Capítulo III
Diseño e Implementación

3.1 Introducción

En este capítulo hablaré del análisis, diseño e implementación del sistema que permite la integración de un agente animado con voz generada por tecnología Text-To-Speech (TTS) y cualquier página en lenguaje HTML. Una aplicación específica es el sistema de *Libros de Texto Gratuito Interactivos* (LTGI), utilizando los libros de lecturas de la SEP, integrados dentro del sistema de *Enciclomedia* como páginas en HTML puro. Al decir "HTML puro" me refiero a que estás páginas carecen de componentes o aplicaciones externas dentro del código HTML como Java Applets, JSP's, controles de Active X o Scripting de cualquier tipo.

3.2 Restricciones del Sistema

Desde el inicio, el objetivo principal era lograr integrar un personaje animado a las lecciones de lectura que se encontraban dentro del sistema de *Enciclomedia*. El personaje animado debía tener la capacidad de leer la lección seleccionada por medio de un sintetizador de voz en español, además, debía ser visible en todo momento y tener una interacción con el estudiante por medio de movimientos, sincronización y gesticulación de las palabras leídas.

Las restricciones presentadas dentro del sistema *Enciclomedia* son las siguientes:

- El sistema está desarrollado para funcionar en computadoras con sistema operativo Windows.
- Los Libros de Texto Gratuito de la SEP se digitalizaron en páginas que utilizan lenguaje HTML "puro".

- Las páginas en HTML de los libros de la SEP deben ser utilizadas sin modificación sustancial del formato o diseño.
- El editor de páginas en HTML utilizado por *Enciclomedia* fue *FrontPage 2000*.
- La implementación de la base de datos de *Enciclomedia* fue desarrollada utilizando SQL y el servidor de datos que utilizaron fue *SQL Server v6.5*.
- Existe un acuerdo de colaboración entre Microsoft y el ILCE para la utilización de la *Enciclopedia Encarta* dentro de *Enciclomedia*.
- El sistema de *Enciclomedia* es accesible por medio de Internet y por lo tanto cualquier componente adicional debe cumplir con este medio de acceso.
- Un usuario de *Enciclomedia* utiliza un navegador Web como medio de acceso al sistema y a las páginas contenidas en el servidor HTML de *Enciclomedia*.

Como primera solución se pensó en usar los tutores animados desarrollados por el CSLR dentro de sus libros interactivos, los cuales ya contaban con un personaje animado que podía leer el texto de una lección ya sea por palabra o por oración, e incluso la lección completa. Al contar con documentación y código disponible de los tutores del CSLR, se inició el análisis correspondiente a la solución propuesta y encontramos los siguientes detalles:

- La arquitectura que permite el funcionamiento de los libros interactivos del CSLR es muy compleja.
- El sistema estaba desarrollado para el sistema operativo Linux.
- Para poder instalar los tutores y libros interactivos en una computadora con sistema operativo Windows, se requiere instalar un emulador de Linux previamente.
- La base de datos que utilizan es *PostgreSQL* [López, 2003].
- Las páginas de los libros interactivos y tutores se desarrollaron utilizando el lenguaje de Scripts PHP y también utilizan el servidor Apache para tener acceso a ellas [López, 2003].
- El sintetizador de voz utilizado es Festival el cual está escrito en C++ [CSTR, 2004].

- El servidor de TTS esta ligado a archivos binarios de Java para poder ejecutarse [López, 2003].
- Los tutores y libros interactivos funcionan bajo tecnologías de Java Applets por lo que se requiere instalar el ambiente de ejecución JRE v1.4.2 [López, 2003].

Después de este análisis concluimos que los componentes desarrollados por el CSLR contaban con demasiadas restricciones para su funcionamiento y que integrar al personaje animado y el sintetizador de voz dentro de una página en HTML puro era un proceso no factible. Lo anterior impedía cumplir con las restricciones del sistema de *Enciclomedia* y la facilidad de uso que debían tener los usuarios, tomando en cuenta que serían estudiantes de los primeros grados de primaria.

Por lo tanto, se desechó la idea de integrar los tutores del CSLR al sistema de *Enciclomedia* y fue necesario buscar otros métodos o tecnologías que cumplieran con los requisitos necesarios. Sin embargo, cualquier otra opción encontrada tendría que cumplir con los conceptos teóricos de los libros interactivos del CSLR. El personaje animado tendría que funcionar como una herramienta de enseñanza interactiva que guié a los estudiantes en el proceso de aprendizaje de lecciones de lectura.

3.3 Requerimientos del Sistema

Este proyecto tiene por objetivo integrar un personaje animado en las lecciones de lectura de los Libros de Texto Gratuito de la SEP contenidos en el sistema de *Enciclomedia*. Además del personaje animado, el sistema debe incluir otras características que permitan un desempeño eficiente para cumplir con las metas del proyecto. Este personaje tiene la capacidad de interactuar con los estudiantes utilizando voz sintetizada para leer en voz alta el texto de una lectura permitiendo al estudiante darse cuenta de sus aciertos y errores de lectura.

3.3.1 Requerimientos Funcionales

Los requerimientos funcionales están clasificados en tres categorías que identifican sus prioridades ante el usuario. Las categorías son las siguientes:

- **Evidente**: Debe realizarse y el usuario debe saber que se ha realizado.
- **Oculto**: Debe realizarse aunque no es visible para el usuario.
- **Opcional**: Su inclusión no repercute significativamente en el costo ni en otros requerimientos funcionales.

Requerimiento Funcional	Categoría
El sistema debe estar disponible por Internet.	Evidente
El sistema debe mostrar las lecciones de lectura sin alteraciones sustanciales a su formato original.	Evidente
El personaje animado debe estar visible.	Evidente
La voz del personaje debe ser audible a un nivel de volumen y velocidad adecuados.	Evidente
La voz del personaje debe estar en idioma español.	Evidente
El sistema debe procesar el texto de la página Web, indicado por el programador, para que pueda ser leído palabra por palabra.	Oculto
El personaje animado debe leer el texto que el usuario elija.	Evidente
El usuario puede controlar las acciones del personaje animado.	Evidente

3.1 Tabla de Requerimientos Funcionales

3.3.2 Requerimientos No Funcionales

Los requerimientos no funcionales especifican otras características que el sistema debe tener para complementar su funcionalidad. Algunos de los siguientes puntos están

basados en la clasificación del modelo FURPS+ por sus siglas en inglés: ***F****unctionality*, ***U****sability*, ***R****eliability*, ***P****erformance*, ***S****upportability* y el símbolo **+** que indica requerimientos adicionales [Larman, 1999].

- **Interfaz**: La interfaz del usuario (en términos de formato) debe ser la misma a la presentada en los libros de lectura del sistema de *Enciclomedia*.
- **Usabilidad**: Este proyecto está destinado a usuarios que en general pueden tener muy poco (o nulo) manejo de herramientas informáticas. Esto impone la necesidad de que todas las interfases con el usuario sean muy claras, muy amigables, y de fácil manejo.
- **Extensibilidad**: Se debe diseñar la aplicación pensando en que debe ser sencillo agregar e intercambiar módulos, sobre todo en lo que respecta a interacción con el exterior (otras lecciones de lectura u otras páginas Web).
- **Fiabilidad**: En este caso, aunque se realizará un control de errores, no es necesaria una gran tolerancia a fallas, ya que no estamos trabajando con un sistema crítico.
- **Desempeño**: Se debe presentar al usuario un diálogo fluido y un tiempo de respuesta considerablemente bajo.
- **Flexibilidad**: El sistema debe permitir que el usuario elija libremente las acciones del personaje animado. Esto incluye las acciones de leer por palabra, por oración o la lectura completa.

3.4 Casos de Uso

En las siguientes figuras se muestran los diagramas de casos de uso para las acciones que puede realizar un usuario con el sistema LTGI.

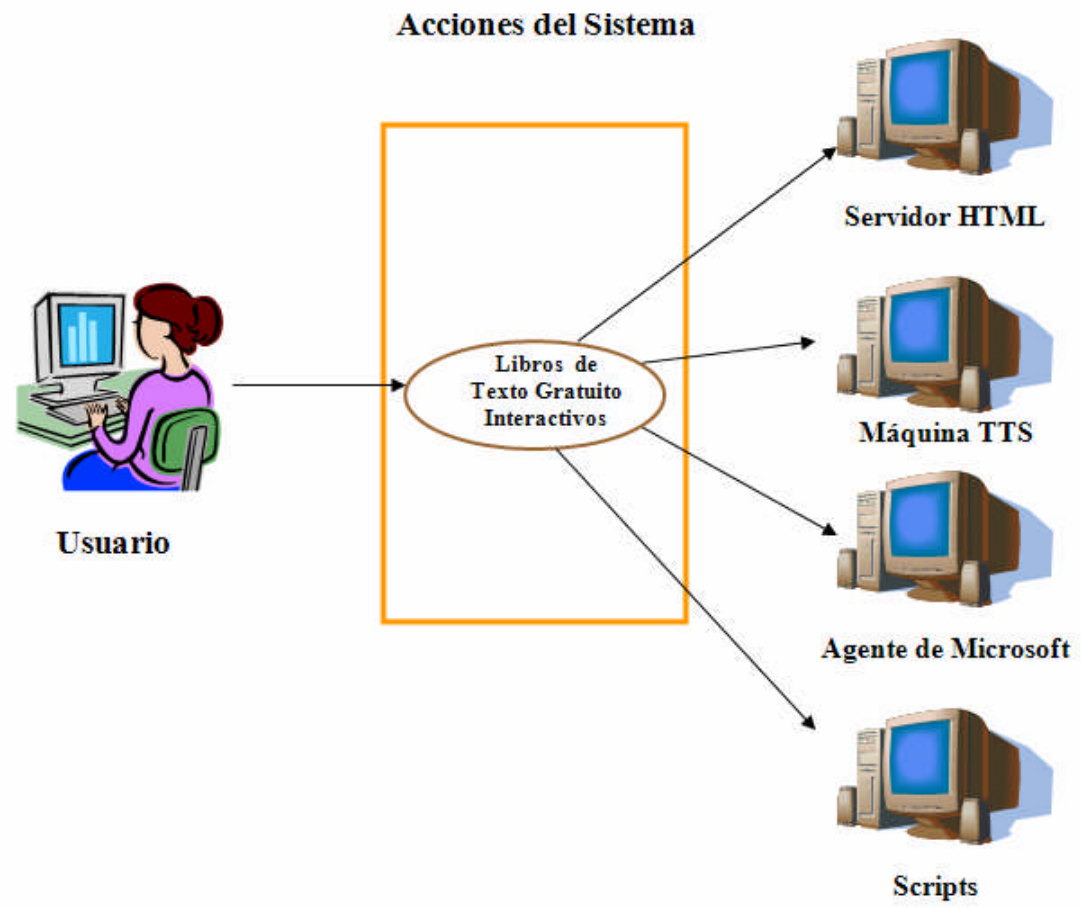

Figura 3.1 Caso de Uso para acceso al sistema.

