

Diseño de Interfaces de Usuario

Principios, Prototipos y Heurísticas para Evaluación

Leopoldo Sebastián M. Gómez
gomezsebastian@yahoo.com

Abstract

El diseño de interfaces de usuario es una tarea que ha adquirido relevancia en el desarrollo de un sistema. La calidad de la interfaz de usuario puede ser uno de los motivos que conduzca a un sistema al éxito o al fracaso. Los principios que se presentan son de utilidad para creación de interfaces funcionales y de fácil operación. A pesar de no ser capaces de resolver todos los aspectos propios del contexto con el que se esté trabajando, pueden ser combinados con la prototipación y la aplicación de heurísticas de evaluación para facilitar el proceso de diseño. El presente artículo se centra en los componentes de software de las interfaces de usuario, quedando fuera del alcance de mismo otros aspectos, como hardware y documentación. Lo anteriormente expuesto se complementa con un caso práctico de diseño de interfaces de usuario, producto de realizar la actividad de “Definición de Interfaces de Usuario” (EFS 4) de la metodología Métrica Versión 2.

Key words: evaluation, heuristics, principles, prototypes, user interfaces.

1. Conceptos Generales

La Interfaz de Usuario, en adelante IU, de un programa es un conjunto de elementos *hardware* y *software* de una computadora que presentan información al usuario y le permiten interactuar con la información y con el computadora. También se puede considerar parte de la IU la documentación (manuales, ayuda, referencia, tutoriales) que acompaña al hardware y al software.

Si la IU está bien diseñada, el usuario encontrará la respuesta que espera a su acción. Si no es así puede ser frustrante su operación, ya que el usuario habitualmente tiende a culparse a sí mismo por no saber usar el objeto.

Los programas son usados por usuarios con distintos niveles de conocimientos, desde principiantes hasta expertos. Es por ello que no existe una interfaz válida para todos los usuarios y todas las tareas. Debe permitirse libertad al usuario para que elija el modo de interacción que más se adecúe a sus objetivos en cada momento. La mayoría de los programas y sistemas operativos ofrecen varias formas de interacción al usuario.

Existen tres puntos de vista distintos en una IU: el del usuario, el del programador y el del diseñador (analogía de la construcción de una casa). Cada uno tiene un modelo mental propio de la interfaz, que contiene los conceptos y expectativas acerca de la misma, desarrollados a través de su experiencia.

El modelo permite explicar o predecir comportamientos del sistema y tomar las decisiones adecuadas para modificar el mismo. Los modelos subyacen en la interacción con las computadoras, de ahí su importancia.

Modelo del usuario: El usuario tiene su visión personal del sistema, y espera que éste se comporte de una cierta forma. Se puede conocer el modelo del usuario estudiándolo, ya sea realizando tests de usabilidad, entrevistas, o a través de una realimentación. Una interfaz debe facilitar el proceso de crear un modelo mental efectivo.

Para ello son de gran utilidad las metáforas, que asocian un dominio nuevo a uno ya conocido por el usuario. Un ejemplo típico es la metáfora del escritorio, común a la mayoría de las interfaces gráficas actuales.

