UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELAS DE INGENIERÍA
EN COMPUTACIÓN Y ELECTRÓNICA

"DESARROLLO DE UNA HERRAMIENTA DE ACCESIBILIDAD DEL TIPO LECTOR DE PANTALLA"

PROYECTO DE GRADUACIÓN PARA OPTAR AL GRADO DE
INGENIERO EN COMPUTACIÓN / ELECTRÓNICA

PRESENTADO POR:
ALVIN MIGUEL FLORES BRAN
ALICIA IVETTE SALINAS BENÍTEZ
LAURA YANIRA SALAZAR CALDERÓN

CIUDADELA DON BOSCO SEPTIEMBRE 2002

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

INTRODUCCIÓN

Desde hace algunos años la tecnología informática comenzó a estar disponible para la comunidad no vidente gracias al desarrollo de programas de computadora especializados, que tienen la capacidad de capturar la información que se presenta en la pantalla, y posteriormente transmitirla al usuario de forma verbal, o a través de periféricos especializados que simulan el alfabeto braile.

Los programas que incorporan estas funciones son conocidos con el nombre de *lectores de pantalla*. Hoy en día, constituyen la principal herramienta de *accesibilidad* en el mundo informático para las personas con discapacidades visuales, dado que sin estos programas las computadoras serían prácticamente imposibles de utilizar.

La disponibilidad de estas herramientas se ha incrementado debido tanto al avance de la tecnología, como a la necesidad de los discapacitados visuales de no permanecer marginados del mundo de la informática. De esta forma, es posible encontrar desde versiones gratuitas de lectores de pantalla, hasta versiones comerciales básicas y profesionales, cuyos precios oscilan entre $75 y $2100.

Entre la gran variedad de lectores de pantalla que existen, algunos comparten características similares y otros presentan mayor funcionalidad que el resto. También ocurre que a pesar de contar con muchas características, algunas veces carecen de detalles presentes en programas más sencillos, existiendo la posibilidad de ofrecer herramientas más completas que atiendan con mayor eficacia las necesidades de accesibilidad asociadas a las personas no videntes.

Relacionando directamente funcionalidad y precio, aquellos lectores de pantalla que ofrecen mayores facilidades al usuario no vidente para el uso de una computadora, son al mismo tiempo los de mayor costo en el mercado,

convirtiéndose en herramientas de lujo y de acceso limitado para la mayoría de la población no vidente en El Salvador, tomando en cuenta el estado de marginación, abandono y pobreza en que muchas de estas personas viven.

Junto al alto costo de los lectores de pantalla y a la inexistencia de estos productos en el mercado local, se encuentra la falta de apoyo económico y legislativo por parte del estado, lo que limita aún más las posibles oportunidades y beneficios que el uso de estas tecnologías de accesibilidad proporcionaría a la población no vidente.

Ante situaciones como ésta, el aporte que pueden brindar otros sectores, como la comunidad educativa, e instituciones no gubernamentales, adquiere gran importancia. Es así como a pesar de que la población de discapacitados no videntes de nuestro país ha sido marginada de la sociedad y del mundo laboral a través de los años, existen en la actualidad instituciones que impulsan y promueven programas para el desarrollo e inserción de estas personas, dentro de la sociedad activa y productiva del país. Sin embargo, el alcance de tales programas de desarrollo y capacitación, se ve limitado por el poco apoyo que reciben dichas instituciones, y por el escaso número de éstas en el país, siendo así mínima la población no vidente beneficiada por estos programas.

Para la comunidad educativa, el área de investigación constituye una de las principales alternativas para contribuir con el desarrollo de distintos sectores de la población, y junto al hecho de que actualmente no existen en nuestro país estudios previos acerca del desarrollo de herramientas como los lectores de pantalla, constituyen una de las principales motivaciones para la realización de este proyecto, cuyo fin social consiste en establecer un antecedente en la elaboración de herramientas de accesibilidad para discapacitados, enfatizando la importancia de tomar en cuenta a este sector de la población en futuras investigaciones que faciliten y agilicen su inserción dentro de la sociedad activa de El Salvador.

Si bien los resultados del mismo no están orientados a resolver el problema de la falta de apoyo legislativo o económico, para la adquisición de herramientas especiales como los lectores de pantalla, se espera que a través de la documentación de los fundamentos teóricos utilizados y del código a emplear para el diseño y desarrollo de un lector de pantalla básico, se establezca un antecedente de utilidad para el estudio y desarrollo de herramientas de accesibilidad de este tipo, y para otros proyectos que beneficien el desarrollo de la comunidad de discapacitados en general.

El programa a desarrollar no pretende entrar en ningún tipo de competencia con programas existentes en el mercado actualmente, en primer lugar, porque no será un producto destinado al mercado, sino un prototipo de investigación; y en segundo lugar, porque será desarrollado a un nivel básico, es decir, que cumplirá algunas de las funciones que son básicas para todo lector de pantalla, como la detección de la ventana en la que el usuario se encuentra trabajando, y la interacción con otros objetos presentes en la pantalla.

II. OBJETIVOS

2.1 OBJETIVO GENERAL

- Desarrollar una aplicación que permita y facilite a las personas con discapacidad visual, la utilización de elementos básicos del sistema operativo Windows y de aplicaciones basadas en el mismo, a través de la lectura del contenido de la pantalla.

2.2 OBJETIVOS ESPECÍFICOS

- Investigar y estudiar elementos de programación que puedan ser empleados para la detección de los objetos que componen una aplicación, así como el texto que se presenta en la pantalla de la computadora como consecuencia de las acciones ejecutadas por el usuario, por la aplicación en curso, y por el sistema operativo.

- Implementar los resultados de la investigación en el desarrollo de una aplicación del tipo lector de pantalla que proporcione al usuario no vidente un nivel de accesibilidad básico al entorno del sistema operativo y sus aplicaciones.

- Incorporar a la aplicación un sintetizador de voz en español, que haciendo uso de la tarjeta de sonido incluida en la computadora, transmita oralmente al usuario la información detectada en la pantalla.

- Facilitar la interacción del usuario con las aplicaciones, transmitiéndole oralmente información adicional, tal como el tipo y estado de los elementos contenidos en las mismas.

- Establecer un precedente, en el estudio e investigación de elementos y herramientas de accesibilidad, para futuras investigaciones que pretendan desarrollar un lector de pantalla de mayor complejidad que el programa básico que se desarrollará en este proyecto.

III. ALCANCES Y LIMITACIONES

3.1 ALCANCES

- Como programa lector de pantalla, la aplicación a desarrollar proporcionará a los usuarios las siguientes características:

- ***Detección de la ventana activa.***

  La ventana activa es aquella que tiene la capacidad de recibir simultáneamente las acciones ejecutadas por el usuario a través del teclado y el ratón. Su detección consiste en la lectura del título de la ventana, el cual puede ser o no visible.

- ***Lectura de las teclas presionadas.***

- ***Reconocimiento de objetos estándares presentes en las aplicaciones, tales como:***

  - Botones de comando (Command Button)
  - Botones de opción (Radio Button)
  - Cuadros de chequeo (Check Box)
  - Cuadros de texto (Text Box)
  - Cuadro de combos (Combo Box)
  - Control estático (Static Control)
  - Vista de lista (List View)
  - Vista de árbol (Tree View)
  - Barra de herramientas (Tool Bar)
  - Ficheros (Tab Control)
  - Menús no flotantes (Menu)

Mediante el reconocimiento de estos objetos, será posible interactuar con las aplicaciones que los contengan. Se indicará el tipo de objeto y el texto contenido en el mismo.

- ***Lectura de la información presente en cuadros de mensaje.***

- ***Lectura del texto contenido en archivos ".txt" (archivos de Notepad) existentes, y del texto digitado por el usuario.***

- ***Lectura del texto contenido en archivos ".doc" (documentos de Word 97 ó 2000) existentes, y del texto digitado por el usuario.***

- ***Lectura del contenido de las celdas en archivos ".xls" (hojas de cálculo de Excel 97 ó 2000) existente, y del texto digitado por el usuario.***

- El sintetizador de voz que será utilizado para la lectura de la información detectada tendrá la ventaja de poderla vocalizar en español, facilitando de este modo su comprensión por parte del usuario.

- La aplicación incorporará scripts, con los cuales puede lograrse o mejorarse el acceso a las aplicaciones que utilizan elementos que no forman parte de la colección de objetos estándar de Windows. Los scripts consistirán en archivos de texto ".txt" en los que se almacenará información sobre las posibles combinaciones de teclas que permiten utilizar la aplicación.

- La aplicación incorporará un sistema de ayuda, adicional a la brindada por otros programas lectores de pantalla, que permitirá al usuario tener información complementaria de los objetos y aplicaciones propios del sistema operativo.

- A través de la investigación se resalta la importancia de la incorporación de opciones de accesibilidad para discapacitados al desarrollar proyectos y sistemas orientados al área de informática, así como la creación de herramientas que permitan accederlos.

- El proyecto establecerá un precedente para la implementación de aplicaciones e investigaciones futuras que beneficien a la comunidad con discapacidades visuales.

- Los principios y elementos básicos utilizados para el desarrollo del sistema son aplicables a futuros proyectos que se desarrollen en plataformas posteriores a Microsoft Windows 95/98.

- La estructura del programa será por módulos, así cada modulo trabajara sobre una aplicación o parte del sistema operativo, esto otorgará facilidades a futuros desarrolladores que pretendan hacer cambios o mejoras al mismo.

- La forma de acceder a la información de algunos elementos del sistema operativo y sus aplicaciones es generalizada, es decir que todas aquellas aplicaciones que compartan estos elementos podrán ser accedidas por el programa lector de pantalla.

3.2 LIMITACIONES

Desde su etapa inicial, la investigación se vio limitada por los siguientes factores:

- Falta de promoción y conocimiento en El Salvador, de la existencia y utilidad de las medidas y herramientas de accesibilidad.

- Ausencia de investigaciones, estudios previos, o personal capacitado, que provean información sobre el desarrollo de las herramientas de accesibilidad del tipo a realizar en el proyecto.

Los factores antes mencionado impiden en gran mediad el desarrollo de una aplicación de mayor alcance y funcionalidad. Por ello, la aplicación a realizar no incluirá todas las características que debería poseer un lector de pantalla para ser un sistema totalmente funcional, por tanto queda a nivel de prototipo de investigación.

Las limitaciones que presenta el prototipo son las siguientes:

- El prototipo desarrollado es de nivel básico, y se basa en el reconocimiento de objetos estándares del sistema operativo.

- La aplicación puede ejecutarse únicamente bajo los sistemas operativos Windows 95/98.

- En la vista “detalles” de un objeto tipo vista de lista, sólo reconoce el nombre de los ítems pertenecientes a este objeto.

- Dado que los controles estáticos no pueden aceptar el enfoque del teclado, sólo se reconocen en casos específicos, como al estar asociados a un cuadro de edición, o a un cuadro de combos.

- No puede leer algunos cuadros de lista que se despliegan dentro de los controles de tipo cuadros de combos.

- No puede detectar el nombre de la ventana que se escoge mediante la combinación de teclas Alt + Tab, sino hasta el momento en que alguna se convierta en la ventana activa.

- No puede reconocer los ítem del submenú Nuevo, que es parte tanto del menú contextual como del menú Archivo del Explorador de Windows.

- Al acceder a la opción de búsqueda del sistema operativo y realizar la búsqueda de un determinado elemento, no reconoce cuando la búsqueda haya finalizado.

- Durante la exploración de un objeto de vista de árbol no es posible reconocer cuando un ítem esta expandido o contraído.

- El reconocimiento de las aplicaciones Microsoft Word y Microsoft Excel, se realiza a un nivel básico. No se reconocen detalles como el formato o color del texto. La Mayoría de los cuadros de diálogo utilizados en estas aplicaciones no utilizan controles estándares, por lo que no pueden ser reconocidos.

IV. MARCO TEÓRICO

4.1 LECTOR DE PANTALLA

Generalidades

El lector de pantalla (Screen Reader, en inglés) es un programa de computadora que realiza la detección del tipo, estado, y texto, de los distintos elementos presentes en la pantalla, tales como ventanas, íconos, menús, y botones, entre otros. A medida que la información es detectada, el programa recurre a un *sintetizador de voz* para hacerla llegar al usuario como una voz artificial que se escucha a través de los altavoces de la computadora.

Estos programas se utilizan principalmente como herramientas que permiten o facilitan la utilización de una computadora a las personas no videntes o con poca visión. Sin embargo, dadas sus características, los lectores de pantalla también suelen ser utilizados para tareas específicas tales como la lectura automática de mensajes y documentos de texto, y la confirmación auditiva de las teclas presionadas, y de las opciones o comando seleccionados en la pantalla.

Las primeras versiones de lectores de pantalla fueron diseñadas para leer solamente texto, puesto que en ese entonces los sistemas operativos (como DOS y UNIX) eran basados en texto. Estas versiones encontraron dificultades al surgir los sistemas operativos con interfases gráficas para los usuarios (en inglés, Graphical User Interface ó GUI), como Windows y Mac OS, en los cuales el texto dejaba de ser el único recurso que debía ser atendido. Sin embargo, las versiones recientes han superado estos problemas y realizan la detección de la información que presentan en pantalla los nuevos sistemas operativos. Algunos de los programas disponibles en el mercado se presentan en la tabla 4.1. Todos ellos funcionan para los sistemas operativos Windows. Algunos se utilizan tanto para Windows como para el DOS, y otros tienen versiones para Windows y Macintosh.

Tabla 4.1 Lista de Programas Lectores de Pantalla

Programa	Sistema Operativo	Fabricante
ASAP	Windows	MicroTalk
Simply Talker 98	Windows 98	EconoNet International Speech Series
HAL	Windows	Dolphin Systems
JAWS		Henter – Joyce Inc.
OutSpoken	Macintosh / Windows	ALVA Access Group Inc.
Screen Reader/2	Windows, OS2, y DOS	IBM
Screen Power	Windows	Telesensory Corp.
Windows Bridge		Syntha – Óbice Computers, Inc.
Window Eyes		GW Micro
WINKLINE		Speech Systems for the Blind
WinVision	Windows & DOS	Artic Technologies

Para trabajar con una herramienta de este tipo, el usuario utiliza el teclado de la computadora tal como lo haría con cualquier otra aplicación, es decir, para interactuar con el sistema operativo y las aplicaciones desarrolladas para el mismo. La aportación de la herramienta consiste en que a medida que el usuario navega en los programas y aplicaciones, recibe a través de los altavoces, indicaciones sobre las teclas presionadas, la ubicación tras cada movimiento realizado, los mensajes desplegados en pantalla, y el texto, tipo, y estado del elemento hacia el que se ha desplazado.

Algunos lectores de pantalla también permiten utilizar dispositivos conocidos como *matrices braile* (braille displays, en inglés), los cuales se conectan externamente a la computadora y tienen la finalidad de representar en alfabeto braile el texto detectado en la pantalla. Estos dispositivos contienen un número fijo de celdas que son utilizadas para la representación de los caracteres braile correspondientes a las letras que forman el texto detectado por el lector de pantalla. El usuario no vidente (y con conocimientos en la técnica de lectura braile) realiza la lectura de la información detectada, colocando su mano sobre dichas celdas.

Figura 4.1 Matriz Braile

El teclado de la computadora se coloca usualmente por encima del dispositivo, teniendo de esta forma un fácil acceso tanto a la matriz como al teclado convencional.

Si bien esta tecnología provee, junto al sintetizador, mayor soporte al usuario, su principal desventaja radica en su alto costo. Los precios de éstos dispositivos resultan más elevados que el del propio lector de pantalla, y dependen en gran medida del número de celdas con las que cuenten. En promedio, un dispositivo de 40 celdas cuesta alrededor de seis mil dólares. La tecnología de síntesis de voz (en la mayoría de los casos incluida en el lector de pantalla) en contraste, resulta mucho más económica y sustituye eficientemente el sentido de la vista por el sentido del oído.

Estructura Básica

En forma general, la estructura básica de un lector de pantalla involucra la ejecución de distintos procesos en los que tienen lugar, la detección de información, la conversión de la información detectada a texto, y la conversión del texto a voz, proceso conocido como *síntesis de voz*. En algunos casos, la aplicación lee datos contenidos en archivos (scripts) que se adicionan a la misma con el objetivo de complementar la información detectada.

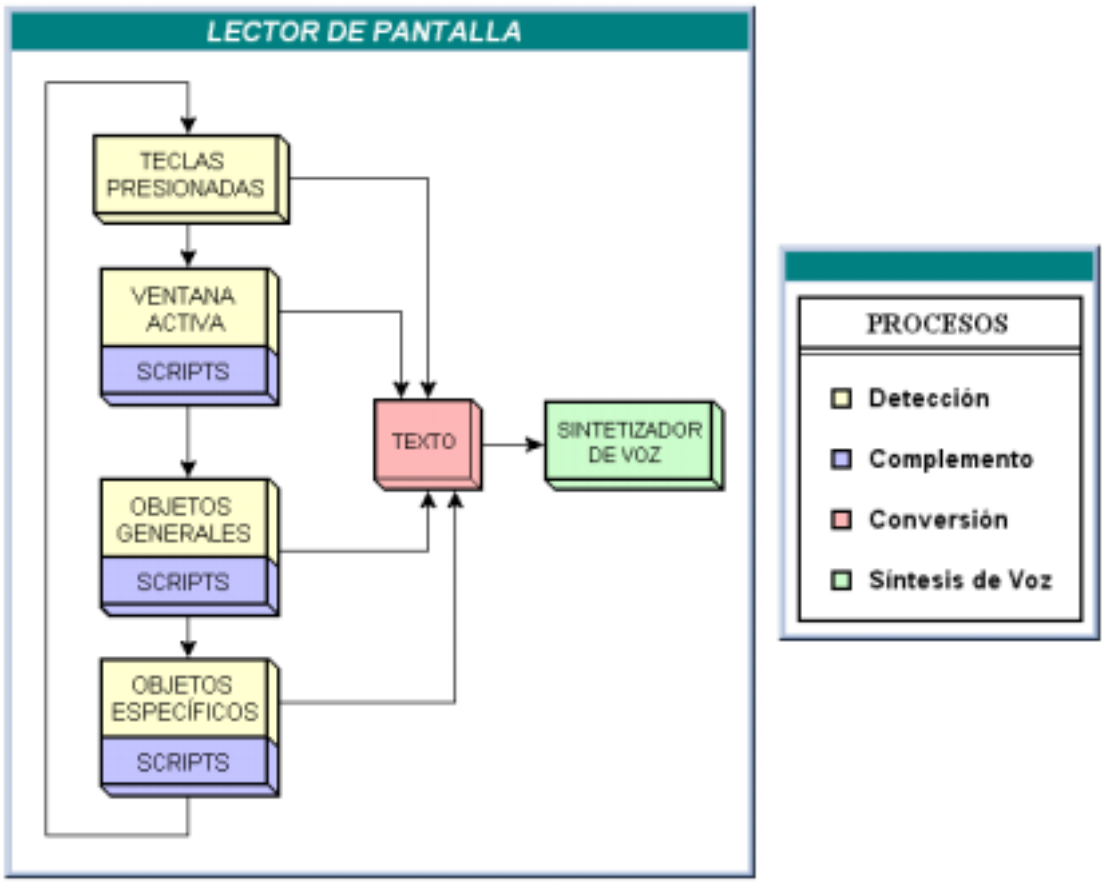

Figura 4.2 Estructura Básica de un Lector de Pantalla

Todas las etapas que componen el proceso de detección de información, pueden ser ubicadas dentro de un bucle principal en el que constantemente se verifican cambios ocurridos en la pantalla de la computadora, debido a la pulsación de teclas, a la activación de ventanas, y a la selección o modificación de los objetos (botones, íconos, ítems de menú, y cuadros de texto, por ejemplo) que las componen.

Al ser detectada alguna información, ésta se convierte en una cadena de texto, que es el formato de datos aceptado por el sintetizador de voz. Una vez que el texto llega a este componente del lector de pantalla, el proceso de síntesis genera una voz artificial que se transmite al usuario a través de los altavoces de la computadora.

- **Teclas Presionadas**

  Etapa que consiste en la detección de las teclas presionadas por el usuario.

  Con un lector de pantalla, como en toda aplicación, el usuario utiliza el teclado de la computadora para interactuar con el sistema operativo y las aplicaciones desarrolladas para el mismo. A través del teclado se generan distintas acciones ante las que el sistema operativo y sus aplicaciones deben responder. Entre éstas se tiene la ejecución o finalización de una aplicación, la navegación sobre los distintos elementos que la componen, la selección de objetos, y la escritura de texto.

  El lector de pantalla también responderá a las acciones que el usuario genera con el teclado, realizando la detección de los posteriores cambios que tienen lugar en los objetos presentes en la pantalla, y la detección de las teclas presionadas.

  Éstas se identifican mediante un código que es específico para cada una de ellas. Posterior a la identificación, se da a conocer la tecla presionada, asignando una cadena de texto que se envía al sintetizador de voz para ser escuchada por los altavoces.

  En ocasiones, recibir la información de las teclas presionadas puede resultar innecesario. Al desplazarse por distintos elementos, es posible que sea de mayor utilidad escuchar únicamente los ítems que son seleccionados. Por esta razón la confirmación de teclas presionadas suele brindarse como parte de las opciones de configuración.

- **Ventana Activa**

  Etapa que consiste en la detección del título de la aplicación o ventana activa, es decir aquella que recibe las entradas del teclado.

  El lector de pantalla permanece constantemente verificando si la ventana que captura las entradas del teclado ha cambiado, situación que puede

presentarse cuando el usuario abre una nueva aplicación o ventana para trabajar en ella.

Al detectar una nueva ventana activa, el lector de pantalla recupera el texto presente en la barra de título de dicha ventana. Esta operación ocurre igualmente cuando el usuario se desplaza hacia ventanas que ya han sido creadas, o en general cada vez que una nueva ventana se convierte en la ventana activa.

Dado que la información (el título de la ventana activa) se recupera como una cadena de texto, se envía directamente al sintetizador de voz para ser escuchada por los altavoces de la computadora.

- **Objetos Generales**

  Etapa que consiste en la detección del tipo, estado, y texto contenido en objetos generales pertenecientes a la ventana activa.

  Los objetos son considerados generales ó estándares, cuando son definidos por el sistema operativo, es decir que éste provee los métodos para acceder sus características, entre las que se incluyen, el tipo de objeto, su estado, y su contenido.

Figura 4.3 Ejemplo de Barra de Menús Estándar (Menú de NotePad)

La barra de menús contenida en la aplicación NotePad, constituye un ejemplo de un objeto estándar.

Al seleccionar el ítem Word Wrap perteneciente al menú Edit, como se muestra en la figura, el lector de pantalla usualmente detecta la siguiente información acerca del objeto:

- Tipo de Objeto: “ítem de menú”
- Contenido: “Word Wrap”
- Estado: “Chequeado”

Los objetos estándar son creados y definidos por el sistema operativo con el fin de proveer a los programadores, elementos listos para acoplar a las interfases gráficas de sus aplicaciones. Este hecho facilita el trabajo que realizan herramientas como los lectores de pantalla, dado que para todas las aplicaciones que contengan objetos generales de un mismo tipo, el proceso de detección a utilizar será el mismo.

- **Objetos Específicos**

  Etapa que consiste en la detección del tipo, estado, y texto contenido en objetos pertenecientes a la ventana activa, y que no forman parte del conjunto de objetos definidos por el sistema operativo.

  Estos objetos son definidos por la aplicación específica a la que pertenecen. El acceso a sus características es posible mientras la aplicación provea los métodos para realizarlo.

  Como ejemplo, una hoja de cálculo de Excel, o un documento de Word, constituyen objetos creados y definidos por dichas aplicaciones, y es poco probable que otros programas (fuera de la familia de productos Microsoft Office) los utilicen y definan como lo hacen sus respectivas aplicaciones. Objetos como éstos, son usualmente detectados atendiendo a la aplicación a la que pertenecen, y de acuerdo a los métodos definidos por las mismas.

  La ventaja de utilizar otros objetos, además de los ofrecidos por el sistema operativo, reside en la capacidad de mejorar la apariencia o interacción de las aplicaciones, y en general de no limitar la creatividad o las necesidades del diseñador de aplicaciones. Sin embargo, para un lector de pantalla tal libertad podría representar un obstáculo en su tarea de informar al usuario no vidente

sobre lo que está ocurriendo en pantalla, si se trabaja únicamente en los aspectos visuales o funcionales, sin tomar en cuenta cómo otras aplicaciones podrían tener acceso a sus elementos. Por esta razón, en términos de accesibilidad, el mayor compromiso que se adquiere al utilizar este tipo de objetos en las aplicaciones, es el de proveer los métodos necesarios para asegurar que la aplicación podrá ser reconocida por herramientas como los lectores de pantalla.

- **Scripts**

  Etapa que consiste en complementar los procesos de detección de información sobre los objetos que componen la ventana activa.

  Durante las etapas de detección de información, el lector de pantalla puede auxiliarse de scripts ya sea para complementar la información acerca de los objetos o para hacer posible la detección de la misma. Cada script se define para una aplicación específica, y contiene información adicional para llevar a cabo la detección de objetos en dicha aplicación, para incorporar nuevas teclas de acceso, o para proporcionar información adicional acerca de la misma.

  Algunos scripts ofrecen como ventaja adicional, que el código del lector de pantalla no necesita ser modificado para hacer posible la interacción con otras aplicaciones. En su lugar, la información que posibilita dicha interacción se incluye en estos archivos, y sólo se agregan en un directorio ya conocido por el lector de pantalla.

  Los scripts son de bastante ayuda para incrementar y mejorar la capacidad de detección de los lectores de pantalla. Sin embargo, no todos los lectores de pantalla utilizan scripts, y como consecuencia, posibilitan únicamente la interacción con las aplicaciones para las que fueron diseñados y para aquellas que contienen objetos estándar.

- **Texto**

  En todos los procesos de detección de información que se llevan a cabo en un lector de pantalla, se busca que ésta sea transmitida al usuario mediante una voz sintetizada. El componente que realiza este último proceso (el sintetizador de voz) requiere que dicha información le sea proporcionada como una cadena de texto.

  Cuando el formato de la información recuperada sobre los objetos, corresponde al de una cadena de texto, como en el caso del título de una ventana, o el contenido de un ítem de menú, dicha cadena le es proporcionada directamente al sintetizador de voz para que éste transmita la información por los altavoces de la computadora.

  Toda información detectada que no corresponda a texto, debe ser traducida a este formato. Esto ocurre por ejemplo, al realizar la detección de las teclas presionadas, y del estado de los objetos, donde la información recuperada corresponde a códigos numéricos a los que debe asociarse una cadena de texto según el significado de dichos códigos.

- **Sintetizador de Voz**

  Componente que recibe la información detectada en formato de cadena de texto, y que realiza posteriormente el proceso de síntesis de voz para transmitirla al usuario a través de los altavoces de la computadora.

  La síntesis de voz, proceso de generar voz a partir de texto, es una emulación del proceso de habla generado por el ser humano a través de las cuerdas vocales.

  Anteriormente se empleaban circuitos integrados especializados para llevar a cabo el proceso de síntesis. Tales circuitos eran incluidos en aparatos externos que se comunican vía serie con la computadora, y que disponen de un altavoz para escuchar la voz generada. También se incluían en tarjetas para ser adaptadas internamente a la computadora. En ambos casos, el

sistema de síntesis de voz implicaba adicionar hardware a la computadora.

Hoy en día las computadoras cuentan con su propio sistema de sonido, por lo que no necesitan incorporar hardware adicional, y la síntesis de voz es realizada vía software a través de los motores TTS (Text to Speech) o sistemas de conversión de texto a voz, cuya arquitectura se presenta en la figura 4.4.

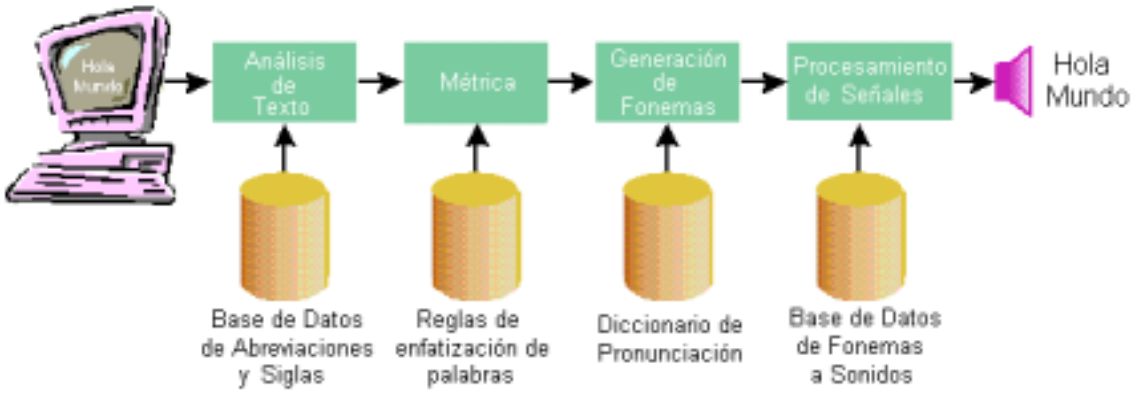

Figura 4.4 Arquitectura de un Motor TTS

El proceso inicia cuando la aplicación le proporciona al motor TTS una cadena de texto (como “Hola Mundo” en la figura 4.4). El módulo analizador de texto convierte números en palabras, identifica signos de puntuación tales como puntos, comas, y signos de admiración; convierte abreviaciones a palabras; e incluso realiza la pronunciación de siglas.

