Capítulo 4.

Diseño de un sistema para reconocimiento y consulta de las tarjetas Hu

En este capítulo se describe el diseño de un sistema, denominado HuSystem, planteado para cumplir dos objetivos:

Búsqueda en línea de cualquier elemento interno al texto de las tarjetas, en forma rápida y confiable.

Integración sencilla a los desarrollos actuales del laboratorio.

En un principio se planteó la opción de efectuar el reconocimiento en línea, es decir, efectuar búsquedas sobre la imagen al tiempo en el que el usuario introdujera el texto a buscar. Sin embargo, por las pruebas efectuadas con diferentes reconocedores fue sencillo determinar que el tiempo para efectuar esta búsqueda sería muy por encima de lo aceptable.

Tomando en cuenta que el proceso de OCR tarda aproximadamente 1 minuto por imagen, multiplicar por 350,000 tarjetas (número aproximado de tarjetas) nos arrojaría la exorbitante suma de aproximadamente 350,000 minutos (5833 horas o 243 días). Ese tiempo no incluye la inserción de los resultados a la base de datos ni el tiempo de transferencia de los archivos de su sitio original a la máquina de reconocimiento. Además esto es tomando en cuenta que no existan interrupciones durante el trabajo.

Por esta razón se decidió dividir la tarea en dos componentes:

Un subsistema que construye la base de datos usando OCR, creando así un almacén de imágenes y texto reconocido en ellas.

Un componente que usa la base de datos preconstruida por el anterior para efectuar consultas acerca de las tarjetas, además de proveer al usuario experto la capacidad de mejorar la calidad del texto reconocido.

De esta manera la búsqueda del contenido de las tarjetas se hace sobre el texto reconocido y no sobre la imagen en sí. Lo cual hace que el tiempo de búsqueda sea sólo el tiempo de una búsqueda típica en la base de datos.

El sistema se integra a la arquitectura de la FDL (Figura 1.1) en el módulo llamado ”Cards Management”, que se encarga del manejo de

las tarjetas, en éste se encuentra el desarrollo hecho por Miriam Amavizca y César Flores, denominado HuBrowser. Además, podrá ser usado como parte del ambiente del 3Dtree [Amavizca 1998].

Los dos sistemas permiten la navegación del usuario en las tarjetas usando los niveles de la taxonomía en la que están clasificadas. El problema de éstos es que no se puede hacer búsquedas en el texto que está dentro de las tarjetas, la cual es la facilidad de el sistema propuesto.

Veremos ahora, las dos partes que contituyen la solución planteada en este trabajo: El HuProcessor y el HuSeeker.

4.1 HuProcessor

El primero de los dos subsistemas fue llamado HuProcessor, su función es la de transformar las tarjetas en texto, efectuando todos los procesos intermedios de transporte, reconocimiento y almacenaje.

4.1.1 Dificultades

Para el diseño del sistema se plantearon ciertos obstáculos a resolver, entre ellos:

La lentitud en el reconocimiento de texto en imágenes.

El gran volumen de información manejado, el cual hace que no sea factible procesar todas las tarjetas a la vez, es un proceso tedioso y repetitivo, pues requiere la transportación de lotes de archivos al reconocedor, efectuar el reconocimiento y posteriormente insertar los resultados a la base de datos, lo cual puede generar problemas de duplicación de datos, por errores humanos, entre otros.

4.1.2 Solución

Al analizar los requerimientos se optó por paralelizar y distribuir el trabajo, para aprovechar el poder de varios CPUs y así reducir el tiempo total de procesamiento.

Figura 4.1: El HuProcessor, distribuye imágenes a los esclavos, y a su vez recolecta los resultados que éstos producen.

Se adoptó un esquema maestro-esclavo [Hwang & Briggs 1984], en la cual un procesador se encarga de repartir trabajo a diferentes esclavos, (Figura 4.1) observándolos en su desarrollo, evaluando y recolectando su trabajo. Así, el HuProcessor hace el trabajo de coordinador de procesos entre todos los CPUs.

La arquitectura maestro-esclavo provee una sencillez para programar la concurrencia, pues una máquina solamente se encarga de coordinar a las demás. Las máquinas esclavas no se preocupan por la concurrencia, trabajan línealmente.

