UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

MOUSE PARA MINUSVÁLIDOS

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: 20 Mención Publicación

J U R A D O E X A M I N A D O R

Firma:	Firma:	Firma:
Nombre: Biagio Cante	Nombre: Pedro Castillero.	Nombre: JUAN NUÑEZ

REALIZADO POR	Fazzino, Michelangelo Sánchez, Oswel
PROFESOR GUIA	Ing. Cante, Biagio
FECHA	28-10-2003

ÍNDICE GENERAL

Pág.

SINOPSIS 1

INTRODUCCIÓN 2

CAPITULO I: PROBLEMA DE INVESTIGACIÓN 4

1.1. Planteamiento del Problema 4

1.2. Objetivo general 5

1.3. Objetivos específicos 5

1.4. Justificación de la Investigación 6

1.5. Limitaciones y alcances 7

CAPITULO II: MARCO TEÓRICO 8

2.1. Antecedentes de la investigación 8

2.2. El mouse 11

2.3. Discapacidad 13

2.3.1. Discapacidad motora 14

2.3.1.1. Alteraciones de la coordinación 14

2.3.1.1.1. Mal de Parkinson 15

2.3.1.1.2. Distonía muscular 15

2.3.1.2. Déficit de movimiento 15

2.3.1.3. Alteración estructural o ausencia de miembros 16

2.4. Análisis de imágenes 16

2.5. Segmentación de imágenes 17

2.6. Sistemas de reconocimiento de voz 18

2.7. Java Media Framework (JMF) 19

2.8. Java Speech API (JSAPI) 22

CAPITULO III: MARCO METODOLÓGICO 26

3.1. Proceso de desarrollo del software 26

3.2. Proceso unificado 26

3.2.1. Dirigido por casos de uso 27

3.2.2. Centrado en la arquitectura 28
3.2.3. Iterativo e incremental 29
3.2.4. Ciclo de vida del proceso unificado 30
3.2.4.1. Fase dentro de un ciclo 32
3.2.4.1.1. Inicio 33
3.2.4.1.2. Elaboración 33
3.2.4.1.3. Construcción 33
3.2.4.1.4. Transición 34
3.2.4.2. Flujo de trabajo 34
3.2.4.2.1. Captura de requisitos 34
3.2.4.2.2. Análisis 35
3.2.4.2.3. Diseño 36
3.2.4.2.4. Implementación 37
3.2.4.2.5. Pruebas 38

CAPITULO IV: DESARROLLO 41
4.1. Captura de requisitos 41
4.2. Análisis 43
4.3. Diseño 45
4.4. Implementación 49
4.4.1. Captura de imágenes 51
4.4.2. Procesamiento de imágenes 52
4.4.3. Procesamiento de la voz 53
4.4.4. Manejo del mouse 54
4.4.4.1. Movimiento del puntero del mouse 56
4.4.4.2. Presionar el botón izquierdo del mouse 56
4.4.4.3. Presionar el botón derecho del mouse 56
4.4.4.4. Presionar 2 veces el botón izquierdo del mouse ... 57
4.4.4.5. Dejar botón izquierdo del mouse presionado 57
4.4.4.6. Soltar el botón izquierdo del mouse 57
4.4.4.7. Dejar botón derecho del mouse presionado 57

4.4.4.8. Soltar el botón derecho del mouse 58

4.5. Pruebas 58

CAPITULO V: RESULTADOS 63

5.1. Alcance de los objetivos específicos 63

5.2. Condiciones de uso 64

CONCLUSIONES 65

RECOMENDACIONES 66

BIBLIOGRAFÍA 67

APÉNDICE A..... 69

APÉNDICE B..... 72

APÉNDICE C..... 83

ANEXOS 86

ÍNDICE DE TABLAS Y FIGURAS

TABLAS Pág.

4.1 Requisitos obtenidos del mouse 42
4.2 Obstáculos encontrados ..... 43 y 44
4.4 Casos de uso 45
4.10 Desplazamiento del puntero del mouse 53
4.11 Eventos del mouse 56

FIGURAS Pág.

2.1 Componentes en una aplicación con un sistema de reconocimiento de voz 19
3.1 Un proceso de desarrollo del software 26
3.2 Los casos de uso enlazan los flujos de trabajo fundamentales 28
3.3 Los casos de uso conducen el desarrollo de la arquitectura, y la arquitectura indica qué casos de uso pueden realizarse..... 29
3.4 Un ciclo con sus fases e interacciones..... 31
3.5 Modelo del proceso unificado..... 31
3.6 Los cinco flujos de trabajo..... 32
3.7 El trabajo de los requisitos se hace fundamentalmente durante el inicio y la elaboración 35
3.8 El trabajo del análisis se hace fundamentalmente durante la elaboración..... 36
3.9 El trabajo de diseño se hace fundamentalmente durante la elaboración y la construcción..... 37
3.10 El trabajo de implementación se hace fundamentalmente durante la construcción..... 38
3.11 Las pruebas se hacen durante todas las fases, pero mayormente durante la construcción y la transición..... 39
4.3 Modelo general de casos de usos..... 44
4.5 Diagrama de clases del sistema 46

4.6 Diagrama de secuencia de comandos de voz..... 47
4.7 Diagrama de secuencia del movimiento del puntero..... 48
4.8 Diagrama de componentes..... 49 y 50
4.9 Direcciones de movimientos 52

SINOPSIS

El presente trabajo especial de grado, está inspirado en ayudar a aquellas personas que por alguna razón presentan problemas motores o de discapacidad. La idea fundamental, es que puedan interactuar con las computadoras mediante un mouse, para que tengan la oportunidad de realizar las cosas que a diario realizan todos, y puedan integrarse más en los procesos de nuestra estructura social. Esto a través de una interfaz que permita a los usuarios hacer uso de las funciones de un mouse, apoyándonos en el uso de una webcam y un micrófono.

Para llevar a cabo dicho propósito, se usó la metodología del Proceso Unificado, que guió todo el proceso de desarrollo del sistema final. Además, se utilizaron como herramientas de apoyo, los diagramas de UML y el lenguaje de programación Java versión 1.3.

Durante todo el tiempo de desarrollo del sistema, se tuvo en mente, que la comodidad y el confort eran características que debían estar presentes en el producto final. Lo que obligó, a realizar estudios sobre la movilidad de las personas minusválidas, con la finalidad de determinar cuáles partes del cuerpo serian consideradas para realizar el prototipo del mouse.

Más adelante se explicará con mayor detalle cómo se desarrolló el producto, "mouse para minusválidos", desde sus inicios en el planteamiento del problema hasta la creación total del sistema.

INTRODUCCIÓN

La humanidad en su evolución ha experimentado tres grandes cambios socio-económicos, orientados hacia la búsqueda de una mejor riqueza, ellos son: la revolución agrícola, la revolución industrial y la revolución de la información.

Con la aparición de los computadores y las telecomunicaciones, a finales del siglo XX, se desarrolla la revolución de la información. Su principal recurso: la tecnología de computadores, que extrae, refina y moldea los datos crudos y los transforma en información, y a través de evaluaciones y análisis consecuentes, se convierten en conocimiento.

Sin embargo, la mayoría de las tecnologías informáticas no han tomado en cuenta a las personas con problemas motores o de discapacidad, por lo que, su intervención dentro de la estructura social es muy poca. Entre estas tecnologías se encuentra el mouse, que se convirtió en la parte fundamental de una PC pero agregó un nuevo problema a quienes no tienen la capacidad física de manejarlo y, además, prácticamente inutiliza una de las manos de los usuarios convencionales. Por eso, desde hace años se viene trabajando en tecnologías alternativas de manejo de datos que utilizan elementos del rostro para ejecutar tareas en la PC.

A continuación se describirá como fue la realización de este trabajo especial de grado, iniciando desde el capítulo I, problema de investigación, donde se explicará el

problema de la tesis, cuál es su objetivo principal y cuáles son los específicos, cuál es la justificación y cuáles son las limitaciones y alcance de la misma.

Luego en el capítulo II se expondrá el marco teórico, se mencionarán los antecedentes de la investigación y la teoría en la que se fundamenta esta tesis.

Más adelante en el capítulo III se detallará el marco metodológico, el cual contiene los pasos a seguir para la realización de esta tesis, se describen los diagramas y métodos a seguir.

Luego en el capítulo IV se definirá el desarrollo, de cómo se llevaron a cabo los pasos de la metodología para la culminación de la tesis y de la implementación del código fuente para la creación del producto final.

Finalmente en el capítulo V, se mostrarán los resultados de la tesis, como se alcanzaron los objetivos específicos y cuáles son las condiciones de uso del sistema.

Nota: Las palabras que están en negritas a lo largo del todo el documento, pueden consultarse en el glosario del Apéndice C.

PROBLEMA DE INVESTIGACIÓN

1.1 Planteamiento del Problema

Las computadoras junto al factor humano, forman el recurso esencial de la revolución tecnológica de la información, y se evidencia su intervención a diario cuando se realizan los servicios de correo, búsquedas de información en Internet, al momento de utilizar herramientas **software** para la elaboración de un trabajo o para jugar en el tiempo libre. De igual forma, en el mundo de los negocios, se puede observar que el impacto de las computadoras ha sido considerable y más en la actualidad gracias a los avances en **hardware**, los cuales han permitido construir máquinas con mayor velocidad de procesamiento.

Sin embargo, así como esta revolución informática ha mejorado las condiciones de vida, tanto en el plano personal como en el organizacional y el social, también ha dejado por fuera a muchas personas que, por diversos problemas físicos, se les hace difícil interactuar con dichas tecnologías. En su mayoría, las nuevas tecnologías, requieren del contacto directo del usuario para funcionar y las personas con discapacidades motoras, pero mentalmente sanas, se sienten excluidas de ese mundo que podría brindarles el realizar funciones sin la continua presencia de un asistente.

Actualmente, existe una diferencia muy notable entre las oportunidades que tienen las personas con problemas motores para interactuar con el mundo computacional, comparadas con las personas que no presentan problemas de salud. Ellos, estarían muy agradecidos de que la tecnología computacional les brindara nuevos

avances que les permitan integrarse de manera más activa en los procesos que se llevan a cabo en nuestra estructura social. ¿Será posible desarrollar un tipo de mouse o ratón que les permita a las personas con discapacidades motoras interactuar con la tecnología en computación?

En repuesta a este problema, surge la idea de desarrollar este trabajo especial de grado con la finalidad de permitir la interacción con la computadora a estas personas, y así poder acercarlas un poco al mundo informático que tanto ayuda a diario.

1.2 Objetivo General

Realizar un mouse que permita a personas con limitaciones físicas (minusválidas) interactuar con la computadora.

1.3 Objetivos Específicos

- Conocer cual es la movilidad de las personas con limitaciones físicas (Operadores finales del sistema).
- Conocer a fondo las funciones del mouse y cómo se llevan a cabo.
- Buscar información de cómo enviar los datos del **hardware** al computador.
- Determinar barreras, inconvenientes y obstáculos (¿Cómo eliminarlos o minimizarlos?).
- Realización del proyecto final, con sus respectivos manuales de usuario.

1.4 Justificación

Una de las cualidades que ha caracterizado al ser humano a lo largo de su historia, es la curiosidad que siente por las cosas desconocidas, cualidad que sin duda ha contribuido al desarrollo del fascinante mundo del descubrimiento y la creación.

La computadora es una de las tantas creaciones del hombre, y ha sido objeto de estudio desde hace décadas, con el fin de mejorarla para obtener el mayor rendimiento posible. Sin embargo, todavía sigue siendo objeto de estudio debido a que la tecnología avanza a una gran velocidad y las computadoras deben adaptarse a dichos cambios, para poder mantenerse en los mercados competitivos de nuestros días.

Actualmente, las computadoras despiertan una gran curiosidad en aquellas personas que las observan por primera vez, o cuando escuchan las cosas que éstas pueden realizar, o cuando tienen su primer contacto con ellas. Esta curiosidad es la que tarde o temprano conduce a que el hombre busque su primera interacción con este medio, luego de esto, se abren las puertas del apasionante mundo de la computación.

Proveer un recurso mediante el cual las personas minusválidas puedan interactuar con las computadoras, parece una gran oportunidad para que puedan hacer uso de las máquinas, tal como lo hacen muchas personas a diario. Se les abrirían así otras posibilidades de participar más activamente, como por ejemplo, en empleos, intercambio de conocimientos entre otros.

1.5 Limitaciones y Alcances

El producto va dirigido especialmente hacia personas que tengan movilidad de la cabeza, que puedan hablar y posean el sentido visual. El mouse cumplirá con todas las funciones de un mouse común.

MARCO TEÓRICO

2.1 Antecedentes de la Investigación

A continuación se señalan algunos estudios realizados que aportan información significativa que orientarán la presente investigación:

Un programador español[1] diseñó un programa que permite capturar imágenes de una **webcam** con Java. Estas imágenes las muestra en una **ventana** y permite capturar una foto y guardarla en la computadora. Una de las características más importantes de este trabajo es la posibilidad de interactuar con una **webcam** mediante sus **controladores** usando Java.

El sistema NOUSE[2], desarrollado por el Instituto de Tecnología de la Información de Canadá, forma parte de un proyecto más amplio que reúne otras investigaciones sobre manejo alternativo de datos, orientado especialmente a facilitar el uso de la PC a personas con diversas discapacidades físicas.

Hasta ahora eran los ojos y la rotación de la cabeza o del cuerpo los más investigados, pero ahora parece que la nariz toma la delantera al desarrollarse un sistema que, prácticamente, la utiliza como **puntero** para ejecutar los movimientos del mouse, por eso se bautizó al sistema como Nouse ("NOuse as moUSE" o "Nariz como mouse",

[1] MONDELO, José. Home page. <http://weblogs.javahispano.org/page/mondelo>.
[2] Nouse (Nose as Mouse) Perceptual Vision Technology, <http://www.cv.iit.nrc.ca/research/Nouse/index2.html>.

en español) y no necesita de ninguna prótesis o aparato sino simplemente una cámara conectada a la PC.

El concepto que usan los investigadores es el de "interfaces perceptuales" (Perceptual user interfaces) que significa usar movimientos del usuario (por ejemplo los de la cara) para controlar un programa. "A diferencia de otras partes del cuerpo, la cara no tiene movimientos rígidos por lo que no resultan viables las técnicas usuales", por eso decidieron utilizar la punta de la nariz como eje de los movimientos al ser el punto más cercano a la cámara que enfrenta el rostro y así tomarla como **puntero** del mouse.

Las principales características que sobresalen de este trabajo son:

- El poder utilizar la **webcam** como medio para mover el **puntero** del mouse.
- El utilizar como **puntero** del mouse la punta de la nariz.
- El utilizarlo para personas con dificultades para manejar un mouse convencional.

CameraMouse Inc.[3] creció fuera del Master de Ciencia en Ciencia y Comercialización Tecnológica, programa de la Universidad de Texas en Austin. La tecnología se desarrolló en la Universidad de Boston dónde los profesores Jim Gips y Margrit Betke tienen una larga historia de trabajar para adiestrar a las personas con problemas motores.

[3] CameraMouse hands free computer control. 2003 <http://www.cameramouse.com>.

CameraMouse™ es un software manos libres para el mando de la computadora. Usando una cámara normal de **puerto USB**, los usuarios pueden controlar los movimientos del **puntero** y realizar acciones de manera precisa con movimientos simples de la cabeza o del dedo.

El **software** rastrea los movimientos del cuerpo (la cabeza, por ejemplo), y convierte esos movimientos en los movimientos del **puntero** en la pantalla del computador. Una barra de herramientas innovadora incorporada le permite a los usuarios emular todo el mouse.

CameraMouse no posee ningún alambre, puntos, rayos infrarrojos, u otros aparatos para su activación. Cualquiera con el movimiento controlado de su cabeza o de un dedo puede operar CameraMouse. El **software** puede personalizarse y puede ajustarse para cada usuario y puede diseñarse para acomodar grados variantes de mando.

Las características más importantes de este **software** son:

- El hecho de poder utilizar una **webcam** de **puerto USB** para mover el **puntero** del mouse.
- El utilizar una barra de herramientas que posibilita realizar las funciones del mouse y ajustarlo a las medidas del usuario.
- El utilizarlo para personas con problemas motores.

2.2 El mouse[4]

El mouse o ratón, fue inventado por Douglas Engelbart en el Centro de Investigación de Stanford en 1963 y empezado a utilizar con fines comerciales por Xerox en los años 70. Esta invención representó un gran avance en el campo de las aplicaciones **software** ya que facilitó enormemente la comunicación con el usuario, permitiendo el desarrollo de programas basados en entornos gráficos.

Existen muchos tipos de mouse (mecánicos, opto-mecánicos, ópticos, inalámbricos) que se conectan con el PC de variadas formas (puerto serie, puerto PS/2, etc.). En 1983 Microsoft definió una interfaz de funciones para su mouse (MS-MOUSE) con el MS-DOS que posteriormente se ha establecido como un estándar para todos los **controladores** de mouse bajo dicho sistema operativo.

Gracias a esta interfaz, un programa de aplicación que funcione bajo MS-DOS no tiene por qué conocer el protocolo de comunicación con el **hardware** del mouse y ni siquiera se debe preocupar por la forma en el que el mouse se conecta con el PC (COM1, COM2 ó PS/2 generalmente).

