

- Sencillez de uso
- Modernidad

- Orientado a objetos
- Orientado a componentes
- Recolección de basura
- Seguridad de tipos
- Instrucciones seguras
- Extensión de los operadores básicos
- Extensión de modificadores
- Compatible

1.7.4. Entornos Integrados de Desarrollo

Un entorno integrado de desarrollo, conocido también como IDE de sus siglas en inglés, es un programa informático compuesto por un conjunto de herramientas de programación, que puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

- **Visual Studio .NET**

Microsoft Visual Studio es un IDE para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Visual Studio 2012 es la última actualización para Visual Studio, que proporciona nuevas funcionalidades y correcciones. Es un lanzamiento acumulativo que también incluye todos los beneficios obtenidos en versiones anteriores. Como parte del compromiso de Visual Studio para la entrega continua, esta versión cuenta con mejoras que abarcan áreas características en todas las ediciones, con un enfoque especial en las capacidades del ciclo de vida moderno introducido en Premium y Ultimate, centrando la atención en cuatro áreas claves: desarrollo sobre Windows, desarrollo sobre Microsoft SharePoint, los equipos ágiles, y la calidad continua (40).

- **NetBeans**

NetBeans es un IDE modular, de base estándar (normalizado), escrito en el lenguaje de programación Java. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general para compilar cualquier tipo de aplicación (41).

1.7.5. Propuesta de las herramientas y tecnologías a utilizar

Como herramientas y tecnologías para el desarrollo de la solución propuesta se propone que:

- El modelado del software se realice usando UML para visualizar, especificar, construir y documentar la solución informática, dando soporte a la metodología de desarrollo de software seleccionada.

- La herramienta CASE será la versión 8.0 de Visual Paradigm Enterprise Edition para UML, por ser un software libre que permite realizar ingeniería tanto directa como inversa y soportar múltiples usuarios trabajando sobre el mismo proyecto, además de generar la documentación del proyecto automáticamente en varios formatos.
- El lenguaje de programación será C# por ser orientado a objetos y desarrollado y estandarizado por Microsoft, además de que la herramienta existente en el centro a la cual se debe integrar el componente está implementada utilizando dicho lenguaje, lo que facilitará el trabajo a la hora de llamar a funciones y usar tipos de datos nativos de la plataforma.
- Dicho lenguaje se pondrá en práctica haciendo uso de Visual Studio como IDE al permitir la creación de aplicaciones de alta calidad con gran rapidez, seguridad y confiabilidad.

Conclusiones Parciales

El análisis de las características y funcionamiento de algunos de los AFIS creados a nivel mundial, así como las soluciones existentes en el país, demostraron la necesidad de que se desarrolle un módulo de clasificación de huellas dactilares. La selección de Visual Paradigm como herramienta CASE para modelar UML, C# como lenguaje de programación y Visual Studio como IDE de desarrollo, para apoyar el ciclo de vida de un software guiado por un enfoque ágil con la metodología XP, demostró las potencialidades de todas estas tecnologías, en aras de lograr la creación del módulo propuesto en la presente investigación de acuerdo a sus necesidades específicas.

CAPÍTULO II: CARACTERÍSTICAS DE LA SOLUCIÓN PROPUESTA.

Introducción

En el presente capítulo se especifican las principales características propias de la solución informática a desarrollar. Se define el modelo de dominio de la aplicación con el propósito de englobar los principales conceptos con los que trabaja el componente. Se realiza la captura de los requisitos funcionales y no funcionales, además de conformarse las historias de usuario relacionadas a estos. En correspondencia con la etapa de planificación del software se puntualizan el plan de entregas, el de iteraciones y las tareas ingenieriles. Se describen la arquitectura candidata del módulo, los patrones de diseño a aplicar y como unos de los artefactos más importantes del diseño se exponen el diagrama de clases del diseño y las tarjetas CRC correspondientes a cada una de las clases definidas en él.

2.1. Descripción del problema

El sistema SourceAFIS existente en el CISED, realiza el proceso de búsqueda y cotejo de una huella dactilar de manera centralizada, por lo que sus tiempos de respuesta y rendimiento son insatisfactorios. El módulo a desarrollar clasificará una huella dactilar de entrada a partir de la ubicación de sus puntos característicos y la orientación de su imagen. El criterio de clasificación resultante podrá ser utilizado en un posterior fraccionamiento del banco de datos de cualquier AFIS al que se integre.

2.2. Modelo de dominio

La presente propuesta tendrá sus bases en un modelo de dominio o modelo conceptual[5], debido a que no se tienen claros los procesos del negocio. Este modelo permitirá obtener una mejor comprensión del entorno del módulo, mostrando los principales conceptos con los que trabaja, que a su vez describen las clases más importantes dentro del contexto de la aplicación.

Se definen como conceptos o clases fundamentales a:

- Usuario: Encargado de clasificar la huella dactilar tras haberla insertado en formato de imagen al componente.
- Imagen_Huella_Dactilar: Imagen que contiene la huella dactilar que se desea clasificar.
- Huella dactilar: Matriz de dimensiones $ancho * alto$ de la imagen cargada, derivada de su mapa de bits, donde a cada posición (i, j) se le va a asignar el valor de la intensidad del pixel análogo en la imagen original. La relación entre este concepto y el de Imagen_Huella_Dactilar se define con una multiplicidad uno a muchos, puesto que en la solución a desarrollar se considera que una

[5] Modelo de dominio: A este tipo de esquema se recurre cuando no se logran determinar los procesos del negocio con fronteras bien establecidas.

imagen dada solo puede tener una huella dactilar, mientras que una huella determinada puede tener varias imágenes.

- Punto característico: elemento que caracteriza una huella dactilar, ya sea un núcleo, un delta o un espiral. En una huella dactilar puede que no exista ningún punto característico, o de lo contrario estar presente uno o varios, en dependencia de su clasificación específica.

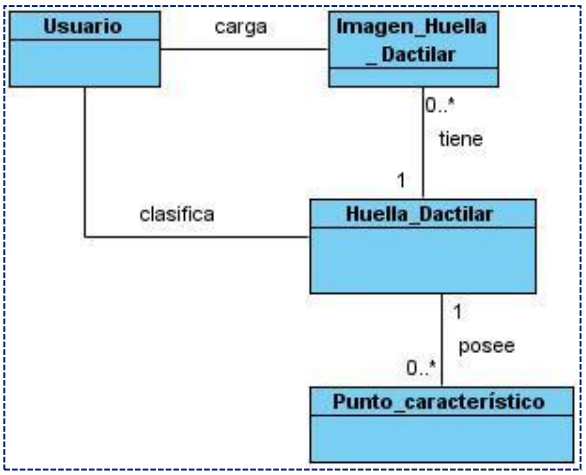

Figura 2.1 Modelo de dominio.

2.3. Sub-procesos claves de la solución

Para realizar la clasificación de huellas dactilares se requiere la ejecución de varios subprocesos secuenciales que pre-procesan la imagen y detectan los puntos característicos, a partir de los que se define el tipo de huella dactilar. Se iniciará el proceso con una imagen de la huella dactilar, discretizada en niveles de gris de 0 (negro) a 255 (blanco).

- **Pre-procesamiento de la imagen:**

Aunque el algoritmo de Kalle Karu y Anil K. Jain, detalle el procedimiento correspondiente a los subprocesos de la etapa de pre-procesamiento de la imagen, en el caso de esta propuesta de solución se utilizarán otros algoritmos para obtener resultados más robustos.

  - Segmentación de la imagen:

La segmentación es el proceso de separar o distinguir las regiones de primer plano de la imagen, de las zonas que pertenecen al fondo de la misma. Las regiones de primer plano corresponden al área donde está definida la huella, por lo que representan la zona de interés. El fondo de la imagen engloba la región fuera de los bordes del área de la huella dactilar, que no contiene ninguna información válida de ésta. Cuando los algoritmos de extracción de minucias o puntos característicos, son aplicados a las regiones del fondo de una imagen, resultan en una extracción e identificación de falsas minucias o puntos característicos, por ello la segmentación es empleada para descartar estas regiones de fondo, lo cual tributa a la fiabilidad de estos procesos.

En la imagen de una huella dactilar, las regiones del fondo generalmente muestran un valor muy bajo de la varianza de la escala de grises, mientras que las que corresponden a la zona donde está definida la huella tienen una varianza muy alta, así un método basado en la varianza y un umbral definido puede ser aplicado para realizar la segmentación (42). El umbral en este caso va a ser la media global de la imagen.

La imagen es dividida en bloques de dimensiones $9\,x\,9$, y se calcula la varianza de los niveles de gris para cada uno. Si la varianza es menor que la media global el bloque forma parte del fondo de la imagen, de lo contrario, se asigna a la región de interés. La varianza del nivel de gris para cada bloque es definida como:

$$V(k) = \frac{1}{W^2}\sum_{i=0}^{W-1}\sum_{j=0}^{W-1}(I_{(i,j)} - M_{(k)})^2$$

donde $V_{(k)}$ es la varianza para el bloque k, $I_{(i,j)}$ es el valor del nivel de gris del pixel (i,j), $M_{(k)}$ es el valor de la media del nivel de gris del bloque k y W es la dimensión del bloque que en este caso es 9.

- Orientación de la imagen:

La imagen direccional de una huella dactilar es una matriz del mismo tamaño que la imagen de la huella, donde el elemento (i,j) contiene la dirección de las crestas papilares al pasar por el pixel (i,j). El gradiente de un vector indica la dirección de máxima variación de f en el punto (x,y), es conocido que en forma analógica este operador tiene la siguiente expresión:

$$\nabla f_{(x,y)} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}$$

Para calcular la orientación se divide la imagen en bloques de $9\,x\,9$. En el caso particular de esta propuesta con el objetivo de optimizar el tiempo de ejecución, en este subproceso no se van a analizar los píxeles ubicados en bloques que pertenecen a la región del fondo de la imagen.

Luego para cada pixel (p,q) en cada bloque, se calcula $\partial_x(p,q)$ y $\partial_y(p,q)$, que son las magnitudes del gradiente en la dirección x y y respectivamente. El gradiente en la dirección x se calcula usando el operador horizontal de Sobel y el de la dirección y con el vertical:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Operador horizontal de Sobel

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Operador vertical de Sobel

Se calcula:

$$V_x(i,j) = \sum_{p=i-\frac{S}{2}}^{i+\frac{S}{2}}\sum_{q=j-\frac{S}{2}}^{j+\frac{S}{2}} 2\partial_x(p,q)\,\partial_y(p,q)$$

$$V_y(i,j) = \sum_{p=i-\frac{S}{2}}^{i+\frac{S}{2}}\sum_{q=j-\frac{S}{2}}^{j+\frac{S}{2}} {\partial_x}^2(p,q) - {\partial_y}^2(p,q)$$

Posteriormente la orientación local para cada bloque centrado en el pixel (i,j) se calcula como:

$$\theta(i,j) = \frac{1}{2}\tan^{-1}\frac{V_y(i,j)}{V_x(i,j)}$$

Para obtener una orientación que se ajuste mejor al recorrido de las crestas papilares, se suaviza la imagen direccional obtenida aplicando un filtro Gaussiano de paso bajo. A partir de $\theta(i,j)$ se crea un vector definido por:

$$\varphi_x(i,j) = \cos(2\theta(i,j)) \qquad \varphi_y(i,j) = \sin(2\theta(i,j))$$

donde φ_x y φ_y son las componentes x y y del vector, respectivamente. El suavizado Gaussiano se define como:

$$\varphi'_x(i,j) = \sum_{p=-\frac{S_\varphi}{2}}^{\frac{S_\varphi}{2}} \sum_{q=-\frac{S_\varphi}{2}}^{\frac{S_\varphi}{2}} G(p,q)\varphi_x(i-ps, j-qs) \qquad \varphi'_y(i,j) = \sum_{p=-\frac{S_\varphi}{2}}^{\frac{S_\varphi}{2}} \sum_{q=-\frac{S_\varphi}{2}}^{\frac{S_\varphi}{2}} G(p,q)\varphi_y(i-ps, j-qs)$$

donde G es un filtro de paso bajo de dimensiones $S_\varphi \; X \; S_\varphi$, que en este caso son $5 \; X \; 5$. (43)

Finalmente el valor del campo de orientación del pixel (i,j) es:

$$O(i,j) = \frac{1}{2}\tan^{-1}\frac{\varphi'_y(i,j)}{\varphi'_x(i,j)}$$

- Imagen Direccional por bloques:

Para homogeneizar la imagen y hacer una representación de las direcciones se divide la misma en bloques de tamaño $9 \; x \; 9$. Se le asigna a cada bloque una única dirección que se calcula mediante un promedio de ángulos dobles: si α grados es el ángulo de la dirección de un pixel, éste se transforma en un vector unitario en la dirección $v = (\cos 2\alpha \, , \sin 2\alpha)$.

Se denota por (x,y) al vector resultante de promediar todos los vectores unitarios asociados a los píxeles del bloque. Luego la dirección de cada bloque estará dada por $d = \frac{1}{2}\arctan(\frac{y}{x})$ (21).