El caso de uso de la figura 3.1 muestra la interacción del usuario al entrar a cualquiera de los libros interactivos de lecturas. A su vez, los libros están relacionados con funciones integradas dentro de ellos, como la máquina TTS que permite escuchar el texto de la lectura en voz sintética; el agente de Microsoft que aparece como un guía de lectura presente en todo momento; y los Scripts que controlan las acciones del agente de Microsoft y el sintetizador de voz con las lecturas en HTML. Por último el servidor HTML permite mostrar en el navegador del usuario todos los componentes de los libros interactivos para que el usuario tenga acceso a ellos.

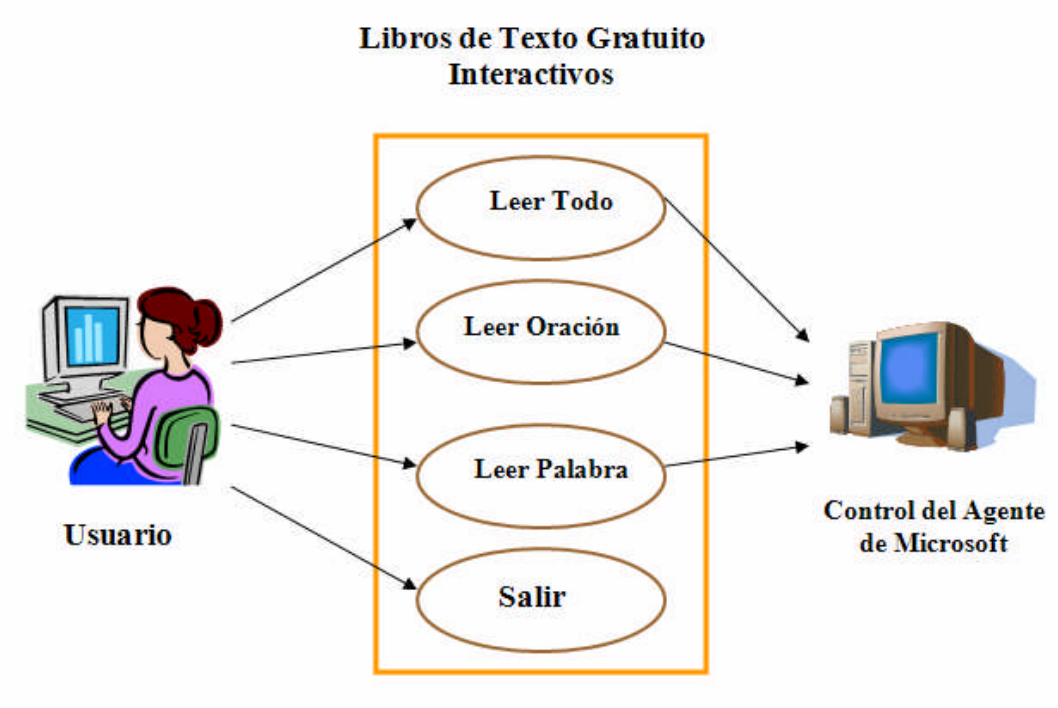

Figura 3.2 Caso de Uso para *Libros de Texto Gratuito Interactivos*.

La figura 3.2 muestra el caso de uso en el cual el usuario interactúa con los LTGI. Las acciones que un usuario puede realizar dentro de una lección de lectura son: "Leer Todo", que permite que el agente de Microsoft lea la lectura completa de forma continua; "Leer Oración", con la cual el usuario puede escuchar la oración que desea y el agente leerá esa oración deteniéndose al terminar; y "Leer Palabra", con esta acción es posible que el usuario elija una palabra dentro de toda la lectura para que el agente de Microsoft la lea. Todas las acciones anteriores son gestionadas por funciones de control del agente de Microsoft a las cuales el usuario tiene acceso. Por último el usuario también tiene la posibilidad de salir de la lectura y escoger otra o salir completamente del sistema.

3.5 Diagrama de Contexto

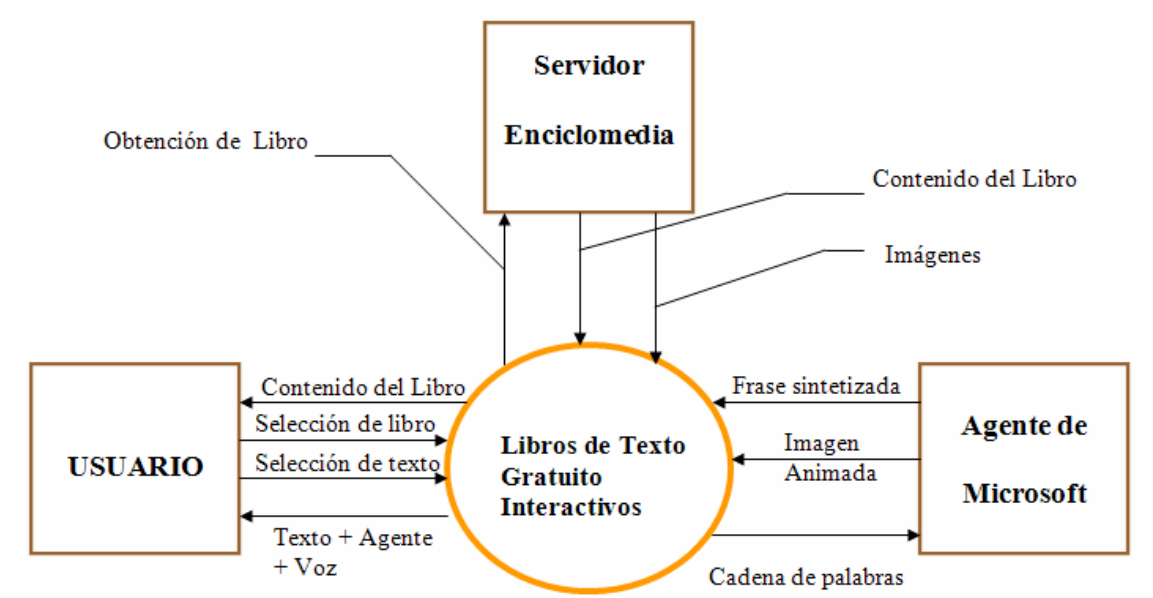

Figura 3.3 Diagrama de Contexto de LTGI.

El contexto general del sistema se puede apreciar en el diagrama de la figura 3.3. El diagrama muestra tres entidades que interactúan entre sí por medio de los LTGI: el Usuario, el Servidor de *Enciclomedia* y el Agente de Microsoft. Es importante mencionar que si bien, los LTGI fueron diseñados para funcionar dentro del servidor de *Enciclomedia*, actualmente están contenidos en el servidor de la UDLA.

El Usuario selecciona las lecturas del libro que desea leer desde su navegador. Por su parte, los *Libros de Texto Gratuito Interactivos* obtienen información del libro que se va a leer y mandan esa petición al Servidor de *Enciclomedia*. El Servidor muestra el contenido del libro y las imágenes correspondientes en formato HTML en el navegador del Usuario. El Usuario tiene tres opciones que le permiten seleccionar el texto que desea escuchar y esa petición es recibida y enviada al Agente de Microsoft en una cadena de

palabras para que la procese y devuelva la imagen animada del Agente y la frase sintetizada.

3.6 Agente de Microsoft

Después de determinar que los personajes animados de los tutores desarrollados por el CSLR no eran una opción viable para integrarlos a los Libros de Texto Gratuito de la SEP dentro del proyecto *Enciclomedia*, se evaluaron otras herramientas buscando que fueran apropiadas para los fines del proyecto. Los Agentes de Microsoft resultaron una opción factible, sugerida por los desarrolladores de *Enciclomedia*, por lo que su aprobación fue total. Las razones principales para considerar la utilización de los Agentes de Microsoft fueron la facilidad de acoplamiento con el sistema de *Enciclomedia* y el acuerdo de colaboración entre Microsoft y el ILCE.

Los Agentes de Microsoft son un conjunto de herramientas que permiten incorporar personajes animados en aplicaciones y páginas Web. Estos personajes (figura 3.4) pueden hablar, utilizando TTS o audio pregrabado, e incluso aceptar señales de audio por medio de un reconocedor de voz [Microsoft 2, 2004].

