UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE INFORMACIÓN PARA EL ACOMODO DE PIEZAS DE CALZADO

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

PRESENTA:

FÉLIX VIDRIOS HÉCTOR

DIRECTOR: M.I. LUIS GUEVARA PALMA

MÉXICO, D.F. AGOSTO 2009

A mís padres,

Por todo el amor y apoyo que me han brindado

A Ana Lília, Te Amo

A Adrian,

Quien definitivamente es más importante que Java

Héctor Félix

Contenido

CAPÍTULO 5 Pruebas y resultados

Agradecimientos

A la Universidad Nacional Autónoma de México

A la Facultad de Ingeniería

Le agradezco infinitamente a mi familia, a mi padre por su gran ejemplo, a mi madre por todo el cariño que me ha dado y a mi hermana Blanca, los amo.

Al Consejo Nacional de Ciencia y Tecnología por el apoyo brindado para la realización de esta tesis a través del proyecto: Diseño mecatrónico de una plataforma para el acomodo de piezas (nesting), con número GTO-2006-C01-31927 formando parte del Fondo Mixto de Fomento a la Investigación Científica y Tecnológica CONACYT – Gobierno del Estado de Querétaro.

A la distribuidora Flexi, por permitirme participar en este proyecto multidisciplinario.

Al CDMIT por las facilidades otorgadas para la realización de éste trabajo.

Al Dr. Saúl Santillán y al M.I. Luis Guevara por el apoyo y la confianza que depositaron en mí al incluirme en el proyecto.

A todos mis amigos, quienes me han ayudado inclusive desde antes de comenzar con este proyecto, muchísimas gracias por estar ahí en todo momento.

A la FES Aragón y muy especialmente a los maestros Luis y Marcelo (el tigre), sin su ayuda ingresar a la maestría seguiría siendo un sueño.

Introducción

Actualmente la industria del calzado nacional se encuentra en la problemática de haber descendido su producción de manera significativa en los últimos años, en la búsqueda de una solución a este problema, la UNAM y una empresa líder de calzado han efectuado un proyecto en el cual se considera el diseño e implementación de un sistema integrado de manufactura por computadora para el proceso de acomodo de piezas de calzado y corte de piel.

En este sistema, se contempla la automatización desde el pre-procesamiento de la piel, pasando por optimización en la selección de la piel dado un determinado modelo de calzado, optimización del aprovechamiento de piel mediante un acomodo adecuado de las piezas en zonas de calidad correspondientes, el corte, registro de piezas cortadas, inventario, y trazabilidad de las piezas para su siguiente etapa de fabricación.

El mayor valor agregado en este sistema es el proceso de acomodo automático de piezas, conocido como ***Nesting***, que permite aprovechar al máximo la piel y garantizar el cumplimiento de los parámetros requeridos, ya que aún no existe un sistema de nesting que cumpla con los requisitos de la empresa.

En este Sistema de Nesting se ubica el **Sistema de Información para el acomodo de piezas de calzado,** cuyo objetivo es *almacenar, obtener y administrar* la información necesaria para llevar a cabo el proceso de nesting, la cual es:

- orden de trabajo
- piezas del modelo de zapato requerido y
- pieles a procesar para cubrir la cantidad de zapatos especificados en la orden de trabajo, que permitan minimizar el desperdicio de la piel

Ésta información se encuentra distribuida en diferentes equipos y su generación depende de otros sistemas, a partir de dicha información se generan las estructuras de datos que requiere el programa que se encarga de realizar el acomodo.

El Sistema está conformado por programas desarrollados en diferentes lenguajes que operan en diferentes tiempos para ajustarse al proceso de producción de la empresa, debido a que el objetivo final es instalar el sistema en la planta.

En lo referente a la estructura de la tesis, ésta se divide en cinco capítulos, los cuales se presentan a continuación.

En el Capítulo 1 se describen las generalidades del proceso de nesting como son: el panorama general y nacional de la industria de calzado, así como los sistemas de corte automático de cuero existentes.

En el capítulo 2 se describe la información necesaria para el proceso de nesting y se analizan las alternativas de solución, con las que se podría desarrollar el sistema de

información; estas alternativas incluyen los modelos de proceso del software, los modelos de base de datos, los lenguajes de programación y los sistemas gestores de base de datos, para así poder seleccionar el software y hardware que contribuya de mejor manera al desarrollo del sistema de información.

El capítulo 3 presenta el diseño del sistema de información, mostrando los elementos que conforman al sistema, se describen los equipos de cómputo en los cuales se encuentra distribuido dicho sistema, el diseño de la base de datos; para concluir con el flujo de información dentro del sistema.

El desarrollo del sistema se fundamenta en el capítulo 4, agrupando los programas que lo integran en tres categorías: programas que insertan información en la base de datos, programas que seleccionan, obtienen y cargan información y por último los programas que procesan la información, además se declaran los problemas encontrados en la elaboración de los mismos.

El capítulo 5 trata sobre las pruebas y resultados del sistema, pruebas hechas a cada uno de los programas por separado y pruebas hechas en conjunto, para verificar que el sistema responde de manera correcta a la petición de las estructuras de datos hechas por el programa de nesting, al igual que ejecuta las transacciones hacia la base de datos.

Finalmente en las conclusiones se comenta la manera en que se cumplieron los objetivos, el desempeño de las herramientas de desarrollo empleadas y el trabajo posterior que puede realizarse.

Capítulo 1 Generalidades

1.1 Antecedentes

1.1.1 Panorama general

La evolución de la competitividad en calzado, en varios países ha tenido cambios sustanciales en el periodo 1985-98. Entre 1985 y 1990, Italia era el líder exportador con 28.5 por ciento del mercado, le seguía Taiwán y Corea (18.66 y 12.6 por ciento), Brasil y España (7.29 y 6.04 por ciento respectivamente). En conjunto, estos cinco países concentraban 78.6 por ciento del mercado mundial.

1985		
1	Italia	28.49
2	Taiwan	18.66
3	Corea	12.16
4	Brasil	7.29
5	España	6.04
6	Francia	4.00
7	Alemania	2.84
8	Portugal	2.51
9	China	2.35
10	Hong-Kong	1.34
11	Reino Unido	1.23
12	Estados unidos	0.99
13	Holanda	0.81
14	México	0.57
15	Rumania	0.56
16	Tailandia	0.44
17	India	0.35
18	Belgica-Luxemburgo	0.26
19	Indonesia	0.04
20	Vietnam	0.00

1990		
1	Italia	21.80
2	China	14.67
3	Corea	14.45
4	Taiwan	10.20
5	Brasil	5.25
6	España	4.76
7	Portugal	4.30
8	Francia	2.79
9	Alemania	2.60
10	Tailandia	2.31
11	Indonesia	2.19
12	Estados unidos	1.37
13	Reino Unido	1.31
14	Hong Kong	0.98
15	Holanda	0.95
16	India	0.58
17	México	0.57
18	Rumania	0.28
19	Belgica-Luxemburgo	0.26
20	Vietnam	0.01

1998		
1	China	41.25
2	Italia	13.71
3	Indonesia	5.10
4	España	4.22
5	Portugal	3.75
6	Brasil	3.49
7	Vietnam	3.37
8	Tailandia	2.10
9	Alemania	1.81
10	Reino Unido	1.71
11	Taiwan	1.56
12	Francia	1.49
13	Corea	1.42
14	Belgica-Luxemburgo	1.41
15	Holanda	1.09
16	Estados Unidos	1.07
17	Hong-Kong	1.04
18	Rumania	0.96
19	India	0.80
20	México	0.78

Fuente: CECIC con datos del CAN 2000 (CEPAL)

Tabla 1.1 *Competitividad Mundial según exportaciones mundiales de calzado*

En la tabla 1.1 se observa que en tan solo 13 años, China paso del 9º sitio al 1º, acaparando un 41.25% del mercado mundial, Italia pasó del primero al segundo lugar (13.7 por ciento), España y Portugal (4.22 y 3.75 por ciento respectivamente), y Brasil en sexto lugar con 3.49 por ciento del mercado. México ocupaba el decimocuarto sitio en 1985 descendiendo para 1990 y a pesar de que en 1998 incrementó su participación en el mercado de calzado, descendió al vigésimo lugar debido al despunte de países como Indonesia, el cual transitó desde el lugar decimonoveno en 1985 al onceavo en 1990 y al tercero en 1998 (0.04, 2.19, llegando a un 5.10 por ciento del mercado respectivamente). En el año 2003 los primeros sitios fueron ocupados por China, India y Brasil.

1.1.2 Panorama nacional

Actualmente la industria del calzado nacional se encuentra en la problemática de haber descendido su producción de manera significativa en los últimos años, de acuerdo al Programa para la Competitividad de la Industria del Cuero y del Calzado [1], en los últimos años, las importaciones totales de calzado se han incrementado de forma considerable y de igual forma, se ha reducido la ganancia comercial del sector (tan solo en 2000 y 2001, las importaciones originarias de Asia aumentaron su participación en las importaciones totales de calzado de México en un 38 por ciento), como lo muestra la figura 1.1, destacando las importaciones de países como China, Indonesia, Vietnam y Tailandia.

Aunado a lo anterior, se tiene el problema del contrabando de calzado en nuestro país, provocando reducciones en la producción y empleo; reduciendo así el margen de maniobra de los apoyos institucionales al sector nacional del cuero y calzado, ya que el ingreso de cargamentos ilegales provenientes principalmente del continente asiático dificulta el repunte de la industria del calzado en México. Se estima que se dejan de pagar impuestos por más de 10 mil 500 millones de pesos de Impuesto al Valor Agregado (IVA) y otros mil 400 millones de pesos por concepto de Impuesto Sobre la Renta (ISR) por parte del calzado que entra de manera ilegal, por lo cual se deben realizar acciones efectivas que en verdad apoyen a la industria del calzado. El problema es tan grave que actualmente el contrabando ocupa la cuarta parte del mercado nacional, dicha parte en su mayoría consiste de calzado deportivo.

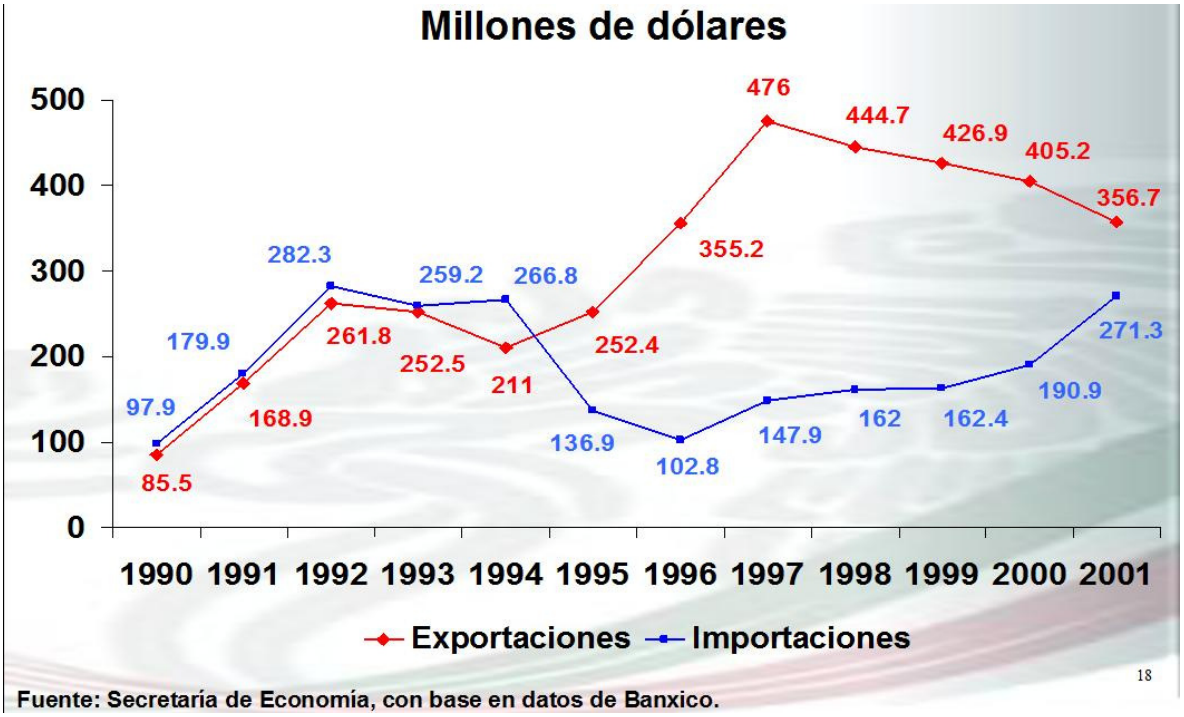


Fig. 1.1 *Importaciones y exportaciones de calzado México – mundo*

1.2 Definición del problema

El aumento acelerado de la participación del calzado de origen asiático en el mercado mexicano representa ya una amenaza a la industria de calzado nacional, por lo cual, las empresas nacionales de la industria del cuero y calzado para poder enfrentarse a la competencia global de forma exitosa, buscan mejorar sus procesos de producción para competir con los productos de Oriente.

Un proceso clave en la fabricación de calzado es el corte de piel. Actualmente, este proceso se lleva a cabo por un operador, quien ubica manualmente los suajes de corte sobre la piel y aplica presión con una máquina hidráulica para completar el suajado. Este proceso manual solo contempla un método de optimización muy subjetivo a cada operador, para el aprovechamiento de piel, cada operador requiere de cuatro a cinco años de experiencia y una vez que la adquiere, el operador tiene una gran destreza para realizar el acomodo, es por esto que en el campo laboral los operadores con experiencia son ampliamente solicitados y bien pagados; ya que si el almacén estima que para una orden de trabajo, se necesita una determinada área de piel, el operador ahorra de 2 a 3% adicional a lo contemplado por el almacén.

Fig. 1.2 *Empleado cortador de cuero*

También esta forma de corte no permite una flexibilidad en la línea de producción, dado que cualquier cambio en un modelo de calzado implica la fabricación de nuevos suajes correspondientes, lo cual implica tiempo y recursos económicos adicionales. Además, la logística de guardar, mantener y asignar los suajes a los puestos de corte también es complicada.

Así, los objetivos de automatizar el proceso de corte de piel son:

- Disminuir los costos del proceso de corte
- Aumentar la eficiencia en el aprovechamiento de la piel
- Disminuir el tiempo de corte
- Mejorar la calidad de las piezas cortadas

1.3 Estado del arte

1.3.1 Esquemas de producción de calzado

En el mundo de la producción de calzado, existen diferentes esquemas de competitividad, los cuales pueden ser adoptados por las empresas mexicanas, para participar con mayor impacto en el panorama internacional [1]; los esquemas más exitosos son:

1. **El modelo chino:** Su estrategia se basa en la potencialización del capital productivo, en la manufactura a escala masiva de productos estandarizados a mínimo costo con una organización gerencial de primer nivel. Lo anterior, combinado con una alianza estratégica con el capital comercial y financiero proveído por sus socios de Taiwán y Hong Kong quienes inyectan grandes inversiones de capital y tecnología moderna. En esta alianza, China contribuye con eficiencia en el abastecimiento de materias primas, componentes y manufactura además de que ha desarrollado una eficiencia óptima en la integración con sus socios que lideran los eslabones de logística y comercialización.

2. **El modelo italiano:** Se basa en la eficiencia, la integración dentro de eslabones de diseño e innovación con diferenciación de producto y marca, apoyándose en el capital organizacional de un distrito industrial o cluster y subcontratando las operaciones de manufactura intensiva en mano de obra y manteniendo la operación de montado y acabado del producto, con lo que mantiene buenos márgenes de ganancia.

3. **Modelo de la empresa multinacional:** Se basa en la especialización por línea de producto con producción a gran escala y controlando los eslabones de diseño e innovación de producto lo mismo que los de distribución, comercialización y venta. Cuenta con un esquema de eficiencia en la integración de abastecimiento de materias primas y componentes con subcontratación de manufactura, todo bajo mínimo costo de manufacturas y sobre todo de mano de obra.

Referente a los sistemas de manufactura en la industria del calzado nacional, actualmente se propone que para aumentar la competitividad la manufactura se lleve a cabo de acuerdo a la filosofía de los sistemas Just-in-Time (JIT) con las ventajas y defectos del mismo, el cual "se basa en la eliminación continua de todo lo que implique desperdicio (por desperdicio se entiende todo aquello que no añade valor al producto), esto se consigue llevando el material exacto al lugar necesario en el momento concreto, ni antes ni después. Cada operación está perfectamente sincronizada con las que le siguen para hacer posible este proceso, como lo comenta Álvarez [2]. Así, a nivel interno encontramos que las empresas de calzado cuentan con un esquema administrativo de buen nivel, pero el esquema de producción actual no les permite responder rápido a los cambios de la demanda. Aunque las empresas tienen esquemas mecanizados, varias operaciones unitarias básicas están basadas en la operación manual.

Uno de los inconvenientes que surgen en el actual enfoque manufacturero de la industria de calzado nacional es que en el mundo ya existen tendencias a cambiar los esquemas de producción para adaptarse a los cambios del mercado, ejemplo de esto es la tendencia a personalizar el calzado, como lo proponen el proyecto NIKEid y el esquema de proyecto EUROSHOE, los cuales aprovechan los avances tecnológicos existentes y se mencionan a continuación.