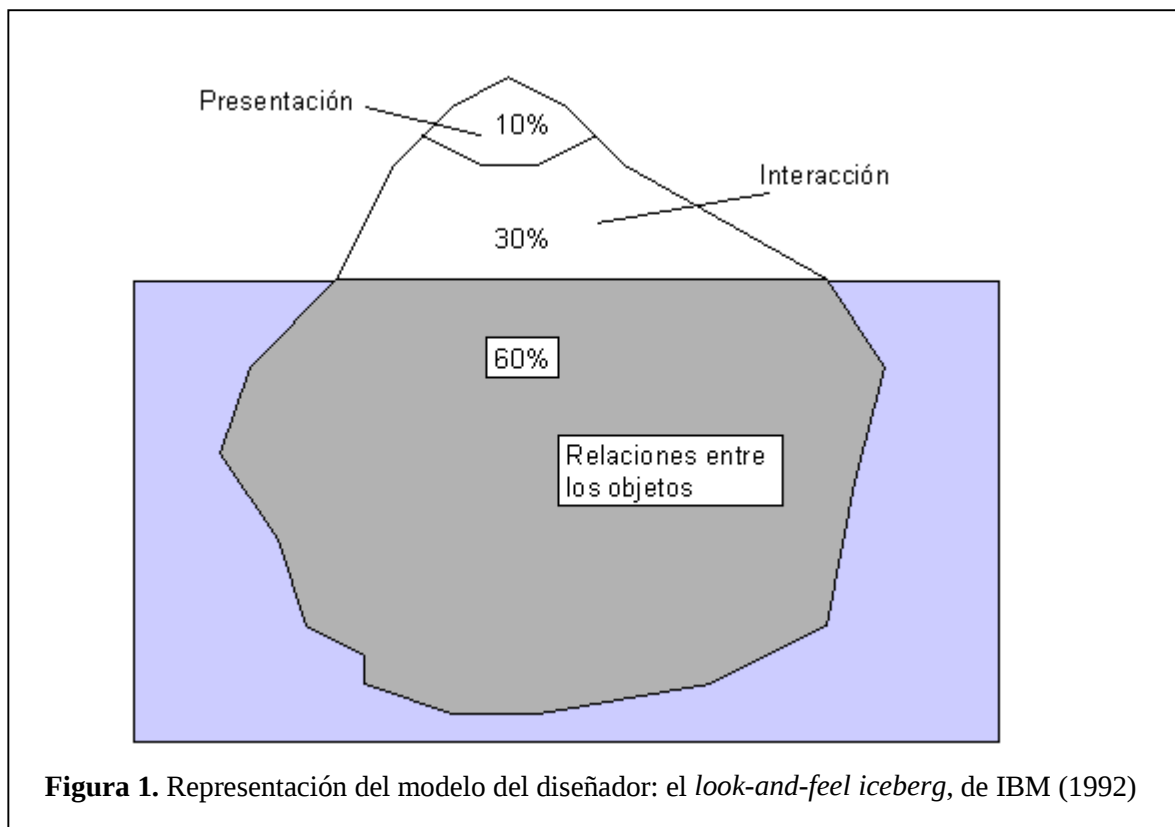
Modelo del diseñador: El diseñador mezcla las necesidades, ideas, deseos del usuario y los materiales de que dispone el programador para diseñar un producto de *software*. Es un intermediario entre ambos.

El modelo del diseñador describe los objetos que utiliza el usuario, su presentación al mismo y las técnicas de interacción para su manipulación. Consta de tres partes: presentación, interacción y relaciones entre los objetos (Figura 1).

La *presentación* es lo que primero capta la atención del usuario, pero más tarde pasa a un segundo plano, y adquiere más importancia la *interacción* con el producto para poder satisfacer sus expectativas. La presentación no es lo más relevante y un abuso en la misma (por ejemplo, en el color) puede ser contraproducente, distrayendo al usuario.

La segunda parte del modelo define las técnicas de *interacción* del usuario, a través de diversos dispositivos.

La tercera es la más importante, y es donde el diseñador determina la *metáfora* adecuada que encaja con el modelo mental del usuario. El modelo debe comenzar por esta parte e ir hacia arriba. Una vez definida la metáfora y los objetos del interfaz, los aspectos visuales saldrán de una manera lógica y fácil.



Estos modelos deben estar claros para los participantes en el desarrollo de un producto, de forma que se consiga una interfaz atractiva y a la vez efectiva para el trabajo con el programa.

Una interfaz no es simplemente una cara bonita; esto puede impresionar a primera vista pero decepcionar a la larga. Lo importante es que el programa se adapte bien al modelo del usuario, cosa que se puede comprobar utilizando el programa más allá de la primera impresión.

Modelo del programador: Es el más fácil de visualizar, al poderse especificar formalmente. Está constituido por los objetos que manipula el programador, distintos de los que trata el usuario (ejemplo: el programador llama base de datos a lo que el usuario podría llamar agenda). Estos objetos deben esconderse del usuario.

Los conocimientos del programador incluyen la plataforma de desarrollo, el sistema operativo, las herramientas de desarrollo y especificaciones. Sin embargo, esto no significa necesariamente que tenga la habilidad de proporcionar al usuario los modelos y metáforas más adecuadas. Muchos no consideran el modelo del usuario del programa, y sí sus propias expectativas acerca de cómo trabajar con la computadora.

2. Principios para el Diseño de Interfaces de Usuario

Existen principios relevantes para el diseño e implementación de IU, ya sea para las IU gráficas, como para la Web.

Anticipación

Las aplicaciones deberían intentar anticiparse a las necesidades del usuario y no esperar a que el usuario tenga que buscar la información, recopilarla o invocar las herramientas que va a utilizar.