Una vez que el texto es convertido a palabras, el motor analiza qué palabras deben ser enfatizadas, vocalizándolas con una mayor fuerza o por un período de tiempo más prolongado.

A continuación, el motor TTS determina la pronunciación de las palabras, ya sea buscándolas en un diccionario de pronunciación, o mediante un algoritmo de cálculo. Luego, los fonemas son analizados gramaticalmente y su pronunciación es obtenida de una base de datos de fonemas a sonidos, que describe numéricamente como deberían sonar los fonemas.

Finalmente los valores que representan a los sonidos, son tratados mediante técnicas de procesamiento de señales para ser enviados como una señal digital de audio hacia la tarjeta de sonido de la computadora, y posteriormente ser escuchados a través de los altavoces.

Métodos de Detección

El proceso más importante que se realiza en un lector de pantalla lo constituye la detección de información. Describir la forma específica en que cada uno de estos programas lleva a cabo dicha proceso constituye una tarea muy difícil, dado que ese tipo de información por lo general es reservada.

Algunas publicaciones electrónicas sobre lineamientos de accesibilidad tratan aspectos generales sobre la detección de información. Particularmente, una publicación de Adobe Systems titulada *"Designing Accesible Electronic Forms"*[1], menciona que las herramientas de accesibilidad emplean tres métodos diferentes para recolectar información sobre los objetos presentes en la pantalla: *Mensajes de Windows, Active Accesibility, y el Off-Screen Model.*

1. Mensajería de Windows

Cuando los objetos poseen *controladores de ventana* propios, las herramientas pueden ser informadas por el sistema operativo (a través de su sistema de mensajes) acerca de cuándo ocurre la creación o destrucción de dichos objetos. Además mediante los controladores, es posible acceder a la *clase de ventana* del objeto, y si se trata de una clase estándar, recuperar información sobre el objeto, como su nombre, por ejemplo.

Este método es considerado el más estándar, y constituye el empleado en el desarrollo de esta investigación.

[1] http://www.accelio.com/products/products_capture_designtips.cfm

2. Active Accesibility

Las herramientas de accesibilidad pueden utilizar Active Accesibility (un conjunto de interfases desarrolladas por Microsoft) para recuperar información directamente del objeto y ser notificado cuando el objeto cambia. Esta interfaz es soportada automáticamente por los controles estándar de Windows y también puede ser soportada por controles o clases personalizadas.

3. Off-Screen Model

Las herramientas de accesibilidad pueden monitorear las operaciones de dibujo que colocan texto en la pantalla, y crear un modelo sobre los contenidos de la misma (off-screen model). Sin embargo este método se basa en soluciones complejas e hipotéticas, y aún no proveería información acerca del significado del texto o de los gráficos. Por esta razón es considerado como el último recurso para la detección de la información presente en pantalla.

4.2 PROGRAMACIÓN BASADA EN WINDOWS (WIN32)

Generalidades

Windows es un sistema operativo que provee tres características principales: *una interfaz gráfica de usuario, capacidad multitarea, e independencia de dispositivos hardware*. La capacidad multitarea consiste en permitir al usuario, ejecutar simultáneamente muchas aplicaciones o muchas “copias” de la misma aplicación. La independencia de dispositivos hardware se refiere a que se libera al programador de tener que incluir en sus aplicaciones los controladores (drivers) para cada tipo de dispositivo de entrada o salida (tarjeta de video, teclado, mouse, e impresora, por ejemplo) con los que la aplicación interactúa, permitiendo que ésta se ejecute en una variedad de configuraciones de hardware.

La primera característica (descrita en la siguiente sección) es la más apreciable, y dado que en ella se definen los elementos de la aplicación que serán visibles en la pantalla, adquiere gran importancia para el desarrollo de un lector de pantalla basado en este sistema operativo.

Interfaz Gráfica de Usuario

El concepto clave a tratar, cuando se habla de la interfaz gráfica que Windows proporciona al usuario, lo constituye la *ventana*. Una ventana, es un área rectangular de la pantalla en la que una aplicación despliega su salida, y en la que recibe los datos de entrada del usuario. De esta forma la ventana constituye la interface entre el usuario y la aplicación.

Todas las aplicaciones de Windows que cuentan con interfaces gráficas, crean al menos una ventana, que se conoce usualmente como la *ventana principal*. Generalmente, la mayoría de aplicaciones utilizan diferentes tipos de ventanas además de su ventana principal. Entre dichas ventanas se encuentran los *controles, cuadros de diálogo, y cuadros de mensaje.*

Un control es una ventana utilizada por una aplicación para obtener información especifica del usuario, como el nombre de un archivo que requiere ser abierto, o el tipo de letra a utilizar para escribir en un documento, por ejemplo. Los controles se utilizan con mayor frecuencia dentro de los cuadros de diálogo, pero también pueden ser utilizados en otras ventanas.

En general, las aplicaciones utilizan los controles junto con otras ventanas para llevar a cabo tareas de entrada y salida.

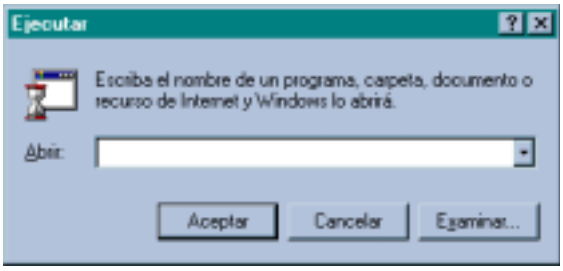

Figura 4.5 Cuadro de Diálogo "Ejecutar"

Los cuadros de diálogo son ventanas que contienen uno o más controles, y son utilizados por las aplicaciones para pedir al usuario los datos necesarios para completar alguna tarea.

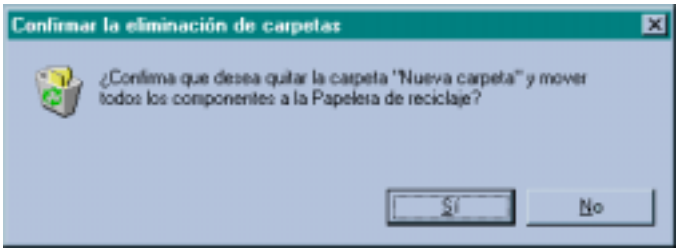

Figura 4.6 Cuadro de Mensaje "Confirmar Eliminación"

Los cuadros de mensaje son ventanas que despliegan notificaciones, advertencias o alertas para el usuario. Por lo general, la mayoría de cuadros de mensaje utilizan como parte de su interfaz visual, dos tipos específicos de controles: *controles estáticos y botones de comando*. Éstos forman parte de los tipos de controles descritos a continuación.

- **Controles Predefinidos:**

El sistema operativo Windows provee a las aplicaciones varias *clases de ventana* predefinidas mediante las que se pueden crear e incorporar diferentes tipos de controles a sus interfaces gráficas. Los controles pertenecientes a estas clases de ventana son llamados *controles predefinidos*; entre éstos se tienen los siguientes:

☑ BUTTON (BOTÓN)

Un botón es un control que el usuario puede presionar para proporcionar datos de entrada a una aplicación.

Existen diferentes tipos de botones; los más comunes son: *los botones de comando, de chequeo, y de opción*.

- **Push Button (Botón de Comando)**

Figura 4.7 Botón de Comando

Un botón de comando es un rectángulo que contiene texto definido por la aplicación (una etiqueta), o un ícono o mapa de bits, para indicar lo que el botón hace cuando el usuario lo presiona.

Típicamente, un botón de comando es utilizado para iniciar una operación.

- **Check Box (Botón de Chequeo)**

Figura 4.8 Botón de Chequeo

Un botón de chequeo posee un pequeño cuadro y una etiqueta que indica una opción que se elige al colocar un cheque sobre el cuadro.

Una aplicación puede contener botones de chequeo agrupados para permitir al usuario escoger de manera independiente distintas opciones sobre un mismo conjunto.

- **Radio Button (Botón de Opción)**

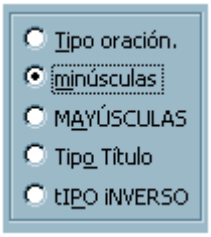

Figura 4.9 Botón de Opción

Este control contiene varias opciones indicadas en etiquetas, de entre las cuales el usuario puede elegir sólo una, al seleccionar el botón. Una aplicación usa comúnmente estos botones agrupados para permitir al usuario escoger en forma excluyente opciones sobre un mismo conjunto.

☑ COMBO BOX (Cuadro de Combos)

Figura 4.10 Cuadro de Combos

Un cuadro de combos es un control que puede contener una lista de opciones predeterminadas, de entre las cuales el usuario puede elegir. Algunos también permiten agregar a la lista opciones introducidas mediante el teclado.

☑ EDIT (Cuadro de Edición)

Figura 4.11 Cuadro de Edición

Un control edit permite introducir y editar texto desde el teclado. Se utilizan comúnmente en los cuadros de diálogo para solicitarle información específica al usuario.

☑ STATIC (Control Estático)

Mediante el control estático, una aplicación provee al usuario información en formato de texto o gráficos. Usualmente las aplicaciones utilizan estos controles para etiquetar otros controles como los cuadros de edición, y los cuadros de combo.

☑ MENU BAR (Barra Menús)

Figura 4.12 Barra de Menús

Es un control formado por menús que contienen opciones. A su vez, estas opciones (conocidas como ítems de menú) pueden convertirse en submenús al contener otras opciones dentro de ellas.

Algunas opciones realizan funciones específicas al ser elegidas, mientras que otras activan y desactivan características de las aplicaciones.

☑ LIST BOX (Cuadro de Lista)

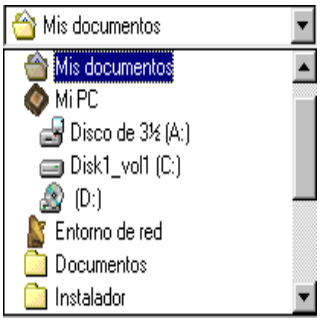

Figura 4.13 Cuadro de Lista

Contiene una lista de ítems de entre los cuales el usuario puede elegir. Los ítems pueden representarse por cadenas de texto con o sin íconos.

Usualmente se realizan operaciones sobre el ítem elegido, a través de otros controles.

- **Controles Comunes (Common Control)**

  Los controles comunes son un conjunto de ventanas implementadas por la *librería de controles comunes*, la cual es una *librería de enlace dinámico* incluida en el sistema operativo Windows. Como otras ventanas que representan controles, un control común es una *ventana hija* utilizada por una aplicación en conjunto con otras ventanas para desempeñar tareas de entrada / salida.

  Los siguientes tipos de controles forman parte de los controles comunes utilizados por el sistema operativo:

☑ List View (Vista de Lista)

Figura 4.14 Vista de Lista

Contiene una colección de ítems, que consisten en un ícono con etiqueta. En ocasiones, los ítems pueden representan elementos del sistema operativo como archivos, carpetas y unidades de disco.

Estos controles poseen diferentes formas para arreglar sus ítems y para desplegarlos en forma individual.

☑ ToolBar (Barra de Herramientas)

Figura 4.15 Barra de Herramientas

Es un control que contiene uno o más botones con imágenes, texto, o ambos. Comúnmente, los botones en las barras de herramientas corresponden a ítems que se encuentran en el menú de la aplicación, de esta manera se provee una forma adicional y más directa de acceder a los comandos de una aplicación.

☑ Tree View (Vista de Árbol)

Figura 4.16 Vista de Árbol

Es una ventana que despliega una lista jerárquica de ítems, tales como los archivos y directorios en un disco.

Cada ítem contiene un ícono con etiqueta, y puede tener una lista de subítems asociados. Al elegir el ítem, el usuario puede expandir o contraer la lista de subítems que se encuentra dentro del ítem.

☑ Tab Control (Fichero)

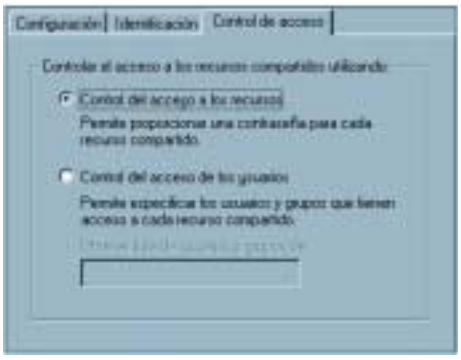

Figura 4.17 Fichero

Un Tab Control es análogo a los separadores de una libreta. Con este control, una aplicación puede definir diferentes fichas utilizando la misma área de la ventana. Cada ficha reúne cierto tipo de información o un grupo de controles que se muestran cuando el usuario selecciona la ficha.

Todos los controles mencionados anteriormente, al igual que todas las ventanas en Windows, pertenecen a una *clase de ventana*, la cual contiene una serie de atributos que el sistema operativo utiliza como plantilla para crear la ventana. Además, toda clase de ventana posee un *procedimiento de ventana* compartido por todas las ventanas pertenecientes a la misma clase. Junto con la clase de ventana, el procedimiento de ventana (descrito en la siguiente sección) define la mayoría de características relacionadas a la apariencia y comportamiento de una ventana.

El sistema operativo Windows registra algunas clases (conocidas como clases del sistema) durante su carga inicial, para que puedan ser utilizadas por todas las aplicaciones. La tabla 4.2 muestra algunas clases del sistema dentro de las que se encuentran las correspondientes a los controles predefinidos y a los cuadros de diálogo.

La mayoría de clases de los controles comunes están definidas dentro de la librería de controles comunes **Comctl32.dll**. Tanto para los controles predefinidos, como para los comunes, las aplicaciones utilizan funciones definidas por el sistema operativo para crearlos y manipularlos.

Tabla 4.2 Clases del Sistema

Nombre	Descripción
Button	La clase para un botón
ComboBox	La clase para un combo box
Edit	La clase para un cuadro de edición
ListBox	La clase para un cuadro de lista
ScrollBar	La clase para una barra de desplazamiento
Static	La clase para un control estático
#32768	La clase para un menú
#32769	La clase para la ventana del escritorio
#32770	La clase para un cuadro de diálogo

Introducción a la Programación Basada en Windows

Las distintas ventanas que Windows presenta en la pantalla de la computadora son administradas por dicho sistema operativo, asignando a cada una un valor numérico único conocido como *controlador de ventana* (window handle ó hwnd) que utiliza para identificarla.

El sistema verifica continuamente si han ocurrido *eventos* en las ventanas. Éstos pueden producirse mediante acciones del usuario, como hacer clic con el mouse o presionar una tecla, mediante programación, como modificar el estado de un control a través de código, o incluso como resultado de las acciones de otras ventanas.

Cada vez que se produce un evento, se envía un *mensaje* al sistema operativo. Éste procesa el mensaje y lo transmite a la ventana o ventanas a las que afecta dicho mensaje para que puedan realizar la acción apropiada, basándose en sus propias instrucciones para tratar ese mensaje en particular.

- **Mensajes**

Cuando el usuario trabaja con una aplicación basada en Windows, interactúa con ella a través del teclado o del ratón, seleccionando alguna opción de menú, arrastrando su ventana, o redimensionando su tamaño, por ejemplo. La ventana recibe todas estas entradas o eventos generados por el usuario en forma de mensajes.

Desde el punto de vista de la aplicación, el mensaje es visto como una notificación acerca de la ocurrencia de algún evento, el cual puede o no requerir una acción determinada. El efecto neto de este sistema de comunicación, es que las aplicaciones están orientadas hacia el procesamiento de mensajes.

Todos los mensajes son dirigidos hacia una ventana, e independientemente de su tipo, poseen los siguientes parámetros: un *controlador de ventana,* el *tipo de mensaje,* y dos valores que contienen información adicional del mensaje e identificados comúnmente con los nombres *wparam*, y *lparam.* En las versiones de 32 bits de Windows (Win32), estos cuatro parámetros son valores numéricos con un tamaño predefinido de 32 bits.

El primer parámetro que se especifica en un mensaje es el *controlador de la ventana* a la cual se dirige el mensaje. En un ambiente de programación orientada a objetos, un controlador es simplemente el identificador de un objeto, y en este caso particular, se trata del identificador de la ventana hacia la que se dirige el mensaje.

Con frecuencia, los controladores hacen referencia a un objeto definido en forma de estructura que está ubicado en una región de memoria que puede ser reubicada. Sin embargo, es importante aclarar que aunque dicha región pueda ser trasladada hacia otra ubicación en algún momento dado, el controlador permanece inalterado. De esta forma, Windows maneja la memoria sin que estos cambios sean visibles para la aplicación.

El segundo parámetro especificado en un mensaje es *el tipo de mensaje*. Cada tipo de mensaje tiene un valor numérico constante de 32 bits, definido en formato hexadecimal. Usualmente, es nombrado con letras utilizadas como mnemónicos seguidas por un carácter de subrayado, y finalmente una palabra utilizada como descriptor. El tipo más común de mensajes que una aplicación debe procesar son los mensajes de ventana. Algunos ejemplos de estos mensajes son: WM_CREATE, y WM_MENUSELECT, para los cuales WM indica Window Message (mensaje de ventana). Los controles poseen también sus propios tipos de mensaje, y cuando sea necesario, una aplicación puede crear y registrar sus propios tipos de mensajes, los cuales se conocen como mensajes privados.

El propósito de los últimos dos parámetros, es el de proveer información adicional que es necesaria para interpretar el mensaje, y dado que se emplean distintos tipos de mensajes, sus valores varían de acuerdo a dichos tipos. Como ejemplo, cuando el ratón se mueve por encima de una ventana, ésta recibe el mensaje WM_MOUSEMOVE (cuyo valor hexadecimal es 11F). En este mensaje, el valor de wparam indica si se han presionado las teclas CONTROL, SHIFT, o los botones del ratón, mientras que el valor de lparam indica las coordenadas del cursor.

- **Procedimientos de Ventana**

Todas las ventanas poseen una función conocida como *procedimiento de ventana,* la cual se encarga de procesar todos los mensajes que llegan a la misma. El comportamiento de las ventanas dependerá de las respuestas que este procedimiento genere ante los mensajes.

Como se sabe, toda ventana pertenece a una clase de ventana en particular. La clase de ventana determina el procedimiento de ventana que una ventana individual utilizará para procesar sus mensajes. Todas las ventanas pertenecientes a la misma clase, utilizan el mismo procedimiento de ventana.

El procedimiento de ventana recibe como argumentos, los cuatro parámetros asociados con todo mensaje: el *controlador de ventana,* el *tipo de mensaje,* y los parámetros de información adicional *wparam*, y *lparam.*, y devuelve un valor de 32 bits cuya interpretación depende del tipo de mensaje. Un procedimiento de ventana típico presenta una estructura como la siguiente:

< Nombre del Procedimiento de Ventana > (Controlador, Mensaje, wParam, lParam)
Inicio de la función
Dependiendo de **Mensaje**
En caso de ser **WM_CREATE**
Inicializar la ventan y retornar 0
En caso de ser **WM_PAINT**
Actualizar el área de dibujo de la ventana y retornar 0
En caso de ser **WM_SIZE**
Asignar posición y tamaño de la ventana y retornar 0
En caso de ser **WM_DESTROY**
Terminar la aplicación y retornar 0
En caso de ser **<Otros Mensajes>**
Procesar este mensaje
En caso de no ser alguno de los anteriores
Devolver parámetros de mensaje al procedimiento
de ventana por defecto y retornar respuesta
Fin de la función

El procedimiento de ventana utiliza el argumento “Mensaje” dentro de una sentencia de selección de casos para manejar diferentes mensajes. Para cada caso, el procesamiento del mensaje retorna un valor (que depende del tipo de mensaje) para indicar que el mensaje ha sido procesado.

Por lo general, cuando se reciben mensajes que no están incluidos en los casos posibles, suelen enviarse a un procedimiento de ventana conocido como *procedimiento de ventana por defecto*, para asegurar el procesamiento de todos los mensajes que llegan a una ventana. Esto se realiza a través de la llamada a la función **DefWindowProc,** la cual acepta como argumentos los cuatros parámetros asociados con el mensaje que llega al procedimiento de ventana.

- **Subclasificación**

  Cuando una aplicación crea una ventana, el sistema operativo utiliza un bloque de memoria para almacenar información específica de la ventana. Entre dicha información se encuentra la dirección del procedimiento de ventana que procesa los mensajes que llegan a ella. Cuando el sistema necesita enviar un mensaje a la ventana, busca la dirección del procedimiento de ventana asociado a la misma, y pasa el mensaje a dicho procedimiento para que sea procesado.

  A través de la técnica conocida como subclasificación (subclassing, en inglés), la aplicación puede interceptar los mensajes que llegan a las ventanas que ha creado, antes de que ésta pueda procesarlos. Utilizando esta técnica, una aplicación puede extender, modificar, o monitorear el comportamiento de una ventana particular.

  La aplicación subclasifica a una ventana reemplazando la dirección del procedimiento de ventana original de dicha ventana, con la dirección de un nuevo procedimiento de ventana, conocido como *procedimiento de subclasificación.* Posteriormente, este procedimiento recibe los mensajes enviados hacia la ventana, los cuales puede modificar para luego devolverlos al procedimiento de ventana original, o procesarlos y no devolverlos al procedimiento original.

- **Hooks**

  Un enlace o hook, es un punto o ubicación dentro del mecanismo de manejo de mensajes del sistema operativo en donde una aplicación puede instalar una subrutina para monitorear el tráfico de mensajes en el sistema, y procesar cierto tipo de mensajes antes que lleguen a su procedimiento de ventana destino.

El sistema operativo Windows, soporta diferentes tipos de hooks; cada uno proporciona acceso a diferentes aspectos del mecanismo de manejo de mensajes.

Para poder capturar la información que transportan los mensajes que se desea monitorear, la instalación de un hook va acompañada con la especificación de la dirección de un procedimiento conocido como *procedimiento de hook*, el cual es utilizado para hacer llegar dicha información.

Entre los distintos tipos de hooks disponibles se encuentran, WH_KEYBOARD HOOK, el cual permite monitorear el tráfico de mensajes asociados al evento de pulsar teclas, WH_MOUSE, para monitorear mensajes asociados a eventos generados con el mouse, WH_CALLWNDPROC, para el monitoreo de mensajes enviados al procedimiento de ventana a través de la función **SendMessage**, y WH_GETMESSAGE, el cual permite monitorear mensajes que están apunto de ser retornados por la función **GetMessage**.

4.3 INTRODUCCIÓN A LA API DE WINDOWS

¿Qué es una API?

La Interfaz de Programación de Aplicaciones (API), sirve como una interfaz de software que las aplicaciones utilizan para solicitar al sistema operativo de la computadora la realización de distintas tareas de nivel inferior.

Una API puede ser considerada como el nivel fundamental de la programación de alto nivel. Frecuentemente, en la programación de alto nivel, un programa no ejecuta tareas por sí mismo, en lugar de ello, transfiere dichas tareas a otros programas. En el caso de sistemas operativos, los programas frecuentemente delegan varias tareas al sistema operativo. Por ejemplo, si un programa desea escribir datos en un disco, no envía directamente al disco los

comandos para buscar, leer, y escribir. En su lugar, simplemente le indica al sistema operativo que escriba una cierta cantidad de datos en un archivo específico, y el sistema operativo maneja todo el trabajo de bajo nivel. De esta forma, los programadores no tienen que preocuparse por conocer las tareas básicas y fundamentos involucrados en todos los programas (como el acceso a discos, administración de memoria, y operaciones de dibujo), ahorrando tiempo en el diseño de sus programas y consumiendo menos espacio.

Interfaz de Programación de Aplicaciones de Windows

Como se mencionó anteriormente, una API maneja todo lo relacionado al control de la pantalla, acceso a discos y memoria, entre otos. El sistema operativo Windows provee a los desarrolladores de aplicaciones funciones para realizar estas tareas, a través de su interfaz de programación de aplicaciones. Sin embargo, también provee la mayoría de las características comunes a todo el software basado en Windows. Por ejemplo, los cuadros de diálogo comunes (Abrir, Guardar como, Elegir fuente, etc.), configuraciones del sistema operativo, e incluso las propias ventanas son proporcionados por la API de Windows.

Los programas basados en Windows utilizan extensamente la API de Windows en la mayoría de tareas. Aunque al programar no se utilice explícitamente la API, el lenguaje de programación casi siempre incorporará llamadas a la API de Windows en el programa ejecutable que genere, para realizar distintas tareas.

Ubicación de la API de Windows

Casi todas las funciones de la API de Windows forman parte de *librerías de enlace dinámico* ubicadas en el directorio del sistema (C:\Windows\System), como *user32.dll, kernell32.dll,* y *gdi32.dll*.

En general, las librerías de enlace dinámico (en inglés, Dynamic Link Libraries o DLL's) son archivos de extensión **.dll** que contienen procedimientos que se cargan y vinculan a una aplicación en tiempo de ejecución, en lugar de hacerlo en tiempo de compilación. Esto implica que las bibliotecas se pueden actualizar independientemente de la aplicación.

Los archivos DLL de la API de Windows exportan las funciones que contienen, lo cual significa que permiten a los programas externos utilizarlas. De esta forma, cualquier programa basado en Windows puede acceder a las distintas funciones de la API. La parte principal de las funciones API se encuentran en los archivo user32.dll (funciones de interfaz de usuario), kernel32.dll (funciones del kernel del sistema operativo), gdi32.dll (funciones de interfaz para dispositivos gráficos), y shell32.dll (funciones del shell de Windows).

Componentes de la API de Windows

A pesar de haber hecho referencia sólo a las funciones de la API, éstas son sólo una parte de lo que constituye completamente la API de Windows. El conjunto completo lo componen *funciones, estructuras, constantes nombradas, funciones callback, y mensajes.*

- **Funciones:**

  Son la parte fundamental de la API de Windows. Constituyen el código que cumple con la variedad de tareas realizadas por la API. Las funciones están almacenadas en archivos DLL y pueden ser accesadas desde cualquier programa basado en Windows.

- **Estructuras:**

  Son combinaciones de variables de distintos tipos, frecuentemente utilizadas en la API de Windows para almacenar información relacionada a un objeto.

Muchas funciones de la API requieren como argumento una estructura, con el objetivo de transferir una cantidad mayor de información sin utilizar un gran número de parámetros.

- **Constantes Nombradas:**

  La API de Windows utiliza con frecuencia nombres que hacen referencia al significado de valores numéricos constantes. Los nombres de las constantes proveen una manera más conveniente para referirse a estos valores en el código.

- **Funciones Callback:**

  Una función callback es una función definida en el programa, que es llamada por una función de rutina de la API cuando ésta ejecuta su tarea. Algunas funciones de la API requieren definir este tipo de funciones para complementar alguna tarea específica.

- **Mensajes:**

  En cierta forma los mensajes son un tipo especial de constantes nombradas. A pesar de ser también valores numéricos con nombres asociados, se comportan de manera distinta. Los mensajes son enviados a los objetos (especialmente ventanas) para que lleven a cabo alguna acción.

4.4 OBJETOS ESPECÍFICOS

Como se mencionó anteriormente, el acceso a objetos específicos se realiza a través de los medios definidos por la aplicación a la que pertenecen, a diferencia de objetos como los controles predefinidos y comunes, definidos por el sistema operativo.

Particularmente, las aplicaciones de Microsoft Office dan a conocer su *funcionalidad* como un conjunto de objetos programables. Estas aplicaciones integran el *Editor de Visual Basic* (ubicado en el ítem *Macro* del menú *Herramientas*) como herramienta para automatizar tareas dentro las mismas utilizando las *propiedades* y *métodos* de estos objetos. De esta forma es posible trabajar dentro de Microsoft Word o Microsoft Excel con los datos de sus propias aplicaciones.

Objetos y Modelos de Objeto

Un objeto puede ser una parte de una aplicación, como un control o una ventana, o también puede ser la aplicación entera. En términos de A nivel de programación, un objeto puede ser visto como una combinación de código y datos que pueden ser tratados como una unidad.

La funcionalidad de una aplicación se refiere a todas las formas con las que se puede trabajar sobre su contenido. Este último se refiere a los elementos que la aplicación contiene, tales como los documentos y las palabras, números y gráficos incluidos en los mismos. Comúnmente se trabaja con el contenido de una aplicación abriendo, cerrando, copiando, o modificando sus elementos.

El contenido y la funcionalidad de la aplicación se dividen en unidades discretas, conformando distintos objetos que guardan relaciones entre sí. Por lo general, los objetos más familiares para los usuarios son los elementos que forman parte de la interfaz gráfica de la aplicación, como los libros, hojas y celdas en Microsoft Excel, o los documentos y secciones en Microsoft Word.

La manera en que se ordenan los objetos que componen una aplicación, junto con la forma en que se divide su contenido y funcionalidad sobre los mismos, se conoce como el *modelo de objeto o jerarquía de objeto* de la aplicación. Normalmente, un modelo de objeto expresa el hecho de que unos objetos son "más grandes" o “más importantes” que otros, de tal forma que contienen o se componen de otros objetos.

Usualmente, el objeto que se encuentra en la cima de la jerarquía es el objeto *Application* (Aplicación) el cual representa a la aplicación misma. Por ejemplo, el propio Microsoft Excel es el objeto Application en el modelo de objeto de Microsoft Excel. El objeto Application contiene otros objetos a los que se puede acceder sólo cuando existe el objeto Application (es decir, cuando la aplicación se está ejecutando). Cuando la existencia de un objeto depende de la existencia de otro, se dice que el primero es “hijo” del segundo, y en forma inversa, el segundo es “padre” del primero. Muchos objetos hijos tienen también sus propios objetos hijos, y al igual que un objeto padre puede tener muchos objetos hijos, un objeto hijo puede tener muchos objetos padre.