NIKEid.- Nos permite personalizar, diseñar y comprar en línea calzado de tipo deportivo, con las siguientes características: selección de los materiales del calzado, elección de colores, posibilidad de añadir un iD personal, corte estrecho y ancho, tallas independientes en pie izquierdo y derecho, elección de suela, tallas grandes así como tallas pequeñas; en un plazo máximo de entrega de un mes.

EUROSHOE.- Este proyecto se desarrolló del año 2002 a 2004 con apoyo financiero de la comisión europea reuniendo a 33 organismos de nueve países de la UE, entre universidades, centros de investigación y empresas. Se trata de la investigación y desarrollo de equipo para lograr la producción de calzado personalizado; para poner en práctica este proyecto, a los consumidores elegidos se les escaneó el pie, y ellos eligieron el modelo de zapato que querían, con las características que ellos lo deseaban, logrando un zapato completamente a medida del cliente, en un plazo máximo de entrega de 10 días.

Para que la industria nacional pueda competir a nivel global es necesario que se trabaje en la modernización de sus sistemas de producción; la automatización nos permitirá un aumento rápido en el volumen de producción, así como una mejora en los niveles de calidad, uniformidad y regularidad del producto con una gran flexibilidad de producción, características que están muy limitadas en el actual esquema manual.

1.3.2 Análisis de sistemas comerciales existentes

Actualmente existen dos sistemas de corte automatizado de piel para calzado, líderes en el mercado, Zipor (portugués) y Taglio (italiano); cuyas características principales se describen a continuación:

1.3.2.1 Zipor

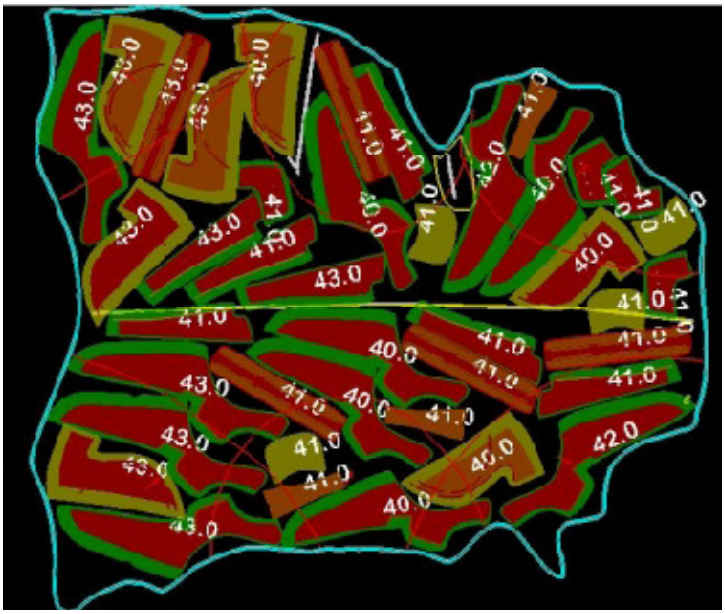
Zipor es un sistema de corte de piel, el cual consta principalmente de: una mesa de vacío en donde se lleva a cabo la digitalización de la piel; una estación de trabajo donde se efectúa el proceso de nesting automático y una máquina que realiza el corte de la piel por chorro de agua.

Entre sus características más sobresalientes se encuentran:

- Reducción del desperdicio de piel entre un 3% y 7%
- Producción estimada de 200 pares en 8 horas
- Costo bajo de mantenimiento
- Una alta calidad en el corte de las piezas
- Tiene diversas aplicaciones industriales como: interiores de aviones y automóviles, zapatos y muebles de piel.
- El sistema es capaz de digitalizar la piel a través de una cámara, para posteriormente vectorizar la imagen con una alta precisión, además de que es posible marcar defectos dentro de la misma (huecos).
- Se tiene un sistema de Administración de órdenes y materiales, cuya **base de datos** permite un control en tiempo real del almacén de pieles.
- Presenta reportes sobre los tipos de pieles empleadas

A pesar de que el nesting se lleva a cabo de manera automática, mostrando los resultados antes del corte, e inclusive puede cambiarse del modo automático al modo manual; de forma que el nesting sea realizado por un operador, el acomodo obtenido por el proceso de nesting es ineficiente; como se puede observar en la figura 1.3 b, el Sistema deja muchos huecos entre las piezas, no maneja líneas de estiramiento, siendo estas últimas junto con las zonas de calidad dos características fundamentales para la fabricación del calzado.

(a)



(b)

Fig.1.3 *Sistema Zipor (a), nesting del sistema (b); imágenes tomadas de [3]*

1.3.2.2 Taglio

El modelo CutVision 15-30 de la empresa Taglio es otro sistema para el corte automático de piel que permite:

- Posicionar inmediatamente los modelos en producción
- Tienen una gran flexibilidad de producción ya que se le puede pedir distintos estilos de diferentes tallas y una determinada cantidad de pares para cada estilo tan variada como se necesite.
- Ofrece una alta calidad en el corte de la piel
- Elevado rendimiento

Los módulos del sistema son:

- Detección de defectos y escaneo de la piel
- Nesting
- Cortado
- Administración

Este sistema puede operar in-line, siguiendo la secuencia: escaneo de la piel, nesting y corte; o bien off-line realizando la digitalización de la piel y el proceso de nesting en una etapa previa, para después únicamente identificar el acomodo con la piel que le corresponde y efectuar el corte.

En lo referente a la detección de defectos y escaneo de la piel, las diferentes áreas de calidad y defectos pueden ser marcadas de dos formas:

A través de la digitalización directa de las características de la piel por un operador, el cual se lleva a cabo en una tabla de marcado que puede ubicarse desde una posición horizontal hasta una inclinación vertical, ofreciendo un espectro completo de reflexión de luz sobre la piel.

La segunda opción es con el sistema de escaneo de la piel el cual reconoce automáticamente todos los detalles de la piel con una alta fidelidad en tan solo unos cuantos segundos. Se puede digitalizar piel de cualquier color sin mayores reconfiguraciones del sistema de escaneo, el proceso completo se lleva a cabo en tan solo 60 segundos en promedio.

Administración

El sistema cuenta con un software llamado CV Gest que permite tener control sobre el almacén de pieles y manejo de la orden de pedido. A través de este sistema de administración de producción, una vez que una orden ha sido emitida, es posible comenzar el proceso de nesting sobre un número de pieles almacenadas definido por el usuario.

Este proceso es ejecutado en un periodo de tiempo anterior a aquel en el que se introducen las pieles en el sistema de corte. Al no ser dependiente del tiempo de corte, el periodo de tiempo usado para el nesting es muy superior al tiempo que lleva el corte, para lograr la más alta optimización en cada piel, el nesting puede ser realizado simultáneamente por varias estaciones de trabajo para garantizar una continua y alta productividad para el sistema de corte.

Para optimizar el tiempo de trabajo, es necesario disponer de un sistema administrador de la orden de producción que sea eficiente. La tarea del sistema de administración del CutVision es preparar los modelos para el corte, importar los diseños y convertirlos de un formato de archivo CAD al formato que maneja el nesting.

Cuando se recibe la correspondiente orden de producción desde el sistema de administración de orden de la empresa; el sistema de administración CV Gest automáticamente optimiza las work stations con las que cuenta, de esta forma el sistema asegura que no habrá tiempos muertos y ofrece una inserción inmediata de la orden de producción.

CV Gest se encarga de la administración de archivos CAD y la preparación de órdenes para la estación de corte. Para la integración con el sistema de administración de la empresa, CV Gest provee un gran conjunto de formatos de intercambio de datos. CutVision Gest es capaz de operar como una única estación, o bien puede ser implementado en una red de varias work stations, servidores y máquinas de corte. CV Gest también puede ser configurado para servir como un sistema warehouse, ofreciendo una eficiente y certera administración del almacén de pieles. El programa de nesting automático está completamente integrado en el sistema.

La detección de todos los defectos de la piel se realiza durante el proceso de *evaluación de calidad*, una vez que este finaliza la piel es puesta sobre la estructura de trabajo (*zona de cargado*) , la cual se encarga de llevarla dentro de la *zona de escaneo y nesting* donde el contorno de la piel es escaneado y el nesting comienza a realizarse, luego la piel se sitúa en el *área de corte* donde el corte por chorro de agua se lleva a cabo, para finalizar en la *zona de descarga* donde las piezas y los residuos de la piel son retirados; cabe mencionar que todas estas etapas se llevan a cabo en una misma máquina y que el sistema ha sido diseñado para un uso industrial, trabajando 24 horas al día los 7 días de la semana.

A pesar de que este sistema logra una mejor administración de los archivos que definen la geometría de las piezas, el almacén de pieles y el proceso de corte en general; el nesting del sistema sigue siendo muy ineficiente ya que no considera las líneas de estiramiento de la piel, en la figura 1.4 b, se ven dos chinelas acomodadas a 90° lo cual es contrario a las restricciones de la empresa; el sistema Taglio pone un mayor énfasis a la rapidez con la cual se realiza el nesting, disminuyendo el aprovechamiento de la piel al dejar muchos huecos.

(a)

(b)

Fig. 1.4 *Sistema Taglio (a), nesting del sistema (b); imágenes tomadas de [4]*

1.4 Objetivo

El presente trabajo se enmarca dentro del proyecto de nesting y su finalidad es:

- Realizar un sistema que de forma automatizada, obtenga la información necesaria para que sea posible el acomodo de las piezas de un modelo de zapato en las hojas de piel necesarias para cumplir con una orden de trabajo común.
- La selección de la información de las pieles a utilizar sea la adecuada para minimizar el desperdicio de piel.
- Los resultados del proceso permitan obtener información que sea de utilidad para la directiva de la empresa.

Esta información se encontrará en diferentes equipos de cómputo y su generación depende de otros sistemas.

1.4.1 Hipótesis

Es posible desarrollar un sistema de información alojado en distintos equipos, capaz de almacenar, obtener y administrar la información necesaria para llevar a cabo el proceso de nesting. El sistema estará conformado por programas desarrollados en diferentes lenguajes que operarán en diferentes tiempos para ajustarse al proceso de producción de la empresa.

1.5 Alcance

El presente trabajo tiene por alcance, realizar un sistema que obtenga la información necesaria (orden de trabajo, piezas del modelo de zapato requerido, pieles a procesar para cubrir la cantidad de zapatos especificados en la orden de trabajo, etc.) y proporcionar dicha información al programa que se encargará de realizar el nesting.

1.6 Metodología

La metodología empleada durante la realización del presente trabajo es la siguiente:

- Análisis del problema
- Determinar las Necesidades y especificaciones a cumplir
- Análisis y selección de la alternativa de solución
- Diseño de la solución seleccionada
- Determinar y seleccionar las herramientas de hardware software
- Desarrollo de la solución
- Pruebas
- Implantación
- Conclusiones

Capítulo 2 Análisis de las alternativas de solución

2.1 Esquema del sistema de nesting

El sistema de nesting, del cual forma parte el Sistema de Información de Nesting (SIN), tiene como objetivo realizar el acomodo de las piezas de calzado sobre una piel virtual de acuerdo a ciertas restricciones, llamado proceso de nesting en lo sucesivo, para dicho proceso los elementos son suministrados por el sistema de información de nesting.

El usuario directo del Sistema de Información es el Sistema de Nesting, cuyo esquema es el siguiente:

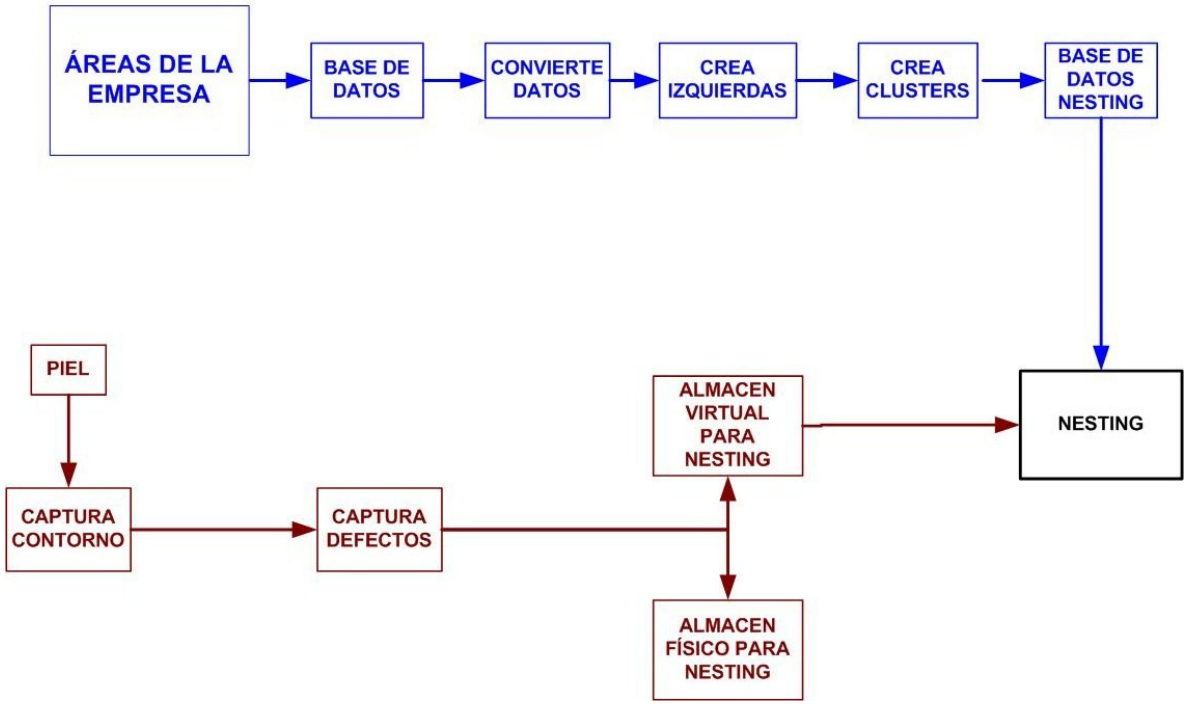


Fig. 2.1 *Esquema del sistema de Nesting*

2.2 Información necesaria para el proceso de Nesting

Piel virtual.- Es un archivo de **Di**gitalización de una hoja de **Pi**el (**DiPi**), en el cual se incluyen sus características, como lo son, zonas de calidad, dirección de estiramiento y defectos.

Pieza.- Es la digitalización de cada una de las piezas que conforman el zapato.

Cluster.- Es la agrupación de dos piezas para la optimización de un área local

Modelo.- Se refiere al nombre con que se identifica a cada estilo y color de calzado, ejemplo: Ofelia_negro, Ofelia_cafe.

2.3 Sistema de información de nesting

El Sistema será capaz de almacenar, obtener y administrar toda la información necesaria para efectuar el proceso de Nesting.

2.3.1 Requisitos del sistema de información

El sistema debe ser capaz de:

- Almacenar la información de los elementos necesarios para el nesting
- Mantener la integridad de dicha información
- Ser compatible para interactuar con los diferentes módulos del Sistema de Nesting
- Proveer la información necesaria para el proceso de nesting de forma ágil y segura
- Almacenar los resultados del proceso en forma de Reportes.
- En algunos casos interactuar con el usuario para obtener y corroborar información

Para lo anterior se propone:

- Un sistema que contenga una Base de Datos (BD) para almacenar la información.
- La posibilidad de acceder a la información de manera remota a través de una LAN.
- Acceso a la BD desde diferentes lenguajes de programación.
- Desarrollar programas para interpretar la información

2.4 Metodologías de desarrollo de software

Una vez expuestos los componentes de nuestro sistema, debemos indagar las formas y metodologías con las cuales podemos desarrollarlo. Para esto, analizaremos las metodologías de desarrollo de software, los modelos de proceso y los modelos de datos.

Primeramente para el desarrollo del software del sistema, se analizan dos metodologías; el Diseño Modular y el Diseño Orientado a Objetos.

2.4.1 Diseño modular

El *diseño modular* es uno de los métodos de diseño más flexible y potentes para mejorar la productividad de un programa. Dividiendo un programa en módulos donde cada uno de ellos ejecuta una única actividad o tarea, lo cual permite analizar, codificar, probar y optimizar los módulos por separado; todo programa bajo esta metodología tiene un módulo controlador el cual se encarga de llamar a los demás módulos y controlar el flujo del programa. Cuando es necesario, se transfiere el control a submódulos de modo que éstos

puedan ejecutar sus funciones regresando el control al módulo controlador o programa principal.