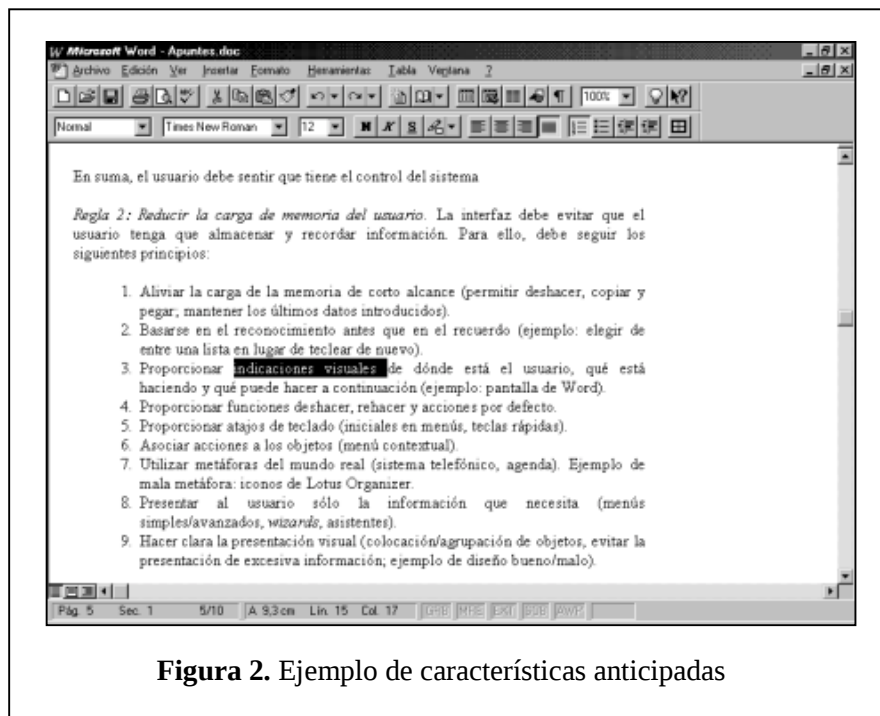


Figura 2. Ejemplo de características anticipadas

En la Figura 2 se ilustra como el procesador de texto se anticipa a las necesidades del usuario, proporcionando las características del texto seleccionado -fuente, tamaño, alineación, etc.- permitiendo que el usuario pueda modificarlas ágilmente.

Autonomía

La computadora, la IU y el entorno de trabajo deben estar a disposición del usuario. Se debe dar al usuario el ambiente flexible para que pueda aprender rápidamente a usar la aplicación. Sin

embargo, está comprobado que el entorno de trabajo debe tener ciertas cotas, es decir, ser explorable pero no azaroso.

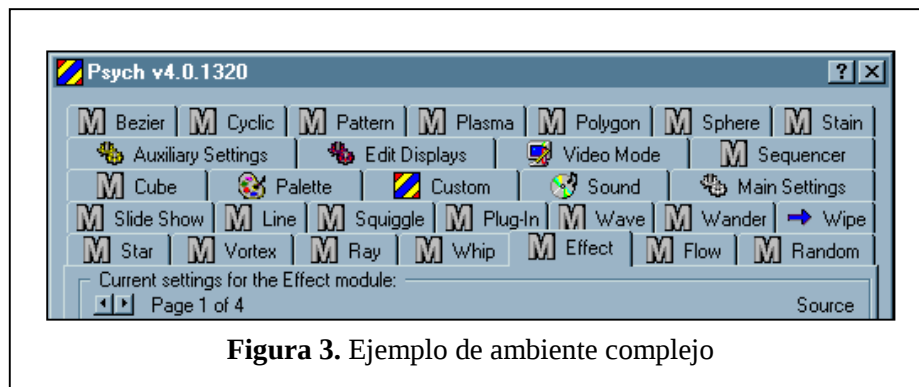


Figura 3. Ejemplo de ambiente complejo

En la Figura 3 se visualiza un diseño incorrecto de interfaz de usuario. La cantidad de opciones propuestas propone un grado de complejidad que no permite que el usuario pueda aprender a utilizar el sistema en forma progresiva.

Es importante utilizar mecanismos indicadores de estado del sistema que mantengan a los usuarios alertas e informados. No puede existir autonomía en ausencia de control, y el control no puede ser ejercido sin información suficiente. Además, se debe mantener información del estado del sistema en ubicaciones fáciles de visualizar.