La posición que ocupa el mouse en la pantalla se define siempre en coordenadas gráficas, aunque se esté trabajando en modo texto, considerando tantos puntos en la pantalla como permita la tarjeta gráfica. Es importante destacar, que la zona sobre la que trabaja el mouse no coincide con la pantalla real. El mouse se mueve sobre una *pantalla*

[4] [en línea]. Disponible en: <http://atc.urg.es/~jbernier/asignaturas/perifericos/practicas/practica2.pdf> [25-07-2003].

virtual, que depende del modo de video en el que se esté trabajando, es decir, de la **resolución**. En las funciones básicas de manejo del mouse, referidas a pantalla de texto, se hace necesario traducir estas coordenadas virtuales a coordenadas de texto; la cual se consigue dividiendo entre ocho las coordenadas gráficas, pues cada carácter ocupa ocho **píxeles** de pantalla.

El desplazamiento del **puntero** se realiza mediante una unidad de movimiento del mouse que recibe el nombre de mickey y equivale a 1/200 pulgadas. Cuando este se mueve, el **controlador** (el programa controlador) a su vez, mueve el **puntero** horizontal y verticalmente un número de **píxeles** que depende de la sensibilidad del mismo. Por defecto, cuando se instala el **controlador** del mouse para conseguir un desplazamiento de ocho (8) **píxeles** se precisan ocho (8) mickeys en horizontal y 16 en vertical. El control del mouse se realiza mediante la interrupción (int 0x33).

De esta forma, el **controlador** propio de cada mouse se instala en el sistema operativo, redireccionando el puntero de la interrupción (int 0x33) a una rutina propia que implementa una serie de funciones, para que cualquier programa gestione el mouse sin necesidad de acceder al nivel **hardware**. Cada una de estas funciones se caracterizan por el valor que se le asigne al **registro** AX cuando se llame a la interrupción (int 0x33).

2.3 Discapacidad[5]

El término "discapacidad" fue aceptado por la Real Academia Española hace 10 años. Existen otros términos quizás más comunes como "incapacidad", "minusválido", "inválido" pero estos pueden dar a entender que las personas con discapacidad son personas "sin habilidad", "de menor valor" o "sin valor". En comparación con estas acepciones, la discapacidad tiene que ver con "la disminución de una capacidad en algún área específica", por lo que el uso de este término reconoce que todos los individuos con discapacidades tienen mucho que contribuir a nuestra sociedad. Sin embargo se hace necesario diferenciar los conceptos de:

Deficiencia: "es toda pérdida o anormalidad de una estructura o función psicológica, fisiológica o anatómica". O bien, "es una alteración anatómica o funcional que afecta a un individuo en la actividad de sus órganos."

Discapacidad: "es la consecuencia de una deficiencia, sobre las actividades físicas, intelectuales, afectivo-emocionales y sociales" o también se la puede definir como "toda restricción o ausencia (debido a una deficiencia) de la capacidad de realizar una actividad en la forma o dentro del margen que se considera normal para un ser humano."

La Organización Mundial de la Salud (**O.M.S.**) define la discapacidad como "la pérdida de la capacidad funcional secundaria, con déficit en un órgano o función, y que

[5] Discapacidad. Que es?. 2002. Disponible en: <http://www.cilsa.com.ar/web_espanol/discapacidad/que_es.asp>.

trae como consecuencia una minusvalía en el funcionamiento intelectual y en la capacidad para afrontar las demandas cotidianas del entorno social."

Existen cuatro principales tipos de discapacidad, clasificadas según los ámbitos del ser humano al que afectan:

- Discapacidad Intelectual.
- Discapacidad Motora.
- Discapacidad Auditiva.
- Discapacidad Visual.

2.3.1 Discapacidad motora[6]

Se define como discapacidad física o motora a la dificultad para realizar actividades motoras convencionales. Esta dificultad puede deberse a múltiples causas, sean congénitas o adquiridas y se dividen en:

2.3.1.1 Alteraciones de la coordinación[6]

Son enfermedades donde está conservada la movilidad pero no el control o la efectividad del movimiento. Se producen por alteraciones del encéfalo que pueden ser estructurales, farmacológicas, tumorales, etc.

[6] Discapacidad. Discapacidad Motora. 2002. Disponible en: <http://www.cilsa.com.ar/web_espanol/discapacidad/motora.asp>.

2.3.1.1.1 Mal de Parkinson[6]

El mal de Parkinson es una enfermedad neurodegenerativa paulatina y progresiva que se caracteriza por presentar temblores en el reposo, anormalidades en la marcha y rigidez. Afecta al 1% de las personas mayores de 60 años. Está causado por una degeneración celular de los ganglios de la base cerebral, los que están encargados de coordinar los movimientos posturales y la motricidad fina.

2.3.1.1.2 Distonía Muscular[6]

Consiste en permanentes contracciones involuntarias de los músculos de una o más partes del cuerpo, que a menudo se manifiesta como torsiones, movimientos repetitivos o deformaciones de esa parte del cuerpo. Estos desórdenes musculares no afectan las otras funciones del cerebro, como la personalidad, la memoria, las emociones, los sentidos y la capacidad intelectual.

2.3.1.2 Déficit de movimiento[6]

Según los grados

- Plejía: Consiste en la abolición del movimiento
- Paresia: Es la disminución de la fuerza muscular.

Según la cantidad de miembros afectados

- Monoplejía: Es la parálisis de una sola extremidad
- Hemiplejía: Es la parálisis de un brazo y una pierna del mismo lado.
- Diplejía: Es la parálisis de dos partes correspondientes situadas en lados opuestos del organismo, tales como ambos brazos.

[6] Ident.

- Paraplejía: En este caso ambas piernas están comprometidas.
- Cuadriplejía o tetraplejía: Es la parálisis de las cuatro extremidades.

2.3.1.3 Alteración estructural o ausencia de miembros[6]

- **Congénitas:** Malformaciones durante el embarazo producto de la ingesta de diferentes sustancias.
- **Adquiridas:** Amputaciones. Lesión del cartilago de crecimiento.

2.4 Análisis de imágenes[7]

Sobre una imagen capturada se pueden hacer diferentes métodos de DSP (Digital Signal Processing), y basándose en estos métodos identificar patrones específicos de la imagen.

También, es factible el uso de operaciones aritméticas como substracción de color y herramientas de probabilidad y estadística tales como la correlación, con el fin de determinar movimientos o cambios en una imagen.

El color es definido como una experiencia en la percepción humana, es decir, como fenómeno físico, el color es el resultado de la incidencia de un haz de luz en la retina del ojo humano.

[6] Ident.
[7] CASTILLO, Carlos, CLAVIJO, Carlos y CORREDOR, Edgar. Marco teórico. En Sistema automático para localización de rostros en imágenes de video. [en línea]. Disponible en: <http://venus.javeriana.edu.co/tgrado/99-2/Rostros.PDF>.

En 1931, la Comisión Internacional de Iluminación (CIE), dividió en grupos primarios y generó funciones de colores. Estas funciones se utilizan para describir cualquier color visible en términos de colores primarios.

Existen muchos métodos de visualización que se desprenden del sistema desarrollado por la CIE. Estos subconjuntos reciben el nombre de Espacios de Color. Los espacios de color más usados en las diferentes aplicaciones son: RGB, YIQ y HSV.

En el espacio de color RGB, que son las siglas en inglés de los colores rojo, verde y azul respectivamente (Red, Green, Blue), una foto está compuesta por una serie de **píxeles**, cuya cantidad depende del tamaño de la imagen. Cada uno de estos **píxeles** posee un valor en RGB, que representa la porción de ese color. Estos valores van del 0 al 255, rango que va de la ausencia total de color a la presencia máxima del color.

2.5 Segmentación de imágenes[8]

La segmentación de una imagen consiste en extraer propiedades o características comunes de alguna región de interés.

El registro de la imagen se efectúa por los métodos convencionales (**Webcam**, Escáner, cámara digital, etc.), y se almacena en un medio digital (discos, disquete, memoria, etc.). De allí el registro de la imagen es tomado para que mediante operadores sea transformado de su formato original a un formato que permita su manipulación.

[8] DUARTE, Domingo, VÁZQUEZ, Raimundo, SAMPALLO, Guillermo, GONZÁLEZ, Arturo. Implementación. En Segmentación de Imágenes mediante su Histograma. [en línea]. Disponible en: <http://www.unne.edu.ar/cyt/2001/7-Tecnologicas/T-084.pdf>.

A partir de aquí, es posible conocer las distintas partes que constituyen el archivo de la imagen. En particular, se accede al cuerpo principal del mismo que contiene la información sobre los **píxeles** con el objetivo de discriminar aquellos pertenecientes a la imagen que cumplen con un nivel de color particular, y contarlos.

2.6 Sistemas de reconocimiento de voz[9]

El reconocimiento de voz generalmente es utilizado como una interfaz humano/computadora para algún software y consiste en transformar la voz de entrada en texto. Este debe cumplir tres (3) tareas (Fig. 2.1):

- Preprocesamiento: Convierte la entrada de voz a una forma que el reconocedor pueda procesar.
- Reconocimiento: Identifica lo que se dijo (traducción de señal a texto).
- Comunicación: Envía lo reconocido al sistema (Software/Hardware) que lo requiere.

Estos componentes deberían ser transparentes al usuario. Este los nota de manera indirecta como: *certeza* en el reconocimiento y *velocidad*. Estas características las utiliza para evaluar una interfaz de reconocimiento de voz.

[9] KIRSCHNING, Ingrid. Procesamiento Automático de Voz. [en línea]. Disponible en: <http://mailweb.udlap.mx/~ingrid/ingrid/RV/Proc_de_Voz.html>.

Fig. 2.1. Componentes en una aplicación con un sistema de reconocimiento de voz. Elaborado por los autores.

2.7 Java Media Framework (JMF)[10]

El **API** (Application Programming Interface) de JMF (Java Media Framework), es una definición de la interfaz utilizada por Java para la utilización de datos **multimedia**. Como definición de interfaz, JMF indica cómo los proveedores de dichas clases deben encapsularlas y registrarlas en el sistema. Permite a las aplicaciones Java incluir componentes de audio, video, así como capturar, reproducir y decodificar múltiples formatos.

Para el tratamiento de **multimedia** en casos especiales existen además otras **APIs** complementarias a JMF y más simples o mejor adaptadas a otros problemas. Para el

[10] PRADES, Carlos. (2001). Tratamiento multimedia en Java con JMF. [en línea]. Disponible en: <http://cprades.eresmas.com/Tecnica/programarJMF.pdf>.

caso particular de la voz existe JSAPI (Java Speech API) que define la interfaz existente para trabajar con sintetizadores y reconocedores de voz.

JMF nació con el objetivo de poder representar datos **multimedia** en los programas, pero en la última versión (JMF 2.0) las capacidades se extienden a todo tipo de tratamiento como la adquisición, procesado y almacenado de datos **multimedia**, así como la transmisión a través de la red mediante el protocolo **RTP** (Real Transmition Protocol).

JMF ha sido desarrollada por Sun e IBM y no está incluida en las especificaciones de Java 2 o en la máquina virtual, por lo que es necesario obtener el paquete adicional que contiene el JMF para la plataforma que se esté utilizando.

JMF, básicamente, realiza tres pasos para el tratamiento de datos **multimedia**: la adquisición de datos, captura desde un dispositivo físico, lectura de un archivo o recepción desde la red; procesado, aplicación de efectos como filtrado o realces, compresión y/o descompresión, conversión entre formatos; y la salida de datos, presentación, almacenamiento en archivo o transmisión a través de la red.

Características de JMF[11]

Entre las características de Java Media Framework se encuentran las siguientes:

- Los diferentes estándares para el desarrollo de la transmisión de video en tiempo real.
- Permite poner en ejecución soluciones basadas en el **API** existente.
- Fácilmente integra nuevas características con el marco existente.
- Provee el acceso a los datos **multimedia** sin procesarlos.
- Mantiene compatibilidad con versiones anteriores (JMF 1.0). No obstante, es recomendable tener presente que hay que verificar las versiones para saber qué es lo que se va renovando o cambiando en cada una de ellas.

Ventajas[11]

1.- Es fácil de programar.

2.- Soporta la captura de datos **multimedia** y el manejo de los mismos.

3.- Es escalable, debido a que mantiene las mismas funcionalidades del JMF 1.0 e incorpora nuevas adaptaciones.

4.- No es necesario que el usuario de Solaris tenga que instalar alguna versión de JMF para hacer uso de ella.

5.- Al estar las librerías instaladas en el servidor Web, en donde también estará el .class de la clase, es suficiente para que sean usadas por el usuario a través del servidor.

[11] HERNANDEZ, Jéssica. Capítulo 4. Herramientas para el desarrollo del sistema. En Ambiente de aprendizaje interactivo en Internet, basado en la tecnología JSP para la Educación Ambiental [en línea]. Disponible en: < http://mail.udlap.mx/~tesis/lis/hernandez_s_ja/capitulo4.pdf>.
[11] Ident.

6.- Los applets cuentan con un buen desempeño/velocidad

7.- Al estar elaborada con la tecnología Java, permite una alta portabilidad, ya que esta tecnología se acopla perfectamente a varios sistemas operativos como es el caso de Windows, Unix, Solaris, etc., y su único requisito es tener la máquina virtual y las librerías necesarias.

2.8 Java Speech API (JSAPI)[12][13][14]

El JSAPI, desarrollado por Sun Microsystems en cooperación con compañías de tecnología del discurso (Apple Computer Inc., AT&T, Dragon Systems Inc., IBM Corporation, Novell Inc., Philips Speech Processing, Texas Instruments Incorporated, entre otras), define una interfaz del **software** que les permite a los diseñadores aprovechar la tecnología del discurso. El JSAPI define un estándar fácil de usar y soporta dos tecnologías del discurso: el reconocimiento del discurso y el sintetizador del discurso.

El reconocimiento del discurso proporciona la habilidad a las computadoras para escuchar el idioma hablado y determinar lo que se ha dicho. En otros términos, procesa la entrada de audio que contiene el discurso convirtiéndolo en texto. La síntesis del discurso proporciona el proceso inverso de producir el discurso, sintetiza el texto generado por una aplicación, un applet o un usuario y lo convierte en sonido.

[12] Sun microsystems. (1998). Java Speech API Programmer's Guide [en línea]. Disponible en: <http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide.pdf>.
[13] Sun microsystems. (1998). Java Speech Grammar Format Specification [en línea]. Disponible en: <http://java.sun.com/products/java-media/speech/forDevelopers/JSGF.pdf>.
[14] Sun microsystems. (1997). Java Speech Markup Language Specification [en línea]. Disponible en: <http://java.sun.com/products/java-media/speech/forDevelopers/JSML.pdf>.

La tecnología del discurso mejora la accesibilidad a las computadoras para muchas personas con limitaciones físicas. Para muchos usuarios es más fácil y más natural producir las órdenes orales que recordar la situación de funciones en los menús y cuadros de diálogo.

El JSAPI es una extensión a la plataforma de Java. Las extensiones son los paquetes de clases escritos en el lenguaje de programación de Java -y cualquier código nativo asociado- y que los diseñadores de la aplicación pueden usar para extender la funcionalidad de la plataforma de Java; esto permite complementar la interfaz del usuario existente con la entrada y salida de la voz.

Características del reconocedor de discurso:

- Es mono-lingual: apoya un solo idioma especificado.
- Procesa una sola entrada de audio a la vez.
- Puede adaptarse opcionalmente a la voz de sus usuarios.
- Sus gramáticas pueden ponerse al día dinámicamente.

La mayoría de los procesos de un reconocedor del discurso son automáticos y no son controlados por el diseñador de la aplicación. Por ejemplo, la colocación del micrófono, el ruido del fondo, la calidad de la tarjeta de sonido, el poder del CPU y los acentos del portavoz. Todos afectan la actuación del reconocimiento del discurso pero están más allá del control de una aplicación.

La manera principal en que una aplicación controla la actividad de un reconocedor de voz es por medio de las gramáticas. Una gramática es un objeto en el JSAPI que indica qué palabras se espera que diga un usuario y en qué modelos esas palabras pueden ocurrir. Las gramáticas son importantes para el reconocedor de la voz porque ellas reprimen el proceso del reconocimiento.

El JSAPI apoya dos tipos de gramáticas básicas, la gramática de la regla (the rule grammar) y la gramática del dictado (the dictation grammar):

La gramática de la regla, es una aplicación que proporciona reglas que definen lo que se espera que el usuario diga al reconocedor. Esto hace que sea rápido y bastante exacta a la hora de analizar la voz.

Una gramática del dictado define un juego de palabras, posiblemente miles de palabras, que pueden decirse sin restricción. Las gramáticas del dictado son más cercanas a la meta del discurso natural y aunque son más flexibles que las gramáticas de la regla, se requiere una entrada de audio de alta calidad y se tienden a cometer más errores.

Uno de los mayores beneficios de incorporar la voz en una aplicación es que la voz es natural: las personas encuentran que hablar es fácil, la conversación es una habilidad que la mayoría la domina a temprana edad en la vida.

Un factor crucial que determina el éxito de una aplicación con voz es si hay un beneficio claro al usar la voz o no. Esto significa que la voz se usa mejor cuando la necesidad está clara, por ejemplo, cuando los ojos y/o manos del usuario están ocupados, cuando los usuarios tienen una invalidez física, como el uso limitado de manos o cuando la voz habilita algo que no puede hacerse por otra parte. El paquete JSAPI define las clases e interfaces que definen la funcionalidad básica de un objeto con manejo de voz.

