BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Interfaces Adaptativas

Trabajo de Grado

- LACOSTA Marisa
- FAVA Laura

DIRECTOR :
DIAZ Javier

TES
96/6
DIF-01932
SALA

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMATICA
Biblioteca
50 y 120 La Plata
catalogo.info.unlp.edu.ar
biblioteca@info.unlp.edu.ar

DIF-01932

Introducción

El presente trabajo de grado surgió para canalizar inquietudes respecto de la relación de los sistemas de aplicación con las personas que los utilizan.

Si bien la separación y el desacoplamiento, de la interfaz y la aplicación se logra a partir de los sistemas de gerenciamiento de interfaz de usuario -abreviados UIMS en la literatura- el proyecto decidió focalizar en un área nueva como ser los **sistemas adaptativos**, o de fácil y aún automática personalización.

A partir del análisis de las características deseables para el usuario, las posibilidades de las necesidades de la interfaz, y de los modelos de representación de interfaz de usuario mas establecidos, se llegó a un modelo de desarrollo conceptualmente distinto.

El modelo descripto en el capítulo VI, es conceptual, y cubre las facetas susceptibles de ser mejoradas en las interfaces actuales. Por su complejidad hace uso de diversas teorías, desde aprendizaje, generación de planes, bases de conocimientos, etc. La carencia de un kernel básico donde estas características -modelización de usuarios y adaptatividad- pudieran agregarse, llevó a la necesidad de desarrollar un prototipo de generador en Smalltalk, necesariamente incompleto pero conceptualmente completo, como para estudiar los alcances y limitaciones.

A fin de testear las posibilidades del mismo, se desarrolló la herramienta CUIMS, que es un gerenciador de interfaces de usuarios, **con características innovativas**. El mismo se desarrolló en Smalltalk y realmente constituye un sistema de alta complejidad en si mismo.

Por último y a fin de ilustrar las capacidades de la implementación del prototipo, y las características de los sistemas implementados a partir de él, se instancia la interfaz de usuario para una aplicación que realiza el soporte de una agenda.

Los pasos seguidos fueron complejos, pero el punto final resulta una herramienta valiosa para el desarrollo de prototipos rápidos y para el estudio de sistemas adaptativos y/o personalizables. A fin de facilitar la lectura, a lo largo de los capítulos se asocia la sigla GIA para referirse al modelo teórico propuesto y CUIMS a la implementación de una parte del modelo propuesto GIA.

Agradecimientos:

- A *Javier Diaz*, quien confió en nosotras, nos respaldó y nos aconsejó en aspectos novedosos.
- A los integrantes del *LINTI*, por su aporte de ideas.
- Debemos un agradecimiento muy especial a *Alejandro Raimondo* por su incondicional ayuda y valiosas sugerencias para la etapa de implementación.
- A los *alumnos* de la Cátedra de 'Prototipación e Interfaces', por haber testeado las herramientas desarrolladas.

Indice

III.3 Características customizables de los estilos de interacción.

Capítulo 4

Gerenciador de Interfaces de Usuarios

IV.1 Introducción. Componentes de un Gerenciador de Interfaces.

Servicios de las UIMS.

IV.2.1. Servicio de Diseño.

IV.2.2. Servicio de Construcción.

IV.2.3. Servicio de Evaluación

IV.2.4. Servicio de Mantenimiento.

IV.3 Herramientas de Especificación de las Componentes de una Gerenciador de Interfaces.

Capítulo 5

Fundamentos para el desarrollo del prototipo

V.1 Introducción.

V.2 Selección del Lenguaje. Evolución.

V.3 Bibliografías Estudiadas

Capítulo 6

Gerenciador de Interfaces Adaptativas - GIA -

VI.1 Introducción.

VI.2 Arquitectura para una Interfaz Adaptativa. Modelo Propuesto.

Capítulo 7

Implementación del Prototipo - CUIMS -

VII.1 Introducción.

VII.2 Características de las interfaces logradas con esta Herramienta.

VII.3 Limitaciones del Smalltalk / V.

VII.4 Modificación de la jerarquía de Smalltalk para el desarrollo de los sistemas.

APÉNDICE

Manual de Usuario

Ayuda para la utilización de las herramientas para Diseñadores de interfaces, Diseñadores de Modelos de Usuarios y Usuarios Finales. Pantallas del sistema y significado de los íconos utilizados.

BIBLIOGRAFIA

1

Customización Automática

I.1 Introducción.

Es sabido, que el rango de usuarios de sistemas de cómputos es cada vez más amplio, estando éste formado por personas muy heterogéneas, que pueden poseer un alto conocimiento de computación o ser novatos en la materia, desempeñarse en distintas disciplinas, poseer distintos intereses, preferencias y hábitos personales, siendo utilizada a menudo la misma interfaz.

Por estas razones, es que en nuestra tesis nos dedicaremos al desarrollo de una herramienta que permita construir interfaces, que posibiliten a un **amplio rango de usuarios interactuar efectivamente con una variedad de tipos de aplicaciones.**

I.2 Definición de customización automática.

La automática, por definición, es la ciencia que trata del estudio y la realización de los mecanismos y sistemas que tienen por finalidad la sustitución del operador humano por un operador artificial en la ejecución de una tarea física o mental previamente programada.

Nosotras automatizaremos algunas tareas del usuario, es decir, la sustitución del operador humano será hecha por medio de procedimientos que realizarán algunas tareas que el usuario efectua con cierta frecuencia. Las tareas a automatizar son específicas de cada humano particular.

¿ Qué es Customización ?

" Es el acto de alterar algunas cosas para satisfacer los requerimientos de una persona. " [LER 89].

La customización es el esfuerzo realizado para hacer o cambiar algunas cosas para adaptarse al estilo de cada usuario individual. Una interfaz customizable está capacitada para cambiar su aspecto y comportamiento, de manera que satisfaga las preferencias de cada usuario en todo momento. Si tal interfaz es customizable **automáticamente**, la adaptación debe ser hecha sin intervención explícita del usuario final.

Los sistemas actuales proveen una gran variedad de facilidades de customización, que le permiten al usuario adaptar el sistema a sus necesidades; por medio de un menú de arranque, o solicitando explícitamente un cambio dentro del ambiente, como el color de la pantalla, estilos de letras, etc.

I.3 Razones para customizar una interfaz de usuario

Existen varias razones para que un sistema se customice, debido a que hay una gran variedad de personas usando un mismo sistema. Haciendo un análisis mas detallado, encontramos diferencias marcadas por distintos aspectos:

- Capacidad de aprendizaje de cada usuario particular, puede estar muy relacionado con el nivel cultural de cada usuario o con la propia capacidad intelectual de cada uno de ellos.
- Experiencias en cuanto al manejo de sistemas, aquellas personas que alguna vez han tenido algún tipo de contacto con la máquina y aquellos que sólo la miran de reojo y temen su utilización.
- Conocimiento en cuanto al dominio de la aplicación, supongamos un Sistema Experto desarrollado para una especialidad en Medicina, no tendrá el mismo desempeño un estudiante recién recibido que el propio experto en la Materia.
- Diferentes roles dentro de una misma aplicación.
- Desempeños en distintas disciplinas o con distintos intereses, supongamos un diseñador publicitario y un abogado utilizando un procesador de textos. Al abogado podría interesarle aprender la utilización de los tabuladores, márgenes, copiados y borrados de bloques, para

un rápido tipeo y una buena presentación , mientras que al diseñador le podría interesar insertar figuras, entremezclarlas con textos y la utilización de una variedad de tipos de letras.

- Distintas preferencias. Así como muchas personas se visten de diferentes maneras cuando van a una fiesta, con distintos vestidos y en colores, un mismo sistema, puede ser utilizado en forma distinta dependiendo de las preferencias de cada persona.
- Hábitos personales. Cada persona, por lo general, diariamente realiza tareas repetidas veces, peinarse, cepillarse sus dientes, etc.; también en el uso de un sistema se pueden hallar personas que desarrollen repetidas veces la misma tarea.

Esto nos lleva a hacer la distinción entre dos tipos de sistemas :

SISTEMAS ADAPTABLES : El sistema admite ser modificado por el usuario. Esta sería la customización hecha por parte del usuario.

SISTEMAS ADAPTATIVOS : El sistema se adapta a las necesidades del usuario, haciendo uso de modelos de comportamientos observados durante la interacción del usuario final con la aplicación. -customización automática-.

I.4 Características de los sistemas customizables.

Que un sistema sea customizable implica que la rigidez del mismo es menor que la de un sistema que no provea esa característica.

Aun cuando los cambios parezcan simples al usuario:

- Se siente mas cerca del sistema, ya que puede influir en aspectos visibles.
- Mejora su productividad pues se pueden evitar características que lo distraigan o lo confundan.
- Tiene mas opciones permitiéndosele, no solo usar la interfaz del sistema, sino operar con ella.
- Se evitan idas y vueltas pidiéndole al diseñador de interfaz que modifique cuestiones que ahora el puede solucionar.

- Refleja cambios que personalizan el sistema, posibilitando que el mismo sistema sea utilizado en forma radicalmente distinto por dos personas con distintas formaciones y gustos personales.

Cabe destacar que customizar permite cambiar un sistema, mientras que customizar automáticamente -personalizar-, posibilita que un mismo sistema, actúe y luzca de diferente forma, cuando lo utilicen distintos usuarios. En este sentido además de situaciones y particularidades de cada sistema que se puedan instanciar, se requiere una identificación plena del usuario que active un comportamiento diferenciado del sistema, lo que implica la necesidad de un molde o registro de usuario donde se mantienen sus características.

I.5 Aspectos a customizar.

Los aspectos de una Interfaz de Usuario que se pueden customizar automáticamente son:

Los Hábitos del Usuario

Customizar los hábitos del usuario implica encontrar secuencias de comandos que el usuario haya aplicado frecuentemente y cuyo resultado haya sido exitoso. Para determinar estas secuencias analizamos la historia de las interacciones del usuario con el sistema. La existencia de esta customización aliviaría al usuario la tarea de desarrollar repetidas veces la misma secuencia de comandos o le evitaría la tarea de crear macros específicas para esas secuencias.

Tareas del Usuario

Customizar tareas del usuario, implica completar tareas que el usuario no desarrolló, infiriendo del contexto en el que se esta desarrollando la aplicación. Un ejemplo sería si estamos editando un programa, para un lenguaje de programación fuertemente tipado y con un editor chequeador semántico, podría inferir que si una variable no esta declarada pero dentro del programa se le asigna un valor=0, corresponde a una variable numérica o con un editor chequeador sintáctico podría deducir ciertas reglas del lenguaje.

Gustos y Preferencias

Customizar gustos y preferencias, implica, customizar helps y mensajes, esto es, permitir la existencia de distintos estilos de helps y mensajes, ya sea, gráficos o textuales o combinación de

ellos, según los tipos de usuarios definidos.Customizar los aspectos de la presentación de las interfaces, consiste en adaptar los estilos de letras, colores de los objetos de interacción y de las ventanas, gráficos preferidos por los usuarios, etc.

En realidad, la customización automática está principalmente orientado a mejorar el uso de un sistema para un usuario en particular. De esta forma, esencialmente las componentes que son más afectadas son las de visualización y las de interacción, por lo que, un análisis de sistemas customizables debe iniciarse en un estudio profundo de las interfaces de usuarios.

2

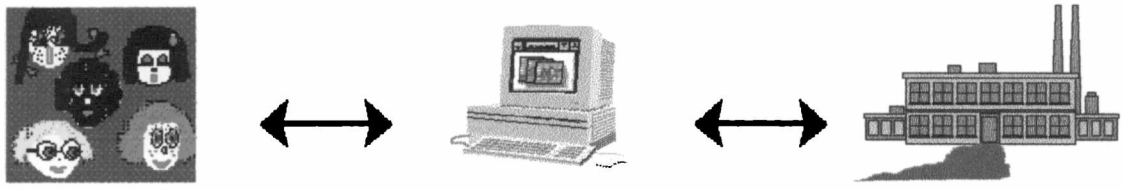
Desarrollo de Interfaces de Usuarios

II.1 Introducción.

Por lo general, una misma aplicación admite distintas maneras de ser utilizada, es decir, diversos usuarios puedan manipular la misma aplicación de diferente forma, haciendo uso de distintas modalidades existentes. Esto es cada día mas evidente, con el surgimiento de los sistemas, especialmente los de estilo Windows, donde la integración de gráficos y textos, juegan un papel fundamental en la calidad de todo sistema. Cuanto más amigable sea la interfaz, mayor será la aceptación por parte de los usuarios.

Conceptualmente, una interfaz de usuario, mantiene toda la información relevante para la interacción Hombre-Computadora y abstrae las características intrínsecas de las aplicación. Una interfaz de usuario, hace uso de las potencialidades del hardware que flexibilizan su uso.

Figura 2.1: Modelo interactivo.



- **Programador de la aplicación** : es el encargado de construir una aplicación de propósitos especiales, su esfuerzo se reduce a un gran numero de comandos, que pueden ser provistos por la herramienta de construcción, entre las mencionadas.

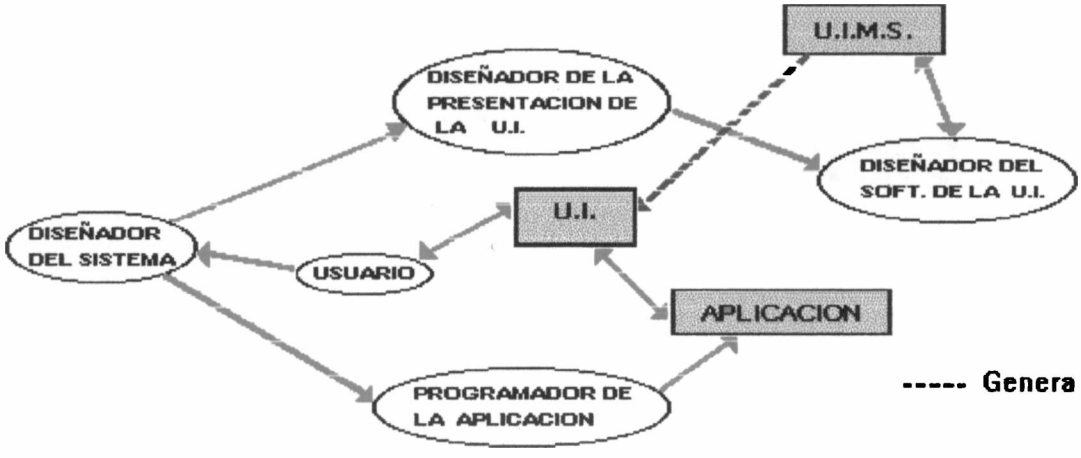
- **Usuario** : es el único que interactúa con la interfaz de usuario. Para diferentes programas de propósitos especiales derivada de la misma herramienta de construcción, este encuentra una cierta consistencia, lo que le permite actuar por analogía.

- **Diseñador del sistema** : este especifica los requerimientos del usuario teniendo en cuenta la existencia de distintas clases de usuarios, (Por ejemplo: novatos, intermedios, expertos) y sus atributos (permisos para realizar ciertas tareas). No tiene una conexión directa con la interfaz de usuario, sí con el diseñador de la presentación de la interfaz de usuario y con el programador de la aplicación.

- **Diseñador de la presentación de la interfaz de usuario** : define aquella parte con la que interactuará el usuario, recibe información del anterior y se esfuerza en representar los requerimientos del usuario por medio de las técnicas de interacción, teniendo en cuenta la posibilidad de los distintos tipos de usuarios y cual será la mejor interacción en cada situación.

- **Diseñador del soft de la interfaz de usuario** : este es el encargado de codificar la interfaz de usuario definida por el diseñador de la presentación de la interfaz de usuario. Este debe combinar las herramientas existentes para la generación de la interfaz de usuario de acuerdo a los niveles de usuarios y sus atributos en el sistema.

Figura 2.2: Esquema de roles.



II.2 Definiciones. Herramientas de construcción.

Definimos a la interfaz de usuario como una herramienta que permite definir secuencias de diálogo interactivo y manejar la interacción física entre la aplicación y el usuario. De acuerdo a la complejidad de las herramientas, podemos básicamente agruparlas, en 2 grandes categorías :

- La primera, menos sofisticada, está formada por las 'User Interface ToolKits', que son las que determinan el uso de dispositivos físicos de entrada/salida (se detallarán en el capítulo III). Con esta herramienta la responsabilidad de coordinar las técnicas de interacción caen en el diseñador o programador de la interfaz de usuario.

- La otra categoría la forman las UIMS, que significa 'User Interface Managment System' y está compuesta por un conjunto de herramientas que intentan facilitar el diseño, prototipación, desarrollo y mantenimiento de las interfaces.

Además de encapsular herramientas, posee una característica que las distingue de las TOOLKITS, esta es, que provee mecanismos para manejar el control del diálogo entre el usuario y la aplicación. (CAPITULO III.1)

II.3 Individuos que intervienen en el desarrollo de un sistema.

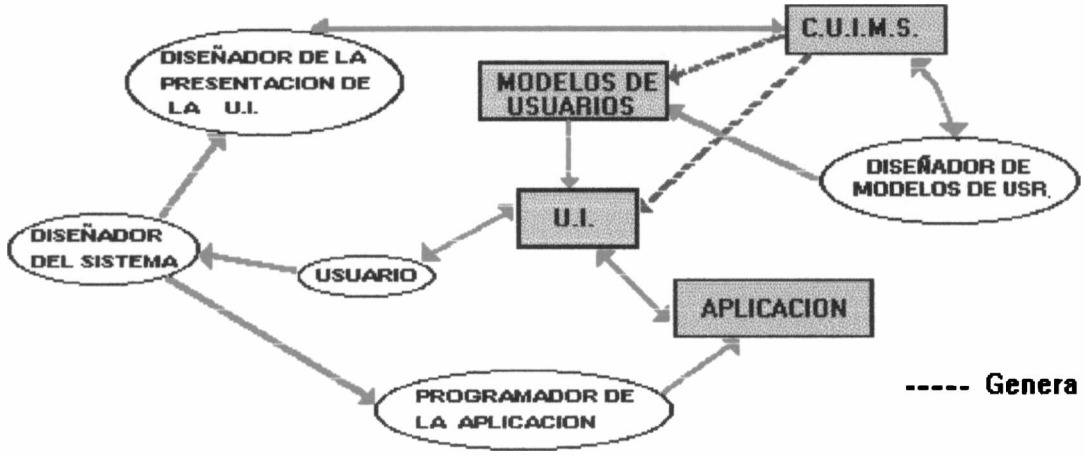
La utilización de un sistema dado, tiene una implicancia que va desde lo operativo a cuestiones estructurales que lo afectan. Estos aspectos, si bien están presentes también en la aplicación, aparecen más fuertemente en las componentes de la interfaz Hombre - Máquina.

De esta forma, se multiplican los actores involucrados, pudiendo enumerarse:

- el programador de la aplicación.
- el usuario.
- el diseñador del sistema.
- el diseñador de la presentación de la interfaz de usuario.
- el diseñador del software de la interfaz de usuario.

Este sería un esquema de roles tradicional, que genera una interfaz de usuario sin ninguna particularidad. Nuestro objetivo principal es generar una interfaz customizable, por lo que debemos incorporar el concepto de **modelo de usuarios**. Para esto el esquema sufre dos alteraciones, el diseñador del soft de la interfaz de usuario no es necesario, dado que el diseñador de la presentación, puede interactuar fácilmente con la herramienta y ésta genera automáticamente el código de la interfaz de usuario, sustituyendo la tarea desempeñada por el diseñador del soft de las interfaces. La otra alteración es la incorporación de los modelos de usuarios. Estos definen los tipos de usuarios que interactuarán con la herramienta de desarrollo -diseñador de la interfaz y diseñador de los modelos de usuarios-.

Figura 2.3: Esquema de roles modificado.



II.4 Conocimientos necesarios en el desarrollo de una interfaz adaptativa.

La **aplicación** es un conjunto de **tareas** dentro de un dominio definido. La secuencia de eventos llevada a cabo desde que el usuario entra a la aplicación hasta que sale de ella se denomina **Sesión.**

Entendemos por **eventos** a aquellas acciones llevadas a cabo por los usuarios por medio de dispositivos físicos, que van desde que el usuario entra los comandos o requerimientos, hasta que el sistema retorna una respuesta. La secuencia básica de fases comienza cuando el usuario entra algún imputa **-fase de input-**, luego el requerimiento es enviado al sistema para su

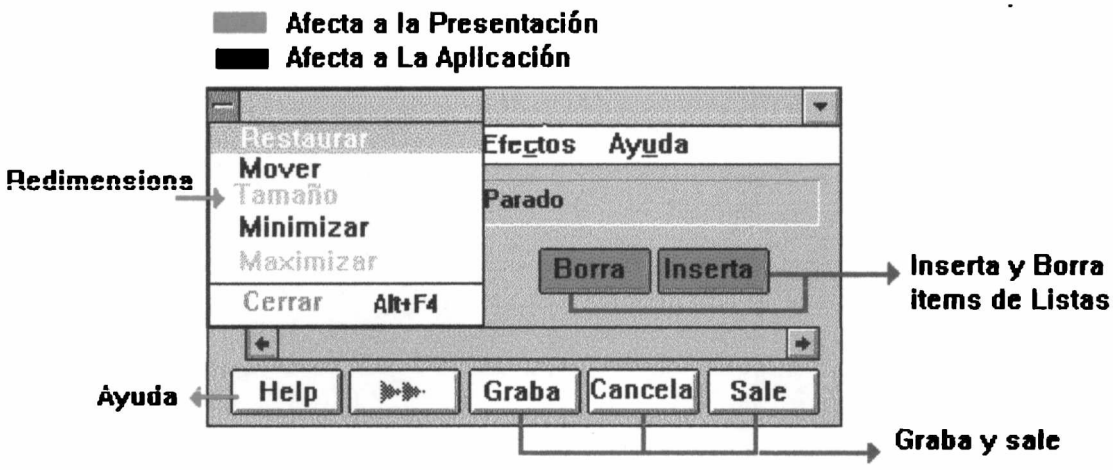
ejecución y proceso **-fase de ejecución-**, y culmina cuando la respuesta del sistema es desplegada al usuario **-fase de respuesta del sistema-.**

Podemos distinguir entre dos tipos de eventos:

Eventos de la interfaz : Son las acciones hechas sobre los objetos de interacción, como por ejemplo , clickear un botón, seleccionar un menú, etc.

Eventos de la aplicación : Son aquellos procedimientos disparados por los eventos de la interfaz, que tienen implicancias sobre la funcionalidad de la aplicación, están relacionados con las funciones o tareas de la aplicación.

Figura 2.4 : Tipos de eventos



Como se dijo la adaptabilidad de una interfaz de usuario, significa que se proveen mecanismos para que el comportamiento final del sistema sea distinto, sin la necesidad de intervención explícita del diseñador de la presentación o del diseñador del sistema. Esto significa que debemos ver que cuestiones del trabajo con un sistema, son las que eran tediosas, en cuanto a su modificación, de forma tal que a partir de la recolección de ciertos datos observables, el sistema pueda evolucionar.