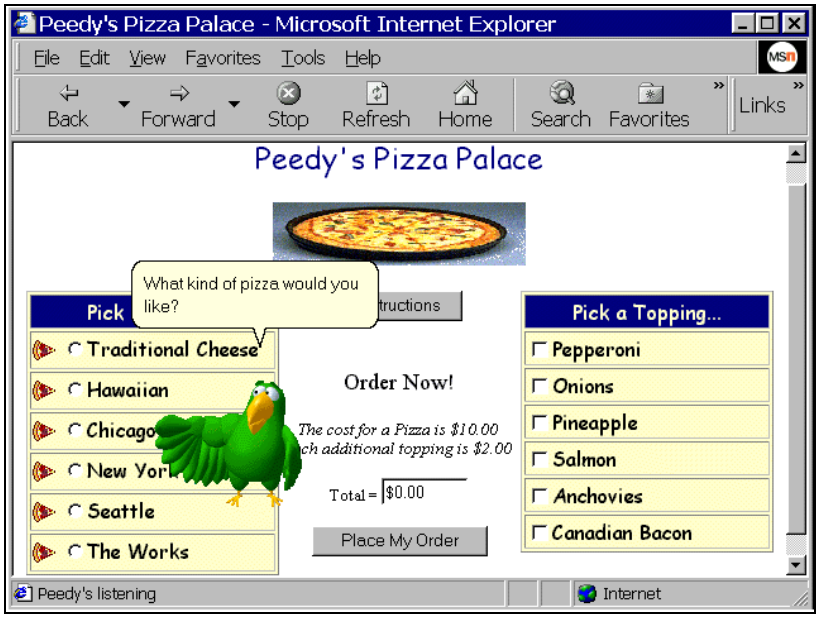

Figura 3.4 Agente de Microsoft [Microsoft 2, 2004].

Como mencioné anteriormente, una ventaja de los Agentes de Microsoft es la versatilidad de posibles usos y la facilidad con la que se pueden acoplar a diferentes aplicaciones. Su implementación resulta ser sencilla pues cuentan con una gran variedad de lenguajes en los que se pueden implementar, por ejemplo: tecnologías de Active X, Visual Basic o Visual C++. También es posible integrar estos agentes en páginas HTML por medio de lenguajes de Scripting como *Visual Basic Scripting Edition* (VBScript) o *Java Script* [Microsoft2, 2004].

Las características principales de los Agentes de Microsoft están enlistadas en la siguiente tabla:

Características del Agente de Microsoft
Interfaz de Usuario
Los agentes se integran como parte funcional de la aplicación en la que están inmersos, no están en una ventana externa.
Salida de audio utilizando una máquina TTS, audio pregrabado y/o texto en un globo de palabras.
Entrada de audio por medio de un reconocedor de voz.
Soporte de Programación
Compatible con una gran variedad de lenguajes de programación y herramientas, incluyendo páginas Web por medio de lenguajes de Scripting.
Control sobre las animaciones, comandos de reconocimiento de voz, y sincronización de eventos.
Compilación previa que permite ejecución inmediata.
Flexibilidad
Mínimo ancho de banda para ejecución; también utiliza componentes para módems de baja velocidad.
Posibilidad de crear personajes propios compilando secuencias de animación.
Diversidad en máquinas de síntesis voz en idiomas extranjeros.
Los componentes de voz, propiedad de otros desarrolladores, son compatibles con los compilados por Microsoft.

Tabla 3.2 Características del Agente de Microsoft [Microsoft 2, 2004]

3.6.1 Requerimientos del Agente de Microsoft

Los Agentes de Microsoft fueron desarrollados buscando los mínimos requerimientos de sistema para su instalación. Su funcionamiento y desempeño óptimos se pueden lograr en computadoras construidas en los últimos años. En la siguiente tabla se enlistan los requerimientos mínimos para su funcionamiento:

Requerimientos
Microsoft Windows® 95, Windows 98, Windows Me, Windows NT® 4.0 (x86), Windows 2000 o Windows XP®.
Internet Explorer version 3.02 o superior.
Procesador Pentium a 100-megahertz (MHz) PC (o superior).
Al menos 16 megabytes (MB) de memoria RAM.
Al menos 1 MB de espacio libre en disco duro para los componentes principales.
2-4 MB de espacio libre en disco duro adicional para cada personaje instalado.
32 KB de espacio libre en disco duro adicional para cada componente (dll) de lenguaje adicional.
Recomendaciones
1.6 MB de espacio libre en disco duro adicional si se desea usar la Máquina TTS de Lernout & Hauspie® TruVoice para la salida de audio.
22 MB de espacio libre en disco duro adicional si se desea usar la Máquina de Reconocimiento de Voz de Microsoft para la entrada de voz.
Una tarjeta de sonido compatible con Windows.
Micrófono y bocinas compatibles con Windows.

Tabla 3.3 Requerimientos de Sistema para los Agentes de Microsoft [Microsoft 2, 2004]

3.6.2 Opciones de Personajes del Agente de Microsoft

Los Agentes de Microsoft cuentan con funcionalidad muy versátil además de una gran flexibilidad. Algunas de las opciones disponibles para el usuario final son: elegir dentro de 4 personajes animados (un robot, un perico, un mago y un genio), elegir el lenguaje del agente, controlar sus movimientos y acciones, etc.

Para poder utilizar a los diferentes personajes en cualquier tipo de aplicación, estos deben ser instalados previamente descargando el archivo de ejecución desde la página de descargas del Agente. No es necesario instalar todos los personajes pues funcionan independientemente entre sí, y tampoco es completamente necesario instalarlos localmente en la máquina del usuario pues se pueden descargar automáticamente desde un sitio en Internet. Posteriormente se explicará a detalle como realizar la instalación desde la página de descargas o automáticamente dentro del código de esta aplicación.

El Agente de Microsoft puede ser utilizado en muchos tipos de aplicaciones como se mencionó anteriormente, en el caso de este proyecto se utiliza dentro de una página HTML y su implementación se realizó con lenguajes de "Scripting".

3.6.3 Opciones de Voz

Una de las características más importantes de los Agentes de Microsoft es la posibilidad de elegir el idioma con el cual el usuario puede interactuar. Microsoft compiló diferentes componentes de lenguaje para el Agente, los cuales están enlistados a continuación:

Lenguajes soportados por el Agente de Microsoft	
Árabe	Noruego
Vasco	Polaco

Lenguajes soportados por el Agente de Microsoft	
Croata	Portugués
Checo	Portugués (Brasil)
Danés	Rumano
Holandés	Ruso
Finlandés	Chino Simplificado
Francés	Eslovaco
Alemán	Esloveno
Griego	Español
Hebreo	Sueco
Húngaro	Tailandés
Italiano	Chino Tradicional
Japonés	Turco
Coreano	

Tabla 3.4 Lenguajes soportados por el Agente de Microsoft [Microsoft 3, 2004].

Sin embargo, los lenguajes anteriores están contenidos en librerías que sólo dan soporte a algunas características del Agente, como el texto en los globos o ventanas. Para que el Agente pueda hablar en un idioma en particular, es necesario instalar máquinas de voz en TTS por separado. El Agente de Microsoft utiliza el sintetizador de voz "TrueVoice" de Lernout & Hauspie para el idioma en inglés americano y utiliza el sintetizador "TTS3000" de la misma compañía, para los idiomas enlistados a continuación:

Lenguajes en TTS
Inglés Británico
Holandés
Francia

Lenguajes en TTS
Alemán
Italiano
Japonés
Coreano
Portugués (Brasil)
Ruso
Español

Tabla 3.5 Idiomas de las máquinas TTS3000 de Lernout & Hauspie soportados por el Agente de Microsoft [Microsoft 3, 2004].

Además de las máquinas TTS de Lernout & Hauspie, existen otro tipo de máquinas TTS compatibles con el Agente de Microsoft. En la página del Agente se mencionan las siguientes [Microsoft 4, 2004]:

- Digalo Text-To-Speech Engine
- Elan Speech Engine
- Eloquent Technology ETI-Eloquence
- IBM ViaVoice^{TM} Outloud

Todas estas máquinas tienen diferentes lenguajes disponibles y para poder funcionar con el Agente de Microsoft, es necesario instalar los componentes binarios de SAPI 4.0a. Estos componentes se pueden bajar de la página de descargas del Agente y la instalación se explicará en el punto 3.8 de este reporte.

3.6.4 Características de la Voz

Uno de los servicios que ofrece el Agente de Microsoft, es la posibilidad de modificar la salida de la señal de voz (proveniente de una máquina TTS) por medio de etiquetas (o *tags*) especiales. Estás etiquetas se insertan en el cadena de texto de entrada para el Agente y de esta forma se pueden modificar diferentes características de la señal de voz de salida como el volumen, entonación, ritmo, etc.

La siguiente tabla describe las etiquetas de salida de voz disponibles para el Agente de Microsoft, cabe mencionar que no todos los idiomas se comportan de la misma forma utilizando la misma etiqueta e incluso algunas etiquetas no son soportadas en algunos idiomas:

Etiquetas de salida de voz del Agente de Microsoft	
Etiqueta de Salida de Voz	**Características**
Chr	Especifica el tipo de voz: "Normal" (Voz normal) "Monotone" (Voz monótona) "Whisper" (Voz en susurro)
Ctx	Especifica el contexto de la salida de voz: "Address" (Direcciones o números telefónicos) "E-mail" (Dirección de correo electrónico) "Unknown" (Default – El contexto es desconocido)
Emp	Esta etiqueta hace énfasis en la siguiente palabra dicha. La etiqueta (\Emp\) debe estar inmediatamente antes de la palabra.
Lst	Repite la última oración dicha por el Agente. Esta etiqueta debe aparecer sola dentro del método Speak; ninguna otra etiqueta o texto debe ser incluido. Por ejemplo: Peedy.Speak = \Lst\ La etiqueta \Lst\ repite la señal de voz de salida en TTS o de cualquier archivo .wav o .lwv usado.

Etiquetas de salida de voz del Agente de Microsoft	
Etiqueta de Salida de Voz	**Características**
Map	Permite "mapear" el texto escrito dentro del globo de texto y lo que será dicho por el Agente. \Map="*spoken text*"="*balloontext*"\ spokentext: Es el texto que será dicho por el Agente balloontext: Es el texto que aparecerá dentro del globo de texto Esto permite que el Agente diga algo diferente a lo que aparece dentro del globo de texto.
Pau	Produce una pausa en el texto dicho por el número de milisegundos especificado. \Pau=*número de milisegundos*\
Pit	Especifica el tono base del texto dicho por el Agente en hertz. \Pit=*número en hertz para el tono*\
Rst	Reinicia todas las etiquetas a la configuración normal.
Spd	Especifica la velocidad promedio del texto dicho por el Agente. \Spd=*número de palabras por minuto*\
Vol	Especifica el volumen base de la señal de salida de voz. \Vol=*volumen de voz; 0 es el mínimo y 65535 es el máximo*\

Tabla 3.6 Etiquetas de salida de voz del Agente de Microsoft [Microsoft 5, 2004].