Como lo menciona Joyanes [5], si la tarea asignada a cada submódulo es demasiado compleja, éste deberá romperse en otros módulos más pequeños. El proceso sucesivo de subdivisión de módulos continúa hasta que cada módulo tenga solamente una tarea específica que ejecutar. Un módulo puede transferir temporalmente el control a otro módulo; sin embargo, cada módulo debe eventualmente devolver el control al módulo del cual se recibe originalmente el control. Los módulos son independientes en el sentido en que ningún módulo puede tener acceso directo a cualquier otro módulo excepto el módulo al que llama y sus propios submódulos. Sin embargo, los resultados producidos por un módulo pueden ser utilizados por cualquier otro módulo cuando se transfiera a ellos el control. Dado que los módulos son independientes; diferentes programadores pueden trabajar simultáneamente en diferentes partes del mismo programa. Esto reducirá el tiempo del diseño del algoritmo y posterior codificación del programa. Además, un módulo se puede modificar radicalmente sin afectar a otros módulos, incluso sin alterar su función principal. La descomposición de un programa en módulos independientes más simples se conoce también como el método de "*divide y vencerás"* en el cual se diseña cada módulo con independencia de los demás, y siguiendo un método ascendente o descendente se llegará hasta la descomposición final del problema en módulos en forma jerárquica.

2.4.2 Diseño orientado a objetos

El *Diseño Orientado a Objetos* (DOO) es una metodología que tiene la capacidad de modelar el software en términos similares a los que se utilizan para describir objetos en el mundo real. Como se manifiesta en [6], este diseño aprovecha las relaciones entre las *clases*, en donde los objetos de cierta clase tienen las mismas características. También aprovecha las relaciones de *herencia*, en donde las nuevas clases de objetos "heredan" las características de las clases existentes y además agregan sus propias características. El DOO ofrece una manera más natural e intuitiva de ver el proceso de diseño: modelando los componentes de software de igual forma que como describimos los objetos del mundo real teniendo en cuenta sus atributos y comportamientos. El DOO también modela la comunicación entre los objetos. Así como las personas se envían mensajes unas a otras, los objetos también se comunican mediante mensajes. El DOO *encapsula* los atributos y las operaciones (comportamiento) en los objetos; los objetos tienen una propiedad que se conoce como ocultamiento de la información. Esto significa que, aunque los objetos saben cómo comunicarse entre sí a través de interfaces bien definidas, generalmente no se les permite saber cómo se implementan otros objetos; ya que los detalles de la implementación se ocultan dentro de los mismos objetos.

Cuando el software se empaqueta en forma de clases, éstas pueden reutilizarse en sistemas de software posteriores, así los grupos de clases relacionadas se empaquetan comúnmente como componentes reutilizables; permitiendo que los programas que ocupen las mismas estructuras de información reutilicen las definiciones de objetos y los procedimientos que los manipulan; que fueron diseñados y empleados en otros programas o que son parte de la biblioteca de clases del lenguaje de programación que estemos empleando para desarrollar el código. De esta forma, el desarrollo de un programa puede

llegar a ser una simple combinación de objetos ya definidos, ya sea por el programador o por la biblioteca de clases, donde estos están relacionados de una manera particular. Finalmente podemos mencionar que esta metodología de desarrollo de software engloba a la programación estructurada.

2.5 Modelos de proceso de software

Para la planificación y diseño de nuestro Sistema de Información, es necesario definir un modelo de proceso de software el cual nos guiara en la planeación y desarrollo del software definiendo qué hacer, cómo y cuándo; durante todo el desarrollo y mantenimiento de un proyecto, nos ayuda a la comprensión del problema, facilita el mantenimiento del producto final y permite la reutilización de partes del producto. Además optimiza el proceso y los productos de software que son los programas, documentos y datos producidos, todo esto gracias a las actividades de ingeniería del software.

2.5.1 El Modelo lineal secuencial

También llamado ciclo de vida básico o modelo en cascada, describe un enfoque sistemático y secuencial para el desarrollo del software, realizando el análisis, diseño, codificación, pruebas y mantenimiento del sistema a realizar; las etapas que componen a este modelo son cinco:

- Análisis de los requisitos del software. Esta etapa debe realizarse para comprender la naturaleza del problema y tener en claro el programa que se va a construir, para comprender el dominio de información del software, la función requerida, comportamiento, rendimiento e interconexión. Ésta última característica es la que más interesa para nuestro sistema de información ya que durante esta etapa se especificará quien será el usuario directo, así como los sistemas o programas con los que se comunicará nuestro Sistema de Información.

- Diseño. El diseño del software es realmente un proceso de muchos pasos que se centra en cuatro atributos distintos del programa: estructura de datos, arquitectura de software, representaciones de interfaz y algoritmo. Esta etapa traduce los requisitos en una representación del software anterior a la codificación (diagramas de flujo o diagramas de clases) [7].

- Generación de código. Si la etapa anterior se desarrolló cuidadosamente, la generación de código en el lenguaje de desarrollo que se elija, se llevará a cabo casi mecánicamente.

- Pruebas. Para la detección de errores, se debe probar la lógica interna de los programas, así como sus métodos para asegurar que a una entrada específica se obtendrá una salida válida dentro de los resultados especificados. Además de probar el funcionamiento con entradas de datos no válidas.

- Mantenimiento. El software realizado sufrirá cambios debido a las mejoras funcionales o de rendimiento que se requieran. Por lo cual, las etapas anteriores se aplicarán al software actual y no a uno nuevo.

Aunque el modelo lineal secuencial es el más utilizado, los sistemas del mundo real no siguen este modelo ya que no se permiten las iteraciones y tiene como limitaciones que los requisitos se congelan al principio del proyecto y no existe un proyecto "enseñable" hasta el final del mismo.

2.5.2 El modelo de construcción de prototipos

Generalmente el cliente define los objetivos del sistema, sin detallar las entradas y salidas de los procesos, el ingeniero de software podría no estar seguro de la capacidad de solución de los algoritmos que empleará, de que el programa se adapte bien al sistema operativo o de las suposiciones que está haciendo entre la interface hombre-máquina; en este tipo de situaciones el enfoque de construcción de prototipos es el más adecuado. Este paradigma comienza con la recopilación de requisitos de software con el cliente, definiendo los objetivos globales para el software y entonces el desarrollador crea un diseño rápido representando las características que serán visibles tanto para el cliente como para el usuario del sistema. El diseño rápido es el punto de partida para la construcción de un prototipo, el cual es evaluado por el cliente y por los usuarios y al interactuar los usuarios con este prototipo se van especificando más a fondo los requisitos del software a desarrollar. El prototipo evoluciona iterativamente al irse mejorando para satisfacer las necesidades del cliente, lo que permite que el desarrollador comprenda de mejor forma lo que se le está pidiendo.

De manera ideal el modelo de construcción de prototipos debe de ser un mecanismo para identificar los requisitos de software, es por esto que cuando el ingeniero de software no tiene muy clara la solución más adecuada se construye un prototipo de una parte o bien de todo el sistema. Teniendo como ventajas: verificar los requisitos principales del usuario durante toda la realización del sistema reduciendo la divergencia de expectativas entre el cliente y el desarrollador, al igual que permite verificar la viabilidad del diseño del sistema. En general, este modelo no se diseña para entregar un sistema de producción.

2.5.3 El modelo DRA

El Desarrollo Rápido de Aplicaciones (DRA) es un modelo de proceso de desarrollo de software que pone énfasis en un tiempo de desarrollo extremadamente corto. Este modelo es una adaptación del lineal secuencial considerando una alta velocidad de desarrollo utilizando una construcción basada en componentes. Si en un proyecto se entienden bien los requisitos y se exponen claramente los límites del proyecto, el modelo DRA permite a los desarrolladores crear un sistema completamente funcional en poco tiempo.

Cuando se utiliza para aplicaciones de sistemas de información, éste modelo mantiene cinco etapas:

- Modelado de Gestión. En esta etapa se modela el flujo de información entre las funciones de administración de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la procesa?

- Modelado de datos. El flujo de información obtenido en la etapa previa se refina como un conjunto de objetos necesarios para apoyar a la empresa definiendo los atributos de cada uno de los objetos y las relaciones entre ellos

- Modelado del proceso. Se crean las descripciones de proceso para añadir, recuperar, o suprimir un objeto de datos.

- Generación de aplicaciones. El proceso DRA trabaja utilizando componentes de programas ya existentes (reutilización de código) o bien crea componentes reutilizables cuando es necesario.

- Pruebas y entrega. Debido a que el proceso DRA se basa en la reutilización, la mayoría de los componentes del nuevo sistema ya han sido probados, de manera que se facilita la etapa de pruebas y se reduce su duración limitándose a probar todos los componentes nuevos.

Como desventajas del modelo DRA se tiene que para proyectos grandes se requiere de recursos humanos suficientes para cada uno de los equipos DRA, no es adecuado cuando los riesgos técnicos son altos; por ejemplo, cuando se desarrolla el sistema empleando nuevas tecnologías.

2.5.4 El modelo incremental

El modelo incremental combina los componentes del modelo lineal secuencial con la filosofía de interacción del modelo de construcción de prototipos aplicando secuencias lineales de forma escalonada a medida que va avanzando el tiempo de duración del proyecto, en donde cada secuencia lineal produce un incremento de software que puede ser entregado al cliente, para mostrarle el avance en el desarrollo del sistema de software. Este modelo entrega el software en partes pequeñas pero utilizables y cada incremento se realiza sobre el anterior que ya ha sido entregado y aprobado. Cuando se le muestra un incremento al cliente, una vez hecha la evaluación y/o utilización se crea un plan para el incremento siguiente para cumplir de mejor manera las necesidades del cliente y la entrega de funciones y características adicionales. El proceso se repite en cada una de las entregas de cada incremento, hasta llegar a la entrega del producto completo.

El modelo incremental es iterativo y se centra en la entrega de un producto operacional con cada incremento. Siendo los primeros incrementos versiones incompletas del producto final, pero proporcionan al usuario cierta funcionalidad y también una plataforma para la evaluación [7]. Cuando se tiene una fecha de entrega imposible de cambiar o no se dispone del número de programadores necesarios para realizar una entrega completa, el modelo incremental es una buena elección.

2.6 Modelos de datos

2.6.1 Bases de datos relacionales

El modelo de datos relacional (MR) aísla la representación de los datos de su estructura física consiguiendo que la representación de los datos sea independiente de la que exige un Sistema Gestor de Base de Datos (SGBD) para su implementación. El MR está basado en la teoría matemática de las relaciones, por lo que la relación es la estructura básica y única, la integridad referencial se mantiene mediante el uso de llaves foráneas; así, el MR tiene tres características principales [8]:

- Sencillez y uniformidad: debido a que los usuarios ven la base de datos relacional como un conjunto de *tablas*, y al ser la tabla la estructura fundamental del modelo, se obtiene una gran uniformidad.
- Fundamentación teórica robusta: Ya que el MR está basado en fundamentos matemáticos (relaciones), el diseño y la evaluación pueden realizarse de manera sistemática basada en abstracciones.
- Independencia de la interfaz de usuario: Para manipular la información necesitamos un lenguaje relacional el cual manipula conjuntos de registros (resultset) brindándonos una gran independencia respecto a la manera en que los datos están almacenados.

El lenguaje más común para construir las consultas a bases de datos relacionales es el Structured Query Language (SQL) en el cual sólo hay que indicar lo qué se desea que el SGBD devuelva, sin preocuparse del cómo, es decir, es un lenguaje declarativo.

El MR responde a la parte lógica del diseño de base de datos y una gran ventaja que presenta es que se ha logrado un gran estándar, así el esquema conceptual de ANSI será el esquema relacional y los esquemas externos corresponden a las vistas, además de que en él se basan la mayoría de los SGBD. La desventaja de este modelo es que sólo maneja campos atómicos y para ciertas aplicaciones este enfoque puede ser insuficiente.

2.6.2 Bases de datos orientadas a objetos

Debido a que las bases de datos relacionales usan datos conceptualmente simples, varias áreas de aplicación de los sistemas de bases de datos están limitadas por las restricciones del modelo de datos relacional. El modelo de Base de Datos Orientado a Objetos (BDOO), está basado en el paradigma de los lenguajes de programación orientada a objetos y presenta una solución a la necesidad de las aplicaciones de manejar datos más complejos ya que no tiene limitación debida a los tipos de dato. En las BDOO los elementos de datos son objetos agrupados en clases que en general, se corresponden con las entidades del diagrama Entidad-Relación (E-R) y cada objeto está compuesto por:

- Variables
- Mensajes que forman la interface entre los objetos y el resto del sistema
- Métodos

Si bien la herencia y los tipos complejos son parte del modelo E-R; el encapsulamiento y la identidad de objetos hacen la diferencia entre el modelo de datos orientado a objetos y el modelo E-R. En el modelo relacional, las relaciones son las tablas referenciadas mediante llaves foráneas, por el contrario en el modelo OO (puro), las relaciones se implementan incluyendo en cada objeto los identificadores de los otros objetos con los que se relaciona *(inclusión lógica)* y dichos identificadores son atributos que posee cada objeto y son asignados por el Sistema Gestor de Base de Datos Orientado a Objetos (SGBDOO) y es el único que los utiliza. Para implementar una BDOO se requiere de un lenguaje de programación orientado a objetos extendido que permita la persistencia de objetos (los datos siguen existiendo una vez que el programa que los creó termina de ejecutarse), C++ o Java presentan estas características, pero la desventaja de esto es que es relativamente sencillo que un error en la programación dañe la base de datos además de dificultar la optimización automática de alto nivel (recolección de basura). La manera de obtener un objeto desde una BDOO puede realizarse de tres formas: Dando nombres a los objetos, exponiendo los identificadores de los objetos o los punteros persistentes, guardando las colecciones de objetos y permitiendo que los programas iteren sobre las mismas para hallar los objetos deseados. Las consultas devuelven objetos y se realizan con el Lenguaje de Consulta de Objetos, el cual es declarativo y similar al SQL. Como ejemplo de un SGBDOO se encuentra db4o de Versant.

La principal ventaja de una BDOO es que *los objetos pueden estar compuestos de cualquier tipo de información*, por ejemplo imágenes, sonido y planos arquitectónicos complejos; así como los tradicionales tipos de datos alfanuméricos; otra ventaja es la flexibilidad que tiene el diseñador para especificar y extender objetos complejos así como los métodos que se les puede aplicar a esos objetos. La manipulación de los objetos es de forma ágil y rápida. Por el contrario, las principales desventajas son que los SGBDOO no están ampliamente difundidos y que no existe un verdadero estándar en la industria OO.

2.6.3 Bases de datos objeto-relacionales

Los modelos de datos relacionales orientados a objetos buscan mejorar la representación de los datos y su acceso mediante la programación orientada a objetos pero manteniendo el modelo relacional, es decir extienden al modelo relacional; proporcionan tipos de datos complejos, todo este enfoque se encuentra definido bajo la norma SQL:1999 que por un lado mantiene el lenguaje de consulta declarativo y por el otro extiende la capacidad de diseño, en SQL:1999 se utilizan los tipos Colección (Arrays, Conjuntos y Multiconjuntos que incluyen relaciones anidadas logrando relaciones que no cumplen con la 1ª Forma Normal) y los tipos estructurados (una lista de atributos compuestos de los diagramas E-R), en conjunto los tipos colección y los tipos estructurados permiten que los atributos de las tablas sean colecciones; se permite la definición de funciones, procedimientos y métodos ya sea con el componente procedimental de SQL:1999 o mediante un lenguaje de programación orientada a objetos como Java o C++, lo cual es muy significativo ya que con las funciones podemos hacer un procesamiento directo sobre los datos. Como ventajas de este modelo están la búsquela de la simplificación de los modelos de datos y de las consultas usando tipos de datos complejos, brinda una mayor protección a los datos y lenguajes de consulta potentes. Como ejemplos de Sistemas Gestores de Base de Datos

Comerciales que nos permiten crear un modelo objeto-relacional son: PostgreSQL, Informix, Oracle y DB2.

Los sistemas de bases de datos objeto relacionales proporcionan un modo de cambio adecuado para los usuarios de las bases de datos relacionales que deseen utilizar características orientadas a objetos [9].

2.6.4 Bases de datos distribuidas

Una base de datos distribuida (BDD) es un conjunto de múltiples bases de datos lógicamente relacionadas, las cuales se encuentran distribuidas entre diferentes sitios interconectados por una red de comunicaciones, los cuales tienen la capacidad de procesamiento autónomo de forma que un usuario en cualquier sitio puede acceder los datos en cualquier parte de la red como si los datos estuvieran siendo accedidos de forma local.

En un sistema distribuido de bases de datos se almacena la información en varias computadoras, por lo cual los principales factores que distinguen a un Sistema de Base de Datos Distribuido (SBDD) de un sistema centralizado son los siguientes:

- Hay múltiples computadores, llamados sitios o nodos.
- Estos sitios deben de estar comunicados por medio de algún tipo de red de comunicaciones para transmitir datos y órdenes entre los sitios.

El Sistema Gestor de Base de Datos Distribuidas (SGBDD) está formado por las transacciones y los gestores de la base de datos distribuida e implica un conjunto de programas que operan en diversas computadoras, estos programas pueden ser subsistemas de un único SGBDD de un fabricante o podría consistir de una colección de programas de diferentes fuentes.

Existen cuatro alternativas principales para el posicionamiento de los datos en una BDD y estas son: centralizada, replicada, fragmentada, e híbrida. La forma centralizada es muy similar al modelo de Cliente/Servidor en el sentido que la BDD está centralizada en un lugar y los usuarios están distribuidos. El esquema de BDD de replicación consiste en que cada nodo debe tener su copia completa de la base de datos. Es fácil ver que este esquema tiene un alto costo en el almacenamiento de la información. Debido a que la actualización de los datos debe ser realizada en todas las copias, también tiene un alto costo de escritura, pero todo esto vale la pena si tenemos un sistema en el que se va a escribir pocas veces y leer muchas, y dónde la disponibilidad y fiabilidad de los datos sea de máxima importancia.