- **Detección de puntos característicos:**

El cálculo del índice de Poincaré para cada bloque permite determinar cuales corresponden a regiones donde existen puntos característicos. El índice de Poincaré del bloque (i,j), denotado por $P(i,j)$, se define como la rotación total de los vectores de las direcciones de los bloques que rodean al bloque (i,j) (9):

$$P(i,j) = \sum_{k=0\ldots7} ángulo(d_k, d_{k+1})$$

En el caso particular de la solución que se propone, el cálculo de estos ángulos requiere que a cada dirección se le asigne uno de los dos ángulos posibles, entre -180º y 180º, de manera que el valor absoluto de la diferencia entre d_k y d_{k+1} sea menor o igual que 90º. En curvas cerradas el índice de

Poincaré sólo puede tomar los valores 0º, ±180º y ±360º. En el caso de las singularidades de las huellas dactilares:

$$P(i,j) = \begin{Bmatrix} 0° \text{ si en } [i,j] \text{ no existe ningún punto característico} \\ 360° \text{si en } [i,j] \text{ existe un punto característico de tipo espiral} \\ 180° \text{si en } [i,j] \text{ existe un punto característico de tipo núcleo} \\ -180° \text{si en } [i,j] \text{ existe un punto característico de tipo delta} \end{Bmatrix}$$

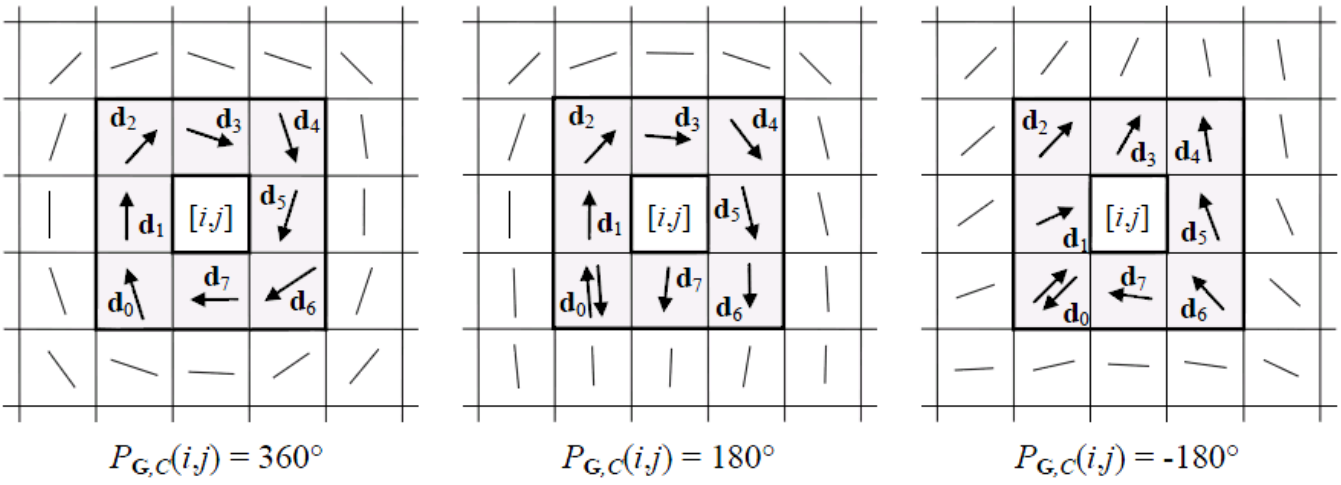

Figura 2.2 Ejemplos del cálculo del índice de Poincaré.

En algunos casos el índice de Poincaré puede detectar falsos puntos característicos, que impiden una correcta clasificación de la huella dactilar. Karu y Jain plantean que si existe un punto característico, usualmente éste no está localizado en un único bloque, por lo que existen bloques contiguos con igual índice. Es por ello que estos bloques con similar comportamiento se deben tratar como uno solo.

- **Clasificación de la huella dactilar:**

Después de localizar todos los puntos característicos se clasifica la imagen de la huella dactilar basada en el número y posiciones de estos puntos. Una huella dactilar tipo arco, no contiene ningún punto característico, en el caso de las de tipo lazos y las de arco tendido, contienen un núcleo y un delta, y las de tipo espiral presentan dos núcleos o un espiral, y dos deltas.

Para diferenciar entre las huellas dactilares de tipo arco tendido y las de tipo lazo se traza una línea recta que conecte al delta y al núcleo En una imagen de arco tendido la orientación de esta línea es a lo largo de la dirección local de los vectores, mientras que en un lazo la línea la intersecta transversalmente. Se calcula el siguiente promedio:

$$\frac{1}{n}\sum_{i=1}^{n} \sin(\alpha_i - \beta)$$

donde β es la pendiente de la línea de conexión, y $\alpha_1, \alpha_2, \ldots, \alpha_n$ son las direcciones de los bloques ubicados en este segmento de línea. Si el resultado de esta operación es menor que un umbral que

inicialmente se puede definir como 0.2, pero que puede variar de acuerdo a los resultados que se obtengan en las pruebas continuas que se le realicen al componente, la huella es de tipo arco tendido (21).

Si este promedio es mayor que el umbral definido, las huellas de tipo lazo izquierdo se van a distinguir de las de lazo derecho de acuerdo al valor de la pendiente. Teniendo en cuenta que el recorrido para analizar la imagen de la huella dactilar, comience por la esquina superior izquierda de ésta, que va a representar el pixel en la posición $(0,0)$, y a medida que se desplace en las filas, el valor de la coordenada y va a aumentar, entonces el valor de la ordenada del núcleo va a ser menor que el de la ordenada del delta, por ello si $\beta < 0$ la huella es de tipo lazo derecho, sino si $\beta > 0$ entonces es de tipo lazo izquierdo.

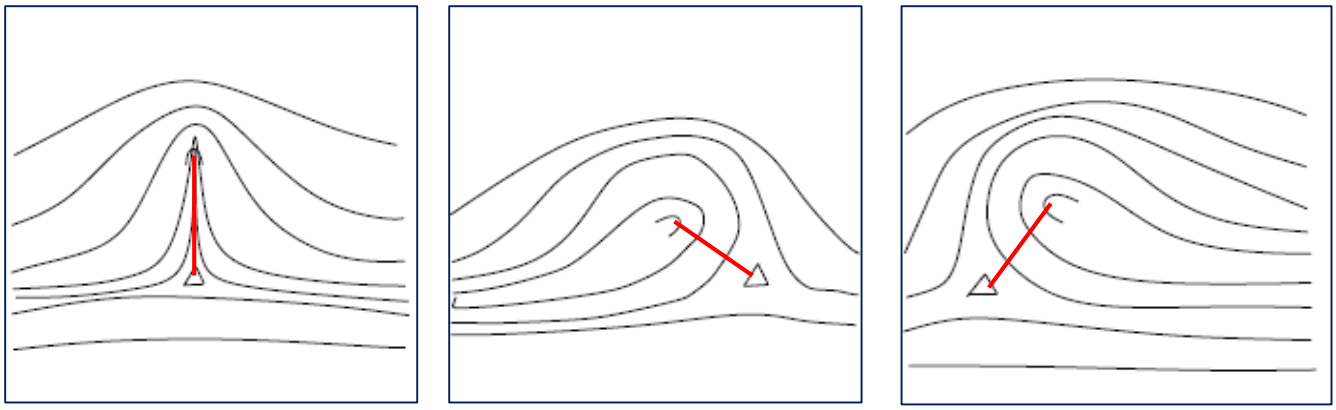

Figura 2.3 Línea entre los puntos característicos en diferentes tipos de huellas dactilares.

2.4. Captura de requisitos

La captura de los requisitos funcionales y no funcionales es uno de los procedimientos más importantes para el desarrollo de cualquier software. En esta etapa el principal objetivo es identificar lo que el cliente realmente necesita, de manera que el equipo de desarrollo pueda trabajar en base a estos requerimientos y logre obtener un producto que funcione con la calidad deseada.

2.4.1. Requisitos funcionales

Los requisitos funcionales que describen las funcionalidades que el componente debe cumplir son:

RF-1. Leer la imagen de la huella dactilar.

RF-2. Pre-procesar la imagen de la huella dactilar.

a) Segmentar la imagen.
b) Obtener la imagen direccional por píxeles.
c) Suavizar la orientación de la imagen obtenida en el subproceso anterior.
d) Obtener la imagen direccional por bloques.

RF-3. Extracción de los puntos característicos.

a) Calcular el índice de Poincaré.
b) Detectar los puntos característicos.

c) Extraer la posición de los puntos característicos.

RF-4. Clasificar la imagen de la huella dactilar.

2.4.2. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que hacen al producto atractivo, usable, rápido y confiable. Se definieron como requisitos no funcionales:

- **Software:**

RNF-1. El Sistema Operativo que debe estar instalado en la estación de trabajo en que puede ejecutarse el componente es Windows.

RNF-2. Framework de desarrollo .NET 4.0.

- **Hardware:**

RNF-3. Periféricos: Mouse.

RNF-4. Procesador Intel Pentium 4 o superior.

RNF-5. 1 GB o más de memoria RAM.

RNF-6. CPU 3 GHZ o superior.

- **Restricciones en el Diseño y en la Implementación:**

RNF-7. La implementación del componente debe ser desarrollada en el lenguaje C#, puesto que el sistema al que se debe integrar está codificado en dicho lenguaje.

2.5. Historias de Usuario (HU)

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, representando una breve descripción del comportamiento del mismo, con el empleo de una terminología del cliente sin lenguaje técnico. Se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, y presiden la creación de las pruebas de aceptación. Deben tener como características fundamentales: ser independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables (29). Las estimaciones de esfuerzo asociado a la implementación de las historias de usuario la establecen los programadores utilizando como medida el punto, que equivale a una semana ideal de programación.

A continuación se definen las historias de usuario que tienen mayor impacto en la solución propuesta, el resto está definido en los anexos. (Ver Anexo 2)

Historia de Usuario	
Número: HU_2	**Usuario:** Alicia Delgado Hernández
Nombre de historia: Pre-procesar imagen de la huella dactilar.	

Prioridad en negocio: Muy Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 6	**Iteración asignada:** 2
Programador responsable: Alicia Delgado Hernández	
Descripción: El componente debe pre-procesar la imagen de la huella dactilar cargada, con el propósito de obtener como resultado final la imagen direccional por bloques necesaria para los procesos posteriores a éste. El pre-procesamiento estará enmarcado en la segmentación de la imagen, el cálculo de la imagen direccional en píxeles y en bloques.	
Observaciones: ----------	

Tabla 2.1 HU Pre-procesar imagen de la huella dactilar.

Historia de Usuario	
Número: HU_3	**Usuario:** Alicia Delgado Hernández
Nombre de historia: Extraer puntos característicos.	
Prioridad en negocio: Muy Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Alicia Delgado Hernández	
Descripción: El componente a través del cálculo del índice de Poincaré para cada bloque, debe detectar y extraer los puntos característicos de la huella dactilar.	
Observaciones: ----------	

Tabla 2.2 HU Extracción de los puntos característicos.

Historia de Usuario	
Número: HU_4	**Usuario:** Alicia Delgado Hernández
Nombre de historia: Clasificar huella dactilar.	
Prioridad en negocio: Muy Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Alicia Delgado Hernández	

Descripción: El componente debe clasificar la huella dactilar a partir de la posición y orientación de sus puntos característicos.
Observaciones: Si la cantidad de puntos característicos identificados o la posición de estos no son válidas, el componente asignará como patrón de clasificación “No definido”.

Tabla 2.3 HU Clasificar huella dactilar.

2.6. Planificación

2.6.1. Plan de entregas

En esta fase el cliente establece la prioridad de cada historia de usuario, y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente (29).

El equipo de desarrollo mantiene un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de los puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La velocidad del proyecto es utilizada para establecer el número de historias de usuario que se pueden implementar antes de una fecha determinada o el tiempo que tomará implementar un conjunto de historias. Se propone el siguiente plan de entregas para la solución propuesta:

Entregable	Iteración	Fin de Iteración
Leer la imagen	1	Enero de 2013
Imagen pre-procesada	2	Marzo de 2013
Detección de los puntos característicos	3	Abril de 2013
Clasificación de la huella dactilar	3	Mayo de 2013

Tabla 2.4 Plan de entregas.

2.6.2. Plan de Iteraciones

El ciclo de desarrollo de software guiado por XP se caracteriza por ser iterativo e incremental, por lo que se realizan varias iteraciones sobre el sistema antes de su fase de producción.

Los elementos que deben tomarse en cuenta durante la elaboración del plan de iteraciones son las historias de usuario no abordadas, la velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero realizadas por parejas de programadores (29).

El componente de clasificación de huellas dactilares se realizará en 3 iteraciones, durante las cuales se codificarán incrementalmente las cuatro historias de usuario definidas.

Iteración	Historia de Usuario	Semanas estimadas
1	Leer la imagen	2
	Imagen pre-procesada	

2	Imagen pre-procesada	5
3	Detección de los puntos característicos	5
	Clasificación de la huella dactilar	

Tabla 2.5 Plan de iteraciones.

En la primera iteración se priorizarán las dos historias de usuario de las cuales depende el resto del funcionamiento del componente. Por la complejidad y cantidad de subprocesos que requiere el pre-procesamiento de la imagen, se comienza la implementación del mismo en la primera iteración, y se extiende hasta la segunda.

Las tareas de programación o ingenieriles a realizarse en cada una de estas iteraciones se especifican en la próxima tabla. En los anexos se detallan los elementos y descripción de cada una de estas tareas. (Ver Anexo 3)

Iteración	Historia de Usuario	Tarea de Ingeniería
1	Leer imagen de huella dactilar.	• Leer desde una dirección especificada por el usuario, la imagen de la huella dactilar.
	Pre-procesar imagen de la huella dactilar.	• Determinar la región de interés de la huella dactilar.
2	Pre-procesar imagen de la huella dactilar.	• Optimizar la región de interés obtenida. • Calcular la orientación de cada pixel de la imagen aplicando el gradiente de Sobel. • Suavizar la orientación obtenida aplicando un filtro gaussiano, para obtener una imagen direccional con mayor calidad. • Obtener la imagen direccional por bloques.
3	Extraer los puntos característicos.	• Cálculo del índice de Poincaré para cada bloque. • Detección de los puntos característicos. • Descartar falsos puntos característicos.
	Clasificar la huella dactilar.	• Analizar la cantidad de puntos característicos de cada tipo que fueron detectados y calcular su posición (x, y), que equivale a (columna, fila). • Clasificación final de la huella dactilar.

Tabla 2.6 Distribución de tareas de ingeniería por iteración.

2.7. Diseño

El papel del diseño en el ciclo de vida del software es adquirir una comprensión de su funcionamiento, además de crear una entrada apropiada o arquitectura de software, que constituya el punto de partida

para las actividades de implementación, dando soporte a los requisitos funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables y sistemas operativos, que debe poseer la aplicación. En este punto del ciclo de desarrollo del software es cuando se descomponen los trabajos de implementación en partes más manejables que puedan ser realizadas por diferentes equipos de desarrollo.