Una cosa importante es el estado de un reconocedor, porque representa la actividad del reconocimiento actual del mismo, el cual está siempre en uno de lo siguiente tres estados:

- Escuchando: El reconocedor está escuchando el sonido entrante de la voz que puede coincidir con una gramática activa, pero que todavía no se ha reconocido. Permanece en este estado hasta que se produce una pausa en la entrada de voz (silencio).
- Procesando: El reconocedor está procesando el discurso entrante reconocido en la gramática activa. Mientras el reconocedor está en este estado se producen los resultados que serán tratados como cadenas de textos, referidas a las palabras reconocidas.
- Suspendido: Es el estado temporal en que el reconocedor puede actualizar sus gramáticas. Mientras el reconocedor está suspendido la entrada de audio se coloca en memoria para ser procesada una vez que vuelva a los estados de Escuchar y Procesar.

MARCO METODOLÓGICO

3.1 Proceso de desarrollo del software[15]

Un proceso de desarrollo de **software** define quien está haciendo qué, cuándo y cómo alcanzar un determinado objetivo. Dicho objetivo dentro de la ingeniería del **software**, es construir un producto **software** o mejorar uno existente. Para llevar a cabo esto, se requieren una serie de normas que guíen el desarrollo del **software** de manera eficiente y de alta calidad. Además el proceso de desarrollo de **software** debe capturar y analizar las mejoras que la tecnología actual presenta, en función de integrar los esfuerzos hacia una visión y cultura común.

3.2 Proceso Unificado[16]

El Proceso Unificado es un proceso de desarrollo de **software**. Un proceso que busca mediante una serie de actividades transformar los requisitos de un usuario en un sistema **software** (figura 3.1).

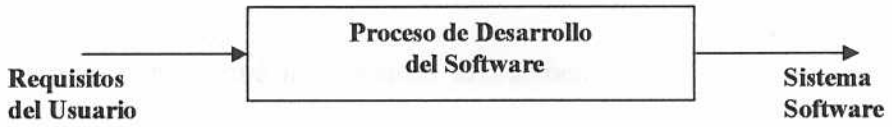

Fig. 3.1. Un proceso de desarrollo del software. Jacobson, Booch y Rumbaugh (2000, Pág. 4).

Sin embargo, el Proceso Unificado, va más allá de lo que es ser un proceso. Es un marco de trabajo genérico que puede adaptarse para diversos sistemas **software**, para

[15] JACOBSON, I., BOOCH, G. y RUMBAUGH, J. Prefacio. En su: El Proceso Unificado de Desarrollo de Software. España, Addison Wesley. 2000. pp. XV-XXV.
[16] JACOBSON, I., BOOCH, G. y RUMBAUGH, J. El Proceso Unificado de Desarrollo de Software. España, Addison Wesley. 2000. pp. 4-123.

diferentes áreas de aplicación, diferentes tipos de organizaciones, y diferentes tamaños de proyectos.

El Proceso Unificado, se basa en componentes **software** interconectados por interfaces bien definidas, además utiliza UML (Lenguaje Unificado de Modelado, Apéndice B), para preparar todos los esquemas que se necesitan para describir un sistema. El Proceso Unificado puede definirse en tres frases claves: dirigido por casos de usos, centrado en la arquitectura, e iterativo-incremental.

3.2.1 Dirigido por casos de uso[16]

La captura de requisitos tiene dos objetivos fundamentales, el primero es encontrar los verdaderos requisitos y el segundo es representarlos de un modo adecuado para los usuarios, clientes y desarrolladores. Entendiéndose por "verdaderos requisitos" aquellos que cuando se implementan añadirán el valor esperado para los usuarios. Con "representarlos de un modo adecuado para los usuarios, clientes y desarrolladores" se quiere que la descripción de los requisitos sea compresible para usuarios y clientes.

Es muy normal que los sistemas interactúen con diferentes tipos de usuarios. Cada tipo de usuario representa a un actor. Los actores utilizan el sistema con los casos de uso. Un caso de uso es una secuencia de acciones que el sistema lleva a cabo para ofrecer algún resultado de valor para un actor. El modelo de casos de uso está compuesto por todos los actores y todos los casos de uso del sistema.

[16] Ident.

Los casos de uso dan inicio a un proceso de desarrollo que progresa a medida que se concretan una serie de flujos de trabajo. Dirigen todo el proceso de desarrollo en su totalidad, es decir, cuando se identifican clases, subsistemas e interfaces, cuando se identifican casos de prueba, y cuando se planifican las iteraciones del desarrollo y la integración del sistema. Por cada iteración, se completan todos los flujos de trabajo, desde la captura de requisitos, pasando por el análisis, diseño e implementación, hasta la prueba. Por cada iteración, se establece una traza entre los diferentes flujos de trabajo (figura 3.2), lo cual permite mantener la integridad y la actualización del sistema cuando se trabajan con requisitos cambiantes.

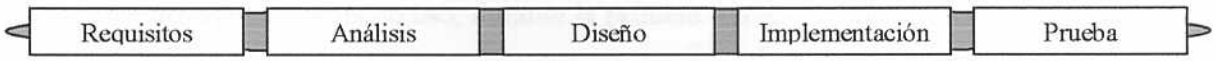

Fig. 3.2. Los casos de uso enlazan los flujos de trabajo fundamentales. La elipse sombreada en el fondo, simboliza como los casos de uso enlazan esos flujos de trabajo. Jacobson, Booch y Rumbaugh (2000, Pág. 37).

3.2.2 Centrado en la arquitectura[16]

Además de la guía de los casos de uso, es necesaria una visión común en la que los empleados (desarrolladores y usuarios) deben estar de acuerdo. Esta visión común es lo que se llama arquitectura, y proporciona una perspectiva del sistema completa, necesaria para controlar el desarrollo.

[13] Ident.

Las diferentes perspectivas, que los desarrolladores pueden observar de un sistema **software** son **vistas** del modelo de sistema, todas ellas juntas forman la arquitectura.

La arquitectura es dirigida por los casos de uso, y está condicionada por ellos (figura 3.3). En las primeras iteraciones del ciclo de vida, se eligen algunos casos de uso, aquellos que sean arquitectónicamente más significativos para la versión actual y para las futuras versiones del sistema. La dependencia entre los casos de uso y la arquitectura muchas veces genera confusión respecto a qué cosa debemos realizar primero. La recomendación de Jacobson, Booch, y Rumbaugh es construir una arquitectura tentativa a partir de algunos casos de uso, durante la primera iteración. Luego esta arquitectura se va ampliando y haciendo más robusta a medida que se agregan casos de uso en las iteraciones posteriores.

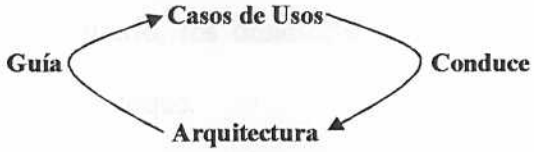

Fig. 3.3. Los casos de uso conducen el desarrollo de la arquitectura, y la arquitectura indica qué casos de uso pueden realizarse. Jacobson, Booch y Rumbaugh (2000, Pág. 64).

3.2.3 Iterativo e incremental[16]

Dentro de la metodología del Proceso Unificado, el trabajo para desarrollar un producto es dividido en partes o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo,

[16] Ident.

y los incrementos al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas, es decir, deben seleccionarse y ejecutarse de una forma planificada. Es por esto que son consideradas miniproyectos.

Los desarrolladores basan la selección de lo que se implementará en una iteración en dos factores: en primer lugar, la iteración trata un grupo de casos de uso que juntos amplían la utilidad del producto desarrollado hasta ahora; en segundo lugar, la iteración trata los riesgos más importantes. Las iteraciones sucesivas se construyen sobre los **artefactos** de desarrollo tal y como quedaron al final de la ultima iteración.

Al ser miniproyectos, comienzan con los casos de uso y continúan a través del trabajo de desarrollo subsiguiente (análisis, diseño, implementación y prueba), que termina convirtiendo en código ejecutable los casos de uso que se desarrollan en las iteraciónes. Si una iteración cumple con sus objetivos, el desarrollo continúa con la siguiente iteración, de lo contrario, los desarrolladores deben revisar sus decisiones previas y probar con un nuevo enfoque.

3.2.4 Ciclo de vida del Proceso Unificado[16]

El proceso Unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición, y cada fase se subdivide en iteraciones. Cada ciclo concluye con una versión del producto para los clientes (figura 3.4).

[16] Ident.

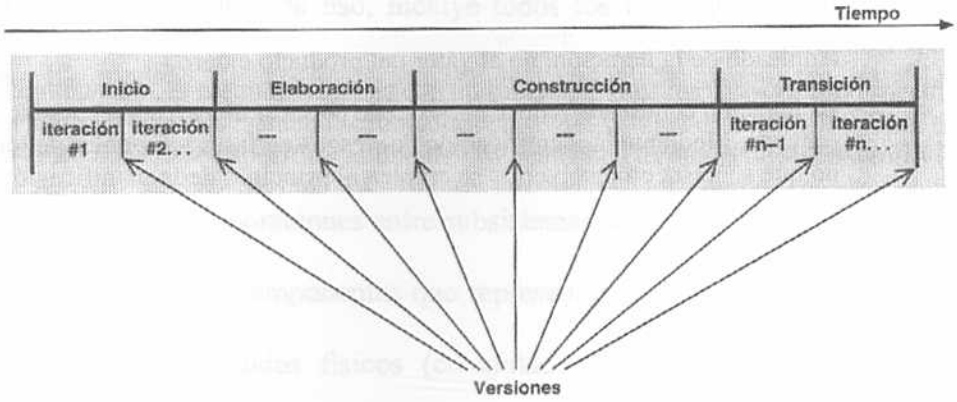

Fig. 3.4. Un ciclo con sus fases e interacciones. Jacobson, Booch y Rumbaugh (2000, Pág. 8).

Cada versión es un producto preparado completo, y consta de un cuerpo de código fuente incluido en componentes que pueden compilarse y ejecutarse, además de los casos de uso, especificaciones no funcionales y casos de prueba. También incluye el modelo de la arquitectura y el modelo visual (elementos modelados con el Lenguaje Unificado de Modelado, figura 3.5).

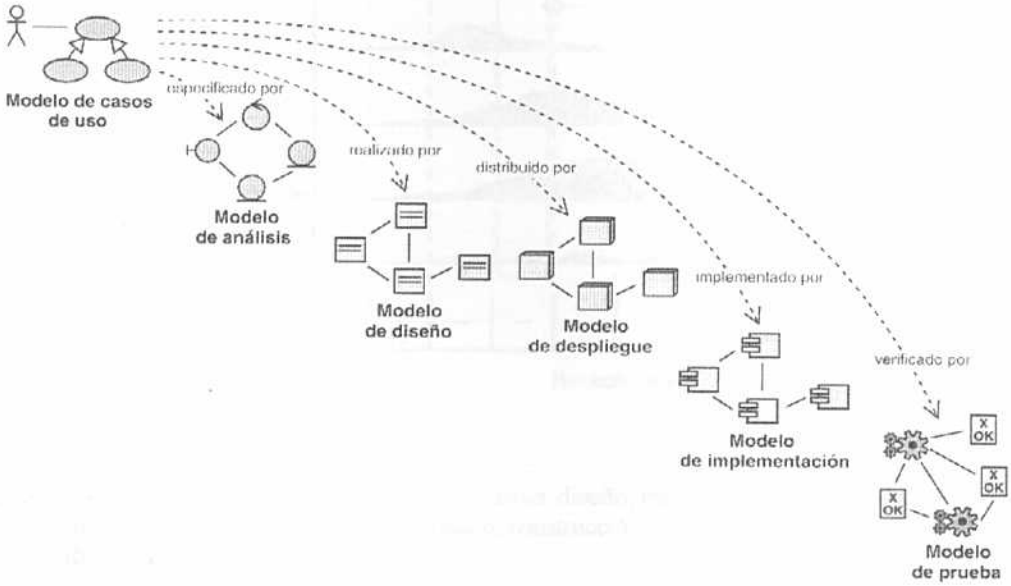

Fig. 3.5. Modelo del proceso unificado. Existen dependencias entre muchos de los modelos. Como ejemplo, se indican las dependencias entre el modelo de casos de uso y los demás modelos. Jacobson, Booch y Rumbaugh (2000, Pág. 9).

El modelo de casos de uso, incluye todos los casos y sus relaciones con los usuarios. El modelo de análisis, mejora los casos de usos para establecer la funcionalidad inicial del sistema. El modelo de diseño, define una estructura estática del sistema en forma de colaboraciones entre subsistemas, clases e interfaces. El modelo de implementación, incluye componentes que representan al código fuente. El modelo de despliegue, define los nodos físicos (computadores) y la correspondencia de los componentes con esos nodos. El modelo de prueba, describe los casos de prueba que verifican los casos de usos.

3.2.4.1 Fases dentro de un ciclo[16]

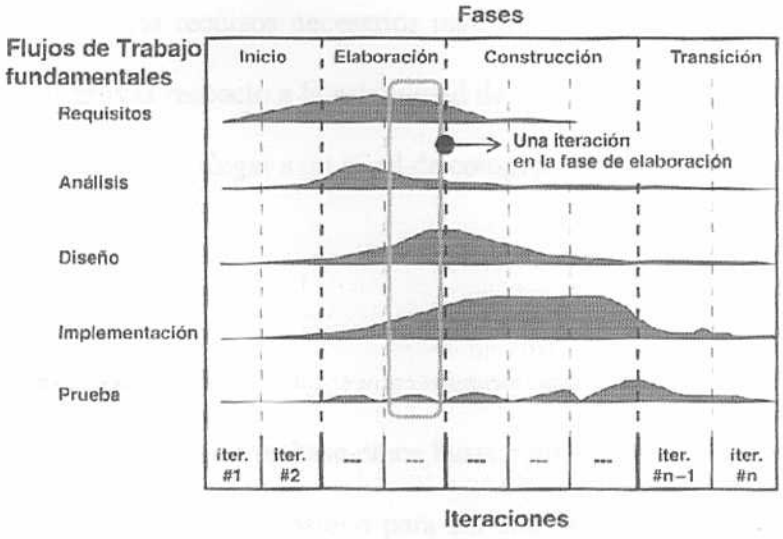

Fig. 3.6. Los cinco flujos de trabajo –requisito, análisis, diseño, implementación y prueba- Tiene lugar sobre las cuatro fases: inicio, elaboración, construcción y transición. Jacobson, Booch y Rumbaugh (2000, Pág. 11).

[16] Ident.

3.2.4.1.1 Inicio[16]

Durante la fase de inicio, se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto, en esta fase se estudia cuáles son las principales funciones del sistema para sus usuarios más importantes, se trata de encontrar y priorizar los riesgos más importantes, y se estima el costo del proyecto.

3.2.4.1.2 Elaboración[16]

Durante la fase de elaboración, se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. El resultado de esta fase es una línea base de la arquitectura, que permite al director del proyecto planificar las actividades y estimar los recursos necesarios para terminar con éxito. Si la línea base satisface las expectativas respecto a la estabilidad de los casos de uso, la arquitectura, el plan, y los riesgos, se puede llegar a un nivel de compromiso para el desarrollo completo del proyecto.

3.2.4.1.3 Construcción[16]

Dentro de esta fase la línea base crece hasta convertirse en el sistema completo, es decir, un producto que está preparado para ser entregado a los usuarios. La mayor parte de los recursos se emplean durante este período del desarrollo. Al terminar la construcción, el producto contiene todos los casos de uso que la dirección y el cliente han acordado para el desarrollo de esta versión.

[16] Ident.

3.2.4.1.4 Transición[16]

En esta fase un pequeño grupo de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Los desarrolladores corrigen los problemas e incorporan algunas de las mejoras sugeridas en una versión nueva. Normalmente los defectos se dividen en dos categorías: los que tienen suficiente impacto en la operación para justificar una versión incrementada, y los que pueden corregirse en la siguiente versión normal.

3.2.4.2 Flujos de Trabajo

3.2.4.2.1 Captura de Requisitos[16]

La captura de requisitos es el proceso de averiguar normalmente en circunstancias difíciles, lo que se debe construir. La técnica inmediata para la captura de requisitos del sistema se basa en los casos de uso. Por lo tanto, los requisitos funcionales como no funcionales son tomados en cuenta con esta técnica. Debido a que los requisitos cambian constantemente, se necesita alguna forma de actualizarlos de manera controlada. Esto se hace en las iteraciones, donde cada iteración reflejará algún cambio en el conjunto de requisitos, pero el número de cambios normalmente disminuirá a medida que se ingrese en la fase de construcción y a medida que se estabilicen los requisitos (figura 3.7).

[16] Ident.

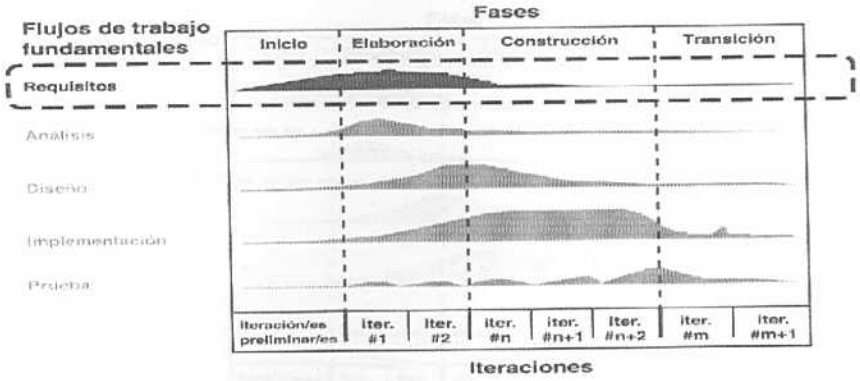

Fig. 3.7. El trabajo de los requisitos se hace fundamentalmente durante el inicio y la elaboración. Jacobson, Booch y Rumbaugh (2000, Pág. 112).