En el modelo particionado consiste en que solo hay una copia de cada elemento, pero la información está distribuida a través de los nodos. En cada nodo se aloja uno o más fragmentos disjuntos de la base de datos. Como los fragmentos no se replican esto disminuye el costo de almacenamiento, pero también sacrifica la disponibilidad y fiabilidad de los datos.

Ventajas

- Refleja una estructura organizacional, los fragmentos de la base de datos se ubican en los departamentos a los que tienen relación.
- Autonomía local, un departamento puede controlar los datos que le pertenecen.
- Disponibilidad, un fallo en una parte del sistema solo afectará a un fragmento, en lugar de a toda la base de datos.
- Rendimiento, los datos generalmente se ubican cerca del sitio con mayor demanda, también los sistemas trabajan en paralelo, lo cual permite balancear la carga en los servidores.
- Modularidad, se pueden modificar, agregar o quitar sistemas de la base de datos distribuida sin afectar a los demás sistemas (módulos).

Desventajas

- Complejidad, se debe asegurar que la base de datos sea transparente, se debe lidiar con varios sistemas diferentes que pueden presentar dificultades únicas. El diseño de la base de datos se tiene que trabajar tomando en cuenta su naturaleza distribuida, por lo cual no podemos pensar en hacer joins que afecten varios sistemas.
- Seguridad, se debe trabajar en la seguridad de la infraestructura así como cada uno de los sistemas.
- Integridad, se vuelve difícil mantener la integridad, aplicar las reglas de integridad a través de la red puede ser muy caro en términos de transmisión de datos.
- Carencia de estándares, aún no existen herramientas o metodologías que ayuden a los usuarios a convertir un SGBD centralizado en un SGBD distribuido.
- El diseño de la base de datos se vuelve más complejo, además de las dificultades que generalmente se encuentran al diseñar una base de datos, el diseño de una base de datos distribuida debe considerar la fragmentación, replicación y ubicación de los fragmentos en sitios específicos.

2.7 Selección de la metodología de desarrollo, modelo de proceso y modelo de datos

El diseño y programación del software se llevará a cabo mediante la metodología de desarrollo **orientada a objetos** ya que nos permite una mejor representación del mundo real y concretamente de nuestro problema, la reutilización de código facilita la programación y construcción de prototipos, además de que agiliza el tiempo de desarrollo del software. Otro aspecto importante es que los actuales lenguajes orientados a objetos manejan un driver para llevar a cabo las consultas SQL, por lo que permite una gran flexibilidad para migrar a otra base de datos, ya que solo se tendrían que migrar los datos hacia otro SGBD y en los programas que realicen las transacciones, declarar el driver del Sistema Gestor de la Base de Datos con la que se conectará.

En cuanto al modelo de proceso de software que se seguirá durante nuestro diseño, se utilizará el **modelo incremental,** ya que se considera es el que mejor se adapta a la manera en que se llevará a cabo la interacción con el usuario, en virtud de que cada mes se efectuará una junta con la empresa para mostrarle el avance del proyecto y retroalimentar la especificación de requisitos de software.

Para la base de datos, aunque una base de datos relacional orientado a objetos nos permitiría una mayor flexibilidad en el diseño, por ejemplo podríamos almacenar en la base de datos un tipo de dato polígono, el cual será muy frecuentemente usado en nuestro sistema de información, y asociarle directamente algunas funciones, como en el caso de una función que compruebe si dos polígonos se intersectan lo cual nos llevaría a detectar las zonas de estiramiento de la piel; pero dado que la base de datos de nuestro sistema no tendrá muchas relaciones, sin embargo, sí tendrá un gran número de registros, se adoptará el **modelo relacional** para su diseño, el cual ya conocemos y contamos con cierta experiencia.

2.8 Análisis de los lenguajes de programación

Para desarrollar el software se consideró el uso de los siguientes lenguajes de programación: C++, Python y Java y se realizó un análisis de ventajas y desventajas sobre estos lenguajes:

2.8.1 C++

Ventajas:

- Al compilarlo, se genera código objeto, nativo de cada máquina. Resultado: C++ es más rápido que los lenguajes interpretados.
- Permite un control total de la memoria y una capacidad de programación de bajo nivel impensable en Java.
- Es un lenguaje que permite crear aplicaciones de gran escala.

Desventajas:

- No es multiplataforma. Para lograr aplicaciones que se ejecuten en varios SO, se requiere de cierto esfuerzo.
- No presenta una arquitectura estándar de desarrollo orientado a Internet.
- No presenta un toolkit tan rico como el de Java. Aunque hay muchas librerías en la red para C++, no son estándar del lenguaje, y algunas son de pago.

2.8.2 Python

Ventajas:

- Python es un lenguaje muy "expresivo", es decir, los programas son muy compactos, un programa en Python suele ser bastante más corto que su equivalente en lenguajes como C++ y Java.
- Es muy legible, la sintaxis de python es muy elegante y permite la escritura de programas cuya lectura resulta fácil, en comparación con otros lenguajes.
- Puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos.
- Dinámicamente tipificado, ya que una instrucción de asignación envuelve un nombre que hace referencia a un objeto que puede ser de cualquier tipo, si un nombre es asignado a un objeto de cierto tipo, posteriormente este puede ser asignado a un objeto de un tipo diferente.
- Al ser un lenguaje interpretado los programas son multiplataforma.

Desventajas:

- Ningún IDE te permite productividad al hacer aplicaciones graficas (ventanas, web, etc).
- No es tan rápido como C++ en el cálculo de números, lo cual lo limita en aplicaciones de alto rendimiento.
- Requiere un intérprete para correr la aplicación.
- Carencia de un verdadero soporte para multiprocesador.

2.8.3 Java

Ventajas:

- La mayoría de los programadores en Java aprovechan las ricas colecciones de clases existentes en las *bibliotecas de clases de Java,* que también se conocen como *APIs (Interfaces de programación de aplicaciones) de Java* [6]. Por lo tanto, soporta el desarrollo rápido de aplicaciones, y muchas de las tareas de un programador están resueltas en sus APIs.
- Su enfoque de programación orientado a objetos proporciona una mayor flexibilidad, modularidad y reutilización de código.
- Independencia de la Base de Datos a través del JDBC.
- El compilador, intérprete, y la ejecución contienen varios niveles de seguridad que están diseñados para reducir el riesgo de comprometer la seguridad, la pérdida de datos y la integridad del programa.

- En Java, la programación multihilo se ha integrado, brindando la capacidad para realizar varias tareas ejecutando diferentes líneas de código al mismo tiempo.
- Java tiene un manejo eficaz de la memoria de la computadora, de tal forma que el programador no tiene que preocuparse por apuntadores, liberación de memoria que no se esté utilizando, etc.

Desventajas:

- Una vez que el código Java es compilado en byte code, la Máquina Virtual Java, ejecuta el programa, así siendo Java un lenguaje interpretado es más lento que C++.
- Estáticamente tipificado. En Java, todos los nombres de variables junto con sus tipos deben ser declarados explícitamente.
- No es compacto; es abundante en palabras, sus instrucciones contienen varias palabras, dificultando la legibilidad del código.

Lenguaje	Ventajas	Desventajas
C++	• Mayor rapidez, en comparación con los lenguajes interpretados. • Manejo total de la memoria • Aplicaciones de gran escala	• No es multiplataforma. • En C/C++ se puede manejar la memoria casi sin ninguna restricción. • No presenta una arquitectura estándar de desarrollo orientado a Internet.
Python	• Dinámicamente tipificado. • Lenguaje muy expresivo • Multiplataforma • Muy rápido de programar.	• Carencia en el soporte para multiprocesador. • No es tan rápido como C++ • Falta de robustez para aplicaciones empresariales.
Java	• Amplia biblioteca de clases (APIs) • Es independiente de la plataforma • Independencia del SGBD (JDBC) • Programación multihilo	• Estáticamente tipificado, todas las variables deben ser explícitamente declaradas y no puede asignárseles un tipo de dato diferente; lo que dificulta la programación. • Al ser interpretado, se pierde velocidad en tiempo de ejecución.

Tabla 2.1 *Comparativa de ventajas y desventajas de lenguajes de desarrollo.*

2.9 Análisis de los sistemas gestores de base de datos

Para la base de datos se realizó el análisis de los siguientes Sistemas Gestores de Base de Datos: SQL Server, Oracle y PostgreSQL.

2.9.1 SQL Server

Ventajas:

- SQL Server asegura la continuidad empresarial, ya que incluye características de creación de mirroring de bases de datos, para aumentar la fiabilidad de las aplicaciones y simplificar la recuperación de éstas en caso de error de almacenamiento.
- Minimiza la supervisión administrativa. Declarative Management Framework (DMF) es un nuevo marco de administración basado en directivas en SQL Server 2008, que simplifica las operaciones de mantenimiento cotidiano y reduce el coste total de propiedad al definir un conjunto común de directivas para la mayoría de las operaciones con bases de datos.
- Integra cualquier tipo de datos. SQL Server 2008 ofrece un rendimiento mejorado de las consultas y un almacenamiento de datos eficiente y rentable que le permite administrar y redimensionar grandes cantidades de usuarios y de datos.
- Tiene un costo menor que otros Sistemas Gestores de Base de Datos, pero su facilidad de uso y la tendencia de los directivos a aceptar preferentemente productos de Microsoft le dan una potencia y calidad que lo hacen una buena opción como backend de publicaciones Web de cierto tamaño, aplicaciones internet u offline, y la mayoría de aplicaciones de media escala, con volúmenes no excesivos.

Desventajas:

- Sólo corre sobre plataformas Microsoft
- No es tan robusto como DB2 o Oracle
- La licencia tiene un costo

2.9.2 Oracle

Ventajas:

- Oracle corre en cualquier plataforma con gran escalabilidad y estabilidad, se puede migrar en el momento que se quiera.
- Oracle maneja bases de datos de cientos de Giga bytes, con varios cientos de usuarios concurrentes, con una respuesta y control muy buena.
- Se pueden desarrollar pequeñas aplicaciones con el Oracle Express Database (OracleXE), basada en la versión Oracle 10gR2, esta es una base de datos gratuita y enfocada a pequeños grupos de usuarios, en la cual el instalador es muy simple.
- Soporta todas las funciones que se esperan de un servidor "serio": un lenguaje de diseño de bases de datos muy completo (PL/SQL) que permite implementar diseños

"activos", con triggers y procedimientos almacenados, con una integridad referencial declarativa bastante potente.

Desventajas:

- El mayor inconveniente de Oracle es su precio. Incluso las licencias de Personal Oracle son caras.
- Otro problema es la necesidad de ajustes. Un error frecuente consiste en pensar que basta instalar el Oracle en un servidor y enchufar directamente las aplicaciones clientes. Un Oracle mal configurado puede tener un desempeño muy lento.

2.9.3 PostgreSQL

Ventajas:

- Instalación ilimitada, es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Con PostgreSQL, nadie puede demandarnos por violar acuerdos de licencia, puesto que no hay costo asociado a la licencia del software, esto tiene varias ventajas adicionales como la flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.
- Ahorros considerables en costos de operación, PostgreSQL ha sido diseñado y creado para tener un mantenimiento y ajuste mucho menor que los productos de los proveedores comerciales, conservando todas las características, como una muy buena estabilidad y un buen rendimiento en consultas con un amplio volumen de datos.
- Multiplataforma, PostgreSQL se ejecuta en casi cualquier elección de hardware y sistema operativo; está disponible en casi cualquier plataforma Unix y sistemas operativos basados en UNIX como FreeBSD, Linux y Mac OS X, está completamente integrado dentro de Solaris 10 e inclusive tiene una versión para Windows.
- En cuanto a las herramientas gráficas de diseño y administración de bases de datos, existen varias herramientas gráficas de alta calidad para administrar las bases de datos (pgAdmin , pgAccess).

Desventajas:

Técnicamente, las principales desventajas de usar PostgreSQL están en cuatro áreas.

- La capacidad que proporciona PostgreSQL con su lenguaje PL/pgSQL para escribir funciones y procedimientos almacenados es un poco más limitada que la que se conseguiría con el lenguaje PL/SQL de Oracle o el T-SQL de Sybase. A no ser que se esté desarrollando un trabajo sofisticado en procedimientos almacenados, ésta no es una seria limitación.
- Características para bases de datos muy grandes (millones de registros), como lo son los table spaces, tablas particionadas y una restricción de acceso muy robusta,

son aún más fuertes en los Sistemas Gestores de Base de Datos (SGBD) propietarios.

- Las herramientas de desarrollo propietarias son poderosas y robustas. Particularmente Microsoft tiene excelentes herramientas y no es una sorpresa que trabajen mejor con el conjunto de productos Microsoft. La ventaja de este conjunto de herramientas tiene un efecto en la elección de la Base de Datos.
- No cuenta con un gran soporte técnico, como ocurre con algunos SGBDs, por lo tanto no se tiene la disponibilidad de un técnico para que dé solución a un determinado problema con nuestra base de datos; por el contrario, sí se tiene una enorme comunidad que siempre está dispuesta a ayudar en los foros y algunas empresas están comenzando a dar soporte para PostgreSQL como lo es SUN MycroSystems que actualmente brinda un soporte integral para la versión nativa de PostgreSQL en su Sistema Operativo Solaris 10 y su versión libre OpenSolaris.

Sistema Gestor de Base de Datos	Ventajas	Desventajas
SQL Server	• Buenas herramientas para la administración de la BD, como el DMF. • Se tiene un gran soporte disponible • Es muy estable con datos de medio nivel.	• Sólo corre sobre plataformas Microsoft • No es tan robusto como DB2 o Oracle • La licencia tiene un costo
Oracle	• Es multiplataforma • Fácilmente escalable • Es más rápido en consultas que involucran conjuntos de datos muy grandes.	• Tiene un costo elevado • Requiere cierto entrenamiento por parte del administrador.
PostgreSQL	• Es software libre y de código abierto. • Gran Estabilidad • Multiplataforma • Tiene una enorme comunidad de desarrollo	• Pequeña diferencia de rendimiento en cuanto a volúmenes de consulta muy grandes. • La capacidad de PL/pgSQL es un poco limitada. • No cuenta con un gran soporte técnico por parte de una empresa

Tabla 2.2 *Comparativa de ventajas y desventajas entre los SGBD.*

2.10 Selección del software y hardware

2.10.1 Selección del software

Una vez analizadas las ventajas y desventajas, se decidió utilizar **Java** como lenguaje de desarrollo para realizar las transacciones a la base de datos, ya que éste es independiente de la plataforma y también logra una independencia del sistema gestor de base de datos a través del JDBC, además Java nos hará más sencilla la programación para el manejo de la información, así como la reutilización de código a través de sus clases.

En lo referente al sistema gestor de base de datos, se eligió **PostgreSQL** debido a que permite implementar un buen diseño de la base de datos, es flexible, escalable y corre en casi cualquier tipo de computadora y sistema operativo, aunado a su eficaz conexión con Java logrando un gran rendimiento en cuanto al tiempo de respuesta a las consultas. Cabe mencionar que el hecho de ser software libre, nos permitió empezar a trabajar con él inmediatamente y así ganamos tiempo al implementar las primeras pruebas.

Finalmente, para la elección del sistema operativo del servidor, en el cual estará alojada la mayor parte de la base de datos, se eligió **Linux** en su distribución Ubuntu, en el cual se integra muy bien PostgreSQL, lográndose un mejor rendimiento y compatibilidad.

2.10.2 Selección del hardware

Debido a la importancia en el tiempo de respuesta de nuestro sistema a las múltiples consultas hechas desde el programa de nesting, se requiere tanto de una computadora capaz de proporcionar una gran rapidez de procesamiento para lograr un rendimiento óptimo, así como un servidor de base de datos que proporcione un buen rendimiento de respuesta a las consultas.

Características del Servidor:

A causa del gran almacenamiento que tendrá el sistema, ya que son muchos modelos los que maneja la empresa durante cada temporada, aunado a que cada modelo tiene un archivo asociado a las piezas derechas, un archivo asociado a las piezas izquierdas, así como 28 clusters en promedio (30 archivos a almacenar como mínimo), y los modelos de las temporadas pasadas prevalecerán en la base de datos; se requiere de un servidor con 2 discos duros de 250 GB de almacenamiento, bajo la configuración Raid1 para replicar la información.