2.7.1. Descripción de la arquitectura

Según el director del Proceso de Desarrollo (RUP) Philippe Kruchten: "*La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.*" (44)

La definición oficial de Arquitectura de Software, también denominada como Arquitectura Lógica es la que brinda el documento de IEEE Std 1471-2000, adoptada también por Microsoft, la misma plantea que: *"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución"*. (45)

O sea, la arquitectura de software es, a grandes rasgos, una vista que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del entorno y las formas en que interactúan y se coordinan para alcanzar la misión principal. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.

Un estilo es un concepto descriptivo que define una forma de articulación u organización arquitectónica. El conjunto de los estilos cataloga las formas básicas posibles de estructuras de software, mientras que las formas complejas se articulan mediante una composición de los estilos fundamentales (46). A estos se les confiere una gran importancia en el proceso de desarrollo de software debido a que:

- Sirven para sintetizar y tener un lenguaje que describa la estructura de las soluciones.
- Pocos estilos abstractos encapsulan una enorme variedad de configuraciones concretas.
- Definen los patrones posibles de las aplicaciones.
- Permiten evaluar arquitecturas alternativas con ventajas y desventajas conocidas ante diferentes conjuntos de requerimientos no funcionales.

En el presente trabajo de diploma se propone como arquitectura base para la organización estructural del módulo, una arquitectura en capas.

Los sistemas o arquitecturas en capas constituyen una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior (47).

Se basa en una distribución jerárquica de roles y responsabilidades para proporcionar una división de los problemas a resolver. Los roles indican el tipo y la forma de interacción con otras capas, y las responsabilidades la funcionalidad que implementan. De manera general en algunos ejemplos, las capas internas están ocultas a todas las demás, menos para las capas externas adyacentes, y excepto para funciones puntuales de exportación, mientras que en otros sistemas, las capas pueden ser sólo parcialmente opacas. En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas.

La arquitectura propuesta para el desarrollo del componente de clasificación de huellas dactilares identifica como capas:

- **Capa Presentación:** representa la interfaz gráfica de prueba para la interacción con el módulo. A partir de varias opciones, muestra los subprocesos que se realizan hasta obtener finalmente la clasificación de la huella dactilar.
- **Capa de Negocio:** refleja la lógica del sistema representando la organización de las clases y relaciones entre estas.

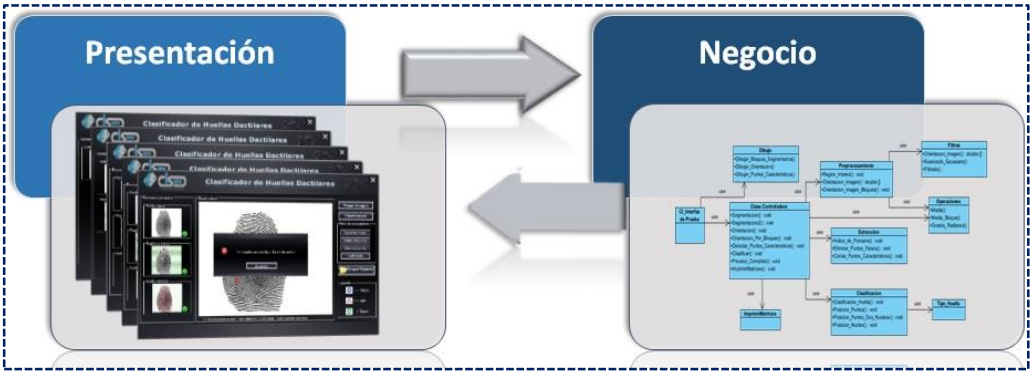

Figura 2.4 Arquitectura de software.

No obstante, aunque se utilice este patrón arquitectónico, se propone el uso de forma secundaria del estilo tubería y filtros, con el propósito de organizar la infraestructura de comunicación entre los sub-procesos que componen la etapa de pre-procesamiento y extracción de características de la huella dactilar.

Una tubería (pipeline) es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Debido a su simplicidad y su facilidad para captar una funcionalidad, es una arquitectura mascota cada vez que se trata de demostrar ideas sobre la formalización del espacio de diseño arquitectónico, igual que el tipo de datos pila lo fue en las especificaciones algebraicas o en los tipos de datos abstractos (48).

Se utiliza dicho patrón puesto que proporciona una estructura para sistemas que procesan un flujo de datos como es el caso. El proceso puede ser descompuesto en etapas sucesivas, donde cada una es

incremental respecto a la anterior. Cada paso del procesamiento está encapsulado en un filtro. El dato pasa a través de la tubería entre los filtros adyacentes.

Las ventajas que ofrece este estilo arquitectónico se enmarcan en:

- Es simple de entender e implementar, es posible implementar procesos complejos con editores gráficos de líneas de tuberías o con comandos de línea.
- Fuerza un procesamiento secuencial.
- Es fácil de envolver en una transacción atómica.
- Los filtros se pueden empaquetar y hacer paralelos o distribuidos.

El flujo propuesto tiene como entrada la imagen de la huella dactilar que se desea clasificar, y como salida sus puntos característicos, a partir de los cuales se va a realizar la clasificación de la misma. Los sub-procesos o filtros identificados son:

1. Segmentación de la imagen.
2. Obtención de la imagen direccional por píxeles.
3. Suavizado de la imagen direccional obtenida.
4. Creación de la imagen direccional por bloques.
5. Detección y extracción de los puntos característicos.

En el siguiente esquema se muestran los resultados visuales de la aplicación de este estilo arquitectónico. En un primer plano, de izquierda a derecha, se encuentra la imagen de la huella dactilar que se desea clasificar, posteriormente aparece delimitada la región de interés de la huella dactilar resultante de aplicar el filtro 1. Luego se encuentra una figura que representa lo que sería el resultado visual de aplicar los filtros 2, 3 y 4. Por último se expone la imagen de la huella dactilar indicando la presencia de los puntos característicos presentes en ella.

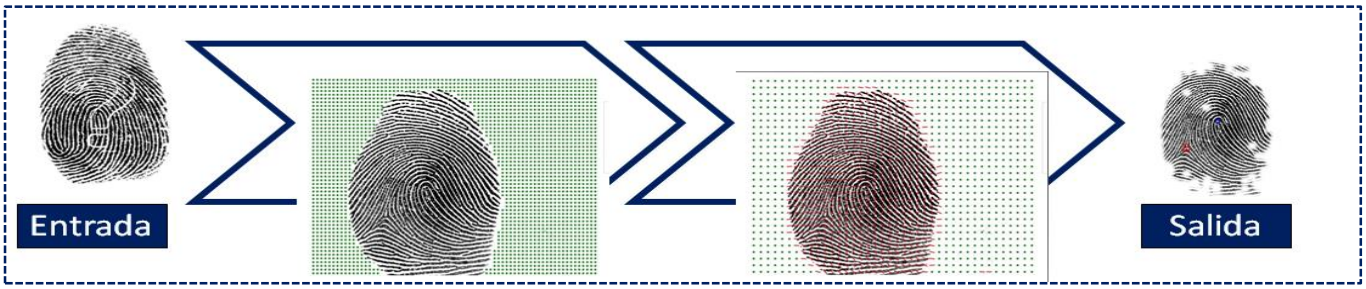

Figura 2.5 Estilo arquitectónico filtros y tuberías.

2.7.2. Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. El mismo identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Estos modelos que se presentan como parejas de problema / solución con un nombre, codifican buenos principios y sugerencias relacionados

con la asignación de responsabilidades, basados en la recopilación del conocimiento de los expertos en desarrollo de software (49).

En resumen los patrones de diseño se caracterizan por:

- Ser soluciones concretas.
- Ser soluciones técnicas.
- Se utilizan en situaciones frecuentes.
- Favorecen la reutilización de código.
- Su uso no se refleja en el código.
- Es difícil reutilizar la implementación de un patrón.

Los patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades) describen los principios fundamentales de la asignación de responsabilidades a objetos (50). El nombre se eligió para indicar la importancia de captar (grasping) estos principios, si se desea diseñar un software orientado a objetos con calidad. En el diseño de la aplicación propuesta se utilizaron los patrones GRASP experto, bajo acoplamiento y alta cohesión.

Patrón Experto: se asigna una responsabilidad a la clase que tiene la información necesaria para cumplirla, alentando con ello definiciones de clases "sencillas" y más cohesivas que son más fáciles de comprender y de mantener (50). Así se brinda soporte a una alta cohesión. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto soporta también un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. En el componente de clasificación de huellas dactilares este patrón se evidencia entre las clases ***Controladora*** y la clase ***Extraccion***, en el momento de detectar los puntos característicos de la huella dactilar, puesto que la última es la que cuenta con los algoritmos necesarios para ejecutar este subproceso.

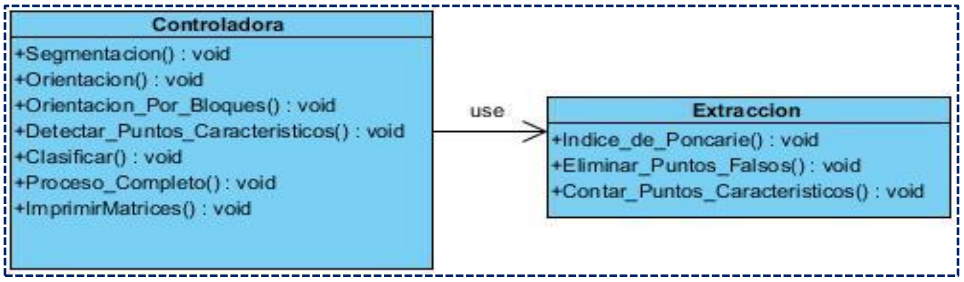

Figura 2.6 Patrón experto.

Patrón Bajo Acoplamiento: el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo o débil acoplamiento no depende de muchas otras (50). Por ejemplo en el componente de clasificación de huellas dactilares se hace uso de este patrón en la clase ***Clasificacion*** para la cual es totalmente transparente todo el procesamiento que se realiza antes de llegar a esta etapa, por ello solo se relaciona con la clase

TipoHuella, que contiene las diferentes clasificaciones de huella que pueden ser asignadas. Los beneficios que tiene el uso de este patrón es que las clases no se afectan por cambios de otras clases, son fáciles de entender por separado, además de ser fáciles de reutilizar.

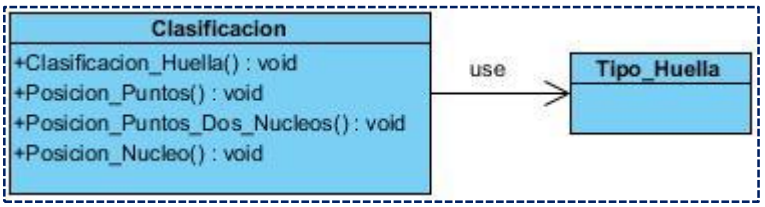

Figura 2.7 Patrón bajo acoplamiento.

Patrón Alta Cohesión: En la perspectiva del diseño orientado a objetos, la cohesión (o más exactamente, la cohesión funcional) es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase (50). Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. En el diseño del componente de clasificación de huellas dactilares, como ejemplo de la aplicación de este patrón se encuentra la relación entre las clases ***Controladora***, ***Preprocesamiento*** y ***Operacion***, durante la obtención de la imagen direccional de la huella dactilar.

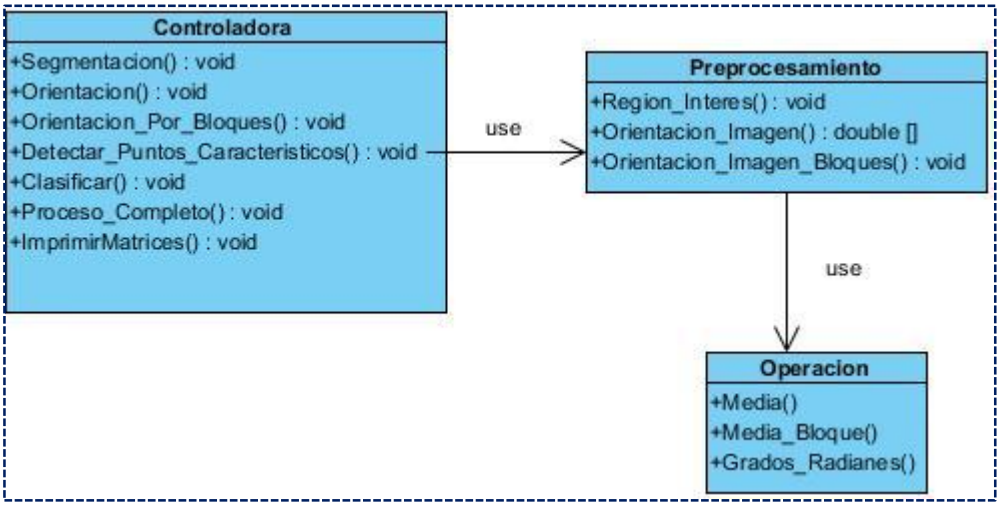

Figura 2.8 Patrón alta cohesión.

2.7.3. Diagrama de clases del diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos, puesto que muestran un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Se utilizan para modelar la vista del diseño estática del software, especificar y documentar modelos estructurales, y para construir sistemas ejecutables, aplicando ingeniería directa e inversa.

La estructura que va a guiar el desarrollo del componente de clasificación de huellas dactilares estará basada en el siguiente diagrama de clases del diseño, que define todas las clases y relaciones, que en su conjunto implementan cada uno de los subprocesos identificados en las historias de usuario.