Además de contener otros objetos, cada objeto en la jerarquía posee parte del contenido y funcionalidad de la aplicación, y se aplican tanto al propio objeto como a todos los objetos por debajo de la jerarquía. Entre más alta la posición en la jerarquía, más amplio es el alcance del contenido y la funcionalidad. Por ejemplo, en Microsoft Excel, el objeto Application posee el tamaño de la ventana de la aplicación, y la capacidad de cerrar la aplicación; el objeto *Workbook* (Libro), contiene el nombre de archivo y el formato del libro, y la capacidad de almacenarlo; y el objeto *Worksheet* (Hoja) posee el nombre de la hoja y la capacidad de borrarla.

Algunos objetos se agrupan dentro de lo que se conoce como *colección de objetos.* Los objetos pertenecientes a una colección permiten ser tratados como un grupo único, y no sólo como entidades individuales. Por ejemplo, en Microsoft Word existe una colección llamada *Documents* (Documentos) que permite trabajar con todos los objetos *Document* (Documento) como un solo grupo. Por lo general, una colección de objetos se identifica con el nombre del objeto en plural, por ejemplo, para el objeto Document existe la colección Documents.

En resumen, el contenido y la funcionalidad de una aplicación se divide entre los objetos que componen su modelo de objeto. En conjunto, los objetos de la jerarquía poseen todo el contenido y funcionalidad de la aplicación, mientras e individualmente, proveen acceso a áreas específicas de estos dos elementos.

- **Objetos de Microsoft Excel**

Entre los objetos más comunes que forman parte de la interfaz visual de la aplicación Microsoft Excel se encuentran: *Application, Workbook, Worksheet, y Range (contenido en Worksheet).*

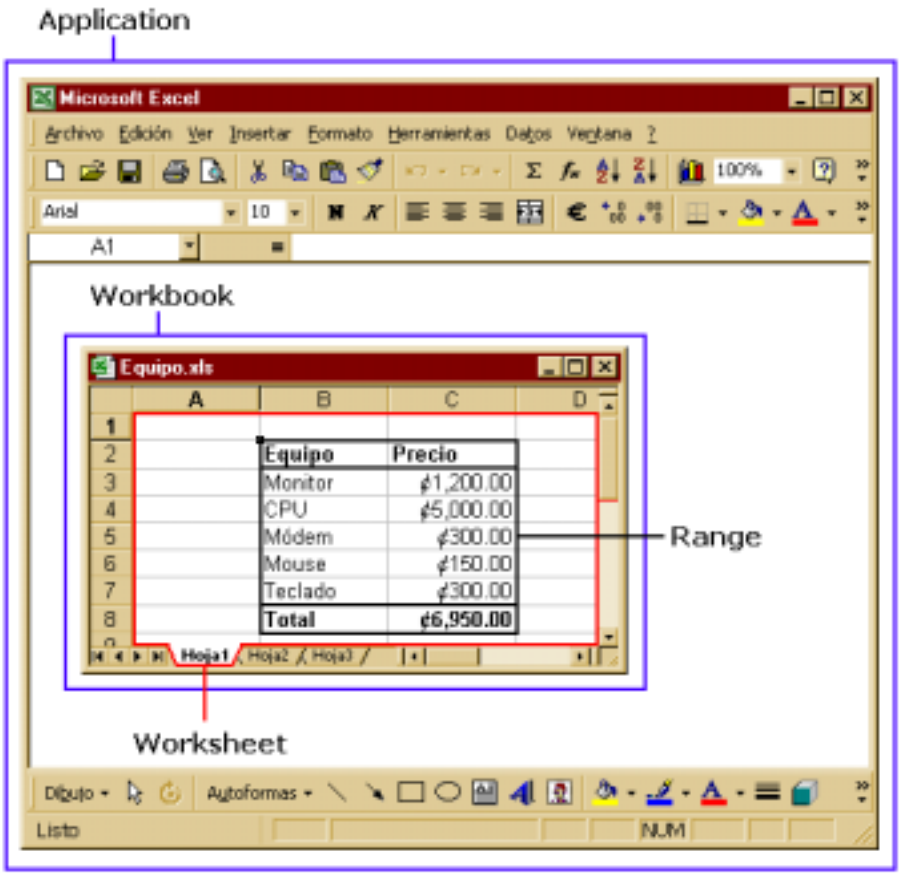

Figura 4.18 Aplicación Microsoft Excel 2000

El objeto Application representa la aplicación completa de Microsoft Excel, mientras que los objetos Workbook, Worksheet, y Range, representan respectivamente, un libro, una hoja de cálculo, y una celda, fila, columna, o selección de celdas. Como se muestra en la figura 4.19, todos forman parte del modelo de objetos de esta aplicación.

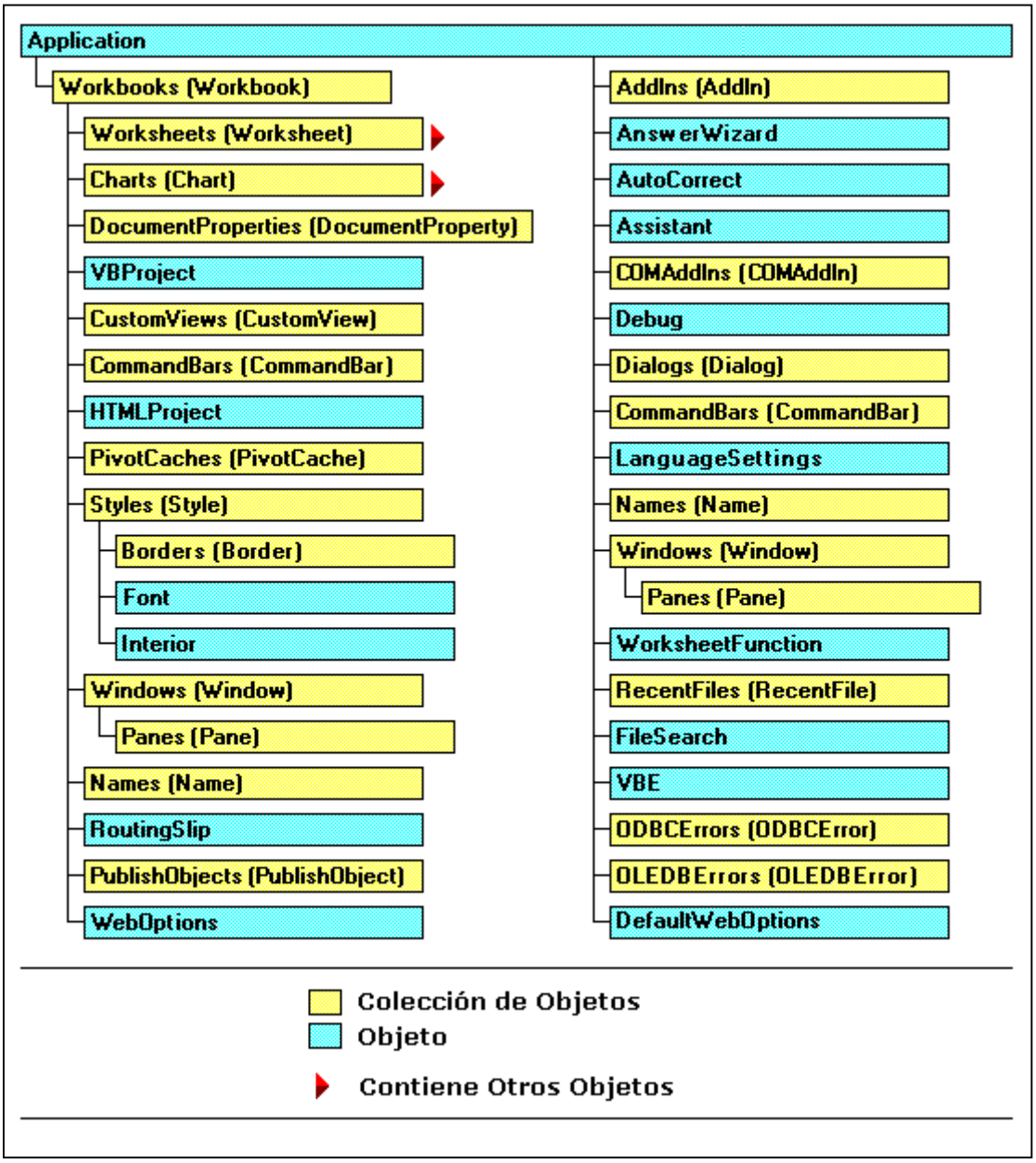

Figura 4.19 Modelo de objeto de Microsoft Excel

- **Objetos de Microsoft Word**

Al igual que la aplicación Microsoft Excel, Word posee su propio modelo de objeto. *Document, CommandBars, y Selection,* constituyen ejemplos de los objetos más comunes que forman parte de su interfaz visual.

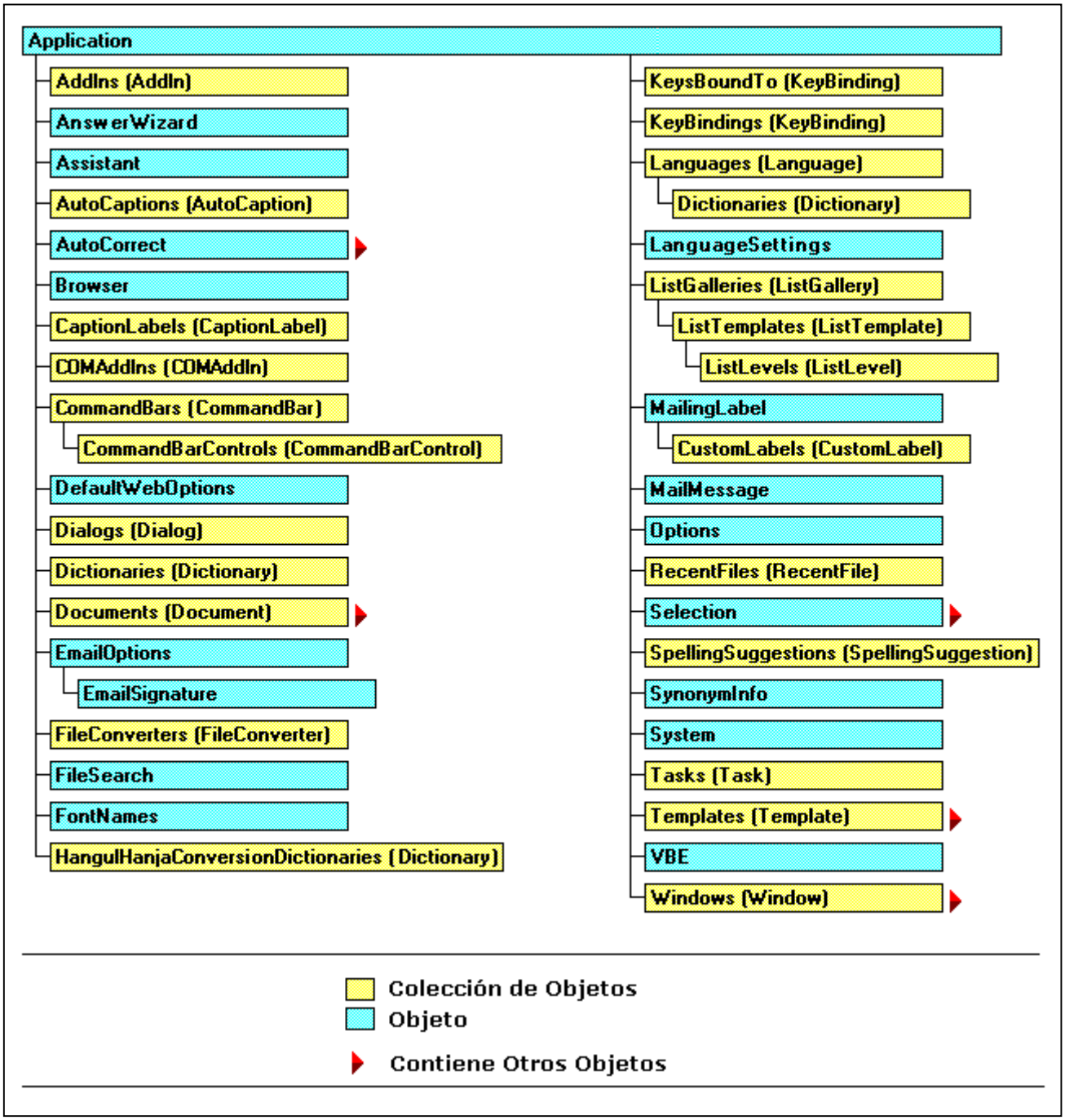

Figura 4.20 Modelo de objeto de Microsoft Word

El objeto Application representa la aplicación completa de Microsoft Word, mientras que los objetos Document, Selection, y CommandBar, representan respectivamente, un documento, un área del documento, y una *Barra de Comandos*.

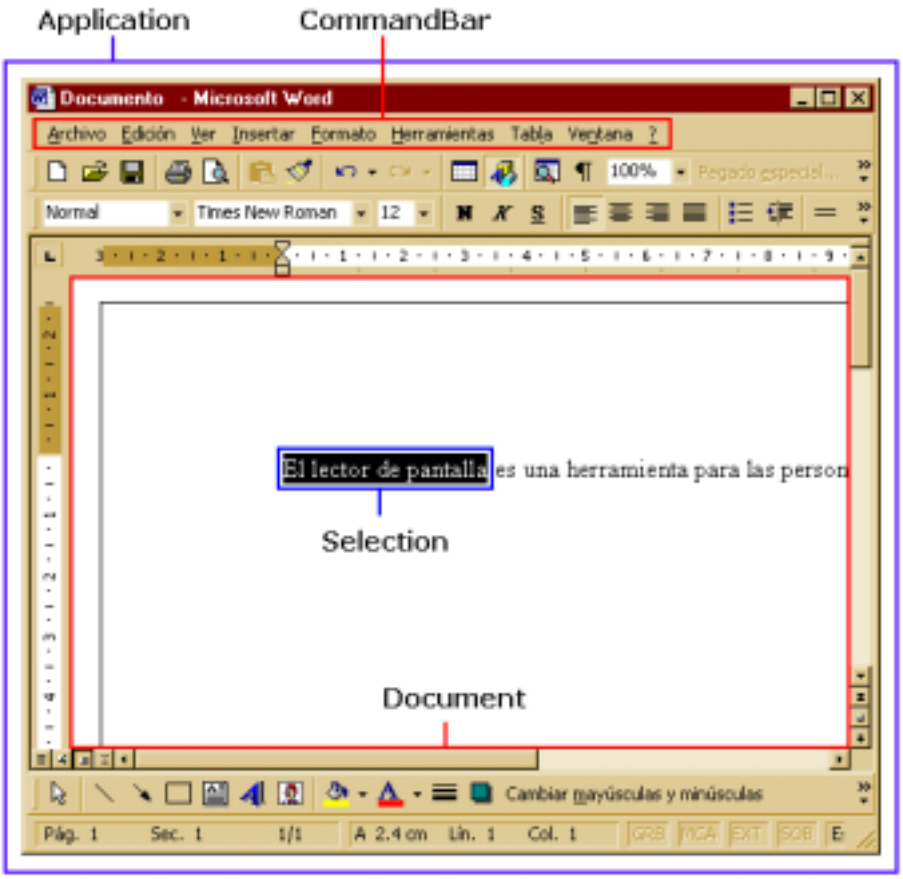

Figura 4.21 Aplicación Microsoft Word 2000

Las barras de comandos están presentes en todas las aplicaciones de Microsoft Office, y junto con otros objetos, conforman el modelo de objeto de Microsoft Office.

Existen tres tipos de barras de comandos:

- **Popup:**
  Equivale a un elemento de menú en una barra de menús.
- **ComboBox:**
  Similar a un control ComboBox; es un botón de barra de herramientas con una flecha adjunta. Al hacer clic en la flecha, aparecen más comandos de menús con sus íconos.
- **Button:**
  Equivale a un botón de barra de herramientas estándar; es un botón sobre el que aparece un ícono.

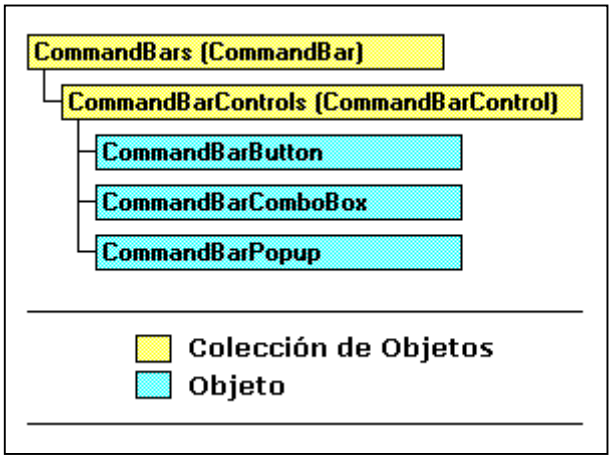

Figura 4.22. Objetos de las Barras de Comando
(Parte del modelo de objetos de Microsoft Office)

El objeto CommandBar, miembro de la colección CommandBars, combina el concepto de menús y barras de herramientas en un único objeto visual y de programación. Al ser barras de comandos, los menús contienen comandos de menú que pueden tener iconos y títulos, y pueden presentar un comportamiento similar al de los botones, conservando a la vez el formato de menú habitual. Un objeto de barra de comandos puede contener otros objetos de barra de comandos, dependiendo de su tipo.

Propiedades, Métodos, y Eventos

Los objetos poseen *propiedades* y *métodos* para acceder a su contenido y funcionalidad. Generalmente las propiedades se utilizan para obtener o establecer el contenido, el cual puede incluir el texto que posee el objeto o el conjunto de atributos de dicho objeto, mientras que los métodos se utilizan para obtener la funcionalidad, la cual incorpora todas las acciones que se pueden llevar a cabo con el contenido.

Los eventos son acciones (tales como abrir un documento, o cerrar la aplicación) reconocidas por los objetos, y para las cuales el programador escribe código que se ejecutará como respuesta a las mismas. Los eventos ocurren tanto por acciones del usuario como por código en el programa, o pueden ser disparados por el sistema operativo.

Antes de utilizar las propiedades, métodos y eventos de un objeto en la aplicación, es necesario recuperar una *referencia* a dicho objeto. La referencia representa el acceso al objeto, y es posible recuperarla directamente, o desplazándose sobre la jerarquía de los objetos hasta alcanzar el objeto deseado. Una forma muy común de tener acceso al modelo de objeto de una aplicación, es a través del objeto más alto en la jerarquía, que usualmente es el objeto Application. Por ejemplo, para iniciar el acceso al objeto Application dentro de Microsoft Word se utiliza la expresión:

```
Application
```

Application es una propiedad que se utiliza para devolver una referencia a un objeto Application. Si luego se quiere descender en la jerarquía y llegar a la colección Documents, se utiliza:

```
Application.Documents
```

Documents es una propiedad que se utiliza para devolver una referencia a la colección Documents. En el caso de colecciones de objetos como Documents, se pueden crear nuevos objetos y añadirlos a la colección utilizando el método **Add**, disponible para esa colección. Si se desea utilizar dicho método para crear un nuevo documento de Word por ejemplo, se debe recuperar primero la referencia a la colección, y luego utilizar el método. La siguiente expresión lleva a cabo ambos procesos:

```
Application.Documents.Add
```

Así mismo, para crear un nuevo libro en Microsoft Excel:

```
Application.Workbooks.Add
```

En las expresiones anteriores se trabaja con un método que se aplica a todos los objetos Document y Workbook como conjunto, dado que se utilizan sus colecciones respectivas. Para hacer referencia a los objetos individuales en esas colecciones, se utiliza el método **Item**. Por ejemplo, para cerrar un libro o documento específico (Libro1.xls o Documento1.doc) previamente abierto, los objetos Document y Worbook disponen del método **Close**, pero primero son necesarias las respectivas referencias.

Para obtener una referencia al libro “Libro1”, se utiliza:

```
Application.Workbooks.Item(“Libro1.xls”)
```

Para obtener una referencia al documento “Documento1”, se utiliza:

```
Application.Documents.Item(“Documento1.doc”)
```

La primera expresión hace referencia a un libro llamado “Libro1”, y la segunda a un documento llamado “Documento1”. La acción completa se realiza llamando posteriormente al método Close.

Para cerrar el libro “Libro1”, se utiliza:

```
Application.Workbooks.Item(“Libro1.xls”).Close
```

Para obtener una referencia al documento “Documento1”, se utiliza:

```
Application.Documents.Item(“Documento1.doc”).Close
```

En las tablas 4.3, 4.4, y 4.5, se presentan ejemplos de otras propiedades, métodos, y eventos de algunos objetos de Microsoft Word, y Microsoft Excel.

Tabla 4.3 Propiedades, Métodos, y Eventos para Objetos de Microsoft Word

Microsoft Word		
Objeto	**Propiedad / Método / Evento**	**Función**
Application	**ActiveDocument**	Devuelve un objeto Document que representa el documento activo.
	Quit	Se utiliza para salir de Word.
	WindowSelectionChange	Se produce cuando cambia la selección en la ventana de documento activo.
	Quit	Se produce cuando el usuario sale de Word.
Document	**Characters**	Devuelve una colección Characters que representa a los caracteres de un documento, intervalo o selección.
	Content	Devuelve un objeto Range que representa el texto del documento principal.
	Close	Cierra el documento especificado.
Selection	**Text**	Devuelve o establece el texto de la selección o del intervalo especificado.
	Information	Devuelve información acerca de la selección o del intervalo especificado.

Tabla 4.4 Propiedades, Métodos, y Eventos para Objetos de Microsoft Excel

Microsoft Excel		
Objeto	**Propiedad / Método / Evento**	**Función**
Application	**ActiveCell**	Devuelve un objeto Range que representa la celda activa de la ventana activa o especificada.
	SheetActivate	Ocurre cuando se activa una hoja.
	SheetSelectionChange	Se produce cuando la selección cambia en una hoja (no se produce si la selección está en una hoja de gráfico).
	WorkbookBeforeClose	Se produce antes de que se cierre cualquier libro.
	WorkbookDeactivate	Se produce antes de desactivar cualquier libro abierto.
	WorkbookOpen	Ocurre al abrir un libro.
Range	**Adress**	Devuelve el nombre de las celdas que conforman el rango.
	Text	Devuelve o establece el texto del objeto especificado.
Workbook	**ActiveSheet**	Devuelve un objeto que representa la hoja activa.
	Close	Cierra el libro especificado.
Worksheet	**Name**	Devuelve o establece el nombre del objeto.

Tabla 4.5 Propiedades para Objetos de Microsoft Office

Microsoft Word y Microsoft Excel		
Objeto	**Propiedades**	**Función**
Application	**CommanBars**	Devuelve una colección CommandBars que representa la barra de menús y todas las barras de herramientas de la aplicación.
CommandBar	**AccState**	Devuelve el estado del objeto.
	Controls	Devuelve un objeto CommandBarControls que representa todos los controles de la barra de comandos o del control emergente.
CommandBarControl	**Type**	Devuelve el tipo de control de barra de comandos.

Automatización

La automatización (conocida formalmente como Automatización OLE) es una interfaz de comunicación que le permite a una aplicación manipular objetos pertenecientes a otra aplicación, o exponer sus propios objetos para que puedan ser manipulados.

Mediante la automatización, se proporcionan referencias a las propiedades, métodos y eventos de un objeto, para que éste pueda ser accedido desde otras aplicaciones. Los objetos de una aplicación a los que se puede hacer referencia desde otra, son conocidos como *objetos de automatización*.

- **Automatización con Microsoft Word 2000**

Para que los objetos de la aplicación Microsoft Word estén disponibles mediante automatización en una aplicación creada en Visual Basic 6.0, el primer paso a realizar es la creación de una referencia a la *biblioteca de objetos* de dicha aplicación. Éstas están disponibles en el ítem Referencias del menú Proyecto, de Visual Basic.

Una biblioteca de objetos es un archivo con una extensión de archivo **.OLB** que proporciona a los controladores de Automatización (como Visual Basic) información sobre los objetos de Automatización que estarán disponibles. Para

Microsoft Word 2000 se escoge la referencia **Microsoft Word 9.0 Object Library**, definida en el archivo MSWORD9.OLB.

Posteriormente se debe crear una variable de objeto que hará referencia al objeto Application de Word, como se indica en el siguiente ejemplo:

```
Dim AppWord as Word.Application
```

Utilizando la funciones CreateObject o GetObject, proporcionadas por Visual Basic, es posible crear o recuperar una referencia al objeto de automatización; por ejemplo:

```
Set AppWord = CreateObject(“Word.Application”)
AppWord.Visible = True
```

La función Createobject devuelve un objeto Application de Word, y lo asigna a la variable AppWord. A través de esta variable es posible utilizar los objetos, propiedades, métodos y eventos que permiten controlar la aplicación.

Posteriormente, la aplicación puede hacerse visible estableciendo la propiedad **Visible** como Verdadera, y luego añadir un nuevo documento mediante la propiedad Add de la colección Documents, siendo el código completo:

```
Dim AppWord as Word.Application
Set AppWord = CreateObject(“Word.Application”)
AppWord.Visible = True
AppWord.Documents.Add
```

Para cerrar la aplicación puede utilizarse el método Quit (`AppWord.Quit`). Cuando el objeto de automatización ya no se utilice más, es recomendable borrar las variables que hagan referencia al objeto con el fin de liberar el objeto de la memoria. Para borrar una variable de objeto, se establece a Nothing, es decir:

```
AppWord = Nothing
```

- **Automatización con Microsoft Excel 2000**

Como Word, Microsoft Excel posee su propia biblioteca de objetos, EXCEL9.OLB, la cual se escoge mediante la referencia **Microsoft Excel 9.0 object Library** en el ítem Referencias del menú Proyecto, de Visual Basic.

Tomando en cuenta los objetos, propiedades y métodos propios del modelo de objeto de Excel, se siguen los pasos expuestos anteriormente, para acceder a la aplicación, por ejemplo:

```
Dim AppExcel as Excel.Application
Set AppWord = CreateObject(“Excel.Application”)
AppExcel.Visible = True
AppExcel.Workbooks.Add
MsgBox “Aplicación de Excel creada desde Visual Basic!!!”
AppExcel = Nothing
```

La exposición de objetos permite utilizar la funcionalidad de una aplicación en otra. Por ejemplo, un procesador de texto podría exponer su capacidad de revisión ortográfica para que pueda ser utilizada por otros programas. Por ser de gran utilidad para intercambiar información entre aplicaciones, la exposición de objetos puede ser aprovechada por las herramientas de accesibilidad para capturar la información contenida en las aplicaciones.

V. DISEÑO DEL PROTOTIPO

5.1 LECTOR DE PANTALLA A DESARROLLAR

- **Estructura**

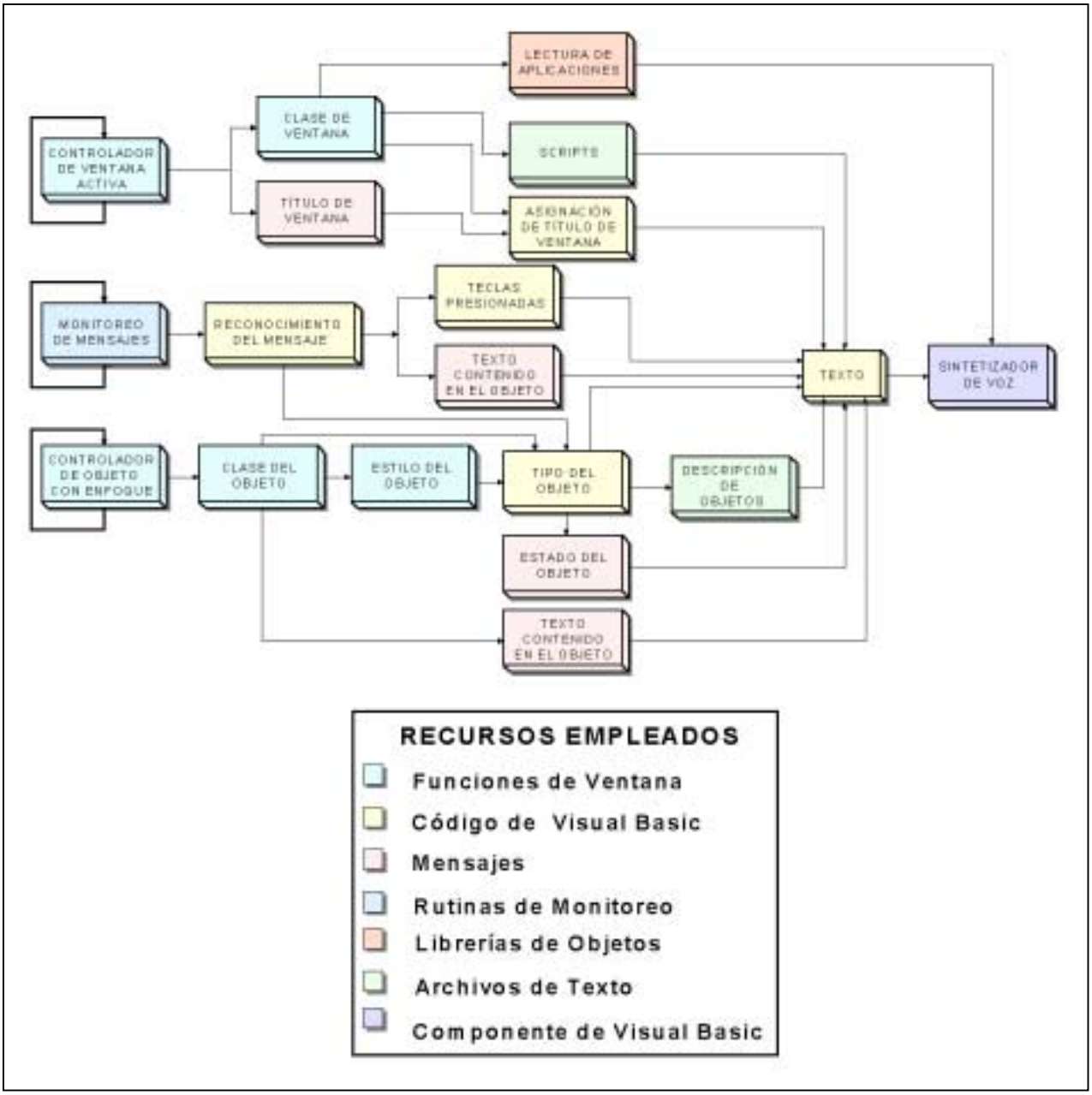

Figura 5.1 Estructura del Lector de Pantalla

El diagrama anterior muestra la estructura interna del lector de pantalla a desarrollar. En éste se representan la serie de procesos que seguirá para la recolección de la información presente en pantalla, y que posteriormente deberá transmitir en forma oral.