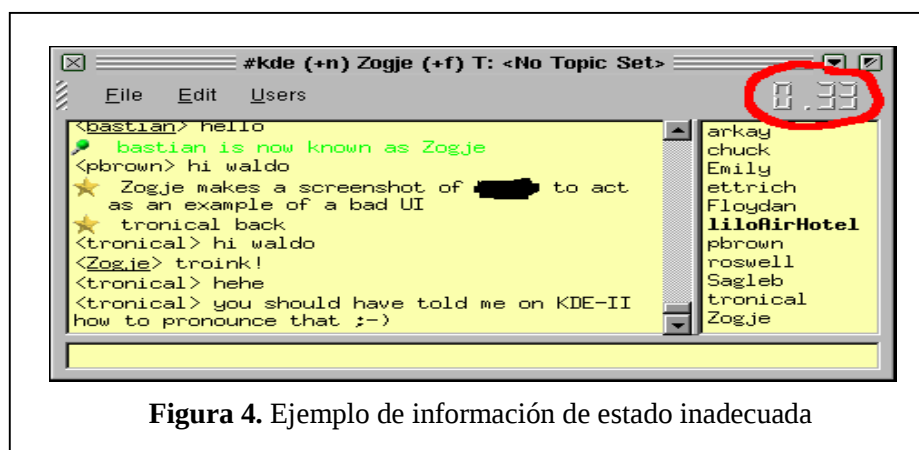


Figura 4. Ejemplo de información de estado inadecuada

En la Figura 4 se ejemplifica una incorrecta disposición de componentes en la IU. El reloj no debe ser incorporado en el menú del sistema ya que aporta confusión al usuario. Para mantenerlo informado sería mas adecuado colocarlo en la barra de estado del sistema.

Percepción del Color

Aunque se utilicen convenciones de color en la IU, se deberían usar otros mecanismos secundarios para proveer la información a aquellos usuarios con problemas en la visualización de colores

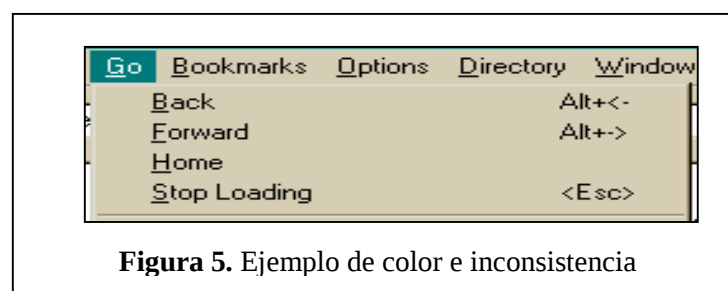


Figura 5. Ejemplo de color e inconsistencia

En la Figura 5 se representa un mecanismo secundario muy utilizado para ejecución de comandos: los comandos abreviados (shortcut-keys). Sin embargo la aplicación presenta un problema de inconsistencia ya que define combinaciones de teclas que difieren a lo esperado por el usuario, por ejemplo Alt+< en lugar de Alt+B.

Valores por Defecto

No se debe utilizar la palabra “Defecto” en una aplicación o servicio. Puede ser reemplazada por “Estándar” o “Definida por el Usuario”, “Restaurar Valores Iniciales” o algún otro término específico que describa lo que está sucediendo. Los valores por defecto deberían ser opciones inteligentes y sensatas. Además, los mismos tienen que ser fáciles de modificar.

Consistencia

Para lograr una mayor consistencia en la IU se requiere profundizar en diferentes aspectos que están catalogados en niveles. Se realiza un ordenamiento de mayor a menor consistencia:

1. *Interpretación del comportamiento del usuario:* la IU debe comprender el significado que le atribuye un usuario a cada requerimiento. Ejemplo: mantener el significado de las los comandos abreviados (shortcut-keys) definidos por el usuario.
2. *Estructuras invisibles:* se requiere una definición clara de las mismas, ya que sino el usuario nunca podría llegar a descubrir su uso. Ejemplo: la ampliación de ventanas mediante la extensión de sus bordes.
3. *Pequeñas estructuras visibles:* se puede establecer un conjunto de objetos visibles capaces de ser controlados por el usuario, que permitan ahorrar tiempo en la ejecución de tareas específicas. Ejemplo: ícono y/o botón para impresión.
4. *Una sola aplicación o servicio:* la IU permite visualizar a la aplicación o servicio utilizado como un componente único. Ejemplo: La IU despliega un único menú, pudiendo además acceder al mismo mediante comandos abreviados.
5. *Un conjunto de aplicaciones o servicios:* la IU visualiza a la aplicación o servicio utilizado como un conjunto de componentes. Ejemplo: La IU se presenta como un conjunto de barras de comandos desplegadas en diferentes lugares de la pantalla, pudiendo ser desactivadas en forma independiente.
6. *Consistencia del ambiente:* la IU se mantiene en concordancia con el ambiente de trabajo. Ejemplo: La IU utiliza objetos de control como menús, botones de comandos de manera análoga a otras IU que se usen en el ambiente de trabajo.
7. *Consistencia de la plataforma:* La IU es concordante con la plataforma. Ejemplo: La IU tiene un esquema basado en ventanas, el cual es acorde al manejo del sistema operativo Windows.