3.2.4.2.2 Análisis[16]

Se analizan las necesidades que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo es conseguir una compresión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero, incluyendo la arquitectura. Las iteraciones iniciales de la elaboración se centran en el análisis. Eso contribuye a obtener una arquitectura estable y sólida y facilita una comprensión en profundidad de los requisitos. Más adelante, al término de la fase de elaboración y durante la construcción, cuando la arquitectura es estable y se comprenden los requisitos, el análisis pasa en cambio al diseño y a la implementación (figura 3.8).

[16] Ident.

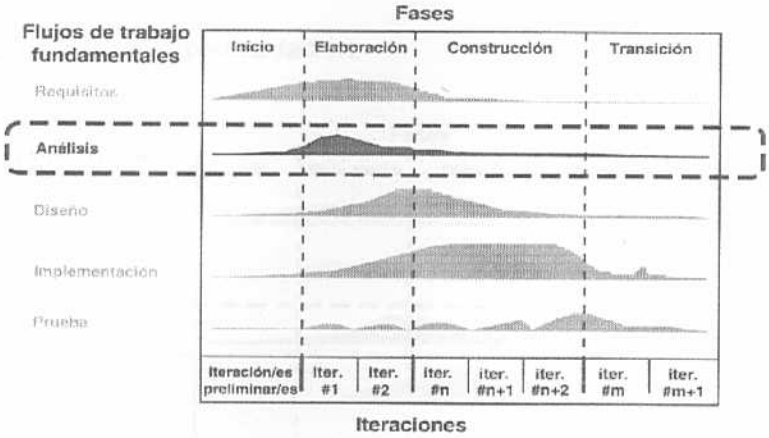

Fig. 3.8. El trabajo del análisis se hace fundamentalmente durante la elaboración. Jacobson, Booch y Rumbaugh (2000, Pág. 171).

3.2.4.2.3 Diseño[16]

En el diseño se modela el sistema y se encuentra su forma para que soporte todos los requisitos. Entre los propósitos fundamentales del diseño están: adquirir una compresión en profundidad de los aspectos relacionados con los requisitos funcionales, crear una entrada apropiada y un punto de partida para la implementación, descomponer los trabajos de implementación en partes manejables, visualizar y reflexionar sobre el diseño utilizando una notación común, crear una abstracción de la implementación que permita la generación de código y la ingeniería de ida y vuelta con el diseño. El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción (figura 3.9). Esto contribuye a una arquitectura estable y sólida y a crear un plano del modelo de implementación. Más tarde, durante la fase de construcción,

[16] Ident.

cuando la arquitectura es estable y los requisitos están bien entendidos, el centro de atención se desplaza a la implementación.

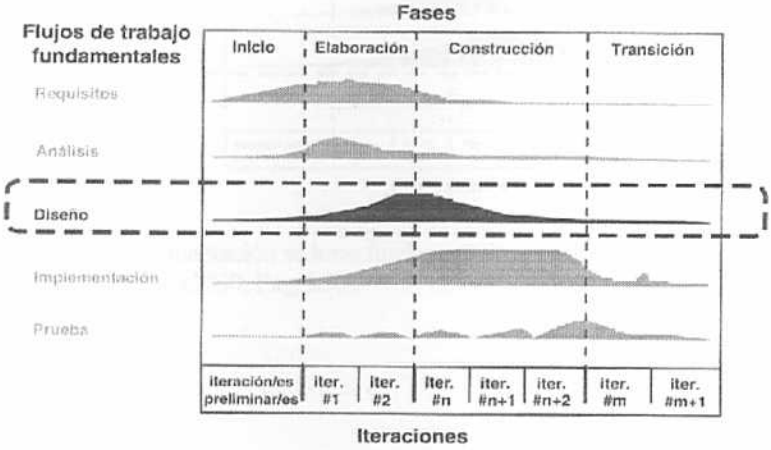

Fig. 3.9. El trabajo de diseño se hace fundamentalmente durante la elaboración y la construcción. Jacobson, Booch y Rumbaugh (2000, Pág. 207).

3.2.4.2.4 Implementación[16]

En la implementación se comienza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, archivos de código fuente, scripts (lista de comandos que pueden ser ejecutados sin la interacción del usuario), archivos de código binario, ejecutables y similares. La implementación es el centro durante las iteraciones de construcción (figura 3.10), aunque también se lleva a cabo trabajo de implementación durante la fase de elaboración, para crear la línea base ejecutable de la arquitectura, y durante la fase de transición, para tratar defectos tardíos (ver pico en la columna de transición. Figura 3.10).

[16] Ident.

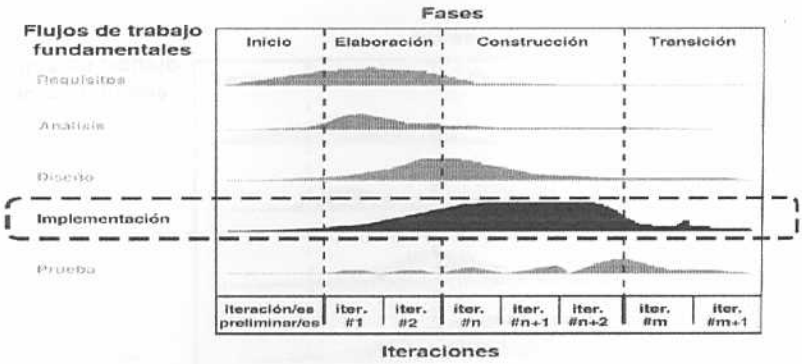

Fig. 3.10. El trabajo de implementación se hace fundamentalmente durante la construcción. Jacobson, Booch y Rumbaugh (2000, Pág. 256).

3.2.4.2.5 Pruebas[16][17]

En el flujo de trabajo de las pruebas se verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros.

Durante la parte de inicio puede hacerse parte de la planificación inicial de las pruebas cuando se define el ámbito del sistema. Sin embargo, las pruebas se llevan a cabo sobre todo cuando una construcción (resultado de una implementación) es sometida a pruebas de integración y de sistema. Esto quiere decir que la realización de pruebas se centra en las fases de elaboración, cuando el grueso del sistema está implementado. Durante la fase de transición el centro se desplaza hacia la corrección de defectos durante los primeros usos y a las pruebas de regresión (figura 3.11).

[16] Ident.
[17] JACOBSON, I., BOOCH, G. y RUMBAUGH, J. El proceso dirigido por casos de uso. En su: El Proceso Unificado de Desarrollo de Software. España, Addison Wesley. 2000. pp. 31-54.

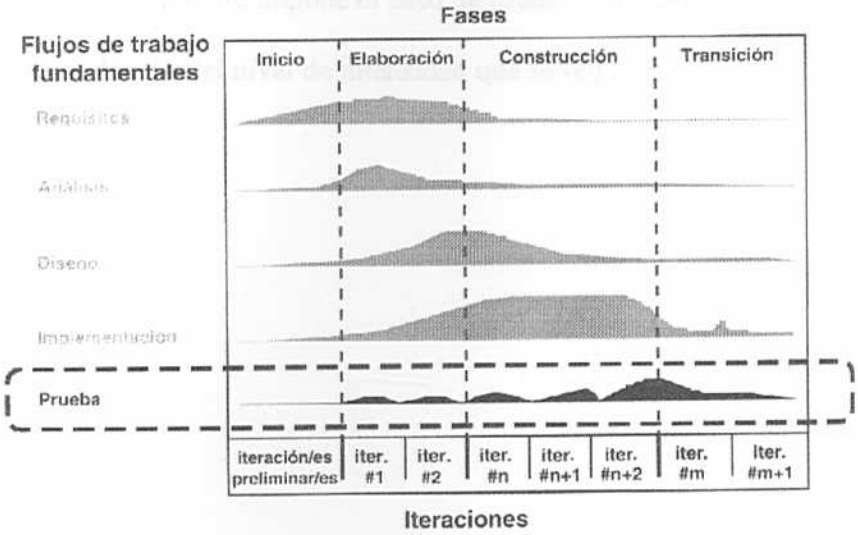

Fig. 3.11. Las pruebas se hacen durante todas las fases, pero mayormente durante la construcción y la transición. Jacobson, Booch y Rumbaugh (2000, Pág. 282).

Existen dos tipos de pruebas: las de caja negra y las de caja blanca. Las pruebas de caja negra, se basan en la verificación de los resultados de la interacción de los actores con el sistema terminado, no son importantes los pasos internos que lleva acabo el sistema, sino el resultado final.

Las pruebas de caja blanca, se basan en la verificación de un escenario particular, es decir, se chequean los pasos internos que realiza el sistema con la finalidad de observar si el comportamiento de los mismos cumple con lo establecido por la prueba.

La metodología del Proceso Unificado, por cada caso de uso, diseña un caso de prueba. Estos describen, como debe ejecutarse la prueba, especificando las entradas necesarias y las salidas esperadas. Además, detallan las acciones que se realizan cuando

se desvían las condiciones que impone el caso de prueba y pueden ser de caja negra o de caja blanca, dependiendo del nivel de intensidad que se requiera.

DESARROLLO

En este capítulo, se explicará el desarrollo del mouse para minusválidos aplicando la metodología del Proceso Unificado. El desarrollo será descrito en función de los flujos de trabajo y del resultado arrojado por todas las iteraciones que se llevaron a cabo.

4.1 Captura de Requisitos

Durante este flujo de trabajo, se realizó un levantamiento de información para aclarar las cosas que debían realizarse en función de los objetivos planteados. Después de haber consultado diferentes fuentes de información, se pudo obtener un primer acercamiento a las especificaciones de las cuales debería estar dotado el producto final.

En las primeras iteraciones se consultaron a cinco (5) doctores (Nadia Di Felice, Vincenzo Guarino, Claudio Del Turco, Zelinda Mariño y José Cardenas) los cuales argumentaron que "la dificultad motora de los pacientes varía según la discapacidad que cada uno posea". Luego de recopilar la información, se concluyó, según los datos de los médicos, que el 80% de los pacientes conservan la función motora del cuello, el sentido de visión y el lenguaje.

En las siguientes iteraciones, se averiguó sobre las funciones de un mouse común de dos (2) botones (Tabla 4.1):

Número	Requisito
1	Mover puntero
2	Presionar botón izquierdo
3	Presionar botón derecho
4	Presionar 2 veces botón izquierdo
5	Seleccionar objetos
6	Arrastrar objetos con botón izquierdo
7	Arrastrar objetos con botón derecho

Tabla 4.1. Requisitos obtenidos del mouse. Creada por los autores.

Los requisitos mostrados en la tabla anterior se realizan combinando o no el movimiento del mouse con el hecho de dejar o no presionado cualquiera de los dos (2) botones.

En las iteraciones siguientes se buscó información de cómo los dispositivos **hardware (webcam** y micrófono) se comunican con el computador y como envían los datos a este. En esta búsqueda, se investigaron varios lenguajes de programación, entre ellos estaban: Visual Basic, Visual C++, C++ y Java. Se encontró que con todos ellos podía desarrollarse el sistema final, pero el estudio demostró que la comunicación estaba contemplada muy bien por las **APIs** de Java, haciéndola fácil para el programador y permitiéndole tener el control sobre los mismos, lo que originó que Java, fuera seleccionado como lenguaje de desarrollo para esta tesis. Además influyó, la portabilidad que ofrece Java para que el sistema pueda instalarse en diversos sistemas operativos.

En las siguientes iteraciones se buscaron reconocedores de voz para el programa. De estos se encontraron los de idioma inglés, español y chino, de los cuales se

seleccionaría el mejor en futuras iteraciones. Existen muchas casas de **software** que proporcionan reconocedores de voz en diferentes lenguajes, entre ellos podemos citar: Microsoft, Scansoft, The Cloud Garden, Dragon Systems Inc., IBM Corporation, Novell Inc., Philips Speech Processing, entre otros.

Este producto puede ser utilizado por cualquier usuario, pero está dirigido especialmente a las personas con limitaciones físicas (usuario minusválido), lo que hace que se conviertan en los actores principales dentro del sistema.

4.2 Análisis

Se realizó un estudio más detallado de los requerimientos obtenidos durante la captura de requisitos con la finalidad de obtener una visión más real de los mismos.

En las primeras iteraciones del análisis, se determinaron las barreras, inconvenientes y obstáculos para buscar cómo eliminarlos o minimizarlos. Este análisis arrojó los siguientes resultados (Tabla 4.2):

Obstáculo	Inconveniente	Acción Tomada
Realizar un prototipo de casco que permita mover el puntero del mouse.	Incomodo para usar, y obliga a la persona a estar unido al computador por medio de cables.	Realizar un prototipo que ofreciera al usuario una forma más libre de interactuar con el computador, con la ayuda de una **webcam.**
Realizar un dispositivo que permita ejecutar los comandos del mouse, mediante la presión de aire.	Engorroso para usar, debido a su funcionamiento.	Ejecutar los comandos del mouse, mediante la voz con la ayuda de un micrófono.

Seleccionar el color que se utilizaría en el análisis de las imágenes.	Problemas en el análisis, debido a que las componentes RGB de los **píxeles** son afectadas por los colores del espacio físico donde se encuentra el usuario, generando imprecisiones en los cálculos.	Buscar un color tal que minimice el rango de error en los cálculos arrojados por el análisis de las imágenes.

Tabla 4.2. Obstáculos encontrados. Creada por los autores.

En las siguientes iteraciones se procedió a desarrollar los casos de uso a partir de los requerimientos obtenidos en las iteraciones iniciales, y para cada uno se describió en detalle: la pre-condición, el flujo principal, los sub-flujos y los flujos alternos. Luego, se elaboró un modelo de casos de uso, que englobaba la funcionalidad total del sistema (Fig. 4.3).

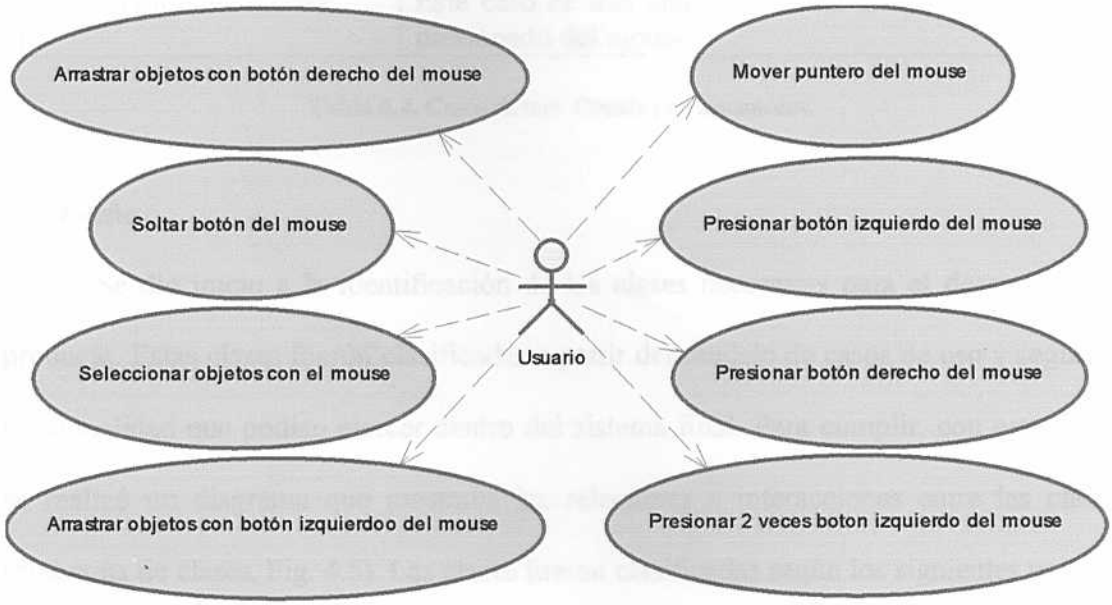

Fig. 4.3 Modelo general de casos de usos. Creado por los autores.

Cada caso de uso de la figura anterior, representa una función que puede ejecutar el usuario dentro del sistema. A continuación se describe brevemente que hace cada uno de ellos (Tabla 4.4), para una explicación más detallada consultar el Apéndice B.

Caso de uso	Descripción
Mover puntero del mouse.	Este caso de uso mueve el **puntero** del mouse a la posición deseada.
Presionar botón izquierdo del mouse.	Este caso de uso simula el pulsar y soltar el botón izquierdo del mouse una vez (clic izquierdo).
Presionar botón derecho del mouse.	Este caso de uso simula el pulsar y soltar el botón derecho del mouse una vez (clic derecho).
Presionar 2 veces botón izquierdo del mouse.	Este caso de uso simula el pulsar y soltar el botón izquierdo del mouse dos veces seguidas (doble clic).
Seleccionar objetos con el mouse	Este caso de uso simula el **seleccionar** uno o varios **objetos de windows** con el mouse.
Arrastrar objetos con botón izquierdo del mouse.	Este caso de uso simula el **arrastrar objetos de windows** con el botón izquierdo del mouse.
Arrastrar objetos con botón derecho del mouse	Este caso de uso simula el **arrastrar objetos de windows** con el botón derecho del mouse.
Soltar botón del mouse.	Este caso de uso simula el soltar el botón que este presionado del mouse.