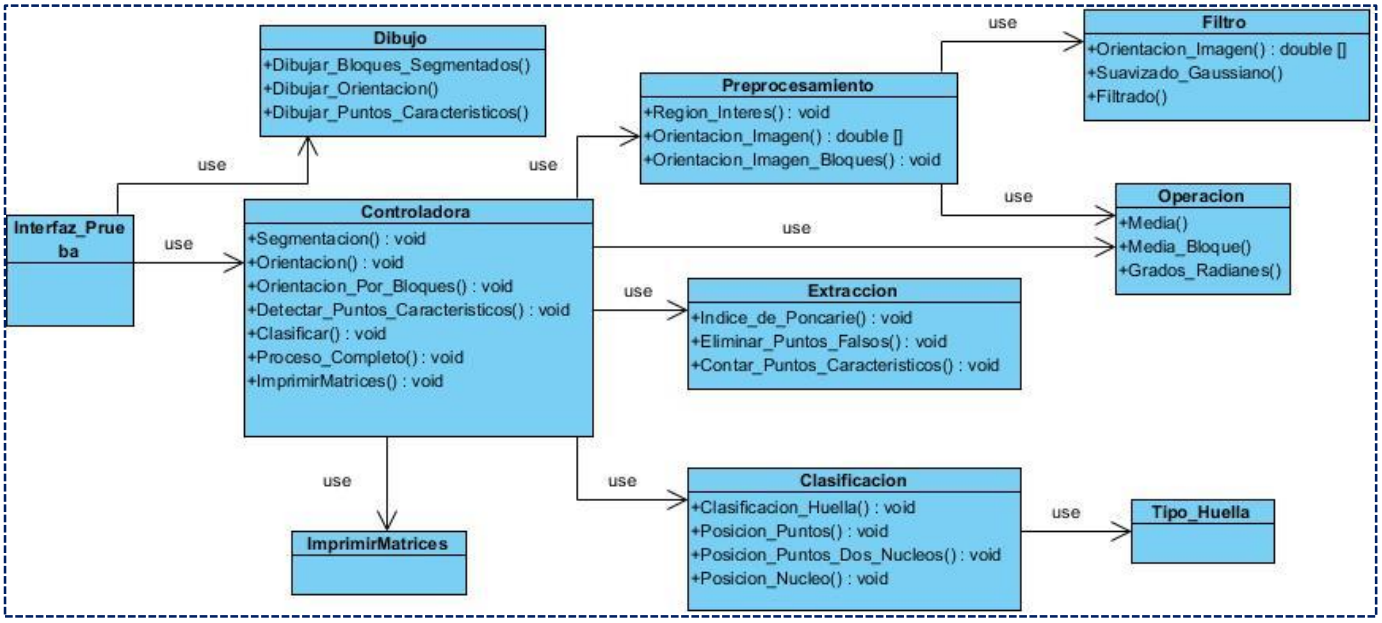

Figura 2.9 Diagrama de clases del diseño.

De manera general se definen tres niveles de clases, de acuerdo a la responsabilidad que realizan:

- **Nivel 1 Clase Controladora:** representa la clase controladora responsable de guiar todos los subprocesos de manera secuencial, hasta obtener el resultado final.
- **Nivel 2 Clases del Proceso:** incluye las clases **Preprocesamiento**, **Extraccion** y **Clasificacion**, encargadas de realizar por separado cada subproceso, y entre las cuales no existe ninguna relación directa.
- **Nivel 3 Auxiliar:** encierra el resto de las clases que implementan funcionalidades secundarias de las clases ubicadas en los dos niveles anteriores.

2.7.4. Tarjetas CRC

Las tarjetas CRC (Clase - Responsabilidad - Colaboración), constituyen uno de los artefactos de la metodología XP que guía el proceso de desarrollo de la solución propuesta. Se dividen en tres secciones que contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores.

Una clase es cualquier persona, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son las funcionalidades que realiza y sus atributos. Los colaboradores son las demás clases con las que interactúa con el fin de cumplir sus responsabilidades. Seguidamente se exponen las tarjetas CRC

correspondientes a las clases que pertenecen a los dos primeros niveles especificados anteriormente, el resto se muestra en los anexos. (Ver Anexo 4)

Clase Controladora	
Responsabilidades	**Colaboradores**
• Realizar la segmentación de la imagen.	Preprocesamiento
• Obtener la imagen direccional por píxeles.	Preprocesamiento
• Obtener la imagen direccional por bloques.	Preprocesamiento
• Detectar puntos característicos.	Extraccion
• Clasificar huella dactilar.	Clasificacion
• Imprimir matrices con los datos obtenidos.	ImprimirMatrices

Tabla 2.7 Tarjeta CRC Controladora.

Clase Preprocesamiento	
Responsabilidades	**Colaboradores**
• Definir la región de interés.	Operaciones, InteligDebug.dll
• Obtener la imagen direccional por píxeles.	Sobel_Orientación
• Obtener la imagen direccional por bloques.	----------

Tabla 2.8 Tarjeta CRC Preprocesamiento.

Clase Extraccion	
Responsabilidades	**Colaboradores**
• Calcular el índice de Poincaré para cada bloque.	----------
• Eliminar falsos puntos característicos.	----------
• Contar la cantidad de puntos característicos que tiene la huella dactilar luego de haber eliminado los falsos.	----------

Tabla 2.9 Tarjeta CRC Extraccion.

Clase Clasificacion	
Responsabilidades	**Colaboradores**
• Clasificar las huellas dactilares.	TipoHuella
• Hallar la posición (x, y) de cada punto característico de la huella, para poder clasificarla.	----------

Tabla 2.10 Tarjeta CRC Clasificacion

Conclusiones Parciales

La definición del modelo de dominio y la explicación detallada del proceso que debe ejecutar el componente, permitieron identificar con mayor facilidad los requisitos funcionales que éste debe satisfacer, propiciando así la comprensión de su funcionamiento. Las historias de usuario especificadas para cada uno de estos requisitos sin emplear lenguaje técnico y sí términos del cliente, detallando entre otros elementos el tiempo que requieren para su codificación, favorecieron la estimación del plan de entregas del producto delimitando el propósito y duración de cada iteración.

Con el uso de la arquitectura global en capas especificada, el estilo arquitectónico tuberías y filtros, y algunos de los patrones GRASP, se logró organizar la vista lógica de la solución, de manera que las clases identificadas y sus relaciones sean las bases para su implementación.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBAS.

Introducción

En este capítulo se exponen todas las particularidades de la implementación del componente, en aras de obtener un producto final que esté en correspondencia con los requisitos definidos por el cliente, lo cual se validará aplicando diferentes tipos de prueba que propone la metodología de desarrollo de software utilizada. Para ello se definen los estándares de codificación que debe cumplir el equipo de desarrollo, se crea el diagrama de componentes y el de despliegue, además de presentarse las interfaces de pruebas creadas para comprobar el funcionamiento de la solución. A partir del código resultante y su funcionamiento se ejecutan las pruebas unitarias y de aceptación del módulo.

3.1. Implementación

Una vez definidas las historias de usuario y concluido el diseño se pasa a la etapa de codificación de la solución propuesta cuyos objetivos van destinados a desarrollar de forma iterativa e incremental un producto completo que esté preparado para la transición a su comunidad de usuarios, consiguiendo versiones útiles (alfa, beta y otras entregas de prueba) de forma rápida y práctica, que paulatinamente completen la planeación, diseño, desarrollo y prueba de toda la funcionalidad necesaria.

Durante la codificación la programación debe ser por parejas, con el objeto de crear código para cada historia de usuario en dependencia de lo concebido en el plan de iteraciones y las tareas ingenieriles. Esto da un mecanismo para la solución de problemas en tiempo real y también para el aseguramiento de la calidad en tiempo real. A medida que concluya cada iteración, el código desarrollado se integrará con el resto, puesto que esta estrategia de integración continua ayuda a evitar los problemas de compatibilidad e interfaces, y a descubrir a tiempo los errores.

3.1.1. Estándares de codificación

Entre las prácticas a aplicar durante el proceso de desarrollo de software que propone la metodología XP se encuentran la refactorización del código y la propiedad compartida de éste de forma que todo el personal pueda corregir y extender cualquier parte del producto. Para complementar estas prácticas, la metodología enfatiza en que la comunicación de los programadores es a través del código, por lo cual es indispensable que se sigan ciertos estándares de programación que le provean legibilidad.

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Las técnicas de codificación incorporan muchos aspectos al desarrollo del software, y aunque

generalmente no afectan a la funcionalidad de la aplicación, sí contribuyen a una mejor compresión del código fuente (51).

En la propuesta de solución para declarar el nombre de las variables, métodos y clases se tendrán en cuenta las siguientes convenciones:

- En todos los casos se utilizarán nombres descriptivos que ayuden a una mejor comprensión del código.
- En los nombres, independiente del estilo que se utilice, las palabras van a estar separadas por un guión bajo.
- Para la nomenclatura de las clases y los métodos se utilizará el estilo de capitalización Pascal, con el cual se capitaliza la primera letra de cada palabra. Ejemplo Región_Interes.
- No se utilizará una misma línea para definir más de una variable y siempre que sea posible éstas se inicializarán en su misma línea de declaración.
- Los nombres de las variables serán escritos completamente en minúsculas.

3.1.2. Diagrama de componentes

Durante la codificación de la solución propuesta se pueden distinguir diferentes elementos de implementación que analizados como una sola unidad, constituyen el componente de clasificación de huellas dactilares. Un componente es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación, un conjunto de interfaces y proporciona la realización de los mismos, típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.). Son las piezas reutilizables de alto nivel a partir de las cuales se pueden construir los sistemas.

Los diagramas de componentes modelan la vista estática del software, se representan como un grafo de componentes unidos por medio de relaciones de dependencia, pudiendo mostrarse las interfaces que estos soporten. A continuación se expone el diagrama definido para la solución propuesta:

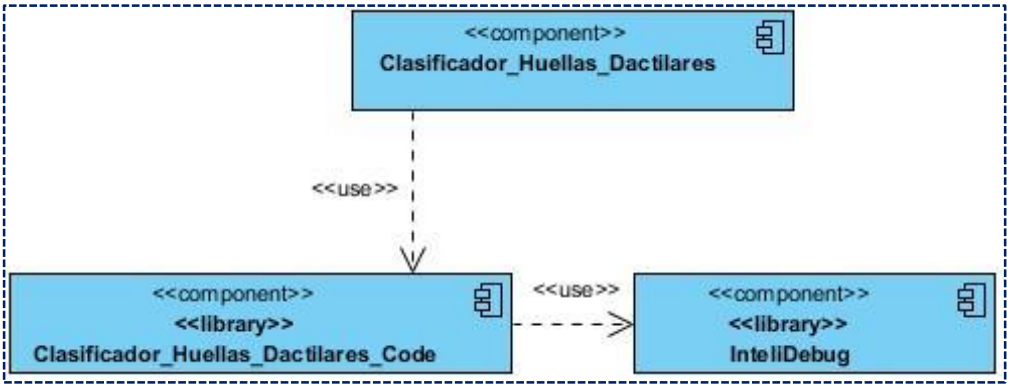

Figura 3.1 Diagrama de componentes.

- **Library Clasificador_Huellas_Dactilares_Code:** librería que incluye todas las clases y relaciones definidas en el diagrama de clases del diseño, a excepción de la clase destinada a la interfaz de pruebas.
- **Library InteliDebug:** librería que provee clases y métodos que apoyan el trabajo con imágenes.

3.1.3. Diagrama de despliegue

El modelo de despliegue es uno de los artefactos obtenidos durante el proceso de desarrollo del software, con el objetivo de capturar los elementos de configuración del procesamiento, las conexiones entre esos elementos y visualizar su distribución en los nodos físicos. De manera general este tipo de modelo está compuesto por:

- **nodos** o elementos de procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos.
- **dispositivos** caracterizados por ser nodos estereotipados sin capacidad de procesamiento en el nivel de abstracción que se modela.
- **conectores** que expresan el tipo de conector o protocolo utilizado entre el resto de los elementos del modelo.

El modelo de despliegue del componente de clasificación de huellas dactilares está compuesto solamente por un nodo, que en este caso es la computadora donde se pueda ejecutar.

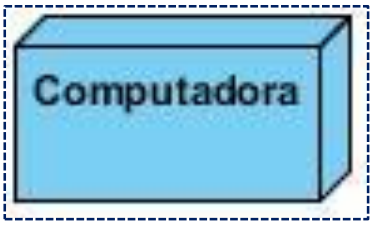

Figura 3.2 Diagrama de despliegue.

3.1.4. Interfaces de usuario

El objetivo del componente creado no es que tenga una interfaz gráfica propia, que visualice todo el proceso de clasificación de la huella dactilar, sino que constituya una librería o módulo que se integre al SourceAFIS. No obstante, para comprobar y visualizar su correcto funcionamiento el equipo de desarrollo definió un conjunto de interfaces de pruebas.

A continuación se expone la interfaz que muestra la clasificación asignada a una huella dactilar determinada. Las interfaces restantes se muestran en los anexos. (Ver Anexo 5)

Figura 3.3 Interfaz que muestra la clasificación de la huella dactilar.

3.2. Pruebas

Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y es hecha una evaluación de algún aspecto determinado. La prueba del software es un elemento crítico para la garantía de su calidad y representa una revisión final de las especificaciones del diseño y de la codificación.

3.2.1 Pruebas unitarias

La creación de pruebas unitarias antes de que comience la codificación es un elemento clave del enfoque de XP. Las pruebas unitarias que se crean deben implementarse con el uso de una estructura que permita automatizarlas, de modo que puedan ejecutarse en reiteradas ocasiones y con facilidad. Esto estimula una estrategia de pruebas de regresión, siempre que se modifique el código. El programador es el encargado de escribir las pruebas unitarias y producir el código del sistema.

Las pruebas de unidad son las que van enfocadas a los elementos más pequeños del software. Son aplicables a funcionalidades para verificar que los flujos de control y de datos están cubiertos, y funcionan como se espera. La prueba de unidad siempre está orientada a caja blanca.

Pruebas de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usan la estructura de control del diseño procedimental para derivarlos, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad.