La estructura toma como base las características generales que se han observado en los programas lectores de pantalla, y que forman parte de la estructura básica detallada en el marco teórico de este documento. Dichas características no definen en sí mismas los métodos utilizados por estos programas para llevar a cabo cada uno de los procesos involucrados en su operación. Por el contrario, la intención del diagrama de la figura 5.1 es revelar, en términos generales, los métodos y recursos a utilizar en los distintos procesos que conformarán el prototipo.

Como fue mencionado anteriormente, uno de los procesos más importantes que se realizan en un lector de pantalla lo constituye la detección de información. La estructura del prototipo a desarrollar emplea tres procesos que pueden ser considerados como los mecanismos principales para realizar las distintas etapas de detección de información: *la recuperación del controlador de la ventana activa, la recuperación del controlador del objeto que posee el enfoque, y el monitoreo de mensajes a nivel de sistema.*

Estos procesos se ejecutan dentro de bucles en los que constantemente se actualiza la información de acuerdo a los cambios ocurridos en el sistema operativo y en las aplicaciones, provocados por acciones del usuario o por procesos internos de las mismas.

- **Controlador de la Ventana Activa**

De todos los objetos que se presentan en pantalla, la ventana que recibe las entradas del usuario (la ventana activa) será la que contenga la información que necesita ser detectada y transmitida al mismo.

Considerando que el acceso a todo objeto dentro del sistema operativo requiere un controlador, el primer paso para la detección de la ventana activa lo constituirá la captura de su controlador. Dicha captura se logra por medio de la función *GetForegroundWindow,* la cual forma parte del conjunto de funciones para ventanas proporcionadas por la API de Windows. Ya que la ventana activa puede cambiar en algún momento dado, el proceso es ubicado dentro de un bucle, con el fin de recuperar constantemente dicho controlador.

Del proceso de captura del controlador de la ventana, se derivan los procesos explicados a continuación:

- **Clase de Ventana**

  Toda ventana perteneciente al sistema operativo Windows, o a las aplicaciones basadas en el mismo, pertenece a una clase de ventana. Una vez que se tiene el controlador de la ventana activa es posible conocer la clase a la que ésta pertenece utilizando la función de la API *GetClassName*, la cual retorna una cadena de texto que representa la clase de la ventana activa.

  La clase de ventana puede ser utilizada para asociar ventanas a aplicaciones. Por ejemplo, la ventana correspondiente a la aplicación Microsoft Excel 2000, posee la clase “XLMAIN”. Es posible identificar si la ventana activa corresponde a la aplicación Microsoft Excel 2000, si al recuperar la clase se obtiene la cadena de texto “XLMAIN”. También es posible utilizar el nombre de clase para identificar ventanas que no poseen título. Por ejemplo, la barra de tareas de Windows no posee título, y pertenece a la clase “Shell_TrayWnd”. De esta forma es posible asociar el título “Barra de Tareas” cuando se detecta la ventana con la clase “Shell_TrayWnd”.

- **Titulo de la Ventana**

  Si bien el controlador de una ventana es empleado para identificarla, no representa información útil para el usuario. En su lugar, el texto contenido en la barra de título de la ventana resulta de mayor utilidad, y formará parte de la información que será transmitida al usuario.

La mayoría de ventanas y aplicaciones poseen un título que las identifica y que a la vez revela su propósito. La captura del título de la ventana activa se realiza por medio de mensajes a través de la función *SendMessage.* Especificando el mensaje WM_GETTEXT, es posible recuperar el texto de una ventana como una cadena de texto.

- **Asignación del Titulo de la Ventana:**

  Consiste en almacenar el título de la ventana activa, para que posteriormente pueda ser convertido a voz. La asignación puede llevarse a cabo por tres razones:

  1. *La ventana activa tiene título*. El texto recuperado en el proceso anterior es almacenado para posteriormente alimentar el proceso de conversión a voz.

  2. *La ventana activa es "Program Manager"* (título asignado por el sistema operativo al escritorio). El texto almacenado será "Escritorio de Windows".

  3. *La ventana activa no tiene título*. Se asigna y se almacena como título el nombre de la clase de la ventana recuperado anteriormente.

A partir de la detección de la clase de ventana es posible realizar cualquiera de los tres procesos siguientes:

1. Scripts:

Los scripts serán archivos de texto que contendrán información de apoyo sobre las aplicaciones: *Microsoft Word 2000, Micorosft Excel 2000, WinZip*, y *Winamp*. La información consiste en una breve descripción de la aplicación, menús que contiene, teclas rápidas, y cómo utilizar el lector de pantalla con dichas aplicaciones. La información se cargará en un cuadro de edición desde el cual podrá ser vocalizada e inspeccionada con mayor detenimiento.

Los scripts serán activados por una combinación de teclas. Al realizar dicha combinación, se utilizará la clase de la ventana activa para buscar el script correspondiente a las aplicaciones antes mencionadas.

2. Lectura de Aplicaciones:

En este proceso se inicia la captura del texto que manejan las aplicaciones Microsoft Word 2000, y Microsoft Excel 2000. El proceso representa un ejemplo de interacción con aplicaciones que utilizan objetos cuyas características y comportamiento están definidos por sus propias aplicaciones.

- **Monitoreo de Mensajes**

El sistema operativo y las aplicaciones se comunican por medio de mensajes. Interceptando estos mensajes es posible detectar cambios en los objetos que poseen el enfoque.

El monitoreo de mensajes de esta etapa se realiza a nivel de todas las aplicaciones que corren bajo el sistema operativo, a excepción de la aplicación “Lector de Pantalla”. Esto se llevará a cabo a través de la instalación de un hook a nivel de sistema.

La función *SetCWPMSGHook* ubicada en una librería de enlace dinámico que no pertenece a Windows, permite instalar un Hook Global (hook a nivel de sistema) para monitorear los mensajes que llegan a todas las aplicaciones, excluyendo aquélla desde la cual se invoca a la función.

Posterior al monitoreo de mensajes se ejecutan los siguientes procesos:

- **Reconocimiento del Mensaje:**

Cuando la función SetCWPMSGHook recupera un mensaje, devuelve cuatro valores: un *controlador* a la ventana a la que se dirigía el mensaje, un *identificador de mensaje*, y *dos parámetros* que contienen información adicional sobre dicho mensaje. Los mensajes se distinguen por su

identificador, el cual es un valor constante predefinido por el sistema operativo. De esta forma el reconocimiento se lleva a cabo comparando los identificadores recibidos con los valores de sus respectivas constantes.

- **Teclas Presionadas:**
  Por medio del mensaje WM_KEYDOWN, reconocido anteriormente, se reconocen las teclas que han sido presionadas y se lleva a cabo una comparación de los códigos de teclas recuperados para asociarlos con una cadena de texto que identifique cada una de dichas teclas. Posteriormente dicha cadena formará parte de la información que será transmitida al usuario.

- **Texto Contenido en el Objeto**
  A través del reconocimiento de mensajes que afectan a objetos específicos, se activan rutinas para recuperar el texto contenido en los mismos. También se recupera el texto de los objetos cuando un nuevo objeto captura el enfoque. En ambos casos se utiliza la función *SendMessage* (con los parámetros apropiados para cada objeto) para recuperarlo. Posteriormente el texto formará parte de la información que será transmitida al usuario.

- **Controlador del Objeto con Enfoque**

Para poder capturar la información perteneciente a un objeto es necesario recuperar el controlador del objeto. Éste se obtiene por medio de la función *GetFocus*.

A partir de este proceso se desencadenan los siguientes:

- **Clase del Objeto:**
  Así como todas las ventanas del sistema operativo pertenecen a una clase, los objetos o controles también son parte de una clase específica.
  Este proceso es el encargado de capturar la clase del objeto que posee el enfoque, tarea que realiza mediante la función *GetClassName*, y puede alimentar directamente diferentes procesos *(Estilo del Objeto, Tipo del Objeto,*

y Texto Contenido en el Objeto) dependiendo de la clase a la que el objeto pertenezca.

- **Estilo del Objeto:**

  A partir de los datos recuperados anteriormente (controlador y clase del objeto), se recupera el estilo del objeto que tiene el foco para todos aquellos controles pertenecientes a la clase “Button”. Esta tarea se lleva a cabo por medio de la función *GetWindowLong*, la cual retorna un valor numérico que contiene banderas que especifican el estilo de este tipo de controles. Mediante estas banderas se obtiene el tipo de botón.

  Entre los estilos que pueden ser recuperados se encuentran: *Botón de Comando, Botón de Chequeo, y Botón de Opción.*

- **Tipo del Objeto:**

  Este proceso puede ser alimentado por varios procesos anteriores, dependiendo de la forma en que haya sido obtenido el objeto que tiene el enfoque, es decir, por medio del reconocimiento de mensajes, o por la obtención de la clase o estilo del objeto. El tipo de objeto es asignado dependiendo de los valores obtenidos en dichos procesos.

- **Estado del Objeto:**

  Dependiendo del tipo de objeto asignado en el proceso anterior, se puede determinar el estado de algunas clases de objetos, entre ellos los controles de tipo botón y los ítems de menú pertenecientes a barras de menú estándares o menús contextuales (los que se activan al hacer clic con el botón derecho del ratón, o mediante la tecla de menú).

- **Descripción de Objetos:**

  La descripción de objetos constituye una característica extra que se añadirá al prototipo a desarrollar, y trabajará en base al tipo o clase del objeto.

  La descripción será activada a través de una combinación de teclas. Al

presionar dicha combinación, el programa vocalizará una pequeña descripción del tipo de objeto que contienen el enfoque en ese momento.

Todos los procesos anteriores tienen el objetivo de detectar o activar información que será transmitida en forma oral a través de los altavoces de la computadora. Esta tarea es representada en el diagrama de la figura 5.1, a través de los siguientes procesos:

- **Texto**

  Este proceso se incluye en la estructura del prototipo a desarrollar, con el objetivo de indicar que antes de dar salida a la información, ésta debe poseer el formato de cadena de texto.

  El proceso consiste en transformar la información detectada a una cadena de texto en caso de que dicha información, no corresponda originalmente a este tipo de datos.

- **Sintetizador de Voz**

  Mediante el componente *Direct Text to Speech* proporcionado por Visual Basic, y el empleo del motor L&HTTS3000, la información detectada en formato de cadena de texto, es reproducida por una voz digital en español y transmitida al usuario por medio de las bocinas de la computadora.

  El motor L&HTTS3000 provee la vocalización del texto en español, y posee dos modos de operación: *vocalización con voz femenina, y vocalización con voz masculina.*

Funciones que realizará

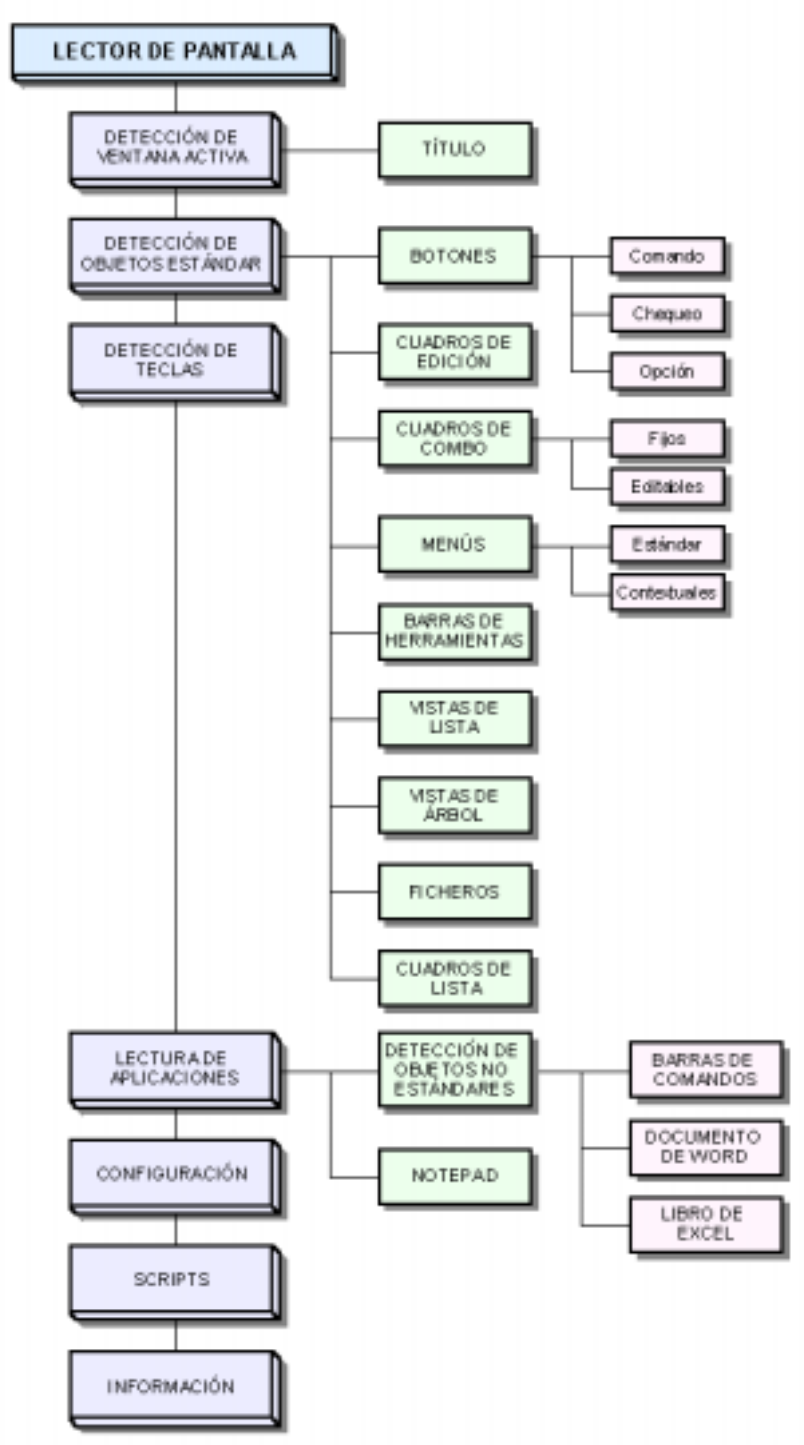

Figura 5.2 Funciones que realizará el Lector de Pantalla

El diagrama anterior muestra las diferentes funciones que será capaz de realizar el programa lector de pantalla a desarrollar. Éstas se han definido de acuerdo a las funciones que debe poseer un lector de pantalla básico y a los alcances planteados.

- **Detección de la Ventana Activa**

Cuando el usuario abra una nueva ventana, o se traslade de una ventana a otra, leerá el título de la ventana activa, permitiendo conocer la aplicación sobre la que se encuentra actualmente.

- **Detección de Objetos Estándar**

Dentro de la ventana activa, algún control en particular adquiere el enfoque para que el usuario pueda interactuar con la aplicación. El lector de pantalla reconocerá una gran parte de los objetos estándar del sistema operativo (controles predefinidos y controles comunes), detectando información que depende del tipo de objeto en particular.

Los controles detectados son los siguientes:

- **Botones**

Reconoce los siguientes tipos de botones presentes en distintas aplicaciones del sistema operativo:

1. **Botón de Comando:**
   Leerá su etiqueta, el tipo de botón, y su estado cuando se mantiene presionado.

2. **Botón de Chequeo:**
   Leerá su etiqueta, el tipo de botón, y su estado.

3. **Botón de Opción:**
   Leerá su etiqueta, el tipo de botón, y su estado.

- **Cuadros de Edición**

Este control puede poseer una gran cantidad de texto en una o varias líneas.
El lector de pantalla leerá todo su contenido, el tipo de control, y si posee una etiqueta asociada, también leerá dicha etiqueta. Al desplazarse dentro del cuadro de edición leerá línea de texto completas, caracteres, palabras, y el texto seleccionado.

- **Cuadros de Combo**

Leerá su contenido, el tipo de control, y si posee una etiqueta asociada, también leerá dicha etiqueta. Podrá leer el texto de cuadros de combos con opciones fijas, y de aquellos que permiten editar texto.

- **Menús**

Leerá el texto del ítem de menú seleccionado, el tipo de control, si contiene o no submenús, y el estado del ítem. Podrá leer los ítems de barras de menú estándar, como la de NotePad, y menús contextuales.

- **Barras de Herramientas**

Leerá el texto del ítem seleccionado, y el tipo de control.
Ejemplos de barras de herramientas lo constituyen, el menú desplegado por la tecla de Windows, conocido como *Menú Inicio*, y los botones ubicados junto al *botón inicio*.

- **Vista de Lista**

Leerá el texto del ítem seleccionado, el tipo de control, y su estado.
Estos controles se encuentran en varias aplicaciones y cuadros de diálogo del sistema operativo. La parte derecha del Explorador de Windows, y la ventana del Panel de Control constituyen algunos ejemplos de estos controles.

- **Vista de Árbol**

Leerá el texto del ítem seleccionado, y el tipo de control.
Una de las vistas de árbol más comunes es la ubicada *en la parte izquierda del Explorador de Windows.*

- **Ficheros**

Leerá el texto de la ficha seleccionada, y el tipo de control.

Los ficheros se encuentran comúnmente en los cuadros de diálogo, por ejemplo el diálogo “propiedades” de un ícono.

- **Cuadros de Lista**

Leerá el texto del ítem seleccionado, el tipo de control, y su estado.

Estos controles se encuentran en cuadros de diálogo del sistema operativo, por ejemplo, el diálogo “Añadir o Remover Programas”.

- **Detección de Teclas**

Leerá las teclas presionadas por el usuario.

- **Lectura de Aplicaciones**

El lector de pantalla tendrá la capacidad de leer el texto que manejan las aplicaciones *NotePad, Microsoft Word 2000, y Microsoft Excel 2000*.

1. **Notepad**

   Esta aplicación es proporcionada por el sistema operativo, y utiliza controles estándar en su interfaz gráfica. El lector de pantalla leerá el contenido de archivos .**txt** dentro de esta aplicación, incluyendo el texto que se agregue a dichos archivos.

2. **Microsoft Word 2000**

   Al abrir un archivo **.doc** para esta aplicación, leerá el texto contenido en el documento. También permitirá leer letras, palabras, texto seleccionado, y el número de página al cambiar de página.

3. **Microsoft Excel 2000**

   Al abrir un archivo **.xls** para esta aplicación, permitirá leer el texto contenido en las celdas del libro y su posición, al desplazarse por las mismas. También leerá la hoja de trabajo al cambiar de hoja, y el rango de celdas seleccionadas.

- **Detección de Objetos no Estándares**

  Muchas aplicaciones que no pertenecen al sistema operativo utilizan otros tipo de objetos que no están definidos por el sistema operativo, por tanto requieren un tratamiento especifico para poder ser reconocidos.

  El lector de pantalla podrá detectar los siguientes objetos no estándares:

  - **Barras de Comandos**

  Leerá el texto del ítem seleccionado, el tipo de control, si contiene o no submenús, y el estado del ítem.
  Las aplicaciones de Microsoft Office 2000 como Word 2000, y Excel 2000, contienen estos objetos en lugar de barras de menús estándar.

  - **Objetos pertenecientes a la Clase _WWG**

  A través de la detección de este objeto, es posible iniciar un proceso de conexión con la aplicación Microsoft Word 2000, para permitir la lectura de dicha aplicación, en la forma descrita anteriormente.

  - **Objetos pertenecientes a la Clase Excel7**

  A través de la detección de este objeto, es posible iniciar un proceso de conexión con la aplicación Microsoft Excel 2000, para permitir la lectura de dicha aplicación, en la forma descrita anteriormente.

- **Configuración**

  Permitirá configurar distintas opciones del lector de pantalla, con el fin de adaptar la aplicación a las necesidades y preferencias del usuario.

- **Scripts**

  Desplegará una ventana de información en la que se cargarán archivos de texto (para poder ser explorados detenidamente) con información de ayuda y apoyo para los siguientes programas no estándares del Sistema Operativo: *Microsoft Word, Microsoft Excel, Winamp, y Winzip.*

- **Información**

  Desplegará una ventana de información en la que se cargarán archivos de texto (para poder ser explorados detenidamente) con la siguiente información:

  1. Información sobre el uso general del programa.
  2. Descripción de aplicaciones tales como Notepad, Menú Inicio, y Explorador de Windows.
  3. Información de teclas de acceso rápido definidas por el Sistema Operativo.

  Además, podrá vocalizar una pequeña descripción del tipo de objeto que contienen el enfoque en ese momento, al presionar una combinación de teclas.

5.2 ÁRBOL DE OPCIONES

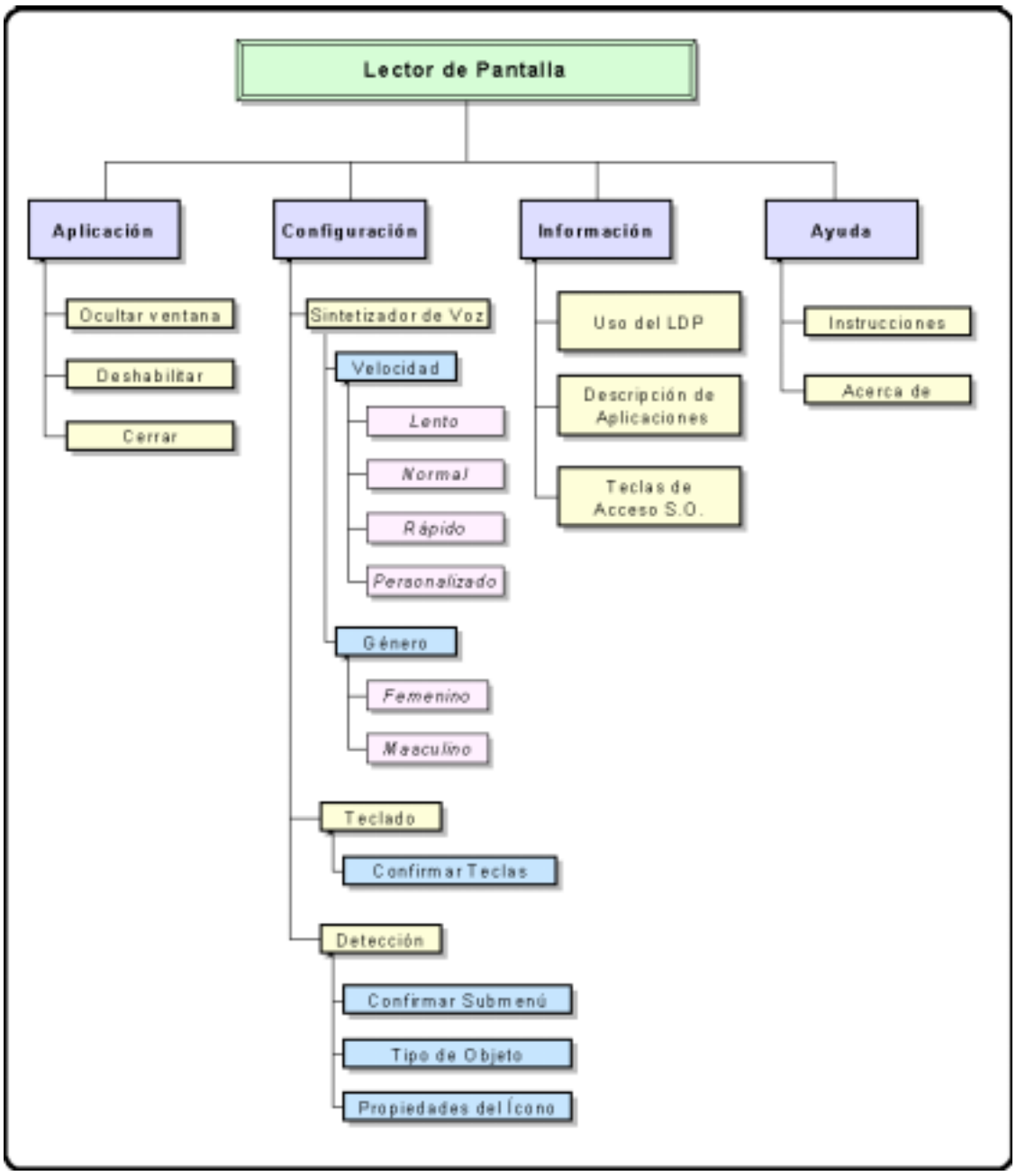

Figura 5.3. Árbol de Opciones del Lector de Pantalla

El diagrama anterior representa las opciones de menú con las que contará el programa lector de pantalla, como parte de su interfaz gráfica de usuario. La interfaz será muy sencilla, y tiene como objetivo brindar algunas opciones con las

que el usuario podrá configurar el sintetizador de voz, aspectos de la detección de información, y activar información del sistema de ayuda que proporcionará guías para la utilización del programa.

La función de cada una de las opciones incluidas se definen a continuación.

- **Aplicación**

El menú aplicación contendrá opciones de manejo de la ventana de la aplicación. Sus componentes son:

- **Ocultar ventana:**

  Ocultará la ventana del programa y colocará un ícono en el área de estado de la barra de tareas. Esta opción puede ser también accedida por medio de la combinación de teclas *"Control"* + *"Signo de División del teclado numérico"*. Para hacer visible de nuevo la ventana, se presiona la misma combinación de teclas.

- **Deshabilitar:**

  Mediante esta opción las funciones de detección del lector de pantalla podrán ser pausadas o restablecidas.

- **Cerrar:**

  Se utilizará para cerrar la aplicación.

- **Configuración**

Este submenú permitirá al usuario realizar distintas configuraciones al funcionamiento interno del programa Lector de Pantalla.

- **Sintetizador de Voz**

  Permitirá hacer arreglos personalizados sobre el sintetizador de voz que utiliza el programa. Las opciones que contendrá son:

a) Velocidad

Permitirá regular la velocidad de la voz del sintetizador. Esta opción contendrá 4 alternativas que podrán ser elegidas una a la vez por el usuario: *Lento, Normal (valor por defecto del programa), Rápido, y Personalizado*. Para esta última opción, el usuario podrá introducir un valor numérico entre 95 y 225, que definirá la velocidad de la voz del sintetizador.

b) Género

Esta opción permite al usuario elegir el género de la voz del sintetizador, el cual puede ser: *Masculino (valor por defecto del programa), o Femenino.*

- **Teclado**

  Contendrá la opción **Confirmar Teclas**, mediante la cual se indicará si el programa vocaliza o no las teclas presionadas. También podrá ser habilitada o deshabilitada utilizando la combinación de teclas *"Control" + "2 del teclado numérico"*. Por defecto la confirmación de teclas estará elegida.

- **Detección**

  Contendrá opciones que podrán ser elegidas para configurar la información a ser vocalizada. Por defecto todas las opciones que este submenú contiene están elegidas.

a) Confirmar Submenús

Permitirá que el programa indique si el ítem de menú que ha sido seleccionado, contiene un submenú.

b) Tipo de Objeto

Se podrá elegir esta opción para que el programa indique o no el tipo de objeto que ha adquirido el enfoque. También podrá ser habilitada o deshabilitada utilizando la combinación teclas *"Control" + "3 del teclado alfanumérico"*.

c) Propiedades del Ícono

Si esta opción está chequeada, la aplicación vocalizará las propiedades del

ícono seleccionado, al presionar la combinación de teclas *"Control"* + *"Signo de Adición del teclado numérico"*.

- **Información**

A través de este menú, se ofrecen al usuario características extras respecto a las de otros programas lectores de pantalla. Le permitirá recibir información adicional sobre el sistema operativo, y algunas aplicaciones del mismo, así como también información sobre el lector de pantalla.

La información se tomará de archivos de texto predefinidos, y se colocará en el cuadro de edición de una ventana (la ventana de información) creada por el lector de pantalla, con el fin de que el usuario pueda acceder fácilmente a la misma. Además, se le proporcionarán las teclas *"Barra espaciadora", "Signo de Adición del teclado numérico", y "Enter"*, las cuales podrá utilizar para desplazarse por los temas principales, o por los subtemas hacia abajo, y hacia arriba, respectivamente.