Para esto se necesitan ciertos conocimientos :

- Conocimiento de dominio

Es conocer que es verdadero a cerca de las aplicaciones en general, todos los objetos creados para cada aplicación deben adherirse a las reglas del dominio .

Es la descripción de la estructura y funcionamiento del dominio del sistema adaptativo. Representamos el dominio en términos de las tareas que el soporta, las funciones y conceptos lógicos y las funciones y conceptos físicos, que se requieren para llevar a cabo las funciones lógicas.

- Conocimiento de contexto

Es el conocimiento que es verdadero para un objeto o conjunto de objetos del dominio, pero no es verdadero para todos los objetos del dominio. Es un conjunto de restricciones dentro del dominio. Un contexto puede definirse con distintos criterios:

- Por los *modos de interacción* : una misma aplicación puede ser manipulada de distintas maneras, haciendo uso de diferentes técnicas, como *manipulación directa*, *lenguaje de comandos*, *menúes*, etc.

- Por los *roles* definidos : dentro de una aplicación, pueden existir usuarios con diferentes tareas o funciones, pudiendo tener acceso a diversos tipos de información.

- *Autorizaciones* : está determinado por las Jerarquías de acceso a datos o funciones, que componen la aplicación.

- Conocimiento de Usuario

Son los hechos y características que son verdaderos para un usuario o conjunto de usuarios. Es el conocimiento que tiene el usuario a cerca del contexto en el que él se desempeña y el conocimiento a cerca del manejo del sistema.

Hay dos puntos que debemos profundizar :

a. Adquisición del conocimiento de los usuarios.

La historia de las interacciones del usuario es una fuente rica de conocimiento acerca de sus preferencias, de su conocimiento inicial y de su evolución en el sistema. Debería existir un **recolector de datos** encargado de capturar los **eventos** que ocurren durante las interacciones del usuario con el sistema.

Entendemos por evento, a una acción llevada a cabo por el usuario durante la ejecución de un comando, por medio de un dispositivo físico; como puede ser un mouse, lápiz óptico, teclado, etc. Deberían recolectarse los eventos relevantes para propósitos de customización.

b. Mantenimiento y administración de información acerca del usuario.

Los *recolectores de datos*, como ya expusimos antes, toman los eventos de las interacciones y filtran los que consideran relevantes para el sistema y se utilizan para actualizar el conocimiento del usuario. Este conocimiento almacenado, será requerido en una etapa posterior para llevar a cabo la customización. Nosotras proponemos un *modelo de arquitectura* para *interfaces adaptativas*, en la que detallamos, cada una de las componentes y sus funcionalidades. (Capítulo VI.2).

3

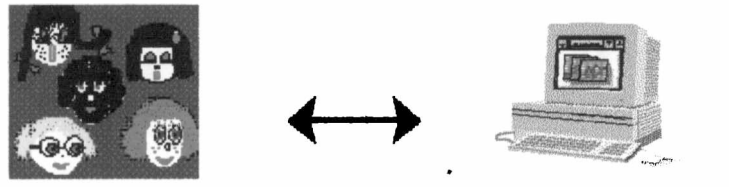
Técnicas de Interacción

I.1 Introducción

Es evidente que diferentes personas tienen distintos estilos cognitivos, y es comprensible que las preferencias particulares puedan variar. No en vano existen múltiples gustos de helados o modelos de autos, por lo que también surgirán distintos estilos de interfaces. Además puede suceder que las preferencias varíen por usuario y por tareas desempeñadas. Es por esto que para cada comunidad de usuario, los diseñadores deben combinar lo mejor de cada estilo de interfaz -al desarrollar las interfaces de usuarios-, de manera que el usuario pueda desarrollar sus intenciones satisfactoriamente.

Por estilo de Interfaz se entiende, las diferentes formas o los diferentes estilos por medio de los cuales los usuarios interactúan con su aplicación, por ejemplo ventanas, menúes, íconos, etc.

Figura 3.1 : Modelo Hombre - Máquina.



III.2 Control de diálogo. Definición de diálogo y módulo de control de diálogo.

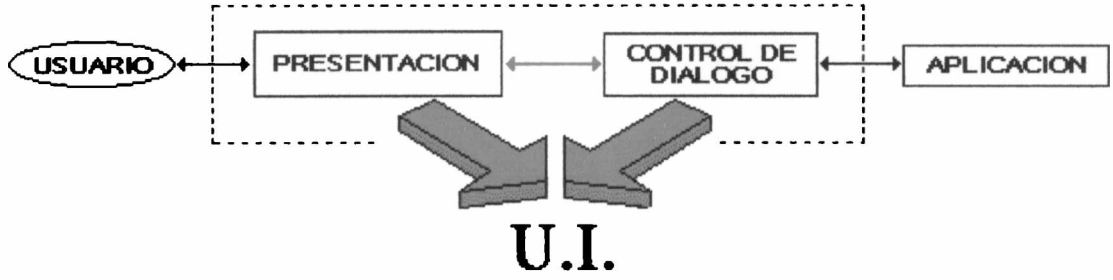
¿ Qué es el Diálogo ?

Es el intercambio de la información entre el usuario y la aplicación. El diálogo, reemplaza la secuencia fija que obligaba a la persona que utilizaba un sistema a cargar información en órdenes predeterminados.

El diálogo gobierna la forma en que la interfaz de usuario es animada, es decir cuando ciertas ventanas son abiertas, cerradas, cuando ciertos mensajes son desplegados o removidos, cuando un ícono está disponible o no. etcétera, pero da la posibilidad de realizar una misma operación por distintas vías.

El diálogo habilita ciertas secuencias de acciones para el usuario -clickear el mouse, seleccionar un ítem de algún menú, etc.- y condiciona la aplicación -despliegue de una cierta información o mensaje de error o advertencia-.

Dentro del Modelo de Aplicación Interactiva mencionamos :



El Diálogo provee un mecanismo de intercambio de información entre el usuario y la aplicación (para ingreso de datos, muestra de resultados intermedios y finales, especificar órdenes y evidenciar estados).

Como resultado de la interacción de dos agentes, el diálogo puede tener distintas características, según quien lo domine y como lo haga. Existen varios estilos de diálogo o formas de interacción visual, pero las agrupamos según quien controla el desarrollo del diálogo, en dos categorías, según gobierne la computadora o el usuario.

A. ESTILOS EN LOS QUE LA COMPUTADORA GOBIERNA EL DIALOGO

En estos casos la aplicación va interrogando o guiando el accionar del usuario en varios aspectos. En estos no significa que el usuario no participe, sino que éste tenga una postura pasiva en la guía del diálogo, sin dejar de ser el centro, pues de él se esperan las respuestas.

A.1 Feedback Dinámico

Se entiende por Feedback Dinámico a la modificación del aspecto de la pantalla reflejado, ya sea, como respuesta que hace la interfaz a alguna acción llevada a cabo por el usuario o bien algún cambio de estado del sistema, que tendrá consecuencias para el usuario. Estas respuestas pueden manifestarse como cambios de objetos en la pantalla, cambio de color -brillo, video inverso, nuevos objetos, etc.- o algún tipo de animación -movimiento de objetos-, que le permite al usuario ir llevando el hilo del diálogo.

Algunos ejemplos de Feedback son:

- Después de varios intentos infructuosos realizados por el usuario o permanencia prolongada en una ventana de selección, podría aparecer un help (sin solicitud explícita del Usuario), que lo oriente en sus acciones.

- Si un usuario esta desarrollando tareas que demandan un tiempo prolongado el feedback podría ir reflejando los distintos pasos que la componen.

- Durante la utilización de algún recurso podría surgir alguna falla, como falta de memoria, disco lleno o rotura del mismo, etc.

A.2 Sistema de Ventanas

El problema general de la mayoría de los usuarios, es la necesidad de consulta de múltiples fuentes de información en forma rápida, interrumpiendo mínimamente la concentración en sus tareas. El uso de sistemas de ventanas, va creciendo día a día, ello es así por cuanto:

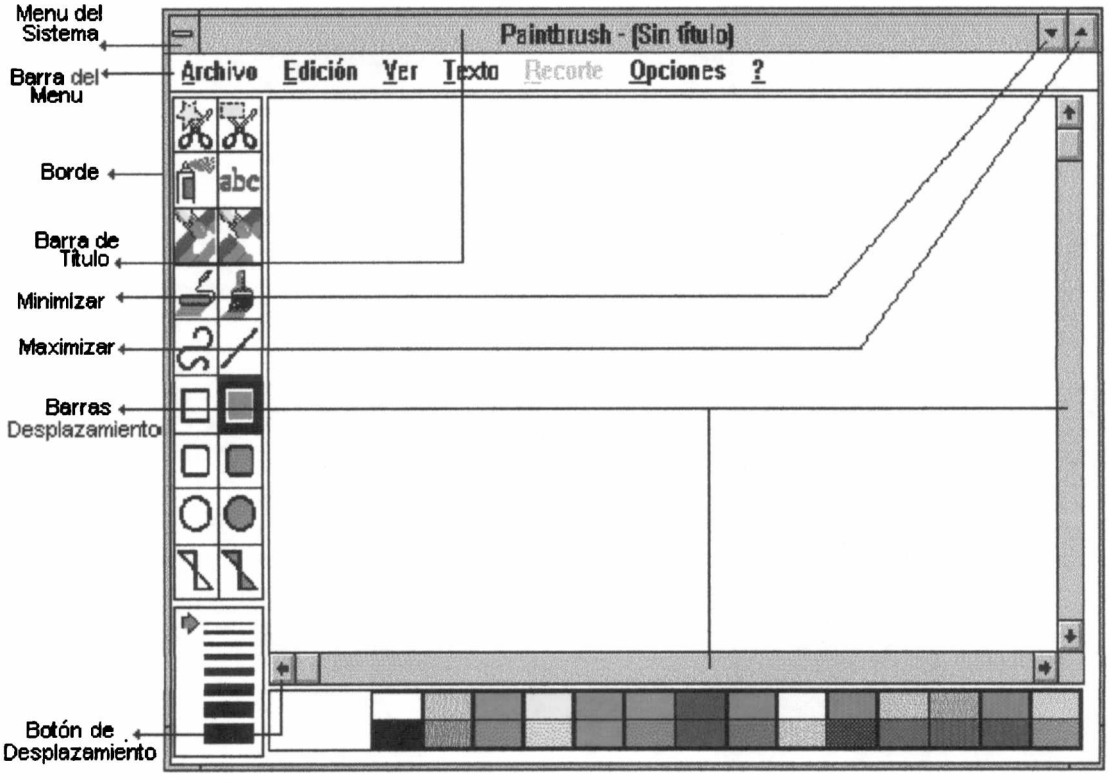
- Permite dividir la pantalla física en múltiples partes, y en cada una desplegar información relativa a distintas tareas que se están desarrollando concurrentemente.

- Facilita la manipulación de documentos externos, o textos y gráficos visualizándolos parcial o totalmente en forma simultánea. Para ello la pantalla se divide en ventanas a las que se le asignan distintas aplicaciones, cada aplicación se limita a escribir en su área sin interferir en las restantes, lo que nos brinda un amplio aprovechamiento del espacio en pantalla.

- Permite desplegarle al usuario ventanas de helps, aclarando algún tópico particular o ventanas con mensajes de error, sin que la información se mezcle, evitando confusiones.

Definición de Ventana : ***" es un área rectangular que contiene una aplicación de software o un archivo de documento. Las ventanas pueden ser abiertas, cerradas, redimensionadas y movidas. Se pueden abrir varias ventanas simultáneamente (sobre el DeskTop), y pueden ser minimizadas (llevarlas a una forma icónica) y volverlas a su tamaño normal."***
MS Windows 3.0 'User's Guide'

Figura 3.3 : Componentes de una Ventana



Componentes de una Ventana -estilo Windows-.

- Menú del Sistema : Permite las operaciones sobre ventana -restaurar, mover, cambiar tamaño, minimizar, maximizar y cerrar-.
- Barra del Menú : Contiene un menú horizontal con las opciones principales de la aplicación, de las que generalmente cuelgan los menúes descolgables.
- Borde : Conservan el espacio donde se muestra el contenido de la ventana. Pueden tener sombras y ser de distintos grosores y colores.
- Barra de Título : La mayoría de las ventanas poseen un título identificatorio al tope de la ventana o rara vez en la base -centrado, a la derecha o a la izquierda-. Es útil para ubicarla cuando hay varias ventanas abiertas en cascada. La ventana activa entre todas las abiertas, mantiene un color que las distingue de las restantes.
- Icono de Minimizar : Lleva a la ventana a su forma icónica.
- Icono de Maximizar : Lleva a la ventana al máximo tamaño permitido.
- Barras de Desplazamiento : Debido a que una ventana puede ser pequeña comparada con el tamaño del contenido que debe mostrar, se utilizan estas barras para poder desplegar en forma parcial dicho contenido. Las operaciones básicas sobre los scroll bar son : mover arriba y abajo -para la barra vertical- y mover a derecha y a izquierda -para la barra horizontal-.
- Botón de Desplazamiento : permite el desplazamiento del contenido de a bloques, clickeando con el mouse se puede arrastrar el botón hasta el lugar deseado.

Operaciones Disponibles sobre las Ventanas

- Apertura : una ventana puede ser abierta a partir de un ícono, de la selección de un Menú, por el tipeo de un comando, o un doble clickeo de Mouse.
- Ubicación y tamaño : una clave importante en la utilidad de los sistemas de ventanas es la elección de donde se abre la ventana. Algunos sistemas siempre abren las ventanas en el mismo lugar que eligió el diseñador. Otros sistemas permiten mover y cambiar el tamaño, esta aproximación es mejor pues satisface las necesidades del usuario. Otras estrategias usan una posición computada en función de las ventanas que están actualmente abiertas.

- Cierre : la ventana puede tener un pequeño ícono para cerrarla y llevarla a su forma icónica. Puede haber un menú de control especial con una acción de cierre o la ventana puede contener un botón de : 'OK' o 'CANCEL', que cerrarán la ventana, aceptando lo hecho o cancelando o directamente un botón de 'CERRAR' o 'SALIR'.
- Activación : cuando existen ventanas superpuestas se necesita de un mecanismo que active la ventana deseada que está total o parcialmente cubierta por otra ventana. Hay diferentes formas para hacerlo : tipear un comando desde el teclado, seleccionarla de una lista de ventanas abiertas, clickear en cualquier parte de la ventana -o una parte restringida tal como el título-, o simplemente moviendo el cursor dentro de la ventana. Los cambios que produce en las ventanas la activación son cambios en el color o grosor del borde, cambio del color del título, etc. Solo una ventana tiene esas características, y significa que el usuario puede interactuar con la aplicación asociada a la misma.

A.3 Animación

La animación nos permite darle movimiento a los objetos sobre la pantalla. Esta técnica es desarrollada con diversos propósitos:

- Permite la representación de sistemas dinámicos, simulando por medios de íconos y movimientos -tiempos de arribos, tiempos de salidas, etc.-, sistemas del mundo real -Bancos, Hospitales, Cadenas de Producción, etc.-, donde se desea conocer previo a la implementación como funcionarán las Colas, Servers, etc.

- Ha tenido creciente éxito en el Mercado comercial, conjuntamente con la representación 3D, se están esforzando por lograr los mejores efectos -para publicidad, diseño, evaluación de piezas mecánicas a ser testeadas- y se ha logrado una increíble representación del mundo real.

- También es de utilidad para sistemas educativos, donde los movimientos pueden dar claridad a las enseñanzas -movimiento de agujas de reloj, acciones de personas, animales para representar verbos de algún idioma, etc.-.

A.4 Sistemas de Menúes

Esta técnica surge para disminuirle al usuario el esfuerzo mental y el entrenamiento de complejas secuencias de comandos. Cuando los ítems del menú se escriben con una terminología familiar, los usuarios pueden seleccionar fácilmente un ítem y pueden indicar su selección con la presión de una o dos teclas ó usar un mouse. Este estilo de interacción reduce la posibilidad de errores y estructura las tareas para los usuarios novatos o intermedios.

Los menúes se diferencian con una interfaz de lenguaje de comandos, en dos aspectos importantes :

- Usando lenguaje de comandos todo el comando debe ser correctamente tipeado mientras que con menúes, sólo se requiere de la presión de una tecla ó el click de un mouse para seleccionar.

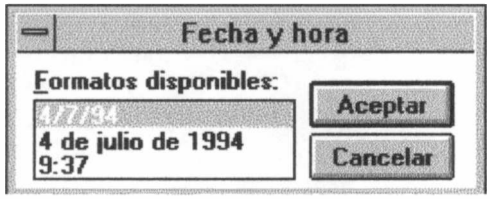
- Otro aspecto importante es que los comandos deben ser memorizados, mientras que con los menúes la selección está visible y una vez efectuada la elección, esta cambia su aspecto -color, forma, etc.-, indicando que se seleccionó.

Un menú es efectivo cuando los usuarios tienen poco entrenamiento, usan el sistema en forma intermitente, no están familiarizados con la terminología y necesitan ayuda para estructurar su proceso de decisión.

El objetivo primario para los diseñadores de menúes es crear una organización semántica sensible, comprensible y conveniente para las tareas del Usuario. La descomposición jerárquica es mas natural y comprensible para la gente, pues es mas fácil estudiar un Libro que esta descompuesto en capítulos, o seguir un programa que esta descompuesto en módulos, aprender el Reino Animal clasificándolo en especies, etc.; es decir al crear el menú debemos recordar que todo ítem pertenece a una categoría simple. Desafortunadamente en algunas aplicaciones, un ítem puede ser difícil de clasificar como perteneciente a una categoría, por lo que es muy probable que existirán usuario que se confundirán o perderán tiempo, buscando por navegación. La tentación de duplicar entradas o crear una red se incrementan, una forma elegante de solucionar esto es tener una estructura de árbol.

Clasificación de los tipos de menú :

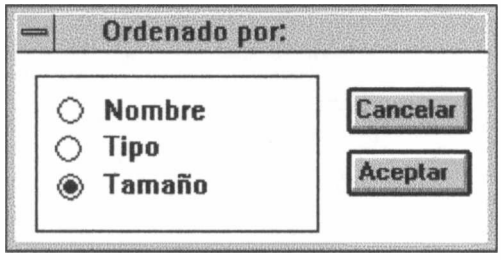
- Menú Binario : Son los mas simple, consisten de sólo 2 opciones, por ejemplo 'Si' 'No' o bien, 'Verdadero' 'Falso' o 'Aceptar' 'Cancelar'.



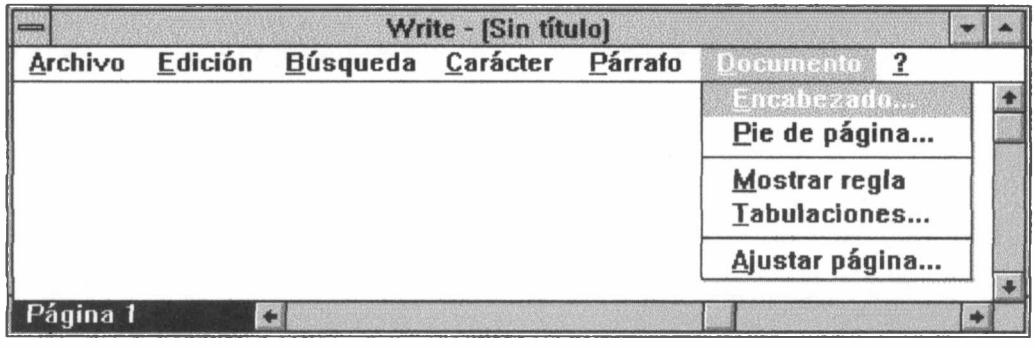
- Menú con múltiples Items y Radio Buttons : Tiene mas de 2 ítems y solamente un ítem puede ser seleccionado al mismo tiempo. En las interfaces gráficas se denominan 'Radio-Buttons' y son preferibles por su claridad. Un ejemplo de pregunta usando línea de comandos :

  ¿ Lo quiere ordenado por nombre, tipo o tamaño ?

  Con Interfaces Gráficas sería :

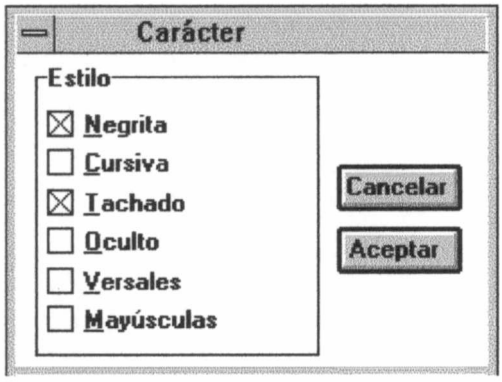


- Menú Pull-Down o Drop-Down : Están siempre disponibles para el usuario en el menubar, una vez seleccionado un ítem del menubar -que sería el título del menú pull-down-, automáticamente se descuelga el menú pull-down.



- Menú Pop Up : son menúes flotantes que aparecen cuando son invocados específicamente por el usuario. Son similares al anterior pero el lugar donde son desplegados es, justamente dónde se los invoca en vez de colgarse del menubar. Proveen una eficiente forma de acceder a los comandos, eliminan la necesidad de navegar por la barra de menú. Generalmente están asociado al botón derecho del mouse.

- Menú con Múltiples selecciones o Check Boxes : tienen la capacidad de poder seleccionarse mas de un ítem al mismo tiempo, por ejemplo elegir varias características para un texto.



A.5 Diálogos Basados en Llenado de Formularios

Acabamos de ver que el menú de selección es efectivo para la elección de uno o más -en el caso de múltiples selecciones- ítems de entre una lista, pero algunas tareas serían muy difícil poder desarrollarlas solamente con un menú. Si los datos de entrada son nombres, apellidos, direcciones, y otros; -lo que podrían ser fichas de personas para distintos fines- el tipeo por medio del teclado se vuelve muy atractivo. Al usuario se le presentan en la pantalla esquemas con forma similar a formularios, donde toda la información complementaria a la que el ingresa -en espacios para llenado-, puede ser vista por él sintiendo que su llenado está siendo correcto.

Los formularios deben reunir ciertas características para lograr efectividad :

- Describir las tareas de los usuarios con terminología similar, tratar de ser breves y expresivos, pudiéndose encontrar **'Tipee su dirección :_____ o Dirección:_____'.**
- Realizar agrupamiento lógico y secuencia de campos, lo primero puede hacerse poniendo líneas o más blancos entre grupos y lo segundo por ejemplo pedir en orden : **dirección, ciudad, provincia, país.**
- Tener una corrección de errores para los campos de entrada : es decir, detectar entradas incorrectas, avisar y permitir la corrección de tales errores.
- Mensajes claros para valores inaceptables : los mensajes pueden desplegar una lista de valores aceptables, por ejemplo, si el código postal es **A458** avisarle que los códigos postales en la Argentina deben tener 4 dígitos.

Marcar claramente los campos opcionales : la palabra 'OPCIONAL' debe estar visible u otra manera de que le indique al usuario que ese campo puede dejarlo en blanco.

Entre otras características, los campos pueden ser codificados para guiar la tarea de carga de los usuarios y pueden agregarse ciertos objetos de interacción, como grupo de botones, de los cuales se pueda optar por uno, para la aceleración de dicha tarea.