El diseño se realizó en dos etapas, primero se modeló analizando el flujo de datos y los procesos que este flujo incluye. Posteriormente se modeló este flujo creando objetos que tuviesen tareas específicas para manejar dicho flujo. Por ello, a continuación se presentan los modelos funcional y de objetos, de acuerdo a la metodología llamada *Técnica de Modelado de Objetos* (OMT) [Rumbaugh, et. al. ,1991].

4.1.3 Diagramas de Flujo de Datos

El diseño se presenta usando en primera instancia el diagrama de flujo de datos, y su correspondiente diccionario de datos.

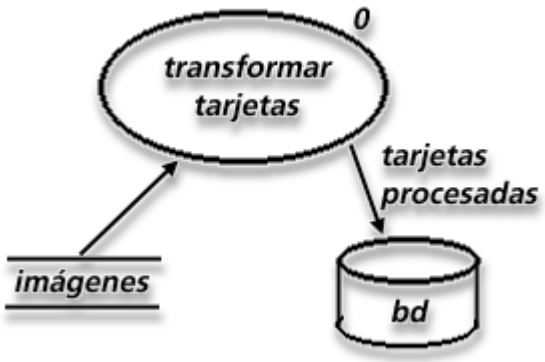

Figura 4.2: Primer Nivel (Nivel cero) del diagrama de flujo de datos para el HuProcessor

El nivel cero (Figura 4.2) corresponde a la descripción general del procesamiento de transformación de las tarjetas. Cabe mencionar el uso del término transformación, que significa un cambio de forma, pues las tarjetas pasan de ser meras imágenes a ser una combinación de texto e imagen. Se muestra claramente cómo se obtienen las tarjetas en forma de imagen de un dispositivo de almacenamiento con un sistema de directorios jerárquicos y se depositan los resultados en la base de datos preparada para el caso.

El esquema de la base de datos diseñada es:

Tabla hucards

Nombre descriptivo	Nombre de archivo	Texto reconocido	Imagen	Nombre de editor
char[]	char[]	char[]	byte[]	char[]

Tabla usuarios

username	password	privilegio
char[]	char[]	char[]

El nombre descriptivo se refiere al nombre de la planta que se trata en la tarjeta, el nombre de archivo es el nombre físico que tiene la imagen que contiene la tarjeta. El privilegio del usuario se refiere al papel que éste juega para el sistema, si el usuario puede hacer cambios el atributo tiene el nombre de ”modify”.

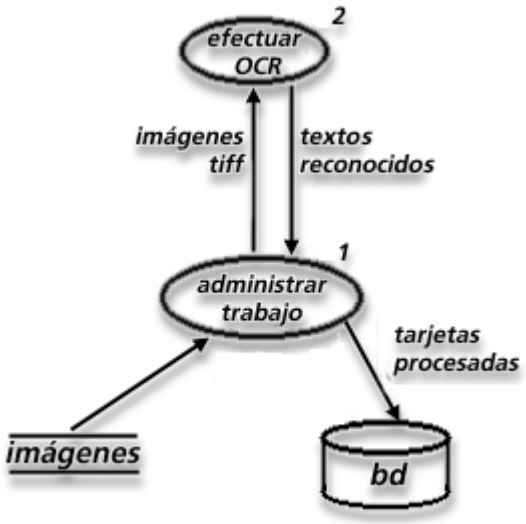

Figura 4.3: Diagrama a segundo nivel del sistema.

En el segundo nivel se puede observar que la división del sistema en dos partes se debe a que se diseñó específicamente para usar paralelismo, el proceso 1 se dedica a administrar el trabajo entre las diferentes entidades ”esclavas” del sistema, y el proceso 2 se dedica a efectuar el reconocimiento (Figura 4.3).

Figura 4.4: Diagrama a tercer nivel, descomposición de 1

Al descomponer aún mas las tareas que se efectúan en el lado de la administración (Figura 4.4), veremos que por administración de trabajo se entiende la distribución , monitoreo y recolección de resultados. Todos estos procesos también se ejecutan en paralelo. Sin embargo, para que puedan coexistir sin interferir con otros hay reglas que se deben cumplir. Es decir, se tiene que diseñar la manera en que sólo se puedan recolectar trabajos que ya estén terminados, y que no

se envíen trabajos si la máquina esclava no está operando.