- Procesador Intel Core2 Duo, 2.13 GHz o superior
- 2GB RAM o superior
- Tarjeta controladora de discos duros SATA-RAID 1
- 2 discos duros de 250GB o superior
- Tarjeta de red Ethernet 1Gb

Características de la estación de trabajo:

- Procesador QuadCore Xeon 2.26GHz o superior
- Sistema Operativo Windows
- Tarjeta de Video Nvidia 256MB o superior
- 4 GB RAM o superior
- Disco Duro 250GB o superior
- Tarjeta de Red Ethernet 1Gb

Capítulo 3 Diseño de la solución

3.1 Elementos del sistema

El Sistema de Información interactúa con los diferentes módulos del Sistema de Nesting para seleccionar, obtener y cargar la información en las estructuras de datos requeridas por el proceso de nesting, así a partir de la orden de trabajo se obtiene la información del modelo y talla para determinar cuanta piel y de qué tipo se requiere, el Sistema selecciona y lee los archivos de digitalización de la piel (DIPIs) proporcionando el contorno de la piel, las zonas de calidad, zonas de estiramiento y defectos, luego se ejecuta otro programa el cual primeramente obtiene las piezas y clusters correspondientes al modelo especificado en la orden de trabajo para después calcular dinámicamente la prioridad tanto de las piezas como de los clusters para indicarle al programa de nesting cuál es el siguiente elemento a acomodar; si el siguiente elemento a acomodar es un cluster un programa distinto proporciona el contorno, pieza base y pieza secundaria, por el contrario, si el siguiente elemento a acomodar es una pieza, existe otro programa que proporciona el polígono de la geometría de la pieza. Al finalizar el proceso de Nesting se borran los registros de la tabla prioridad ya que esta es una tabla temporal.

El esquema del Sistema de Información es el siguiente:

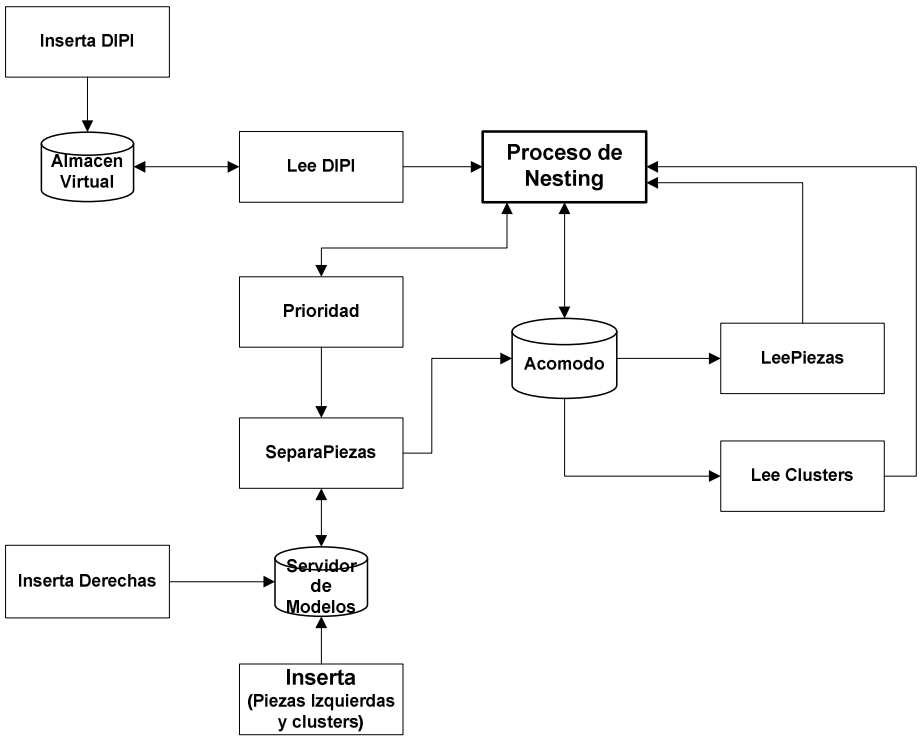


Fig. 3.1 *Esquema del Sistema de Información*

Dado el esquema del Sistema de Nesting y los requisitos de Información, podemos determinar que el Sistema de Información tendrá los siguientes elementos:

- Una Base de Datos Distribuida (BDD) donde se almacenan las pieles virtuales, las características y geometría de cada una de las piezas correspondientes a cada modelo de zapato, además de la información de la orden de trabajo. Ésta información es canalizada por el sistema hacia el programa de Nesting que se encargará de llevar a cabo el proceso de Acomodo.

- Programas que efectuarán transacciones hacia la base de datos y procesarán la información obtenida, alojados en diferentes equipos del Sistema de Nesting, estos programas son:

  - Programas que insertan y actualizan la información en la Base de Datos.
  - Programas que obtienen información de la Base de Datos.

  Con base en los programas anteriores se generan los

  - Programas que seleccionan, obtienen y cargan información en las estructuras de datos requeridas para el proceso de nesting: Leer DIPI, Leer piezas de un determinado modelo, cálculo de prioridad.

  Todos estos programas estarán coordinados con el diseño y la operación de la BDD para mantener la integridad de la información.

3.2 Distribución del sistema de información

El Sistema de Información está distribuido en cuatro equipos de cómputo para llevar a cabo sus funciones:

Equipo Visión.- Es la estación de trabajo en la cual se obtiene la representación digital de la piel de la vaca, dicha representación se guarda junto con sus características (archivo DIPI); dentro de la base de datos.

Estación de trabajo Nesting.- Una estación de trabajo donde se llevará a cabo el proceso de Nesting, se almacenarán inicialmente los resultados del mismo y de donde se podrán obtener los reportes necesarios del proceso.

PC Diseño.- En esta PC se genera la discretización de las piezas en el formato requerido para el proceso de Nesting, con el fin de ser almacenadas en el Servidor de Modelos.

Servidor de Modelos.- En este servidor se aloja la mayor parte de la base de datos **nesting,** en la cual el sistema de información conjunta los datos que definen la geometría de las piezas incluyendo sus respectivos izquierdos y clusters; por cada talla de cada modelo.

El diagrama de distribución de equipos del sistema de nesting es el siguiente:

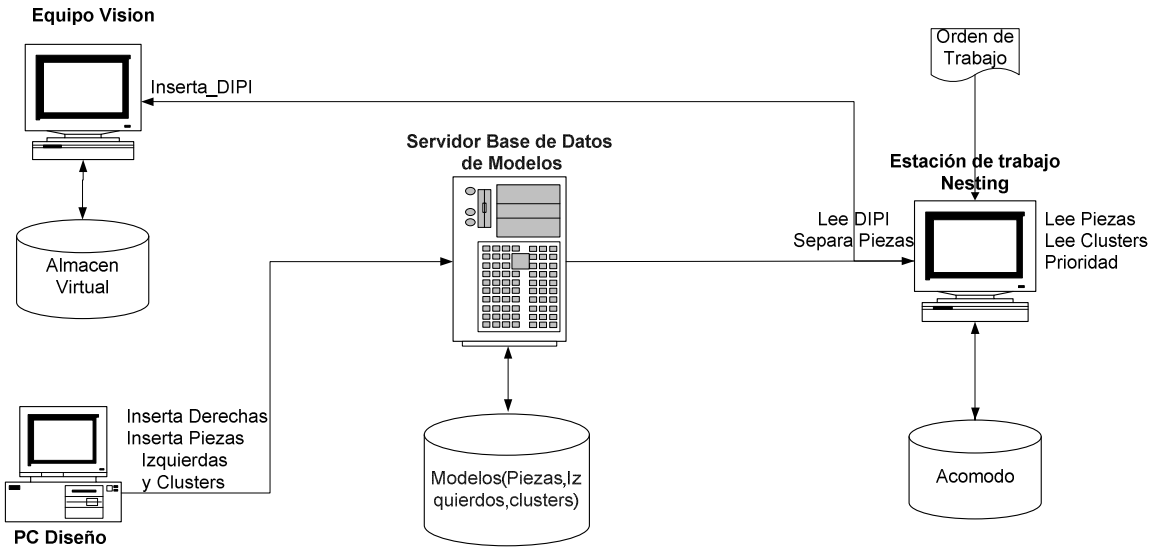


Fig. 3.2 *Diagrama de equipos del Sistema de Nesting*

3.3 Diseño de la base de datos

El objetivo de la base de datos **nesting** es manejar la información de tres sitios diferentes:

- Equipo Vision
- PC Diseño
- La orden de trabajo capturada por la estación de trabajo nesting

Para crear dicha base de datos debemos pasar por tres fases:

- Modelado conceptual
- Diseño lógico
- Diseño físico

3.3.1 Modelado conceptual

La información que requiere el proceso de nesting es:

- El modelo y número de pares que deben acomodarse
- Las características del modelo
- Las hojas de piel, las piezas y los clusters
- Mantener los resultados del proceso de Nesting en disco duro.

En base a lo anterior se realizó el esquema conceptual identificando las entidades, interrelaciones, atributos y restricciones semánticas de la información que estará contenida en nuestra base de datos y lo cual nos permite pasar a la segunda fase que es el diseño lógico.

3.3.2 Diseño lógico

En el diseño lógico se deben coordinar exigencias, como son eliminar redundancias, conseguir la máxima simplicidad y evitar cargas suplementarias de programación, obteniendo una estructura lógica adecuada que venga a establecer el debido equilibrio entre las exigencias de los usuarios y la eficiencia.

Esta etapa converge en el esquema lógico de la base de datos **nesting**, que contendrá la información necesaria para el proceso de nesting.

La base de datos debe ser capaz de almacenar la orden de trabajo con la descripción del producto, la talla y la cantidad de pares solicitados, ya que a partir de dicha orden de trabajo es que comienza el proceso de nesting.

Para que la base de datos pueda mantener la integridad de la información que almacenaremos en ella, es necesario que a cada producto de calzado se le asigne una descripción de producto y una talla como llave primaria, el área de piel por zona de calidad requerida para un par de zapatos, la ruta del archivo de piezas derechas, la ruta del archivo de piezas izquierdas y el material correspondiente al modelo.

La tabla material sirve de catálogo a cada piel virtual almacenada en la base de datos, ya que cada piel virtual deberá corresponderse con un campo descripcion_material (previamente cargado dentro de la tabla material), identificando el tipo de piel de la cual debe cortarse el producto, ya que si el color del calzado cambia, también cambia el tipo de piel de la cual debe cortarse.

La tabla prioridad será temporal para no incrementar demasiado el volumen de la base de datos, debido a que cada modelo tiene de 5 a 7 piezas derechas, 5 a 7 piezas izquierdas, así como un mínimo de 28 clusters (38 archivos) que se tendrían que guardar por cada modelo y talla.

En lo referente a los clusters, cada archivo correspondiente a un cluster debe poder identificarse de forma única con su id y su talla y relacionarse con un único modelo a través de la descripcion_producto.

El diagrama lógico de la base de datos es el siguiente:

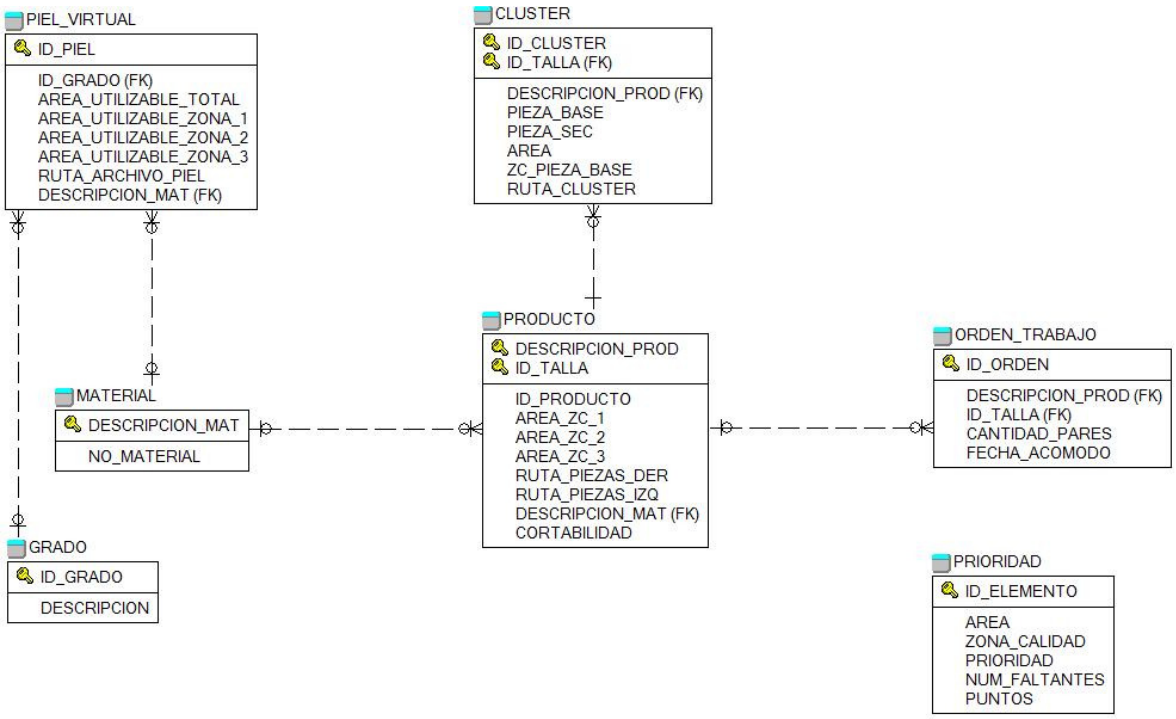


Fig. 3.3 *Diseño lógico de la base de datos nesting*

Del diagrama anterior podemos observar que la base de datos **nesting** cuenta con 7 tablas:

- orden_trabajo
- producto
- cluster
- piel_virtual
- material
- grado
- prioridad

3.3.3 Diseño físico

El diseño físico de la base de datos, con los correspondientes tipos de datos de los atributos, es el siguiente:

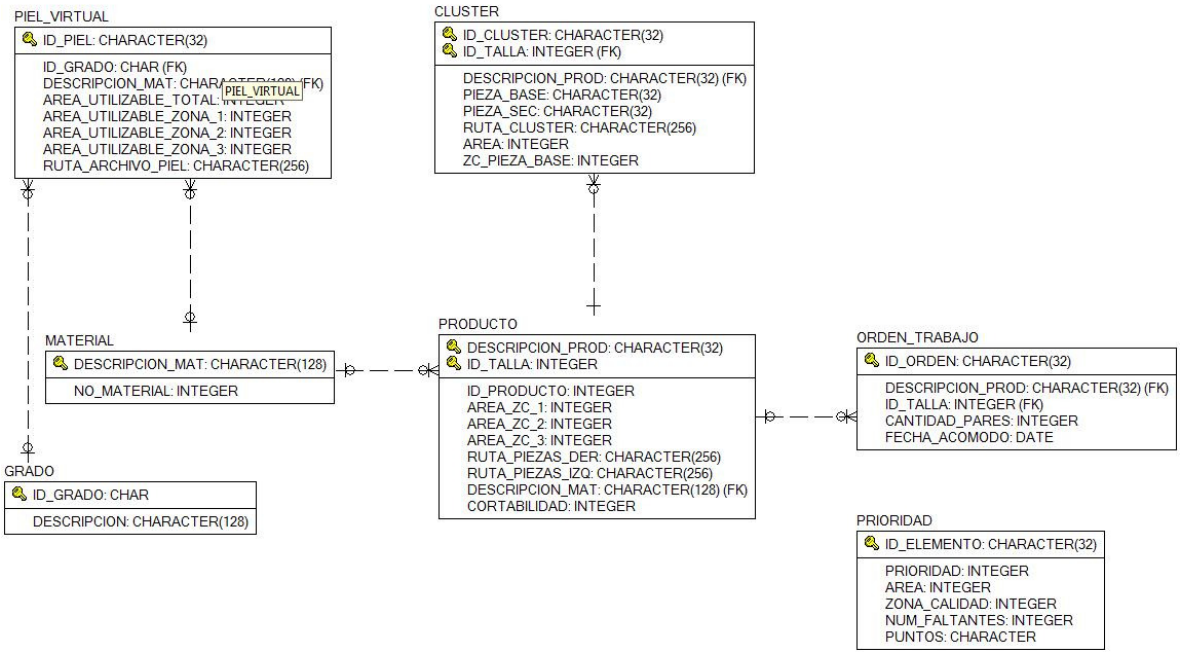


Fig. 3.4 *Diseño físico de la base de datos*

3.4 Flujo de información del sistema

De manera general, el flujo de la información del sistema es:

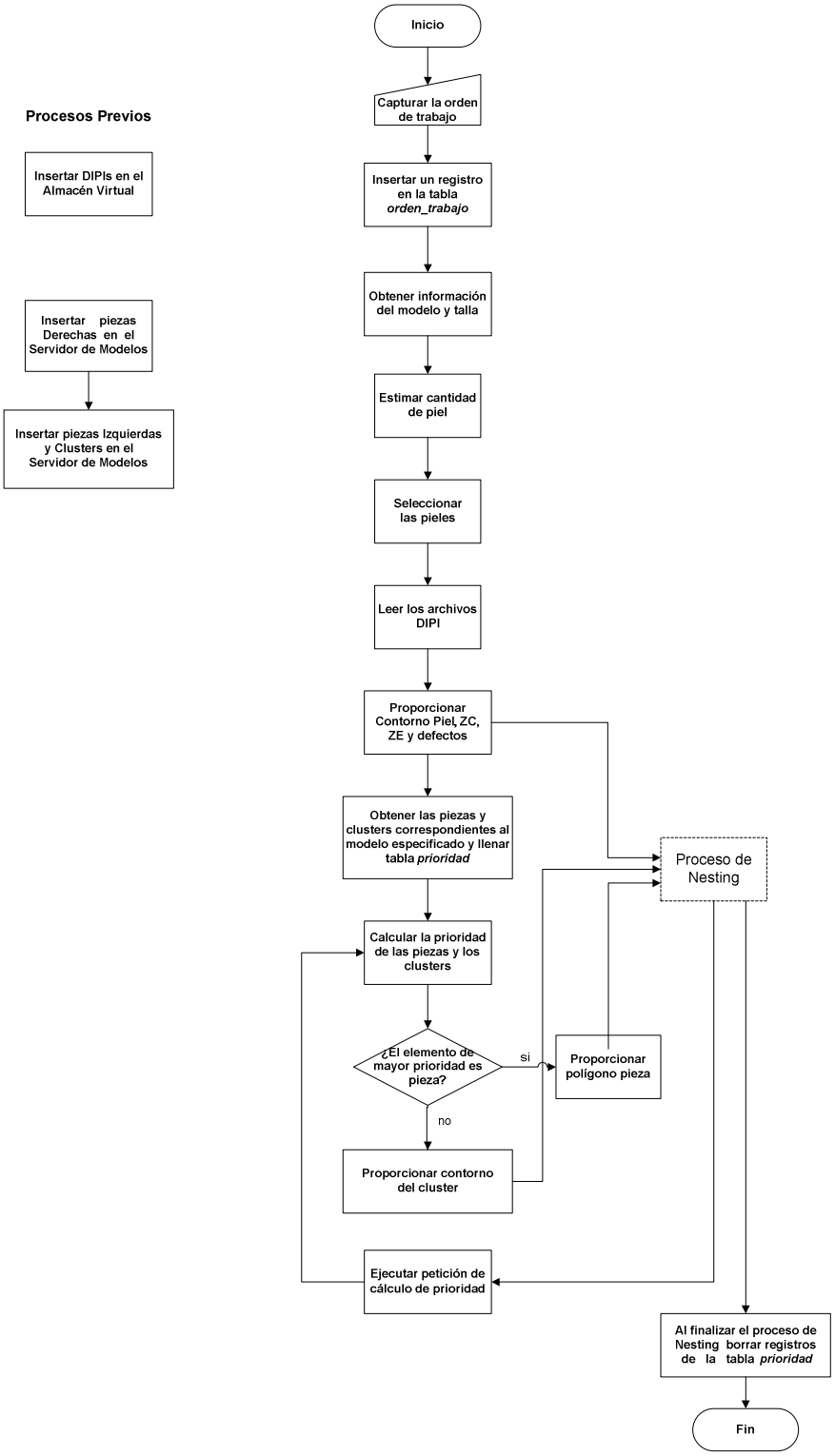


Fig. 3.5 *Diagrama de flujo de información en el sistema*

Capítulo 4 Desarrollo de la solución

4.1 Programas que insertan información en la base de datos

4.1.1 Inserta_DIPI

El programa **Inserta_DIPI** está contenido en el Sistema de Digitalización de la Piel y básicamente lo que hace es establecer la conexión con el Almacén Virtual desde código Visual C++, para insertar en la base de datos, las características esenciales del archivo DIPI como son: material, grado, área por zonas de calidad, ruta del archivo de Digitalización de la Piel (DIPI) y que serán de gran importancia para efectuar la selección de los DIPIs con los cuales efectuar un proceso de acomodo y que nos permitirán lograr un menor desperdicio de piel.

Para insertar las características esenciales, para la selección, de los archivos DIPIs en la base de datos **nesting,** la información debe fluir de acuerdo al siguiente diagrama:

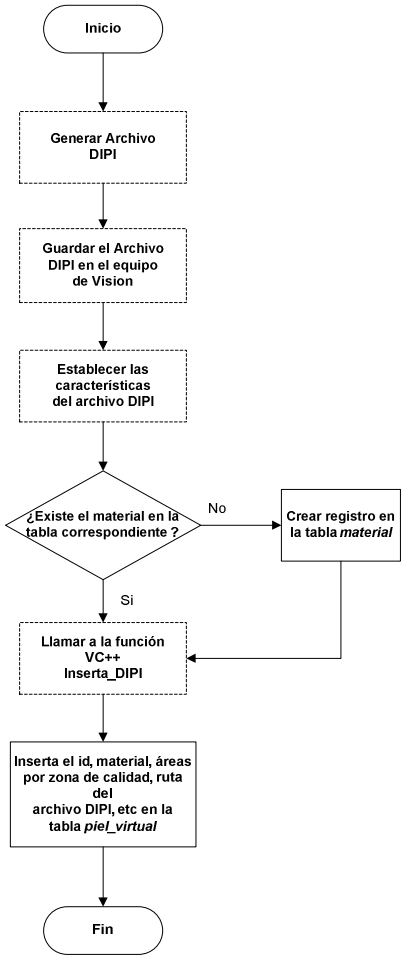


Fig. 4.1 *Diagrama de flujo para insertar las características de un archivo DIPI*

A continuación se muestra un ejemplo de la tabla *piel_virtual.*

id_piel [PK] characte	id_grado character(1)	area real	area_utilizable_total real	area_utilizable_zona_1 real	area_utilizab real	area_u real	ruta_archivo_piel character varying(64)
0000000007	A	1141	1118	600	400	118	C:\Almacen Virtual\0000000007

4.1.2 Inserta_Derechas

Inserta_Derechas es un programa que reside en la PC Diseño donde se efectúa la discretización de las piezas en el formato requerido para el proceso de Nesting, así los procesos previos a la ejecución del programa son: con base en un archivo dxf generar el archivo de piezas derechas (APD) correspondiente, el operador captura la talla, el modelo e indica si se trata de un modelo y talla nuevos o si se trata de una actualización, luego se guarda el APD en el Servidor de Modelos, para finalmente ejecutar el programa Inserta_Derechas a través del cual se inserta en la tabla *producto* la descripción del producto, id, talla, así como la ruta del APD.

Este programa también establece la conexión a la base de datos desde código Visual C++.

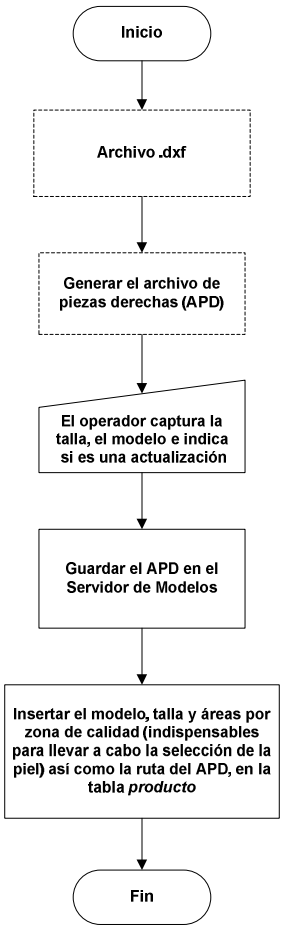


Fig. 4.2 *Diagrama de flujo para insertar el archivo de piezas derechas en la bd.*

4.1.3 Inserta

El módulo **Inserta** opera dentro del módulo Generador de Clusters y funciona tanto para clusters como para las piezas izquierdas, estableciendo una comunicación desde la PC Diseño hacia la base de datos **nesting.** Se tiene un método llamado InsertaCluster el cual inserta dentro de la tabla *cluster* las características de cada uno de los clusters, tanto derechos como izquierdos, que fueron generados; reflejando un registro como el siguiente:

	id_cluster [PK] characte	id_modelo character var	pieza1 character var	pieza2 character var	id_talla [PK] num	ruta_cluster character varying(256)
1	16003_CD0	16003	16003_D1	16003_D2	7.0	C:\Users\Etosh\Desktop\Tesi

El programa también tiene un método llamado insertaPiezas, el cual inserta en la tabla *producto* el archivo de las piezas izquierdas, que son generadas a partir de la geometría de las piezas derechas; para realizar esta acción se verifica si el campo ya existe dentro de la tabla (debido a ejecuciones pasadas del Generador de Clusters) para el modelo y talla en cuestion, si es así, se le preguntará al usuario si quiere actualizar dicha información, si el campo no existe se inserta en automático las características de interés de las piezas izquierdas.

Edit Data - PostgreSQL 8.3 (localhost:5432) - nesting - producto

File Edit View Help

No limit

	id_producto integer	des_producto [PK] character	id_talla [PK] int	area_zc_1 integer	area_zc_2 integer	area_zc_3 integer	ruta_p_derechas character varying	ruta_p_izquierdas character varying(256)
1	1000000016	Ofelia_negro	270	10	20	35	C:\Nesting\Modelos\(	C:\Nesting\Modelos\Ofelia_n
2	1000000017	Italia_negro	270	12	20	27	C:\Nesting\Modelos\(	C:\Nesting\Modelos\Italia_ne

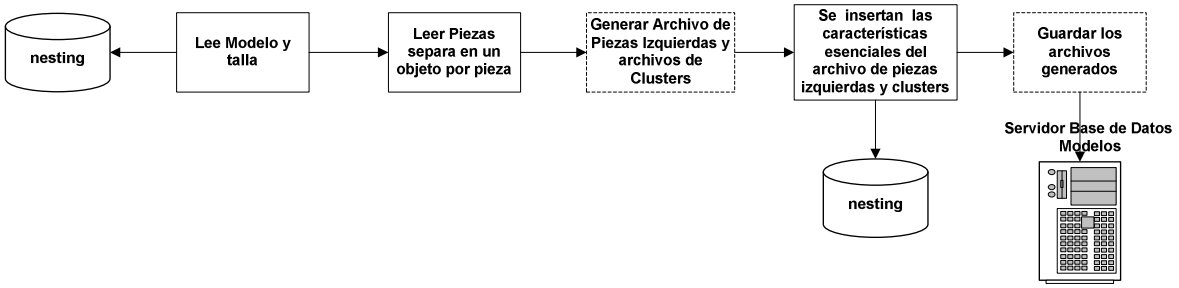


Fig. 4.3 *Diagrama del módulo Inserta*

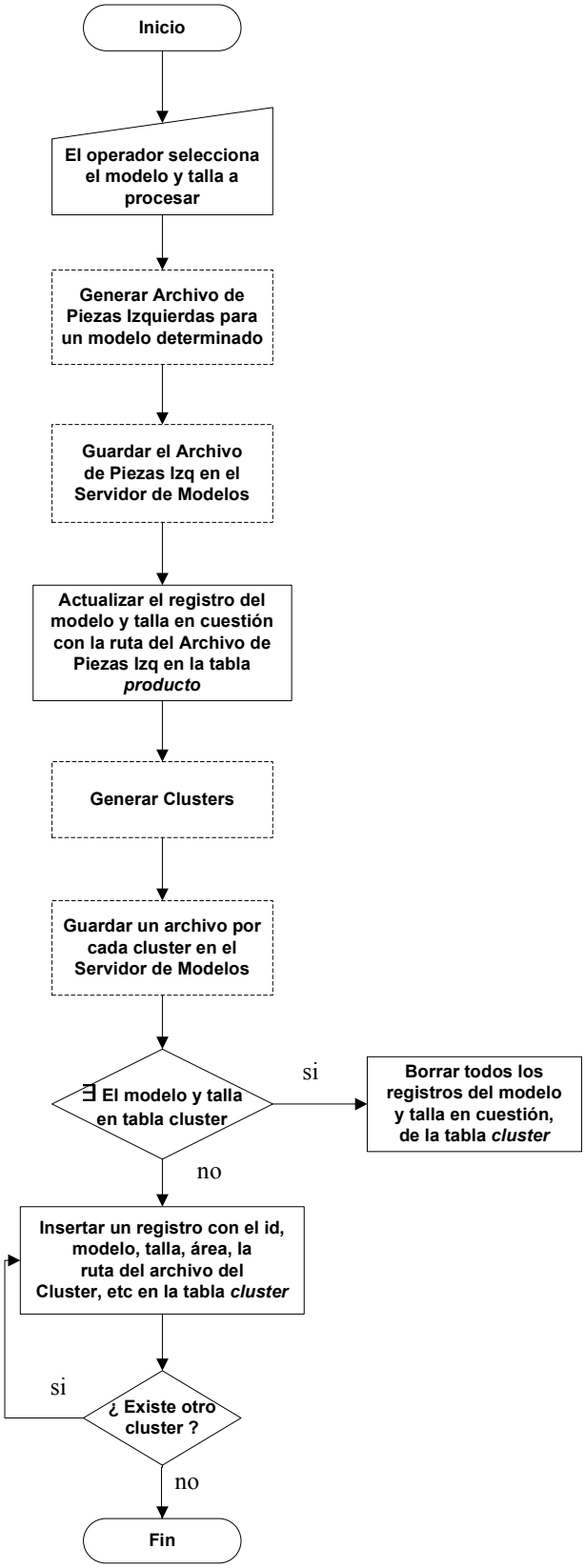


Fig. 4.4 *Diagrama de flujo para insertar el archivo de piezas izquierdas y los clusters*

4.2 Programas que seleccionan, obtienen y cargan información

4.2.1 Lee_DIPI

El programa **Lee_DIPI** tiene como objetivo seleccionar y leer los archivos DIPI, para obtener de ellos las estructuras de datos correspondientes a las pieles virtuales sobre las que se llevará a cabo un proceso de nesting.

Para realizar lo anterior, el programa consulta a la base de datos **nesting** para obtener el área por zona de calidad de cada una de las piezas derechas e izquierdas, correspondientes al modelo de calzado indicado en la orden de trabajo.

Una vez que se obtiene el valor del área por zona de calidad requerida para un par, multiplicamos por la cantidad de pares, indicada en la orden de trabajo, para obtener el **Á**rea **T**otal **R**equerida por **Z**ona de **C**alidad (ATRZC) necesaria para cumplir con un pedido. En base a estos tres valores se realizan consultas al almacén virtual, para efectuar la selección de los archivos de pieles (DIPIs) que al ser procesadas nos darán el menor desperdicio, después se leen los archivos seleccionados, extrayéndose la geometría del contorno de la piel, las zonas de calidad, las zonas de estiramiento y los defectos; para finalmente cargarlos como objetos del tipo polígono, dentro del programa de nesting.

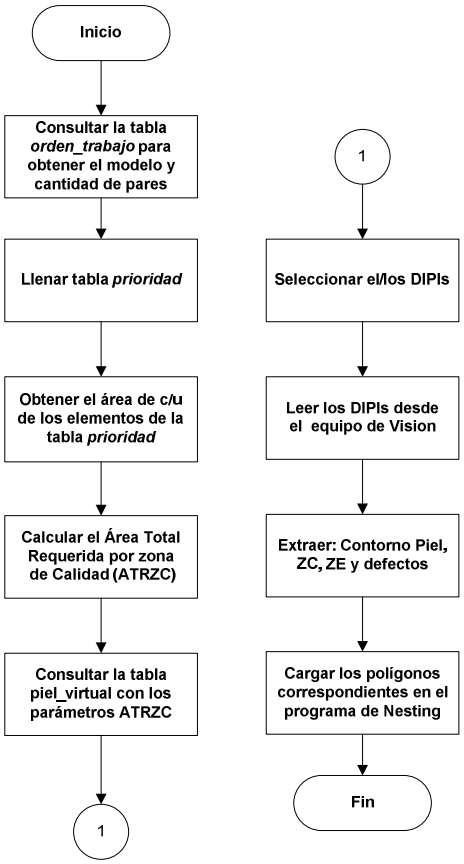


Fig. 4.5 *Diagrama de flujo para leer un archivo DIPI*

4.2.2 Lee_Piezas

El programa **Lee_Piezas** se encarga de crear el objeto correspondiente a una pieza, en la estación de trabajo Nesting. Para ello se selecciona de la tabla *prioridad* la pieza a procesar y se toman los puntos de dicho elemento del campo correspondiente, para crear el objeto pieza que será utilizado por el programa de Nesting para acomodarlo.

Fig. 4.6 *Diagrama de flujo para cargar piezas en el programa de Nesting*

4.2.3 Lee_Clusters

El programa **Lee_Clusters** obtiene los tres polígonos que conforman a un cluster: el Contorno, la pieza base y la pieza secundaria.

El programa se conecta con la base de datos **nesting** para obtener la ruta del archivo que define a cada cluster de un determinado modelo y talla de calzado. Con la ruta obtenida, el programa lee desde el Servidor de Modelos un archivo por cada cluster y de éste último se extrae la información cargando tres polígonos dentro del programa de nesting:

- Contorno
- Pieza_base
- Pieza_secundaria

Cabe señalar que aunque el Contorno define a la pieza_base y la pieza_secundaria en conjunto, es importante poder representar a las piezas de forma separada para el proceso de corte de la piel, posterior al nesting.