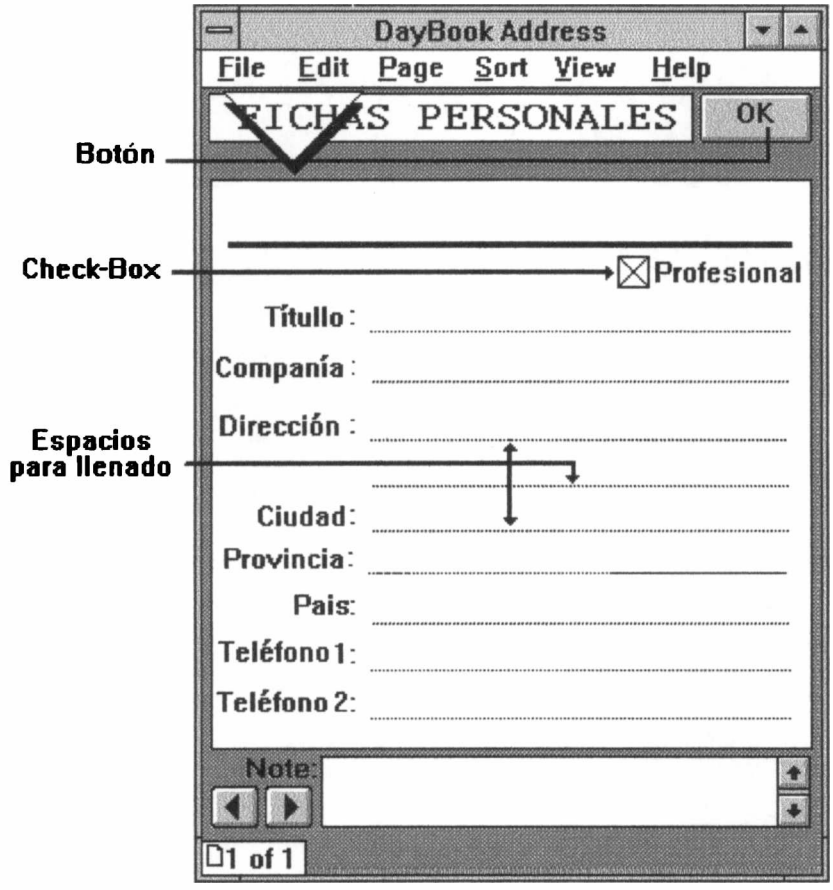
Por ejemplo, los campos codificados podrían ser:

Teléfono : _ _ _-_ _ _ _ _ _ _ _

Número de CUIT : _ _ - _ _ _ _ _ _ _ _- _

Hora : _ _ :_ _ : _ _ (HH:MM:SS)

Fecha : _ _ / _ _ / _ (DD/MM/AA) o (20/01/68 Indica 20 de Enero de 1968)



Los objetos de Interacción pueden ser utilizados en muchas situaciones, en el ejemplo vemos un Check-Box cuando solo hay dos valores posibles (Verdadero o Falso) o podría utilizarse una List-Box con los Países, la cual el usuario pueda consultar y seleccionar.

B. ESTILOS EN LOS QUE EL USUARIO GOBIERNA EL DIALOGO

En este grupo de técnicas el usuario tiene el control del diálogo, el es quien guía la comunicación y quien maneja todos los objetos.

B.1 Manipulación Directa

Varios autores han intentado describir las principales componentes de manipulación Directa. Ted Nelson (1980), percibió que los usuarios se excitan cuando la Interfaz es construida, por lo que él denomina *principios de virtualidad* -una representación de la realidad que puede ser manipulada-. Rutkowski (1982) expone un concepto similar en su principio de transparencia : 'El

usuario es capaz de aplicar su intelecto directamente a sus tareas, y la herramienta en sí, parece desaparecer'. Teheni (1990) encontró que el feedback en los diseños de manipulación directa fueron muy efectivos, ya que redujeron los errores de los usuarios. El Lenguaje LOGO de Papert (1980), crea un micromundo matemático en el cual son visibles los principios de geometría. Basado en la Teoría de Jean Piaget , ofrece a los niños la posibilidad de crear líneas, rotar, caminar con una tortuga sobre la pantalla.

Así, podrían enumerarse un sin fin de ejemplos, veamos cuales son las características más importantes y cuales las desventajas de la Manipulación Directa.

Permite al usuario operar con los objetos que se encuentran disponibles en la pantalla, que deben ser aquellos que resulten de interés en el contexto que se está desarrollando. Los sistemas que utilizan manipulación directa deben elegir inteligentemente y con sentido común las representaciones gráficas de manera que identifiquen claramente a los objetos, para que el usuario pueda comprenderlos y asociarlo con el objeto representado sin demasiado esfuerzo mental.

Las virtudes de la utilización de esta técnica de interacción :

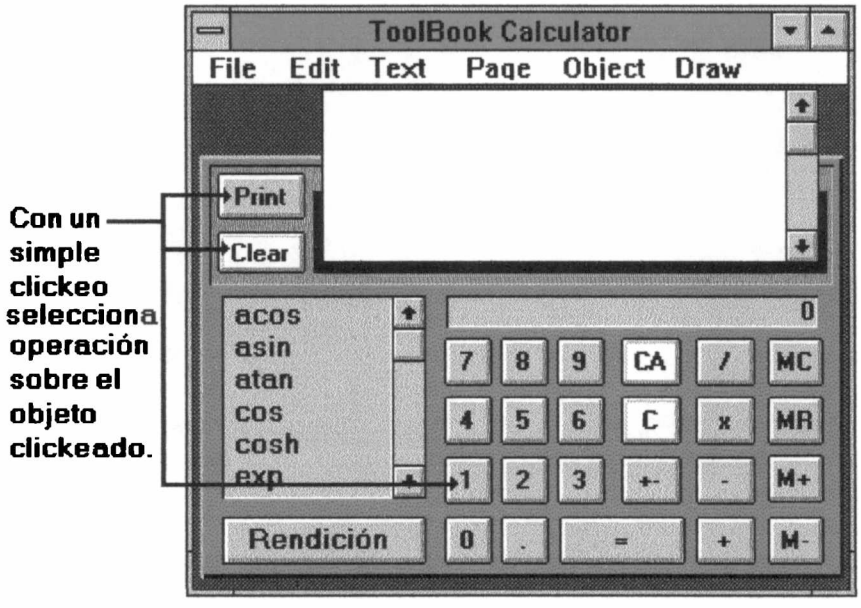
- Los usuarios novatos, pueden aprender rápidamente la funcionalidad de la aplicación, por ejemplo con una demostración de algún usuario más experimentado.
- Los expertos pueden trabajar rápidamente.
- Los errores son menos frecuentes que en otras técnicas de interacción.
- Los usuarios pueden ver inmediatamente los resultados de sus tareas pudiendo deshacerlas si no son las deseadas, con un simple clickeo del mouse.

Las desventajas de la manipulación directa son :

- Los usuarios deben aprender el significado de las componentes de representación visual. Un ícono puede ser significante para un diseñador, pero al usuario puede requerirle mas tiempo de aprendizaje que una palabra. Una manera de solucionar este problema es usar una combinación de gráfico con texto aclarando el ícono.

- Los íconos pueden ser mal interpretados, los usuarios pueden creer rápidamente que interpretaron el significado (por analogía con otro software o propia intuición) y sin embargo llegar a conclusiones incorrectas.
- Para los operadores experimentados, el uso de un mouse puede hacer lenta la entrada de los datos, por lo que les es preferible el tipeo con teclado.
- El manejo de mouse requiere entrenamiento y personas con poca motivación pueden frustrarse rápidamente.

Existen muchos ejemplos, el manejo de una calculadora con uso de manipulación directa puede hacer sentirle al usuario que está operando una verdaderamente, o la simulación de la conducción de un automóvil. La escena es completamente visible ya que en la pantalla se le desplegaría la misma visión que el conductor tiene en su parabrisas, y la performance de las acciones, tales como frenar o doblar se reflejan claramente. Si el conductor desea doblar a la izquierda, simplemente rota el volante, la respuesta es inmediata, pues la escena cambia. Imaginemos usando comandos para esto, primero deberíamos poner izquierda 30 grados y otro comando para que muestre la escena.



B.2 Lenguaje de Comandos

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Utilizando Lenguaje de Comandos, los usuarios tipean un comando y esperan que sucede, si el resultado es el deseado, entonces siguen con otro comando, sino adoptan otra estrategia (otro comando que crean apropiado).

Para poder entablar un diálogo Humano-Computadora, con esta técnica de Interacción, el usuario debe recordar los comandos y la sintaxis que muchas veces es requerida, lo que implica que debe esforzarse mentalmente para el uso de la Interfaz.

Por ejemplo, los comandos de UNIX, no resultan fáciles de tipear y menos aún de memorizar. Si deseamos tan sólo saber cuantas líneas del archivo 'Fuentes' contienen la palabra 'Begin', deberíamos tipear :

```
grep begin Fuentes | wc -l
```

o si queremos iniciar una sesión de PC desde UNIX, debemos poner:

```
pcsim -A 3 -C / PathDos
```

Debido a estas importantes exigencias, han surgido algunas estrategias para organizar los comandos. Podemos citar:

- Una lista de comandos simples.

Cada comando es elegido para ejecutar una tarea simple, con lo que el número de comandos es igual al de tareas. Así, con un pequeño número de tareas, esta aproximación puede producir un sistema simple de aprender y usar. Siguiendo con UNIX, podemos mencionar su editor VI, quien ofrece una gran cantidad de comandos con pocas teclas o combinaciones de las mismas para ejecutar algunas tareas.

H	va a la 1º posición de una línea.
L	va a la última posición de una línea.
Ctr-P	previa línea en la misma columna.
$	final de la línea.

- Comandos mas argumentos.

Cada comando es acompañado por uno o más argumentos. Los comandos pueden ser separados de los argumentos por blanco o algún otro delimitador, al igual que los argumentos entre si.

Por ejemplo:

COPY FILEA , FILEB

DELETE FILEA

- Comandos mas argumentos y opciones.

Los comandos pueden tener opciones para indicar casos especiales. Siguiendo con UNIX tenemos:

Para agregar un archivo al final de otro:

CAT FILEA >> FILEB

Para compilar un programa fuente llamado 'MiProg' y dejarlo en el ejecutable por default:

CC PiProg.C

Para borrar los archivos que empiezan con un asterisco:

find $HOME -name '#*' - exec c rm { }\ ;

- Utilización de abreviaturas

Los nombres de los comandos, deben ser significantes, para que los usuarios puedan aprenderlos, resolver sus problemas y retenerlos. Los nombres abreviados que reemplazarán al comando real, deben elegirse con algún criterio y mantenerlo, para cada uno de los comandos que abrevia.

Por ejemplo:

COPY CPY

DELETE DEL.

Con la experiencia y frecuencia de uso, las abreviaturas se vuelven atractivas para los usuario y necesarias para lograr rapidez en sus tareas.

III. 3 Características customizables de los estilos de interacción.

Para determinar cual técnica de interacción es la más adecuada se debe conocer la aplicación, por lo tanto veremos como podrían llegar a customizarse cada una de ellas.

I. FEEDBACK DINÁMICO

El Feedback puede customizarse con un seteo hecho por el usuario, por ejemplo podría no gustarle el *video inverso*, molestarle *un beep* que representan el feedback, habilitar o deshabilitar la *aparición automática de un help* -cuando se lo detecta desorientado-. Es viable entonces que cada usuario tenga en su modelo, sus tendencias. No cabe en esta técnica la customización automática inferida de gustos, la participación del usuario es indispensable, él toma la iniciativa.

II. SISTEMAS DE VENTANAS

La customización del usuario en esta técnica puede ser hecha con intervención del usuario o sin ella :

II.1 Sin intervención directa del usuario : debido a la existencia de los distintos niveles de usuario, lo que implica distintos conocimientos acerca del sistema. Nuestra intensión es variar el texto desplegado en la ventana o HELP, para proveer al usuario una explicación con léxico apropiado y una profundidad en dicha explicación que satisfaga sus requerimientos.

II.2 Con intervención del usuario: los textos de los helps pueden editarse, es decir permitirle al usuario la modificación a su propio gusto, en helps o mensajes de error. En cuanto a la partición de la pantalla en ventanas, se le podía permitir al usuario el rediseño de las ventanas a su antojo en lo referente a ubicación, color, tamaño, etc.

III. ANIMACIÓN

Debido al dominio de aplicación de esta técnica: Simulación, Presentaciones, Educación, no es posible customizarla.

La customización en el área educacional podría hacese guardando el desempeño del alumno y el último tema utilizado, luego puede utilizarse su registro para determinar el nivel del curso y preferencias detectadas y el tópico como punto de partida de próximas sesiones.

IV. SISTEMAS DE MENÚES

Al igual que en la técnica de ventanas, ésta tiene dos posibilidades:

IV.1 Sin intervención del usuario: La existencia de Menúes extremadamente cargados, no es ágil y puede resultar irritante especialmente para aquellos usuarios expertos. Se le podría permitir al usuario la creación de short-cuts para las opciones que el desee (las mas frecuentemente usadas), de manera de no tener que navegar por todo el menú para llegar a la opción deseada.

IV.2 Con intervención del usuario: uno de los problemas que presenta esta técnica es que los nombres de los procedimientos -opciones del menú-, son generalmente elegidos por compatibilidad con la aplicación o gustos del diseñador y no necesariamente para beneficios del usuario. Para resolver este inconveniente podemos permitirle al usuario que cambie el nombre a su gusto, este nombre externo estará en el modelo para cada usuario, por lo que distintos nombres externos se corresponderán al mismo nombre interno de dicho procedimiento -el original-.

También se le puede permitir al usuario, armarse una rama alternativa de selecciones donde él decide que acciones corresponden a cada nivel del árbol de decisiones.

V. MANIPULACIÓN DIRECTA

Generalmente cuando se piensa en manipulación directa se asocian íconos que deben ser elegidos inteligentemente por el diseñador de la interfaz. La customización podría hacerse pemitiéndole al usuario el cambio de ciertos íconos por otros, o la inserción de nuevos íconos; esto último puede ser peligroso pues el usuario puede no estar capacitado para tal tarea.

VI. LENGUAJE DE COMANDOS

La customización en esta técnica, podría hacerse permitiéndole al usuario el cambio de ciertos comandos por notaciones que le resulten más claras, SINÓNIMOS, o por partes del verdadero nombre, ABREVIATURAS, que es una de las estrategias mencionadas.

¿ es conveniente el uso de esta estrategia ? ¿ para qué tipos de usuario ?

Los usuarios novatos, pueden preferir usar el nombre entero de un comando, porque creen que tendrán más éxito en los resultados -Landover 1983-. En un experimento con usuarios novatos, a un grupo se les pidió el uso de los nombres completos de los comandos antes de aprender las abreviaturas, estos cometieron menos errores con las abreviaturas que aquellos que se les enseño desde un principio con las abreviaturas y que aquellos que podían crear sus propias abreviaturas -Grudin y Barnard 1985-.

Ellos sostuvieron que los novatos preferían tipear el nombre completo, tal como BROWSE y REPLACE en vez de BRWS y RPLC. Después de 5 a 7 veces de usar los nombres completos, la confianza se incrementaba e intentaban usar la abreviación. Por lo tanto, esta es una buena estrategia para usuarios experimentados, quienes frecuentemente buscan rapidez en sus tareas.

Importa destacar que las técnicas referenciadas anteriormente, no deben ser utilizadas necesariamente en forma individual, permiten mayor alcance y se obtienen mejores resultados si las utilizamos en forma conjunta.

Podría tenerse una pantalla con *sistema de ventanas*, en cada ventana podría existir su *sistema de menúes* con los comandos de la aplicación y un *feedback* con video inverso podría indicar cual es la ventana activa.

4

Gerenciador de Interfaces de Usuarios

IV.1 Introducción. Componentes de un gerenciador de interfaces.

La construcción de interfaces de usuarios, para la mayoría de las aplicaciones, consume gran cantidad de tiempo y es dificultosa, más aún cuando el sistema de interfaz de usuario debe dinámicamente crear 'Displays' integrando el uso de varios modos de interfaces y cuando esa interfaz debe captar los cambios del usuario y **adaptarse dinámicamente a ellos.**

Los conceptos fundamentales defendidos en esta tesis son:

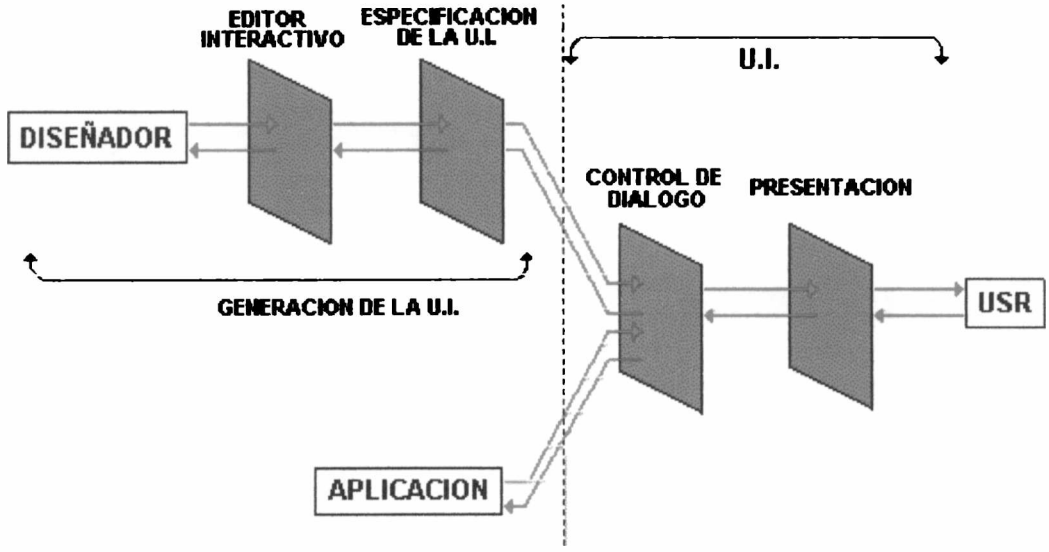
- La interfaz del usuario puede ser **customizada**, es decir adaptarse a sus requerimientos y no que el usuario se adapte a la interfaz.
- La adaptación puede ser hecha **automáticamente**, teniendo en cuenta la cognición y evolución del usuario.

Como ya expusimos, una interfaz de usuario es una herramienta que permite definir secuencias de diálogo interactivo y manejar la interacción física entre la aplicación y el usuario final. Nuestro objetivo es que las interfaces generadas, por medio del gerenciador propuesto, cumpla ciertos puntos :

1. Que el uso de la aplicación sea totalmente natural al usuario, rápido aprendizaje de la utilización y con la mínima cantidad de errores posibles.
2. Que le permita el aprovechamiento máximo del software sin grandes esfuerzos mentales.
3. Que presente distintos estilos de acuerdo a los distintos niveles cognitivos y/o permisivos de los usuarios.
4. Que evolucione de la mano del usuario.

Hay variadas herramientas que me permiten manipular interfaces de usuario, discutidas en el capitulo II, elegimos la UIMS, por considerarla la mas conveniente para un modelo de aplicación interactiva. Esta herramienta, no solamente me permite **crear** interfaz de usuario, sino que es una excelente herramienta para prototipación, además me permite **evaluar y mantener** la interfaz de usuario.

Figura 4.1: Modelo de UIMS (Diseñador - Usuario - Aplicación).



- **Presentación :** es la apariencia externa de la interfaz, es como ve el usuario los objetos de la aplicación.
- **Control de diálogo :** define la estructura del diálogo entre el usuario y la aplicación.
- **Editor interactivo :** es la herramienta utilizada por el diseñador de la interfaz, para construir la apariencia y definir formas de interacción con la aplicación.
- **Especificación de la interfaz de usuario :** es el software subyacente de la interfaz de usuario, que en algunos casos es directamente generado por el editor interactivo.

Un generador de interfaces de usuario automáticamente produce una interfaz de usuario, desde una especificación. Las componentes básicas de una interfaz de usuario, son: una librería de objetos de interacción, un kernel en tiempo de corrida que administra y supervisa aspectos del a interacción -control de diálogo-, una herramienta de especificación.

Una interfaz de usuario incluye una descripción de la presentación, tan bien como una expresión del control del diálogo. Una herramienta de especificación de la presentación asiste al diseñador del soft de la interfaz de usuario, en la construcción de Displays estáticos desde una librería de objetos de interacción, y definiendo enganches entre los objetos del display y el programa cliente. Una herramienta de especificación del control del diálogo, permite al implementador de la interfaz de usuario, definir como serán manejados los objetos de interacción. Hay un tercer tipo de herramienta de especificación, que usa descripción semántica de alto nivel de la aplicación para generar automáticamente la presentación y el control del diálogo.

IV.2 Servicios de las UIMS

Crear una interfaz de usuario requiere diseño y desarrollo de la interfaz de usuario y del software subyacente. Una UIMS es una herramienta o conjunto de herramientas integradas, que soportan algunas facetas de esas tareas. El objetivo de una UIMS es, facilitar el diseño, construcción, evaluación y mantenimiento una interfaz de usuario.

IV.2.1 Tipos de servicios

Los servicios de una UIMS, pueden ser clasificados, de acuerdo a la función que brindan en servicios de : diseño, construcción, evaluación y/o mantenimiento.

IV.2.1.1 Servicios de diseño.

Diseñar un sistema interactivo involucra múltiples pasos:

- Desarrollar análisis de tarea.
- Definir los objetos de cómputos -objetos de interacción y objetos de la aplicación- y su correspondiente dominio de la tarea.
- Definir la apariencia y el comportamiento de la interfaz de usuario.

Cada paso puede ser facilitado por herramientas. En particular, la definición del análisis de las tareas, requiere notaciones conceptuales tales como TAG -Gramática de acción de las tareas-, GOMS -Objetivo, Operador, selección del Modelo-, CLG -Gramática de Lenguaje de Comandos-. Estos permiten al diseñador describir las tareas en varios niveles de abstracción, desde un nivel semántico a un nivel de interacción léxico. La definición de los objetos de cómputos, puede ser soportada por herramientas de especificación de ingeniería de soft. La definición de la interfaz de usuario es soportada por herramientas de prototipación rápida, estas herramientas permiten especificar a la interfaz de usuario gráficamente y puede incluir una simulación.

CUIMS debido a la existencia de los modelos de usuarios incorpora nuevos pasos en el diseño :

- Definir cuantos **modelos de usuarios** desee, para el dominio de aplicaciones, con el criterio que decida el diseñador, pueden ser:

- Como propuso SHNEIDERMAN [1987], organizar a la comunidad de usuarios en tres grupos: *novatos*, *intermedios* y *expertos*; o aproximadamente así, es decir, agruparlos por conocimiento.

- *Por niveles de acceso o jerarquías de usuarios*: generados por permisos decididos por el diseñador para cada usuario o grupo de usuarios.

• Definir el **patrón de inferencias** : tales patrones serán de la misma estructura que los de los usuarios, pero con valores predefinidos, para asumir por default cuando recién comienza la interacción, ellos son:

- *Patrón de preferencias* : para cada uno de los atributos, como tipo, longitud, cantidad mínima de veces que debe detectarse una solicitud de cambio para customizar y valor por default.