La señal de salida de voz mantiene las características especificadas por las etiquetas hasta que el Agente termina de decir el texto indicado. Después de esto, la señal de voz regresa a sus parámetros iniciales.

Las etiquetas de salida de voz tienen algunas reglas de sintaxis generales para el lenguaje en que se programen [Microsoft 5, 2004]:

- Todas las etiquetas inician y terminan con una diagonal invertida (\).

- La diagonal invertida simple no está habilitada dentro de una etiqueta. Para incluir una diagonal invertida simple dentro de un texto, es necesario escribir una doble diagonal invertida (\\).
- Las etiquetas no son sensibles a la capitalización. Por ejemplo, \pit\ es lo mismo que \PIT\.
- Las etiquetas son sensibles a los espacios en blanco. Por ejemplo, \vol\ no es lo mismo que \ vol \.

Por medio de este conjunto de etiquetas es posible generar una diversidad enorme de tipos de voces. Al modificar varias de las características de la voz de los personajes es posible hacerlos cantar, recitar, declamar e incluso contar chistes. Sin embargo, para lograr el resultado deseado es necesario invertir una buena cantidad de tiempo probando diferentes tonos, velocidad o volumen.

Para que el Agente de Microsoft pueda decir algo, es necesario darle un valor de entrada. Este valor de entrada, por lo general, es una cadena de letras; pero también puede ser una variable, un *URL*, o un archivo de audio (únicamente en formato .wav o .lwv).

El método *Speak* es el responsable de manipular la entrada de texto y convertirla en una salida de audio sincronizando los labios del personaje. La sincronización se ejecuta tanto con el método TTS como con los archivos de audio pregrabados. Esta sincronización se logra debido a que las interfaces del Agente están basadas en el estándar SAPI (*Speech Application Programming Interface*). Lo anterior significa que se puede utilizar al Agente con máquinas TTS desarrolladas por empresas diferentes a las compiladas por Microsoft en las interfaces del Agente, siempre y cuando cumplan con las especificaciones de SAPI. Sin embargo, la máquina TTS que se quiera utilizar debe soportar fonemas de IPA (*International Phonetic Alphabet*) como parte de la interfaz *NotifySink* de SAPI para que la sincronización sea adecuada [Microsoft 7, 2004].

El método *Speak* tiene dos parámetros de entrada: [*Text*], [*Url*]. Sin embargo estos parámetros son opcionales y en su lugar se puede utilizar una etiqueta como mencioné en el punto anterior [Microsoft 7, 2004].

El parámetro [*Text*] define como entrada a una cadena de caracteres, la cual se puede escribir directamente dentro de comillas ("cadena de caracteres") o con una variable que la contenga [Microsoft 7, 2004].

Para poder utilizar un archivo de audio (.wav o .lwv) como entrada, es necesario indicar la localización del archivo de audio dentro del parámetro [*Url*] del método *Speak* [Microsoft 7, 2004].

La flexibilidad del Agente de Microsoft en permitir diferentes entradas crea la posibilidad de implementar aplicaciones que funcionen vía Internet con mejor eficiencia (por medio de entradas de texto en cadenas de caracteres y usando una máquina TTS para obtener la voz de salida). O incluso pregrabar con gran calidad el texto que se desea y utilizar los archivos de audio (los cuales tendrán que estar locales en la máquina del usuario) como entradas para el Agente.

3.6.5 Características de Texto

Además de la señal de salida de voz, la interfaz del Agente de Microsoft cuenta con subtítulos textuales, en la forma de globos al estilo de caricaturas, dentro de los cuales va apareciendo el texto de salida que el Agente va diciendo (figura. 3.5). El globo de texto se oculta cuando el Agente termina de decir el mensaje de voz de salida.

Para poder hacer modificaciones al globo de texto del Agente, se necesita utilizar el objeto *Balloon* quien es hijo del objeto *Character*. Con el objeto *Balloon* se pueden modificar propiedades del globo de texto como: tamaño de letra, tipo de letra, color de fondo, color del contorno del globo, número de letras dentro del globo y otras más.

Figura 3.5 Globo de texto del Agente de Microsoft [Microsoft 6, 2004].

3.7 Instalación de Componentes

El Agente de Microsoft no es un programa que funcione por sí mismo y realice una acción. Como se mencionó anteriormente, el Agente es una tecnología de desarrollo de herramientas que nos ayuden a realizar alguna tarea dentro de nuestras aplicaciones. En consecuencia, para poder integrarlo a una aplicación, es necesario instalar diversos componentes como máquinas de síntesis de voz, reconocedores de voz o componentes de lenguaje. En los siguientes puntos se detalla la instalación de los componentes necesarios para que el sistema funcione.

3.7.1 Instalación del Agente de Microsoft

En las últimas versiones del sistema operativo Windows (Windows Me, Windows 2000 y Windows XP) el Agente de Microsoft viene instalado junto con el sistema por lo que no es necesario reinstalarlo. Sin embargo, si se cuenta con una versión anterior a las mencionadas, será necesario descargar e instalar los componentes del núcleo del Agente de Microsoft de la siguiente dirección [Microsoft 3, 2004]:

http://activex.microsoft.com/activex/controls/agent2/MSagent.exe

Una vez instalados estos componentes, se podrá comenzar a utilizar al Agente de Microsoft en diversas aplicaciones. En el punto 3.6.2 mencioné que existen cuatro personajes disponibles para el Agente, estos personajes pueden cargarse automáticamente desde un sitio en Internet sin necesidad de instalarlos localmente. Pero en el caso de que el usuario quiera tener estos archivos localmente, tendrá que descargar los archivos del personaje deseado de las siguientes direcciones:

- Genio: http://download.microsoft.com/download/0/0/c/00cde5f8-321d-4325-baae-eb27f1bde85f/Genie.exe
- Mago: http://download.microsoft.com/download/1/d/b/1dbee406-9b5f-48c5-b901-dd1a3f3c4669/Merlin.exe
- Perico: http://download.microsoft.com/download/a/f/5/af572f68-b83e-4e2c-8b0f-fd5fadf588e7/Peedy.exe
- Robot: http://download.microsoft.com/download/2/b/9/2b904bbb-c0b1-4840-b332-ba0615d1041e/Robby.exe

Además de los componentes anteriores, el Agente de Microsoft requiere de la instalación de componentes de lenguaje, diferentes a los de inglés, en el caso que se deseé usar un idioma diferente. Estos componentes son librerías que dan soporte a diálogos, ventanas, herramientas de ayuda, y globos de texto de los componentes del núcleo del Agente [Microsoft 3, 2004]. Sin embargo, la lista de componentes de lenguaje soportados por el Agente es diferente a los lenguajes de las máquinas TTS disponibles como se puede apreciar en las tablas del punto 3.6.3.

Los componentes de lenguaje se pueden instalar y descargar automáticamente desde una aplicación basada en HTML, pero si el usuario desea tener instalados componentes para diferentes idiomas localmente, debe recurrir a la dirección correspondiente dentro de la siguiente tabla:

Nota: Se debe utilizar el prefijo de la dirección URL más el identificador del idioma deseado como sufijo para obtener la dirección de descarga completa.

Componentes de Lenguajes del Agente de Microsoft			
Prefijo: http://activex.microsoft.com/activex/controls/agent2/			
Idioma	**Identificador**	**Idioma**	**Identificador**
Árabe	AgtX0401.exe	Noruego	AgtX0414.exe
Vasco	AgtX042D.exe	Polaco	AgtX0415.exe
Croata	AgtX041A.exe	Portugués	AgtX0816.exe
Checo	AgtX0405.exe	Portugués (Brasil)	AgtX0416.exe
Danés	AgtX0406.exe	Rumano	AgtX0418.exe
Holandés	AgtX0413.exe	Ruso	AgtX0419.exe
Finlandés	AgtX040B.exe	Chino Simplificado	AgtX0804.exe
Francés	AgtX040C.exe	Eslovaco	AgtX041B.exe
Alemán	AgtX0407.exe	Esloveno	AgtX0424.exe
Griego	AgtX0408.exe	Español	AgtX0C0A.exe
Hebreo	AgtX040D.exe	Sueco	AgtX041D.exe
Húngaro	AgtX040E.exe	Tailandés	AgtX041E.exe
Italiano	AgtX0410.exe	Chino Tradicional	AgtX0404.exe
Japonés	AgtX0411.exe	Turco	AgtX041F.exe
Coreano	AgtX0412.exe		

Tabla 3.7 Direcciones de descarga para instalación de componentes de lenguaje [Microsoft 3, 2004].

Las direcciones anteriores no están disponibles directamente desde ninguna página de documentación o descargas de Microsoft. Para que el usuario tenga acceso a ellas, necesita entrar a la página de descargas del Agente, seleccionar el componente de lenguaje que desea, y oprimir un botón para generar la descarga. Si un desarrollador desea generar una descarga automática desde su aplicación, debe utilizar las direcciones contenidas en la tabla 3.7.1.

3.7.2 Instalación de SAPI

En puntos anteriores mencioné que las interfaces del Agente de Microsoft están basadas en el estándar SAPI. SAPI toma sus iniciales de *Speech Application Programming Interface*, y fue introducido por primera vez en 1995 por Microsoft como parte de los servicios *Windows Open Services Arquitecture* (WOSA). Esto fue con la intención de facilitar la programación, aunque en algunos casos se requiere de varios trucos en el código para poder hacer que funcione, pero este es el mismo caso con cualquier API de Microsoft [Rozak, 1996].