Tabla 4.4. Casos de uso. Creada por los autores.

4.3 Diseño

Se dio inicio a la identificación de las clases necesarias para el desarrollo del producto. Estas clases fueron clasificadas a partir del modelo de casos de uso y según la funcionalidad que podían ejercer dentro del sistema final. Para cumplir, con esta tarea, se realizó un diagrama que mostraba las relaciones e interacciones entre las clases (diagrama de clases, Fig. 4.5). Las clases fueron clasificadas según los siguientes usos:

- Las funciones básicas del mouse.
- La captura de las imágenes desde la **webcam**.
- La captura de la voz, desde el micrófono.
- El análisis de la imagen, por medio de algún algoritmo para ello.

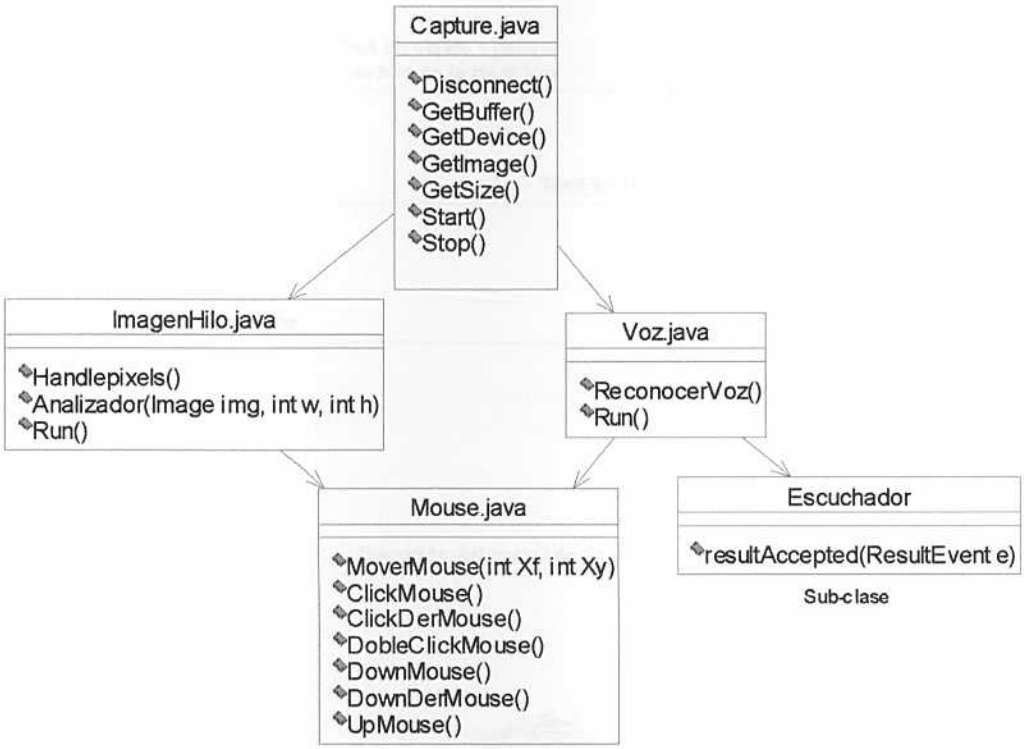

Fig. 4.5. Diagrama de clases del sistema. Cada elemento dentro de una clase es un **método** de la misma.Elaborado por los autores.

Además, se realizaron los diagramas de secuencia, para establecer el flujo de información entre los actores y las clases (Fig. 4.6 y Fig. 4.7). Dicho flujo, permitió definir los parámetros que necesitaban enviarse cada una de las clases, obteniéndose una mejor visión de éstas a nivel de funcionamiento.

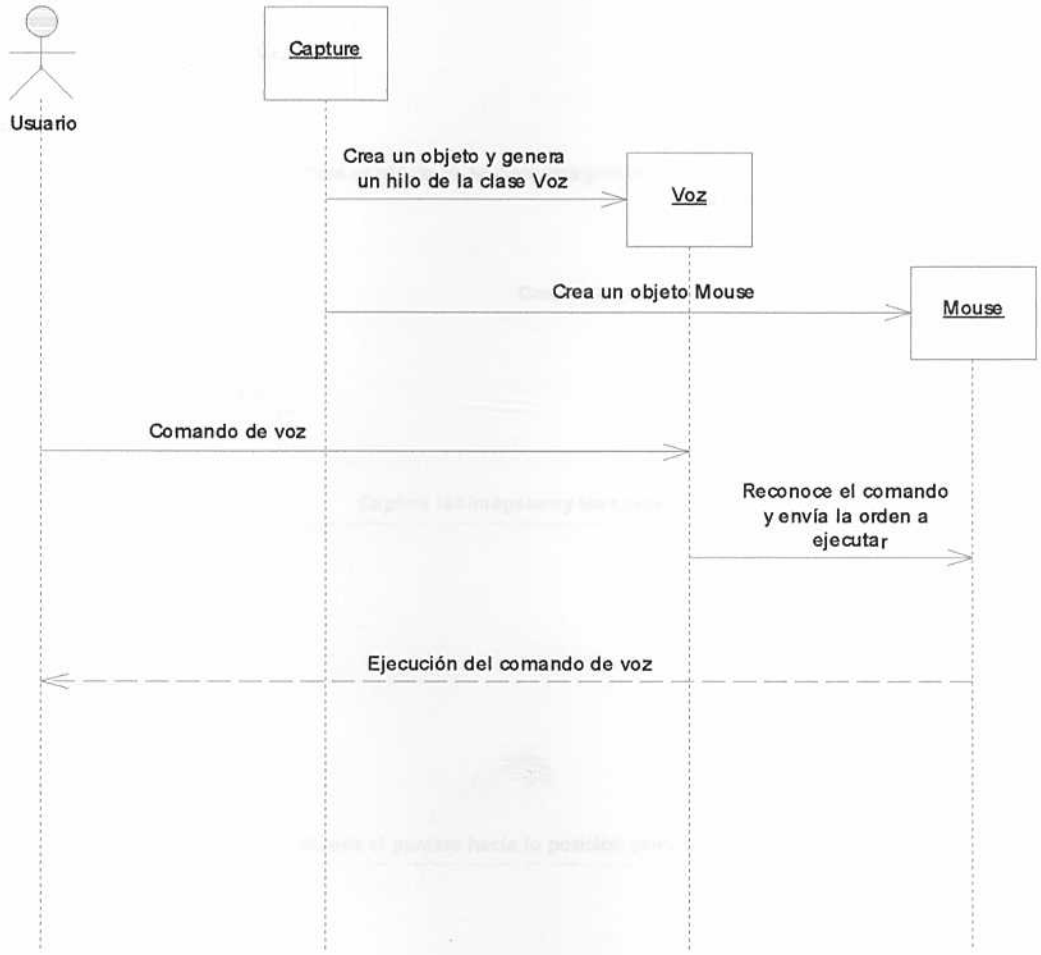

Fig. 4.6. Diagrama de secuencia de comandos de voz. Elaborado por los autores.

La figura anterior, muestra como es la secuencia que se lleva a cabo para ejecutar un comando de voz. Primero la clase *Capture*, crea un objeto de las clases *Voz* y *Mouse*. A partir de aquí, el usuario, puede ejecutar un comando de voz, el cual será recibido y analizado por la clase *Voz*. Inmediatamente, el resultado es informado a la clase *Mouse*, para que realice la acción y la haga visible al usuario.

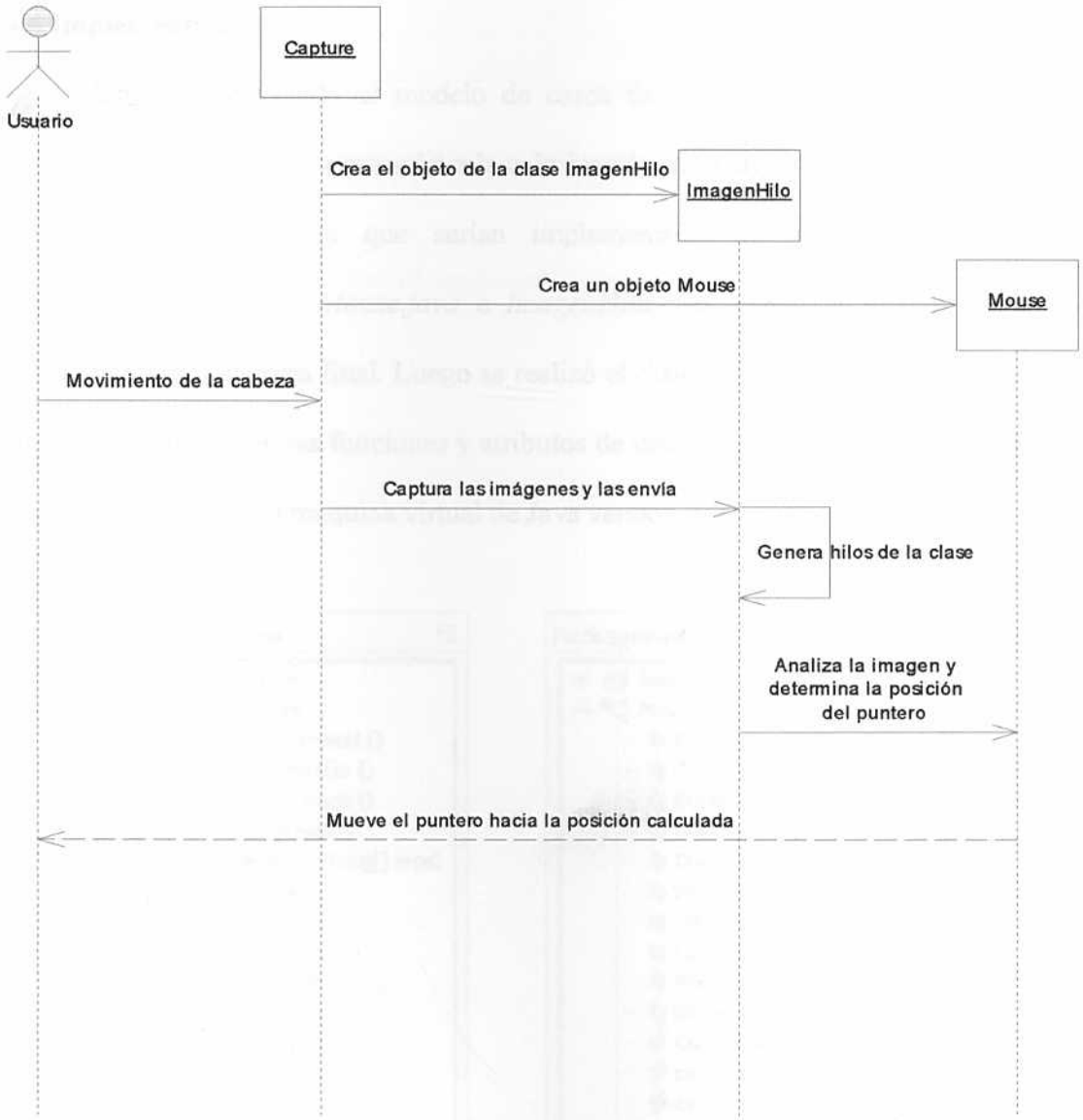

Fig. 4.7. Diagrama de secuencia del movimiento del puntero. Elaborado por los autores.

La figura anterior, muestra como es la secuencia que se lleva a cabo para mover el puntero del mouse. Primero la clase *Capture*, crea un objeto de las clases *ImagenHilo* y *Mouse*. A partir de aquí, el usuario puede realizar movimientos de la cabeza, los cuales son capturados y analizados por la clase *ImageHilo*, que genera varios hilos para realizar esta tarea. El resultado es informado a la clase *Mouse*, para que realice el movimiento y lo haga visible al usuario.

4.4 Implementación

Una vez terminado el modelo de casos de uso, el diagrama de clases y el diagrama de secuencias, se procedió a la culminación de la arquitectura. A partir de aquí, se definieron las clases que serían implementadas físicamente. Ellas fueron: *Capture.java*, *Voz.java*, *Mouse.java* e *ImagenHilo.java*, según la funcionalidad que debían aportar al sistema final. Luego se realizó el diagrama de componentes (Fig. 4.8), en el cual se muestran las funciones y atributos de cada una de las clases. Todo esto fue realizado a través de la máquina virtual de Java versión 1.3.

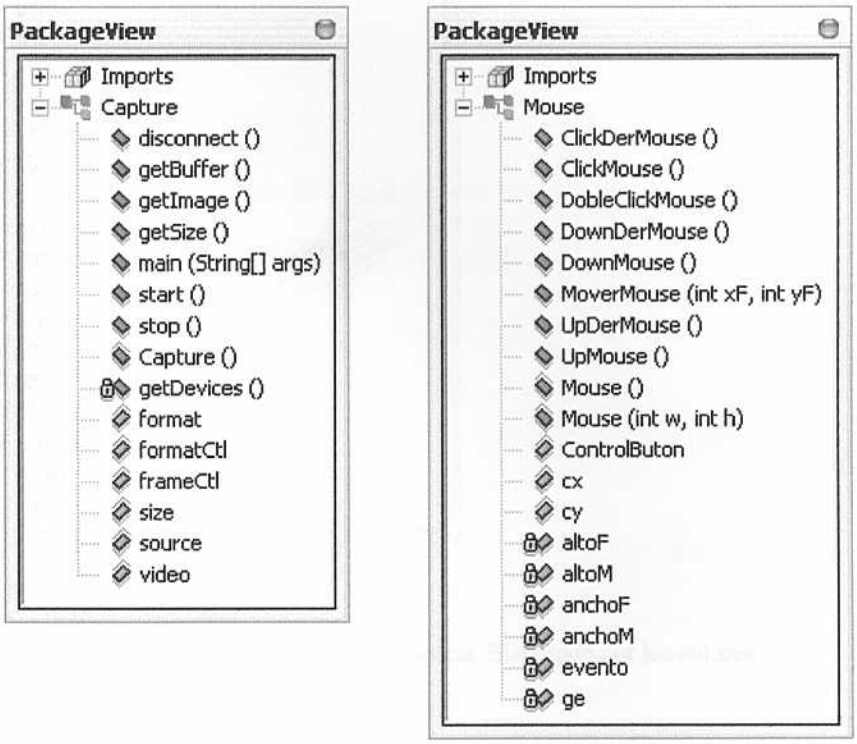

Fig. 4.8. Diagrama de componentes. Elaborado por los autores.

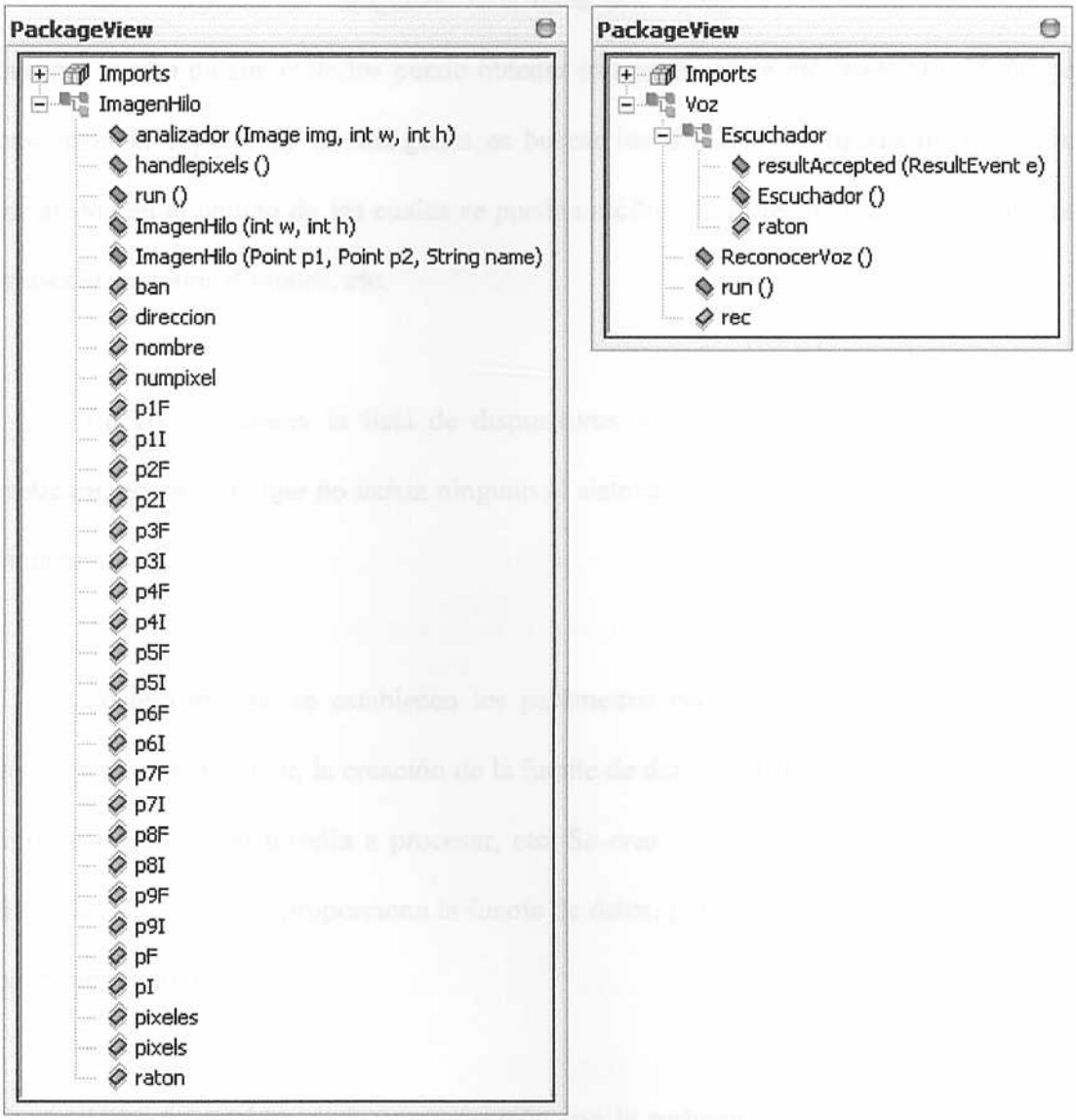

Fig. 4.8. Diagrama de componentes. Elaborado por los autores.