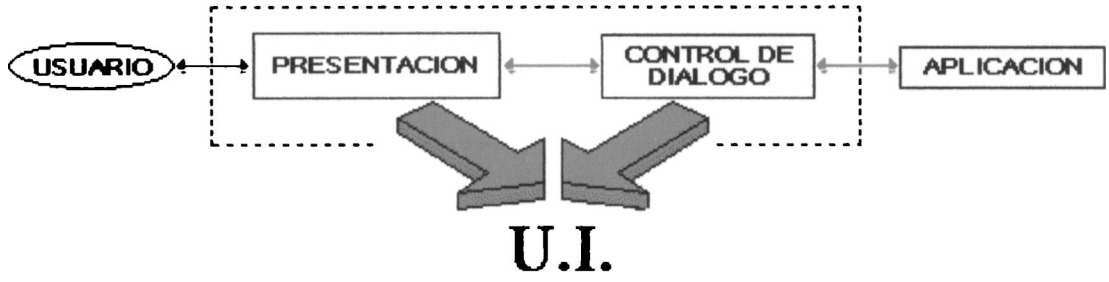
- *Patrón de hábitos* : secuencias previstas de antemano, que pueden customizarse. Cantidad de ocurrencias necesarias para customizar.

- *Patrón de conocimiento* : niveles de conocimiento para cada uno de los tipos de usuario definidos.

CUIMS permite tomar objetos de Interacción de una Librería predefinida de objetos standard, como Radio_Button, Check_Box, List_Box, Menúes y armar las interfaces de usuarios fácilmente, por medio de manipulación directa de dichos objetos. Además de las prestaciones antes mencionadas, permite crear y editar los patrones que conforman el *modelo de usuario*, pudiéndose especificar los atributos que los compondrán y sus valores por default.

IV.2.1.2 Servicios de construcción

Figura 4.5 : Modelo Interactivo



Vemos que en este *modelo interactivo*, la interfaz de usuario tiene dos componentes básicos, la *componente de presentación* y la *componente de control de diálogo*, por lo tanto el software de la interfaz de usuario, tendrá que crear y manipular estas componentes. Por lo tanto, las herramientas para la construcción de la interfaz de usuario, pueden ser clasificadas de acuerdo a cual de los dos aspectos ellos soportan.

Las **herramientas de presentación,** incluyen librerías de clases de objetos de interacción, editores y herramientas de especificación de presentación.

- Las librerías de objetos de interacción van desde un conjunto de clases listas para usar, tales como ToolKits basadas en XWindows y toolKits Macintosh a sistemas programables con restricciones, como Garnet.

- Los *editores* sirven para la definición y modificación de ítems léxicos de una interfaz de usuario, tales como bitmap, cursor, font y editores de objetos de interacción. Los editores de objetos de interacción, tales como ResEdit de Macintosh son útiles para definir nuevas instancias

- Los *editores* sirven para la definición y modificación de ítems léxicos de una interfaz de usuario, tales como bitmap, cursor, font y editores de objetos de interacción. Los editores de objetos de interacción, tales como ResEdit de Macintosh son útiles para definir nuevas instancias desde un conjunto predefinido de clases de objetos de interacción y para modificar los atributos de los objetos de interacción existentes.

El **control de diálogo** necesario implica la existencia de un kernel en tiempo de corrida. Su expresión puede ser programada a mano o soportada por herramientas de propósitos especiales.

CUIMS provee para el servicio de construcción, lo antes expuesto, haciendo uso de una interfaz de manipulación directa.

IV.2.1.3 Servicios de evaluación

La evaluación puede ser desarrollada en varios pasos, particularmente durante la face de diseño o con un prototipo de una corrida. El diseño de una interfaz de usuario puede ser evaluado con :

- *Herramientas de modelización cognitiva* : le dicen al diseñador a cerca de la utilidad de un diseño propuesto, antes de que este sea construído. Un ejemplo es PUM, el cual usa un modelo del usuario, asi como una representación de las tareas.

- *Herramientas mas tradicionales* desarrolladas en el área de la ciencia de la computación : incluyen loggers, simuladores, y soporte para pruebas. Un registro Loggers acumula las acciones físicas del operador. Un análisis de los datos observados pueden fijar la atención del diseñador observando frecuencia de uso de comandos, error en el uso del teclado y planeamiento efectivo del operador. Tales datos pueden ser usados como inputs para predecir herramientas como las discutidas anteriormente.

CUIMS soporta un testeo de la *presentación* durante su desarrollo.

IV.2.1.4 Servicios de mantenimiento

El mantenimiento para un sistema puede ser categorizado como:

- Corrección de errores.
- Mejoras del sistema sobre la plataforma existente.
- Modificación del sistema para correr sobre una plataforma diferente.

Para facilitar el esfuerzo en la fase de mantenimiento, se debe hacer hincapié en los siguientes niveles de abstracción:

- La separación de lo referente a la interfaz de usuario de aquella aplicación que localiza las modificaciones del nivel de presentación de la interfaz de usuario .

- Hacer explícita la interfaz entre la aplicación y el soft de la interfaz de usuario, simplifica las modificaciones que afectan a ambas componentes.

Hay pocas herramientas de propósitos generales diseñadas para la fase de mantenimiento, debido a que muchos de los detalles del software de la interfaz de usuario dependen de las acciones específicas del operador.

CUIMS permite editar las veces que sea necesario la interfaz de usuario definida, corregirla generarla para testearla, produciéndose un ciclo :

DISEÑO ⇒ CONSTRUCCIÓN ⇒ EVALUACIÓN ⇒ MANTENIMIENTO

IV.3 Herramientas de especificación de las componentes de un gerenciador de interfaces.

*Herramientas de especificación de la **presentación***

Definición: Existen algunas herramientas de especificación de la presentación, que pueden ser usadas para definir displays estáticos. El diseñador del soft de la interfaz de usuario, instancia los objetos de interacción desde un conjunto de clases predefinidas, setea sus atributos, y define enganches con la aplicación, en forma de llamadas a procedimientos.

Una vez que una imagen deseada ha sido diseñada, se genera un código esqueleto, que incluye la inicialización de los objetos de interacción y los esqueletos de las llamadas a procedimientos. El diseñador de soft edita el código de los esqueletos, para definir cuando el procedimiento es llamado e incorporar el código de cada procedimiento en el sistema interactivo final.

Como los editores, las herramientas de especificación de presentación, requieren que el diseñador de soft de la interfaz de usuario trabaje en el nivel léxico de la interacción. Los objetos de interacción producidos con la herramienta de especificación de la presentación, pueden ser enganchados a un programa particular. Con este tipo de herramientas, es posible especificar el nombre de las llamadas a procedimientos que serán invocados cuando un determinado evento ocurra para un objeto de interacción particular. Con algunas herramientas de especificación de presentación el diseñador de soft de la interfaz, debe aprender un lenguaje de propósitos especiales y puede visualizar los efectos del seteo de un atributo únicamente, codificando, compilando y ejecutando el programa. Este es un método de construcción de displays estático. Otras herramientas, son interactivas y le permiten al diseñador de soft describir la presentación a través de manipulación directa y evaluar la visualización de la presentación cuando es construida.

Ventajas y desventajas

Las herramientas de especificación de presentación, son buenas para generar componentes estáticas de una interfaz de usuario, provistas esas componentes, pueden ser descriptas con un conjunto de clases predefinidas de objetos de interacción.

La desventaja, es que son pobres para el manejo del comportamiento de componentes dinámicas.

La definición de objetos de interacción de propósitos especiales y la definición de su comportamiento dinámico, requiere programación en un lenguaje usual. La programación requiere que el desarrollador conozca el toolkit de la interfaz corriente y programe conforme a ella.

Herramientas de especificación del ***control del diálogo***

Definición: Las herramientas de especificación de control de diálogo, proveen al diseñador del soft de la interfaz con un lenguaje de propósitos especiales para especificar la secuencia de reglas para los objetos de interacción. Las secuencias de reglas cubren un número de fenómenos dinámicos, tales como:

- Las condiciones para crear, borrar y modificar las instancias de los objetos de interacción.
- Las condiciones para el mapeo y desmapeo de las pantallas de los objetos de interacción.

Hay dos opciones en la especificación : se especifica todo el diálogo con un lenguaje de propósito especial o se especifica el diálogo con una combinación de programación por demostración y programación por medio del lenguaje de propósitos especiales.

La programación por demostración esta limitada a la especificación de una secuencia muy simple de condiciones entre los objetos de interacción. En este método de programación, el código es producido por demostración.

Ventajas y desventajas

Las ventajas de los lenguajes de propósitos especiales para la especificación del control del diálogo, son :

- Refuerza la separación entre lo natural de las técnicas de presentación y sus relaciones dinámicas. Esta separación mejora el refinamiento interactivo. Hace posible la programación de diferentes comportamientos en tiempo de corrida para las mismas instancias de los objetos de presentación.
- Le permite al sistema run-time, automáticamente determinar y reforzar las relaciones entre los datos de la aplicación y los objetos de interacción y entre los mismos objetos de interacción.

Las desventajas es que aún las simples relaciones tales como secuencias, deben ser textualmente especificadas.

Una característica de la mezcla -por demostración más lenguaje propósitos especiales-, es que el diseñador, puede especificar las relaciones simples, por medio de la manipulación directa y las mas complicadas por demostración.

5

Fundamentos para el desarrollo del Prototipo

V.1 Introducción.

Al investigar el tema **Interfaces Adaptativas,** se nos presentaron dos orientaciones diferentes a seguir :

- **Orientado a la Inteligencia Artificial :** el objetivo final será inferir de las interacciones del usuario y finalizar las tareas por ellos. El desarrollo de esta rama involucra el estudio profundo de Planes y el desarrollo de algoritmos inteligentes que permitan inferir intenciones desde una Base de Conocimiento. Estudiamos algunas tesis relacionadas con este tema, "An explanation-Based Learning Approach to Incremental Planning', de Steve Ankuo Chien, y 'A Knowledge-Based Approach to Automatic Program Analysis' de Jim Qun Ning. Otra de las tesis evaluadas, trabajó para lograr el diagnóstico de una enfermedad. Esto requirió años de recolección de antecedentes, desarrollo de algoritmos complejos, apoyo de expertos en la materia, para detectar solo un tipo de Meningitis.

  Encontramos que se requiere de un esfuerzo inmenso para inferir en casi todos los casos una solución a un solo problema y dirigido a un grupo reducido de usuarios, nuestra expectativa, si bien intentaría hacer cosas por el usuario, y asistirlo en la interacción con el sistema, el objetivo principal era abarcar un marco mas general.

- **Orientado a la Ingeniería del Software** : la otra alternativa era la construcción de una herramienta que permitiera desarrollar **Interfaces Adaptativas**, que podrían ser usadas por toda la comunidad de usuarios y que satisfagan la necesidades de la gran variedad de los mismos. Nos decidimos por esta rama.

V.2 Selección del Lenguaje. Evolución.

La herramienta desarrollada hace uso del estilo de Manipulación Directa, dado que su construcción es de alta complejidad, encontramos en el Paradigma Orientado a Objetos ciertas características que facilitan su construcción, *como el encapsulamiento del comportamiento y los datos en clases y la clasificación en subclases con herencia*.

En este paradigma el elemento mas importante es el **O b j e t o** . Un objeto es un *dato* compuesto por atributos o variables- y un conjunto de *procedimientos* -referenciados como métodos-. Los atributos son encapsulados: no pueden ser modificados por referencia directa. El acceso a un atributo se realiza a través de la invocación de un ***método*** del objeto, esta es la única manera que tienen los objetos de comunicarse.

Una **C l a s e** es una especificación genérica para una colección de objetos similares. Esta define un molde desde el cual se instanciarán los objetos en tiempo de corrida.

En general, las instancias de una clase, comparten los métodos definidos en una clase, pero tiene su propia copia de los valores de sus atributos. Algunos lenguajes orientados a objetos, como por ejemplo SMALLTALK, hacen una distinción entre ***variables de Instancia y variables de Clase***. Las variables de instancias son locales a cada instancia, mientras que las variables de clases, son comunes a todas las instancias de esa clase. Las clases, pueden tener *subclases,* cada una de las cuales, hereda procedimientos y datos desde su padre, y puede agregar comportamiento para lograr una especialización del mismo. Por lo tanto, la ***herencia*** permite la reusabilidad de código y crear taxonomías de clases.

Otra característica atrayente del SMALLTALK es el ***polimorfismo***, esta es otra capacidad de abstracción por la cual dos o mas clases de objetos, responden al mismo mensaje, cada una de su propia forma. Esto significa que un objeto no necesita conocer quien será el receptor del mensaje , sino si el objeto receptor está capacitado para responder a ese mensaje.

Debido a las características que posee el paradigma orientado a objetos y las necesidades de nuestra tesis creímos conveniente desarrollarla en el Lenguaje SMALLTALK for WINDOWS.

Evolución del Smalltalk

El Modelo de Interfaz de usuario del Smalltalk 80, esta basado en el paradigma MVC.

1. MVC : Model-View-Controller : este modelo separa la UI de una aplicación en 3 componentes :

El Model es el dato que va a ser mostrado y/o modificado. Recibe este nombre porque su finalidad es modelizar algún aspecto de la realidad.

La View es la componente que determina como el Model va a ser mostrado o visualizado.

Controlar es la componente que maneja las interacciones y eventos de entrada/salida de los dispositivos físicos.

2. MW : Model Windows

El nuevo Smalltalk for Windows es similar al resto de la familia de los Smalltalk, excepto que el View y el Controller se encuentran en una sola componente, por lo que lo denominamos MV. El motivo de esta modificación es que el manejo de interfaces de usuario en Smalltalk se parezca mas a la de Windows, lo cual si bien es conceptualmente mas restringido, permite interactuar simplemente con otras tareas en Windows y utilizar un estilo mas uniforme y extendido para desarrollar aplicaciones de este estilo.

Las clases básicas como Collection, Magnitude, Stream, etc. son idénticas. Las clases que nos permiten crear interfaces de usuarios, se han modificado. El Displacer y el Pane -o el

Controller y la View-, han sido combinados en subclases de **Windows**, BitBlt es reemplazada por GraphicsTool y sus subclases. Se incorpora otra importante clase llamada **ViewManager,** que explicaremos a continuación.

Clase **ViewManager**

Esta es una clase abstracta que implementa protocolo común para la parte de las aplicaciones de usuario que involucran a la interfaz de usuario. En Smalltalk V las clases que modelan la aplicación -Model-, estaban directamente como subclases de Object, el nuevo Smalltalk las agrupa en ViewManager, esta clase implementa métodos que pueden ser compartidos por todas las aplicaciones que se creen.

ViewManager
- **ClassBrowser**
- **ClassHierarchyBrowser**
- **DiskBrowser**
- **Nuevas Aplicaciones ....**
- **WindowDialog**
  - **AboutDialog**
  - **Nuevas Ventanas de Dialogo ....**

Clases **Windows**

Todas las subClases de Pane en el Smalltalk V, pasaron a estar como subclases de Windows, quedando una nueva Jerarquía:

Window
- **ApplicationWindow**
  - **TopPane**
- **SubPane**
  - **ControlPane**
    - **Button**
    - **Otros Subpane de control estilo Windows como CheckBox, Listas ...**
  - **GraphPane**
    - **AnimationPane . . .**

No hay una distancia entre los objetos que manejan las entradas -Controllers- y los objetos que manejan las salidas -views-, ahora los objetos Windows manejas ambas cosas.

ApplicationWindows y TopPane corresponden a TopPane y TopDispatcher del Smalltalk V. La comunicación entre un subPane y su dueño (Model) ha cambiado. SubPane tiene una lista de eventos que puede notificar a su dueño. Cuando se necesita crear una instancia de algún subpane, deberá hacerlo de la siguiente manera :

```
TextPane new
        owner: self;
        when: #getContents perform: #texto: ;
        when: #getMenu perform: #texto: ;
        when: #save perform: #texto: ;
```

En la clase SubPane se agrupa al SubPane y Distpatcher del Smalltalk V. Todos las subclases de subPane tienen un método de clase llamado supportedEvents, el cual informa los eventos que puede notificar.

Clases **Gráficas**

BitBlt y sus subclases han sido reemplazadas por GraphicsTool y sus subclases. Las clases gráficas en Smalltalk for Windows fueron diseñadas de tal forma que provean facilidades gráficas estilo Microsoft Windows. La clase GraphicsTool esta compuesta de la siguiente manera:

```
GraphicsTool
        TextTool
                Pen
```

La clase Form fue eliminada. La case Bitmap puede ser usada para lograr los mismos efectos. Las aplicaciones que dibujaban sobre Forms ahora pueden hacerlo sobre Bitmaps.

Estas son las principales diferencias entre Smalltalk V y Smalltalk for Windows.

Estas son las principales diferencias entre Smalltalk V y Smalltalk for Windows.

V.3 Bibliografías Estudiadas

Se comenzó con la lectura de diversas tesis relacionadas con el tema de interfaces y en particular de interfaces adaptativas, además de bibliografías sobre desarrollo y diseño de interfaces y herramientas de desarrollo.

La primer tesis investigada fue **'Automated Customization of User Interfaces'** , de Barbara Staudt Lerner. Los tres conceptos fundamentales explorados por ella fueron:

- La interfaz de usuario puede ser customizada **sin intervención de usuario:** investiga una solución para disminuir el gap entre el diseñador y el usuario, proponiendo como solución la customización automática de manera que sean tolerantes a distintos tipos de usuarios. Para lograr interfaces tolerantes se basó en 3 técnicas :

  1. Automatización : es el proceso de desarrollar acciones a espaldas del usuario, sin requerir la intervención explícita del mismo. Su objetivo es automatizar aquellas actividades que un usuario ejecuta regularmente. Existen varios tipos de automatización :

  - Las Dependientes del Dominio de la Aplicación, como por ejemplo la compilación automática de procedimientos modificados.
  - Las Dependientes del Contexto en el cual es usuario está operando, como cambiar el tipo de letra para todas las figuras de un libro que se está escribiendo.
  - Las Dependientes de los Patrones del comportamiento del usuario, como correr una corrección gramatical cada vez que se agrega una página.

  2. Interpretación tolerante de comandos : es el proceso que intenta disminuir las precondiciones de los comandos. Entendiéndose por 'Precondiciones de los Comandos', todas aquellas cosas que deben cumplirse para que se desarrolle una acción. Por medio de un Intérprete de Comandos de programas, se interpretan los comandos del usuario que

no reúnan las precondiciones o que sean ambiguos en el contexto dado. Cuando se recibe una entrada del usuario que no puede interpretarse, el intérprete trata de razonar acerca de lo que intentó hacer el usuario.

El intérprete puede resolver estos problemas por varias formas:

- Comando preferido : aplicaría el comando mas comúnmente usado por ese usuario.
- Corrección de deletreo : tratar de corregir el nombre del comando para que sea valido o buscar el nombre del objeto valida al cual se le aplique ese comando.
- Objeto erróneo : tratar de buscar el objeto adecuado para el cual sea válido aplicar ese comando.

Es claro, que las soluciones que Bárbara Lerner propone, son exclusivamente para una interface de comandos.

3. Helps Activos : es proveer ayuda al usuario a cerca de un concepto o comando, cuando se lo nota confundido, antes de que el usuario lo requiera. En aquellas aplicaciones que no proveen este tipo de helps, el usuario debe pedir explícitamente el help, una desventaja de estos sistemas es que la búsqueda de la información útil se puede volver una tarea tediosa y frustrante para aquellos usuarios novatos que no tengan el conocimiento necesario para navegar exitosamente por el sistema de help.

- La Historia de las interacciones del usuario reflejan sus preferencias y le servirán en un futuro.
- El Concepto fundamental es la evaluación de performance.

La anatomía de la interfaz de usuario que Lerner propone, está compuesta por heurísticas y loggers. Las heurísticas son componentes inteligentes que customizan la interfaz de usuario, razonando a cerca de hábitos, contexto y dominio y además evalúan su performance, guardando los resultados -éxito/fracaso-, luego de aplicar las heurísticas para su posterior uso. Los loggers

Análisis de la Propuesta

- Las heurísticas de **Automatización** típicamente automatizan acciones que son fáciles de desarrollar para el usuario. Antes de aplicar una heurística ella hace un análisis de costo/beneficio, lo que da como resultado a menudo un beneficio bajo, ya que las acciones que se pueden customizar automáticamente pueden ser desarrolladas por el usuario sin mayores inconvenientes.
- En cuanto a la **Interpretación Tolerante de Comandos**, vimos que propuso 3 heurísticas. Esto es aplicable solamente a una Interfaz de Lenguaje de Comando y puede llegar a ser muy perjudicial pues no pide intervención del usuario. Supongamos este ejemplo:

  Un usuario borra una cantidad de archivos en un directorio usando DOS:

  DEL Archi1.TXT , DEL Archi2.TXT, DEL Archi3.TXT

  Si después desea ver todo el contenido del directorio y tipea erróneamente:

  DER *.*

  Se encuentran dos comando a aplicar: DEL o DIR, con el criterio elegido por ella se aplicaría el comando mas recientemente usado sin consultarle al usuario en caso de tener mas de una opción, aplicaría :

  DEL *.* perdiéndole toda la información.
- En cuanto a **HELPS Activos**, los activa cuando detecta que el usuario aparenta estar confundido, esta heurística es parecida a nuestra propuesta.
- El otro punto criticable, es que la solución propuesta es aplicable a un sistema específico, que fue un editor de textos; donde las tareas desarrolladas en ese ambiente son reducidas; ella es demasiado específica y su propuesta es imposible de aplicar a un marco general.

La segunda Tesis investigada fue **'Knowledge - Based Design Support and Discourse mangament in User Interface Managment Systems '** , de Jonas Lowgren.

Esta tesis nos sirvió para profundizar el conocimiento a cerca de UIMS y interfaces de usuarios, arquitecturas funcionales de UIMS -modelos lógicos: Edmons, Seeheim, Socket, Hudson- y representación y notación de Interfaces -Redes de Transición, Backus-Naur-Form-BNF-,Modelo de Eventos-.

Esta tesis esta orientada al campo de desarrollo de sistemas expertos, según define Lowgren, las características funcionales son :

- Incorpora conocimiento del Dominio de una tarea dada.
- Usando este conocimiento emula capacidades humanas dentro de ese dominio en un nivel de performance similar al de un experto.

y las características estructurales son :

- El conocimiento esta separado estructuralmente del proceso que lo usa y generalmente esta almacenado en una *'Base de Conocimiento'.*
- El uso del conocimiento para razonar a cerca de los problemas en el dominio de la tarea es manejado por una componente llamada *'Máquina de inferencia'.*

Un sistema Experto tiene 2 tipos de Interfaces de Usuarios : una interfaz de usuario para el ambiente de desarrollo y una interfaz de usuario para el ambiente de entrega, por lo tanto tiene dos categorías de usuarios : ingenieros en conocimiento y usuarios finales.