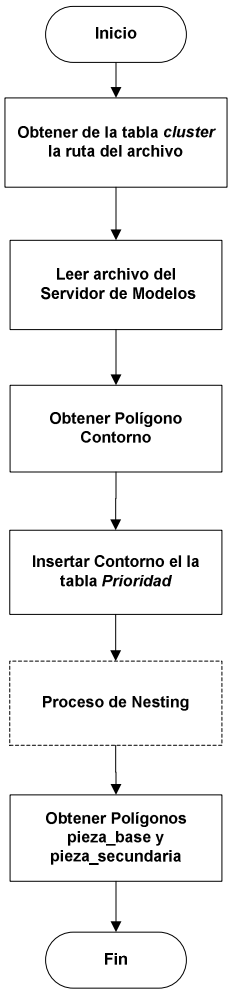


Fig. 4.7 *Diagrama de flujo para leer un cluster*

4.3 Programas que procesan información

4.3.1 Separa_Piezas

El programa **Separa_Piezas** se encarga de leer el archivo de piezas derechas y el archivo de piezas izquierdas y crear un registro en la tabla prioridad por cada una de las piezas descritas en dichos archivos.

El programa se ejecuta a petición de otro programa llamado Prioridad, el cual le pasa como parámetros el modelo y talla específicos de un par de calzado, con estos parámetros el programa Separa_Piezas selecciona de la tabla *producto* los campos: ruta_piezas_der y ruta_piezas_izq, luego con estas rutas se leen desde el servidor de modelos los archivos correspondientes a las piezas derechas e izquierdas, cada archivo tiene un identificador de comienzo por cada pieza que contiene, el cual nos permite extraer las características de las piezas (que serán esenciales para llevar a cabo el cálculo de la prioridad) y separar en una cadena de texto las coordenadas que describen a una pieza, lo cual nos permite calcular el área de la pieza, para finalmente establecer una comunicación con la base de datos y almacenar en la tabla prioridad: id_pieza, área, zona de calidad a la que pertenece y la cadena de texto que describe a dicha pieza.

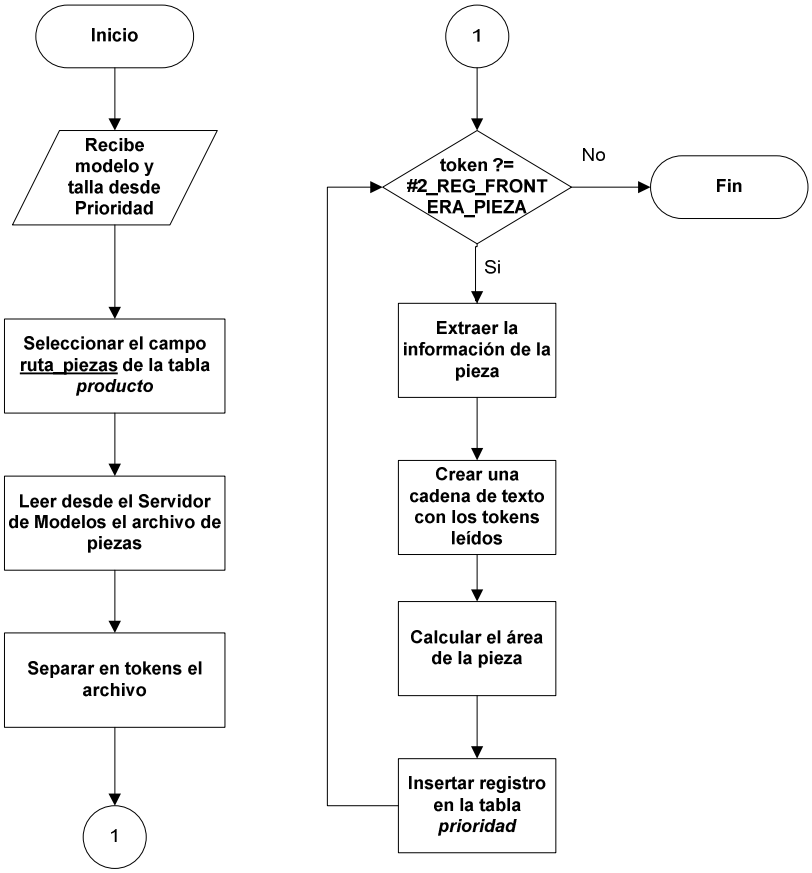


Fig. 4.8 *Diagrama de flujo para separar las piezas y almacenarlas en la base de datos*

4.3.2 Prioridad

El programa **Prioridad** calcula y actualiza la prioridad de cada una de las piezas y de los clusters de manera dinámica, lo que permitirá elegir cuál es el siguiente elemento a acomodar durante el proceso de Nesting.

Este programa interactúa con el programa Separa_Piezas con el fin de llenar la tabla *prioridad* con los registros correspondientes a las piezas y clusters del modelo y talla que se esté procesando en ese momento. El programa prioridad calcula y actualiza la prioridad de las piezas y los clusters cada vez que el Sistema de Nesting acomoda una pieza; de acuerdo a tres criterios:

- Peso de Calidad (Wc): si la zona de calidad actual coincide con la zona de calidad del elemento, se le asigna a este peso el valor del área mayor de los elementos previamente almacenados en la tabla *prioridad*, si no es así se asigna el valor de la zona de calidad correspondiente a la pieza o cluster ya sea 1, 2 o 3.
- El área de la pieza o cluster
- El número de unidades faltantes, para este criterio se definió el factor $\frac{1}{num\ faltantes}$ para que a medida que el número de piezas que se tienen que acomodar disminuya, éste factor aumente y contribuya a obtener una mayor prioridad; logrando que las piezas de las cuales sólo faltan un par de unidades por acomodar, tengan una mayor prioridad.

Los tres criterios anteriores se conjuntan en la siguiente expresión:

$$Prioridad = Wc + area_{\substack{pieza\ o \\ cluster}} + \frac{1}{num\ faltantes}$$

Después de que la prioridad es calculada para cada una de las piezas y clusters de la tabla *prioridad*, se actualiza el campo prioridad en todos los registros de dicha tabla.

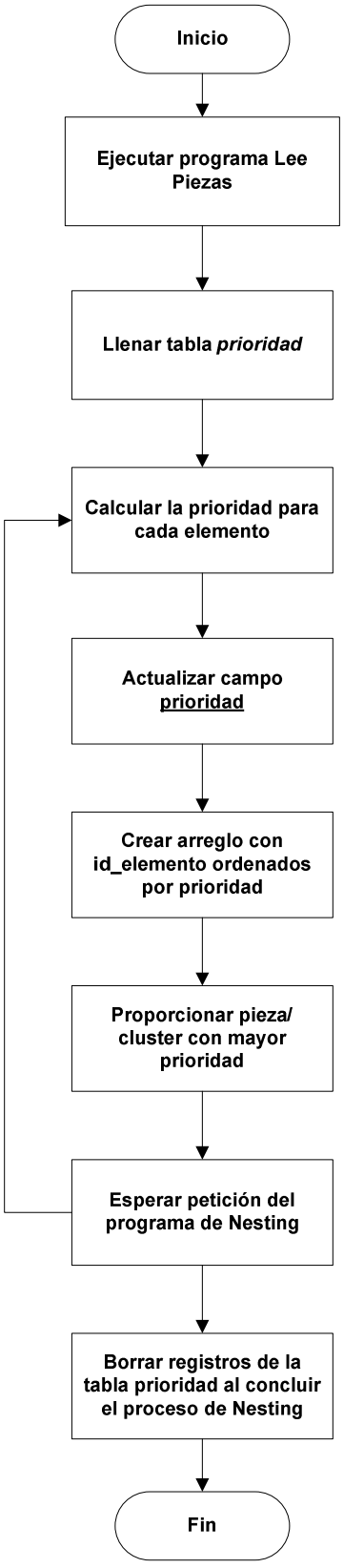
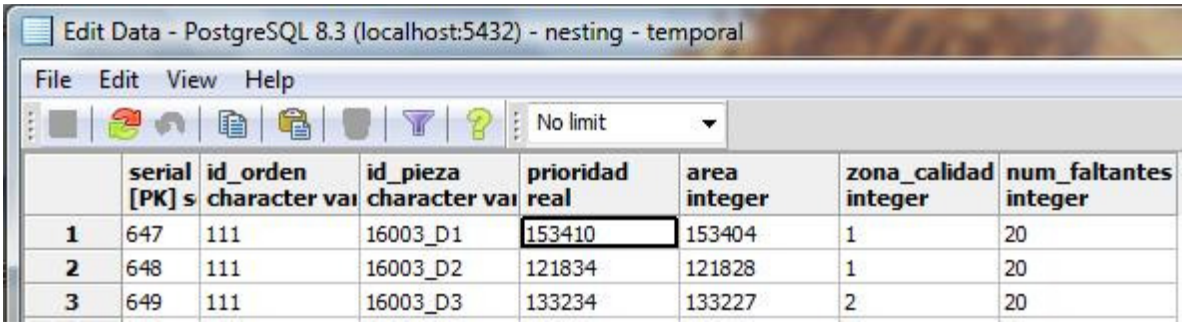


Fig. 4.9 *Diagrama de flujo para calcular dinámicamente la prioridad de los elementos*

A continuación se muestra un ejemplo de la tabla *prioridad*, luego de llevar a cabo las acciones anteriores:


Edit Data - PostgreSQL 8.3 (localhost:5432) - nesting - temporal

File Edit View Help

No limit

	serial [PK] s	id_orden character var	id_pieza character var	prioridad real	area integer	zona_calidad integer	num_faltantes integer
1	647	111	16003_D1	153410	153404	1	20
2	648	111	16003_D2	121834	121828	1	20
3	649	111	16003_D3	133234	133227	2	20

4.4 Problemas encontrados al realizar los programas

Uno de los inconvenientes encontrados durante la realización del sistema fue que se tenía que insertar la información esencial tanto del archivo DIPI como del archivo de piezas derechas en la base de datos, pero sin embargo, los programas que generan éstos archivos están desarrollados en Visual C++ 2005, así es que se tuvo que adoptar éste lenguaje de programación para realizar los programas Inserta_DIPI e Inserta_Derechas.

Otro de los problemas con los que nos enfrentamos dentro del desarrollo del sistema es que el módulo Generador de Clusters no calculaba el área de las piezas izquierdas, por lo cual no había valor para este campo, pero en la tabla *prioridad* el valor del área está definido como not null ya que el programa de Prioridad utiliza este valor para calcular dinámicamente la prioridad de las piezas, por lo cual tuvimos que reutilizar la clase CalculaAreas, perteneciente al programa LeeDIPI, insertándola dentro del Generador de Clusters para obtener el área de las piezas izquierdas.

El programa Lee_DIPI se tardaba mucho en cargar las estructuras de datos correspondientes dentro del programa de nesting, ya que se leía como una única cadena todo el archivo DIPI y luego se separaba en tokens, por lo cual lo modificamos para que leyerá el archivo DIPI línea por línea y a la vez fuera extrayendo las características y las estructuras de datos correspondientes; logrando reducir el tiempo de ejecución para leer el archivo DIPI de 12 seg a solo 2 seg.

Se tuvieron que hacer algunas modificaciones al programa Separa_Piezas para ser reutilizado dentro del generador de clusters, ya que el generador de clusters realiza su procesamiento a partir de las piezas derechas consideradas por separado, así es que el programa fue modificado para proporcionar un objeto por cada pieza derecha y se omitió almacenar los datos de las piezas derechas en la base de datos, como normalmente lo hace el programa Separa_Piezas original.

En cuanto al cálculo de la prioridad, los pesos de la zona de calidad tuvieron que definirse de manera heurística, para obtener buenos resultados.

Capítulo 5 Pruebas y resultados

Cuando los diferentes componentes del sistema ya se habían comprobado individualmente, se conjuntaron los equipos de desarrollo y se realizó la etapa de pruebas para mostrar que el Sistema de Información es correcto. Gracias al enfoque de desarrollo de software incremental, bajo el cual fue desarrollado el Sistema, los distintos programas que lo componen se fueron probando dentro del Sistema de Nesting desde que estaban parcialmente concluidos.

No es conveniente que la etapa de pruebas se considere como el proceso final en la creación de un Sistema de software, puesto que a medida que el sistema vaya creciendo se volverá más complejo detectar una falla en su estructura y más aún encontrar la línea de código exacta donde se está generando el problema. Por ello las pruebas deben ser realizadas durante el desarrollo y al finalizar el proyecto con el fin de asegurar que cada parte del sistema esté funcionando de forma correcta.

5.1 Pruebas durante el desarrollo

Las pruebas incluyeron las transacciones hacia la base de datos, la selección de la información y la generación de las estructuras de datos correspondientes al proceso de nesting, primeramente se probó cada uno de los programas que conforman el sistema de información, para su posterior integración con los diferentes módulos del sistema de nesting, utilizando los equipos de desarrollo del mismo.

Dado que la base de datos de nuestro sistema es una base de datos distribuida, se realizaron transacciones hacia los distintos equipos donde se almacena la información: vision, PC Diseño, Servidor de Modelos y estación de trabajo Nesting.

5.2 Programas que insertan información en la base de datos

El programa **Inserta_DIPI** se probó insertando varios registros correspondientes a las características de los archivos DIPI de ejemplo (proporcionados por el equipo de vision) en la tabla *piel_virtual* y dado que el programa recibe como parámetros los datos que debe insertar, el mismo programa hace el cast de tipos requeridos, para posteriormente realizar la inserción en la base de datos.

La problemática que se observó fue que se necesitaba la opción para dar de alta un nuevo material de la piel cuando éste no existe en la tabla *material*, la cual es un catálogo de *piel_virtual;* para resolverlo se agregó un parámetro booleano para determinar si se trata de un nuevo material, éste parámetro permite ejecutar el código correspondiente a la inserción del nuevo material en el catálogo correspondiente, para finalmente insertar los datos de la piel.

El programa **Inserta_Derechas** es fundamental para el funcionamiento del sistema de información ya que con él se insertan las piezas derechas de los zapatos, a partir de las cuales posteriormente se obtienen las piezas izquierdas y los clusters, así es que se verificó que se insertara en la tabla *producto* un registro por cada archivo de piezas derechas generado por el equipo de vision, dicho registro contiene los campos correspondientes al modelo, talla, área por zonas de calidad (indispensables para llevar a cabo la selección de la piel) y la ruta del archivo de piezas derechas. La única modificación que se tuvo que hacer es permitir al programa hacer la actualización de un determinado campo cuando éste ya existe en la tabla *producto,* pero se quiere insertar nuevamente los datos ya sea porque hubo un error en los parámetros pasados al programa o bien la empresa realiza algún cambio en las piezas de un determinado modelo.

El programa **Inserta** se integró con el programa Generador de Clusters y uno de los primeros problemas encontrados fue que éste siempre intentaba insertar los registros correspondientes a las piezas y los clusters sin importar si ya habían sido insertados en una ejecución pasada del programa, por lo cual se modifico para que se verifique si el id_pieza o id_modelo ya existía en la tabla correspondiente y si es así, se pregunta al usuario si quiere *actualizar* la información.

Para probar la inserción o actualización a la base de datos, se definió una matriz de pruebas correspondiente a los campos y los tipos de datos requeridos para modificar la tabla cluster.

INSERCIÓN EN LA TABLA CLUSTER										
Campos	Id_cluster	id_talla	id_modelo	area	pieza_base	ieza_secundari	zona_calidad	ruta_cluster		Correcto
Tipo de Dato	varchar(32)	integer	varchar(32)	integer	varchar(32)	varchar(32)	integer	varchar(256)		
	Obligatorio	Obligatorio	Obligatorio	Obligatorio	Obligatorio	Obligatorio	Obligatorio	Obligatorio		
1	"16003_C1"	7	"16003"	4700	...16003_D2..txt	erechas/16003	1	6003/derechas/16003_C1..txt		✔
2	"16003_C3"	7	"16003"	3300	...16003_D1..txt	erechas/16003	2	6003/derechas/16003_C3..txt		✔
3	"16003_C6"	7	"16003"	2900	...16003_D4..txt	erechas/16003	3	6003/derechas/16003_C6..txt		✔
4	"16003_C1"	7	"16003"	4700	...16003_D3..txt	erechas/16003	1	6003/derechas/16003_C1..txt		✘
5		7	"16003"	4700	...16003_D1..txt	erechas/16003	1	6003/derechas/16003_C1..txt		✘
6	"16003_C2"		"16003"	5100	...16003_D5..txt	erechas/16003	1	6003/derechas/16003_C2..txt		✘
7	"16003_C4"	7	"16003"	3300	...16003_D4..txt	6003/derechas/	#	6003/derechas/16003_C4..txt		✘
8	"16003_C5"	7	"16003"	2900	...16003_D1..txt	123	3	123		✘

Fig. 5.1 *Matriz de Pruebas*

Finalmente el Generador de Clusters se probó con seis modelos completos que se tenían disponibles almacenando de manera satisfactoria en la base de datos la información relacionada a los clusters de cada modelo y talla, así como la información del archivo de piezas izquierdas.

5.3 Programas que seleccionan, obtienen y cargan información

En cuanto al programa **Lee_DIPI,** se probó que para un estilo y talla existentes en la tabla *producto* de la base de datos **nesting**, fuera capaz de:

- Obtener el área total requerida por ZC
- Leer los archivos DIPIs
- Obtener las estructuras de datos correspondientes

Realizando su función de manera correcta, pero identificando que la tabla *piel_virtual* debía tener un campo del tipo booleano para marcar la piel como ya procesada.