A través de las iteraciones sucesivas, se implementaron versiones de cada una de las clases. Estas versiones fueron mejoradas de iteración en iteración hasta que se llegó a un punto en que la funcionalidad estaba completada. A continuación se describen cada una de las clases implementadas.

4.4.1 Captura de imágenes: Esta tarea está delegada a la clase *Capture.java*, que por medio de sus métodos puede obtener imágenes desde una **webcam**. El primer paso para la captura de las imágenes es buscar los dispositivos **hardware** que están instalados en el equipo de los cuales se pueden recibir imágenes, entre ellos: escáneres, **webcam**, cámaras digitales, etc.

Luego de obtener la lista de dispositivos, se selecciona aquel referido a una **webcam**, en caso de que no exista ninguno el sistema emite un mensaje para indicar la situación.

Posteriormente, se establecen los parámetros necesarios para la comunicación con la **webcam**, es decir, la creación de la fuente de datos, el tamaño de las imágenes, el tipo de formato **multimedia** a procesar, etc. Se crea además un **player**, que recibe el flujo de información y proporciona la fuente de datos, para poder procesarla y analizarla en el momento indicado.

Una vez establecida la comunicación con la **webcam**, por medio del **player**, se procede a convertir el flujo de información en imágenes consecutivas, que son enviadas hacia un analizador para que las examine (*ImagenHilo.java*). Cada vez que el sistema por medio de la **webcam** obtiene una nueva imagen, ésta se envía de inmediato al analizador. Durante todo el tiempo que el sistema está activo, se repite el ciclo de capturar y analizar las imágenes.

4.4.2 Procesamiento de imágenes: Esta labor es realizada por la clase *ImagenHilo.java*. El procedimiento se inicia cuando la clase *Capture.java* envía una imagen para ser analizada. Dicha imagen, es segmentada en nueve (9) recuadros, que indican una dirección de movimiento (Fig. 4.9). Luego por cada dirección, se genera un **hilo** para que analice su porción de la imagen.

El análisis que realiza cada **hilo**, consiste en buscar dentro de la porción de la imagen que le corresponde, los **píxeles** de color azul que se encuentran allí y llevar un conteo de los mismos. Cuando todos los **hilos** terminen su conteo, se verifica cuál de ellos encontró la mayor cantidad de **píxeles**. Finalmente, el **puntero** del mouse se mueve en velocidad constante hacia esa dirección, con la ayuda de la clase *Mouse.java*.

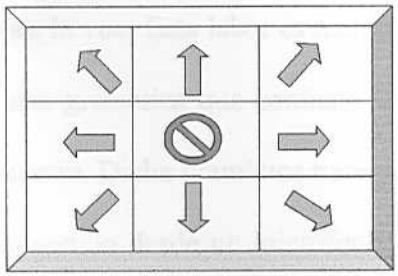

Fig. 4.9. Direcciones de movimientos. Elaborado por los autores.

En la figura anterior, se muestra las direcciones en las que se puede mover el **puntero** del mouse. A continuación se explican cada una ellas (Tabla 4.10):

Dirección	Descripción
1	Se desplaza el **puntero** del mouse, en forma diagonal hacia la parte superior izquierda de la pantalla.
2	Se desplaza el **puntero** del mouse, en forma vertical hacia la parte superior de la pantalla.
3	Se desplaza el **puntero** del mouse, en forma diagonal hacia la parte superior derecha de la pantalla.
4	Se desplaza el **puntero** del mouse, en forma horizontal hacia la izquierda de la pantalla.
5	No se desplaza el **puntero** del mouse. Se utiliza para dejar fijo el puntero del mouse.
6	Se desplaza el **puntero** del mouse, en forma horizontal hacia la derecha de la pantalla.
7	Se desplaza el **puntero** del mouse, en forma diagonal hacia la parte inferior izquierda de la pantalla.
8	Se desplaza el **puntero** del mouse, en forma vertical hacia la parte inferior de la pantalla.
9	Se desplaza el **puntero** del mouse, en forma diagonal hacia la parte inferior derecha de la pantalla.

Tabla. 4.10. Desplazamiento del **puntero** del mouse. Elaborado por los autores.

4.4.3 Procesamiento de la voz: Esta labor es realizada por la clase *Voz.java*, y se define dentro de esta clase, una gramática que contiene la sintaxis de las palabras que serán permitidas captar del usuario. Dicha gramática necesita que un reconocedor de voz esté activo para que reciba el sonido desde un micrófono. Este reconocedor de voz no forma parte de este sistema, por lo cual el equipo debe contar con uno para que el reconocimiento de palabras pueda funcionar. Se encontraron varios reconocedores en idiomas diferentes. Luego de realizar las pruebas pertinentes, se determinó que el que mejor funcionaba era el de idioma inglés, el cual es el utilizado por este sistema.

El sistema cuando es iniciado genera una lista con todos los reconocedores de voz que tenga el equipo y luego selecciona uno que esté en el idioma inglés. En caso de que la lista esté vacía o no exista un reconocedor en inglés, el sistema envía un mensaje

e indica la situación. Una vez completada la selección del reconocedor, se coloca en estado *escuchando* y se le asignan los recursos requeridos para que funcione; inmediatamente la gramática es activada y colocada en estado activo para ser usada por el reconocedor.

El sistema genera un **hilo** del programa principal que se encargará de procesar todos los comandos de voz que el usuario proporcione. Este **hilo** permanece activo durante el tiempo que el sistema se mantiene en ejecución. Por lo tanto, el usuario puede introducir el comando de voz que desee ejecutar en cualquier momento, una vez iniciado y activado el sistema. Estos comandos de voz son procesados por el reconocedor, el cual detecta la expresión y genera un **evento** por medio de la clase *Mouse.java* para que el sistema operativo lo procese (estado *procesando*). Si el reconocedor encuentra que la palabra no satisface la gramática activa, se coloca nuevamente en estado *escuchando*, en espera de un nuevo comando de voz.

4.4.4 Manejo del mouse: Esta labor es desempeñada por la clase *Mouse.java*, que permite ejecutar todas las operaciones desempeñadas por un mouse estándar de dos (2) botones. Para la realización de éstas, es necesario el apoyo de las clases "java.awt.GraphicsEnvironment", "java.awt.event.InputEvent", y "java.awt.Robot".

La primera se utiliza para capturar la configuración de la pantalla principal o de fondo, dicha configuración obtiene mucha información sobre la misma, como por ejemplo los **objetos** que se encuentran dentro de ella o la cantidad de color en **píxeles**

que soporta; para este sistema es vital debido a que informa sobre la **resolución** o las dimensiones dentro de las cuales será ubicado el **puntero**.

En ocasiones será necesario saber si la pantalla gráfica principal contiene **objetos**, como por ejemplo, **ventanas** gráficas, íconos, botones, entre otros, y así poder interactuar con ellos.

La segunda clase en conjunto, con la tercera citada anteriormente, generan **eventos** que permiten ser atrapados por el sistema operativo para su posterior ejecución. Estos **eventos** pueden ser orientados hacia el teclado o el mouse.

Cuando van orientados al teclado, se ejecutan acciones como: dejar presionada una tecla por un determinado período de tiempo, combinar teclas de funciones, liberar las teclas que han sido presionadas, saber si una tecla ha sido presionada, entre otros.

Si están orientados al mouse, se realizan acciones como: dejar presionado un botón por un período de tiempo, liberar un botón que haya sido presionado, mover el **puntero** hacia una posición de la pantalla, entre otras.

El sistema operativo al escuchar el **evento** generado por estas clases, lo recibe y lo procesa sobre los **objetos** que se encuentren activos dentro de la pantalla principal. Para este sistema, sólo se hace uso de los **eventos** orientados al mouse.

En esta clase se hacen uso de varios eventos que simulan las funcionalidades de un mouse común de dos (2) botones. Para ello, se utilizan las clases "java.awt.Robot" y "java.awt.event.InputEvent". Existen cuatro (4) tipos de **eventos** (Tabla 4.11):

Evento	Descripción
1. mousePress(InputEvent.BUTTON1)	Este **evento** simula el mantener presionado el botón izquierdo del mouse.
2.mouseRelease(InputEvent.BUTTON1)	Este **evento** libera al anterior y simula el haber soltado el botón izquierdo del mouse.
3. mousePress(InputEvent.BUTTON3)	Este **evento** simula el mantener presionado el botón derecho del mouse.
4. mouseRelease(InputEvent.BUTTON1)	Este **evento** libera al anterior y simula el haber soltado el botón derecho del mouse.

Tabla. 4.11. Eventos del mouse. Elaborado por los autores.

4.4.4.1 Movimiento del puntero del mouse: Esta función mueve el puntero del mouse hacia la dirección que le sea indicada por la clase *ImagenHilo.java*.

4.4.4.2 Presionar el botón izquierdo del mouse: Esta función consiste en simular el pulsar y soltar el botón izquierdo del mouse (llamado también clic izquierdo), y se lleva a cabo a través de la generación del **evento** 1 seguido del **evento** 2, explicados en la Tabla 4.11.

4.4.4.3 Presionar el botón derecho del mouse: Esta función consiste en simular el pulsar y soltar el botón derecho del mouse (llamado también clic derecho), y se lleva a

cabo a través de la generación del **evento** 3 seguido del **evento** 4, explicados en la Tabla 4.11.

4.4.4.4 Presionar dos veces el botón izquierdo del mouse: Esta función consiste en simular el pulsar y soltar dos veces el botón izquierdo del mouse de manera consecutiva (llamado también doble clic), y se lleva a cabo por la generación repetida de los **eventos** 1 y 2, explicados en la Tabla 4.11.

4.4.4.5 Dejar botón izquierdo del mouse presionado: Esta función consiste en simular el dejar presionado el botón izquierdo del mouse por un tiempo indefinido hasta que sea soltado por otro comando, y se lleva a cabo por la generación del **evento** 1, explicado en la Tabla 4.11.

4.4.4.6 Soltar el botón izquierdo del mouse: Esta función consiste en simular el soltar el botón izquierdo del mouse, y se lleva a cabo por la generación del **evento** 2, explicado en la Tabla 4.11.

4.4.4.7 Dejar botón derecho del mouse presionado: Esta función consiste en simular el dejar presionado el botón derecho del mouse por un tiempo indefinido hasta que sea soltado por otro comando, y se lleva a cabo por la generación del **evento** 3, explicado en la Tabla 4.11.

4.4.4.8 Soltar el botón derecho del mouse: Esta función consiste en simular el soltar el botón derecho del mouse, y se lleva a cabo por la generación del **evento** 4, explicado en la Tabla 4.11.

4.5 Pruebas

Para la realización de este trabajo, se realizaron pruebas de caja negra mediante los siguientes casos de prueba:

Caso de prueba Nº 1: Mover puntero del mouse.

Descripción: Prueba que un individuo pueda mover el puntero del mouse con el movimiento de la cabeza.

Entrada: Se mueve la cabeza hacia abajo.

Salida esperada: Se mueve el **puntero** del mouse hacia abajo a velocidad constante.

Tes. 2.1 Curso alternativo: Marca mal ubicada.

Descripción: Prueba que si la marca no esta ubicada dentro de los cuadrantes, el **puntero** no se mueve.

Entrada: Se gira la cabeza a la izquierda de manera que la cara quede de perfil a la cámara.

Salida esperada: La aplicación no mueve el **puntero** del mouse.

Caso de Prueba Nº 2: Presionar el botón izquierdo del mouse.

Descripción: Prueba que un individuo pueda presionar el botón izquierdo del mouse con un comando de voz.

Entrada: Se coloca el puntero sobre el botón de inicio y se menciona el comando "go click".

Salida esperada: Se presiona el botón izquierdo y se despliega el menú de inicio.

Tes. 2.1 Curso alternativo: Comando incorrecto.

Descripción: Prueba que si el individuo dice un comando que no es el correcto, la aplicación no presiona el botón izquierdo.

Entrada: Se coloca el puntero sobre el botón de inicio y se menciona el comando "botón izquierdo".

Salida esperada: La aplicación no presiona el botón izquierdo.

Caso de uso Nº 3: Presionar el botón derecho del mouse.

Descripción: Prueba que un individuo pueda presionar el botón derecho del mouse con un comando de voz.

Entrada: Se coloca el puntero sobre el escritorio y se menciona el comando "click der"

Salida esperada: Se despliega un menú emergente.

Tes. 3.1 Curso alternativo: Comando incorrecto.

Descripción: Prueba que si el individuo dice un comando que no es el correcto, la aplicación no presiona el botón derecho.

Entrada: Se coloca el puntero sobre el escritorio y se menciona el comando "botón derecho".

Salida esperada: La aplicación no presiona el botón derecho.

Caso de uso Nº 4: Presionar 2 veces el botón izquierdo del mouse.

Descripción: Prueba que un individuo pueda presionar 2 veces el botón izquierdo con un comando de voz.

Entrada: Se coloca el puntero sobre un ícono del escritorio y se menciona el comando "double click".

Salida esperada: Se abre la aplicación seleccionada.

Tes. 4.1 Curso alternativo: Comando incorrecto.

Descripción: Prueba que si el individuo dice un comando que no es el correcto, la aplicación no presiona 2 veces el botón izquierdo.

Entrada: Se coloca el puntero sobre un ícono del escritorio y se menciona el comando "dos clic".

Salida esperada: La aplicación no presiona 2 veces el botón izquierdo.

Caso de uso Nº 5: Seleccionar objetos con el mouse.

Descripción: Prueba que un individuo pueda **seleccionar** un grupo de **objetos**, archivos, etc. con un comando de voz.

Entrada: Posicionarse sobre el escritorio, mencionar el comando "select object", desplazar el puntero para terminar de **seleccionar** los **objetos** y mencionar el comando "go release".

Salida esperada: Los **objetos** quedan sombreados, indicando que han sido seleccionados.

Tes. 5.1 Curso alternativo: Comando incorrecto.

Descripción: Prueba que si el individuo dice un comando que no es el correcto, la aplicación no realiza la selección de **objetos**.

Entrada: Posicionarse sobre el escritorio, mencionar el comando "seleccionar", desplazar el puntero para terminar de **seleccionar** los **objetos** y mencionar el comando "soltar".

Salida esperada: La aplicación no realiza la selección de **objetos**.

Caso de uso Nº 6: Arrastrar objetos con botón izquierdo del mouse.

Descripción: Prueba que un individuo pueda **arrastrar** un conjunto de **objetos**, archivos, etc. con el botón izquierdo de un lugar a otro con un comando de voz.

Entrada: Posicionarse sobre el o los **objetos** a **arrastrar**, mencionar el comando "move object", moverse hacia el lugar de destino y mencionar el comando "go release".

Salida esperada: Mueve el o los **objetos** del lugar de origen al lugar de destino.

Tes. 6.1 Curso alternativo: Comando incorrecto.

Descripción: Prueba que si el individuo dice un comando que no es el correcto, la aplicación no realiza el desplazamiento de los **objetos** con el botón izquierdo.

Entrada: Posicionarse sobre el o los **objetos** a **arrastrar**, mencionar el comando "arrastrar", moverse hacia el lugar de destino y mencionar el comando "soltar".

Salida esperada: La aplicación no realiza el movimiento del o de los **objetos**.

Caso de uso Nº 7: Arrastrar objetos con botón derecho del mouse.

Descripción: Prueba que un individuo pueda arrastrar un conjunto de **objetos**, archivos, etc. con el botón derecho de un lugar a otro con un comando de voz.

Entrada: Posicionarse sobre el o los **objetos** a **arrastrar**, mencionar el comando "move der", moverse hacia la papelera de reciclaje y mencionar el comando "go release".

Salida esperada: Se despliega un menú emergente.

Tes. 7.1 Curso alternativo: Comando incorrecto.

Descripción: Prueba que si el individuo dice un comando que no es el correcto, la aplicación no realiza el desplazamiento de los **objetos** con el botón derecho.

Entrada: Posicionarse sobre el o los **objetos** a **arrastrar**, mencionar el comando "arrastrar derecho", moverse hacia la papelera de reciclaje y mencionar el comando "soltar".

Salida esperada: La aplicación no hace nada.

Caso de uso Nº 8: Soltar botón del mouse.

Descripción: Prueba que un individuo pueda soltar el botón del mouse luego de **arrastrar** o **seleccionar** un conjunto de **objetos**, archivos, etc. con un comando de voz.

Entrada: Mencionar el comando "go release".

Salida esperada: Libera el último botón presionado.

Tes. 8.1 Curso alternativo: Comando incorrecto.

Descripción: Prueba que si el individuo dice un comando que no es el correcto, la aplicación no hace nada.