En el componente creado, se aplicó la técnica de pruebas de caja blanca del camino básico, puesto que permite obtener una medida de la complejidad lógica de un diseño y usarla como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los casos de prueba derivados del camino básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

En el caso del algoritmo diseñado para clasificar la huella dactilar, cuyo código se muestra en los anexos (Ver Anexo 6), la aplicación de la prueba del camino básico quedaría de la siguiente manera:

1. Identificar todos los caminos posibles a partir del código fuente:

Camino 1. (1-2-3-4-5-40)

Camino 2. (1-2-3-4-6-7-40)

Camino 3. (1-2-3-8-9-10-11-12-40)

Camino 4. (1-2-3-8-9-10-11-13-14-15-16-17-18-19-17-20-21-22-40)

Camino 5. (1-2-3-8-9-10-11-13-14-15-16-17-18-19-17-20-21-23-24-40)

Camino 6. (1-2-3-8-9-10-11-13-14-15-16-17-18-19-17-20-21-23-25-40)

Camino 7. (1-2-3-8-9-10-11-13-14-16-17-18-19-17-20-21-22-40)

Camino 8. (1-2-3-8-9-10-11-13-14-16-17-18-19-17-20-21-23-24-40)

Camino 9. (1-2-3-8-9-10-11-13-14-16-17-18-19-17-20-21-23-25-40)

Camino 10.(1-2-3-8-9-26-27-28-29-30-31-32-34-29-35-28-36-37-40)

Camino 11.(1-2-3-8-9-26-27-28-29-30-31-32-34-29-35-28-36-38-40)

Camino 12.(1-2-3-8-9-26-27-28-29-30-31-33-34-29-35-28-36-37-40)

Camino 13.(1-2-3-8-9-26-27-28-29-30-31-33-34-29-35-28-36-38-40)

Camino 14.(1-2-3-8-39-40)

Camino 15.(1-2-40)

2. Se dibuja el grafo de flujo asociado:

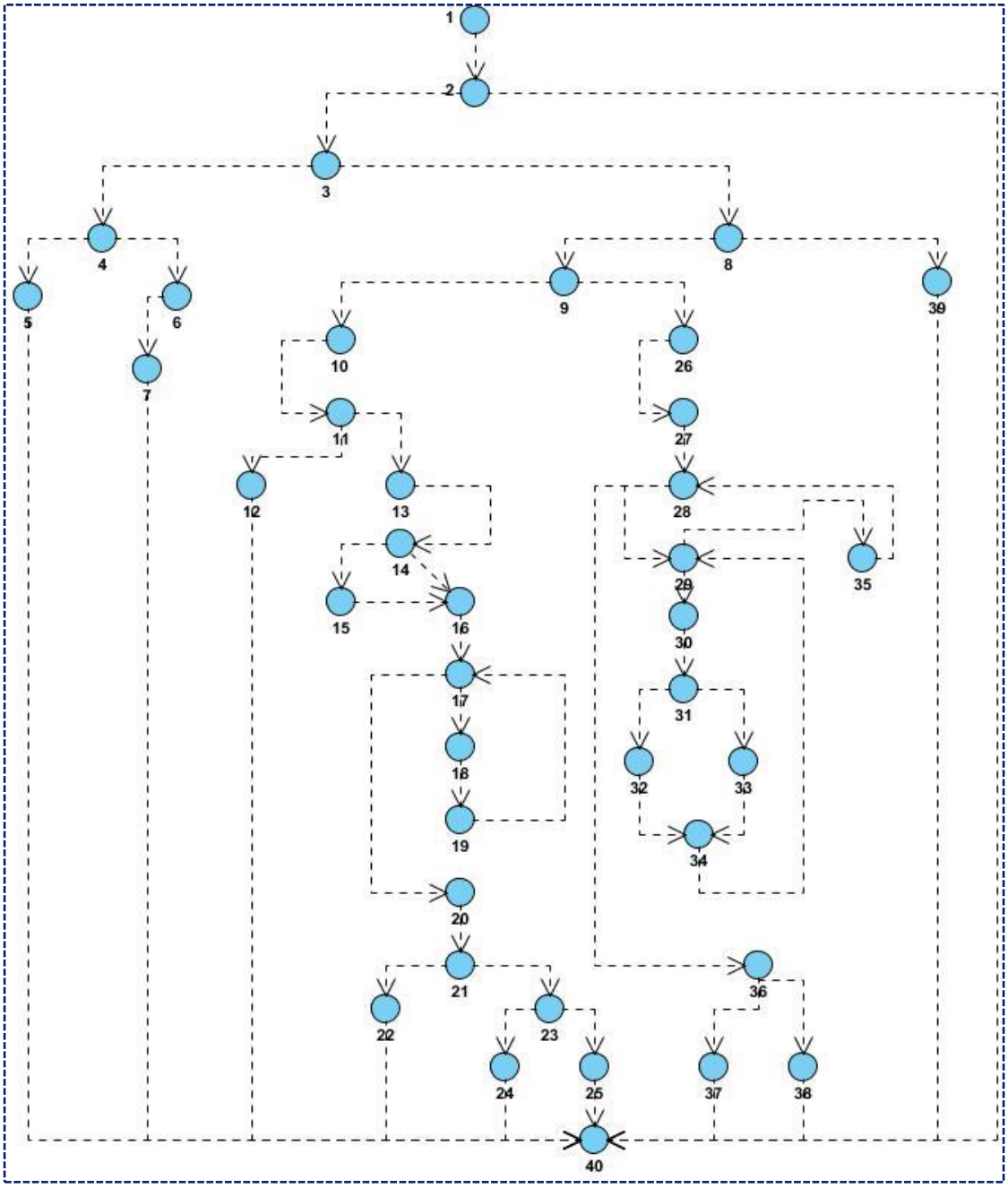

Figura 3.4 Grafo de flujo asociado.

3. Se calcula la complejidad ciclomática del grafo:

La complejidad ciclomática $V(G)$, se define como:

$$V(G) = A - N + 2$$

donde A es el número de aristas del grafo y N el número de nodos.

$$V(G) = 53 - 40 + 2$$

$$V(G) = 15$$

Cada nodo que contiene una condición se denomina nodo predicado y está caracterizado porque dos o más aristas emergen de él. Otra de las variantes para calcular la complejidad ciclomática, y que en este caso se emplea con el objetivo de verificar que los resultados obtenidos sean correctos, es:

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

donde P es el número de nodos predicados contenidos en el grafo G.

$$P: \{2,3,4,8,9,11,14,17,21,23,28,29,31,36\}$$

$$V(G) = 14 + 1 = 15$$

4. Se preparan los casos de prueba que obliguen la ejecución de cada camino del conjunto básico:

Camino 1 (1-2-3-4-5-40)

- Entrada: la cantidad de núcleos, espirales y delta sean 0.
- Salida: clasificación “Arco”.
- Precondiciones:----------

Camino 2 (1-2-3-4-6-7-40)

- Entrada: la cantidad de núcleos debe ser 0, espirales 1 y deltas al menos 1.
- Salida: clasificación “Espiral”.
- Precondiciones:----------

Camino 3 (1-2-3-8-9-10-11-12-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 1, matriz con la ubicación específica de cada punto característico y una matriz que represente la imagen direccional.
- Salida: clasificación “Arco Tendido”.
- Precondiciones: La matriz de los puntos característicos debe tener el valor 1, que es el que indica la presencia de un núcleo, en la misma columna que tenga un 2, que sugiere la presencia de un delta.

Camino 4 (1-2-3-8-9-10-11-13-14-15-16-17-18-19-17-20-21-22-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 1, matriz con la ubicación específica de cada punto característico y una matriz que represente la imagen direccional.
- Salida: clasificación “Arco Tendido”.
- Precondiciones: La matriz de los puntos característicos debe tener el valor 1, que es el que indica la presencia de un núcleo, en una de las columnas ubicadas a la derecha de la columna donde tenga un 2, que sugiere la presencia de un delta. Además la matriz de orientación debe tener los valores correspondientes a los bloques ubicados en la recta que une al delta y el núcleo, de forma tal que el promedio calculado como se definió en el capítulo anterior, sea menor que 0.02.

Camino 5 (1-2-3-8-9-10-11-13-14-15-16-17-18-19-17-20-21-23-24-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 1, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Lazo Izquierdo”.
- Precondiciones: La matriz de los puntos característicos debe tener el valor 1, que es el que indica la presencia de un núcleo, en una de las columnas ubicadas a la izquierda de la columna donde tenga un 2, que sugiere la presencia de un delta. Además la matriz de orientación debe tener los valores correspondientes a los bloques ubicados en la recta que une al delta y el núcleo, de forma tal que el promedio calculado como se definió en el capítulo anterior, sea mayor que 0.02 y la pendiente de esta recta sea positiva.

Camino 6 (1-2-3-8-9-10-11-13-14-15-16-17-18-19-17-20-21-23-25-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y delta 1, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Lazo Derecho”.
- Precondiciones: La matriz de los puntos característicos debe tener el valor 1, que es el que indica la presencia de un núcleo, en una de las columnas ubicadas a la derecha de la columna donde tenga un 2, que sugiere la presencia de un delta. Además la matriz de orientación debe tener los valores correspondientes a los bloques ubicados en la recta que une al delta y el núcleo, de forma tal que el promedio calculado como se definió en el capítulo anterior, sea mayor que 0.02 y la pendiente de esta recta sea negativa.

Camino 7 (1-2-3-8-9-10-11-13-14-16-17-18-19-17-20-21-22-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 1, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Arco Tendido”.
- Precondiciones: La matriz de los puntos característicos debe tener el valor 1, que es el que indica la presencia de un núcleo, en una de las columnas ubicadas a la izquierda de la columna donde tenga un 2, que sugiere la presencia de un delta. Además la matriz de orientación debe tener los valores correspondientes a los bloques ubicados en la recta que une al delta y el núcleo, de forma tal que el promedio calculado como se definió en el capítulo anterior, sea menor que 0.02.

Camino 8 (1-2-3-8-9-10-11-13-14-16-17-18-19-17-20-21-23-24-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 1, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Lazo Izquierdo”.
- Precondiciones: La matriz de los puntos característicos debe tener el valor 1, que es el que indica la presencia de un núcleo, en una de las columnas ubicadas a la izquierda de la columna donde

tenga un 2, que sugiere la presencia de un delta. Además la matriz de orientación debe tener los valores correspondientes a los bloques ubicados en la recta que une al delta y el núcleo, de forma tal que el promedio calculado como se definió en el capítulo anterior, sea mayor que 0.02 y la pendiente de esta recta sea positiva.

Camino 9 (1-2-3-8-9-10-11-13-14-16-17-18-19-17-20-21-23-25-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 1, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Lazo Derecho”.
- Precondiciones: La matriz de los puntos característicos debe tener el valor 1, que es el que indica la presencia de un núcleo, en una de las columnas ubicadas a la derecha de la columna donde tenga un 2, que sugiere la presencia de un delta. Además la matriz de orientación debe tener los valores correspondientes a los bloques ubicados en la recta que une al delta y el núcleo, de forma tal que el promedio calculado como se definió en el capítulo anterior, sea mayor que 0.02 y la pendiente de esta recta sea negativa.

Camino 10 (1-2-3-8-9-26-27-28-29-30-31-32-34-29-35-28-36-37-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 0, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Lazo Izquierdo”.
- Precondiciones: La matriz de orientación debe tener los valores correspondientes a los bloques ubicados alrededor de la misma posición donde la matriz de puntos característicos tenga el valor 1, de forma tal que la cantidad cuyo valor sea mayor que 105 supere a la cifra de los que sean menor.

Camino 11 (1-2-3-8-9-26-27-28-29-30-31-32-34-29-35-28-36-38-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 0, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Lazo Derecho”.
- Precondiciones: La matriz de orientación debe tener los valores correspondientes a los bloques ubicados alrededor de la misma posición donde la matriz de puntos característicos tenga el valor 1, de forma tal que la cantidad cuyo valor sea menor que 105 supere a la cifra de los que sean mayor.

Camino 12 (1-2-3-8-9-26-27-28-29-30-31-33-34-29-35-28-36-37-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 0, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Lazo Izquierdo”.

- Precondiciones: La matriz de orientación debe tener los valores correspondientes a los bloques ubicados alrededor de la misma posición donde la matriz de puntos característicos tenga el valor 1, de forma tal que la cantidad cuyo valor sea mayor que 105 supere a la cifra de los que sean menor.

Camino 13 (1-2-3-8-9-26-27-28-29-30-31-33-34-29-35-28-36-38-40)

- Entrada: la cantidad de núcleos debe ser 1, espirales 0 y deltas 0, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Lazo Derecho”.
- Precondiciones: La matriz de orientación debe tener los valores correspondientes a los bloques ubicados alrededor de la misma posición donde la matriz de puntos característicos tenga el valor 1, de forma tal que la cantidad cuyo valor sea menor que 105 supere a la cifra de los que sean mayor.

Camino 14 (1-2-3-8-39-40)

- Entrada: la cantidad de núcleos debe ser 2, espirales 0 y deltas 0, 1 ó 2, matriz con la ubicación específica de cada punto característico y una matriz que represente una imagen direccional.
- Salida: clasificación “Espiral”.
- Precondiciones: ----------

Camino 15 (1-2-40)

- Entrada: cantidad de núcleos o cantidad de deltas mayor que 2, o cantidad de espirales mayor que uno.
- Salida: clasificación “No Definido”.
- Precondiciones: ----------

3.2.2 Pruebas de aceptación

La prueba de aceptación del usuario es la prueba final antes del despliegue del producto obtenido. Su objetivo es verificar que el software esté listo y que puede ser usado por los usuarios finales para ejecutar aquellas funciones y tareas para las cuales fue construido.

En el caso de XP las pruebas de aceptación son creadas en base a las historias de usuario, en cada iteración del ciclo de desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Las pruebas de aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que sus resultados sean correctos, y en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación (52).

A continuación se muestra el caso de prueba de aceptación correspondiente a la historia de usuario *"Clasificar huella dactilar"*, por ser codificada en la última iteración y englobar el resultado de las implementadas en las iteraciones anteriores. Los casos de prueba de aceptación definidos por el cliente para el resto de las historias de usuario se especifican en los anexos (Ver Anexo 7).