Las opciones que contendrá son:

- **Uso del Lector de Pantalla (LDP)**

  Al seleccionar este ítem, la aplicación cargará en la ventana de información, un resumen sobre el uso del Lector de Pantalla. Consistirá básicamente en una breve descripción del programa, y un listado de las teclas de acceso rápido a las opciones del programa.

- **Descripción de Aplicaciones**

  Describirá en la ventana de información, el uso de los siguientes componentes del sistema operativo: *Menú Inicio, Explorador de Windows, y NotePad.*

- **Teclas de acceso del Sistema Operativo**

  Cargará una lista de las teclas de acceso a ciertas funciones del sistema operativo, como buscar (Windows + F), y mostrar escritorio(Windows + M).

- **Ayuda**

Este menú contendrá opciones que proporcionarán una guía de utilización del programa lector de pantalla al usuario, así como información general del mismo. Contará con la opción **Instrucciones**, con la cual se cargará la guía de uso del lector de pantalla en la ventana de información, y **Acerca de**, la cual desplegará un cuadro de diálogo con información sobre la versión del lector de pantalla.

5.3 DISEÑO DE LA INTERFAZ DE LA APLICACIÓN

Requerimientos de la Interfaz

El diseño de la interfaz de la aplicación se realizará atendiendo los siguientes requerimientos:

- Para la ventana de la aplicación:

  - ✓ Tamaño de ventana fijo.
  - ✓ Botones maximizar y minimizar no requeridos.
  - ✓ Opciones de control de la aplicación en una barra de menús estándar, según el diseño del árbol de opciones.
  - ✓ Incorporación de métodos de tecla abreviada para activar o desactivar en cualquier momento, determinadas características de la aplicación.
  - ✓ Vocalización de la información detectada.
  - ✓ Visualización de una parte de la información detectada.

- Para la ventana de información de la aplicación:

  - ✓ Tamaño de ventana fijo.
  - ✓ Botones maximizar y minimizar no requeridos.
  - ✓ Cuadro de Edición para cargar la información.
  - ✓ Teclas de acceso a los temas y subtemas desplegados
  - ✓ Botón de Comando para cerrar la aplicación.

Diseño de la Interfaz

• Teclas Rápidas de la Aplicación

Las teclas rápidas para la aplicación se incorporarán registrándolas en el sistema operativo a través de la función *RegisterHotKey*, y posteriormente subclasificando la ventana de la aplicación, para recibir las combinaciones de teclas designadas con la función. Dichas combinaciones de teclas se presentan a continuación:

1. **Ctrl + "/"** (teclado numérico): Oculta o restaura la ventana de la aplicación.
2. **Ctrl + "-"** (teclado numérico): Vocaliza una vez más la información detectada recientemente.
3. **Ctrl + "*"** (teclado numérico): Detendrá la vocalización de información.
4. **Ctrl + "+"** (teclado numérico): Vocaliza las propiedades del ícono seleccionado.
5. **Ctrl + "0"** (teclado alfanumérico): Restaura los valores originales de configuración del programa.
6. **Ctrl + "1"** (teclado alfanumérico): Activa el Script disponible para una aplicación determinada.
7. **Ctrl + "2"** (teclado alfanumérico): Activa o desactiva la vocalización de las teclas presionadas.
8. **Ctrl + "3"** (teclado alfanumérico): Activa o desactiva la vocalización del tipo del objeto sobre el que se tiene el enfoque.
9. **Ctrl + "4"** (teclado alfanumérico): Activa o desactiva la descripción del objeto seleccionado.

• Ventana de la Aplicación

De acuerdo con los requerimientos antes mencionados, se ha definido un tamaño fijo para la ventana de la aplicación. Los botones maximizar y minimizar no estarán presentes; únicamente el botón cerrar permanecerá visible. Posee una barra de menús en su parte superior, atendiendo al árbol de opciones.

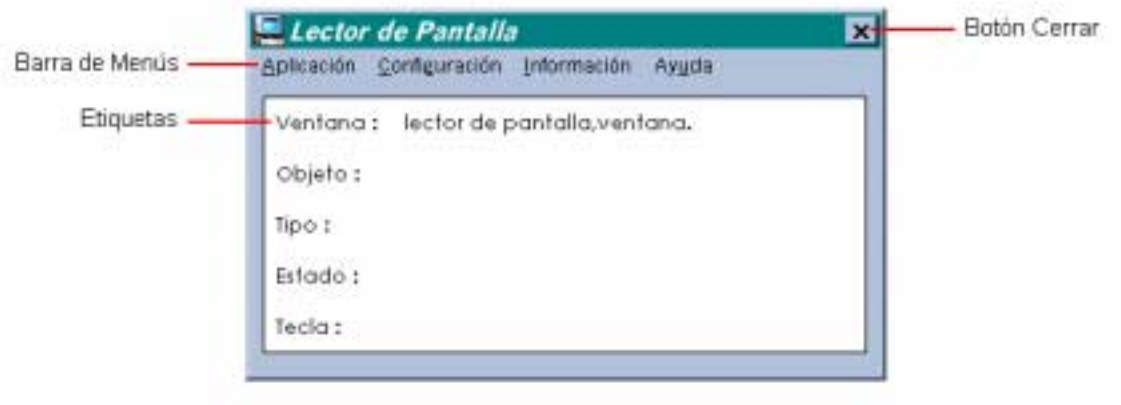

Figura 5.4 Ventana de la Aplicación Lector de Pantalla

Para la vocalización de la información, utiliza un control de conversión de texto a voz (DirectSS). Sin embargo, este componente no será visible para el usuario.

Finalmente, parte de la información detectada será visualizada a través de un conjunto de etiquetas en las que se ha clasificado la información, comenzando por la ventana activa, luego, el objeto con enfoque, el tipo de este objeto, su estado, y la tecla presionada.

- **Teclas de Acceso en la Ventana de Información**

En la ventana de información, la información se despliega en un cuadro de texto con el objetivo de que sea explorada detenidamente por el usuario, aprovechando la capacidad de detección del lector de pantalla sobre este tipo de controles. Para desplazarse con mayor facilidad sobre la información, se han asignado las siguientes teclas: *Barra Espaciadora*, para ubicarse en el siguiente tema, *Signo de Adición del teclado numérico*, para buscar el subtema más próximo a partir de la posición actual del cursor, y *Enter*, para buscar, en sentido inverso, el subtema más próximo a partir de la posición actual del cursor.

- **Ventana de Información**

Al igual que la ventana de la aplicación, la ventana de información posee un tamaño fijo, con el botón cerrar visible.

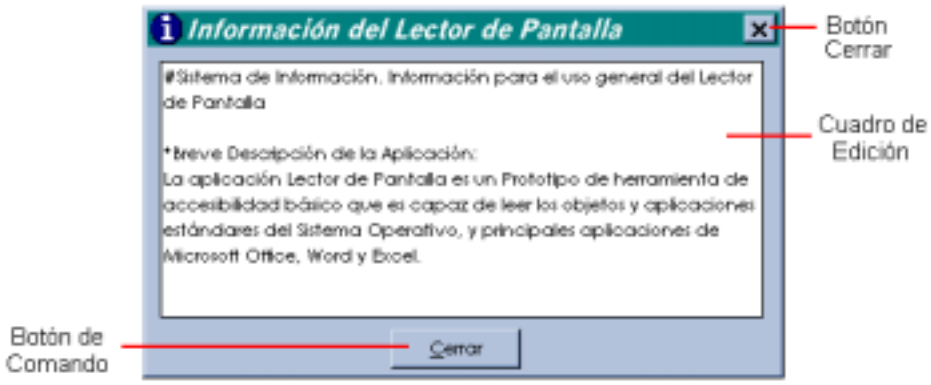

Figura 5.5 Ventana Información del Lector de Pantalla

Posee un cuadro de edición con múltiples líneas, para contener la información que se desea dar a conocer al usuario. El cuadro de edición utilizado para esta ventana, no permite que la información sea alterada, sino únicamente explorada.

La ventana de información puede ser cerrada utilizando la combinación de teclas ALT+F4, o presionando el botón de comando cerrar, al cual puede llegarse por medio de la tecla TAB.

VI. DESARROLLO DEL PROTOTIPO

6.1 INICIALIZACIÓN DEL FORMULARIO DE LA APLICACIÓN

- ***Flujograma de Inicialización de la Aplicación***

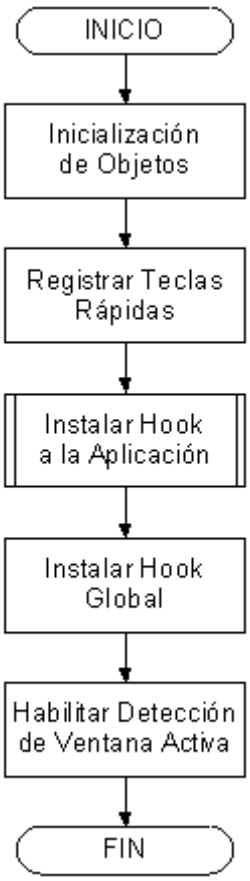

Figura 6.1 Flujograma Inicialización

Durante el proceso de carga de la aplicación, se inicializan distintos objetos:

1. Se indica que la ruta de búsqueda de archivos sea la de la aplicación.
2. Se recupera el identificador de thread de la aplicación.
3. Se inicializa la estructura utilizada para colocar un ícono en la barra de estado.
4. Se inicializa el estado de los ítems del menú (se chequean: Velocidad Normal, Voz masculina, Confirmar teclas y Submenús); y se recuperan los dos modos que posee el sintetizador de voz (voz masculina y femenina), colocándolo en modo de voz masculina.
5. Posteriormente, se realiza la instalación de las teclas rápidas (activación de ventana, detener el sintetizador, y releer la información detectada).
6. Finalmente, se realiza la llamada a una subrutina que inicializa el monitoreo de los mensajes que llegan a la aplicación (instalación de un hook a nivel de aplicación), y se instala un hook a nivel global para monitorear los mensajes recibidos en el resto de aplicaciones que corren bajo el sistema operativo.

6.2 DESARROLLO DE LAS OPCIONES DEL MENÚ DE LA APLICACIÓN

Para el desarrollo y codificación o programación posterior de la aplicación se diseñaron una serie de diagramas de flujo que ilustran y detallan en qué forma trabajaran cada una de las opciones incluidas dentro del menú de la aplicación.

Menú Aplicación

1. Opción Ocultar Ventana

- ***Diagrama de Flujo:***

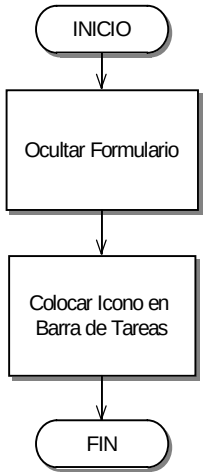

Figura 6.2 Flujograma Ocultar Ventana

La activación de la opción permitirá ocultar la ventana de la aplicación que actúa como lector de pantalla, mostrando únicamente en el área de estado de la barra de tareas, el icono asignado a la aplicación. Esta se realiza a través de la ejecución de dos procesos:

- Ocultar el formulario.
- Colocar Icono en la barra de tareas.

El área de estado de la barra de tareas, contiene iconos que proveen acceso rápido a los programas, otros iconos pueden aparecer temporalmente proporcionando información acerca del estado de las actividades. Es por ello que el ícono asignado a la aplicación lectora de pantalla será enviado a la sección antes mencionada, en espera que la combinación de teclas rápidas asignada para hacer visible la aplicación sea utilizada por el usuario.

2. Opción Deshabilitar

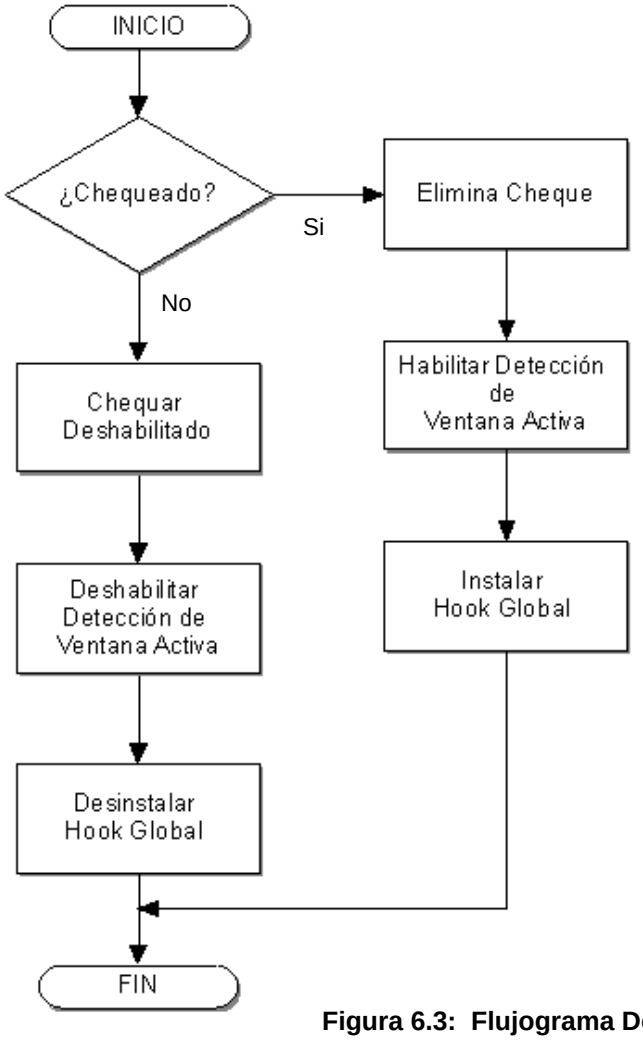

Figura 6.3: Flujograma Deshabilitar

Esta opción es la encargada de pausar o reanudar las funciones de detección de información por parte del Programa Lector de Pantalla. Para ello debe llevar a cabo la ejecución de una serie de procesos detallados a continuación:

1. Verificar si la opción Deshabilitar esta chequeada o no:

   Si esta chequeada, es decir las funciones del lector de pantalla están pausadas, realiza lo siguientes procesos:

   - Elimina el cheque a la opción.
   - Habilita el proceso de detección de la ventana activa
   - Instala el Hook Global, en este proceso se habilita el Monitoreo Global de los mensajes.

De no estar chequeada la opción, realiza los procesos siguientes:

- Chequea la opción.
- Deshabilita el proceso de detección de la ventana activa
- Desinstala el Hook Global.

3. Opción Cerrar

- ***Diagrama de Flujo:***

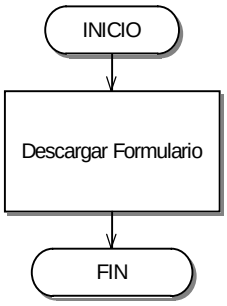

Figura 6.4 Flujograma Cerrar

Esta opción será la encargada de descargar el formulario para terminar la ejecución de la aplicación, siendo este el único proceso a ejecutar.

La aplicación también podrá ser cerrada a través de la combinación de teclas predefinida por el sistema operativo (ALT + F4) o al hacer clic en el botón cerrar de la ventana.

Sin embargo, el proceso descargar formulario realiza un proceso interno en el que desinstala las funciones utilizadas y llamadas por la aplicación, este proceso se presenta en el segundo diagrama de flujo presentado para esta opción.

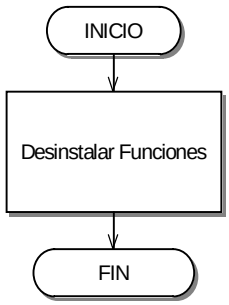

Figura 6.5 Flujograma Descargar Formulario

Menú Configuración

- **Sintetizador de Voz - Submenú Velocidad**

Sintetizador de voz tiene la capacidad de vocalizar el texto a diferentes velocidades, éste ofrecerá cuatro opciones para modificar ésta velocidad.

1. Opción Velocidad Lento

- ***Diagrama de Flujo***

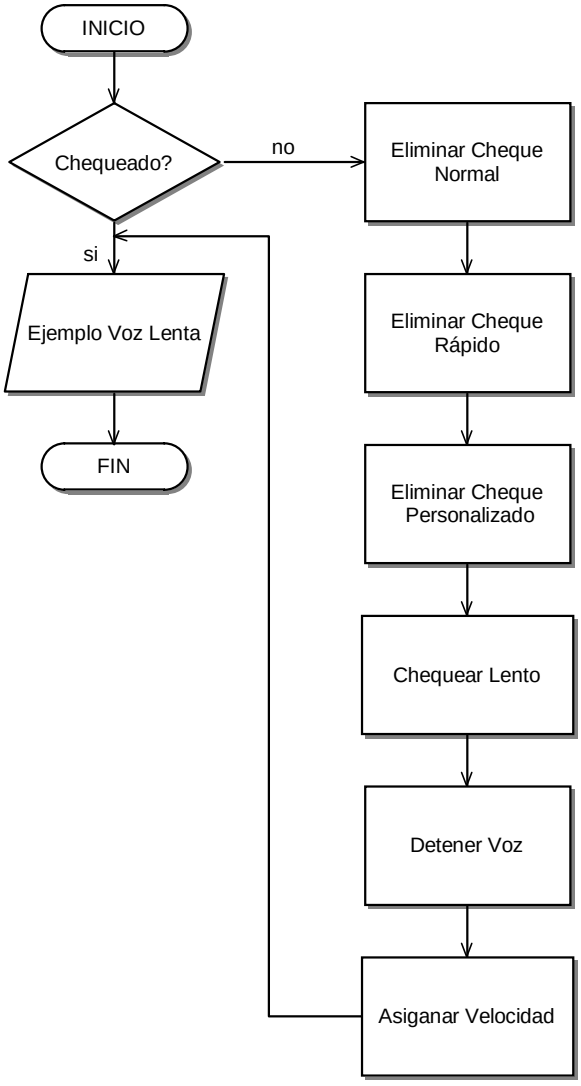

Figura 6.6 Flujograma Velocidad Lenta

Al seleccionar ésta opción, se asignará la velocidad mínima permitida por el sintetizador de voz. Para ello, realiza una serie de decisiones y procesos, que se detallan a continuación:

1. Verificar si la opción Lenta está chequeada.

   De estar chequeada realizará un único proceso:

   - Mandar un mensaje oral con un ejemplo de cómo se escucha la voz lenta del sintetizador.

2. De no estar chequeada ésta opción realiza los siguientes procesos:
   - Eliminar cheque a opción de velocidad Normal.
   - Eliminar cheque a opción de velocidad Rápida.
   - Eliminar cheque a opción de velocidad Personalizada.
   - Poner cheque a opción de velocidad Lenta.
3. Detener Voz.
4. Asignar Velocidad Lenta al sintetizador, éste es un valor numérico que se ha destinado por defecto para ésta velocidad.
5. Mandar un mensaje de forma oral, como ejemplo de cómo se escucha la voz lenta del sintetizador.

2. Opción Velocidad Normal

- ***Diagrama de Flujo***

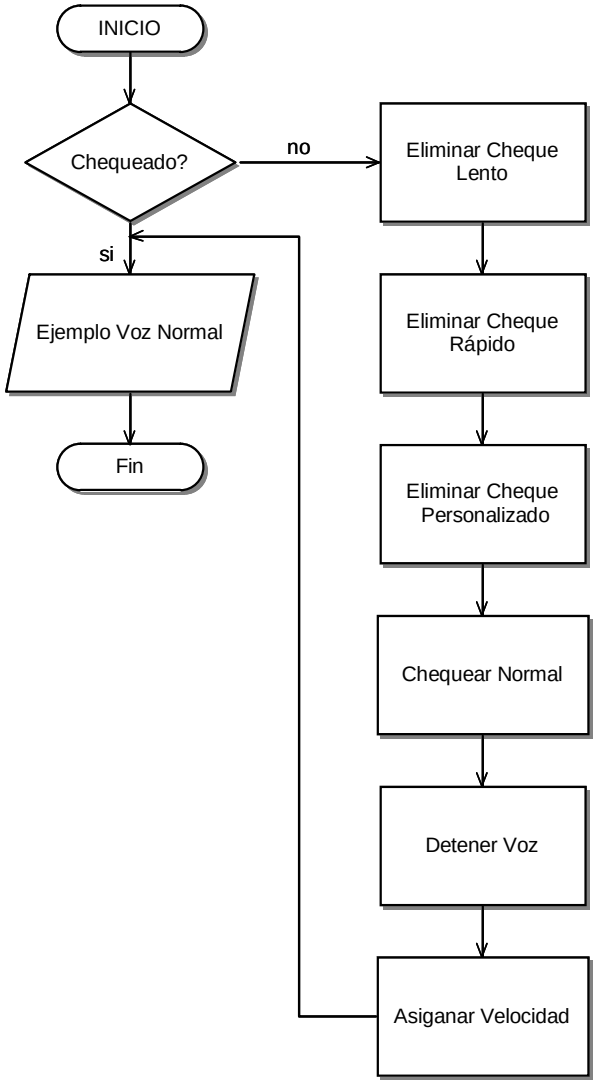

Figura 6.7 Flujograma Velocidad Normal

Al seleccionar ésta opción se asignará la velocidad Normal determinada para velocidad al sintetizador de voz. Cabe destacar que ésta opción será la chequeada por defecto al momento de inicialización de la aplicación.

Para ello realiza una serie de decisiones y procesos, que se detallan a continuación:

1. Verificar si la opción Normal está chaqueada.

   De estar chequeada realizará un único proceso:

- Mandar un mensaje oral con un ejemplo de cómo se escucha la voz normal del sintetizador.

2. De no estar chequeada ésta opción realiza los siguientes procesos:
   - Eliminar cheque a opción de velocidad Lenta.
   - Eliminar cheque a opción de velocidad Rápida.
   - Eliminar cheque a opción de velocidad Personalizada.
   - Poner cheque a opción de velocidad Normal.
3. Detener Voz.
4. Asignar Velocidad Normal al sintetizador, éste es un valor numérico que se ha destinado por defecto para ésta velocidad.
5. Mandar un mensaje de forma oral, como ejemplo de cómo se escucha la voz en velocidad normal del sintetizador.

3. Opción Velocidad Rápida

- ***Diagrama de Flujo***

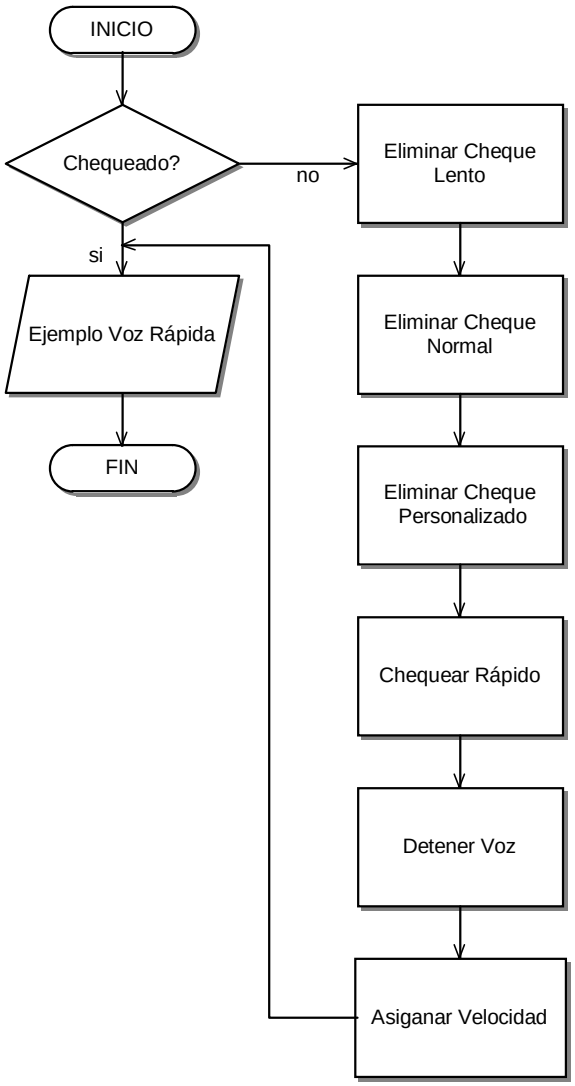

Figura 6.8 Flujograma Velocidad Rápida

Al seleccionar ésta opción, se asignará la velocidad Rápida permitida por el sintetizador de voz. Para ello, realiza una serie de decisiones y procesos, que se detallan a continuación:

1. Verificar si la opción Rápida está chaqueada.
   De estar chequeada realizará un único proceso:
   - Mandar un mensaje oral con un ejemplo de cómo se escucha la voz rápida del sintetizador.
2. De no estar chequeada ésta opción realiza los siguientes procesos:

- Eliminar cheque a opción de velocidad Lenta.
- Eliminar cheque a opción de velocidad Normal.
- Eliminar cheque a opción de velocidad Personalizada.
- Poner cheque a opción de velocidad Rápida.

3. Detener Voz.
4. Asignar Velocidad Rápida al sintetizador, éste es un valor numérico que se ha destinado por defecto para ésta velocidad.
5. Mandar un mensaje de forma oral, como ejemplo de cómo se escucha la voz Rápida del sintetizador.

4. Opción Velocidad Personalizada

- ***Diagrama de Flujo***

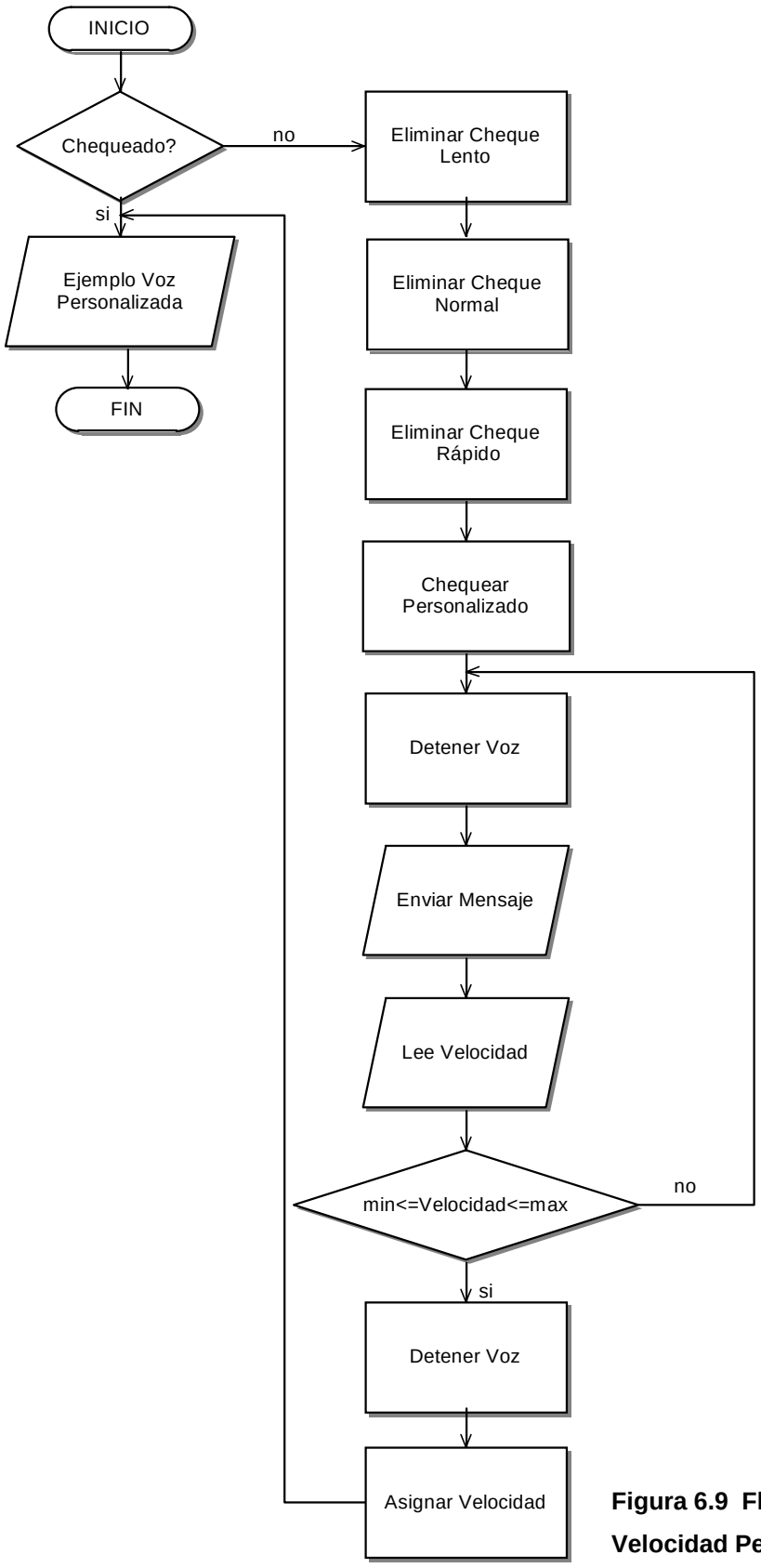

Figura 6.9 Flujograma Velocidad Personalizada

Al seleccionar ésta opción se asignará un valor numérico determinado e introducido por el usuario para especificar la velocidad del sintetizador de voz. Este valor posee un límite inferior y uno superior asignados.

Para ello se realizan una serie de decisiones y procesos, que se detallan a continuación:

- Verificar si la opción Personalizada esta chequeada.

  De estar chequeada realizará un único proceso:

  - Mandar un mensaje oral con un ejemplo de cómo se escucha la voz con la velocidad personalizada por el usuario anteriormente.

- De no estar chequeada ésta opción realiza los siguientes procesos:
  - Limpiar cheque a opción de velocidad Lenta.
  - Eliminar cheque a opción de velocidad Normal.
  - Eliminar cheque a opción de velocidad Rápida.
  - Poner cheque a opción de velocidad Personalizada.
- Detener Voz.