Tradicionalmente los sistemas expertos conversacionales, han sido considerados para ejercer un cierto control sobre el diálogo. Existen diferentes clases de sistemas conversacionales, variando desde aquellos que tienen un control interno total (el sistema tiene la iniciativa del diálogo) a un control externo total (el usuario tiene la iniciativa del diálogo). Según esto tenemos 4 grupos: Sistemas de Consultas, Sistemas Críticos, Sistemas de Alerta y Sistemas de Transacciones.

Análisis de la Propuesta

En su tesis Lowgren extiende el alcance de las UIMS para proveer soporte en el desarrollo de UI de Sistema Expertos Conversacionales (por medio de IGANTIUS, una herramienta para el diseñador de Interfaces de usuarios de Sistemas Expertos).

La característica de las interfaces generadas con el IGNATIUS, herramienta desarrollada en su tesis, está dirigida especialmente al contexto de Sistemas Expertos.

La otra bibliografía investigada fue **'Intelligent User Intefaces'** , editado por Joseph W. Sullivande y Sherman W Tyler.

Este libro ofrece una perspectiva de la Interacción Hombre-Computadora. Responde a ciertas inquietudes tales como:

- ¿Como se puede hacer más eficiente y más clara a la interacción ?
- ¿Como puede la interfaz de usuario ofrecer mejor soporte para las tareas y objetivos de los usuarios ?
- ¿Como se puede presentar la información más eficientemente ?
- ¿ Como se puede hacer más fácil el diseño e implementación de buenas interfaces ?

El enfoque es trascender las técnicas de ingeniería de software standard para poder tratar una nueva generación de sistemas aplicativos.

La otra bibliografía investigada fue **'Designing the User Intefaces'** , de Ben Shneiderman.

Este libro aporta un profundo conocimiento sobre el diseño de Interfaces de Usuarios mostrando las herramientas disponibles, las técnicas de interacción, sus ventajas y posibilidades de utilización en distintos casos.

En el capítulo 1, presenta un conjunto de alternativas en el desarrollo de interfaces de usuarios y en el capítulo 2 extiende las teorías de diseño, construcción y evaluación de las

mismas. Desde el capítulo 3 al capítulo 5 se describen las técnicas de interacción, menúes, llenado de formularios, lenguajes de comando y manipulación directa, respectivamente. En el capítulo 6 muestra las posibilidades existentes de interacciones físicas -Keyboard, Mouse, Displays, Printers y demás dispositivos de multimedia-. El capítulo 7 evalúa los tiempos de respuesta y su impacto en los usuarios. El capítulo 8 lo dedica al diseño de pantallas -mensajes y helps- y el 9 expande el anterior exponiendo las estrategias de desarrollo de múltiples ventanas. En el capítulo 10 enfoca los beneficios de trabajo cooperativo, en el 11 muestra las herramientas de exploración de información - Hipertextos e Hipermedia-, el capítulo 12 muestra la importancia de los Helps on line y tutoriales. El capítulo 13 expone los 3 pilares del diseño : Documentación, Herramientas de Prototipación y, Testeo y Evaluación. Por último en el capítulo 14 lo dedica al Ambiente de desarrollo de Interfaces de Usurario -UIDE-.

Otra bibliografía investigada fue **'Developing Software for the User Intefaces'** , de Len Bass y Joelle Coutaz.

Los autores explican los conceptos subyacentes en el desarrollo de interfaces de usuario, enfocándolo desde el punto de vista del diseñador y desde el punto de vista del usuario. Este esta muy orientado a las necesidades de la Ingeniería del Software, por un lado trata de capacitar a individuos que están involucrados en el desarrollo de interfaces de usuario para comprender el rol que le corresponde a cada uno dentro del ciclo de vida de la ingeniería del software. Enfatizan el estudio de conceptos de software, modelos y arquitecturas usadas en el desarrollo de una interfaz de usuario. Por último propone una UIMS, la cual tiene un mecanismo de especificación de diálogo integrado y una herramienta de presentación, el **Serpent**. El mecanismo de especificación de diálogo, es un lenguaje especial, Slang, que genera el controlador de diálogo y usa algún toolkit para especificar la presentación. El Serpent, no solamente fue diseñado para enfatizar la separación entre la aplicación y la presentación, sino que además permite integrar un conjunto de toolkits.

Slang, el lenguaje para especificar el diálogo, está basado en el modelo de producción. Los programas en Slang, son una colección de sentencias, IF condition THEN action". Usa restricciones como un método para especificar las relaciones entre los objetos del dominio y los objetos de interacción.

Serpent se comunica con la aplicación a través de una colección de datos compartidos. El diseñador del software de la Interfaz y el diseñador del software de la aplicación, deben ponerse de acuerdo respecto a la información que se le brindará al usuario y sobre el nivel semántico de los comandos que se retornarán a la aplicación. La interfaz consiste en dos clases de datos, datos transferidos por la aplicación a Serpent y datos retornados desde Serpent a la aplicación.

Serpent, intenta eliminar a la aplicación, como Slang, de cualquier dependencia sobre una toolkit o medio particular. Slang, captura todas las dependencias en el uso de una toolkit particular, así cuando una nueva toolkit es utilizada, solamente el programa Slang es modificado.

Consultamos también, **'An Adaptative System Developer's TOOL-KIT'** , Davis Benyon, Dianne Murray y Frances Jennings. Interact '90.

En este paper proponen una arquitectura para un sistema adaptativo y describen una UMS - User Modelling Shell-, que es un conjunto de herramientas para diseñadores de interfaces de usuario basada en conocimiento.

Considera que un sistema adaptativo debe tener 3 componentes:

- Modelo de Dominio : proponen un modelo de 3 niveles : **Nivel de Tarea** -detalla los conceptos que el sistema es capaz de conseguir, descriptos en términos que el usuario pueda entender-, **Nivel Lógico** -describe las funciones y conceptos del sistema que son estrictamente necesarios para completar una tarea- y **Nivel Físico** -describe las interacciones físicas para ejecutar las funciones lógicas-.

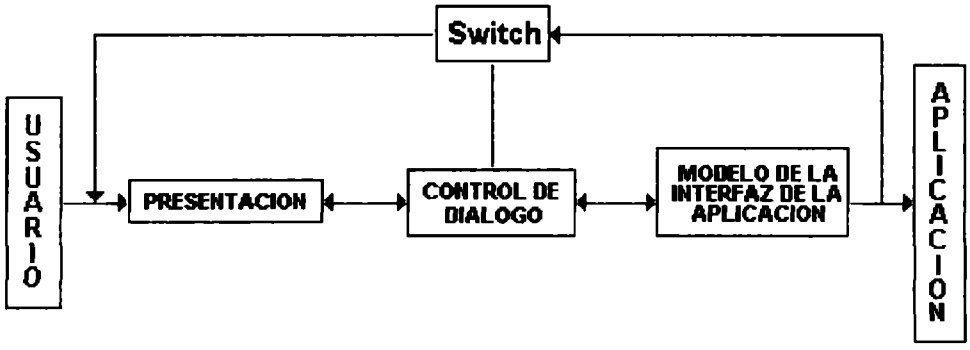
personalidad, estrategias de aprendizaje y estilo cognitivo del usuario- y **Modelo de Estudiante** -contiene información a cerca del entendimiento de los conceptos y funciones del sistema adaptativo-.

- Base de Reglas : Lo definen como compuesto de 3 tipos de Reglas : **Reglas de Adaptación** (controlan las adaptaciones del sistema), **Reglas de Inferencia** (exploran los datos recolectados para inferir características del usuario y actualizar el Modelo de Usuario) y **Reglas de Evaluación** (son las que alimentan las Reglas de Inferencias con datos que tienen que ver con la efectividad de la interacción).

'An Experiment in Interactive Architecturas' , Ernest Edmonds, Noriko Hagiwara.

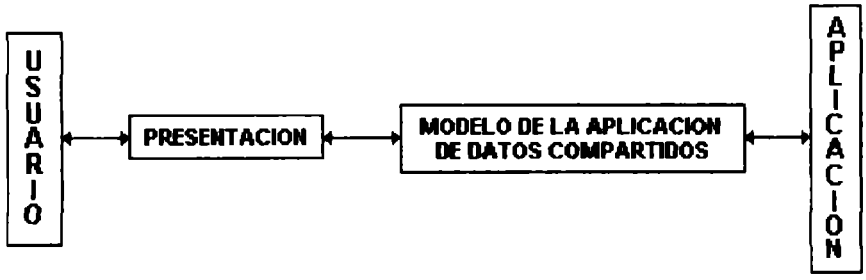
Este paper considera arquitecturas interactivas conocidas como 'Modelos de Seeheim', busca una solución a la manipulación directa y presenta un nuevo modelo de arquitectura para soportarla.

Desde el temprano trabajo de Newman (1968) para programación interactiva gráfica, se le ha dado considerable atención a las arquitecturas para interacción. Moran y Edmonds dieron lugar al ampliamente conocido 'Modelo de Seeheim', en este, el interés estuvo en la separación de los módulos de soft, para proveer al diseñador un sistema interactivo con modularización efectiva, pero no es óptima para manipulación directa.



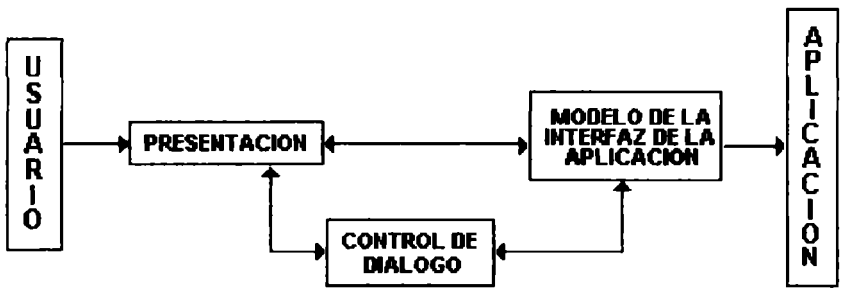
Hudson dedujo que el problema de organizar el soft así, provee dificultades para un apropiado 'Feedback', pues cuando se produce un cambio en un objeto de la aplicación, debe ser

reflejado a través de la interfaz de usuario, necesitando una secuencia para ello. Propone entonces, una nueva arquitectura :



Basándose en esta arquitectura, el objetivo fue construir una herramienta que permita definición de íconos y sus semánticas para sistemas interactivos. Esta herramienta puede ser vista como un elemento esencial de una UIMS que pueda operar exitosamente con la arquitectura empleada -PIT-.

Cuando se intentó aplicar al Modelo de Hudson la construcción PIT, el problema observado fue que un ícono, no necesariamente mapea a un simple objeto de la aplicación -una simple entrada en el Modelo de la Aplicación de Datos Compartidos-. Ellos sostiene que el diseñador está capacitado para asociar un conjunto complejo de acciones de la aplicación inter-relacionados con un simple ícono, o sobre el otro extremo, para decidir que una acción asociada con un ícono solamente afecta a la capa de Presentación. Esto no fue práctico para el control de las acciones de íconos relacionadas y sus consecuencias como parte de la aplicación, fue necesarios entonces, introducir una componente 'Módulo de Control de Acción', que las maneje. Este módulo contiene la descripción de las acciones en términos de cambios de la presentación y las operaciones sobre los objetos de la aplicación.



Es interesante dejar en claro las diferencias entre la temprana arquitectura de Seeheim y la descripta. Seeheim obtuvo una visión mucho mas apropiada para las interfaces modernas que muchos autores -como Hudson-. La diferencia entre Seeheim y la propuesta es que :

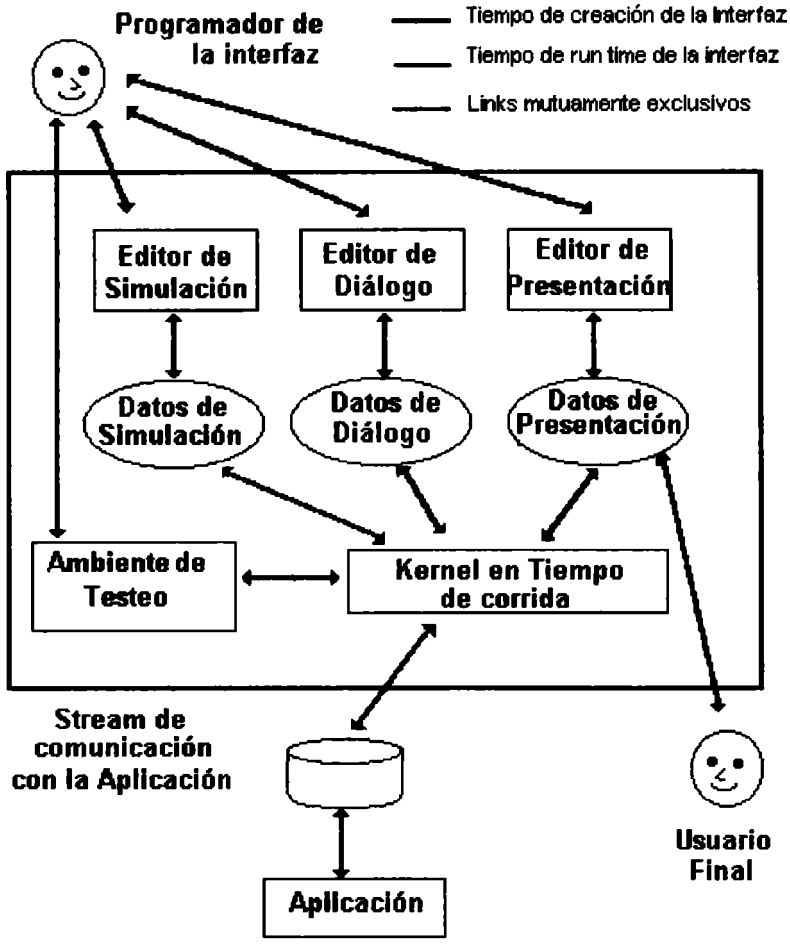
El Módulo de control de acción no tiene acceso a un SWITCH de comunicación directa entre las otras 2 componentes.

Las consideraciones de manipulación directa, imponen que con la tecnología actualmente disponible, es probable que se necesite para una interfaz de usuario completa, la existencia de un proceso como el propuesto. Como conclusión final el sistema propuesto por estos autores, es mucho mas común con el modelo de Seeheim pero apropiado para interfaces de Manipulación Directa.

'Scenarioo : A new Generation UIMS' , Brigitte Roudaud, Valetie Lavigne, Oliver Lagneau.

Este paper estudia el desarrollo de una UIMS, llamada Scenario. SCENARIOO es una UIMS completa, que contiene herramientas para cada fase del proceso de desarrollo de un interfaz de usuario. Permite al diseñador de la interfaz de usuario prototipar, desarrollar, testear y hacer debbug de la misma, también provee de un ambiente de aplicación real y uno simulado, permitiendo correr una o múltiples aplicaciones. Provee editores gráficos interactivos para efectuar el desarrollo de cada parte de la interfaz de usuario (la presentación y el control del diálogo). SCENARIOO extiende la funcionalidad de las UIMS corrientes principalmente en el área de representación del diálogo y ambientes de testeo interactivo para debbuging.

La arquitectura funcional global de SCENARIOO es la siguiente:



Construir una interfaz de usuario con esta herramienta consiste esencialmente en la generación y modificación de tres bases de datos :

- datos de la PRESENTACIÓN -WIDGETS-. Estos datos contienen información sobre los objetos visuales que componen los displays. Por ejemplo: menúes, push-buttons, cajas con texto, etc.
- datos del DIALOGO. Los datos del dialogo consisten de un conjunto de grafos de eventos cada uno de los cuales provee una representación visual del comportamiento dinámico para cada una de las partes de la interfaz.
- datos de la SIMULACIÓN. Estos datos contienen uno o mas conjuntos de datos de simulación de la aplicación, cada uno de estos conjuntos simula valores para todas las funciones en la aplicación. Los diferentes conjuntos simulan diferentes comportamientos de la aplicación .

Cada base de datos tiene un editor interactivo asociado que le permite al diseñador de la interfaz construir la interfaz usando manipulación directa en lugar de la programación. La especificación de la presentación de la interfaz de usuario no es un problema complejo pero puede ser muy tedioso y consumir mucho tiempo. Hoy en día existen un numero de herramientas que permiten que esta especificación sea hecha gráficamente e interjectivamente ahorrando mucho tiempo además generan a partir de esta especificación gráfica de la interfaz una especificación textual asociada o código de implementación.

En cuanto a la especificación de diálogo es mas dificultosa. A medida que el diálogo se hace mas sofisticado y complejo también lo hace su especificación. Hoy no existen herramientas que puedan asistir en la construcción del diálogo y a la vez oculte la complejidad. Si bien se han hecho algunos progresos respecto a esto dándole al diálogo un formalismo tal como en los sistemas basados en gramática o en los sistemas basados en reglas, estos sistemas proveen una forma de especificar el dialogo pero no ayudan en el control y manejo de su complejidad.

Básicamente las representaciones del dialogo actualmente tienen dos problemas, la forma de estructurar el diálogo y la forma de visualizar el diálogo.

SCENARIOO haciendo uso de los grafos de eventos intenta proveer una herramienta que permita que un diálogo compuesto sea descompuesto en uno o mas diálogos simples y proveyendo así, una representación visual del mismo.

En este sistema la especificación del diálogo consiste de la especificación de la serie de acciones y condiciones de su activación. Las condiciones consisten principalmente del testeo de las ocurrencias para un evento particular o el testeo del estado de algunos objetos en la presentación, las acciones consisten principalmente de modificaciones a la presentación o llamadas a funciones en la aplicación.

6

Gerenciador de Interfaces Adaptativas - GIA -

VI.1 Introducción.

Habiendo presentado los tipos de herramientas disponibles para especificar interfaces de usuarios, describiremos nuestra propuesta teórica para un gerenciador de interfaces de usuarios, al que denominamos GIA, cuya particularidad es que genera interfaces adaptativas.

Sabemos que un sistema adaptativo, es un sistema basado en conocimiento, el cual automáticamente altera su aspecto para adaptarse a los requerimiento del usuario. Un sistema adaptativo debe poseer un modelo explícito de cada uno de los usuarios que lo utilicen, y debe poder acceder en cualquier momento a ellos para determinar cuales son sus preferencias, hábitos, y además debe inferir acerca del dominio y del conocimiento que el usuario posee del dominio.

VI.2 Arquitectura para una Interfaz Adaptativa

La arquitectura está compuesta por bases de conocimiento y por módulos inteligentes.

Los **Módulos** son componentes que conforman y manejan la interfaz, alimentados por las Bases de Conocimiento.

• Módulo de Customización : modifica las características de la interfaz para adaptarla al corriente usuario.

• Módulo de Presentación : razona acerca de cual es la mejor manera de presentarle la información al usuario.

• Módulo de Control del Diálogo : determina cuando adaptar la interfaz.

• Módulo de Planificación : asiste al usuario en el logro de sus objetivos.

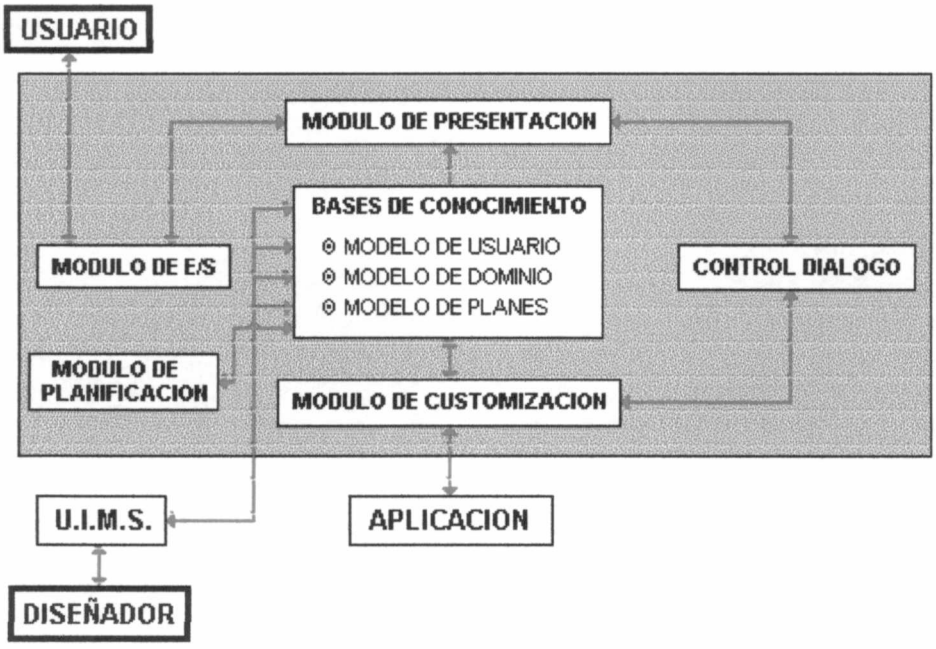
Las **bases de conocimiento** son dependientes del dominio e incluyen:

• Modelo del Usuario: Mantiene la historia de las interacciones de cada uno de los usuarios definidos del sistema.

• Modelo del Dominio: Describe la estructura y funcionamiento del dominio. Está compuesto por objetos, comandos y tareas de alto nivel.

• Modelo de Planes: Describe los planes existentes hasta el momento, inferidos de las intenciones de los usuarios.

Figura 6.1 : Arquitectura para una Interfaz Adaptativa



Todos los **Módulos** de la interfaz usan una o más de estas **bases de conocimiento**, generalmente a través de interacciones con el Módulo de Customización, para lograr sus fines.

Todos los **Módulos** de la interfaz usan una o más de estas **bases de conocimiento**, generalmente a través de interacciones con el Módulo de Customización, para lograr sus fines. Los 4 Módulos constituyen el corazón de la interfaz, c/u de estos incorpora una cierta inteligencia en la funcionalidad de la misma.

Módulo de Entrada / Salida

El propósito de éste módulo, es integrar inputs multimodales, en una representación adecuada, que pueda ser usada por la interacción para razonar acerca de las acciones de los usuarios. Este módulo será el encargado de recibir las secuencias de eventos ingresadas por el usuario, y luego reaccionar a esos estímulos, para que el usuario basándose en esas respuestas, siga interactuando.

El operador interpreta el feedback, evalúa la situación y luego actúa. [Norman 1986].

Este módulo estaría capacitado para correr sobre distintos tipos de dispositivos, es decir habría independencia de dispositivos, y esto es necesario debido al amplio rango de usuarios que utilizarán el sistema.

Módulo de Presentación

Este Módulo, es necesario porque, no sabemos las preferencias y cognición de los usuarios potenciales del sistema, con lo que podríamos desplegarles ciertas presentaciones que no sean las deseadas por el usuario o ciertos usuarios podrían no interpretar las respuestas, por su bajo nivel de conocimiento. Por lo dicho, existe un amplio rango de usuarios, que prefieren y/o necesitan ver la misma información representada con diferentes técnicas de interacción y modalidades.

Este módulo actúa de manera de obtener, la mejor forma de presentarle la información al usuario.

Primero pone en juego la habilidad en la elección de las técnicas de presentación, de manera de obtener a aquellas que expresen la información de la forma mas clara y completa posible. Por Ejemplo : durante la selección de las técnicas de presentación, si hay 2 campos conteniendo información numérica y hay una relación entre ellos, entonces un diagrama de barras podría ser muy expresivo, o también podría decidir por el uso del diagrama de tortas, o en su defecto proponer las dos.