Caso de Prueba de Aceptación	
Código: CPA4_HU4	**HU_4:** Clasificar huella dactilar.
Responsable: Alicia Delgado Hernández	
Descripción: Prueba de funcionalidad para verificar que se clasifique correctamente la huella dactilar.	
Condiciones de ejecución: El usuario debe haber pre-procesado la imagen de la huella dactilar, y detectado sus puntos característicos.	
Entrada / Pasos de ejecución: • Se clasifica la imagen de la huella dactilar a partir de la cantidad de puntos característicos detectados y sus posiciones específicas.	
Resultado esperado: Se muestre en la interfaz de prueba la clasificación de la huella dactilar.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla 3.1 Caso de prueba de aceptación 4.

Este caso de prueba fue aplicado por separado a cada huella dactilar de una base de datos de 100 huellas seleccionadas arbitrariamente, las cuales no están distribuidas uniformemente en las cinco clases, obteniendo los siguientes resultados:

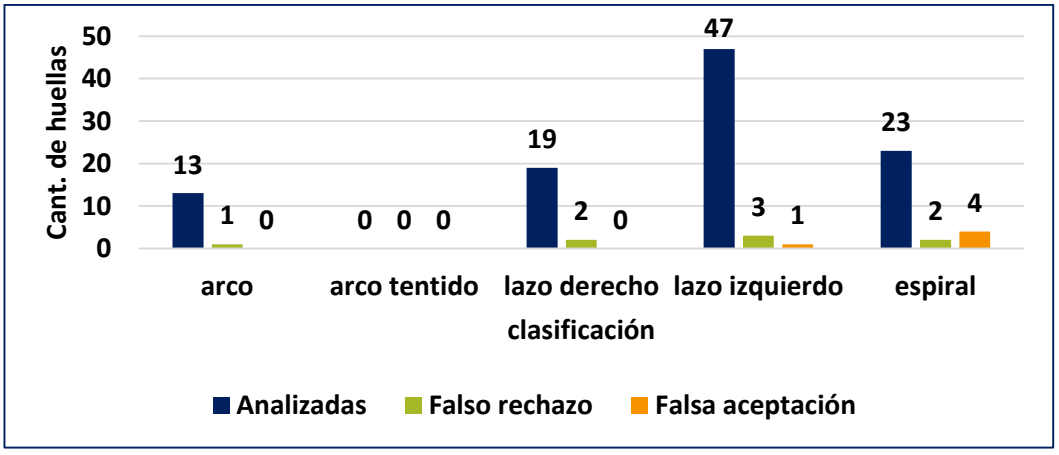

Figura 3.5 Resultados del proceso de clasificación.

A esta misma base de datos se le realizó otra prueba utilizando el algoritmo de comparación del SourceAFIS. Por ejemplo para una de las huellas dactilares de tipo espiral se obtuvieron los siguientes resultados:

- Tiempo de comparación utilizando la base de datos centralizada: 98 milisegundos.
- Tiempo de comparación a partir de la base de datos distribuida en las 5 clases: 45 milisegundos.

O sea que el tiempo de respuesta disminuyó aproximadamente en un 44%.

3.2.3 Resultados de las pruebas

Después de haber realizado las pruebas de aceptación del componente a partir de los cuatro casos de pruebas definidos por el cliente durante las tres iteraciones de codificaciones planificadas, se detectaron en cada una de éstas 9, 12 y 4 no conformidades respectivamente, lo que equivale a un total de 25 no conformidades, que revelaban errores en la codificación que en la mayoría de los casos no incidían significativamente en las respuestas esperadas. Estas dificultades fueron solucionadas en un corto plazo, antes de pasar a la iteración posterior. Una vez obtenido el producto final, se ejecutó adicionalmente una iteración con el propósito de verificar el correcto funcionamiento de todo el módulo, en la que no se detectaron nuevas no conformidades, demostrando que, aunque existan una tasa de falso rechazo y una de falsa aceptación para cada tipo de huella, se cumplieron correctamente todos los requisitos solicitados por el cliente.

Figura 3.6 No conformidades detectadas durante las 3 iteraciones.

Conclusiones Parciales

Durante la etapa de implementación el uso de los estándares de codificación definidos permitió desarrollar un código reutilizable, comprendido por todo el equipo de desarrollo haciendo más simple la puesta a punto de la solución propuesta. La representación del diagrama de componentes, el diagrama de despliegue y las interfaces de pruebas creadas, visualizaron la estructura y funcionamiento del módulo, en el cual tras la ejecución de varias iteraciones cuyas entregas eran inmediatamente probadas se detectaron varias no conformidades, que luego de ser corregidas en su totalidad, validaron su correcto funcionamiento en relación a los requisitos definidos por el cliente.

Conclusiones

Como resultado de la investigación realizada se puede concluir que:

- El análisis de los elementos teóricos y el estado del arte acerca de los procedimientos utilizados en el proceso de búsqueda y comparación de un AFIS, permitió conocer los detalles del tema, proporcionando la posibilidad de definir una propuesta de solución de acuerdo a las necesidades existentes.
- El enfoque ágil propuesto por la metodología XP, y el uso de las tecnologías y herramientas seleccionadas, permitieron analizar y describir los procesos y subprocesos que se debían ejecutar, concretando así, en concordancia con las especificaciones del cliente, las características que debía implementar el módulo.
- La implementación del componente de clasificación de huellas dactilares, luego de ser consecuentemente validada y verificada a partir de las pruebas definidas, y demostrar su correcto funcionamiento, evidenció que su uso sí contribuye a la disminución del campo de análisis del proceso de búsqueda de un AFIS y por ende a la de su tiempo de respuesta.

Recomendaciones

Aunque el uso de la propuesta de solución realizada disminuya en cierta medida, el campo de análisis del proceso de búsqueda del AFIS, esta variante no es la única que se debe aplicar para proveerle escalabilidad y modularidad al sistema al que se integre, puesto que el número de clases que proporciona es bajo y las huellas dactilares se distribuyen de manera desigual entre las distintas clasificaciones. Por ello con el objetivo de seguir optimizando dicho proceso se recomienda que:

- Una vez que el módulo creado se integre al sistema, además de generar una clasificación de patrón considerando los puntos característicos y la orientación de su imagen, se le incorporen los resultados del análisis del flujo de crestas que ejecuta el sistema.
- Se implemente otro componente que realice una clasificación topológica de huellas dactilares que evite los problemas provocados por las huellas ambiguas o las que no se pueden clasificar exclusivamente por la carencia de las singularidades necesarias.

Glosario de términos

- **AFIS:** Sistemas Automáticos de Identificación por Huellas Dactilares.
- **CISED:** Centro de Identificación y Seguridad Digital.
- **FBI:** Buró Federal de Investigaciones.
- **IDE:** Entorno Integrado de Desarrollo.
- **Identificación:** consiste en la comparación de la muestra recogida del usuario frente a una base de datos de rasgos biométricos registrados previamente. No se precisa de identificación inicial por parte del usuario, es decir, el único dato que se recoge es la muestra biométrica, sin apoyo de un nombre de usuario o cualquier otro tipo de reconocimiento. Este método requiere de un proceso complejo, puesto que se ha de comparar esta muestra con cada una de las almacenadas, en búsqueda de una coincidencia (5).
- **Kit de desarrollo:** conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.
- **MINEX** de sus siglas en inglés o **Prueba de Intercambio de Interoperabilidad de las Minucias**: es una evaluación en curso de la plantilla de la huella dactilar INCITS 378. El programa de la prueba tiene dos premisas: proporcionar medidas del funcionamiento e interoperabilidad de las capacidades de codificación y comparación de la plantilla base a los usuarios, a los vendedores y a las partes interesadas, y establecer la conformidad para los codificadores y las unidades de comparación de la plantilla para el Programa de Verificación de Identidad del Personal del gobierno de los Estados Unidos.
- **NIST:** Instituto Nacional de Estándares y Tecnologías.
- **Tasa de falsa aceptación:** estadística utilizada para medir el rendimiento biométrico, que calcula el porcentaje de veces que un sistema produce una falsa aceptación, lo cual ocurre cuando una huella dactilar es erróneamente vinculada a otra clasificación que no es la que le corresponde.
- **Tasa de falso rechazo:** estadística utilizada para medir el rendimiento biométrico, que calcula el porcentaje de veces que el sistema produce un falso rechazo, lo cual ocurre cuando una huella dactilar no es vinculada a su verdadera clasificación.
- **UCI:** Universidad de las Ciencias Informáticas.
- **Verificación:** el primer paso del proceso es la identificación del usuario mediante algún nombre de usuario, tarjeta o algún otro método. De este modo se selecciona de la base de datos el patrón que anteriormente se ha registrado para dicho usuario. Posteriormente, el sistema recoge la característica biométrica y la compara con la que tiene almacenada. Es un proceso simple, al tener que comparar únicamente dos muestras, en el que el resultado es positivo o negativo (5).

Bibliografía Referenciada

1. **Tapiador Mateos, Marino.** *Tecnologías Biométricas aplicadas a la seguridad.* Madrid : Alfaomega, 2005.

2. **Reisz, Carlos F. .** *Identificación Dactiloscópica.*

3. **Berman, Fran y J.G. Hey, Anthony .** *Grid Computing: Making The Global Infrastructure a Reality.* 2003. ISBN: 978-0-470-85319-1.

4. **Candela , G.T., Grother , P.J. y Watson , C.I.** *PCASYS – A Pattern-Level Classification Automation System for Fingerprints.* 1995.

5. **Información, Observatorio de la Seguridad de la.** *Guía sobre las tecnologías biométricas.* España : Instituto Nacional de Tecnologías de la Comunicación (INTECO), Octubre 2011.

6. **Gordon, Jeffrey I.** Forensic identification using skin bacterial communities. [En línea] febrero de 2010. [Citado el: 23 de octubre de 2012.] http://www.pnas.org/content/early/2010/03/01/1000162107.

7. **Rosales Cruz, Aradí.** Clasificación de huellas digitales mediante minucias. [En línea] 28 de abril de 2009. [Citado el: 3 de octubre de 2012.] http://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/reporte_modelos_huellas.pdf.

8. **Martín Rodríguez, Fernando y Suárez López, Fransico Javier.** *Identificación Dactilar basada en filtros de Gabor.*

9. **Maltoni, Davide, y otros, y otros.** *Handbook of fingerprint recognition. Second Edition.* Londres : Springer, 2009. ISBN: 978-1-84882-253-5.

10. **Maltoni, Davide, Maio, Dario y Lumini, Alessandra .** Continuous vs Exclusive Classification for Fingerprint Retrieval. *Pattern Recognition Letters.* 1997.

11. **Biometría, Subcomité de.** [En línea] agosto de 2006. [Citado el: 23 de octubre de 2012.] http://www.biometria.gov.ar/metodos-biometricos/dactilar.aspx.

12. **Salinas Barrera, Virginia y de la Cruz Martínez, Chiristian A.** *Estudio de estándares aplicables a sistemas biométricos.* México : Escuela Superior de Ingeniería Mecánica y Eléctrica, junio de 2009.

13. Neurotechnology, Biometric and Intelligence Artificial Technologies. [En línea] http://neurotechnology.com.co/acerca.html.

14. Neurotechnology, Biometric and Intelligence Artificial Technologies. *MegaMatcher.* [En línea] http://www.neurotechnology.com/megamatcher.html.

15. **Sécurité, Sagem Défense.** *Especificaciones del sistema AFIS Civil.* 2005.

16. **IAFIS, FBI.** Integrated Automated Fingerprint Identification System. [En línea] 7 de octubre de 2010. [Citado el: 23 de enero de 2013.] http://www.fbi.gov/about-us/cjis/fingerprints_biometrics/iafis/iafis.

17. **Moses, Kenneth R., Higgins, Peter y McCabe, Michael.** *Automated Fingerprint Identification System (AFIS).*

18. **DATYS, Tecnologías y Sistemas.** *Especificaciones técnicas Biomesys AFIS Civil.* La Habana : s.n., 2010.

19. **Važan, Robert.** SourceAFIS. [En línea] http://www.sourceafis.org/blog/datasheet/.

20. **Kawagoe , M. y Tojo , A.** *Fingerprint pattern classification.* 1984.

21. **Karu , Kalle y Jain, Anil K. .** *Fingerprint Classification.* s.l. : Department of Computer Science, Michigan State University, 1995.

22. **Hong , L. y Jain , A.K.** *Classification of Fingerprint Images.* 1999.

23. **Rao , K. y Balck , K.** *Type classification of fingerprints: A syntactic approach.* 1980.

24. **Maltoni, Davide y Maio, Dario.** *A Structural Approach to Fingerprint Classification.* 1996.

25. **Jain, Anil K., Prabhakar, S. y Hong, L.** *A multichannel approach to fingerprint classification.* 1999.

26. **Figueroa, Roberth G. , Solís, Camilo J. y Cabrera , Armando A. .** *Metodologías tradicionales vs metodologías ágiles.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

27. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000. ISBN-84-7829-036-2.

28. **Pressman, Roger S.** *Software Engineering: A Practitioner's Approach.* s.l. : McGraw-Hill, 2010. ISBN 978-0-07-337-597-7.

29. **Letelier , Patricio y Penadés, Mª Carmen .** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* España : Universidad Politécnica de Valencia.

30. **Calderón , Amaro y Valverde Rebaza, Sarah Dámaris.** *Metodologías Ágiles.* Trujillo-Perú : Universidad Nacional de Trujillo, 2007.

31. Unified Modeling Language (UML). *IBM.* [En línea] http://www-01.ibm.com/software/rational/uml/.

32. **Rumbaugh, James y Jacobson, Ivar.** *El lenguaje Unificado de Modelado, Manual de Referencia.* 2000. ISBN: 8478290370.

33. **McClure, Carma .** *The CASE Experience.* s.l. : BYTE, Abril 1989.

34. **De Nobrega, Maria .** Herramientas CASE: Rational Rose. [En línea] http://curso_sin2.blogia.com/2005/060401-herramientas-case-rational-rose.-por-maria-de-nobrega.php.