Entrada: Mencionar el comando "soltar".

Salida esperada: La aplicación no hace nada.

RESULTADOS

A continuación, se exponen los resultados obtenidos durante el desarrollo del sistema a través de las diversas iteraciones ejecutadas. Estos sirvieron para conseguir el objetivo general de esta tesis.

5.1 Alcance de los objetivos específicos

A continuación, se describe como fueron alcanzados los objetivos específicos:

Los datos sobre la movilidad de las personas minusválidas (Operadores finales del sistema), fueron obtenidos a través de un estudio sobre los distintos tipos de personas minusválidas (discapacidades motoras), lo que permitió una mejor compresión de los problemas que padecen dichas personas. Esta información está descrita en el punto 2.3 del capítulo II y en el punto 4.1 del capítulo IV.

Se realizó una investigación sobre cómo el computador se comunica con el mouse, dejando como resultado una lista con las funciones que este puede realizar. Esta información ayudó a comprender como interacciona el mouse con el computador, y fue la base que sirvió para buscar y experimentar con diversos lenguajes de programación. Todo esto está descrito en el punto 2.2 del capítulo II y en el punto 4.1 del capítulo IV.

La información de cómo enviar los datos del **hardware** al computador está contemplada por las APIs de Java, esto se explica en el punto 4.1 del capítulo IV.

El determinar barreras, inconvenientes y obstáculos y ¿Cómo eliminarlos o minimizarlos?, se llevó a cabo durante la etapa de análisis y fue útil debido a que permitió clarificar los problemas potenciales que podían desviar el propósito de esta tesis. Esto se describe en el punto 4.2 del capitulo IV.

Los manuales de usuario y su debida explicación están descritos en detalle en el apéndice A.

5.2 Condiciones de uso

Las condiciones ideales para que el producto funcione correctamente son las siguientes:

- El equipo donde va a funcionar el sistema debe estar en buenas condiciones a nivel de **hardware** y **software**.
- La **webcam** debe estar enfocada hacia la cara de la persona que va a utilizar el mouse de manera que ésta quede frente al lente de la cámara.
- El espacio físico donde se encuentre el usuario debe estar iluminado con luz de neón blanca.
- El espacio físico donde se encuentre el usuario debe prescindir de colores iguales o cercanos al color azul.
- Es importante destacar que en el lugar físico donde se encuentre el usuario, debe mantenerse un mínimo de ruido para que los comandos de voz puedan ser procesados correctamente.

CONCLUSIONES

Al realizar este trabajo especial de grado, se pudo demostrar que si es factible el realizar un producto que ayude a las personas discapacitadas a interactuar con la tecnología en computación.

Con este producto, se aumentan las posibilidades de que las personas minusválidas puedan encontrar trabajos, enseñar a otras personas, realizar cosas cotidianas como revisar el correo, realizar un documento digital, conectarse a Internet, entre otras. Al mismo tiempo se les brinda la oportunidad de integrarse a la tecnología y a que no se sientan rechazados por la misma.

Un aspecto vital para desarrollar un buen producto software, es la metodología que se utiliza para la creación del mismo. En el caso de este trabajo especial de grado, se utilizó la metodología del Proceso Unificado, la cual permitió guiar el sistema "mouse para minusválidos" desde sus inicios hasta su culminación de manera exitosa.

La facilidad de uso y el confort están contemplados en este sistema, debido a que las acciones para manejarlo son sencillas y de fácil aprendizaje. Esto le da un valor agregado al sistema permitiendo que su uso se extienda a otras personas y aplicaciones.

Este trabajo puede ser el inicio para que otras personas se motiven a realizar otros productos que ayuden a las personas con problemas motores a que poco a poco formen parte del apasionante mundo informático.

RECOMENDACIONES

Durante el desarrollo de este trabajo especial de grado, surgieron algunos puntos que se pudieran utilizar para futuras investigaciones. A continuación se muestra algunas recomendaciones derivadas de éstos:

- Realizar una versión que siga el movimiento del iris del ojo como puntero del mouse.
- Realizar un modulo que permita trabajar de manera distribuida, utilizando el protocolo RTP.
- Realizar una versión que siga el movimiento de la nariz como puntero del mouse.
- Realizar un mouse que reconozca a la persona que tiene en frente mediante el reconocimiento de patrones de la cara.
- Hacer un producto mediante este mouse, que permita a una persona minusválida enseñar a otras a distancia.
- Realizar adaptaciones al producto, de las funciones que tienen incorporada los mouse modernos. Se podría aprovechar las facilidades del lenguaje Java, para facilitar este trabajo.
- Realizar una nueva versión del sistema para que trabaje con un reconocedor de voz en cualquier idioma.

BIBLIOGRAFÍA

MAZZEO, Andrés. Ciudad Internet > Tecnología > Mundo Tech. 23-04-2003. <http://www.ciudad.com.ar/ar/portales/tecnologia/nota/0,1357,47390,00.asp>.

MONDELO, José. Home page. <http://weblogs.javahispano.org/page/mondelo>.

IBLNEWS. (14 mayo 2003). Inventores preven la muerte del 'ratón' del PC. [en línea]. Disponible en: <http://iblnews.com/news/noticia.php3?id=75973>.

Nouse (Nose as Mouse) Perceptual Vision Technology, 20-12-2002. Disponible en: <http://www.cv.iit.nrc.ca/research/Nouse/index2.html>.

CameraMouse hands free computer control. 2003. Disponible en: <http://www.cameramouse.com>.

CILSA por la integración. 2002. Disponible en: <http://www.cilsa.com.ar/web_espanol/default.asp>.

Fernandez, Ana. 20-03-2001. En Introducción a UML. [en línea]. Disponible en: <http://www-gris.det.uvigo.es/~avilas/UML/UML.html>.

[en línea]. Disponible en: <http://atc.urg.es/~jbernier/asignaturas/perifericos/practicas/practica2.pdf> [25-07-2003].

KIRSCHNING, Ingrid. Procesamiento Automático de Voz. [en línea]. Disponible en: <http://mailweb.udlap.mx/~ingrid/ingrid/RV/Proc_de_Voz.html>.

HERNANDEZ, Jéssica. Capítulo 4. Herramientas para el desarrollo del sistema. En Ambiente de aprendizaje interactivo en Internet, basado en la tecnología JSP para la Educación Ambiental. [en línea]. Disponible en: <http://mail.udlap.mx/~tesis/lis/hernandez_s_ja/capitulo4.pdf>.

PRADES, Carlos. (2001). Tratamiento multimedia en Java con JMF. [en línea]. Disponible en: <http://cprades.eresmas.com/Tecnica/programarJMF.pdf>.

DUARTE, Domingo, VÁZQUEZ, Raimundo, SAMPALLO, Guillermo, GONZÁLEZ, Arturo. Implementación. En Segmentación de Imágenes mediante su Histograma. [en línea]. Disponible en: <http://www.unne.edu.ar/cyt/2001/7-Tecnologicas/T-084.pdf>.

CASTILLO, Carlos, CLAVIJO, Carlos y CORREDOR, Edgar. Marco teórico. En Sistema automático para localización de rostros en imágenes de video. [en línea]. Disponible en: <http://venus.javeriana.edu.co/tgrado/99-2/Rostros.PDF>.

Sun microsystems. (1998). Java Speech API Programmer's Guide [en línea]. Disponible en:
<http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide.pdf>.

Sun microsystems. (1998). Java Speech Grammar Format Specification [en línea]. Disponible en:
<http://java.sun.com/products/java-media/speech/forDevelopers/JSGF.pdf>.

Sun microsystems. (1997). Java Speech Markup Language Specification [en línea]. Disponible en:
<http://java.sun.com/products/java-media/speech/forDevelopers/JSML.pdf>.

LUCENA, Gonzalo y FERREIRA, Charles. Herramienta para la asistencia al análisis y diseño orientado a objetos. Tesis (Especial de Grado). Caracas, Venezuela. Universidad Católica Andrés Bello. 2002. 77 p.

BAEBERA, Michelangelo y PEREZ, Jeannette. Sistema automatizado para el control de la información de los programas asistenciales, sociales y recreacionales de la fundación amigos del niño con cáncer. Tesis (Especial de Grado). Caracas, Venezuela. Universidad Nueva Esparta. 2002. 209 p.

JACOBSON, Ivar., BOOCH, Grandy, y RUMBAUGH, James. El proceso unificado de desarrollo de software. Madrid, España, Addison Wesley. 2000. 438 p.

Pressman, Roger S. Ingeniería del software un enfoque práctico, cuarta edición. Madirid, España, Mc Graw Hill. 1998. 581 p.

Schildt, Herbert. Manual de referencia Java 2, cuarta edición. Madrid, España, Mc Graw Hill. 2001. 957p.

MANUAL DE USUARIO

A.1 ¿Cómo utilizar el mouse?

En principio, el usuario debe colocarse la marca de color azul en la frente. Luego debe ubicarse frente al computador con la webcam enfocada a la cara.

A.1.1 ¿Cómo mover el cursor del mouse?

El usuario debe mover la cabeza en una de las direcciones de los cuadrantes (Fig. A.1). Dependiendo a donde quiera ir (en sentido diagonal, horizontal o vertical) debe mantener la cabeza en esa posición hasta que el puntero llegue al lugar deseado. Para detener el puntero del mouse el usuario debe colocar la cabeza de frente a la pantalla, para que la marca quede en el cuadrante del centro.

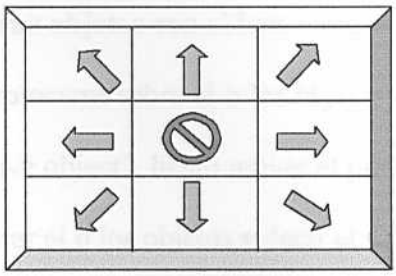

Fig. A.1. Direcciones de movimiento. Elaborado por los autores.

A.1.2 ¿Cómo presionar el botón izquierdo del mouse? (clic izquierdo)

El usuario debe colocarse sobre el objeto que desea hacer clic izquierdo y decir el comando de voz "go click".

A.1.3 ¿Cómo presionar el botón derecho del mouse? (clic derecho)

El usuario debe colocarse sobre el objeto que desea hacer clic derecho y decir el comando de voz "click der".

A.1.4 ¿Cómo presionar 2 veces el botón izquierdo del mouse? (doble clic)

El usuario debe colocarse sobre el objeto que desea hacer doble clic y decir el comando de voz "double click".

A.1.5 ¿Cómo seleccionar objetos con el mouse?

El usuario debe colocarse cerca del o de los objetos que desea seleccionar, decir el comando de voz "select object", mover el puntero del mouse para sombrear el área a seleccionar y decir el comando de voz "go release".

A.1.6 ¿Cómo arrastrar objetos con el botón izquierdo del mouse?

El usuario debe colocarse sobre el o los objetos que desea arrastrar y decir el comando de voz "move object", luego mover el puntero del mouse hacia el lugar donde se desea arrastrar el o los objetos y decir el comando de voz "go release".

A.1.7 ¿Cómo arrastrar objetos con el botón derecho del mouse?

El usuario debe colocarse sobre el o los objetos que desea arrastrar y decir el comando de voz "move der", luego mover el puntero del mouse hacia el lugar donde se desea arrastrar el o los objetos y decir el comando de voz "go release".

A.1.8 ¿Cómo soltar el botón mouse?

Esta acción se realiza luego de seleccionar un conjunto de objetos o después de arrastrar objetos de un lugar a otro. El usuario debe decir el comando de voz "go release".

B.1 Clases y Objetos[1]

Un objeto es una representación de una entidad, ya sea real o conceptual, con límites bien definidos y con un significado dentro de un modelo. Cada objeto se caracteriza por su estado, su comportamiento y su identidad.

El estado de un objeto es una de las posibles condiciones bajo las que éste puede existir. Dicho estado cambia con el tiempo y está definido por un conjunto de propiedades (atributos), y sus valores y además, por las relaciones que dicho objeto puede tener con otros.

El comportamiento de un objeto determina la forma en que éste responde ante las peticiones de otros, y tipifica todo lo que el objeto puede hacer. Este comportamiento se materializa en el conjunto de operaciones definidas para dicho objeto.

La identidad implica que cada objeto es único, incluso si su estado es idéntico al de otro objeto.

B.2 Lenguaje Unificado de Modelado (UML)[1]

El lenguaje unificado de modelado (UML) es un lenguaje estándar de modelado gráfico para **software**, que permite la visualización, especificación, construcción y documentación de los **artefactos** del sistema, en los que el **software** juega un papel importante. Además, el estándar proporciona un marco en el que los desarrolladores

[1] Fernandez, Ana. (2001). Introducción a UML. [en línea]. Disponible en: <http://www-gris.det.uvigo.es/~avilas/UML/node33.html>. [2003, Octubre 22]

pueden analizar y diseñar, evitándoles el duro trabajo de memorizar información durante meses ya que queda todo registrado en el papel o en formato electrónico.

B.2.1 Diagramas en UML

B.2.1.1 Diagrama de Casos de Uso[1]

El modelo de casos de uso ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de los sistemas tienen muchos tipos de usuarios, y cada tipo se representa mediante un actor.

Los actores utilizan el sistema al interactuar con los casos de uso, los cuales proporcionan un medio sistemático e intuitivo de capturar requisitos funcionales centrándose en el valor añadido para el usuario. Un diagrama de caso de uso describe parte del modelo de casos de uso y muestra un conjunto de estos diagramas y actores con una asociación entre cada par actor/caso de uso que interactúan.

B.2.1.2 Diagrama de Clases[1]

El diagrama de clase describe los tipos de objetos que hay en el sistema y las diversas clases de relaciones estáticas que existen entre ellos. Es una **vista** estática porque no describe el comportamiento del sistema dependiente del tiempo, que se describe en otros diagramas. Los diagramas de clases también muestran los atributos (características de los objetos) y las operaciones (procesos o métodos que se pueden ejecutar) de una clase, representan las relaciones entre las instancias de la misma. Cada asociación posee dos finales, cada final está ligado a una de las clases de la asociación.

[1] Ident.

B.2.1.3 Diagramas de Secuencia[1]

Un diagrama de secuencia muestra las interacciones entre objetos, ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre ellos para llevar a cabo la funcionalidad descrita por el escenario. Los diagramas de secuencia, son formalmente diagramas de traza de **eventos** o de interacción de objetos, que se utilizan con frecuencia para validar los casos de uso. También ayudan a comprender los cuellos de botella potenciales para así poder eliminarlos.

B.2.1.4 Diagrama de Componentes[1]

Un diagrama de componentes muestra las relaciones físicas entre los componentes **hardware** y **software** en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes **software** (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes **software** que representan manifestaciones del código en tiempo de ejecución (los componentes que sólo sean utilizados en tiempo de compilación deben mostrarse en el diagrama de componentes).

[1] Ident.

B.3 Casos de uso

Caso de uso Nº 1: Mover puntero del mouse

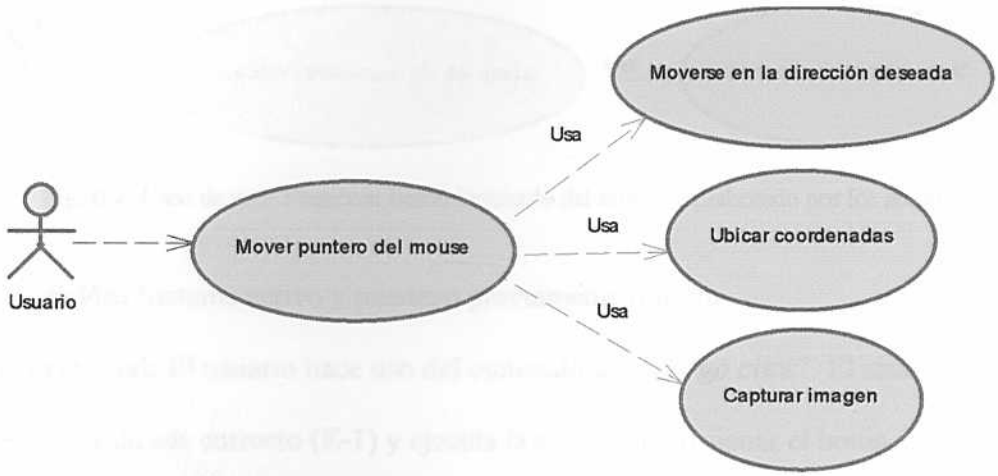

Fig. B.1. Caso de uso "Mover puntero del mouse". Elaborado por los autores.

Precondición: Sistema activo.

Flujo Principal: El usuario mueve la cabeza en dirección a uno de los cuadrantes para mover el **puntero** (superior, superior izquierdo, superior derecho, derecho, centro, izquierdo, inferior izquierdo, inferior e inferior derecho). El sistema captura la imagen, la analiza y determina en que cuadrante se encuentra la marca que usa el usuario **(E-1)** en la cabeza. Seguido a esto el sistema mueve el **puntero** del mouse a velocidad constante en dirección al cuadrante seleccionado hasta llegar a la posición deseada. Luego se coloca la marca en el centro de la imagen, el **puntero** del mouse se detiene y el caso de uso termina.

Flujo Alterno:

E-1. Marca mal ubicada. Si la marca no está ubicada en uno de los cuadrantes antes descritos, la aplicación no hace nada y el caso de uso termina.

Caso de uso Nº 2: Presionar botón izquierdo del mouse.