En este momento la última versión de SAPI es SAPI 5.1, sin embargo existe una situación curiosa en el manejo de versiones pues la versión anterior (SAPI 4) no se ha deprecado y es posible utilizar ambos SDK's (*Software Development Kit*) en el mismo sistema [Microsoft 8, 2004]. Existen enormes diferencias entre las dos versiones, SAPI 5 surgió debido a la gran cantidad de errores y fallas encontradas en SAPI 4, por ello Microsoft decidió crear SAPI 5 desde cero. Sin embargo el extenso uso de SAPI 4 en muchas aplicaciones (incluidas las del propio Microsoft como el caso del Agente), ha provocado que los dos SDK's convivan entre sí. Es importante mencionar que una aplicación compilada para una versión en específico no funciona en la otra versión [Rozak, 1996].

SAPI 4 viene instalado junto con las últimas versiones del sistema operativo Windows (Windows Me, Windows 2000 y Windows XP), pero si se requiere utilizar alguna máquina TTS diferente a las compiladas por Microsoft para el Agente, se deben instalar los componentes binarios de ejecución de SAPI 4, utilizando la siguiente dirección [Microsoft 3, 2004]:

http://activex.microsoft.com/activex/controls/sapi/spchapi.exe

Los componentes de SAPI 4 pueden ser descargados e instalados automáticamente desde una aplicación basada en HTML, en la descripción del código del sistema, en el punto 3.8, detallaré de qué forma se puede realizar lo anterior.

3.7.3 Instalación de la Máquina TTS

El Agente de Microsoft utiliza la máquina TTS "TrueVoice" en inglés americano como el idioma de salida de voz por *default*. Para poder trabajar con un idioma diferente se necesita instalar la máquina TTS correspondiente. Como mencioné anteriormente, existen otras máquinas TTS desarrolladas por otras compañías, que son compatibles con las interfaces del Agente de Microsoft.

En la siguiente tabla, se muestran las direcciones para descargar la máquina TTS TTS3000 de Lernout & Hauspie en el idioma deseado:

Nota: Se debe utilizar el prefijo de la dirección URL más el identificador del idioma deseado, como sufijo, para obtener la dirección de descarga completa.

Máquinas TTS de Lernout & Hauspie	
Prefijo: http://activex.microsoft.com/activex/controls/agent2/	
Idioma	**Identificador**
Inglés Británico	lhttseng.exe
Holandés	lhttsdun.exe
Francés	lhttsfrf.exe
Alemán	lhttsged.exe
Italiano	lhttsiti.exe
Japonés	lhttsjpj.exe
Coreano	lhttskok.exe

Máquinas TTS de Lernout & Hauspie	
Prefijo: http://activex.microsoft.com/activex/controls/agent2/	
Idioma	**Identificador**
Portugués (Brasil)	lhttsptb.exe
Ruso	lhttsrur.exe
Español	lhttsspe.exe

Tabla 3.8 Direcciones de descarga para instalación de máquinas TTS [Microsoft 3, 2004].

Para instalar alguna máquina TTS diferente a las enlistadas en la tabla 3.7.3, es necesario consultar la página de la compañía que las fabrique para obtener los archivos y permisos de instalación.

3.8 Scripts

En puntos anteriores mencioné que el Agente de Microsoft es una tecnología que incluye una interfaz de programación que puede ser codificada desde cualquier lenguaje que soporte tecnología COM (*Common Object Model*) como C++ o Microsoft Visual Basic [Microsoft 2, 2004]. Los objetos COM son parte de una arquitectura de software desarrollada por Microsoft para construir aplicaciones basadas en componentes. Cada objeto COM tiene una identidad única la cual expone interfaces que permiten a otros componentes y aplicaciones tener acceso a las propiedades y métodos de los objetos [Webopedia, 2004].

El Agente de Microsoft también incluye controles de ActiveX que permiten se pueda programar desde lenguajes de Scripting como VBScript, Java Script, JScript, e incluso lenguajes creados por compañías alternas como MASH (*Microsoft Agent Scripting Helper*) [BellCraft, 2004].

Para el caso de este proyecto, se utilizaron dos lenguajes de Scripting para programar al agente: Visual Basic Scripting Edition 5.6 y JavaScript 1.3. Las versiones de ambos Scripts son soportadas por versiones superiores a Internet Explorer 5.0 y a Netscape Navigator 4.06. Una de las ventajas de utilizar lenguajes de Scripting es el poder combinar ambos lenguajes dentro del mismo código de HTML, de esta forma se logra aprovechar las ventajas que ambos lenguajes ofrecen. Las especificaciones y métodos de estos Scripts están descritos en los siguientes puntos.

3.8.1 Script de Agente de Microsoft

Para codificar las interfaces del Agente de Microsoft se pueden utilizar diferentes lenguajes de programación, siempre y cuando sean compatibles con tecnología COM. En el caso de este sistema, utilice el lenguaje *Visual Basic Scripting Edition* para programar el Script que manipularía al Agente. La utilización de este lenguaje obedece a factores de compatibilidad total con las interfaces del Agente (ya que VBScript es propiedad de Microsoft, contiene elementos de ActiveX y utiliza tecnología COM), y a la facilidad de programación (la sintaxis de este lenguaje es menos restringida y permite invocar clases y métodos de forma más natural).

En los siguientes puntos se describe a detalle las características de las propiedades, métodos y clases utilizadas para programar al Agente.

3.8.1.1 Etiquetas de Objetos

La primera particularidad para poder codificar al Agente en una página HTML, es tener que incluir etiquetas (*tags*) de los Objetos necesarios para el funcionamiento adecuado del Agente. Estas etiquetas son indispensables para declarar las clases que serán utilizadas, independientemente del lenguaje en el que se este programando, ya que los controles de ActiveX guardan en archivos diferentes las librerías de cada lenguaje.

El Objeto principal para el funcionamiento del Agente de Microsoft es el Objeto de Control. Para tenerlo disponible en nuestro código se necesita incluir una etiqueta de la siguiente forma [Microsoft, 2005]:

```
<OBJECT id=Agent codeBase=#VERSION=2,0,0,0 classid=clsid:D45FD31B-5C6E-
11D1-9EC1-00C04FD7081F></OBJECT>
```

Al incluir esta etiqueta dentro del código, se genera una petición para que el control del Agente se descargue e instale automáticamente, en caso de que no se encuentre instalado previamente en la máquina del cliente. Pero el objetivo principal de esta etiqueta es declarar la clase (*classid*) de Control del Agente para tener acceso a todos los métodos y propiedades que permiten manipular al Agente. El atributo *codeBase* se utiliza para especificar la última versión del Agente. El objeto del Agente podrá ser referenciado dentro del código del Script por medio del campo *id* de la etiqueta *OBJECT*, en este caso se utiliza como identificador la palabra *Agent*.

Para que los personajes del Agente de Microsoft puedan utilizar un idioma distinto al inglés americano, es necesario instalar el componente de lenguaje de dicho (*dll*) idioma. Sin embargo, la etiqueta de objeto para declarar el componente de lenguaje debe estar presente en el código antes que la etiqueta de control del Agente. En nuestro caso, el componente de lenguaje del idioma español será descargado e instalado automáticamente utilizando la siguiente etiqueta:

```
<OBJECT classid="CLSID:C3480C0A-A7F8-11D1-AA75-00C04FA34D72"
codeBase=#VERSION=2,0,0,0 height=0 id=AgentIntlDLL width=0></OBJECT>
```

Si deseamos utilizar una máquina TTS diferente a las compiladas por Microsoft (“TrueVoice” y “TTS3000”) para los personajes del Agente, se necesitan instalar componentes binarios de ejecución de SAPI 4 (Speech Application Programming Interface). Estos componentes permiten la ejecución de otras máquinas TTS y se pueden descargar e instalar automáticamente con la siguiente etiqueta:

```
<OBJECT classid="CLSID:0C7F3F20-8BAB-11d2-9432-00C04F8EF48F"
codeBase=#VERSION=4,0,0,0 height=0 id=SAPI width=0></OBJECT>
```

Una vez instalados los componentes de ejecución de SAPI 4, podemos utilizar la máquina TTS en el lenguaje deseado. En nuestro caso utilizamos la máquina TTS TTS3000 de Lernout & Hauspie en español de España pues aún no se cuenta con una máquina TTS en español de Latinoamérica. La etiqueta necesaria para descargar e instalar automáticamente la máquina TTS en español es la siguiente:

```
<OBJECT classid="CLSID:1D87F5B9-05F1-11d2-AD7C-0000F8799342"
codeBase=#VERSION=1,0,0,0 height=0 id=TTS3000 width=0></OBJECT>
```

Cualquier Script que se inserte dentro de una página HTML debe estar contenido dentro de las etiquetas del encabezado (<head></head>), de esta forma se logra que el Script se ejecute al momento de cargar la página y antes que se despliegue en pantalla. Todas las etiquetas anteriores deben estar insertadas en las primeras líneas del código HTML dentro de las etiquetas del encabezado, y antes de iniciar el bloque del Script correspondiente.

3.8.1.2 Declaración de Variables

Para declarar el inicio de un Script dentro de una página HTML, se deben incluir las etiquetas (<Script></Script>) e indicar el lenguaje en el que esta escrito (VBScript, JavaScript, JScript, etc.) para que el navegador sepa como interpretarlo. De esta forma, el navegador del usuario sabrá que todo lo que se escriba dentro de estás etiquetas será código de Script y que tendrá que interpretarlo. Todos los lenguajes de Scripts son interpretados en tiempo de ejecución por el motor de lenguajes de Script del navegador del cliente [Alarcón, 98]. Por lo tanto es conveniente escribir los Scripts dentro de las etiquetas del encabezado (<head></head>) de la página para que el Script se cargue antes de que se muestre la página.

```
<SCRIPT language=VBScript>
Dim Agente
```

```
Dim LoadRequestUNC
Dim LoadRequestURL
Dim GetShowAnimation
```

En el fragmento de código anterior se inicia la escritura del Script en lenguaje VBScript y se hace la declaración de variables globales que estarán accesibles para todos los métodos dentro del Script.