35. Visual Paradigm. [En línea] enero de 2013. http://www.visual-paradigm.com/.

36. **Baeza, Pablo Nicolás.** Visual Paradigm DB Visual ARCHITECT SQL. [En línea] enero de 2013. http://www.docstoc.com/docs/96492173/Visual-Paradigm-Studio.

37. **Wilson, Leslie Blackett.** *Comparative Programming Languages.* s.l. : Addison-Wesley, 1993. ISBN 0-201-56885-3.

38. **AmericaTI.** Ventajas y Desventajas: Comparación de los Lenguajes C, C++ y Java. [En línea] diciembre de 2012. http://www.americati.com/doc/ventajas_c.pdf.

39. **Microsoft.** Introducción al lenguaje C# y .NET Framework. *msdn.* [En línea] diciembre de 2012. http://msdn.microsoft.com/es-es/library/z1zx9t92%28v=VS.80%29.aspx.

40. —. Visual Studio 2012. *Visual Studio.* [En línea] diciembre de 2012. http://www.microsoft.com/visualstudio/esn/visual-studio-update.

41. **NetBeans.** Información NetBeans IDE 6.1. [En línea] https://netbeans.org/community/releases/61/index_es.html

42. **Thai, Raymond .** *Fingerprint Image Enhancement and Minutiae Extraction.* s.l. : University of Western Australia, 2003.

43. **Babatunde , Iwasokun G. , y otros, y otros.** *A Multi-Level Model for Fingerprint Image Enhancement.* Hong Kong : Proceedings of the International MultiConference of Engineers and Computer Scientists, 2012.

44. **Kruchtens, Philippe .** *Architectual Blueprints – the "4+1" View Model of Software Architecture.*

45. **Hilliard, Rich .** *IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems.* noviembre del 2000.

46. **Billy Reynoso, Carlos .** *Introducción a la Arquitectura de Software.* s.l. : UNIVERSIDAD DE BUENOS AIRES.

47. **Garlan , David y Shaw, Mary.** *An introduction to software architecture.* s.l. : CMU Software Engineering Institute Technical Report, 1994. CMU/SEI-94-TR-21.

48. **Reynoso , Carlos y Kicillof, Nicolás .** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* s.l. : UNIVERSIDAD DE BUENOS AIRES, 2004.

49. **GAMMA, E., HELM, R. y JOHNSON, R. y VLISSIDES, J.** *Patrones de diseño.* 2000.

50. **Larman, Craig.** *UML y patrones.* 1999.

51. **Microsoft.** Técnicas de codificación y prácticas de programación. *MSDN.* [En línea] http://msdn.microsoft.com/.

52. **Joskowicz, Ing. José .** *Reglas y Prácticas en eXtreme Programming.* España : Doctorado de Ingeniería Telemática de la Universidad, 2008.

Bibliografía Consultada

- **Tapiador Mateos, Marino.** *Tecnologías Biométricas aplicadas a la seguridad.* Madrid : Alfaomega, 2005.
- **Reisz, Carlos F. .** *Identificación Dactiloscópica.*
- **Información, Observatorio de la Seguridad de la.** *Guía sobre las tecnologías biométricas.* España : Instituto Nacional de Tecnologías de la Comunicación (INTECO), Octubre 2011.
- **Rosales Cruz, Aradí.** Clasificación de huellas digitales mediante minucias. [En línea] 28 de abril de 2009. [Citado el: 3 de octubre de 2012.] http://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/reporte_modelos_huellas.pdf.
- **Martín Rodríguez, Fernando y Suárez López, Fransico Javier.** *Identificación Dactilar basada en filtros de Gabor.*
- **Maltoni, Davide, Maio, Dario y Lumini, Alessandra .** Continuous vs Exclusive Classification for Fingerprint Retrieval. *Pattern Recognition Letters.* 1997.
- **Biometría, Subcomité de.** [En línea] agosto de 2006. [Citado el: 23 de octubre de 2012.] http://www.biometria.gov.ar/metodos-biometricos/dactilar.aspx.
- **Salinas Barrera, Virginia y de la Cruz Martínez, Chiristian A.** *Estudio de estándares aplicables a sistemas biométricos.* México : Escuela Superior de Ingeniería Mecánica y Eléctrica, junio de 2009.
- **Sécurité, Sagem Défense.** *Especificaciones del sistema AFIS Civil.* 2005.
- **IAFIS, FBI.** Integrated Automated Fingerprint Identification System. [En línea] 7 de octubre de 2010. [Citado el: 23 de enero de 2013.] http://www.fbi.gov/about-us/cjis/fingerprints_biometrics/iafis/iafis.
- **DATYS, Tecnologías y Sistemas.** *Especificaciones técnicas Biomesys AFIS Civil.* La Habana : s.n., 2010.
- **Važan, Robert.** SourceAFIS. [En línea] http://www.sourceafis.org/blog/datasheet/.
- **Kawagoe , M. y Tojo , A.** *Fingerprint pattern classification.* 1984.
- **Karu , Kalle y Jain, Anil K. .** *Fingerprint Classification.* s.l. : Department of Computer Science, Michigan State University, 1995.
- **Hong , L. y Jain , A.K.** *Classification of Fingerprint Images.* 1999.
- **Figueroa, Roberth G. , Solís, Camilo J. y Cabrera , Armando A. .** *Metodologías tradicionales vs metodologías ágiles.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

- **Joskowicz, Ing. José .** *Reglas y Prácticas en eXtreme Programming.* España : Doctorado de Ingeniería Telemática de la Universidad, 2008.
- **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000. ISBN-84-7829-036-2.
- **Pressman, Roger S.** *Software Engineering: A Practitioner's Approach.* s.l. : McGraw-Hill, 2010. ISBN 978-0-07-337-597-7.
- **Letelier , Patricio y Penadés, Mª Carmen .** *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* España : Universidad Politécnica de Valencia.
- **Thai, Raymond .** *Fingerprint Image Enhancement and Minutiae Extraction.* s.l. : University of Western Australia, 2003.
- **Babatunde , Iwasokun G. , y otros, y otros.** *A Multi-Level Model for Fingerprint Image Enhancement.* Hong Kong : Proceedings of the International MultiConference of Engineers and Computer Scientists, 2012.
- **Billy Reynoso, Carlos .** *Introducción a la Arquitectura de Software.* s.l. : UNIVERSIDAD DE BUENOS AIRES.
- **Garlan , David y Shaw, Mary.** *An introduction to software architecture.* s.l. : CMU Software Engineering Institute Technical Report, 1994. CMU/SEI-94-TR-21.
- **Reynoso , Carlos y Kicillof, Nicolás .** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* s.l. : UNIVERSIDAD DE BUENOS AIRES, 2004.
- **GAMMA, E., HELM, R. y JOHNSON, R. y VLISSIDES, J.** *Patrones de diseño.* 2000.
- **Larman, Craig.** *UML y patrones.* 1999.
- **Microsoft.** Técnicas de codificación y prácticas de programación. *MSDN.* [En línea] http://msdn.microsoft.com/.
- *Acerca de la prueba de MINEX.* [En línea] http://www.precision.es/minex-testing.php.
- Selecting a development approach. [En línea] https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads.
- **Zorita, Danilo Simón.** *Tesis doctoral Reconocimiento automático mediante patrones biométricos de huella dactilar.* España : Universidad Politécnica de Madrid, 2003.
- **Landi Diez, Juan Carlos.** *Introducción a la biometría informática y análisis de la huella dactilar como fuentes de autenticación en sistemas de seguridad.* s.l. : Universidad Politécnica Salesiana, Sede Cuenca, 2007.

- **Aguilar, Gualberto , y otros, y otros.** *Reconocimiento de huellas dactilares usando características locales.* s.l. : Facultad de Ingeniería Universidad de Antioquia, 2008.
- **Mukherjee, Rajiv .** *Indexing Techniques for Fingerprint and Iris Databases.* Virginia Oeste, EE.UU : Universidad del estado Virginia Oeste, 2007.
- **Departamento de Justicia.** *Integrated Automated Fingerprint Identification System (IAFIS) and related records.* EE.UU : FBI, 2004.
- **Interpol Implementation of ANSI/NIST-ITL 1-2000.** *Data format for the interchange of fingerprint, facial & smt information .* s.l. : The Interpol AFIS Expert Group, 2004.
- **NIST Special Publication 500-245.** *American National Standard for Information Systems Data Format for the Interchange of Fingerprint, Facial, & Scar Mark & Tattoo (SMT) Information.* 2000.
- **Puebla Martínez, Ángel Luis.** *Tesis doctoral Identificación de individuos mediante comparación borrosa de elementos lingüísticos obtenidos a partir de las impresiones dactilares.* Madrid : Departamento de Tecnología Fotónica, Facultad de Informática.
- **Ruiz Marín, Milton, Rodríguez Uribe, Juan Carlos y Olivares Morales, Juan Carlos.** *Una mirada a la biometría.* Colombia : Universidad de Magdalena, 2009. ISSN 1657-7663.
- **Bhuyan , Monowar Hussain, Saharia, Sarat y Bhattacharyya, Dhruba Kr .** *An Effective Method for Fingerprint Classification.* India : Department of Computer Science & Engineering, Tezpur University, 2009.
- **Carvajal Riola, Jose Carlos .** *Metodologías ágiles: Herramientas y modelo de desarrollo para aplicaciones Java EE como metodología empresarial.* España : UPC Barcelona, 2008.
- **Wei, Liu .** *Fingerprint Classification Using Singularities Detection.* s.l. : INTERNATIONAL JOURNAL OF MATHEMATICS AND COMPUTERS IN SIMULATION, 2008.
- **Tan, Xuejun , Bhanu , Bir y Lin, Yingqiang .** *Learning Features for Fingerprint Classification.* EE.UU : Universidad de California.
- **Zhou, Jie y Gu, Jinwei .** *Modeling orientation fields of fingerprints with rational complex functions.* China : Tsinghua University, 2003.
- **Conejero, José Alberto y Fernández, Margarita.** *Desarrollo de un modelo matemático de clasificación dactiloscópica.* s.l. : Universidad Politécnica de Valencia.

Anexos

- **Anexo 1:** Tabla de comparación entre metodologías ágiles y tradicionales.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo de desarrollo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas y normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.

Tabla A.1 Diferencias entre metodologías tradicionales y ágiles.

- **Anexo 2:** Historia de usuario correspondiente al requisito funcional leer imagen de huella dactilar.

Historia de Usuario	
Número: HU_1	**Usuario:** Alicia Delgado Hernández
Nombre de historia: Leer imagen de huella dactilar.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Alicia Delgado Hernández	
Descripción: El componente debe ser capaz de cargar la imagen de la huella dactilar, la cual debe estar ubicada en una de las carpetas de la computadora.	

Tabla A.2 HU Leer imagen de huella dactilar.

- **Anexo 3:** Tareas ingenieriles para codificar cada una de las historias de usuario.

Tarea de ingeniería	
Número: TI_1_HU_1	**Historia de usuario:** Leer imagen de huella dactilar.
Nombre de tarea: Leer desde una dirección especificada por el usuario, la imagen de la huella dactilar.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 14/01/2013	**Fecha fin:** 18/01/13
Programador responsable: Alicia Delgado Hernández	
Descripción: El componente debe cargar una imagen almacenada en un archivo específico.	

Tabla A.3 TI_1_HU_1

Tarea de ingeniería	
Número: TI_1_HU_2	**Historia de usuario:** Pre-procesar imagen de la huella dactilar.
Nombre de tarea: Determinar la región de interés de la huella dactilar.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 21/01/2013	**Fecha fin:** 25/01/13
Programador responsable: Alicia Delgado Hernández	
Descripción: Se debe definir la región de interés de la huella dactilar haciendo uso de la varianza y la media de cada bloque, y un umbral que en caso de la solución propuesta es la media global.	

Tabla A.4 TI_1_HU_2

Tarea de ingeniería	
Número: TI_2_HU_2	**Historia de usuario:** Pre-procesar imagen de la huella dactilar.
Nombre de tarea: Optimizar la región de interés obtenida.	

Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 04/02/2013	**Fecha fin:** 08/02/13
Programador responsable: Alicia Delgado Hernández	
Descripción: Se optimiza la región de interés obtenida en la iteración anterior, teniendo en cuenta el valor de cada bloque y el comportamiento de sus vecinos.	

Tabla A.5 TI_2_HU_2

Tarea de ingeniería	
Número: TI_3_HU_2	**Historia de usuario:** Pre-procesar imagen de la huella dactilar.
Nombre de tarea: Calcular la orientación de cada pixel de la imagen aplicando el gradiente de Sobel.	
Tipo: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 11/02/2013	**Fecha fin:** 22/02/13
Programador responsable: Alicia Delgado Hernández	
Descripción: Se calcula la imagen direccional de la huella dactilar, que describe el recorrido de las crestas papilares.	

Tabla A.6 TI_3_HU_2

Tarea de ingeniería	
Número: TI_4_HU_2	**Historia de usuario:** Pre-procesar imagen de la huella dactilar.
Nombre de tarea: Suavizar la orientación obtenida a partir de un filtro gaussiano, para obtener una imagen direccional con mayor calidad.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 04/03/2013	**Fecha fin:** 08/03/13
Programador responsable: Alicia Delgado Hernández	
Descripción: El propósito de esta tarea es obtener una imagen direccional de mejor calidad.	

Tabla A.7 TI_4_HU_2

Tarea de ingeniería	
Número: TI_5_HU_2	**Historia de usuario:** Pre-procesar imagen de la huella dactilar.
Nombre de tarea: Obtener la imagen direccional por bloques.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 11/03/2013	**Fecha fin:** 15/03/13
Programador responsable: Alicia Delgado Hernández	
Descripción: Se divide la imagen en bloques de dimensiones $9\,x\,9$, y se halla la orientación de cada uno a partir de la dirección de los píxeles que lo integran.	

Tabla A.8 TI_5_HU_2

Tarea de ingeniería	
Número: TI_1_HU_3	**Historia de usuario:** Detección de los puntos característicos.
Nombre de tarea: Cálculo del índice de Poincaré para cada bloque.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 25/03/2013	**Fecha fin:** 29/03/13
Programador responsable: Alicia Delgado Hernández	
Descripción: El índice de Poincaré se define como la rotación total de los vectores de las direcciones de los bloques que rodean a un bloque determinado. En dependencia del valor que tome éste en el bloque de análisis se puede detectar un punto característico.	