4. Enviar mensaje, la aplicación mandará un mensaje oral al usuario que indicará que debe introducir un valor entre un rango determinado para configurar la velocidad del sintetizador de voz.
5. Leer Velocidad, leerá el valor numérico introducido por el usuario.

4. Comparar valor introducido con los valores establecidos como límites para éste.

   Si el valor esta entre el rango establecido:

   - Detener Voz.
   - Asignar Velocidad escogida por el usuario al sintetizador, éste es un valor numérico escogido por el usuario para esta velocidad.
   - Mandar un mensaje oral con un ejemplo de cómo se escucha la voz del sintetizador.

   Si el valor esta fuera del rango establecido:

   - Regresa al proceso número 3.

- **Sintetizador de Voz - Submenú Género**

1. Opción Género Femenino

- ***Diagrama de Flujo***

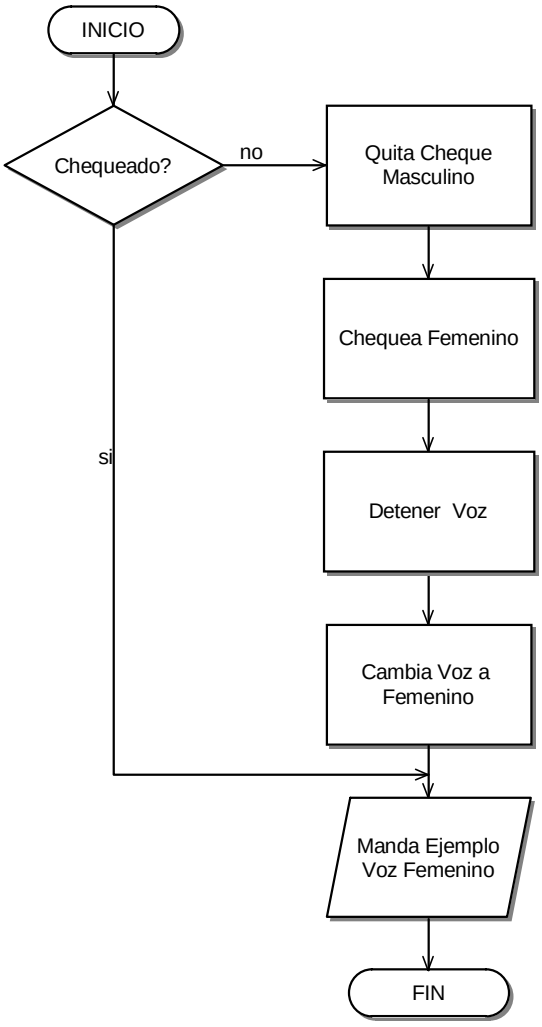

Figura 6.10 Flujograma Género Femenino

Esta opción permitirá cambiar el género de la voz al sintetizador de la aplicación, para ello realizará decisiones y procesos detallados a continuación:

1. Verificar si la opción Femenina esta chequeada.
   De estar chequeada realizara un único proceso:
   - Mandar un mensaje oral con un ejemplo de cómo se escucha la voz Femenina.

De no estar chequeada esta opción realiza los siguientes procesos:

- Elimina cheque a opción de Voz Masculina.
- Poner cheque a opción de Voz Femenina.

2. Detener Voz.
3. Cambia Voz a Femenino.
4. Manda Mensaje oral con un ejemplo de cómo se escucha la voz en género femenino.

2. Opción Género Masculino

- ***Diagrama de Flujo***

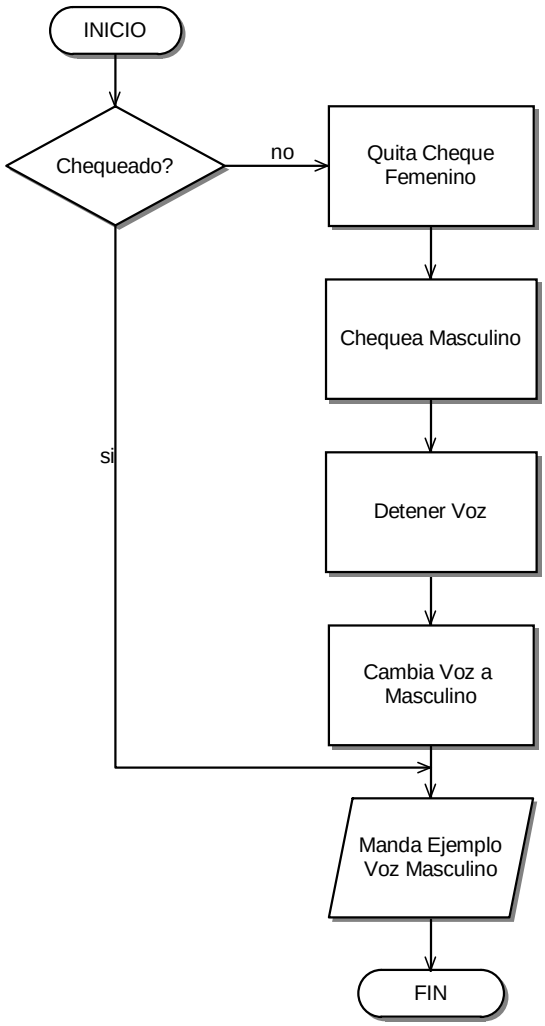

Figura 6.11 Flujograma Género Masculino

Esta opción permitirá cambiar el género de la voz al sintetizador de la aplicación a masculino, para ello realizará decisiones y procesos detallados a continuación:

1. Verificar si la opción Masculino esta chequeada.

   De estar chequeada realizara un único proceso:

   - Mandar un mensaje oral con un ejemplo de cómo se escucha la voz Masculina.

   De no estar chequeada esta opción realiza los siguientes procesos:

   - Elimina cheque a opción de Voz Femenina.
   - Poner cheque a opción de Voz Masculina.

2. Detener Voz.
3. Cambia Voz a Masculino.
4. Manda Mensaje oral con un ejemplo de cómo se escucha la voz en género masculino.

- **Submenú Teclado y Detección**

- ***Diagrama de Flujo de para Opciones de los submenús Teclado y Detección***

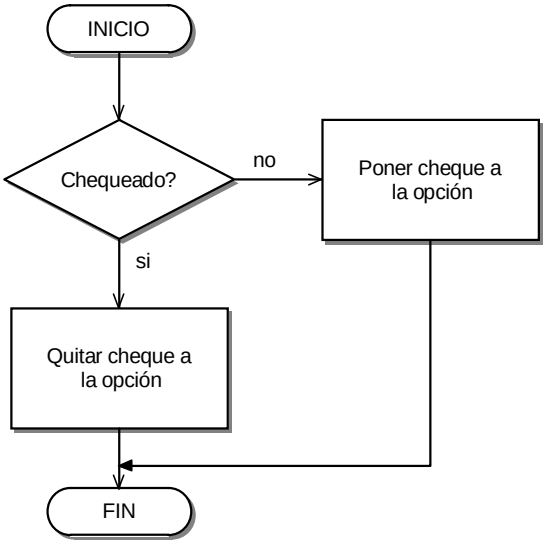

Figura 6.12 Flujograma Detección de Teclado y Submenús

Los diagramas de flujo que describen la serie de procesos que siguen éstos submenús para poder activarse es el mismo para todas las opciones, es decir, el diagrama de flujo para las opciones Confirmar Tecla, Confirmar Submenú, Tipo de Objeto y Propiedades del Icono tienen un proceso similar.

Estas opciones permitirán escoger al usuario si desea o no recibir cierta información de los objetos, que en un dado momento no puede ser necesaria o ya puede ser conocida por el usuario.

Los procesos que realizan estas opciones son:

1. Verificar que la opción este chequeada.

   Si no esta chequeada

   - Pone cheque a la opción, este cheque será leído posteriormente por otros procesos que ejecutaran las instrucciones dadas para la opción seleccionada.

   Si esta chequeada

   - Quita el cheque a la opción, este estado será leído por otros procesos y se detendrá la ejecución de dicha opción.

Menú Información

- **Submenú Uso del LDP, Descripción de Aplicaciones y Teclas al Sistema Operativo.**

Figura 6.13: Flujograma Menú Información

Este proceso detalla la serie de pasos que se deben llevar a cabo por las opciones del menú Información para el despliegue de la información al usuario.

- Almacenar nombre del archivo, posterior a este paso se deben haber creado los archivos extensión .txt correspondientes, y posterior almacenar en una variable el nombre de cada archivo, que será utilizada por otra función posteriormente.
- El proceso Activar Formulario de Información es el encargado de cargar el formulario que desplegara la información, para ello hace uso de una función desplegará en este formulario el archivo .txt correspondiente.

6.3 DESARROLLO DE DETECCIÓN DE VENTANA ACTIVA

- **Flujograma Inicialización de la Aplicación**

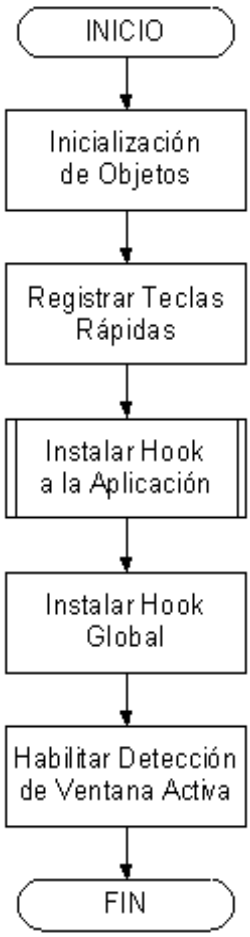

Figura 6.14 Flujograma De Inicialización

Durante el proceso de carga de la aplicación, se inicializan distintos objetos: en primer lugar se indica que la ruta de búsqueda de archivos sea la de la aplicación; se recupera el identificador de thread de la aplicación; se inicializa la estructura utilizada para colocar un ícono en la barra de estado; se inicializa el estado de los ítems de los menús (se chequean: Velocidad Normal, Voz masculina, Confirmar teclas y Submenús); y se recuperan los dos modos que posee el sintetizador de voz (voz masculina y femenina), colocándolo en modo de voz masculina.

Posteriormente, se realiza la instalación de las teclas rápidas (activación de ventana, detener el sintetizador, y releer la información detectada). Finalmente, se realiza la llamada a una subrutina que inicializa el monitoreo de los mensajes que llegan a la aplicación (instalación de un hook a nivel de aplicación), y se instala un hook a nivel global para monitorear los mensajes recibidos en el resto de aplicaciones que corren bajo el sistema operativo.

- **Flujograma Detección de Ventana Activa y Objetos con el enfoque**

Este proceso, desde su inicio hasta su fin, está contenido dentro de un temporizador, con el fin de ejecutarlo continuamente cada 200 milisegundos. Dentro de éste, se lleva a cabo la detección de la ventana activa por medio de su controlador; también se lleva a cabo la recuperación del controlador que indica el control u objeto que posee el enfoque, y a partir de éste se recupera su respectiva clase, con el fin de identificar el tipo de control, y llevar a cabo el proceso apropiado para recuperar su texto.

Visto en forma global, el proceso inicia con la recuperación del controlador de la ventana activa (a través de GetForegroundWindow), y luego realiza otros subprocesos dependiendo de tres condiciones:

1. Si la ventana activa (VA) detectada es distinta a la detectada anteriormente.
   De resultar cierta, debe recuperarse el nuevo título de la ventana y en el caso de haber una conexión anterior entre los threads de la ventana activa anterior y la ventana de la aplicación, es necesario desconectarlo. Luego debe reconectarse el thread de esta última con la nueva ventana detectada, siempre y cuando la nueva ventana no corresponda a la aplicación, dado que se realizaría una conexión con la misma ventana, lo cual es incorrecto. El cumplimiento de la primera condición termina aquí (conector 1), y no hay otro subproceso que ejecutar, de resultar falsa. A continuación, se llega a la segunda condición.
   La conexión y desconexión de threads está relacionada con la captura del objeto que posee el enfoque, realizada en la siguiente condición mediante la función GetFocus. En condiciones normales, esta función sólo recupera controladores correspondientes a objetos que pertenecen al thread que creó la ventana que llama a la función, es decir a la aplicación lector de pantalla. Sin embargo, su capacidad de detección puede ser extendida al realizar una conexión entre threads, abarcando de esta forma los objetos pertenecientes a otras aplicaciones. Esta operación se realizó bajo el cumplimiento de la primera condición, tratada anteriormente.

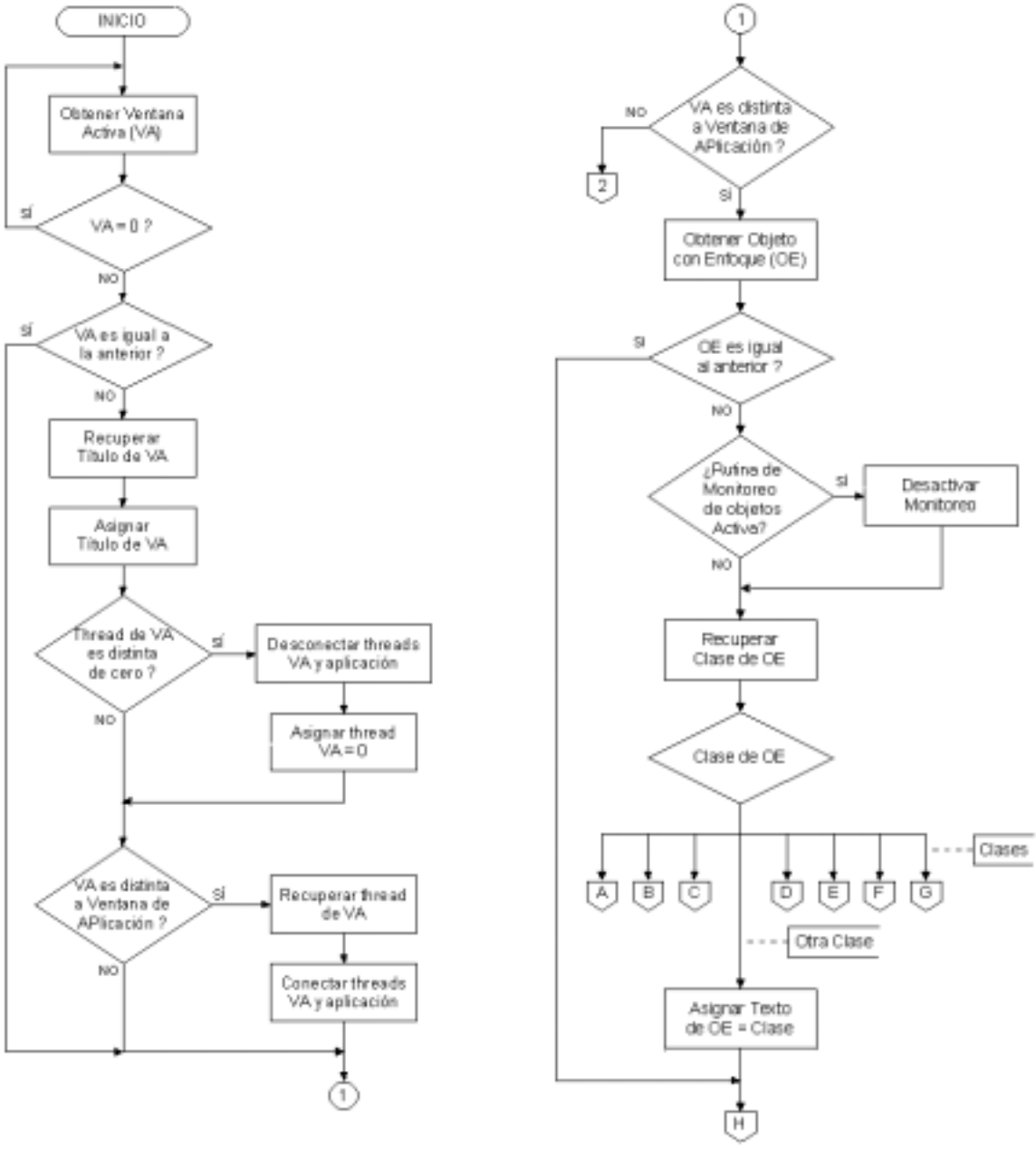

Figura 6.15 Flujograma Ventana Activa y Objetos con Enfoque

2. Si la ventana activa detectada es distinta de la ventana de la aplicación.
   De resultar cierta, debe recuperarse el nuevo objeto que posee el enfoque (a través de GetFocus). En condiciones normales, la función GetFocus sólo recupera controladores correspondientes a objetos que pertenecen al thread que creó la ventana que llama a la función, es decir a la aplicación lector de

pantalla. Sin embargo, su capacidad de detección puede ser extendida al realizar una conexión entre threads, abarcando de esta forma los objetos pertenecientes a otras aplicaciones. Esta operación se realizó bajo el cumplimiento de la primera condición, tratada anteriormente.

Posterior a la recuperación del objeto que contiene el enfoque, se presenta una condición interna (con respecto a la actual condición). Ésta verificará si el objeto con enfoque (OE) detectado es distinto al OE detectado anteriormente. De ser cierta, se ejecuta lo siguiente:

- Desactivar el monitoreo de objetos de tipo Vista de Lista, Vista de Árbol, Ficheros, Control Edit, si anteriormente se estaba realizando alguno de ellos. Esto debe ocurrir debido a que se ha encontrado un OE, distinto a uno de los tipos mencionados anteriormente.

- Recuperar la clase del objeto OE, la cual puede corresponder a distintas clases predefinidas (Button, Edit, y otros). Para cada clase predefinida que sea detectada, se realiza una operación que recupera su texto, y a partir de su clase, se asigna un tipo de objeto que será el escuchado posteriormente. Con la clase de objeto “Button” se lleva a cabo una tarea más que no es realizada en el resto: la recuperación del estilo de botón. Esto se debe a que la clase botón representa a tres controles: *botones de comando, botones de chequeo, y botones de opción*. Así, la asignación del tipo de objeto para la clase “Button” dependerá de su estilo. Los controles pertenecientes a las clases “ListBox”, “SysListView32”, “SysTreeView32”, y “SysTabControl32” activan rutinas de monitoreo para recuperar el texto asociado a los mismos. En cuanto a la detección de las aplicaciones de tipo “OpusApp” y “Xlmain”, realizan una verificación de la existencia de objeto de tipo aplicación de Word, para el caso de “OpusApp” y objeto de tipo aplicación de Excel para el caso de “Xlmain”; de no existir, estos objetos son creados. El cumplimiento de la condición interna termina luego de extraer la información dependiendo de la clase del OE (conector 3), y no hay otro subproceso que ejecutar, de ser falsa. A continuación, se llega a la tercera condición.

Ahora bien, si la segunda condición no se cumple, significa que la ventana activa es ahora la aplicación (el lector de pantalla). En este se asigna directamente (sin procesos de detección) como OE. Esta asignación sólo se realiza una vez mientras la ventana activa corresponda a la aplicación. El incumplimiento de la segunda condición termina aquí (conector 4). A continuación, se llega a la segunda condición.

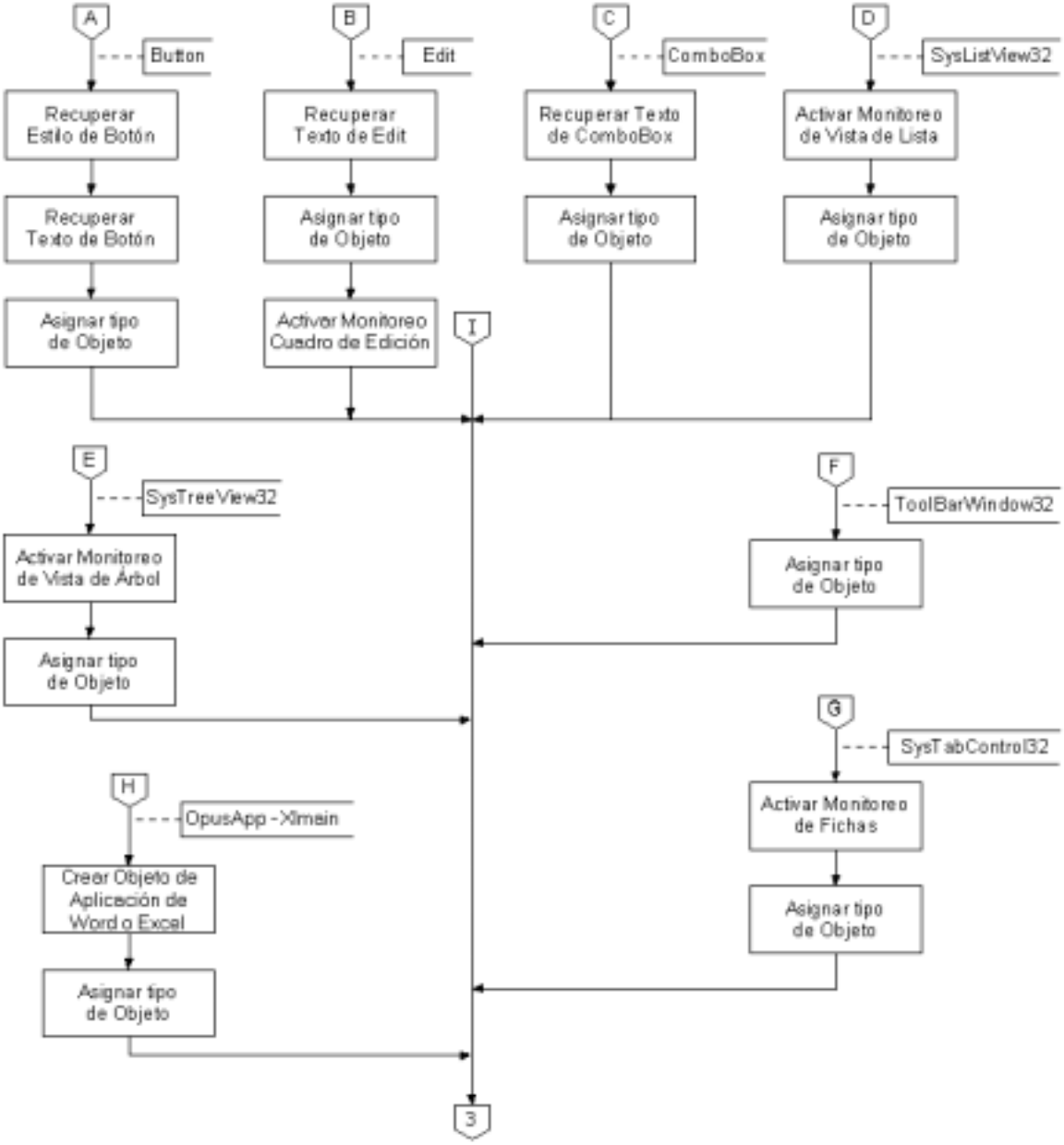

Figura 6.16 Flujograma Tipo de Controles Elección de Controles

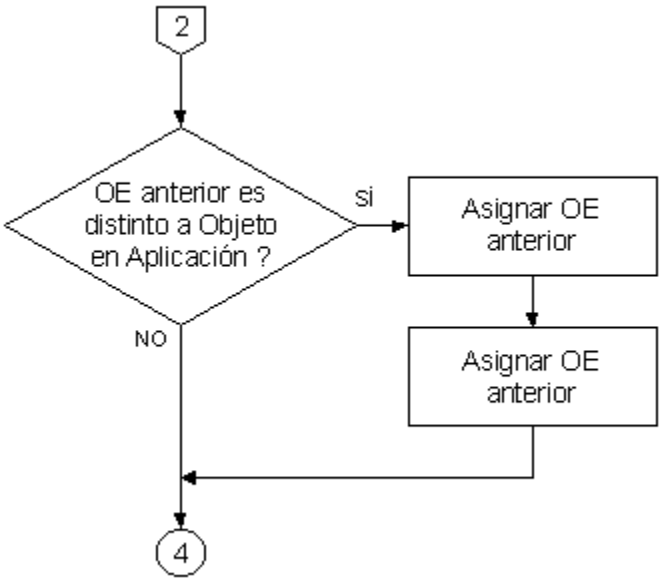

Figura 6.17 Flujograma Asignación Objeto con Enfoque

3. Si la ventana activa (VA) es nueva, lo cual significa que aún no ha sido vocalizado su título, la última tecla presionada, y la información del OE.

   De resultar cierta, debe vocalizarse la última tecla presionada, el título de la ventana, el contenido o texto del OE, su tipo, y su estado. El cumplimiento de la tercera condición termina aquí. Si la tercera condición es falsa, se realizarán otros procesos dependiendo de una nueva condición (interna al incumplimiento de la tercera condición): *Si existe un nuevo OE*. De ser cierta, es necesario vocalizar la última tecla presionada, y la información del OE, dejando fuera la vocalización del título de la ventana, dado que fue realizada inicialmente al cumplirse la tercera condición. Sin embargo, esto no es del todo cierto, debido a que mientras aún se está vocalizando el título de la ventana y el resto, la ventana activa ya no constituye una ventana nueva, y si un nuevo objeto adquiere el enfoque, la vocalización se detiene para dar paso a la nueva información sobre el OE. Para aliviar esta situación, se ha incluido una condición que permite vocalizar una vez más la información completa (la que incluye el título de la ventana) aún después que la ventana activa no sea una nueva ventana. El cumplimiento de la tercera condición (y todo el proceso)

termina aquí. Finalmente resta el incumplimiento de la condición interna al incumplimiento de la tercera condición (si existe un nuevo OE). De ser falsa, únicamente se vocalizarán las teclas que se presionen, siempre y cuando no se esté vocalizando otra información. De esta forma termina el incumplimiento de la tercera condición (y todo el proceso).

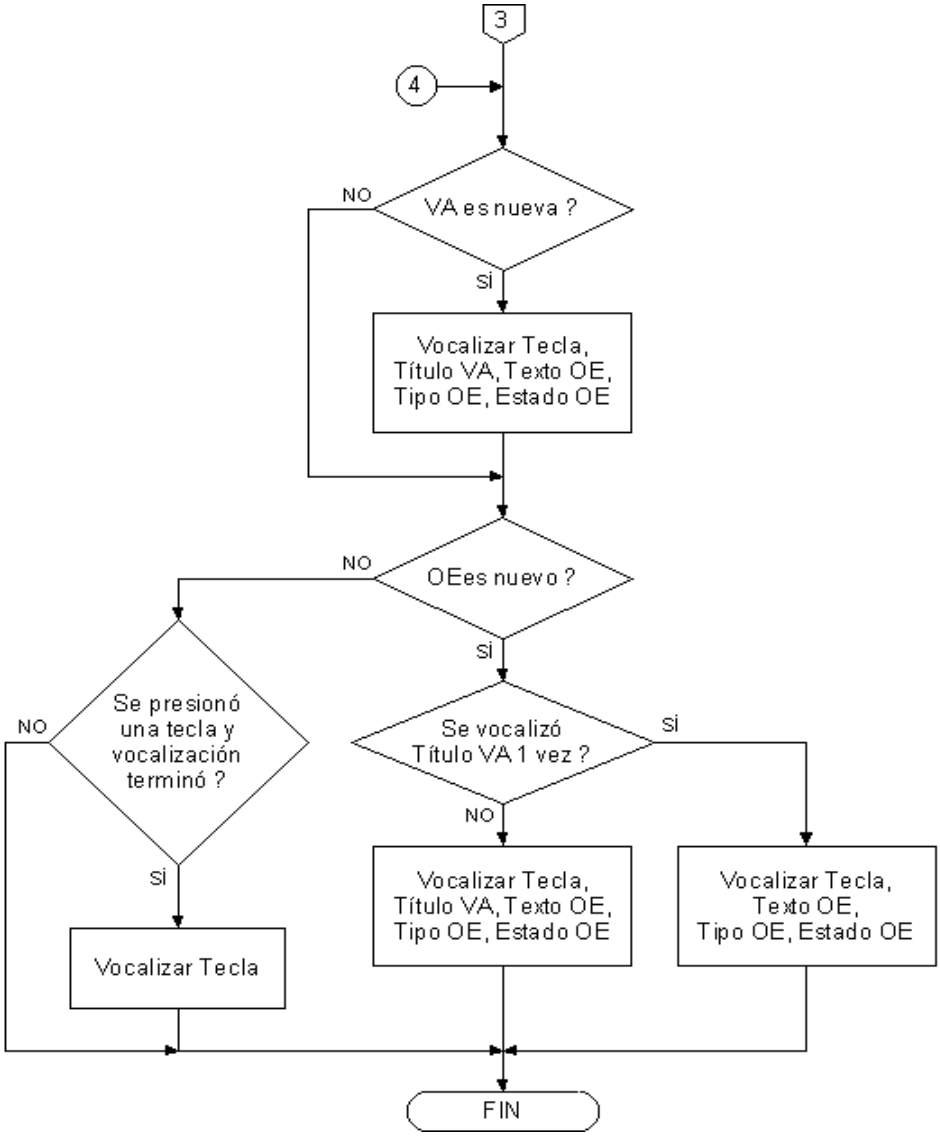

Figura 6.18 Flujograma Finalización y Vocalización de Ventana Activa y Controles con Enfoque

6.4 DESARROLLO DE DETECCIÓN DE OBJETOS ESTÁNDARES

Diagrama de Flujo Cuadros de Combos

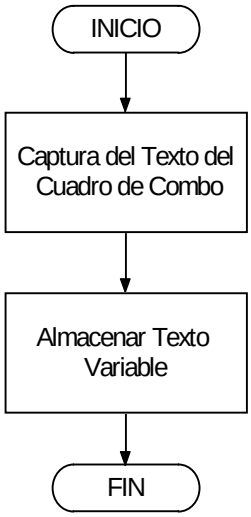

Figura 6.19 Flujograma Combo Box

El diagrama de flujo representa los procesos a seguir para la captura del texto de un control del tipo Cuadro de Combo. Los pasos a seguir son los siguientes:

1. Se obtiene el índice, por medio del mensaje CB_GETCURSEL capturado en el proceso de monitoreo de mensaje a nivel global.
2. Se reserva espacio en una cadena de texto para que almacene el texto extraído del cuadro de combo.
3. Por medio el mensaje WM_GETTEXT se captura la longitud del texto del control.
4. Se asigna el texto recuperado a una variable para que posteriormente sea utilizado en la ejecución de otro proceso.