La variable *Agente* es una variable de Objeto y servirá para asignar el Objeto de Control del Agente de Microsoft y el personaje deseado. La segunda y tercer variable, *LoadRequestUNC* y *LoadRequestURL,* se utilizarán para asignarles un Objeto de tipo **Request** y poder seguir el estatus de la petición que carga los archivos del personaje en nuestra variable *Agente*. La última variable, *GetShowAnimation,* sirve para cargar las animaciones deseadas antes de mostrar al Agente en la página.

3.8.1.3 Métodos para Cargar al Agente

Para poder mostrar al Agente en una página HTML, primero se deben cargar los archivos del personaje y asignarlos a un Objeto de Control para poder manipular sus acciones. Debido a que los componentes del personaje se cargan al momento de ejecución de la página, es preferible escribir un método que se encargue de cargar al Agente antes de que la página se despliegue en el navegador del usuario.

```
Sub Window_OnLoad
      LoadCharacter
End Sub
```

El procedimiento anterior se ejecuta al momento de que se hace la petición de una página al servidor HTML de *Enciclomedia*, y está se carga en el navegador del usuario. Lo único que hace es llamar a la subrutina *LoadCharacter*, la cual se encargará de cargar al Agente en el navegador del cliente o usuario.

```
Sub LoadCharacter
      On Error Resume Next
      Set LoadRequestUNC = Agent.Characters.Load ("Agente",
"Peedy.acs")

      If LoadRequestUNC.Status <> 0 Then
            Set LoadRequestURL = Agent.Characters.Load ("Agente",
"http://agent.microsoft.com/agent2/chars/Peedy/Peedy.acf")

      Else
            Set Agente = Agent.Characters("Agente")
            DoIntro
      End If
End Sub
```

La subrutina *LoadCharacter* inicia generando una petición para cargar el archivo del personaje deseado (en este caso *Peedy.acs*). Esta petición crea un Objeto **Request** el cual se asigna a la variable global *LoadRequestUNC*. Este archivo ACS se encuentra localmente en el directorio de archivos de los personajes del Agente de Microsoft (C:\WINDOWS\msagent\chars). En caso de que no se encuentre el archivo, la petición regresa como valor 1 y a continuación se trata de cargar al Agente desde la dirección en Internet que contiene los archivos del personaje, pero en formato ACF. La petición por el archivo ACF genera otro Objeto **Request** que se asigna a la variable *LoadRequestURL*. Si el servidor esta ocupado, apagado, o el archivo no existe, la petición regresa un valor de 1.

Si el archivo del personaje es encontrado ya sea en el directorio local del usuario o en la dirección de Internet, entonces se asigna el Objeto de Control del Agente a la variable global *Agente* logrando con éxito cargar al Agente y poder llamar a la subrutina que hace la introducción del Agente en la página (*DoIntro*).

```
Sub Agent_RequestComplete(ByVal Request)
      If Request = LoadRequestURL Then
            If Request.Status = 1 Then
                  Msgbox "No se pudo cargar al personaje, el servidor
parece estar ocupado.  Favor de refrescar la página."
            Exit Sub
```

```
            ElseIf Request.Status = 0 Then
                  Set Agente = Agent.Characters("Agente")
                  Set GetShowAnimation = Agente.Get ("state", "showing,
speaking")
                  Agente.Get "animation", "Blink, Greet, Pleased,
Explain, Think, Read, GestureRight, Idle1_1, Idle2_2, Announce,
Uncertain", False
            End If

      ElseIf Request = GetShowAnimation Then
            If Request.Status = 1 Then
                  Msgbox "Hubo una falla en la petición para cargar la
animación Show.  El servidor parece estar ocupado."
            Exit Sub
            ElseIf Request.Status = 0 Then
                 DoIntro
            End If
      End If
End Sub
```

En caso que la petición de cargar al Agente desde el sitio en Internet de Microsoft falle, la función *Agent_RequestComplete(ByVal Request)* se activa y manda mensajes de error al usuario para informarle que no se pudo concretar la descarga del Agente. Si la petición se logró, entonces se asigna a la variable *Agente* el Objeto de Control del Agente y a continuación se utiliza el método *Get* para generar una petición, asignarla a la variable *GetShowAnimation* y obtener las animaciones requeridas. Si no se pueden obtener estás animaciones, se manda un mensaje de error al usuario para informarle de la falla. En caso contrario, las animaciones se cargan correctamente y se llama a la subrutina *DoIntro* para que el Agente haga su introducción al usuario.

3.8.1.4 Inicialización de Comandos

Cuando el Agente de Microsoft y sus componentes se han cargado correctamente en el navegador del usuario, aparece un icono del personaje seleccionado en la barra de tareas del usuario (figura 3.6).

Figura 3.6 Icono del Personaje del Agente de Microsoft.

Una vez asignado el personaje a la variable *Agente*, se hace un llamado a la subrutina *DoIntro* para que el Agente se muestre en el navegador del usuario y haga una introducción.

```
Sub DoIntro
      InitCommands
      On Error Resume Next
      Agente.MoveTo window.screenLeft+400, window.screenTop+105
      Agente.Show
      Agente.Speak "\Pit=83\ Lección de Lectura: Paco el Chato"
End Sub
```

La primera línea del procedimiento *DoIntro* hace un llamado al método *InitCommands* la cuál se encargará de iniciar los comandos y características de texto y lenguaje deseados. El método del Agente *MoveTo* hace que el personaje aparezca en un punto predeterminado en la pantalla del usuario. A continuación el personaje aparece en el punto deseado (al usar el método *Show*) y hace una pequeña introducción al decir la lección de lectura en la que se encuentra el usuario (con el método *Speak* y utilizando una etiqueta de salida de voz, como vimos en el punto 3.6.4) (figura 3.7).

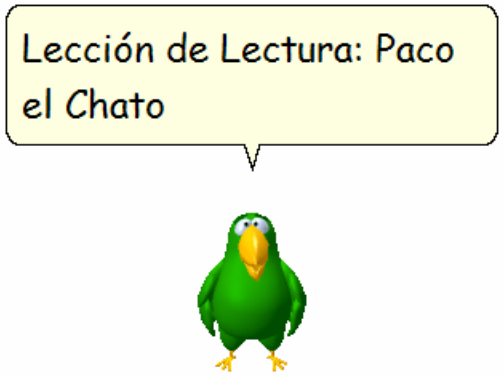

Figura 3.7 Introducción del Agente.

```
Sub InitCommands()
      Agente.LanguageID = &H0C0A
      Agente.Commands.Add "Stop", "Detener"
      Agente.Commands.Add "ACO", "Advanced Character Options"
      Agente.Balloon.Fontsize=18
      Agente.Balloon.FontName="Comic Sans MS"
End Sub
```

La subrutina *InitCommands* inicializa los valores de lenguaje, comandos y texto del Agente. Debido a que el Agente de Microsoft tiene asignado el idioma inglés americano por *default*, es necesario asignar el código del identificador de lenguaje (*LanguageID*) del idioma que deseamos usar, en este caso el código (*&H0C0A*) corresponde al español. El método *Commands.Add* permite agregar comandos particulares al Agente. Estos comandos se despliegan al oprimir el botón derecho del *mouse* sobre el personaje (figura 3.8).

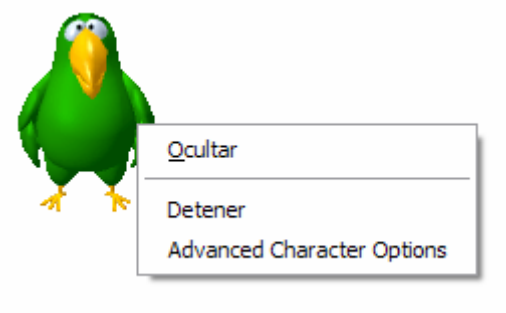

Figura 3.8 Comandos incluidos en el Agente.

Por último, para modificar el formato del texto desplegado dentro del Globo de Texto del Agente, se utiliza el Objeto *Balloon* y se asignan los valores deseados a las propiedades que se quieran modificar; en este caso el tamaño y tipo de la fuente (*FontSize* y *FontName*).

3.8.1.5 Métodos de Lectura de Texto

Para que el Agente lea una lección de lectura (o cualquier tipo de texto) se crearon tres formatos de lectura, basándome en la estructura de los Libros Interactivos desarrollados por el CSLR. Esta estructura divide los tipos de lectura de su agente o tutor en: lectura por palabra, lectura por oración, y lectura de todo el libro.