Luego compara las técnicas expresivas y modalidades obtenidas, con el contexto en el que está actuando el usuario, con los modelos de usuario o Estereotipos -Novato, Intermedio y Experto- y con información sobre ventajas y desventajas de cada una de las técnicas versus las otras.

Por último centra su atención al usuario individual, todas las técnicas que resultaron expresivas y efectivas, se mapean con las restricciones obtenidas del modelo de usuario individual, es decir, con sus preferencias, hábitos, etc.

Después de este análisis, representa la información al usuario, con un alto grado de confianza, de que será exitosa.

Módulo de Control de Diálogo

No es adecuado tener una simple interfaz cuando el rango de usuarios habilitados y roles es amplio. Cuando diferentes usuarios hacen utilización del sistema, necesitarán una información descriptiva mas o menos amplia, niveles de ayuda distintos, al igual que una variedad de comandos disponibles. La interacción con el sistema debe ir **adaptando sus aspectos funcionales**, ya sea para distintos usuarios como para los mismos usuarios durante el desarrollo de sus tareas, proceso en el que va variando su nivel de conocimientos.

Hay tres problemas que la interfaz debe resolver y que son: **Cuándo Adaptar? , Qué Adaptar? , Cómo Adaptar?**

La tarea encomendada al Módulo de Control de Diálogo responde a la primer pregunta. El sistema debe tener alguna forma de mirar el diálogo, que le permita detectar cambios en el corriente estado del diálogo. En el Módulo de Control de Diálogo, estos cambios puede detectarlos mirando la interacción entre el usuario y el sistema, como consistente de una serie de eventos de interacción.

Si vemos al diálogo como una serie de **eventos de interacción**, c/u compuesta de esas 3 fases -definidas en el capítulo 2 Punto 2.3.-, el Módulo de Control de Diálogo debe detectar cuando se produce un cambio de fase y proceder en función de eso. Si fue una fase de Input, debe avisarle al Módulo de Customización que el usuario entró un requerimiento, para que lo evalúe, entrando en la Fase de Ejecución para que actúe adecuadamente y culmine con la Fase de Respuesta del sistema, desplegando las respuestas al usuario.

Módulo de Customización

Esta es la Componente más inteligente e imprescindible para la **Adaptación,** pues determina que adaptar y cómo hacerlo.

Para determinar qué adaptar, la interfaz necesita conocer cuál es el posible rango de opciones para cada atributo que queremos customizar de la interfaz, para cada uno de los patrones del modelo de usuario. Es sabido, que el modelo de usuario está compuesto por 3 patrones, veremos entonces qué y cómo adaptar cada uno de ellos.

¿ Qué Adaptar ?

Se adaptan las views de cada objeto de interacción, según las preferencias del usuario, que se encuentran en el **Patrón de Preferencias** inferido del usuario.

Secuencias de comandos detectadas, que están almacenadas en el **Patrón de Hábitos**. Algunas de estas secuencias podrían afectar la presentación de un objeto de interacción. Esta secuencia puede estar relacionada directamente con la presentación -tal como acomodar una

¿ Cómo Adaptar ?

Gran parte del poder del Customizador se debe a la fluída interacción que mantiene con los otros módulos de la interfaz, ya que actúa sobre el modelo de usuario, para actualizar la historia de las interacciones, para tomar de las interacciones pasadas y atributos heredados, para poder interpretar inputs ambiguos, y por último, para reducir el espacio de búsqueda de acciones a llevar a cabo infiriendo del nivel de usuario y del contexto en el que se esta desarrollando. Indica cual es la mejor forma de proceder para lograr satisfacer al usuario. Para determinar cómo adaptar, hacemos uso de las bases de conocimiento.

Debido a sus funcionalidades, dividimos al Customizador en 4 módulos :

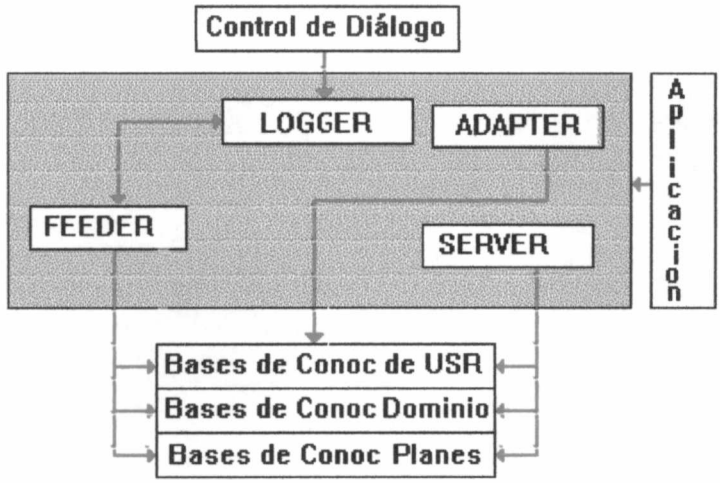
El Logger encargado de recolectar los eventos producidos durante la interacción, es decir, aquellos datos relevantes para la historia del usuario y customización de la interfaz.

El Feeder actualiza los patrones del Modelo de Usuario y Modelo de Planes, haciendo uso de las nuevas interacciones. El Adapter es activado por el 'Feeder', después de la actualización de los patrones, éste testea si es necesario hacer la adaptación, comparando con el patrón de preferencias y si lo es, actualiza los valores activos de los patrones de usuario.

El Server provee funciones para obtener información de las bases de conocimiento permitiendo a las otras componentes de la Interfaz, usar las bases de conocimiento sin tener que conocer los detalles internos de su estructura.

El Adapter es activado por el Feeder, después de las actualizaciones de los patrones, este testea si es necesario hacer la adaptación comparando con el patrón de preferencias, si lo es hace efectiva la adaptación reflejando los cambios en la representación de la interfaz.

Figura 6.2 Componentes del Módulo de Customización



Cuando el Customizador es notificado desde el Controlador de Diálogo que un evento se produjo, este módulo actualiza las Bases de Conocimiento por medio del Feeder, pone en práctica los mecanismos de adaptación usando el Adapter. Este consulta el estado de las Bases de Conocimiento apropiadas y basándose en los contenidos de esos patrones, determina que características serían presentadas en la interfaz, basándose en los valores de esos parámetros y la interfaz es modificada adecuadamente.

Cabe notar, la diferencia fundamental entre la customización de Hábitos y Preferencias y la de Conocimiento.

- La *customización de Hábitos y Preferencias*, puede mantenerse fija durante los distintos niveles de Usuarios o puede variar junto con ellos, pues es claro que los gustos y preferencias, al igual que los hábitos no están directamente relacionados con el conocimiento del usuario que es lo que marca la diferencia de niveles. Supongamos que 2 nuevos usuarios ingresan al sistema, y por default se los ubica en un nivel intermedio, ambos poseen los patrones de preferencia y hábitos idénticos. Con las interacciones de ambos en las sucesivas sesiones, dichos patrones pueden ir diferenciándose, y sin embargo no significa que hayan cambiado de nivel de conocimiento.

- La *customización del Patrón de Conocimientos*, está íntimamente relacionado con el conocimiento del usuario, supongamos la situación expuesta arriba, al ingresar poseen un patrón

- La *customización del Patrón de Conocimientos*, está íntimamente relacionado con el conocimiento del usuario, supongamos la situación expuesta arriba, al ingresar poseen un patrón de conocimiento en común, pero con las sucesivas interacciones, el conocimiento de ellos puede variar, uno puede cometer más o menos cantidad de errores, puede pedir mas o menos veces ayudas al sistema, etc., lo que implica un cambio de nivel en el conocimiento.

Módulo de Planificación

Esta fuera del alcance de esta tesis la profundización sobre el estudio de planes. Haremos una breve referencia sobre el contendido de éste módulo y su tarea, con el fin de inquietar a algún lector y continúe con su estudio.

La función del Planificador es asistir al usuario en el logro de sus objetivos. Dentro de las Bases de Conocimiento, se encuentra el Modelo de Planes -formado por planes-, representados con una estructura de grafo que va desde un estado inicial, **Nodo Inicial** hasta el nodo final es el **Nodo Objetivo,** por distintas ramas. 'La palabra **plan**, es usada como un método para lograr un fin, a menudo customizar alguna acción a seguir'. [CHIEN 90]

Definimos a un plan, como una sucesión de pasos u operaciones automatizadas para llegar a un objetivo deseado; es decir, partiendo de un *Estado Inicial*, encontrar acciones que transformen a este estado inicial en el *Estado Objetivo*.

Para llegar a los nodos, puede hacerse de 2 maneras :

1.- Con Búsqueda desinformada, es decir por medio de los métodos de recorrido de grafos conocidos (BPS,DPS, etc.).

2.- Con Búsqueda informada, se puede determinar cuál es el nodo óptimo para expandir la búsqueda para llegar al objetivo -se haría por medio de conocimiento explícito almacenado en los distintos estados del grafo y teoría de decisiones que permiten seleccionar la opción más adecuada con el grado de información que se disponga-.

Con cualquier tipo de búsqueda que se trabaje, el objetivo es encontrar la solución al problema si es que existe -buscando en los planes existentes en el sistema-, con el mínimo esfuerzo computacional posible, de lo contrario, detectar que no hay solución posible.

Operando con el conocimiento de los objetivos y planes del usuario, la interfaz podría lograr lo siguiente:

- Detectar y tratar de corregir errores globales.
- Completar tareas de alto nivel.
- Llenar parámetros por default e interpretar requerimientos ambiguos.
- Completar tareas del Usuario.
- Ayudar al usuario en el mapeo de sus objetivos de alto nivel, en comandos de la aplicación.

Las bases de conocimientos están compuestas por 3 Modelos :

Modelo de Dominio

Está compuesto por todos los procesos (funciones) y datos (objetos de interacción) que podrán utilizarse para la creación de la UI. El diseñador es el encargado de definir procesos o funciones que están involucradas con las aplicaciones. Este modelo se codifica en la base de Conocimiento del Dominio .

Modelo de Planes

Está compuesto por los Planes inferidos de las acciones de los Usuarios. Los Planes son estructuras de conocimientos, para capturar la experiencia e intenciones de un experto en un dominio dado , sin esfuerzos de programación.[CHIEN 90].

No se profundizará en esta tesis, el tema Planes.

Modelo de Usuario

Uno no puede esperar construir un sistema que sea fácil de usar y apropiado, sin haber identificado las propiedades salientes de los usuarios.

Los modelos de usuario son muy importantes, se crean para adaptar el comportamiento del sistema al usuario, no son necesarios para todos los sistemas, si lo son para sistemas en los que :

- se adapta su comportamiento a usuarios individuales.
- se asumen ciertas acciones, es decir, comparte responsabilidad con el usuario.
- se desconoce la clase potencial de usuarios del sistema.

Los modelos de usuarios nos sirven para, identificar obstáculos en un plan de usuario, reconocer cuando una acción o consulta no refleja las metas del usuario, para responder de acuerdo a las perspectivas del usuario, tratando en todo momento de satisfacer sus objetivos, contemplando gustos, preferencias y conocimientos -tomados del modelo de usuario-.

No es posible la identificación de las características de los usuarios del sistema, dada la generalidad que tendrá la interfaz de usuario , por no conocerse la aplicación ni los potenciales usuarios que operarán con la interfaz de usuario, por lo que la especificación podría ser de infinitas combinaciones.

Una clasificación típica de la comunidad de usuarios es agruparlos según su '*Conocimiento*' en ***Novatos, Intermedios y Expertos***, pero antes de especificar cuales son sus características definimos que tipos de conocimientos existen:

Conocimiento Sintáctico: representa las expresiones válidas del lenguaje, que el usuario debe conocer para comunicarse con el sistema interactivo, para:

- Requerir objetivos al sistema (Expresiones de Entrada).

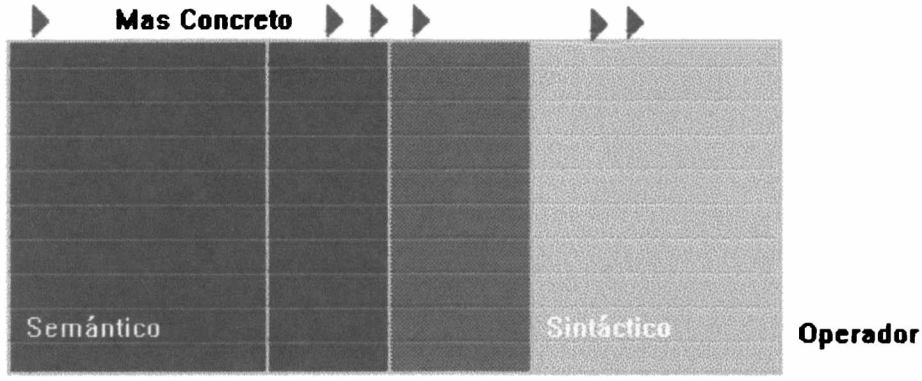
- Interpretar respuestas del Sistema (Expresiones de Salida).

Conocimiento Semántico: es una jerarquía organizada de objetos y operaciones válidas sobre dichos objetos. Dentro de un sistema interactivo, el *conocimiento semántico dependiente del dominio*, es inherente a la tarea del dominio, mientras que *el conocimiento semántico dependiente del sistema* es el conocimiento de la funcionalidad del sistema interactivo.

sistema interactivo y de la tarea del dominio. En esta representación, los conceptos de interés en un momento dado para una tarea particular, son llamadas variables psicológicas [Norman y Draper 1986].

El conocimiento semántico es generado por la aplicación y convertido por los niveles de la Interfaz de usuario en sintáctico, para la presentación al operador.

Figura 6.3 Niveles de Conocimiento



Ahora si podemos decir que : [Shneiderman 87]

Los Usuarios NOVATOS : no tienen conocimiento sintáctico, posiblemente poco conocimiento semántico acerca del dominio de la computadora y de las tareas del dominio.

Los Usuarios INTERMEDIOS : tienen un buen conocimiento semántico acerca de los dominios de las tareas y de los resultados de la computadora. Carecen del conocimiento detallado de la sintaxis.

Los Usuarios EXPERTOS : están familiarizados con los aspectos de semántica y de sintáctica del sistema.

Estos son los tres niveles clásicos, pero podemos encontrarnos con usuarios que sean novatos en su conocimiento semántico, pero expertos en su conocimiento sintáctico. Podría por ejemplo tener conocimiento del dominio de una tarea y desconocer el uso del sistema.

Los Usuarios EXPERTOS : están familiarizados con los aspectos de semántica y de sintáctica del sistema.

Estos son los tres niveles clásicos, pero podemos encontrarnos con usuarios que sean novatos en su conocimiento semántico, pero expertos en su conocimiento sintáctico. Podría por ejemplo tener conocimiento del dominio de una tarea y desconocer el uso del sistema.

La comunidad de usuarios, también podría clasificarse por los roles que desempeña cada usuario, o por jerarquías de acceso.

Independientemente del conocimiento que posea cada usuario, respecto de una aplicación, se debe tener en cuenta, que también los distinguen sus preferencias respecto a la presentación de la interfaz. Por esto, debemos prestar particular atención, a los pedidos de cambio que estos efectuén en la apariencia de los objetos de la interfaz, tales como tipo de letra, tamaño, colores, íconos preferidos, etc, por lo tanto para cada usuario y para cada aplicación, debe mantenerse una base de conocimiento con las preferencias para dicha aplicación.

Las distintas apariencias que puede tomar una interfaz depende del contenido de la base de conocimiento, que va variando de acuerdo con las sucesivas interacciones de usuario con las aplicaciones generadas con la herramienta :

- Si es la primera vez que se logonea al sistema, pueden ocurrir dos cosas :

1. no existe modelos para su tipo de usuario, por lo que la apariencia de la UI, será aquella que definió el diseñador de la interfaz. -modelo genérico para cualquier tipo de usuario-.
2. existen los modelos para su tipo de usuario, entonces la apariencia de la interfaz será aquella que definió el Diseñador de Modelos de Usuarios, para su tipo de usuario. -modelo genérico para un tipo de usuario particular-.

- Si no es la primera vez que se logonea al sistema -ya interactuó con alguna aplicación anteriormente-, pueden darse dos cosas :

3. Que ingrese a una aplicación por primera vez : tomará el modelo del usuario inferido por su interacción con otros usuarios. -modelo genérico de un usuario particular-.

4. Que ya haya interactuado con tal aplicación : tomará el modelo inferido para esta aplicación en particular. -modelo particular de un usuario para una aplicación determinada-.

Otro aspecto a modelar serían los hábitos del usuario. Consideramos hábito a una secuencia ordenada de 5 eventos de la aplicación que se detectan en el comportamiento del usuario repetidas veces en las sucesivas sesiones. (los eventos fueron detallados en el capítulo 2, Inc. 4).

Dentro del modelo de usuario, para cada una de las aplicaciones del dominio, se mantienen las últimas sesiones, y de ellas se infieren los patrones de conducta, eliminando ambigüedades. Veremos como un hábito detectado se convierte en activo para el usuario para las sucesivas sesiones.

Sean H1 y H2 hábitos detectados durante la interacción y $Ev_{I,J}$ eventos de la aplicación, tal que :

H1 = (D1, R1) , H2 = (D2, R2), tal que :

D1=($Ev_{1,1}$; $Ev_{1,2}$) y R1=($Ev_{1,3}$; $Ev_{1,4}$; $Ev_{1,5}$).

D2=($Ev_{2,1}$; $Ev_{2,2}$) y R2=($Ev_{2,3}$; $Ev_{2,4}$; $Ev_{2,5}$).

Si D1=D2 y R1<>R2 => H1 ambiguo con H2, por los tanto ambos hábitos se rechazan. Si D1<>D2 y R1<>R2 => H1 y H2 pasan a conformar patrón de hábitos del usuario.

Otra particularidad es que los hábitos pueden tener dos estados: *activo o pasivo.* Cuando se detecta un hábito, esta en estado activo, por lo tanto cuando se detecte alguna secuencia Di se disparará automáticamente la subsecuencia Ri. Pero en cualquier momento, el usuario puede consultar cuales son los hábitos que el sistema detectó y en caso que crea conveniente cambiarle el estado a pasivo, esto implica una desactivación temporaria del mismo.

7

Implementación del Prototipo - CUIMS -

VII.1 Introducción

CUIMS es nuestra implementación del prototipo GIA. La pantalla principal de la misma, integra tres sistemas que pueden ser utilizados independientemente por personas con roles bien diferenciados, pero desde el punto de vista de su funcionalidad se encuentran íntimamente relacionadas. Los ingresos a tal sistema pueden ser, como **Diseñador Interfaces de Usuarios** o como **Diseñador de Modelos de Usuarios** o como **Usuario Final**.

- Si accede como Diseñador de Interfaces, accede a una herramienta que le facilite cada una de las etapas del desarrollo de las interfaces de usuarios, que son Diseño, Construcción, Evaluación y Mantenimiento.

  La herramienta gráfica le permite **Diseñar** la Presentación de la interface por medio de manipulación directa y especificar el control de diálogo para todos los objetos de la interfaz. Esta herramienta a partir de esta especificación gráfica de la interfaz de usuario genera automáticamente el código de implementación de la interfaz -Presentación y Control de Diálogo- por lo tanto la **Construcción** es automática. La **Evaluación**, también es provista por esta herramienta en tiempo de desarrollo. Por último, el código generado puede editarse desde la misma herramienta. permitiéndole al diseñador definir secuencias de diálogo más complejas que las generadas automáticamente. El **Mantenimiento** puede efectuarse por 2 vías, por medio de manipulación directa sobre los objetos de la Interfaz o editando su código y modificándolo en forma manual.

- Si ingresa como Diseñador de Modelos de Usuarios, podrá definir y mantener estereotipos de usuarios. Para cada tipo de usuario, se requiere la definición de un modelo, que implica la especificación de características comunes a todos los usuarios de ese tipo, tales características se definen en los correspondientes patrones. Estos patrones son el medio que permite lograr la adaptabilidad de las interfaces. El diseñador de los modelos define características genéricas para cada tipo de usuario desde el punto de vista de la presentación y también las características desde el punto de vista del conocimiento

  La creación y mantenimiento del Patrón de Preferencias, se ve facilitada por un editor interactivo en el cual el diseñador de los modelos de usuario puede establecer características en la presentación de los objetos de la interfaz, que él considera apropiadas para ese grupo de usuarios -tales como : color de fondo y frente, fonts, etc.-.

  Para mayor comodidad en esta definición, aquí se intento agrupar los objetos de interacción que poseen características y funcionamiento similares, por ejemplo existe un grupo Botón que incluye aquellos objetos que poseen todas las funcionalidades de un botón : check-box, radio-buttons, buttons , etc.

  Para la creación y mantenimiento del Patrón de Conocimientos se provee de un editor gráfico interactivo que permite definir los mensajes de errores y advertencias , los helps que asistirán al usuario para cada uno de los modelos definidos y para cada aplicación. Con esta herramienta se pueden tener distintas visiones de un mismo mensaje o help teniendo en cuenta el tipo de usuario. El diseñador de los modelos de usuarios podría, por ejemplo, desear definir para un tipo de usuario novato, helps y mensajes mas gráficos y expresivos que para usuarios con mas experiencia , esta herramienta le facilita enormemente esta tarea.

- Si ingresa como usuario final, se le pedirá una identificación de usuario, que activará patrones de comportamiento asociados al mismo posibilitando así, que una misma aplicación actúe y luzca de forma diferente cuando la utilicen distintos usuarios. Existe un conjunto de

aplicaciones generadas con CUIMS que conforman el domino de aplicaciones disponibles, del cual el usuario podrá seleccionar, definiendo así su propio subconjunto de aplicaciones.

Seguramente todas aquellas aplicaciones que son utilizadas por un mismo usuario, tendrán características homogéneas entre ellas, pudiendo ser totalmente distintas al subconjunto de otro usuario.

VII. 2 Características de la Interfaces logradas con esta Herramienta.

Bimodal

En una interfaz generada con CUIMS, un usuario final podría operar con los objetos de la misma en dos modos :

- Funcional : los efectos producidos por tal interacción tendrán consecuencias directamente sobre la aplicación.
- Presentacional : los efectos producidos en este modo realizarán un cambio automático en la apariencia de los objetos.

Con ambos modos el usuario alimenta las bases de conocimiento, a nivel funcional servirá para detectar hábitos en la interacción del usuario y a nivel presentacional, para determinar cambios de gustos y preferencias.

Todas las interfaces generadas con esta herramienta tendrán un botón ubicado en la esquina inferior derecha que le permitirá al usuario intercambiar de modo. Para ir al modo presentacional, debe clickearse el botón de herramientas, y para ir al modo funcional debe clickearse el botón que dice Aplicación.

En el modo presentacional, el click sobre un objeto, ocasionará la apertura de un editor, para peticionar cambios en los atributos del objeto corrientemente seleccionado pudiendo tener este cambio un alcance local o global.

- Un cambio local afectará solamente al objeto seleccionado.

Un cambio global, afectará a todos los objetos del tipo seleccionado. Esto es, si hay 5 check box en la interface, y fue seleccionado uno de ellos, los 4 restantes, también serán seleccionados.