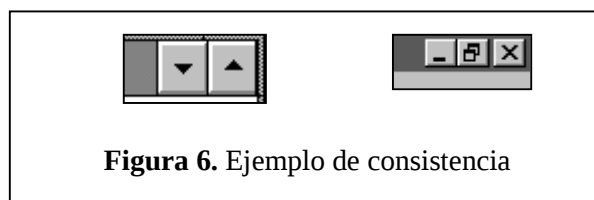


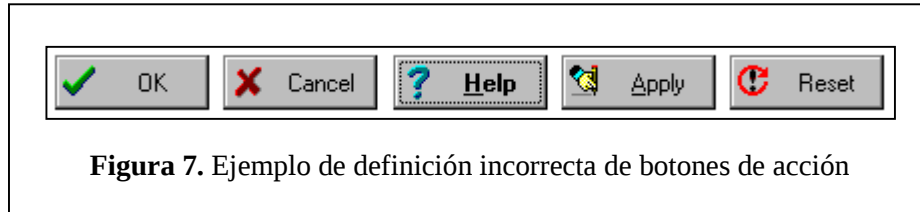
Figura 6. Ejemplo de consistencia

En la Figura 6 puede observarse la mejora en la consistencia de las *pequeñas estructuras visibles* (3.) para los sistemas gráficos basados en ventanas. La inclusión de la opción  para cerrar la ventana –operación comunmente utilizada en estas aplicaciones- simplifica la operatividad del mismo.

La inconsistencia en el comportamiento de componentes de la IU debe ser fácil de visualizar. Se debe evitar la uniformidad en los componentes de la IU. Los objetos deben ser consistentes con su comportamiento. Si dos objetos actúan en forma diferente, deben lucir diferentes. La única forma de verificar si la IU satisface las expectativas del usuario es mediante testeo.

Eficiencia del Usuario

Se debe considerar la productividad del usuario antes que la productividad de la máquina. Si el usuario debe esperar la respuesta del sistema por un período prolongado, estas pérdidas de tiempo se pueden convertir en pérdidas económicas para la organización. Los mensajes de ayuda deben ser sencillos y proveer respuestas a los problemas. Los menús y etiquetas de botones deberían tener las palabras claves del proceso.



En la Figura 7 se demuestra como una incorrecta definición de las palabras clave de las etiquetas de los botones de comando puede confundir al usuario. Los botones **OK** y **Apply** aparentan realizar el mismo proceso. Esto puede solucionarse suprimiendo uno de ellos si realizan la misma tarea o etiquetándolos con los nombres de los procesos específicos que ejecutan.

Ley de Fitt

El tiempo para alcanzar un objetivo es una función de la distancia y tamaño del objetivo. Es por ello, que es conveniente usar objetos grandes para las funciones importantes.