Tabla A.9 TI_1_HU_3

Tarea de ingeniería	
Número: TI_2_HU_3	**Historia de usuario:** Detección de los puntos característicos.
Nombre de tarea: Detección de los puntos característicos.	
Tipo: Desarrollo	**Puntos estimados:** 1

Fecha inicio: 01/04/2013	**Fecha fin:** 05/04/13
Programador responsable: Alicia Delgado Hernández	
Descripción: A partir del valor del índice de Poincaré de cada bloque se detectan los puntos característicos.	

Tabla A.10 TI_2_HU_3

Tarea de ingeniería	
Número: TI_3_HU_3	**Historia de usuario:** Detección de los puntos característicos.
Nombre de tarea: Descartar falsos puntos característicos.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 22/04/2013	**Fecha fin:** 26/04/13
Programador responsable: Alicia Delgado Hernández	
Descripción: Se ha demostrado que un punto característico no suele estar ubicado en un único bloque, por lo que bloques contiguos tienden a tener el mismo índice de Poincaré, esta es la causa por la que se debe aplicar un algoritmo que elimine falsos puntos característicos.	

Tabla A.11 TI_3_HU_3

Tarea de ingeniería	
Número: TI_1_HU_4	**Historia de usuario:** Clasificar huella dactilar.
Nombre de tarea: Analizar la cantidad de puntos característicos de cada tipo que fueron detectados y calcular su posición (x, y), que equivale a (columna, fila).	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 29/04/2013	**Fecha fin:** 03/05/13
Programador responsable: Alicia Delgado Hernández	
Descripción: Se debe calcular las coordenadas de cada punto característico, para conocer su posición respecto al resto.	

Tabla A.12 TI_1_HU_4

Tarea de ingeniería	
Número: TI_1_HU_4	**Historia de usuario:** Clasificar huella dactilar.
Nombre de tarea: Clasificación final de la huella dactilar.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 06/05/2013	**Fecha fin:** 10/05/13
Programador responsable: Alicia Delgado Hernández	
Descripción: Se le asigna a la huella su clasificación correspondiente de acuerdo a la cantidad de puntos característicos que posee y a la posición de éstos.	

Tabla A.13 TI_2_HU_4

- **Anexo 4:** Tarjetas CRC de las clases correspondientes al nivel 3

Clase Interfaz_Prueba	
Responsabilidades	**Colaboradores**
• Aunque el propósito de la presente investigación sea crear un componente que se integre a un sistema, por lo que no debe contar con una interfaz propia, se propone la creación de una de prueba que valide el correcto funcionamiento de todas las historias de usuario. En ésta el usuario puede realizar cada uno de los subprocesos y observar visualmente las transformaciones que se le aplican a la imagen de la huella dactilar, concluyendo con la clasificación de la misma.	Dibujo, Controladora

Tabla A.14 Tarjeta CRC Interfaz_Prueba.

Clase Filtro	
Responsabilidades	**Colaboradores**
• Calcular la imagen direccional de la huella dactilar para cada pixel (i, j) a través del gradiente de Sobel.	----------
• Suavizar la orientación de la imagen, aplicando un filtro Gaussiano de paso bajo, con el propósito de lograr que las direcciones calculadas se ajusten mejor al recorrido real de las crestas papilares.	----------
• Filtrar la imagen multiplicando el valor de la intensidad del nivel de gris de un pixel y el de sus píxeles	----------

circundantes por una matriz llamada kernel, que se mueve por cada uno de los píxeles de la imagen original y cada pixel que queda bajo la matriz se multiplica por un valor de ésta, se suma y divide después por un valor específico. El resultado de esta operación es el valor del pixel que cae en el centro de la matriz.	

Tabla A.15 Tarjeta CRC Filtro.

Clase Operacion	
Responsabilidades	**Colaboradores**
• Calcular la media global de los valores de gris de cada pixel de la imagen de la huella dactilar.	----------
• Calcular la media de los bloques de dimensiones $\mathbf{9}\ \boldsymbol{x}\ \mathbf{9}$ de la imagen.	----------
• Dada una matriz de ángulos en radianes, retornar una matriz con el valor de los ángulos en grados.	----------

Tabla A.16 Tarjeta CRC Operacion.

Clase Tipo_Huella	
Responsabilidades	**Colaboradores**
• Es un enumerativo que establece todos los tipos de clasificaciones que se le pueden asignar a una huella dactilar determinada.	----------

Tabla A.17 Tarjeta CRC Tipo_Huella.

Clase Dibujo	
Responsabilidades	**Colaboradores**
• Muestra en la interfaz de prueba creada, el resultado visual de la segmentación de la imagen, la orientación de la misma y sus puntos característicos detectados.	----------

Tabla A.18 Tarjeta CRC Dibujo.

Clase Imprimir_Matrices	
Responsabilidades	**Colaboradores**
• A partir de una matriz y una dirección de archivos específicas, crea en esta última un cuadro de texto con los valores de la primera.	----------

Tabla A.19 Tarjeta CRC ImprimirMatrices.

- **Anexo 5:** interfaces del componente de clasificación de huellas dactilares

Esta interfaz se obtiene luego de que el usuario haya cargado la imagen de la huella dactilar que desea clasificar.

Figura A.1 Interfaz que presenta la imagen original de la huella dactilar.

Si el usuario desea realizar el proceso por etapas presiona el primer botón del menú procesamiento “Segmentar imagen”.

Figura A.2 Interfaz que muestra el área de interés.

Después se halla la imagen direccional o campo de orientación de la huella dactilar:

Figura A.3 Interfaz que muestra la imagen direccional de la huella dactilar.

Se detectan y extraen los puntos característicos:

Figura A.4 Interfaz que muestra los puntos característicos detectados.

- **Anexo 6:** código del algoritmo para clasificar la huella dactilar.

```
/* A partir de la cantidad de núcleos, deltas y/o espirales que tenga una huella dactilar, y la posición de los mismos, se clasifica de acuerdo a
la clasificación de henry. */
        public static void Clasificacion_Huella(int cantidad_nucleo, int cantidad_delta, int cantidad_espiral,                    (1)
int[,] puntos_caracteristicos, double[,] matriz_orientacion_bloques, out TipoHuella clasificacion)
        {
            double media=0;                                                                                                      (1)
            clasificacion=TipoHuella.No_Definido;                                                                                (1)
            //Chequea que la cantidad de núcleos, de delta y de espirales sean válidas
            if (cantidad_delta <= 2 && cantidad_espiral <= 1 && cantidad_nucleo <= 2)                                             (2)
            {
                if (cantidad_nucleo == 0)                                                                                        (3)
                {
                    //Si la huella carece de puntos característicos, la clasificación asignada es de tipo arco.
                    if (cantidad_espiral == 0 && cantidad_delta == 0)                                                            (4)
                        clasificacion = TipoHuella.Arco;                                                                         (5)
                    else
                        //Si no existe núcleo y existe un espiral y uno o más deltas, la huella es de tipo espiral
                        if (cantidad_espiral == 1 && cantidad_delta >= 1)                                                        (6)
                            clasificacion = TipoHuella.Espiral;                                                                  (7)
                }
                else
                {
                    if (cantidad_nucleo == 1)                                                                                    (8)
                    {
                        /*Si tiene un núcleo, un delta y no tiene espiral la huella puede ser de tipo lazo izquierdo,
                        lazo derecho o arco tendido, en dependencia de la posición de los dos puntos
                        característicos.*/
                        if (cantidad_delta == 1 && cantidad_espiral == 0)                                                        (9)
                        {
                            int pos_X_nucleo = 0;                                                                                (10)
                            int pos_Y_nucleo = 0;                                                                                (10)
                            int pos_X_delta = 0;                                                                                 (10)
                            int pos_Y_delta = 0;                                                                                 (10)
Posicion_Puntos(puntos_caracteristicos, out pos_X_delta, out pos_Y_delta, out pos_X_nucleo, out  pos_Y_nucleo);                  (10)
                            // Chequea que ambos puntos estén en la misma columna.
                            if (pos_X_delta == pos_X_nucleo)                                                                     (11)
                                clasificacion = TipoHuella.Arco_Tendido;                                                         (12)
                            else
                            {
                                double m = (float)(pos_Y_delta - pos_Y_nucleo) / (pos_X_delta - pos_X_nucleo);                   (13)
                                double n = pos_Y_nucleo - m * pos_X_nucleo;                                                      (13)
                                int x_inicial = pos_X_nucleo;                                                                    (13)
                                int x_final = pos_X_delta;                                                                       (13)
                                if (pos_X_nucleo > pos_X_delta)                                                                  (14)
                                {
                                    x_inicial = pos_X_delta;                                                                     (15)
                                    x_final = pos_X_nucleo;                                                                      (15)
                                }
                                double suma_promedio = 0;                                                                        (16)
                                int cont = 0;                                                                                    (16)
                                for (int j = x_inicial+1; j < x_final + 1; j++)                                                  (17)
                                {
                                    int coor_Y = (int)(m * j + n);                                                               (18)
                                    double argumento_seno = matriz_orientacion_bloques[coor_Y, j] - m;                            (18)
                                    suma_promedio += Math.Sin(argumento_seno);                                                   (18)
                                    cont++;                                                                                      (18)
                                }                                                                                                (19)
                                                                                                                                 (20)
                                       clasificacion = TipoHuella.Arco_Tendido;                                                  (21)
```

```
            else                                                                                    (22)
               if (m > 0)                                                                           (23)
                   clasificacion = TipoHuella.Lazo_Izquierdo;                                       (24)
               else
                   clasificacion = TipoHuella.Lazo_Derecho;                                         (25)
    }
}
else
{  //Esto es para las huellas donde solamente se identifica un núcleo.
    if (cantidad_delta == 0 && cantidad_espiral == 0)                                               (26)
    {
        int pos_X_nucleo = 0;                                                                       (27)
        int pos_Y_nucleo = 0;                                                                       (27)
        int cont1 = 0;                                                                              (27)
        int cont2 = 0;                                                                              (27)
        Posicion_Nucleo(puntos_caracteristicos, out pos_X_nucleo, out pos_Y_nucleo);                (27)
        for (int i = pos_Y_nucleo; i < pos_Y_nucleo + 4; i++)                                       (28)
        {
            for (int j = pos_Y_nucleo - 3; j < pos_Y_nucleo + 3; j++)                               (29)
            {
                media = (matriz_orientacion_bloques[i, j] * 180) / Math.PI;                         (30)
                if (media > 105)                                                                    (31)
                    cont1++;                                                                        (32)
                else
                    cont2++;                                                                        (33)
            }                                                                                       (34)
        }                                                                                           (35)
           if (cont2 < cont1)                                                                       (36)
            clasificacion = TipoHuella.Lazo_Izquierdo;                                              (37)
        else
                        clasificacion = TipoHuella.Lazo_Derecho;                                    (38)
                    }  }
            }

          //Esto es para cuando existen dos núcleos que indican la presencia de un espiral.
            else
            {
             clasificacion = TipoHuella.Espiral;                                                    (39)
            }
        }
    }
}                                                                                                   (40)
```

- **Anexo 7:** casos de prueba de aceptación.

Caso de Prueba de Aceptación	
Código: CPA1_HU1	**HU_1:** Leer imagen de huella dactilar.
Responsable: Alicia Delgado Hernández	
Descripción: Prueba de funcionalidad para verificar que el componente lee correctamente desde un archivo especificado por el usuario la imagen de la huella dactilar.	
Condiciones de ejecución: El usuario debe especificar la dirección.	
Entrada / Pasos de ejecución: • Leer la dirección dada por el usuario. • Almacenar la imagen en una variable determinada.	
Resultado esperado: Se muestre en la interfaz de prueba la imagen de la huella dactilar.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla A.20 Caso de prueba de aceptación 1.

Caso de Prueba de Aceptación	
Código: CPA2_HU2	**HU_2:** Pre-procesar imagen de la huella dactilar.
Responsable: Alicia Delgado Hernández	
Descripción: Prueba de funcionalidad para verificar que el componente procese correctamente la imagen de la huella dactilar.	
Condiciones de ejecución: El usuario debe haber cargado la imagen de la huella dactilar.	
Entrada / Pasos de ejecución: • Se segmenta la imagen dada, con el objetivo de definir la región de interés. • Se aplica un algoritmo sobre la segmentación obtenida con el objetivo de optimizar el resultado, analizando los bloques que se hayan marcado incorrectamente como fondo de la imagen de la huella dactilar o como parte de ésta. • Se calcula la imagen direccional de cada uno de los píxeles correspondientes a la región de interés, utilizando en un primer momento el gradiente de Sobel, y luego para lograr un mejor ajuste respecto al recorrido real de las crestas papilares, se suaviza la orientación aplicando un filtro de Gauss. • Se calcula la imagen direccional por bloques.	

Resultado esperado: Se muestre en la interfaz de prueba la imagen de la huella dactilar orientada.
Evaluación de la Prueba: Prueba satisfactoria

Tabla A.21 Caso de prueba de aceptación 2.

Caso de Prueba de Aceptación	
Código: CPA3_HU3	**HU_3:** Extracción de los puntos característicos.
Responsable: Alicia Delgado Hernández	
Descripción: Prueba de funcionalidad para verificar que se detecten correctamente los puntos característicos presentes en la huella dactilar.	
Condiciones de ejecución: El usuario debe haber pre-procesado la imagen de la huella dactilar.	
Entrada / Pasos de ejecución: • Se calcula el índice de Poincaré para cada bloque de la imagen que pertenece a la región de interés y se detectan los puntos característicos. • Se aplica un método para eliminar los falsos puntos característicos detectados.	
Resultado esperado: Se muestre en la interfaz de prueba la imagen de la huella dactilar con sus puntos característicos detectados.	
Evaluación de la Prueba: Prueba satisfactoria	

Tabla A.22 Caso de prueba de aceptación 3.