Diccionario de datos del HuProcessor

Flujos de Datos

imágenes = {imagen tiff | imagen GIF}

imágenes GIF= { imagen GIF}

imágenes tiff= { imagen tiff}

imagen GIF= {byte} (en formato GIF)

imagen tiff= {byte} (en formato tiff)

lista de trabajo= {nombre de archivo}

lista de resultados= {nombre de archivo}

nombre de archivo= {char}

tarjeta de Hu= imagen tiff | imagen GIF | tarjeta procesada

tarjetas procesadas= { tarjeta procesada }

tarjeta procesada= imagen GIF+texto reconocido

textos reconocidos= {texto reconocido}

texto reconocido= {char}

El diccionaro de datos define los datos usados, nótese la definición de tarjeta Hu, en realidad para el usuario debe ser transparente si se trata de una imagen en algún formato u otro, o si se trata de un texto.

4.1.4 Diseño OMT

Se puede observar a simple vista que las tareas se pueden dividir en diversas entidades, las cuales se pueden encargar de efectuar tareas diferentes de acuerdo a su papel o rol.

Para ello se crearon 3 clases que dividen la tarea de transformación en:

Una entidad que se encarga de administrar la información general común a las múltiples instancias de las segundas dos clases. A esta se le llamó Maestro.

Una entidad que obtiene lotes de tarjetas y los envía a la máquina esclava, así como también se asegura de que la máquina esclava siga trabajando. Esta clase se llamó Esclavo.

Una clase que recolecta los trabajos al mínimo tiempo, es decir, justo cuando la máquina esclava los termine de crear, y que además se encargue de generar e insertar la información completa en la base de

datos. Su nombre es Recolector (Ver Figura 4.5).

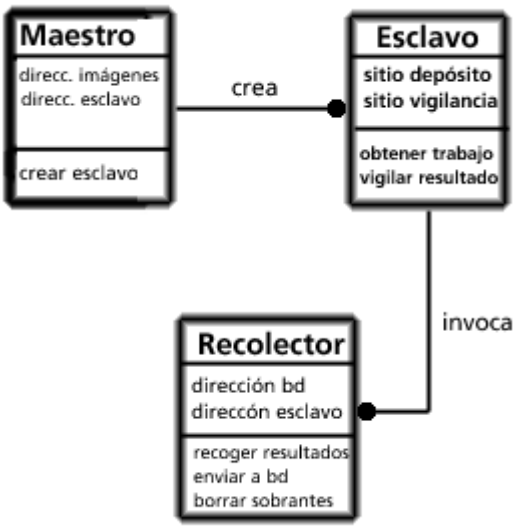

Figura 4.5: Modelo orientado a objetos del HuProcessor

La clase Maestro tiene los datos comúnes a todos los esclavos, como lo es la localización de las imágenes en alguna máquina remota, y la dirección de cada esclavo. Además es ésta clase la que tiene el poder de crear múltiples instancias de la clase esclavo, para que cada una de ellas trabaje con una máquina esclava diferente.

La clase esclavo, a su vez, se encarga de obtener una lista de trabajo, la cual tiene que procesar, primeramente transportando los archivos de su sitio original hacia la máquina esclavo de la que él se encarga. Mientras envía los archivos a la máquina esclava, la clase esclavo crea una instancia de la clase recolector, específicamente para recolectar los resultados del OCR de la máquina esclava. Este recolector se encarga de permanecer activo hasta obtener todos los resultados, de no hacerlo, espera un tiempo razonable y continúa trabajando.

Después el esclavo se asegura de la terminación del trabajo y envía un nuevo lote de imágenes, comenzando de nuevo el ciclo de trabajo.

4.2 HuSeeker

Este es el sistema que utiliza los resultados obtenidos por el HuProcessor para efectuar búsquedas acerca del contenido de las tarjetas.

Para este propósito se consideró que la búsqueda de texto en imágenes debía ser una facilidad adicional a los sistemas

desarrollados en el mismo laboratorio. Por ello se trató de usar una arquitectura abierta y un diseño sumamente simplificado.