Al realizar la prueba del programa **Lee_Clusters** se observó que la geometría de las piezas base y secundaria era un reflejo horizontal con respecto a la orientación del contorno del cluster, por lo cual se le requirió al programador del Generador de Clusters hiciera los cambios pertinentes para que tanto el contorno, como las piezas base y secundaria coincidieran.

5.4 Programas que procesan información

El programa **Prioridad** se probó con la petición del cálculo dinámico de la prioridad de las piezas para una determinada zona de calidad y un número de faltantes por cada pieza, percatándonos de que los registros de la tabla *cluster* no tenían asignada zona de calidad por lo cual dicho campo era evaluado como cero y afectaba de manera incorrecta al cálculo de la prioridad, para lo cual se le pidió al programador del Generador de Clusters; que proporcionara este valor al insertar los registros en la base de datos **nesting**.

Además se corrigió el peso de la zona de calidad asignándole el valor del área mayor en el caso de que la zona de calidad actual y la zona de calidad de la pieza coincidan. Con ello la prioridad es calculada dinámicamente, indicándole al programa de nesting cuál es la siguiente pieza a acomodar.

A continuación se presenta una tabla resumiendo los resultados de las pruebas del sistema de información.

Programas	**Descripción**	**Funcionalidad**	**Cambios efectuados**
Inserta_DIPI	Inserta en la tabla *piel_virtual* un registro correspondiente a un archivo de Digitalización de Piel	Correcto	Dar de alta un nuevo material
Inserta_Derechas	Inserta los datos del archivo de piezas derechas de un determinado modelo y talla	Correcto	Permitir la actualización
Inserta	Inserta en la bd nesting un registro correspondiente a las piezas izquierdas y los registros correspondientes a los clusters	Correcto	Permitir la actualización
Lee_DIPI	Selecciona los DIPIs mas apropiados y proporciona las estructuras de datos de la piel	Correcto	Marcar registros de la tabla *piel_virtual* como ya procesados
Lee_Clusters	Obtiene el contorno, la pieza base y la pieza secundaria de un cluster	Correcto	
Prioridad	Calcula y actualiza dinámicamente la prioridad, tanto de las piezas como de los clusters	Correcto	Reasignar peso de la zona de calidad

Tabla 5.1 *Resumen de las pruebas realizadas*

Se llevaron a cabo diferentes tipos de pruebas en el sistema para corroborar que funciona de manera correcta, proporcionando toda la información que es requerida para llevar a cabo el proceso de nesting. Las pruebas incluyeron la petición de estructuras de datos, integración de procesos y las transacciones en la base de datos.

Conclusiones

Podemos concluir que el sistema de información obtiene, almacena, administra y proporciona efectivamente toda la información necesaria para llevar a cabo el proceso de nesting. El Sistema se integra de forma adecuada al proceso de producción de la empresa ya que sus distintos programas operan en distintos tiempos, logrando cumplir completamente con sus objetivos:

- Inserción automática de piezas en la base de datos
- Inserción automática de clusters en la base de datos
- Proporcionar la información necesaria, para llevar a cabo el proceso de nesting de un modelo de zapato, sobre las hojas de piel necesarias para cumplir con una orden de trabajo.
- Preservar la integridad de la información

En lo referente a las herramientas de desarrollo seleccionadas, Java como lenguaje principal de desarrollo y PostgreSQL como el sistema gestor de base de datos, se obtuvieron los resultados esperados logrando con Java una completa compatibilidad con el sistema de nesting (debido a que tienen el mismo lenguaje de desarrollo) y con Postgres se logra un buen rendimiento a las consultas, manteniendo siempre la integridad de la información.

La base de datos, al igual que los programas para el manejo de la misma, serán protegidos por derecho de autor como parte del sistema de nesting. Como parte final del proyecto, el sistema completo será instalado en la planta de calzado esperando que mejore la productividad de la manufactura de la empresa.

Podemos decir que este trabajo cumplió con los requerimientos y expectativas del patrocinador, esperando acordar con la empresa de calzado, una versión 2 del sistema de nesting.

En lo personal, el presente trabajo me ayudó a:

- Desarrollar la habilidad de resolver problemas específicos
- Reafirmar el aprendizaje del trabajo en equipo, ya que fue muy importante tener una fuerte comunicación con los demás miembros del equipo de nesting; así como interactuar de manera remota con el equipo de visión (ubicado en Saltillo, Coahuila) para conjuntar el software y cumplir con los requisitos del sistema.
- Aplicar y ampliar los conocimientos adquiridos sobre programación y base de datos, para contribuir al desarrollo de un proyecto multidisciplinario.
- Adquirir experiencia para desarrollarme en el aspecto laboral.

Finalmente, como trabajo posterior se prevé que para una siguiente versión del sistema de nesting, se extienda tanto el número de modelos admitidos en cada ejecución del proceso,

como el número de tallas, para lo cual se tendrá que adaptar el sistema de información de manera tal que en una ejecución del programa de nesting se suministre la geometría de las piezas y clusters referentes a los distintos modelos y tallas indicados.

Actualmente el sistema de información atiende a una sola estación de trabajo, pero si se desea ampliar la producción, se pueden colocar varias estaciones de trabajo para realizar el proceso de nesting con el único requisito de replicar los programas del sistema en cada uno de los clientes.

La base de datos se podría implementar bajo el modelo objeto-relacional, el cual nos permitiría una mayor flexibilidad en el diseño, por ejemplo podríamos almacenar en la base de datos un tipo de dato polígono, el cual es muy frecuentemente usado por el proceso de nesting, y asociarle directamente algunos métodos, como en el caso de comprobar si dos polígonos se intersectan; lo cual nos llevaría a detectar las zonas de calidad y estiramiento de la piel.

Como parte del trabajo posterior también se realizará un módulo de reportes y administración del sistema.

Glosario

Cluster.- Es un conjunto de dos piezas tangentes entre si, cuya relación de la suma de las áreas de ambas entre el área de su envolvente convexa es mínima.

Defectos.- Son las distintas imperfecciones de la piel, tales como agujeros y rasgaduras.

Generador de Clusters.- Programa que a partir de las piezas derechas genera las piezas izquierdas y los clusters para in determinado modelo de calzado.

Modelo.- Se refiere al nombre con que se identifica a cada estilo y color de calzado, ejemplo Ofelia_negro, Ofelia_cafe.

Nesting.- Proceso automático de acomodo de piezas de calzado sobre una piel virtual, efectuado por un equipo de cómputo y que tiene como fin minimizar el desperdicio de la piel.

Piel virtual.- Es un archivo de **Di**gitalización de una hoja de **Pi**el (**DIPI**), en el cual se incluyen sus características, como lo son, zonas de calidad, dirección de estiramiento y defectos.

Pieza.- Es la digitalización de cada una de las piezas que conforman el zapato.

Prioridad.- Valor calculado dinámicamente que índica cual es la siguiente pieza a acomodar, considerando la zona de calidad actual y el área que ocupará.

Transacción.- Consulta, inserción o actualización en una tabla de la base de datos.

Zonas de Calidad (ZC).- Se refiere a las regiones de la piel, en las cuales el grosor varía, existen los tipos: 1, 2 y 3 siendo la zona 1 la de mayor calidad.

Zonas de Estiramiento.- Son regiones de la piel, en las cuales ésta estira en una dirección específica.

Bibliografía

[1] Secretaría de Economía. Programa para la Competitividad de la Industria del Cuero y del Calzado. México: 2003. Disponible en: http://www.economia.gob.mx/pics/p/p1325/Texto.pdf

[2] Álvarez M, Universidad Iberoamericana Campus Santa Fe, La esencia de los sistemas Just-in-time y su relación con Balanced Scorecard, [fecha de consulta 25 de abril 2009] Disponible en: http://www.monografias.com/trabajos17/relacion-jit-balanced/relacion-jit-balanced.shtml.

[3] http://demo.paloalto.pt/ziporen/imdata/p4_003.pdf

[4] http://www.taglio.it/cutvision/materiale/depli15-30_ENG.pdf

[5] Joyanes-Aguilar L. Programación *en C, Metodología, algoritmos y estructura de datos*. España. McGraw-Hill.2001. pag. 39.

[6] Deitel H., Deitel P. *Cómo programar en Java*, 5ª edición. México. Pearson. 2004. pp. 8, 17-18.

[7] Pressman R. *Ingeniería del Software un enfoque práctico,* 5ª edición. España. McGraw-Hill. 2002, pp. 20-24.

[8] De Miguel-Castaño A., Piattini-Velthuis M., Martínez E. *Diseño de bases de datos relacionales,* Colombia. Alfaomega. 2000. pp. 94, 343.

[9] Silberschatz A., Korth H., Sudarshan S. Bases de datos relacionales orientadas a objetos. En su: Fundamentos de Bases de Datos, 4ª edición. España. McGraw-Hill, 2002. pp. 211-226. ISBN: 84-481-3654-3

Recursos electrónicos

http://www.dei.uc.edu.py/tai2002/BDOO/index.html
http://www.microsoft.com/spain/medianaempresa/products/sql/evaluate.mspx#topBenefits
http://soporte.tiendalinux.com/portal/Portfolio/postgresql_ventajas_html
http://searchenterpriselinux.techtarget.com/tip/0,289483,sid39_gci1222466,00.html

Anexo A

Diccionario de datos

PIEL VIRTUAL

Campo	Tipo	Llave	Descripción
ID_PIEL	VARCHAR(32)	PRIMARY KEY	Id para las pieles virtuales
ID_GRADO	CHAR(1)	FOREIGN KEY	Porcentaje utilizable: A (98%), B (95), C (92), D(90), E(85), E*(75)
NO_MATERIAL	INTEGER	FOREIGN KEY	Identificador del tipo de piel
AREA_UTILIZABLE_TOTAL	INTEGER		El área utilizable en cada piel virtual
AREA_UTILIZABLE_ZONA_1	INTEGER		Área sin contar defectos de la zona 1
AREA_UTILIZABLE_ZONA_2	INTEGER		Área sin contar defectos de la zona 2
AREA_UTILIZABLE_ZONA_3	INTEGER		Área sin contar defectos de la zona 3
RUTA_ARCHIVO_PIEL	VARCHAR(256)		Path del archivo que contiene el archivo DIPI
FACTOR_CORTABILIDAD	INTEGER		Relación entre porcentaje y grado de la piel

MATERIAL

NO_MATERIAL	INTEGER	PRIMARY KEY	Identificador del tipo de piel
DESCRIPCION_MAT	VARCHAR(128)		Nombre del tipo de piel

GRADO

ID_GRADO	CHAR(1)	PRIMARY KEY	Porcentaje utilizable: A (98%), B (95), C (92), D(90), E(85), E*(75)
DESCRIPCION	VARCHAR(128)		Descripción de los porcentajes referidos por id_grado

CLUSTER

Campo	Tipo	Llave	Descripción
ID_CLUSTER	VARCHAR(32)	PRIMARY KEY	Identificador para cada cluster
ID_TALLA	INTEGER	PRYMARY & FOREIGN KEY	La talla a la que pertenece una pieza
DESCRIPCION_PROD	VARCHAR(32)	FOREIGN KEY	Nombre para cada modelo
PIEZA1	VARCHAR(32)		Pieza base
PIEZA2	VARCHAR(32)		Pieza secundaria
RUTA_CLUSTER	VARCHAR(256)		Path del archivo que describe la geometría del cluster
AREA	INTEGER		Área de piel cubierta por un cluster
ZC_PIEZA_BASE	INTEGER		Zona de calidad de la pieza base

PRODUCTO

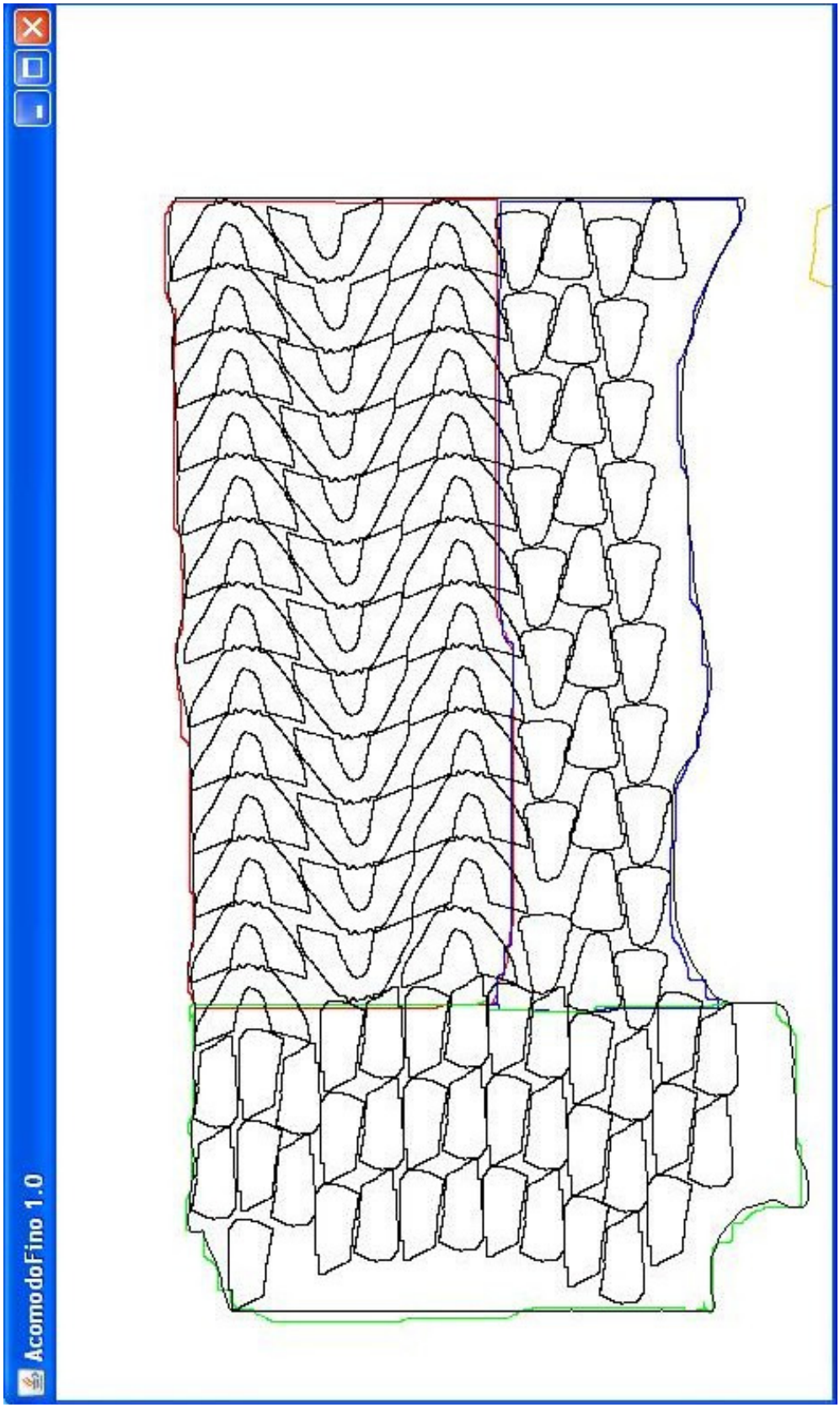
DESCRIPCION_PROD	VARCHAR(32)	PRIMARY KEY	Nombre para cada modelo
ID_TALLA	INTEGER	PRIMARY KEY	La talla a la que pertenece una pieza: 25,25.5…30,31
ID_PRODUCTO	LONG		No identificador de cada modelo
AREA_ZC_1	INTEGER		Área de la zona de calidad 1
AREA_ZC_2	INTEGER		Área de la zona de calidad 2
AREA_ZC_3	INTEGER		Área de la zona de calidad 3
RUTA_PIEZAS_DER	VARCHAR(256)		Ruta del archivo de piezas derechas
RUTA_PIEZAS_IZQ	VARCHAR(256)		Ruta del archivo de piezas izquierdas
MATERIAL	VARCHAR(128)		Tipo de piel

ORDEN TRABAJO

Campo	Tipo	Llave	Descripción
ID_ORDEN	VARCHAR(32)	PRIMARY KEY	Identificador de la orden de trabajo
ID_TALLA	INTEGER	FOREIGN KEY	La talla a la que pertenece una pieza
DESCRIPCION_PROD	VARCHAR(32)	FOREIGN KEY	Nombre para cada modelo
FECHA_ACOMODO	DATE		La fecha en la cual se llevó a cabo el proceso de nesting
FECHA_CORTE	DATE		La fecha en la cual se llevó a cabo el proceso de corte
CANTIDAD_PARES	INTEGER		Num de pares de calzado que deben procesarse

PRIORIDAD

ID_ELEMENTO	VARCHAR(32)	PRIMARY KEY	Identificador único para cada pieza o cluster
PRIORIDAD	INTEGER		El valor mayor indica el siguiente elemento a acomodar
AREA	INTEGER		Área del elemento
ZONA_CALIDAD	INTEGER		Calidad de la pieza o cluster, Puede tomar los valores: 1, 2 o 3.
NUM_FALTANTES	INTEGER		No de artículos que faltan por acomodar
PUNTOS	TEXT		Parejas *x,y* que describen la geometría del elemento


AcomodoFino 1.0