Además, en este nivel clickeando el botón derecho del mouse se podrán visualizar los hábitos detectados por el sistema, pudiendo también cambiar el estado de los mismos .

Adaptativas

Cuando el sistema logra tener un patrón confiable para un usuario, todas las características del mismo se verán reflejadas en las interfaces automáticamente. Un patrón confiable, significa que la cantidad y calidad de pedidos efectuados por el usuario para un objeto determinado, ha alcanzado la cantidad definida por el diseñador de modelo de usuarios, para que se efectúe la adaptación.

Asistentes

Es sabido, que el sistema recolecta datos relevantes de las interacciones del usuario con el sistema, y de estos trata de inferir hábitos. Se desprende, que, después de un numero de sesiones estudiadas, un usuario tendrá ciertos hábitos para cada aplicación, por ello, cuando se detecta una sucesión de eventos que forman parte de un hábito, se lo asiste finalizando esas tareas automáticamente.

Uniformes

Todas las interfaces de un usuario, generalmente tendrán el mismo aspecto, salvo que éste explícitamente, solicite lo contrario, ya que el sistema detecta un patrón de preferencia genérico para cada usuario, que lo aplicará a todas las interfaces que se vayan incorporando a su dominio de aplicaciones.

VII. 3 Limitaciones del Smalltalk

No soporta generación de DLLs, por ello, cuando deseamos cambiar la forma del cursor para proveer al usuario una indicación gráfica respecto al modo en el que está operando, debimos recurrir a otro lenguaje, como Turbo Pascal, que lo permite. Los objetos provistos por Smalltalk/V, son más limitados que los provistos por Windows. Por ejemplo no provee objetos

esencial para nuestros objetivos, a pesar de parecer trivial, la incorporación de esta característica requirió un considerable tiempo de desarrollo -cabe destacar, que no es una solución posible agregar #clicked a los eventos soportados por un pane-. El drag de los objetos, fue incorporado a la jerarquía del Smalltalk por medio la clase InterfaceObject del WindowBuilder, pero se requirió de la creación de objetos paralelos, que soporten la característica de adaptabilidad, además de todo el código que soporte la adaptabilidad, la detección de hábitos, etc., todo generado automáticamente por la herramienta.

VII. 2 Modificación de la Jerarquía de Smalltalk, para el desarrollo de los sistemas.

Object
- **Atributo**
- **ModelUsr**
- **ObjetosInteraccion**
  - **AnimationP**
  - **Botones**
    - **Boton**
    - **Check**
    - **Radio**
    - **ThreeState**
  - **DrawnB**
    - **BitmapE**
  - **EntryF**
  - **GraphP**
  - **GroupB**
  - **Listas**
    - **Combo**
    - **List**
    - **MultipleList**
  - **StaticB**
  - **StaticG**
  - **StaticT**
  - **TextP**
- **Patrones**
  - **Conocimiento**
  - **Habitos**
  - **Preferencias**
- **ViewManager**
  - **Aplicaciones**
    - **Clases creadas con la Herramienta**
  - **Arranque**
  - **BitEditor**
    - **EditorIconos**
      - **IconEditor**
  - **ClassHierarchyBrowser**
  - **EditaCodigo**

Generador
EditaPatCon
EditaPatHab
EditaPatPref
VentanaUser
WindowDialog
AboutCuims
Editores de las Clases de Objetos de Interacción (EditaCheck, EditaRadio, . . .)

ViewUser
Window
ApplicationWindow
TopPane . . .
SubPane
ControlPane
Button
DrawnButton
BitmapEstatico
Toggle . . .
EntryField
ListBox . . .
StaticPane . . .
GraphPane . . .
TextPane
GroupPane

WinHandle . .
WinInfo
WinLogicalObject . . .
WinStructure . . .


Clases Agregadas


Clases Existentes

Haremos un breve detalle de las características de las Clases agregadas :

Atributo : es una característica de los objetos de Interacción (Color de Frente, Color de Fondo, Font. . .) agregados para efectos de customización. Cada Objeto de Interacción, esta compuesto por un conjunto de atributos.

Las variables de Instancia son :

- nomatr : Nombre del atributo (ej: 'COLOR DE FONDO')
- valact : valor actual del atributo (ej: 'ROSA', si el atributo corresponde a un Color)
- frecuencias : es un Bag, donde se guardará los pedidos de cambios efectuados en las sesiones, de la forma pedido y frecuencia. Cuando la frecuencia alcanza el numero para customizar, se efectúa la customización automática.

- Hay métodos que son utilizados por la UI, transparentes para el usuario, como customice: unObjeto, actFrecuencia : unObjeto, etc. para lograr la adaptación.

ModelUsr : Representa el Modelo del Usuario. Mantiene todas las aplicaciones actuales del usuario y los registros de las interacciones del mismo con las diferentes aplicaciones en su pasado. Esta Base de Conocimiento del Usuario, se usa para filtrar sus preferencias, cada vez que entra a una sesión, ya sea utilizando una aplicación existente o para ejecutar por primera vez una nueva.

Las Variables de Instancia son :

- nombre : PassWord del usuario.
- ppref : es una instancia del patrón de Preferencias.
- phab : es una instancia del patrón de Hábitos.
- pcon : es una instancia del patrón de Conocimiento.
- tipoUsr : Es uno de los Modelos definidos por Diseñador de los Modelo de Usuarios, al que pertenece, como por ejemplo : Novato, Experto o Intermedio.
- viewUser : Es la View actual del usuario, como la esta viendo en este momento, es el aspecto visual.
- aplicaciones : Es una colección de aplicaciones adaptativas que conforman el dominio de aplicación del usuario.

Arranque : Esta es la ventana principal de la Tesis. Simplemente es la que agrupa todas las posibilidades de ingresar al sistema.

Aplicaciones : Esta clase implementa los métodos que son comunes a todas las aplicaciones necesarios para efectuar la adaptación de la Interfaz -Presentación-, así como de los hábitos del usuario.

Objetos de Interacción : Es una clase que agrupa a todos los objetos de interacción que se pueden adaptar. Las Variables de Instancia son:

- posición y tamaño : son instancias de la clase Point. Posición es el vértice superior izquierdo y tamaño es el alto y ancho.
- colorFondo, colorFrente, aFont, nombre, estilo : dan las características que sus nombres indican.

ViewUser : es una subclase de Object, cuyas variables de instancia son :

- pane : es una instancia de alguna de las instancias de Objetos de Interacción (Check, Radio, etc).
- colorFondo, colorFrente, etiqueta, font y estilo : son instancias de Atributo.

Esta clase se usa para guardar una instancia de cada uno de los objetos a adaptar.

Patrones : es una subclase de Object, que agrupa a los 3 tipo de Patrones que conforma el Modelo de Usuario. Dichos patrones no tienen nada en común, su agrupación se debe a una cuestión de orden.

Preferencias : esta clase implementa la administración de las características de presentación para cada usuario particular. Por ejemplo, supongamos que el Diseñador, definió que para los Novatos los BOTONES serán Rojos y para los EXPERTOS serán Verdes. Si un UsuarioX entra por **primera vez**, pertenecerá a uno de estos grupos, y por ello los botones -CheckBox, RadioButton, etc.-, serán del color correspondiente a su tipo, a partir de ahí, se irá infiriendo del UsuarioX sus preferencias y se almacenará uno por usuario.

Tiene solo una variable de Instancia :

- atributos : es la colección de objetos genéricos respecto a la presentación deseada por cada usuario.

Conocimiento : proporciona la funcionalidad necesaria para administrar el Patrón de conocimiento de los modelos de usuario.

Tiene solo una variable de Instancia :

- helps : mantiene los distintos mensajes de ayuda, error o advertencia para cada uno de los tipos de usuarios para cada aplicación.

Hábitos : esta clase implementa el comportamiento de una parte del Módulo de Customización del modelo de arquitectura de una Interface Adaptativa, propuesto en el capítulo 6.

Tiene solo 2 variables de Instancia :

- Hábitos : mantiene los hábitos activos de un usuario para una aplicación.
- Sesiones : mantiene las últimas sesiones recolectadas de las interacciones del usuario. En el momento de desechar una sesión del sistema, se lo hace por medio de la metodología FIFO, previa detección de hábitos.

Generador : esta clase implementa la herramienta que le permite al Diseñador de la UI generar el código de la Interfaz Adaptativa.

Posee muchas y complejas variables de Instancia, no creemos oportuno detallarlas para no complicar al lector.

EditaPatPref : esta clase permite crear los Patrones de Preferencias para los distintos Modelos de Usuarios.

EditaPatHab : esta clase permite crear los Patrones de Hábitos para los distintos Modelos de Usuarios.

EditaPatCon : esta clase permite crear los Patrones de Conocimientos para los distintos Modelos de Usuarios.

Palabras Finales

El proyecto posibilita contar con un sistema que evidencia algunas de las características fundamentales de la próxima generación de interfaces de usuarios, es decir, *interfaces adaptativas.*

La arquitectura en bloques propuesta compite ventajosamente con otras existentes y permite un mayor grado de profundización en ciertos temas a partir de cambios internos a cada módulo sin necesidad de cambios globales.

La implementación realizada además de ilustrar una cantidad de puntos para ejemplificación

de los conceptos básicos, fue desarrollada con una metodología incremental que permite incorporar nuevas funciones en forma simple y manteniendo el esquema de trabajo. Como todo trabajo pionero en su área, no todas las posibilidades fueron consumadas, pero se inició una línea que admite varias continuaciones tanto en lo que hace a implementarlo en otras plataformas como incorporar componentes más potentes para inferencia y planificación.

A partir de la utilización empírica para el desarrollo de prototipos en la asignatura *Prototipación e Interfaces de Usuarios*, ha facilitado la comprensión de la arquitectura básica de una interfaz de usuario y de una interfaz de usuario adaptativa -como una extensión de la primera- y consecuentemente se ha logrado una mejor comprensión de los distintos roles que intervienen en el desarrollo del software. Así mismo, se han vislumbrado posibles extensiones para enriquecer el modelo propuesto y para potenciar la herramienta de desarrollo.

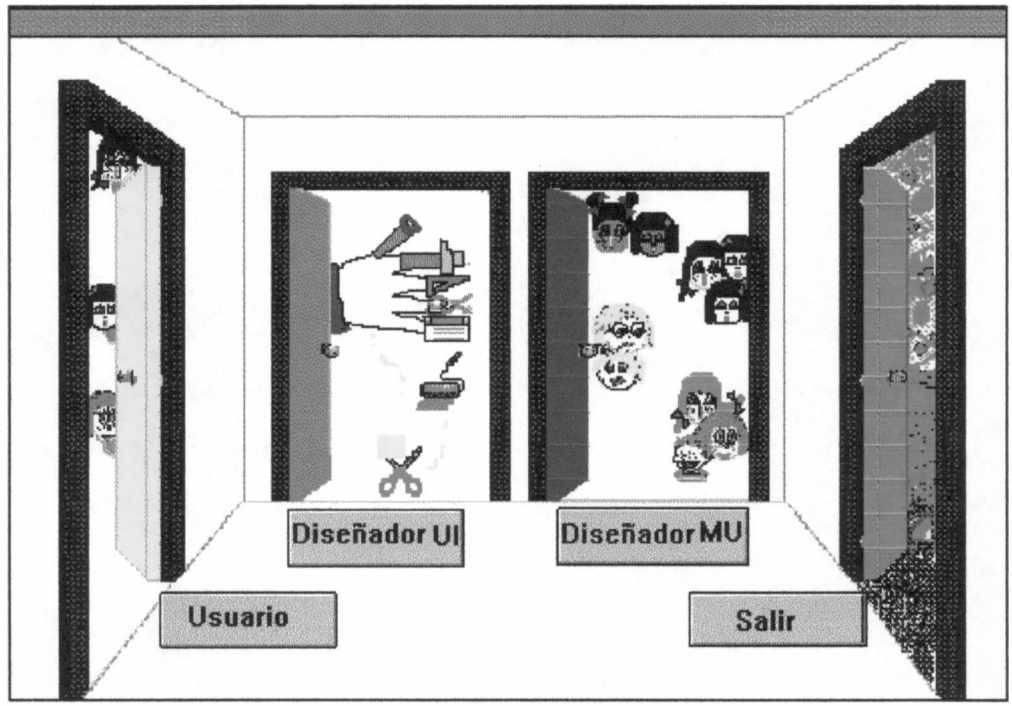
Entre las extensiones posibles, podemos mencionar :

- Potenciar los mecanismos para inferencia y aprendizaje de hábitos y preferencias.
- Agregar capacidad para tener en consideración la finalidad de la tarea del usuario -haciendo uso de la planificación-.
- Hacer a la misma herramienta de construcción de interfaces *más inteligente*, para beneficiar mas aún al diseñador de interfaces.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Manual del Usuario

La ventana principal consiste en cuatro puertas, que indican los distintos accesos, como diseñador de Interfaces de Usuarios **-Diseñador UI-**, como diseñador de Modelos de Usuarios **Diseñador MU-**, como usuario final de alguna aplicación **-Usuarios-** y salir del sistema **-Salir-**.

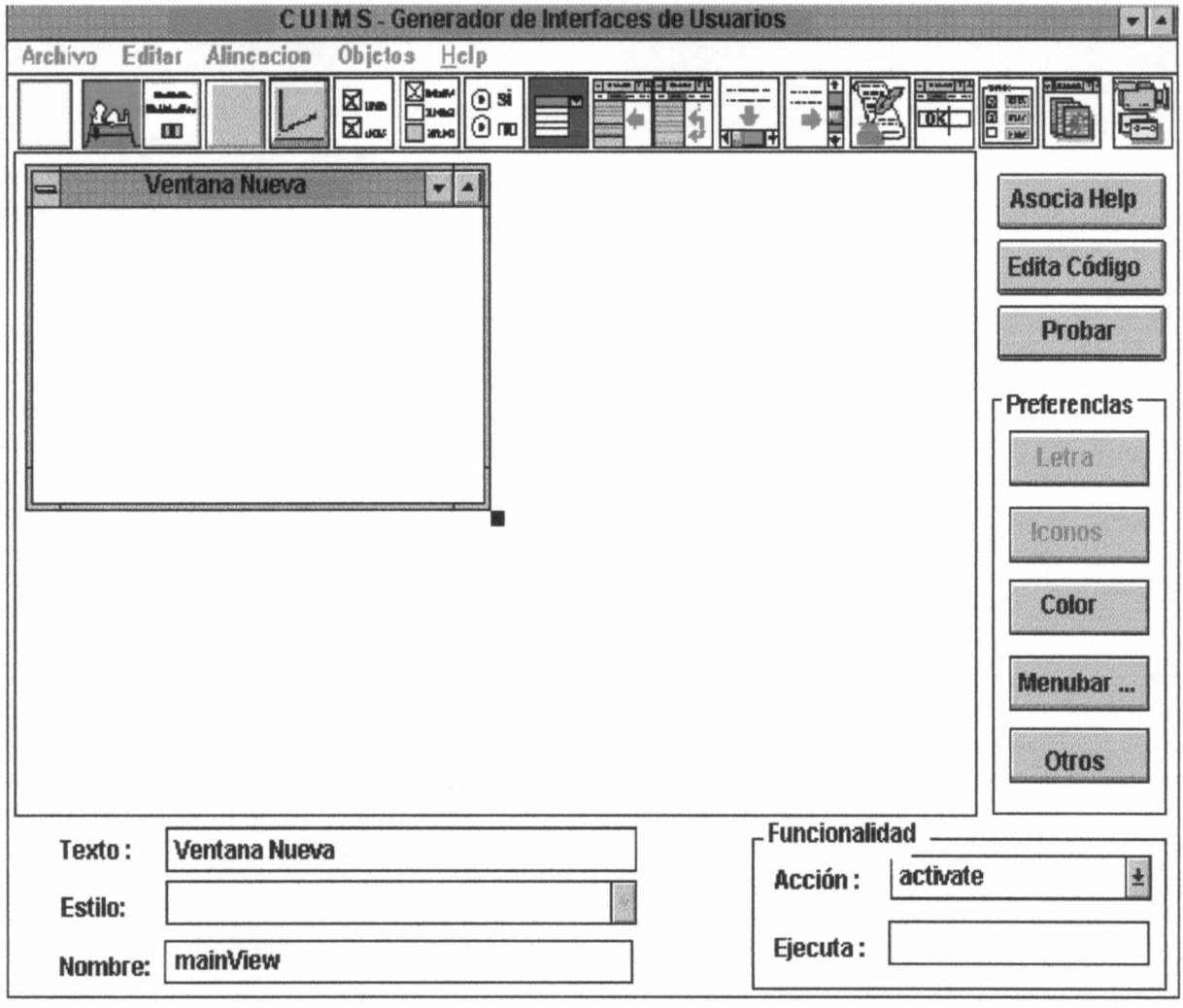


A. Diseñador de Interfaces de Usuarios

Esta herramienta permite diseñar prototipos de interfaces rápidamente. Las facilidades que esta provee son :

- La creación y edición de la interface de la aplicación se hace por medio de manipulación directa de los objetos de interacción que forman parte de la interface.
- La generación automática del código de la interface de la aplicación.

La ventana que se le presentará para comenzar a crear una interface es la siguiente :



Como puede apreciarse esta compuesta de la siguiente manera :

1. <u>Barra de Menú</u> : Provee las siguientes opciones :

1.1 Opciones de Archivo (se aplican a la interface que esta siendo editada).

- Ventana Nueva.
- Ventana nueva de Dialogo.
- Salvar / Salvar como / Salvar por Default.
- Edición de código.
- Tamaño.

1.2 Opción de *Edición* : Son las opciones que se aplican a los objetos de interacción de la interface tales como *Copiar, Cortar .., etc.* y por ultimo probar la ventana que se esta editando.

1.3 Opción de *Alineación* : Se aplica a grupos de objetos de interacción.

1.4 Opción de *Objetos* de Interacción, permite seleccionar un objeto de interacción para luego incorporarlo a la interface de la aplicación con un clic del mouse en la posición donde se desea colocarlo.

2. <u>Barra Gráfica</u> : provee la misma función que la opción *Objetos* de la Barra de Menú. El significado de los Iconos es el siguiente :

Static_Box (Cajas Estáticas) : Este objeto permite dibujar bordes, líneas, no tiene otra funcionalidad que la gráfica.

Static_Graphic (Gráfico Estático) : Este objeto fue creado especialmente para esta tesis, permite colocar sobre la ventana imágenes estáticas, muy útil para fondos.

Static-Text (Texto Estático) : se utiliza para poner frases estáticas sobre una Ventana, como pueden ser descripciones de objetos sobre la Pantalla, para las pantallas 'Acerca de .. ', en la que se pone información del software, etc. Permite darle formato al texto, est es, centrar justificar a derecha a izquierda.

Button (Botón) : Provee el servicio de selección, clickeando sobre este, se ejecuta una acción y puede tener una etiqueta que guíe al usuario sobre su funcionalidad.

 **Drawn-Button (Botón con Icono)** : Tiene la misma funcionalidad del anterior pero es más gráfico, la etiqueta se reemplaza por un gráfico que indique se función.

 **Check-Box** : Este provee al usuario de un conjunto de cajitas que cuando las clickea se dibuja una 'X' dentro, significando que esa condición está seleccionada (verdadera), de lo contrario no esta seleccionada (falso).

 **Three-State-Box** : Este aparenta y se comporta como n Check-Box, pero además tiene un estado intermedio, lo que significa que ese tópico no esta seleccionado (verdadero) , ni tampoco no seleccionado (falso), devolviendo vacío.

Radio-Button : Este objeto es un Botón circular con una etiqueta a su derecha. Cuando es seleccionado se dibuja un circulo sólido en su interior, devolviendo verdadero. Este difiere de los dos anteriores, porque este permite seleccionar 'solo una', entre varias posibilidades.

 **List-Box** : Provee la capacidad de una selección de entre una lista de items que están siempre visibles para el operador.

 **Multiple-List-Selection (Múltiples Selecciones)** : Su apariencia es igual al anterior, pero pueden seleccionarse varios items, entre una lista que está siempre visible al usuario. A medida que se van seleccionando quedan en color inverso para que el usuario tenga noción de su selección.

 **Combo-Box** : Este es un híbrido entre el EntryField y el List-Box, ya que consiste del primero que esta siempre visible y una lista desplegada debajo, disponible en todo momento. Tiene dos capacidades mas que el List-Box, y son que es capaz de captar cambios tipeados en el campo de entrada y que la lista puede o no estar visible.

 **Scroll-Bar (Barra de Desplazamiento Horizontal)** : Permiten desplazamiento horizontal en una pantalla donde no toda la información esta visible.

 **Scroll-Bar (Barra de Desplazamiento Vertical)** : Permiten desplazamiento vertical en una pantalla donde no toda la información esta visible.

 **Text-Pane** : Provee la función de edita texto en una ventana, además de dejarlo modificar y rolar si su visión no es completa. Es un pequeño editor de texto.

 **EntryField** : Provee la capacidad de editar una simple línea de texto.

 **Group-Box** : Es un rectángulo que permite poner un texto en la esquina superior izquierda, como un título de un grupo, ya que en su interior pueden colocarse diversos objetos.

 **Graph-Pane** : Es una ventana, en donde es posible dibujar gráficos, como con un lápiz, pegar imágenes, combinar colores, etc. Es un pequeño editor gráfico.

 **Animation-Pane** : un pane de este tipo contiene una cantidad de objetos animados y soporta movimientos a ellos.

3. <u>Área de Desarrollo de la interface</u> : En esta área se visualiza la interface, su tamaño puede modificarse clickeando sobre rectángulo negro de su extremo inferior derecho y su posición a través de la opción *Editar / Tamaño/ Posición* se dibujara una ventana transparente que puede ser desplazada a la posición deseada de la pantalla .

Para incorporar un objeto de interacción a la interface, se debe clickear sobre la barra gráfica el objeto de interacción deseado (o sobre la opción *Objetos* de la Barra de Menú) , luego el cursor

toma la forma de una cruz , situar el cursor en el área de desarrollo y desplazar el objeto a la posición deseada.

4. Botones de la interface de usuario

- **Asocia Help** : permite asociar un nombre de help creado con el Generador de Modelos de Usuarios (Patrón de Conocimiento).
- **Edita Código** : Permite editar el código fuente de la clase corrientemente editada.
- **Probar Ventana** : Permite dentro de la herramienta probar el prototipo de la interface de usuario.

5. Opciones para la definición y funcionalidad de los objetos de interacción

Se puede dividir en tres grupos :

5.1 *Identificación del Objeto de Interacción :* En esta sección se pueden definir :

- **Texto** : Generalmente se define la etiqueta del objeto dependiendo del tipo al que pertenezca el mismo.
- **Estilo** : Permite setear el estilo del Objeto de interacción desplegando una lista de los estilos soportados por el mismo.
- **Nombre** : Es el nombre del objeto de interacción con el cual se lo identifica unívocamente dentro de la aplicación. Debe ser completado obligatoriamente y debe ser único para una aplicación.