En nuestro sistema, la lectura de todo el libro se codificó en una subrutina llamada *LeerTodo()*:

```
Sub LeerTodo ()
      On Error Resume Next
      Agente.Stop
      Agente.Show
      Agente.Play "Read"

      Agente.Speak "\Pit=83\ Paco el Chato vivía en un rancho."
      Agente.Speak "\Pit=83\ Al cumplir seis años Paco debía entrar a
la escuela."
      Agente.Speak "\Pit=83\ Para eso su papá lo llevó a la ciudad,
donde vivía su abuelita."
      Agente.Play "ReadContinued"

      Agente.Speak "\Pit=83\ Al llegar a la escuela, el primer día de
clases, la abuelita le dijo: a la salida me esperas en la puerta."
      Agente.Speak "\Pit=83\ Paco esperó un rato, después empezó a
caminar y se perdió."
      Agente.Speak "\Pit=83\ Paco se asustó y empezó a llorar."
      Agente.Play "ReadContinued"

      Agente.Speak "\Pit=83\ Un policía le preguntó su nombre, su
apellido y su dirección."
      Agente.Speak "\Pit=83\ Paco no sabía ni su apellido ni su
dirección."
      Agente.Speak "\Pit=83\ El policía llevó a Paco a la estación de
radio para que avisaran que ahí estaba."
      Agente.Speak "\Pit=83\ La abuelita de Paco oyó el aviso y fue a
buscarlo."
```

```
      Agente.Speak "\Pit=83\ Paco se alegró y prometió aprender su
nombre completo y su dirección."
      Agente.Play "ReadReturn"

      Agente.Play "Pleased"
End Sub
```

La subrutina anterior está basada en la lección de lectura "Paco el Chato" que pertenece al conjunto de lecturas del Libro de Lecturas de Español Primer Grado de la SEP. La subrutina *LeerTodo()* se ejecuta al momento que el usuario oprime el botón "Leer Todo" que se encuentra a un lado del título de la lección (figura 3.9). El código inicia con un manejo de errores en el cual, al ocurrir un error, la ejecución continua con la siguiente línea de código que provoco el error. Si ningún error ocurre, se obliga al Agente a detenerse (método *Stop*) en el caso de que este realizando una acción previa y el usuario quiera que se inicie la lectura. Después se muestra al Agente (método *Show*) en el caso que el usuario haya ocultado al Agente, si lo anterior no aplica (y el Agente esta presente) no se aprecia ningún cambio pues el método verifica si el Agente esta mostrado en la pantalla del usuario.

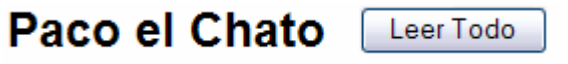

Figura 3.9 Ejemplo del botón de lectura completa.

Una vez que el Agente se encuentra en pantalla se procede a leer todo el texto de la lectura. Lo anterior se logra utilizando el método *Speak* dándole como entrada las oraciones del texto a leer (una por una), y ejecutando la animación de lectura corrida (método *Play*, atributo "*ReadContinued*"). Se podría incluir todo el texto de la lección en una sola instancia del método *Speak* pero esto ocasionaría que todo el texto se mostrara dentro del Globo de Texto del personaje, saturando la pantalla del usuario. Al final de la lectura se ejecuta la animación *"Pleased"*, dando a entender que el personaje esta satisfecho de haber leído la lectura y ayudado al usuario.

```
Sub LeerLinea (ByVal linea)
      On Error Resume Next
      Agente.Stop
      Agente.Show
      Agente.Play "Read"

      select case linea
            Case 1
                  Agente.Speak "\Pit=83\ Paco el Chato vivía en un
                  rancho."
            Case 2
                  Agente.Speak "\Pit=83\ Al cumplir seis años Paco
                  debía entrar a la escuela."
            Case 3
                  Agente.Speak "\Pit=83\ Para eso su papá lo llevó a la
                  ciudad, donde vivía su abuelita."
            Case 4
                  Agente.Speak "\Pit=83\ Al llegar a la escuela, el
                  primer día de clases, la abuelita le dijo: a la
                  salida me esperas en la puerta."
            Case 5
                  Agente.Speak "\Pit=83\ Paco esperó un rato, después
                  empezó a caminar y se perdió."
            Case 6
                  Agente.Speak "\Pit=83\ Paco se asustó y empezó a
                  llorar."
            Case 7
                  Agente.Speak "\Pit=83\ Un policía le preguntó su
                  nombre, su apellido y su dirección."
            Case 8
                  Agente.Speak "\Pit=83\ Paco no sabía ni su apellido
                  ni su dirección."
            Case 9
                  Agente.Speak "\Pit=83\ El policía llevó a Paco a la
                  estación de radio para que avisaran que ahí estaba."
            Case 10
                  Agente.Speak "\Pit=83\ La abuelita de Paco oyó el
                  aviso y fue a buscarlo."
            Case 11
                  Agente.Speak "\Pit=83\ Paco se alegró y prometió
                  aprender su nombre completo y su dirección."
      End Select
      Agente.Play "ReadReturn"
```

```
End Sub
```

La subrutina *LeerLinea(Byval linea)* se ejecuta al momento que el usuario oprime cualquiera de los botones que se encuentran al inicio de cada oración (figura 3.10). Esto dispara una acción en la que se llama al procedimiento (*LeerLinea*) y le manda el número de la línea en la que se encuentra el botón. *LeerLinea()* obtiene el valor del número en la variable *linea* y la utiliza para asignar la oración que se va a leer por medio de la estructura *Select – Case*.

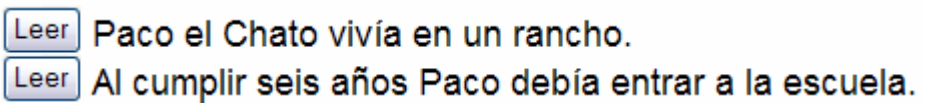

Figura 3.10 Ejemplo de los botones de lectura por oración.

La función que realiza la lectura por palabra con el Agente de Microsoft está descrita en el punto 3.8.2.2 ya que pertenece al Script de manipulación de páginas HTML.

3.8.2 Script de Manipulación de Páginas HTML

Uno de los requerimientos del sistema era tener la capacidad de ofrecer los tres formatos de lectura de los Libros Interactivos del CSLR. Uno de esos formatos era la lectura por palabra del texto contenido en una lección. Para lograrlo, se debía encontrar la forma de saber cuál palabra sería seleccionada por el usuario con sólo poner el puntero sobre ella. Lo anterior representaba un reto ya que una de las restricciones del sistema, consistía en modificar al mínimo el formato de las páginas HTML de los libros de la SEP. Por lo tanto, no se podía insertar el texto en celdas dentro de una tabla, palabra por palabra, pues esto modificaría el formato de la página. Y tampoco se podía ocupar aplicaciones (como Java Applets o JSP's) pues alteraría el formato de HTML puro de los libros.

La solución al problema anterior se encontró desarrollando un Script en lenguaje JavaScript. La utilización de JavaScript para escribir este nuevo código, obedeció a la compatibilidad de este lenguaje en los navegadores más importantes (Internet Explorer y Netscape) y a la enorme fuente de recursos disponibles. Otro punto a favor de JavaScript, fue la gran eficiencia y libertad que tiene al momento de manipular etiquetas en lenguaje HTML, lo que era básico para cumplir con el requisito de mínima alteración del formato de los Libros de Texto Gratuito de la SEP digitalizados.

3.8.2.1 Método de Separación del Texto en Palabras

Dentro de las opciones de lectura del sistema, debía estar incluida la opción de lectura por palabra. Para lograrlo era necesario convertir cada una de las palabras de un texto en un elemento independiente el cual debía ser mandado como entrada al Agente para que pudiera decir el texto contenido en ese elemento.

El usuario tenía que poder elegir la palabra que quisiera escuchar de una forma sencilla, por ejemplo, seleccionándola con el *mouse* u oprimiendo un botón. Sin embargo, incluir un botón por cada palabra en un texto no era una solución óptima, y hacer que el usuario tuviera que seleccionar la palabra y después oprimir algún botón del *mouse*, tampoco era lo deseado.

La solución se encontró identificando el texto que comprendía la lección de lectura, por medio de una etiqueta que no alterará su contenido. La etiqueta ( ) define una división lógica dentro de una página Web [Kyrnin, 2004]. Una propiedad importante de esta etiqueta es poder nombrar ciertas secciones, o divisiones, de una página Web y así poder identificarlas y manipularlas fácilmente. Estos factores cumplen con los requisitos que el sistema necesitaba por lo que se utilizó la etiqueta ( ) para separar el texto de una lección de lectura, dentro de la página Web en que estuviera contenida.

```
window.onload=function() {
separarPalabras(document.getElementsByTagName('div')[0]); }
```

La línea de código anterior realiza tres acciones:

1) Se ejecuta en el momento que se carga la página del lado del cliente o usuario (*window.onload*);
2) Llama a la función (*separarPalabras*);
3) Manda como parámetro de entrada para la función (*separarPalabras*) el valor de regreso del método (*getElementsByTagName*).

El método (*getElementsByTagName*) es muy útil para los fines del sistema pues regresa una colección de Objetos (de tipo *nodeList*) basándose en el nombre de la etiqueta asignada como parámetro (en este caso, la etiqueta *div*). Esto quiere decir que obtendremos un conjunto de nodos en el cual, cada nodo tendrá un elemento encontrado dentro de la etiqueta ( ).

```
function separarPalabras(elem) {
      var num=elem.childNodes.length;

      for(var i=0; i<num; i++) {
            var text=elem.firstChild.data;
            var oc = elem.removeChild(elem.firstChild);

            if(oc.nodeType==3) {
                  textArray=text.split(" ");
                  for(var j=0; j<textArray.length; j++) {
                        var link=document.createElement("A");
                        link.onclick=function() {
decirPalabra(this.firstChild.data); }
                        link.on mouse over=function ()
{this.style.color='blue'};
                        link.on mouse out=function ()
{this.style.color='black'};
                        link.className="plain1";
                        var ltxt=document.createTextNode(textArray[j]+"
");
                        link.appendChild(ltxt);
                        elem.appendChild(link);
                  }
```

```
            }
            else {
                  separarPalabras(oc);
                  elem.appendChild(oc);
            }
      }
}
```

La función *separarPalabras()* recibe la colección de nodos (*nodeList*), con los elementos encontrados dentro de la etiqueta (`<div>`) guardados en cada nodo hijo de la colección de nodos. Los elementos encontrados por *getElementsByTagName* se basan en la jerarquía de la plataforma DOM (*Document Object Model*), cada nodo dentro de la colección obtenida es de diferente tipo (*ELEMENT_NODE, TEXT_NODE, COMMENT_NODE, etc.*). En nuestro caso, el tipo de nodo que buscamos es el de texto y es identificado por el número 3, pero no sabemos en que parte de la colección de nodos se encuentra. Para encontrar el nodo que contiene el texto que nos interesa, se realiza una búsqueda recursiva (primer *for()*) en la cual se hace una comparación del primer nodo en la colección (*elem.firstChild*) para verificar si es de tipo texto (*oc.nodeType==3*).