Diagrama de Flujo Cuadro de Edición

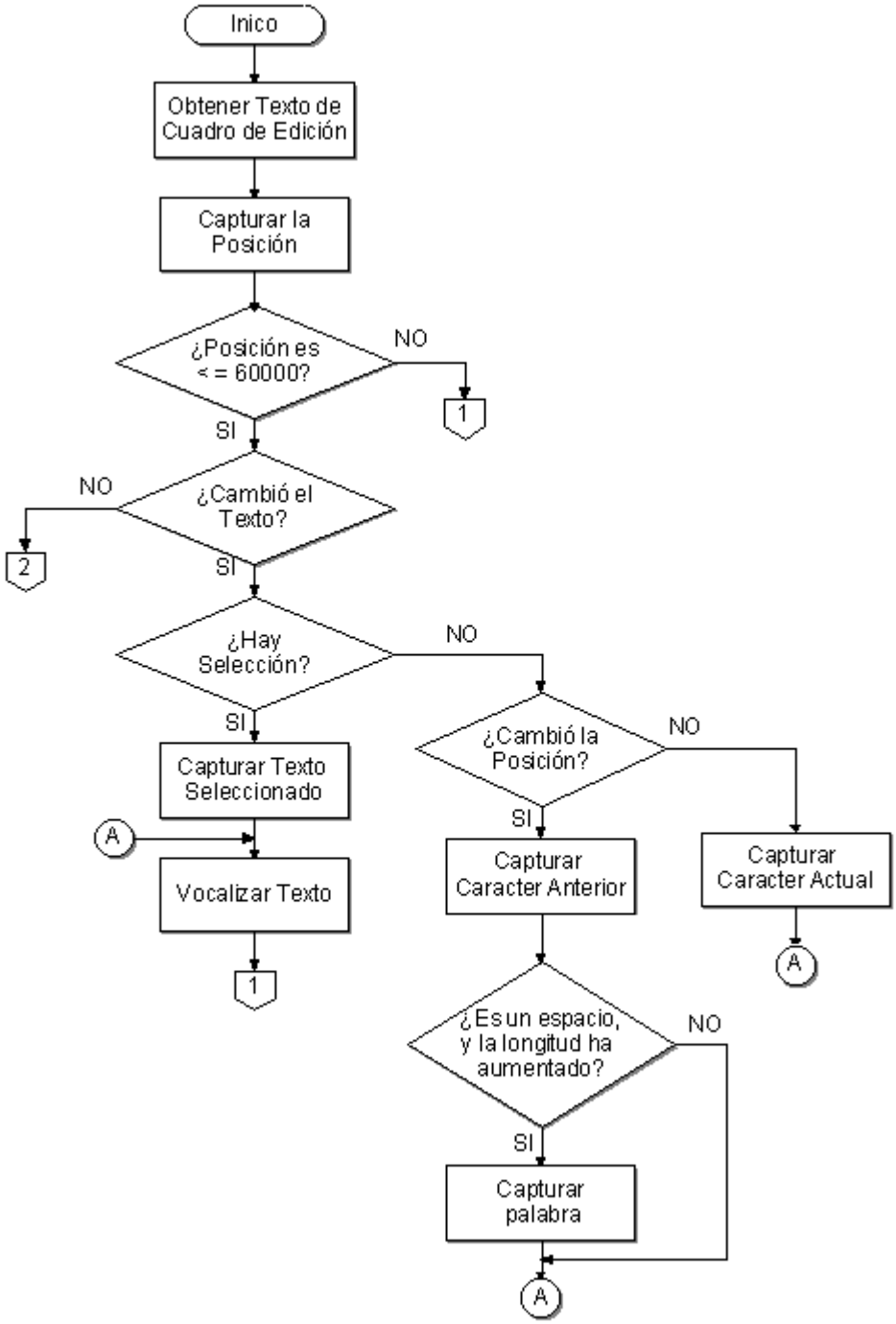

Figura 6.20 Flujograma Cuadro de Edición (A)

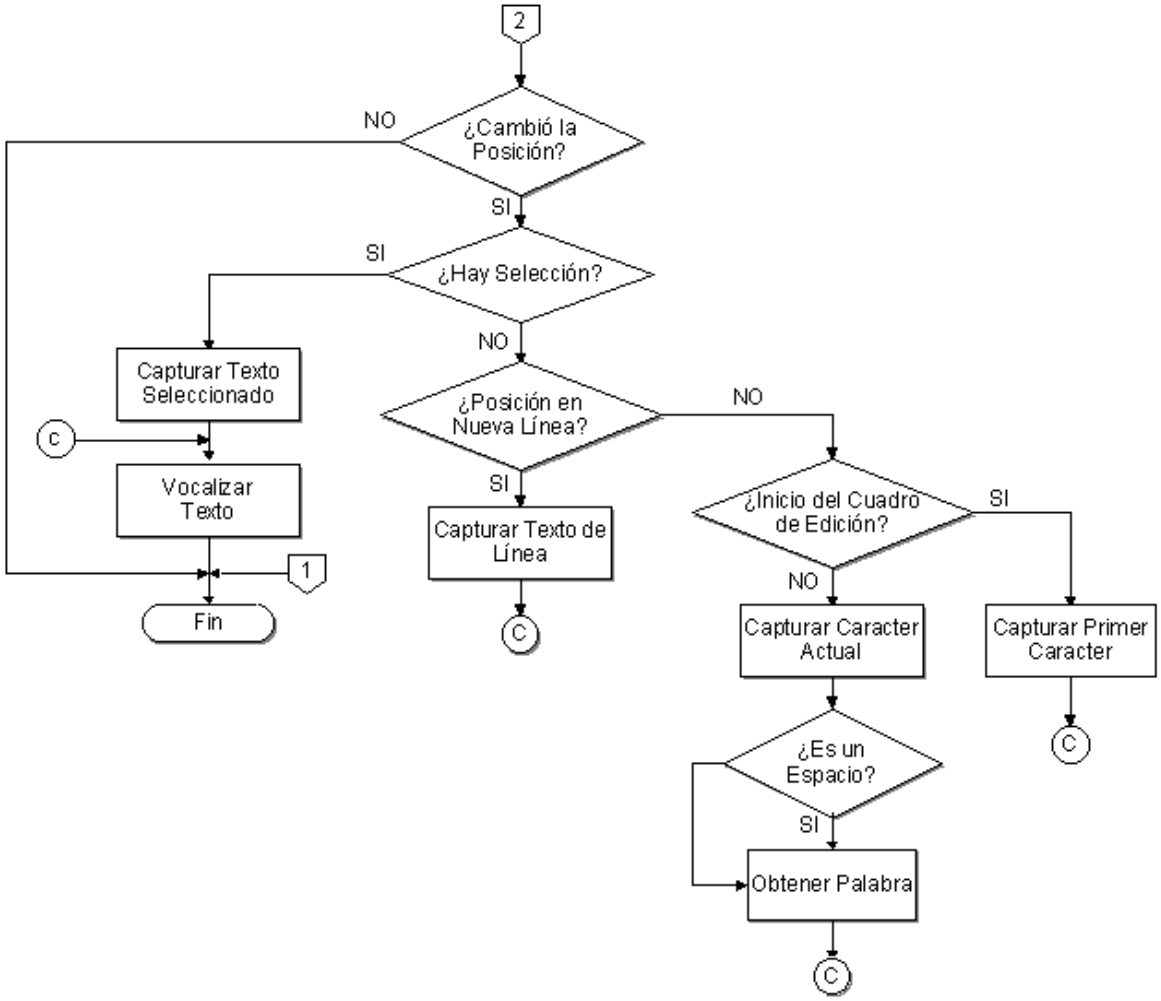

Figura 6.21 Flujograma Cuadro de Edición (B)

El diagrama de flujo representa los procesos a seguir para la captura y la interacción del texto en un Cuadro de Edición. Los pasos a seguir son los siguientes:

1. Este proceso inicia obteniendo el texto del cuadro de edición.
2. Posteriomente se captura la posición del cursor dentro del control y se realiza una verificación para no sobrepasar la cantidad máxima de texto que podrá ser detectada y vocalizada.
3. Si la posición del texto sobrepasa los 60,000 caracteres, dicho texto no será detectado. Si se está dentro de los límites, se verifica si el texto ha sido modificado o simplemente ha ocurrido un cambio de posición dentro del control.

4. Un cambio en el texto junto con un cambio en la posición, pueden implicar que se escribió una letra, en cuyo caso debe ser vocalizada, o se borró una letra (con backspace), en cuyo caso se vocaliza la letra que antecede al cursor. Si el texto cambia y no hay cambio de posición, una letra ha sido borrada (con delete) por lo que se vocalizará la nueva letra que ocupa la posición actual del cursor.
5. Durante un cambio de texto junto con un cambio de posición, si ocurre que la longitud del texto ha aumentado y el caracter anterior es un espacio, se busca leer la palabra que ha sido digitada.
6. Si no ocurre cambio en el texto, se verifican cambios en la posición. Si la posición del cursor cambia de una línea a otra, la nueva línea es vocalizada. Si se está en la misma línea, se leerán las letras a medida que el cursor se desplaza por las mismas. Al encontrar que el caracter anterior a una letra es un espacio, se lee una palabra.
7. En los casos en que se selecciona texto, las letras o palabras contenidas en dicha selección, constituyen el texto a vocalizar.

Diagrama de Flujo Vista de Listas

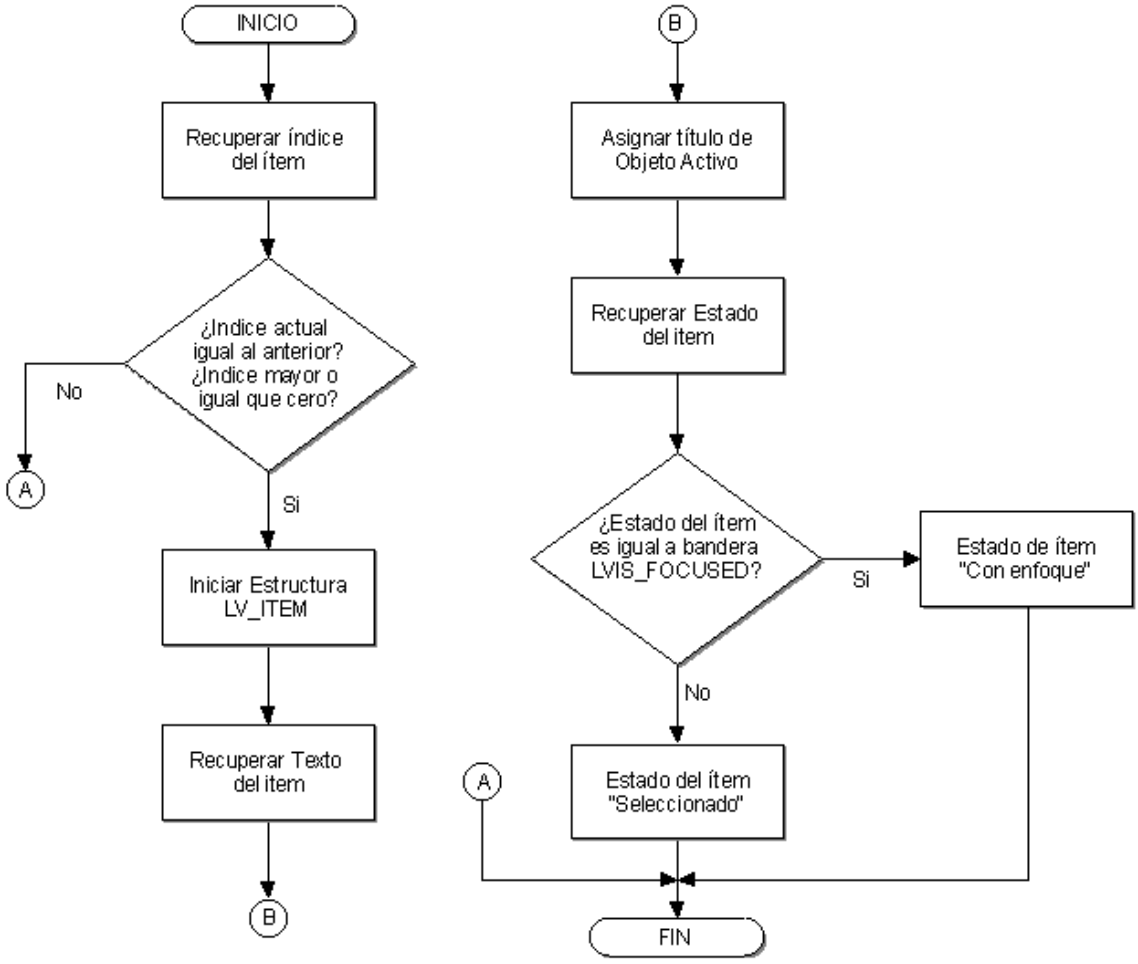

Figura 6.22: Flujograma Vista de Lista

El diagrama de flujo representa la serie de pasos a seguir para la captura del texto y estado de un objeto dentro de una vista de lista.

Los procesos a seguir son los siguientes:

1. Capturar el índice del ítem a través del mensaje LVM_GETSELECTIONMARK.
   Condición: Si el índice anterior es diferente al actual y el índice es mayor o igual que cero:
   - Si no se cumple la condición, termina el proceso.
   - Si se cumple, se ejecutan los siguientes procesos:
2. La inicialización de la estructura LV_ITEM consiste en asignar un valor determinado a la variable que almacenará la longitud del buffer que contendrá

el texto del ítem seleccionado. Luego se captura la longitud de la estructura LV_ITEM, se le asigna al miembro mask de esta estructura la bandera LVIF_TEXT que indica la captura del texto, se le asigna al miembro ítem de la estructura el valor del índice cuya información será recuperada.

3. La captura del texto del ítem requiere de un puntero a una cadena de texto y un puntero a la estructura LV_ITEM. Los punteros son obtenidos por medio de la creación de archivos mapeados en memoria.
4. Luego se envía un mensaje LVM_GETITEMTEXT que será el encargado de recuperar el texto del ítem.
5. Almacenar el texto extraído en una variable, para su posterior utilización.
6. Se recupera el estado del ítem, a través del envío del mensaje LVM_GETITEMSTATE.
7. Si lo que retorno el mensaje es igual a la bandera de estado LVIS_FOCUSED, se le asigna al ítem el estado “Con Enfoque”, de lo contrario el ítem posee el estado “Seleccionado”.

Diagrama de Flujo Vista de Árbol

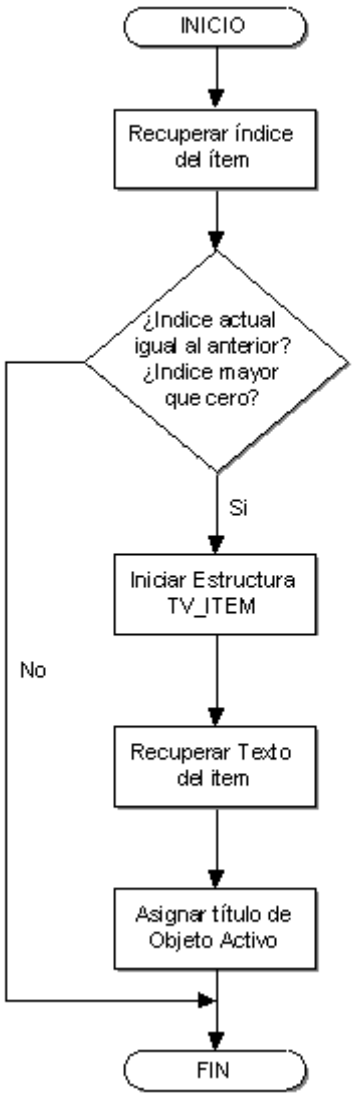

Figura 6.23: Flujograma Vista de Árbol

El diagrama de flujo representa la serie de pasos a seguir para la captura de un objeto dentro de una vista de árbol.

Los procesos a seguir son los siguientes:

1. Capturar el índice del ítem a través del mensaje TVM_GETNETXTITEM.

Condición: Si el índice anterior es diferente al actual y el índice es mayor que cero:

- Si no se cumple la condición, termina el proceso.
- Si se cumple, se ejecutan los siguientes procesos:

2. La inicialización de la estructura TV_ITEM consiste en asignar un valor determinado a la variable que almacenara la longitud del buffer que contendrá el texto del ítem seleccionado. Luego se captura la longitud de la estructura TV_ITEM, se le asigna al miembro mask de esta estructura la bandera TVIF_TEXT que indica la captura del texto, se le asigna al miembro ítem de la estructura el valor del índice cuya información será recuperada.
3. La captura del texto del ítem requiere de un puntero a una cadena de texto y un puntero a la estructura TV_ITEM. Los punteros son obtenidos por medio de la creación de archivos mapeados en memoria.
4. Luego se envía un mensaje TVM_GETITEM que será el encargado de recuperar el texto del ítem.
5. Se almacena el Texto extraído en una variable, para su posterior utilización.

Diagrama de Flujo Ficheros

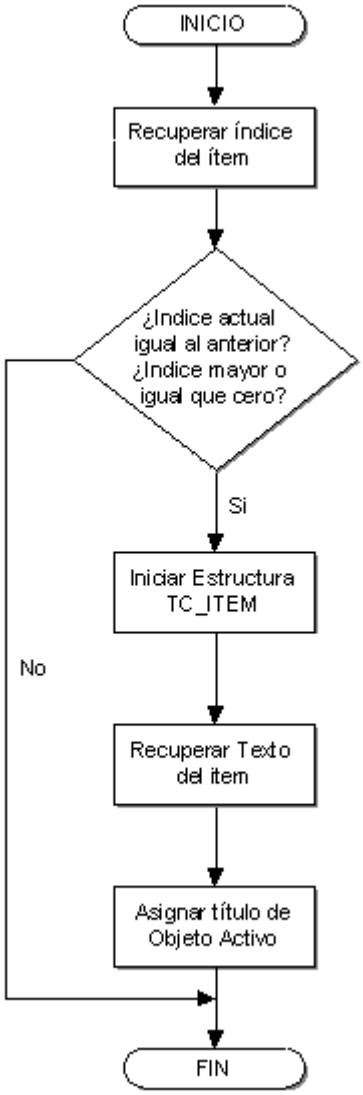

Figura 6.24: Flujograma Fichero

El diagrama de flujo representa la serie de pasos a seguir para capturar el titulo de un control de tipo Fichero.

Los procesos a seguir son los siguientes:

1. Capturar el índice del ítem a través del mensaje TCM_GETCURFOCUS.

Condición: Si el índice anterior es diferente al actual y el índice es mayor o igual que cero:

- Si no se cumple la condición, termina el proceso.
- Si se cumple, se ejecutan los siguientes procesos:

2. La inicialización de la estructura TC_ITEM consiste en asignar un valor determinado a la variable que almacenara la longitud del buffer que contendrá el titulo del fichero.

Luego se captura la longitud de la estructura TC_ITEM, se le asigna al miembro mask de esta estructura la bandera TCIF_TEXT que indica la captura del texto, se le asigna al miembro ítem de la estructura el valor del índice cuya información será recuperada.

3. La captura del titulo de la ficha requiere de un puntero a una cadena de texto y un puntero a la estructura TC_ITEM. Los punteros son obtenidos por medio de la creación de archivos mapeados en memoria.
4. Luego se envía un mensaje TCM_GETITEMA que será el encargado de recuperar el titulo de la ficha.
5. Se almacena el Texto extraído en una variable, para su posterior utilización.

Diagrama de Flujo Barras de Herramientas

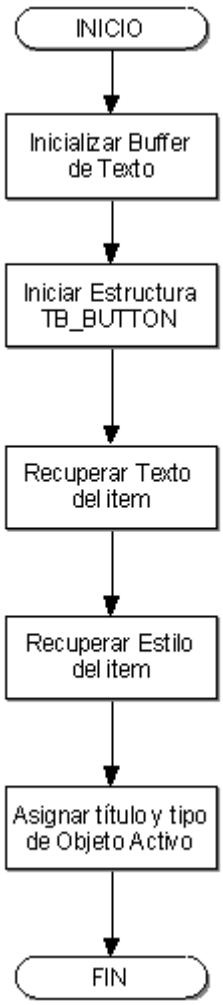

Figura 6.25: Flujograma Barra de Herramientas

El diagrama de flujo muestra la serie de pasos a seguir para recuperar el texto y el estilo de un objeto perteneciente a una barra de herramientas.

Nota: El índice del botón y el controlador de la barra de herramientas se obtienen por medio del mensaje TB_COMMANDTOINDEX capturado en el proceso de monitoreo de mensaje a nivel global.

1. Se asigna espacio al buffer que almacenará el texto del botón.
2. La estructura TB_BUTTON es utilizada para recuperar el estilo del botón de la barra de herramientas, su inicialización consiste en asignar al miembro ítem de la estructura el valor del índice cuyo estilo será recuperado. Se captura la longitud de la estructura TB_BUTTON.
3. La captura del texto se realiza a través del mensaje TB_GETBOTTONTEXTA, esta captura requiere de un puntero a una cadena de texto. El puntero es obtenido por medio de la creación de archivos mapeados en memoria.
4. Para recuperar la información contenida en la estructura TB_BUTTON, es necesario enviar el mensaje TB_GETHOTITEM, el cual requiere de un puntero a dicha estructura, y este puntero es obtenido por medio de la creación de archivos mapeados en memoria.
5. Se almacena el texto del botón en variable.
6. El estilo del ítem de la barra de herramientas se obtiene del miembro fsStyle de la estructura TB_BUTTON, y es almacenado en una variable.

Diagrama de Flujo Cuadro de Lista

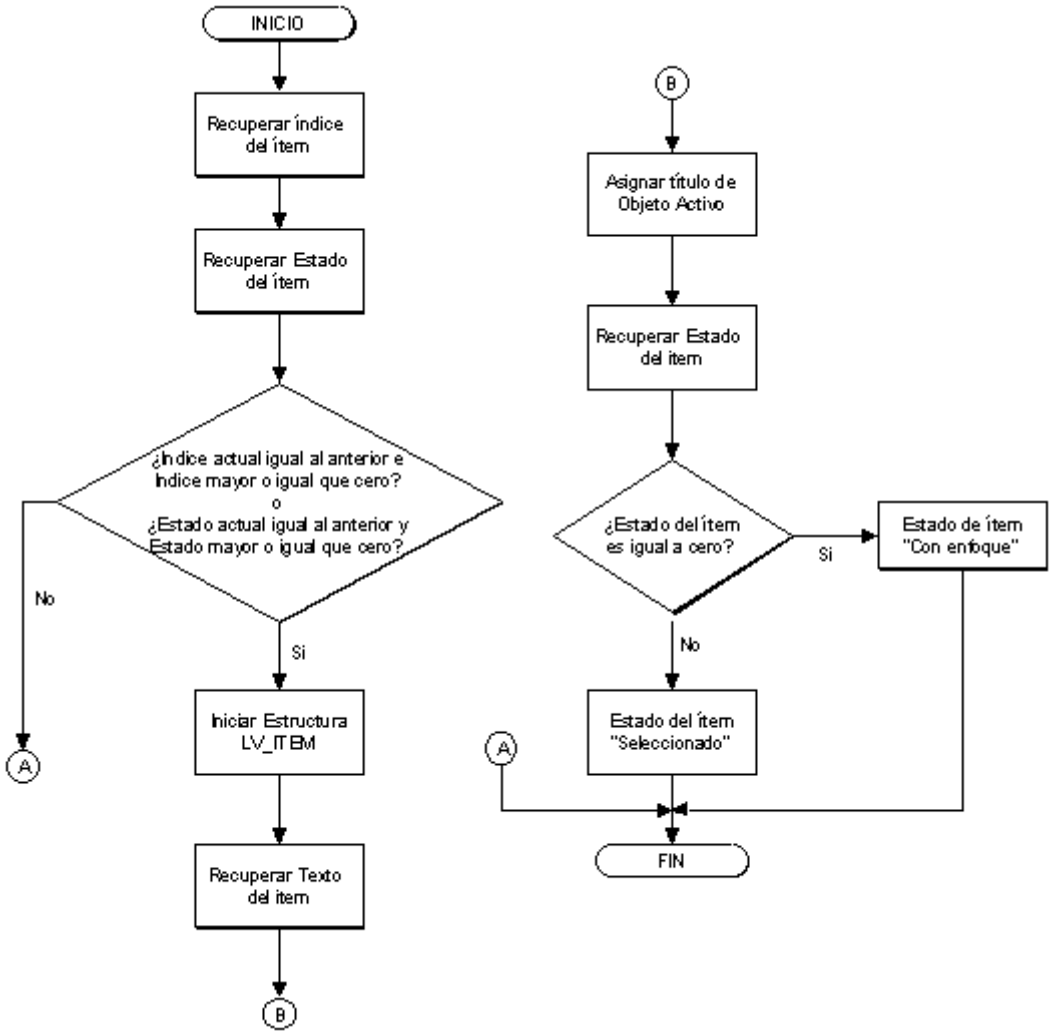

Figura 6.26: Flujograma Cuadro de Lista

El diagrama de flujo representa la serie de pasos a seguir para capturar el texto y estado de un objeto dentro de un cuadro de lista.

Los procesos a seguir son los siguientes:

1. Se captura el índice del ítem a través del mensaje LB_GETCARETINDEX y se obtiene el estado del ítem por medio del mensaje LB_GETSEL.

   Condición: Si el índice anterior es diferente al actual y el índice es mayor o igual que cero; o el estado del índice es diferente al estado actual y el estado es mayor o igual que cero:

   - Si no se cumple la condición, termina el proceso.
   - Si se cumple, se ejecutan los siguientes procesos:

2. Se asigna espacio al buffer que almacenará el texto del botón.
3. La captura del texto del ítem requiere de un puntero a una cadena de texto. El puntero es obtenido por medio de la creación de archivos mapeados en memoria.
4. Luego se envía un mensaje LB_GETTEXT que será el encargado de recuperar el texto del ítem.
5. Se almacena el texto extraído en una variable, para su posterior utilización.
6. Si lo que retornó el mensaje LB_GETSEL es igual a cero se le asigna al ítem el estado “Con Enfoque”, de lo contrario el ítem posee el estado “Seleccionado”.

Diagrama de Flujo Barra de Menú Estándar

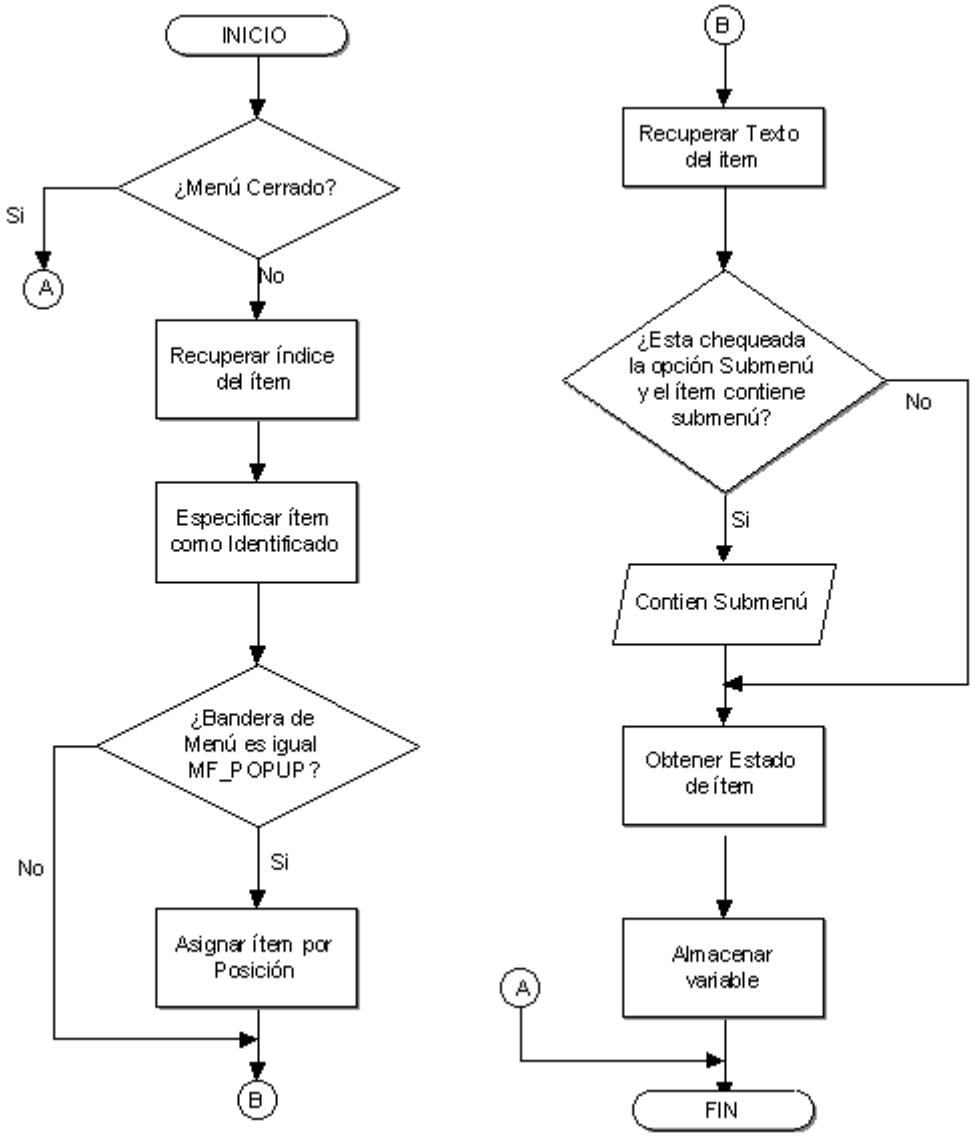

Figura 6.27 Flujograma de Barra de Menú Estándar

El diagrama de flujo representa la serie de pasos a seguir para recuperar el texto y estado de un ítem de menú seleccionado.