5.2 *Funcionalidad del Objeto de Interacción* : Permite asociar al objeto de interacción seleccionado acciones a seguir cuando se produce un evento determinado. Para ello se deben completar los siguientes campos :

- **Acción** : Es una lista de objetos seleccionables de eventos soportados por el objeto de interacción. En esta lista todos aquellos eventos que tienen un '*' delante del nombre del ítem tiene un método asociado.
- **Ejecuta** : Se debe especificar el nombre de un método que será ejecutado cuando el evento asociado sea disparado.

5.3 *Preferencias del Objeto de Interacción* : Permite definir todas aquellas características de apariencia del objeto de interacción seleccionado tales como:

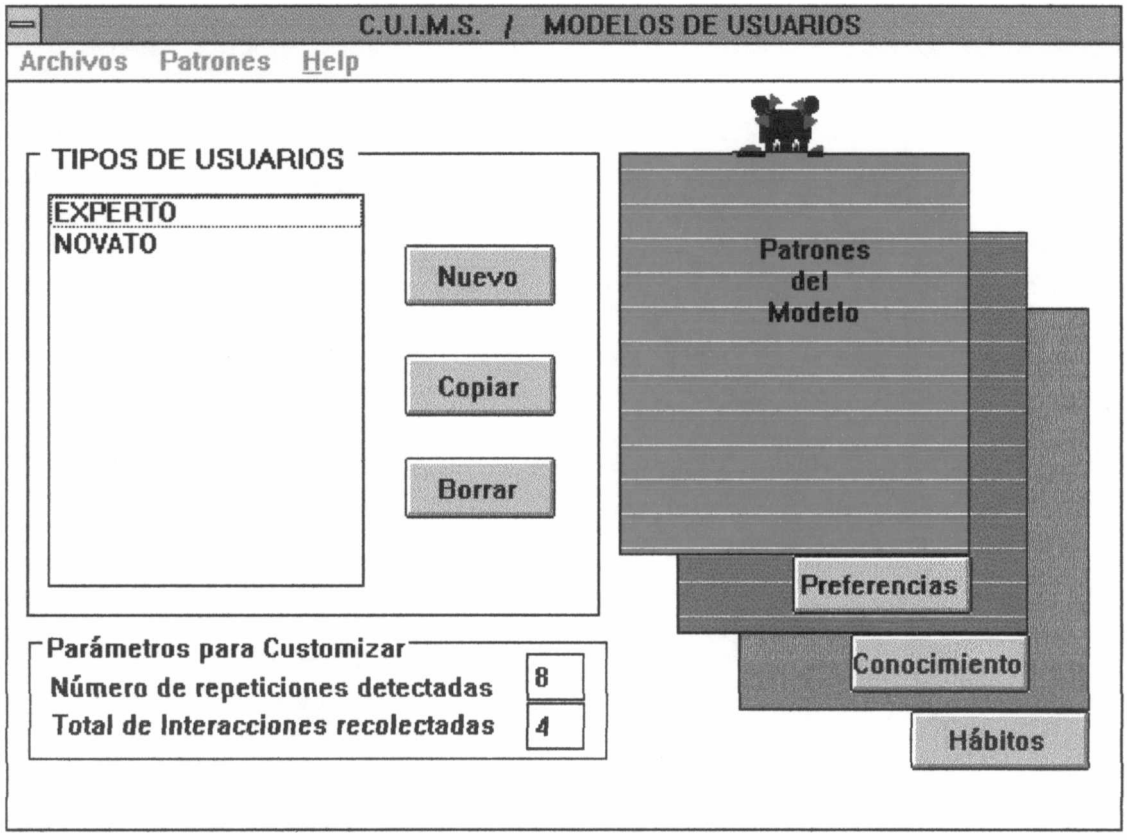
- **Letra** : Se desplegara una ventana que permitirá seleccionar el tipo de letra, estilo y tamaño (ver Apéndice de ventanas).
- **Iconos** : Se desplegara una ventana que permite seleccionar una imagen al objeto de interacción seleccionado. Este opción solo esta habilitada para los Bitmap Estáticos y Drawn Buttons. (ver Apéndice de ventanas)
- **Color** : Se desplegara una Ventana que permitirá seleccionar el color de frente y color de fondo del objeto seleccionado. (ver Apéndice de ventanas)
- **Menú / Menú Bar** : Permite asociar un menú al objeto seleccionado. Si el objeto es una Ventana entonces esta opción permite definir un menuBar para el mismo. Si el objeto no es una ventana entonces permite definir un menú estilo PopUp.
- **Otros**: Esta solamente habilitado cuando el objeto seleccionado es una ventana y permite definir el estilo que tendrá la misma (menú del sistema, minimiza, maximiza, titulo de la ventana, tamaño).

B. Diseñador de Modelos de Usuarios

Esta herramienta permite definir los Modelos de Usuarios, lo que implica la definición de los patrones de Preferencia y Conocimiento de los tipos de usuarios. Los patrones de Hábitos se infieren de las interacciones de los usuarios con las aplicaciones.

La ventana principal de esta herramienta esta compuesta de la siguiente manera:

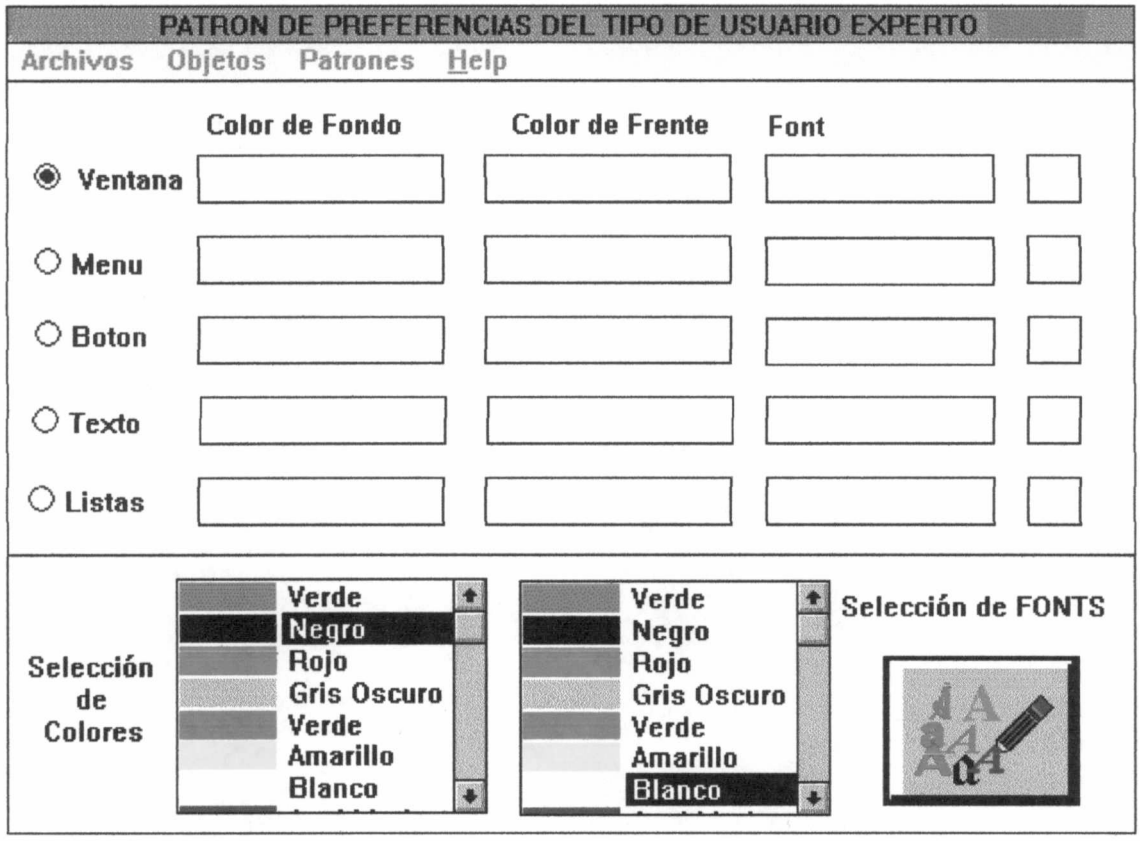
1. Barra de Menú.
2. Tipos de Usuarios.
3. Parámetros para Customizar.
4. Definición de los Patrones del Modelo de usuario.



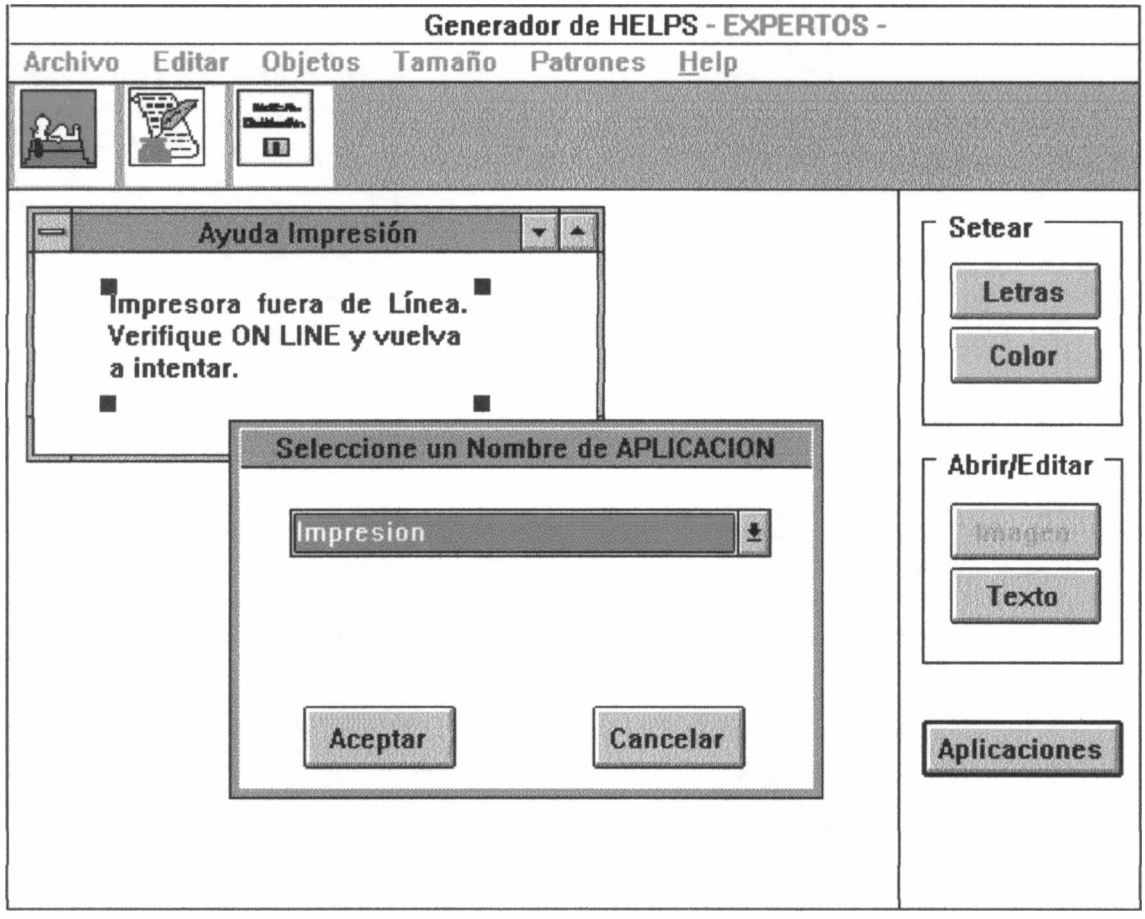
1. *Barras de Menú:* permite acceder a los patrones del Modelo seleccionado en la lista 'Tipos de Modelos'.
2. *Tipos de Modelos* : es una lista de los modelos definidos por el diseñandor del modelo permitiéndole a este la incorporación de un nuevo tipo de modelo (Nuevo) , copiar un tipo de modelo existente en otro (Copiar) o eliminar un tipo de modelo existente (Borrar).

3. *Parámetros para Customizacion* : Esto significa que si se recolectaron 8 peticiones de cambio sobre una característica de un mismo objeto de Interacción, y entre ellas se detecta 4 veces exactamente la misma peritación, entonces se realizara la adaptación solicitada.
4. *Definición de Patrones* : Cuando se presiona en los botones 'Preferencias' o 'Conocimiento' (o por selección de la opción Patrones de la Barra de Menú), se desplegara una ventana correspondiente al patrón seleccionado y al Modelo corrientemente elegido en 'Tipos de Modelo'.

Cuando selecciona para definir el patrón de Preferencias, se encontrara con una ventana con estas características :



Cuando seleccione para definir los Patrones de Conocimientos, la ventana tendrá las siguientes características :



En el titulo de esta ventana puede observase para que modelo de usuario pertenece el Help. Esta ventana esta compuesta de la siguiente manera :

1. <u>Barra de Menú</u> : Provee las siguientes opciones :

1.1 Opciones de *Archivo* (se aplican a la interface que esta siendo editada).

- Abrir Help : edita un help previamente definido.
- Salvar / Salvar como / Salvar por Default.

1.2 Opciones de *Edición* : Permite cortar/copiar/ texto

2. <u>Barra Gráfica</u> : provee la misma función que la opción *Objetos* de la Barra de Menú. El significado de los Iconos es el siguiente :

 **Static_Graphic (Gráfico Estático)** : Este objeto fue creado especialmente para esta tesis, permite colocar sobre la ventana imágenes estáticas, muy útil para fondos.

 **Text-Pane** : Provee la función de edita texto en una ventana, además de dejarlo modificar y rolar si su visión no es completa. Es un pequeño editor de texto.

 **Static-Text (Texto Estático)** : se utiliza para poner frases estáticas sobre una Ventana. Permite darle formato al texto, est es, centrar justificar a derecha a izquierda.

3. Botones definir preferencias de Objetos

- **Letra** : Se desplegara una ventana que permitirá seleccionar el tipo de letra, estilo y tamaño (ver Apéndice de ventanas).
- **Color** : Se desplegara una Ventana que permitirá seleccionar el color de frente y color de fondo del objeto seleccionado. (ver Apéndice de ventanas

4. Botones para Insertar

- **Texto** : Permite insertar una archivo de Texto.
- **Imagen** : Se desplegara una ventana que permite seleccionar una imagen al objeto de interacción seleccionado.

5. Botones para Enganchar con la Aplicación

- **Aplicaciones** : Cuando clickea le aparecerá la ventana que se ve en la figura y su funcionalidad es la siguiente.

¿ Cómo se conecta un Help a una Aplicación ?

Supongamos que usted (como diseñador de UI), ha generado una Interface de Aplicación y le dio el nombre **'Impresion'**. Ahora desea crear los helps para esa aplicación, y debido a que posee distintos modelos de usuarios -supongamos novato y experto-, tales tendrán distintas características. Existe una sola restricción y es que todos los helps que corresponden a una misma aplicación, aun cuando sean para distintos modelos deben tener el mismo nombre. Supongamos entonces como en la figura que se esta generando un help para el modelo **Experto**

y cuando salvamos lo llamamos '**HelpImpre**'. Ahora supongamos que queremos crear el help para el modelo de usuario **Novato**. Tenemos 2 opciones desde el menub Bar :

- **Archivos/Abrir Help** y abrir el help creado para el modelo Experto, ya que si bien tendrá distintas característica y apariencia, es muy probable que tengan similitudes y ayude en la creación.
- **Archivos/Crear Help** y esto implicaría crear un help nuevo en su totalidad.

En ambos casos, cuando salva el help generado debe colocar el mismo nombre del anterior modelo, es decir '**HelpImpre**'.

Por lo tanto, hasta ahora tenemos :

1 Aplicación : Impresión.

2 Modelos : Experto y Novato.

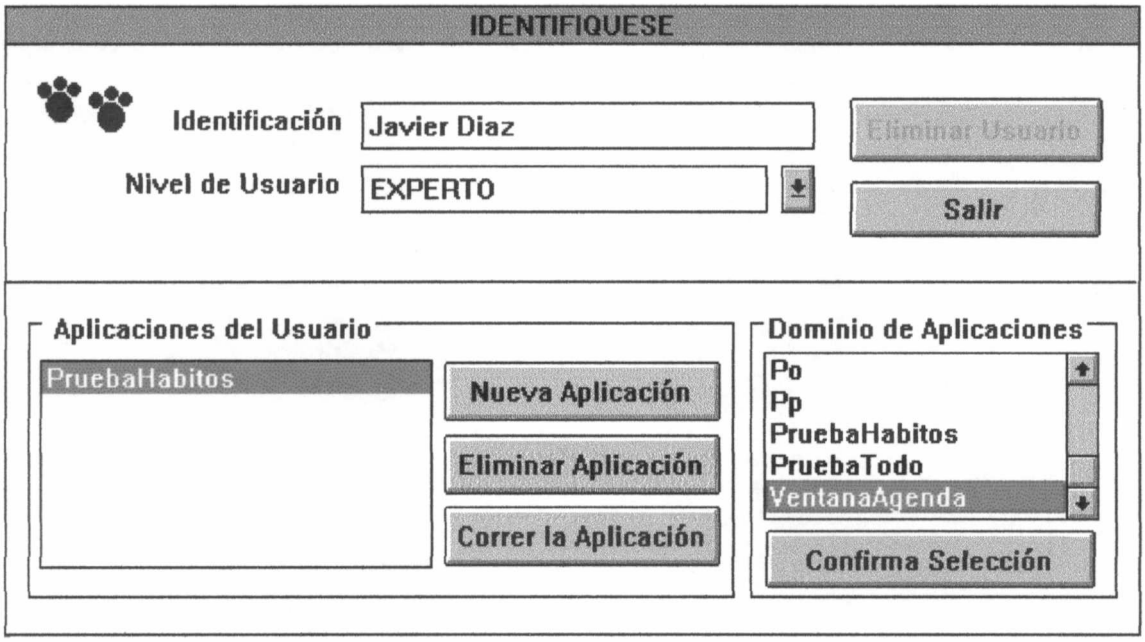
2 Helps con el mismo nombre : HelpImpre (distintos y pueden editarse si ingresa con el modelo correspondiente).

Por último para que cuando el usuario final corra la aplicación tome el help correspondiente, solo debe tipear editando el código desde el generador de interfaces, lo siguiente :

```
nom_metodo: aPane
    self abrirHelp: Pane.
```

C. Usuario Final

Esta ventana le permite identificarse como usuario para poder correr su aplicación :



Si es la primer vez que ingresa al sistema debe seleccionar un tipo de usuario al que quiere pertenecer, optando por una de las opciones de la lista presentada en el campo nivel de usuario, de lo contrario al identificarse automáticamente toma el tipo de usuario y su conjunto de aplicaciones.

Aplicaciones del Usuario, son aquellas que están asignadas al perfil de un usuario, por consiguiente pueden haberse corrido y recolectado su interacción, mientras que el **Dominio de Aplicaciones** es el conjunto de todas las aplicaciones disponibles para los usuarios. Para incorporar una nueva aplicación debe clickearse el botón 'Nueva Aplicación' y seleccionar una de la lista. Para correr una aplicación, debe seleccionar una aplicación de la lista Aplicaciones del Usuario y presionar el botón 'Correr la Aplicación'.

Instalación y Requerimientos

Coloque el diskette #1 y ejecute INSTALAR. La instalación le irá pidiendo los sucesivos diskettes.

Para poder correr la herramienta se necesita una computadora 80386 o superiores, WINDOWS 3.1 y 4 MegaBytes de memoria (recomendamos 8 para lograr buena performance).

Información útil de este producto

Para obtener ayuda de lo que desee, pulse F1 y un help estilo WINDOWS, lo asistirá de inmediato.

Bibligrafías

Bibligrafía Básica

[BACO 91] Len Bass. Software Engineering Institute. Joelle Coutaz. University Of Grenoble(IMAG-LGI). 'Developing Software for the User Interface'. ADDISON-WESLEY PUBLISHING COMPANY.

[CHI 90] David Benyon, Frances Jenning. PACIS, research Group, Computer Departament, Open University, and Dianne. Departament Business Computing, The City University, Northampton , LONDON. 'An Adaptive System Developer's Tool-Kit.' [CHI 90]

[JSST 91] Joseph W. Sullivan and Sherman W. Tyler. 'Intelligent User Interfaces'. ADDISON-WESLEY PUPLISHING COMPANY.

[LER 89] Diana Lerner. School of Computer Science. Carnegie Mellon University PittsBurgh. 'Automated Customization of User Interfaces'.

[LOW 91] Jonas Lowgren. Departament of Computer an Information Science Linkoping University. 'Knowledge_Based Disign Support and Discourse Management in User Interface Management Systems.'

[STE 91] Steve Ankuo Chien. 'An Explication-Based Learning Approach to Incremental Planning'.

[CHI 90] 'An Experiment in Interactive Architecture', Ernest Edmonds, LUTCHI Research Centre, Noriko Hagiwara, NEC Coporation. Human Computer Interface.

Bibligrafia Consultada

[OLSEN 86] Dan R. Olsen JR. Bragham Young University. 'MIKE:The Menu Interaction Kontrol Enviroment.'

[NAN 89] JOOP. Variatons of Model-View-Controller' y 'Why Object-Oriented User Interface Toolkits Are Better?', Nancy T. Knolle.

[BEN 92] Ben Shneiderman. 'Designing the User Intefaces'. Strategies for Effective Human-Computer Interaction.

[RBL 90] Rebecca Wirfs-Brock, Btian Wilkerson, Lauren Wiener. 'Designing Object-Oriented Software.

[SMK 91] Smalltalk/V Windows. Tutorial and Programming Handbook. DIGITALK.

[COM 92] 'Demonstrational Interfaces : A Sptep Beyond Direct manipulation', Brad Myers. Carnegie Mellon University.

[CHI 90] 'An Object-Oriented UIMS for Rapid Prototiping', Yen Ping Shan. Human Computer Interaction - INTRACT 90.

[CHI 90] 'HCI Seen from the Perspective of Software Developers', Jonh Bennett, peter [Conklin, Karmen Guevara, Wendy Mmackay, Tom Sancha. Human Computer Interaction - INTRACT' 90.

[CHI 90] 'New Approaches to Theory in HCI: How Should we judge their Aceptability ?', Andrew Monk. Human Computer Interaction - INTRACT' 90.

[CHI 90] 'Two ways to fill a bath, with and without knowing It', Anne Ankrah, David Frohlich, G.Nigel Gilbert. Human Computer Interaction - INTRACT 90.

[HFG 90] 'Tradicional Dialogue Design Applied to Model User Interfaces'. Human Factors, Graphicals and Multimedia Applications.

[COML 93] 'Looking at the world through cheap sunglasses', Thomas Murphy. Computer Lenguage.

[CHI 90] 'Scenarioo : a new generations UIMS', Brigitte Roudaud Valerie Lavigne, Oliver Iagneau, Earl Minor. Human Computer Interaction - INTRACT 90.

[CHIEN 90] 'AN EXPLICATION-BASED LEARNING APPROACH TO INCREMENTAL PLANNING', Steve Ankuo Chien. University of Illinois.

[VAR 90] 'Variaciones de la Programación'. Isabel Medwid, Gabriela Demo. UNLP.

[UIMS 90] 'User Interface Management System'. Osvaldo Rosenfeld, pablo Safar. Universidad Nacional de LaPlata.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

DONACION ..... T 65
$ ..... 0 % 16
Fecha 19-8-05
Inv. E 1932