Al momento de encontrar el nodo con el texto que nos interesa, se crea un arreglo conteniendo cada palabra encontrada dentro del texto (utilizando el método *split(" ")* para conseguirlo). Después, cada palabra se convierte en un link (*var link=document.createElement()*) y se le asignan propiedades de color, acciones cuando el puntero del *mouse* esta sobre él y fuera de él, y el llamado a la función *decirPalabra()* cuando se oprime el botón izquierdo del *mouse* sobre el link.

Al final el texto de la lección no se ve modificado en su formato (pues los links carecen de formato (*link.className=plain1*)), cada palabra es un elemento independiente dentro de la página, y se puede tener acceso a ellas y manipularlas.

3.8.2.2 Método de Lectura de Palabras

Gracias a la eficiencia en la manipulación de etiquetas en HTML, logramos separar cada palabra de la lección de lectura por medio de la función *separarPalabras()*. Ahora tenemos cada palabra identificada y separada por medio de links a los cuales les incorporamos atributos para manipularlas. Cuando el usuario oprima el botón izquierdo del *mouse* sobre alguna de las palabras, se hará un llamado a la función *decirPalabra()*.

```
function decirPalabra(palabra) {
      Agente.LanguageID = "&H0C0A";
      Agente.Stop();
      Agente.Show();
      Agente.Speak (palabra);
}
```

La función anterior es sumamente sencilla. Recibe la palabra que el usuario desea escuchar al oprimir el botón izquierdo del *mouse* sobre ella, y se le manda al Agente para que la diga. Pero antes de decirla, nos aseguramos que el Agente tenga asignado el lenguaje correcto (en nuestro caso, español; *Agente.LanguageID=”&H0C0A”*), se obliga al Agente a detenerse (en caso que este realizando una acción previa), se verifica que el personaje aparezca en la pantalla del usuario (*Agente.Show*), y por último se le manda la palabra para que la diga (*Agente.Speak (palabra)*).

3.9 Integración del Sistema en Páginas HTML

El sistema desarrollado puede funcionar en cualquier página HTML deseada, siempre y cuando lo integremos de forma correcta. En los siguientes puntos detallaré la forma en que se deben integrar los elementos que componen al sistema dentro de cualquiera página HTML que queramos usar.

3.9.1 Insertar Etiquetas de Objetos

Supongamos que tenemos la siguiente página HTML a la cual queremos integrar el sistema:

```
<html>
<head>
<title>Página de ejemplo para integrar el sistema</title>
</head>
<body>
<font face="arial, helvetica" size=+2><b>Página de
Ejemplo</b></font><font face="arial, helvetica" size=+1</font><br><br>
Este es el texto que queremos que el Agente lea para nosotros.<br>
Es muy sencillo integrar este sistema a cualquier página HTML que
queramos.<br>
</body>
</html>
```

La página anterior es muy sencilla y sólo tiene dos oraciones como contenido y un título, sin embargo el procedimiento para integrar el sistema se aplica de igual forma sin importar la complejidad de la página.

Lo primero que necesitamos integrar son las etiquetas de Objeto que permiten el acceso a las interfaces de control del Agente de Microsoft.

```
<html>
<head>
<title>Página de ejemplo para integrar el sistema</title>

<OBJECT classid="CLSID:C3480C0A-A7F8-11D1-AA75-00C04FA34D72"
codeBase=#VERSION=2,0,0,0 height=0 id=AgentIntlDLL width=0></OBJECT>
<OBJECT id=Agent codeBase=#VERSION=2,0,0,0 classid=clsid:D45FD31B-5C6E-
11D1-9EC1-00C04FD7081F></OBJECT>
<OBJECT classid="CLSID:0C7F3F20-8BAB-11d2-9432-00C04F8EF48F"
codeBase=#VERSION=4,0,0,0 height=0 id=SAPI width=0></OBJECT>
<OBJECT classid="CLSID:1D87F5B9-05F1-11d2-AD7C-0000F8799342"
codeBase=#VERSION=1,0,0,0 height=0 id=TTS3000 width=0></OBJECT>
```

```
</head>
<body>
      .
      .
      .
</body>
</html>
```

Como vimos en el punto 3.8.1.1, es importante recordar que las etiquetas de Objeto se insertan en el orden indicado y dentro de las etiquetas del encabezado de la página (<head></head>), para que se carguen antes de que la página se despliegue en la pantalla.

3.9.2 Insertar Script de Agente

Para insertar el Script que controla las funciones del Agente debemos asegurarnos que las etiquetas de Objeto ya están integradas dentro de las etiquetas de encabezado. Si esto se ha completado, proseguimos a insertar el Script del Agente enseguida de la última etiqueta de Objeto declarada.

```
<html>
<head>
<title>Página de ejemplo para integrar el sistema</title>

<OBJECT>Etiquetas de Objetos</OBJECT>
      .
      .
      .
<SCRIPT language=VBScript>

Sub DoIntro
      .
      .
      .
      Agente.Speak "\Pit=83\ Integración del sistema a una página HTML"
End Sub

Sub LeerTodo ()
      .
      .
      .
    Agente.Speak "\Pit=83\ Este es el texto que queremos que el Agente
lea para nosotros."
    Agente.Speak "\Pit=83\ Es muy sencillo integrar este sistema a
cualquier página HTML que queramos."
      .
```

```
      .
      .
End Sub

Sub LeerLinea (ByVal linea)
      .
      .
      .
      select case linea
        Case 1
                  Agente.Speak "\Pit=83\ Este es el texto que queremos
que el Agente lea para nosotros."
        Case 2
                  Agente.Speak "\Pit=83\ Es muy sencillo integrar este
sistema a cualquier página HTML que queramos."
      End Select
      .
      .
      .
End Sub
</SCRIPT>
</head>
<body>
      .
      .
      .
</body>
</html>
```

En el código anterior están marcadas en negritas únicamente las partes que se necesitan cambiar para adaptar el sistema a la página deseada. En la práctica es necesario insertar el Script completo y modificar las partes señaladas. Básicamente lo que se tiene que cambiar es lo siguiente:

1) La introducción que queramos que el personaje haga al momento de aparecer en pantalla;
2) El texto completo para que el Agente lo lea por medio de la función *LeerTodo()*;
3) El texto de cada oración que el Agente leerá al ejecutar la subrutina *LeerLinea()* y aumentar o disminuir el número de casos por oración (*Case n*).

Ahora sólo nos falta insertar el Script que se encarga de la separación del texto por palabras y dividir la sección en la que se encuentra el texto que deseamos utilizar.

3.9.3 Insertar Script de Manipulación de Páginas HTML

El Script que convierte el texto de una página en links separados por palabras se puede insertar inmediatamente después del Script del Agente o antes que él. Esto no modifica su comportamiento y nos da flexibilidad para dar claridad de lectura a nuestro código.

```
<html>
<head>
<title>Página de ejemplo para integrar el sistema</title>

<OBJECT>Etiquetas de Objetos</OBJECT>
      .
      .
      .
<SCRIPT language=VBScript>
      .
      .
      .
</SCRIPT>

<Script type="text/javaScript">

function separarPalabras(elem) {
      .
      .
      .
}

function decirPalabra(palabra) {
      .
      .
      .
}

window.onload=function() {
separarPalabras(document.getElementsByTagName('div')[0]); }

</Script>

</head>
<body>
      .
      .
      .

</body>
</html>
```

Este Script no necesita mayores cambios pues se ajusta a cualquier página HTML con sólo incluir dentro de las etiquetas ( ) el texto que deseamos manipular, lo cual veremos en el siguiente punto. En la práctica se debe insertar el Script completo con

todas sus funciones dentro de las etiquetas de encabezado de la página, para que se cargue antes de que la página se muestre en la pantalla del usuario. De esta forma el resultado final que verá el usuario es la página con el texto convertido a links.

3.9.4 Insertar Elementos de HTML

Una de las finalidades de este sistema es que se hagan los menores cambios posibles al formato de la página HTML que deseamos integrar. Como veremos a continuación, los cambios necesarios que visualmente serán notados por el usuario son mínimos.

```
<html>
<head>
<title>Página de ejemplo para integrar el sistema</title>

<OBJECT>Etiquetas de Objetos</OBJECT>
      .
      .
      .
<SCRIPT language=VBScript>
      .
      .
      .
</SCRIPT>

<Script type="text/javaScript">
      .
      .
      .
</Script>

</head>
<body>
<div id="texto">
<font face="arial, helvetica" size=+2><b>Página de
Ejemplo</b></font><font face="arial, helvetica" size=+1>
<INPUT class=button75 id=cmdLeer onclick=LeerTodo() type=button
value="Leer Todo" name=cmdLeer></font><br><br>
<INPUT class=button75 id=cmdLeer1 onclick=LeerLinea(1) type=button
value="Leer" name=cmdLeer1> Este es el texto que queremos que el Agente
lea para nosotros.<br>
<INPUT class=button75 id=cmdLeer2 onclick=LeerLinea(2) type=button
value="Leer" name=cmdLeer2> Es muy sencillo integrar este sistema a
cualquier página HTML que queramos.<br>
</div>
</body>
<html>
```

Son sólo dos cosas que se deben cambiar al formato de la página HTML: a) Separar el texto con el cual Agente trabajará, por medio de la utilización de etiquetas (`<div></div>`) y; b) Insertar los botones que disparan las acciones de lectura del Agente, *LeerTodo()* y *LeerLinea()*. En el caso de los botones de lectura por oración, se debe cambiar el identificador de cada botón (*id=cmdLeerN*) y asignarle el número apropiado para que se mande como parámetro de la función *LeerLinea(n)*.

Finalizando los puntos antes mencionados, tendremos una nueva herramienta funcionando en nuestra página. El Agente de Microsoft nos brindará una mayor interacción con el usuario, y un acercamiento más humano a las tareas asignadas por el usuario.