Nota:

El índice y el tipo del ítem de menú seleccionado se obtienen por medio del mensaje WM_MENUSELECT capturado en el proceso de monitoreo de mensaje a nivel global.

1. Se verifica el estado de la Barra de Menú:

Condición: Si el estado es inactivo, es decir, el menú esta cerrado:

- Si se cumple la condición, termina el proceso.
- Si no se cumple, se ejecutan los siguientes procesos:

2. Se recupera el índice del ítem de menú seleccionado.
3. Se especifica el ítem de menú como Identificador.
4. Se verifica si el ítem posee submenú por medio de la bandera MF_POPUP, si la bandera es de este tipo se especifica el ítem como Posición.
5. El texto del ítem de menú es recuperado a través de la función GetMenuString y almacenado en una variable.
6. Si la opción de verificar Submenú del menú configuración del lector de pantalla está habilitada, y el ítem posee la bandera MF_FLAG, se vocaliza que el ítem posee Submenú.
7. El estado del ítem es obtenido a través de la verificación de las banderas de menú y se almacena en una variable.

6.5 DESARROLLO DE DETECCIÓN DE TECLAS

- **Diagrama de Flujo**

Figura 6.28 Flujograma Detección de Teclas

El diagrama de flujo muestra la serie de pasos a seguir para recuperar la tecla presionada.

1. Se obtiene el código de la tecla presionada por medio del mensaje WM_KEYDOWN capturado en el proceso de monitoreo de mensaje a nivel global.

2. Se compara el código de la tecla obtenida con los valores predefinidos de las teclas virtuales.

Si una de las teclas virtuales coincide con las teclas presionadas, se almacena la pronunciación de la tecla en una variable.

3. Se vocaliza la pronunciación de la tecla.

6.6 DESARROLLO DE LAS FUNCIONES DE CAPTURA DE MENSAJES

CWPFunction

A través de esta función se reciben los mensajes que son enviados a las aplicaciones utilizando la función SendMessage. Algunos de estos mensajes transportan información concerniente a distintos controles en las aplicaciones. Aquellos que son de interés son escogidos dentro de una sentencia de selección de casos, y afectan a un tipo de control específico. A continuación se presentan los mensajes que se desea capturar:

- **WM_CREATE**: Notifica cuando una aplicación solicita la creación de una nueva ventana. Si la clase de la aplicación es de tipo ExploreWclass, dmgServerClass, CabinetWclass, #32770, Shelldll_Defview, OpusApp y Xlmain se desinstala el Hook Global y se deshabilita la Ventana Activa. Para el caso de las ventanas de clase OpusApp y Xlmain, además de realizar las desinstalación del Hook Global y la deshabilitación de la Ventana Activa, se habilitan las rutinas de creación de objetos para cada una de estas clases.

- **BM_SETCHECK, BM_SETSTATE**: Notifica cuando se ha detectado un cambio en el estado de un control de tipo Botón, activando la rutina de monitoreo de ventana activa.

- **CB_GETCURSEL**: Notifica cuando un nuevo ítem de un ComboBox ha sido seleccionado, activando la rutina de monitoreo de este control. Se recupera su controlador (hwnd) para acceder posteriormente al texto del ítem.

- **LB_GETCURSEL**: Notifica cuando un nuevo ítem de cuadro de lista posee el foco o ha sido seleccionado, activando la rutina de monitoreo de este control. Se recupera su controlador (hwnd) para acceder posteriormente ítem del cuadro de lista.

- **WM_MENUSELECT**: Notifica cuando un nuevo menú se ha desplegado, o un ítem de menú ha sido seleccionado, realizando un llamado a la función DetectarMenú. Se recupera su controlador (lParam) para acceder posteriormente al texto de la ficha, y el identificador del ítem de menú (Byte bajo de wParam).

- **WM_SYSCOMMAND, WM_ENTERIDLE**: Notifican cuando en las ventanas pertenecientes a Excel y Word se ha desplegado el menú principal, o un ítem de menú ha sido seleccionado, activando la rutina para el monitoreo de este control.

- **TB_COMMANDTOINDEX**: Notifica cuando un nuevo botón de la ToolBar ha sido seleccionado, activando la rutina de monitoreo de este control. Se recupera su controlador (hwnd) para acceder posteriormente al texto de la ficha, y el identificador del botón (wParam).

- **TB_SETANCHORHIGHLIGHT**: Notifica cuando un control de tipo Barra de Herramientas ha perdido el enfoque.

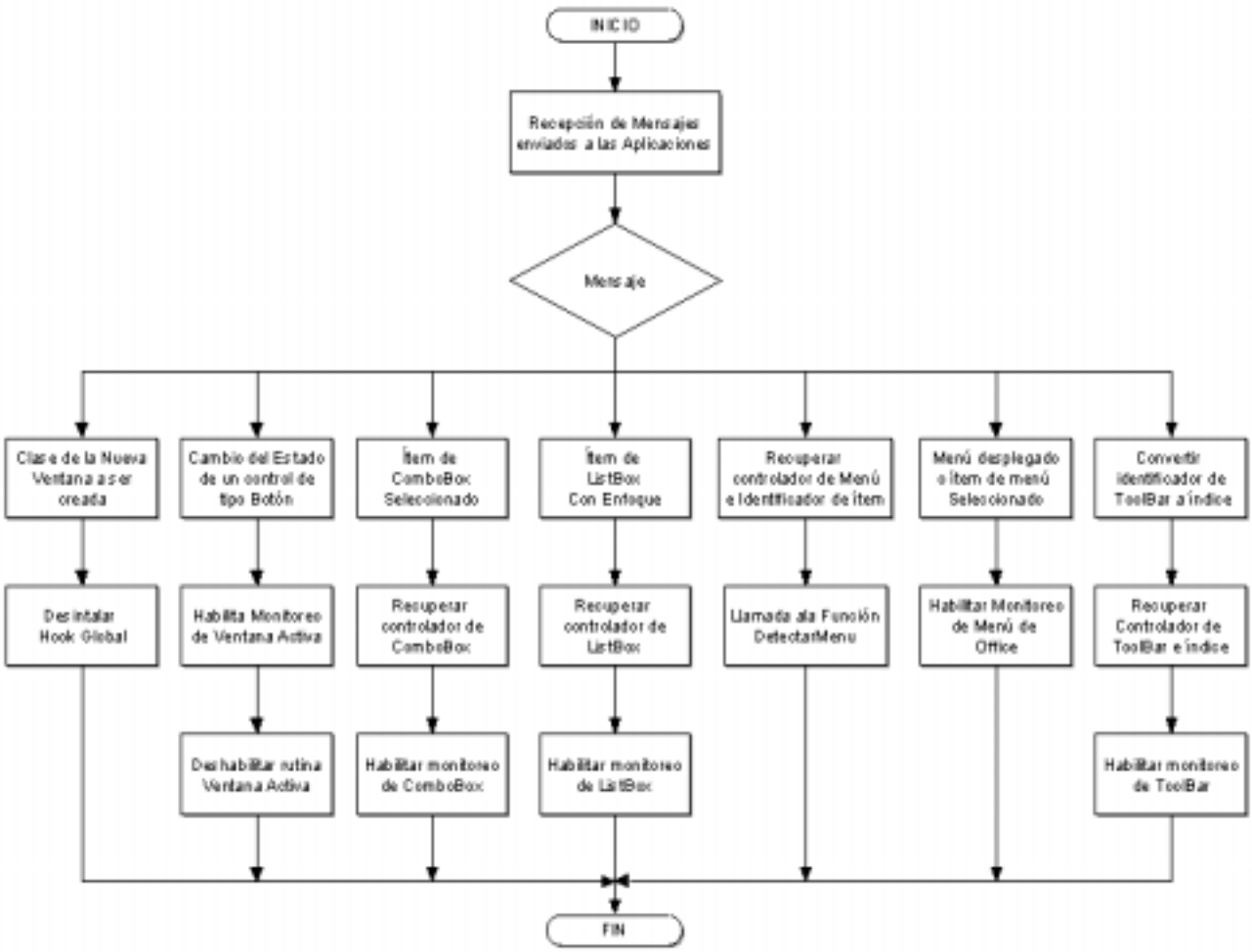

Figura 6.29: Flujograma Función CWPFunction

MSGFunction

A través de esta función se reciben los mensajes que son extraídos de la cola de mensajes de las aplicaciones utilizando la función GetMessage o PeakMessage. En esta función se recupera el mensaje WM_KEYDOWN, mediante el cual se notifica cuando una tecla ha sido presionada, activando la rutina de monitoreo de teclas. Se recupera el código de tecla virtual (wParam) para acceder posteriormente al texto del ítem.

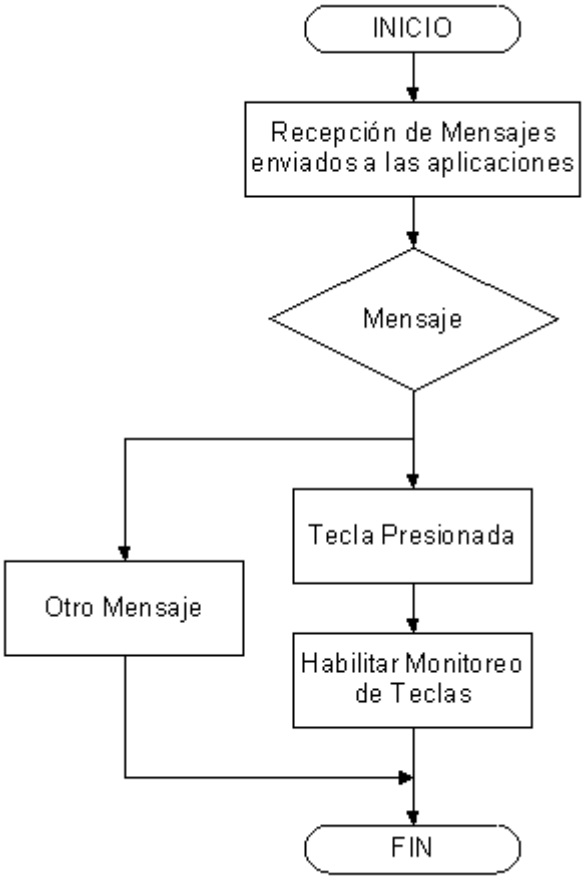

Figura 6.30: Flujograma Función MSGFunction

6.7 DESARROLLO DE DETECCIÓN DE OBJETOS ESPECÍFICOS

Procedimiento Detección de Ítem activo de la Barra de Menú de Office

Para recuperar el texto y estado de un ítem de menú de la barra de menú de las aplicaciones que forman parte de la Suit de Office se realizaron los siguientes procesos:

Nota:

A través de los mensajes WM_SYSCOMMAND y WM_ENTERIDLE capturados en el proceso de monitoreo a nivel global, se notifica cuando en las ventanas pertenecientes a Excel y Word se ha desplegado el menú principal, o un ítem de menú ha sido seleccionado, activando la rutina para el monitoreo del ítem de menú seleccionado.

1. Dependiendo de la aplicación en la que se detecte, ya sea, el mensaje WM_SYSCOMMAND o WM_ENTERIDLE se crea el objeto CommandBar específico para el menú principal de dicha aplicación.
2. En un bucle se inicia la búsqueda, control por control, del control activo de tipo CommandBarPopup de la CommandBar específica.
3. **Condición:**

   Se compara la propiedad accState de cada control con un valor de Bandera que indique si dicho control posee el estado de Seleccionado. Los controles en este nivel son los ítems principales del menú, los cuales tienen a su cargo otros ítems de menú:

   - Si la propiedad accState del control no coincide con la Bandera, el bucle continúa con el siguiente control.
   - Si la propiedad accState del control coincide con la Bandera, se realizan los siguientes pasos:
4. Al encontrar un control con el estado de Seleccionado, se llama a la función DetectarItemOM, a la cual se le da como parámetro el control activo de tipo CommandBarPopup seleccionado.
5. Esta función inicia a la vez un bucle que tiene como objetivo buscar, control por control, el control de tipo CommandBarControl del CommandBarPopup enviado como parámetro. Luego se inician de nuevo los procesos a partir del

paso No. 3. hasta que se encuentre un ítem de menú con estado Seleccionado, y se continúa con los procesos siguientes.

Si el bucle finaliza y no se encontró ningún control de tipo CommadBarControl con estado de Seleccionado, es el control de tipo CommandBarPopup el que se convierte en el ítem de menú seleccionado.

6. Una vez encontrado el ítem de menú con estado de Seleccionado dentro del objeto CommandBar, se obliga la salida del bucle de búsqueda del control activo.
7. **Condición:**
   - Si el ítem de menú activo es diferente al ítem de menú previo, el ítem de menú activo se convierte en el Objeto con Enfoque y su texto y estado son almacenados en variables.
8. Se destruyen los objetos de tipo CommandBar, CommandBarPopup y CommandBarControl utilizados.

Procedimiento Detección Objeto Word Aplication

Para la interacción y recuperación del texto en un documento de Word se realizaron los siguientes procesos:

Nota:

La creación de un Objeto de tipo Aplicación de Word puede ser realizado por medio de dos procesos:

- A través del temporizador WDCreate que es habilitado al recibir un mensaje WM_CREATE capturado en el proceso de monitoreo a nivel global, en el cual se notifica de la creación de una nueva ventana de clase OpusApp (aplicación de Word).
- Por medio del monitoreo de la Ventana Activa, cuando se detecta que el Objeto con Enfoque es de clase _Wwg o _Wwf (Objeto Documento de Word)

8. Una vez creado el objeto Aplicación de Word, se procede a recuperar la longitud del texto del documento. El texto del documento es provisto por la propiedad ActiveDocument.Content.Text.

9. **Condición:**
   - Si la longitud del texto sobrepasa los 60,000 caracteres, el texto del documento es recortado a 60,000 caracteres, y luego es vocalizado por el sintetizador de voz. La condición de recortar el texto se incluye para no sobrepasar el límite del texto que puede ser vocalizado por el sitentizador de voz
   - Si la longitud del texto es menor a los 60,000 caracteres, el texto del documento es vocalizado por el sintetizador de voz.
10. La interacción con el contenido del documento (es decir, el reconocimiento de palabras, caracteres y texto seleccionado) es realizada cuando ocurre el evento WindowSelectionChange del objeto Word Aplication, el cual se produce cuando cambia la selección en la ventana del documento activo.
11. El objeto Aplicación de Word es destruido cuando ocurre el evento Quit, el cual se produce cuando se cierra la ventana de la aplicación.

Procedimiento Detección Objeto Excel Aplication

Para la detección y recuperación del texto en la celda activa en una hoja de cálculo de Excel se realizaron los siguientes procesos:

Nota:

La creación de un Objeto de tipo Aplicación de Excel puede ser realizado por medio de dos procesos:

- A través del temporizador XLCreate que es habilitado al recibir un mensaje WM_CREATE capturado en el proceso de monitoreo a nivel global, en el cual se notifica de la creación de una nueva ventana de clase Xlmain (aplicación de Excel).
- Por medio del monitoreo de la Ventana Activa, cuando se detecta que el Objeto con Enfoque es de clase Excel7 o Xlmain (Objeto WorkBook de Excel)

1. Una vez creado el objeto Aplicación de Excel, se procede a recuperar la dirección y el texto de la celda activa. La dirección de la celda es provista por

la propiedad ActiveCell.Address, y el texto por la propiedad ActiveCell.Text.

2. La recuperación de la celda activa o del rango de celdas seleccionadas es realizada cuando ocurre el evento SheetSelectionChange del objeto Aplicación de Excel, el cual se produce cuando la selección cambia en una hoja (no se produce si la selección está en una hoja de gráfico).
3. El objeto Aplicación de Excel es destruido cuando ocurre el evento WorkbookDeactivate, el cual se produce antes de desactivar cualquier libro de Excel abierto.

IX. CONCLUSIONES

- En el mundo de la informática, los programas de computadora conocidos como *lectores de pantalla*, constituyen la principal herramienta de accesibilidad para las personas con discapacidades visuales. Mediante estos programas se lleva a cabo la sustitución de la información visual provista en la pantalla, por información que puede ser captada por el sentido del oído o del tacto, haciendo posible la utilización de las computadoras por parte de dichas personas.

- A pesar de que algunos lectores de pantalla permiten utilizar dispositivos adicionales a los incluidos en la computadora, tales como las *matrices braile*, para presentar la información contenida en pantalla, todos en general recurren a la síntesis de voz para hacer llegar al usuario no vidente dicha información, puesto que a diferencia de estos dispositivos, se trata de una tecnología de bajo costo que sustituye eficientemente el sentido de la vista por el sentido del oído.

- La mayor limitante que existe para la población no vidente de El Salvador, en cuanto al aprovechamiento de los beneficios que trae consigo la utilización de los lectores de pantalla, la constituye la adquisición de aquellos programas que proporcionan mayores facilidades al usuario no vidente para la utilización de la computadora, debido a la relación directa que existe entre su funcionalidad y su precio. Por otro lado, la falta de apoyo legislativo por parte del estado, limita aún más las posibles oportunidades y beneficios que el uso de estas tecnologías de accesibilidad proporcionarían a la población no vidente.

- El prototipo desarrollado en esta investigación contempló la construcción de un modelo básico que incluye algunas de las características que debe poseer todo lector de pantalla para llegar a ser un sistema totalmente funcional que

pueda ser implementado para su utilización por parte de la comunidad no vidente. Su mayor aporte consiste en el establecimiento de un precedente para el estudio y desarrollo de este tipo de herramientas de accesibilidad, y en servir de incentivo para la realización de otros proyectos destinados al beneficio de la comunidad de discapacitados en general.

- El prototipo realiza el reconocimiento de objetos, y la extracción del texto contenido en los mismos, a través de funciones proporcionadas por la interfaz de programación de aplicaciones de Windows, y utiliza un sintetizador de voz que cuenta con dos voces en español de distinto género, mediante las cuales se transmite la información detectada. Estas características permiten, respectivamente, que las técnicas utilizadas para el desarrollo del programa sean aplicables a otras plataformas de Windows, en las que también se cuenta con una interfaz de programación de aplicaciones, y que la información transmitida sea comprendida con mayor facilidad.

- El empleo del sistema de mensajes de Windows como método para acceder a distintos aspectos de la información relacionada a los objetos que componen la interfaz gráfica de usuario, tiene la ventaja de proporcionar una visión general de los fundamentos sobre los que se basa su comportamiento bajo el sistema operativo. De esta forma, es posible considerar con mayor amplitud distintas técnicas para el desarrollo de una herramienta como el lector de pantalla, sin limitarse a la información parcial brindada por métodos como el reconocimiento gráfico.

- En general, la capacidad de un lector de pantalla para detectar la información correspondiente a un objeto presente en pantalla, dependerá de los métodos provistos para accederlos. Los objetos estándar de Windows pueden ser detectados gracias a que el sistema operativo provee dichos métodos. En el caso de objetos definidos por la aplicación específica a la que pertenecen, el acceso a sus características es posible mientras la aplicación provea los métodos para realizarlo. Es por ello que la inclusión de objetos, en las aplicaciones, cuya información pueda ser reconocida por las herramientas de

accesibilidad, figura entre los principios básicos a seguir para el diseño de aplicaciones accesibles.

- El prototipo de pantalla desarrollado, superó algunos de los alcances propuestos originalmente para el mismo. Entre ellos se encuentran, la interacción que provee con el texto de la aplicación NotePad, para la cual se había propuesto únicamente la lectura del archivo; la interacción que provee con el texto de Microsoft Word, para el cual también se propuso únicamente la lectura del archivo; y la lectura de las barras de menús de Microsoft Word, y Microsoft Excel, las cuales no habían sido contempladas dentro de los alcances.

- Para que un programa lector de pantalla constituya una herramienta mediante la cual, una persona con discapacidad visual pueda hacer uso de una computadora, debe incluir como características generales: *brindar al usuario una guía coherente que le permita conocer la aplicación que recibirá las acciones ejecutadas, así como los distintos elementos que componen la aplicación, a medida que este se desplaza por la misma; informar al usuario sobre las acciones que ha ejecutado a través del teclado, y cuando fuese posible, las acciones que tendrían lugar al presionar determinadas teclas; brindar información que a pesar de no formar parte del texto escrito en la pantalla, complemente la comprensión de las acciones a realizar; y finalmente mantener una vía de comunicación entre el usuario y la herramienta, la cual debe proporcionar opciones para su propia configuración, así como soporte adicional que en general, facilite la utilización de la computadora y sus programas.*

X. RECOMENDACIONES

- Es importante para el desarrollo integral de la población Salvadoreña, incluir dentro de los planes de estudio superior, la promoción de investigaciones y proyectos de desarrollo dirigidos al mejoramiento de las condiciones de vida de la población discapacitada. Al igual, resulta importante promover el seguimiento de estas investigaciones y proyectos, por medio de estudios que estén enfocados a la búsqueda de elementos y métodos que mejoren la funcionalidad del proyecto original.

- La promoción de investigaciones en el área de desarrollo tecnológico puede ser enfocada, por ejemplo, al complemento de sistemas y programas que no incluyen medidas de accesibilidad, así como también al desarrollo local y al complemento y / o la mejora de la funcionalidad de herramientas ya existentes en el mercado, pero que no satisfacen completamente las necesidades de los usuarios con algún tipo de discapacidad.

- Es fundamental que dentro de los procesos de enseñanza orientados al desarrollo de sistemas se haga énfasis en la importancia de hacer aplicaciones “accesibles”, es decir tomar en cuenta las necesidades de las personas discapacitadas, utilizando las distintas normas de accesibilidad en los programas desarrollados, e incluyendo objetos cuya información pueda ser reconocida por las herramientas de accesibilidad. De esta forma se contribuirá a la generación de condiciones informáticas que faciliten el desarrollo social, laboral, e intelectual de la población de discapacitados en el país.

- En futuras mejoras del prototipo desarrollado se recomienda incluir el tipo de Scripts que permiten añadir código al programa, para que aplicaciones cuyo reconocimiento no fue contemplado, sean incluidas en los procesos de detección de información, sin necesidad de actualizar el programa.

- Para mejorar la interacción con las aplicaciones Microsoft Word y Excel, deben estudiarse con mayor profundidad los modelos de objeto de dichas aplicaciones. Dado que a través de los modelos de objeto se facilita el reconocimiento de distintos objetos que no forman parte de los definidos por el sistema operativo, también se recomienda estudiar los modelos de objetos pertenecientes a otras aplicaciones con el fin de mejorar la funcionalidad del Programa Lector de Pantalla. Para ello, se puede iniciar con el estudio de las librerías que Visual Basic provee para acceder a distintas aplicaciones.

- Active Accesibility es una herramienta brindada por Microsoft para proveer accesibilidad a sus aplicaciones. Ya que la mayoría de objetos utilizados en sus aplicaciones no son fácilmente accesibles, se recomienda estudiar e investigar las formas de implementación y utilización de esta herramienta.

- El prototipo de Lector de Pantalla puede ser adaptado para correr sobre la familia de Sistemas Operativos Windows 2000, buscando actualizaciones o alternativas para las funciones de mapeo de archivos en memoria, ya que las utilizadas en la versión actual del programa intentan escribir sobre espacios de memoria que son reservados dentro de dichos sistemas.

- Para el desarrollo del Lector de Pantalla se utilizo Microsoft Visual Basic como lenguaje de desarrollo, por la claridad de código entre otros beneficios que este ofrece, sin embargo por el tipo de aplicación desarrollada y metodología utilizada, Microsoft Visual C puede ser otra alternativa como lenguaje de desarrollo, ya que es altamente potente para el manejo de los elementos utilizados en esta aplicación. Por tanto podrían realizarse pruebas de implementación del programa.

- Si el prototipo se convirtiera en una herramienta funcional para luego ser implementado, sería fundamental que el instalador incluyera instrucciones orales para que una persona no vidente pueda instalarlo.

XI. FUENTES DE INFORMACIÓN

- Kendall y Kendall. **Análisis y Diseño de Sistemas**. Prentice Hall Hispanoamericana, SA. México, 1991.

- Mark Andrews. **Aprenda Visual C++ Ya.** McGraw – Hill. Madrid, España, 1997.

- Microsoft Corporation. **MSDN Library Visual Studio 6.0.**

- William H. Murray / Chris H. Pappas. **WINDOWS PROGRAMMING: An Introduction.** Osborne McGraw – Hill. USA, 1990.

- An Introduction to Screen Readers
  http://mason.gmu.edu/~swidmaye/portfolio/edit797assistivetech.htm

- Windows API Guide
  http://www.vbapi.com

XII. GLOSARIO

- **Accesibilidad:**
  La accesibilidad para personas con discapacidades (o simplemente, "accesibilidad") se refiere a aquellas características de software que permiten a estos usuarios utilizar las aplicaciones de forma eficiente.

- **API (Application Programming Interface ó Interfaz de Programación de Aplicaciones):**
  Interfaz de software utilizada por las aplicaciones para solicitar al sistema operativo de la PC la realización de distintas tareas de nivel inferior.

- **Bandera (Flag):**
  Marca en un programa que señala una condición o estatus en particular.

- **Braile:**
  Escritura en relieve para el uso de los ciegos.

- **Buffer:**
  Espacio de memoria para almacenamiento temporal de datos.

- **Clase de Ventana (Window Class):**
  Un conjunto de atributos que Windows utiliza como plantilla para crear una ventana. Cada clase de ventana tiene un procedimiento de ventana que procesa mensajes para todas las ventanas de esa clase. Todas las ventanas en una aplicación basada en Windows son miembros de una Clase de Ventana.

- **Controlador de Ventana (Window Handle):**
  Es una variable que identifica un objeto; una referencia indirecta a un recurso del Sistema Operativo.

- **Discapacidad:**
  Condición de disminución de discapacidades físicas, sensoriales, psicológicas y mentales, ya sea por causas congénitas o adquiridas.

- **DLL (Dynamic Link Library ó Biblioteca de enlace dinámico):**
  Un archivo que contiene una o más funciones que son compiladas, vinculadas, y almacenadas de forma separada a la aplicación que las utiliza, permitiendo que sean llamadas, cargadas, y vinculadas a dicha aplicación en tiempo de ejecución. Estos archivos utilizan usualmente la extensión ".dll"

- **Driver:**
  Programa que controla un dispositivo de Hardware.

- **Hook:**
  Punto o ubicación dentro del mecanismo de manejo de mensajes del sistema operativo en donde una aplicación puede instalar una subrutina para monitorear el tráfico de mensajes en el sistema, y procesar cierto tipo de mensajes antes que lleguen a su procedimiento de ventana destino.

- **Hot Key:**
  Una combinación de teclas definida por una aplicación, mediante la que se obtiene alta prioridad de la entrada generada por teclado, y que se utiliza para ejecutar alguna acción específica.

- **Instancia:**
  Es una copia de una aplicación que se ejecuta en la computadora. La computadora puede ejecutar múltiples instancias de una aplicación.

- **Mensajes:**
  Es un paquete de datos utilizado para comunicar información o un requerimiento. Los mensajes pueden ser pasados entre el Sistema Operativo y una aplicación o entre diferentes aplicaciones.

- **Orden Z:**

  El orden z indica la posición de una ventana dentro de una pila de ventanas superpuestas. Esta pila de ventas está orientada a lo largo de un eje imaginario, el eje z, el cual se extiende hacia el exterior de la pantalla. La ventana al inicio del orden z se superpone al resto de ventanas. La ventana al final del orden z es superpuesta por el resto de ventanas.

- **Software:**

  Término que designa al conjunto de programas operativos que posibilitan el uso de la computadora.

- **Subclassing (Subclasificación):**

  Técnica, mediante la cual una aplicación puede interceptar los mensajes que llegan a su procedimiento de ventana, antes de que éste pueda procesarlos.

- **Thread:**

  Es un fragmento de código que forma parte de una aplicación. Un Thread puede ejecutar cualquier parte del código de una aplicación, incluyendo el código que esté siendo ejecutado por otro Thread. Toda aplicación posee al menos un Thread para ejecutar su código, aunque es posible que posea más de uno.

- **Ventana:**

  Es un área rectangular sobre la pantalla donde una aplicación despliega información de salida y recibe entradas del usuario. Las ventanas son también los medios por los cuales las aplicaciones envían y reciben mensajes del Sistema Operativo.

- **Virtual Key:**

  Un valor independiente de los dispositivos, que identifica el propósito de una tecla que ha sido presionada al igual como lo hace el driver del teclado de Windows.