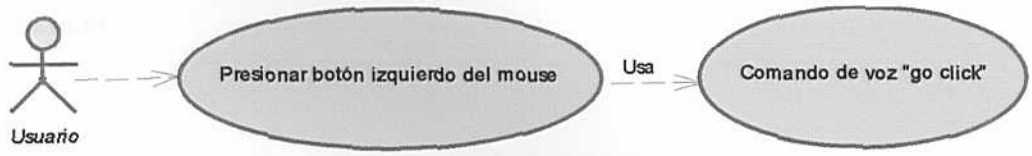

Fig. B.2. Caso de uso "Presionar botón izquierdo del mouse". Elaborado por los autores.

Precondición: Sistema activo y **puntero** previamente ubicado.

Flujo Principal: El usuario hace uso del comando de voz "*go click*". El sistema verifica que el comando sea correcto **(E-1)** y ejecuta la acción de presionar el botón izquierdo del mouse. Luego el caso de uso termina.

Flujo Alterno:

E-1. El comando no es el correcto. El sistema no presiona el botón izquierdo y el caso de uso termina.

Caso de uso Nº 3: Presionar botón derecho del mouse.

Fig. B.3. Caso de uso "Presionar botón derecho del mouse". Elaborado por los autores.

Precondición: Sistema activo y **puntero** previamente ubicado.

Flujo Principal: El usuario hace uso del comando de voz "*click der*". El sistema verifica que el comando sea correcto **(E-1)** y ejecuta la acción de presionar el botón derecho del mouse. Luego el caso de uso termina.

Flujo Alterno:

E-1. El comando no es el correcto. El sistema no presiona el botón derecho y el caso de uso termina.

Caso de uso Nº 4: Presionar 2 veces botón izquierdo del mouse.

Fig. B.4. Caso de uso "Presionar 2 veces botón izquierdo del mouse". Elaborado por los autores.

Precondición: Sistema activo y **puntero** previamente ubicado.

Flujo Principal: El usuario hace uso del comando de voz "*double click*". El sistema verifica que el comando sea correcto **(E-1)** y ejecuta la acción de presionar dos (2) veces el botón izquierdo del mouse. Luego el caso de uso termina.

Flujo Alterno:

E-1. El comando no es el correcto. El sistema no presiona dos (2) veces el botón izquierdo y el caso de uso termina.

Caso de uso Nº 5: Seleccionar objetos con el mouse.

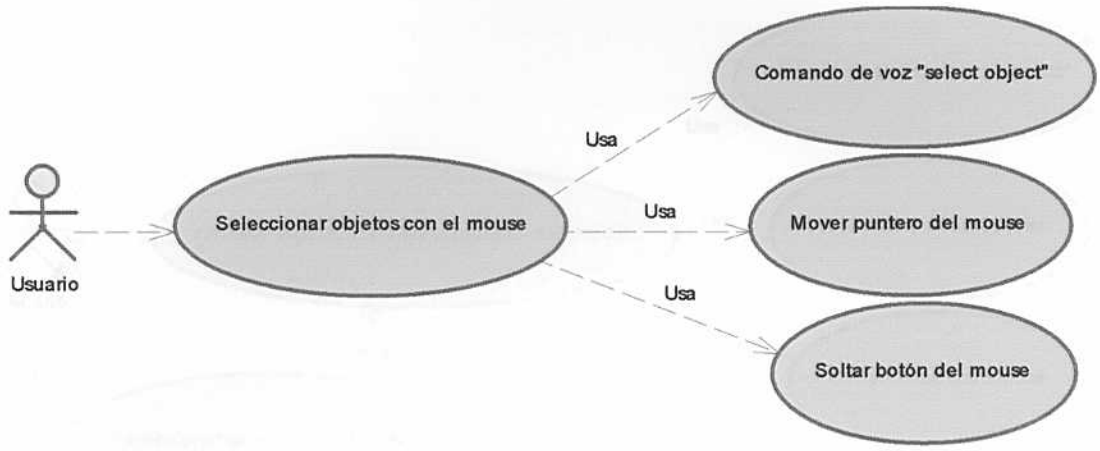

Fig. B.5. Caso de uso "Seleccionar objetos con el mouse". Elaborado por los autores.

Precondición: El sistema debe estar funcionando.

Flujo Principal: El usuario hace uso del comando de voz "*select object*" **(E-1)** y el sistema verifica que el comando sea correcto **(E-2)**. Luego, se debe mover el **puntero** para seleccionar el o los objetos que desea seleccionar. Finalmente debe hacer uso del comando de voz "*go release*". El sistema verifica que el comando sea correcto **(E-2)** y luego el caso de uso termina.

Flujo Alterno:

E-1. El sistema debe verificar que no existan casos de uso esperando ser liberados, como lo son: "Arrastrar objetos con botón izquierdo del mouse", "Arrastrar objetos con botón derecho del mouse" y "Seleccionar objetos con el mouse" que requieren del comando de voz "*go release*" para terminar. Luego el caso de uso termina.

E-2. El comando no es el correcto. El sistema no realiza la selección y el caso de uso termina.

Caso de uso Nº 6: Arrastrar objetos con botón izquierdo del mouse.

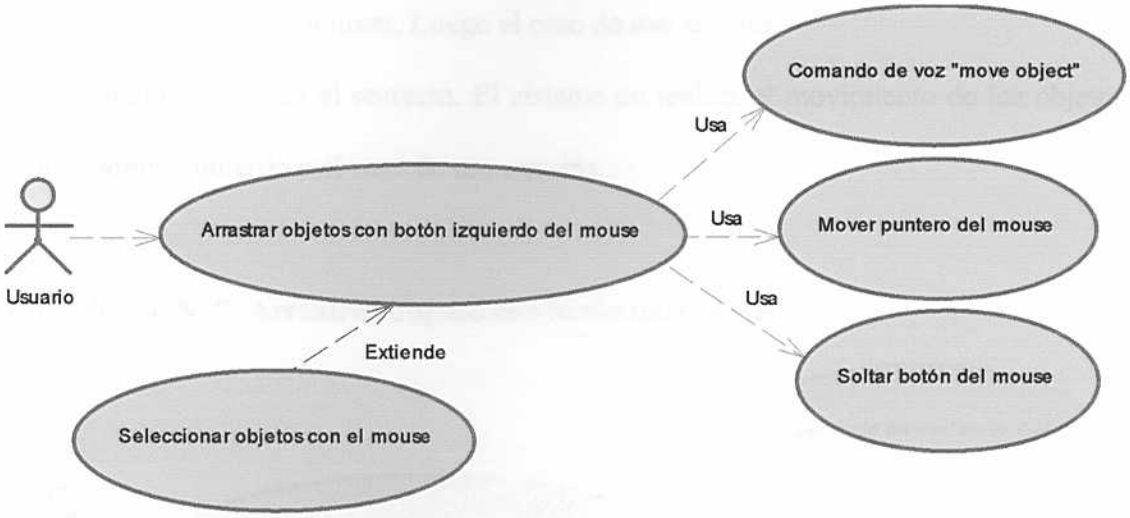

Fig. B.6. Caso de uso "Arrastrar objetos con botón izquierdo del mouse". Elaborado por los autores.

Precondición: El sistema debe estar funcionado.

Flujo Principal: El usuario debe colocar el **puntero** sobre el objeto que desea arrastrar **(S-1)**. Luego hace uso del comando de voz *"move object"* **(E-1)** y el sistema verifica que el comando sea correcto **(E-2)**. Seguidamente mueve el **puntero** hacia la posición donde quiere dejar el o los objetos y hace uso del comando de voz "*go release*" para soltarlos, el sistema verifica que el comando sea correcto **(E-2)** y luego el caso de uso termina.

Subflujos

S-1. Arrastrar varios objetos.

El usuario debe utilizar el caso de uso "seleccionar objetos con el mouse", para que los objetos que desea arrastrar queden seleccionados.

Flujo Alterno:

E-1. El sistema debe verificar que no existan casos de uso esperando ser liberados, como lo son: "Arrastrar objetos con botón izquierdo del mouse", "Arrastrar objetos con botón

derecho del mouse" y "Seleccionar objetos con el mouse" que requieren del comando de voz "*go release*" para terminar. Luego el caso de uso termina.

E-2. El comando no es el correcto. El sistema no realiza el movimiento de los objetos con el botón izquierdo y el caso de uso termina.

Caso de uso Nº 7: Arrastrar objetos con botón derecho del mouse.

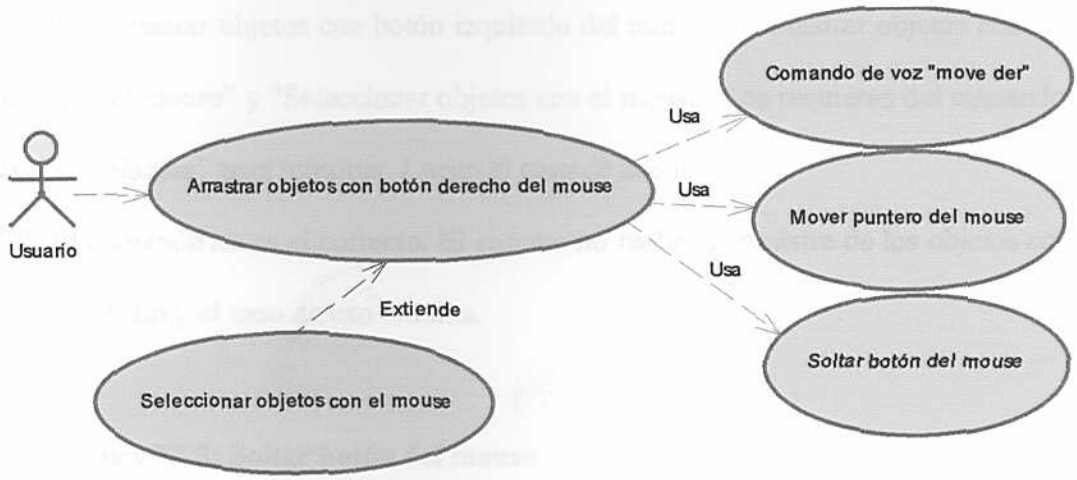

Fig. B.7. Caso de uso "Arrastrar objetos con botón derecho del mouse". Elaborado por los autores.

Precondición: El sistema debe estar funcionado.

Flujo Principal: El usuario debe colocar el **puntero** sobre el objeto que desea arrastrar **(S-1)**. Luego hace uso del comando de voz "*move der*" **(E-1)** y el sistema verifica que el comando sea correcto **(E-2)**. Seguidamente mueve el **puntero** hacia la posición donde quiere dejar el o los objetos y hace uso del comando de voz "*go release*" para soltarlos, el sistema verifica que el comando sea correcto **(E-2)** y luego el caso de uso termina.

Subflujos

S-1. Arrastrar varios objetos.

El usuario debe utilizar el caso de uso "seleccionar objetos con el mouse", para que los objetos que desea arrastrar queden seleccionados.

Flujo Alterno:

E-1. El sistema debe verificar que no existan casos de uso esperando ser liberados, como lo son: "Arrastrar objetos con botón izquierdo del mouse", "Arrastrar objetos con botón derecho del mouse" y "Seleccionar objetos con el mouse" que requieren del comando de voz "*go releasse*" para terminar. Luego el caso de uso termina.

E-2. El comando no es el correcto. El sistema no realiza el arrastre de los objetos con el botón derecho y el caso de uso termina.

Caso de uso Nº 8: Soltar botón del mouse

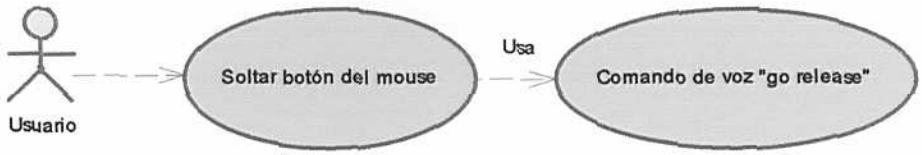

Fig. B.8. Caso de uso "Soltar botón del mouse". Elaborado por los autores.

Precondición: El sistema debe estar funcionando, y debe existir un caso de uso esperando ser liberado ("Arrastrar objetos con botón izquierdo del mouse", "Arrastrar objetos con botón derecho del mouse", "Seleccionar objetos con el mouse").

Flujo Principal: El usuario hace uso del comando de voz "*go release*" **(E-1)** y suelta virtualmente el botón presionado por la ejecución de un comando de voz, luego el caso de uso termina.

Flujo Alterno:

E-1. El comando no es el correcto. El sistema no libera el caso de uso en espera y el caso de uso termina.

GLOSARIO

API (Interfaz para programas de aplicación): Conjunto de convenciones de programación que definen cómo se invoca un servicio desde un programa.

Artefacto: Pieza de información tangible que (1) es creada, modificada y usada por los trabajadores al realizar actividades; (2) representa un área de responsabilidad, y (3) es candidata a ser tenida en cuenta para el control de la configuración. Un artefacto puede ser un modelo, un elemento de un modelo, o un documento.

Arrastrar Objetos: Mover un elemento por la pantalla seleccionándolo y manteniendo presionado el botón del mouse (ratón) mientras se mueve el mouse. Por ejemplo, puede mover una ventana para llevarla a otra ubicación de la pantalla arrastrando su barra de título. Tomado de la ayuda del sistema operativo Windows XP.

Comunicaciones multimedia: Tecnologías necesarias para manipular, transmitir y controlar las señales audiovisuales a través de redes de datos.

Controlador: Programa que permite a un dispositivo específico, como un mouse, un adaptador de red o una impresora, comunicarse con el sistema operativo. Tomado de la ayuda del sistema operativo Windows XP.

Evento: La especificación de una ocurrencia significativa que tiene una ubicación en el tiempo y en el espacio; en el contexto de máquinas de estados, un evento es una ocurrencia de un estímulo que puede disparar una transición de estados.

Flujo Alterno: Descripción de cómo un caso de uso termina de manera no debida.

Flujo Principal: Descripción de cómo y cuando un caso de uso inicia y termina, cómo y cuando este interactúa con los usuarios y que objetos son intercambiados durante esta interacción.

Hardware: Componentes físicos de un sistema informático, incluidos todos los periféricos, como los módems, impresoras y *mouse* (ratón). Tomado de la ayuda del sistema operativo Windows XP.

Hilo: Es la unidad de código más pequeña que se puede seleccionar y define un camino de ejecución independiente y de manera concurrente al que lo genera. Esto hace que los programas sean muy eficientes y que realicen una utilización óptima del CPU.

Método: La implementación de una operación.

Multimedia: Representación mezclada de diferentes modos de información (texto, datos, imagen, audio y vídeo) en señales digitales.

Objetos de windows: Entidad, como un archivo, carpeta, carpeta compartida, impresora u objeto de Active Directory, descrita por un conjunto de atributos diferenciado y con nombre.

Píxel: Abreviatura de *picture element*, que significa elemento de imagen. Es un punto de una cuadrícula rectilínea de miles de puntos que forman una imagen generada por un equipo, en la pantalla, o por una impresora, en el papel. Tomado de la ayuda del sistema operativo Windows XP.

Player: Objeto encargado de procesar cierto flujo de datos multimedia y presentarlos en su preciso momento.

Puerto USB: Interfaz en el equipo que permite conectar un dispositivo Bus serie universal (USB).

Puntero del mouse: Pequeña flecha u otro tipo de indicador que se desplaza sobre la pantalla del ordenador y que es manejado por el mouse.

Registro: Depósito de base de datos que contiene información acerca de la configuración de un equipo. Tomado de la ayuda del sistema operativo Windows XP.

Relación caso de uso (extiende): Modela la adición de una secuencia de acciones a un caso de uso. Una extensión se comporta como si fuera algo que se añade a la descripción original de un caso de uso.

Relación caso de uso (usa): Define una clase de herencia, en la que un caso de uso A hacia un caso de uso B, indica que una instancia del caso de uso A incluirá también el comportamiento especificado en B.

Resolución de la pantalla: Configuración que determina la cantidad de información que aparece en la pantalla, medida en píxeles. Tomado de la ayuda del sistema operativo Windows XP.

RTP (Protocolo de Tiempo Real): Protocolo utilizado para la transmisión de información en tiempo real, como por ejemplo audio y vídeo en una videoconferencia.

Seleccionar: Resaltar o marcar un conjunto de objetos para especificarlos, con la intención de realizar con ellos alguna operación. Tomado de la ayuda del sistema operativo Windows XP.

Software: Conjunto de programas, instrucciones y reglas informáticas que sirven para ejecutar ciertas tareas en una computadora.

SubFlujo: Descripción de las diferentes opciones que puede tomar un usuario para la culminación de un caso de uso.

Vector: Serie ordenada de elementos utilizados en una consulta.

Ventana: Parte de la pantalla donde se pueden ejecutar los programas y los procesos. Tomado de la ayuda del sistema operativo Windows XP.

Vista: Proyección de un modelo, que es visto desde una perspectiva dada o un lugar estratégico, y que omite las entidades que no son relevantes para esta perspectiva.

Webcam: Dispositivo hardware de puerto USB, que permite capturar imágenes de manera consecutiva.

Anexos

Movimiento del usuario para mover el puntero del mouse a velocidad constante, hacia la parte inferior derecha de la pantalla.

Ubicación del usuario para que el puntero del mouse no se desplace.

ANEXOS

Movimiento del usuario para mover el puntero del mouse a velocidad constante, hacia la parte superior derecha de la pantalla.

Movimiento del usuario para mover el puntero del mouse a velocidad constante, hacia la izquierda de la pantalla.