El HuSeeker se diseñó de la misma manera que el HuProcessor, analizando primeramente el flujo de datos y los procesos que los usan, y posteriormente se planteó la estructura orientada a objetos.

4.2.1Diagramas de Flujo de Datos

El flujo de datos es simple, se podría ver como dos entidades que acceden al sistema, pero los dos dan los mismos datos, y reciben el mismo tipo de información.

Esto es debido a que el usuario puede acceder al HuSystem directamente dirigiéndose a él, pero también puede usarlo sin saber que existe, pues puede estar escondido como un servicio adicional al HuBrowser o 3DTree [Amavizca 1998].

Como podemos ver en la Figura 4.6 el usuario tiene la opción de identificarse, pues si es un experto, éste puede editar la tarjeta, para mejorar el texto que esté en ella.

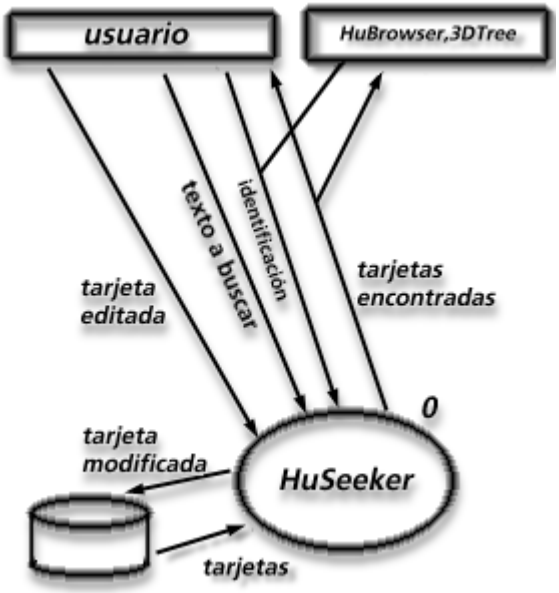

Figura 4.6: Diagrama de flujo de datos a primer nivel del HuSeeker

El diagrama en la figura 4.7 nos muestra cómo se descomponen las tareas del HuSeeker. El usuario es identificado para saber si es un editor experto o si sólamente es un usuario de la Biblioteca Digital. De esa manera, los dos pueden buscar tarjetas, pero sólo el experto puede modificar la tarjeta.

Figura 4.7: Segundo nivel del DFD del Huseeker

Diccionario de Datos del HuSeeker

Flujos de Datos

tarjeta editada= texto

tarjeta modificada= texto + nombre

texto a buscar= texto

identificación= nombre + password

tarjetas encontradas= tarjetas

tarjetas= {tarjeta}

tarjeta = imagen + texto

texto= char[]

nombre=char[]

password=char[]

Se puede notar que en esta definición se generaliza a las tarjetas como el texto y la imagen, junto con el autor que modificó la tarjeta por última vez, si lo hay. Al enviar el usuario la tarjeta editada, ésta se asocia con el nombre del usuario y se convierte en la tarjeta modificada. Este será el sello del autor, para que expertos que posteriormente consulten la misma tarjeta sepan quién la revisó. El nombre y password pueden ser variables que vengan de otro sistema, así la identificación se haría fuera del HuSeeker. En realidad, los pocos atributos de esta clase la hacen fácil de contemplar como adición a otros diseños posteriores.

4.2.2 Diseño OMT

Para implementar en un lenguaje orientado a objetos, como lo es Java, es indispensable tener un diseño orientado a objetos, el HuSeeker se modeló como un sólo objeto.

Este ojeto se encarga de una sóla tarea, encontrar una lista de tarjetas

que cumplan con las especificaciones dadas por el usuario.

Figura 4.8: Diagrama de objetos del HuSeeker

La sencillez de las operaciones del HuSeeker se refleja en su diseño, pues consta de una sola clase (Figura 4.8).

Entre los atributos que lo hacen operar está el texto que busca el usuario y la identificación del mismo. De esta manera él sabrá si el usuario puede hacer uso de su método de modificación de tarjeta.

Dircio Palacios Macedo, R. 1998. Reconocimiento y Consulta de Imágenes Textuales en Bibliotecas Digitales. Tesis Licenciatura. Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas-Puebla. Diciembre.