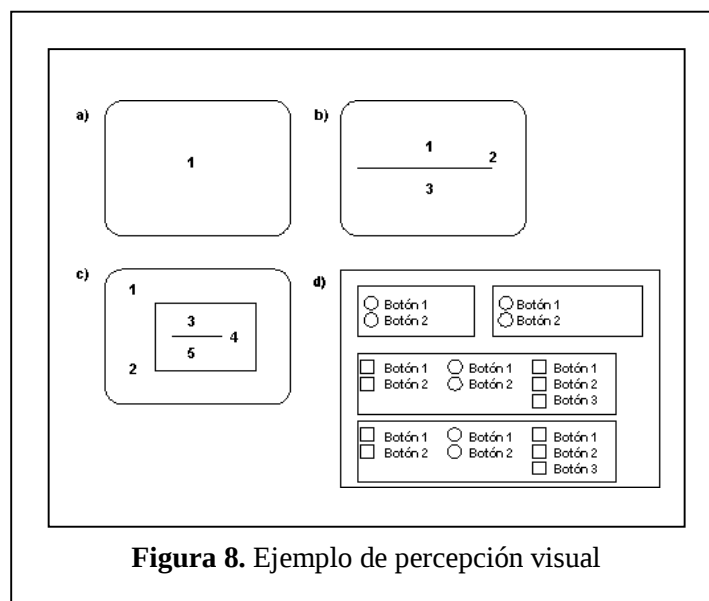


Figura 8. Ejemplo de percepción visual

En la Figura 8 se puede apreciar la relación entre los elementos de diseño de pantalla y su percepción visual. El número de elementos visuales que perciben son: en el caso a) 1 (el fondo); en b) 3 (la línea, lo que está encima y lo que está debajo); en c) son 5 (el espacio fuera del recuadro, el recuadro, la línea y el espacio encima y debajo de ésta); finalmente, en d) el número se eleva a 35, siguiendo el mismo criterio. Conclusión: cada elemento nuevo que se añade influye más de lo que se piensa en el usuario.

Interfaces Explorables

Siempre que sea posible se debe permitir que el usuario pueda salir ágilmente de la IU, dejando una marca del estado de avance de su trabajo, para que pueda continuarlo en otra oportunidad.

Para aquellos usuarios que sean noveles en el uso de la aplicación, se deberá proveer de guías para realizar tareas que no sean habituales.

Es conveniente que el usuario pueda incorporar elementos visuales estables que permitan, no solamente un desplazamiento rápido a ciertos puntos del trabajo que esté realizando, sino también un sentido de “casa” o punto de partida.

La IU debe poder realizar la inversa de cualquier acción que pueda llegar a ser de riesgo, de esta forma se apoya al usuario a explorar el sistema sin temores.

Siempre se debe contar con un comando “Deshacer”. Este suprimirá la necesidad de tener que contar con diálogos de confirmación para cada acción que realice en sistema.

El usuario debe sentirse seguro de poder salir del sistema cuando lo desee. Es por ello que la IU debe tener un objeto fácil de accionar con el cual poder finalizar la aplicación.

Objetos de Interfaz Humana

Los objetos de interfaz humana no son necesariamente los objetos que se encuentran en los sistemas orientados a objetos. Estos pueden ser vistos, escuchados, tocados o percibidos de alguna forma. Además, estos objetos deberían ser entendibles, consistentes y estables.

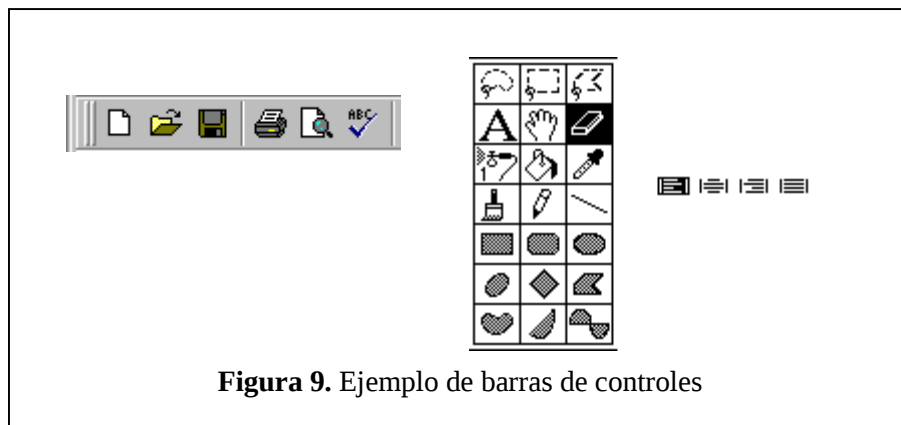


Figura 9. Ejemplo de barras de controles

En la Figura 9 se presentan barras de controles que simplifican la operación de un sistema. A través de las ilustraciones que poseen los mismos, el usuario puede aprender fácilmente su uso. Si se mantienen para estos botones las mismas asignaciones de procesos en diferentes sistemas, la comprensión del funcionamiento de los mismos se hace más sencilla.

Uso de Metáforas

Las buenas metáforas crean figuras mentales fáciles de recordar. La IU puede contener objetos asociados al modelo conceptual en forma visual, con sonido u otra característica perceptible por el usuario que ayude a simplificar el uso del sistema.

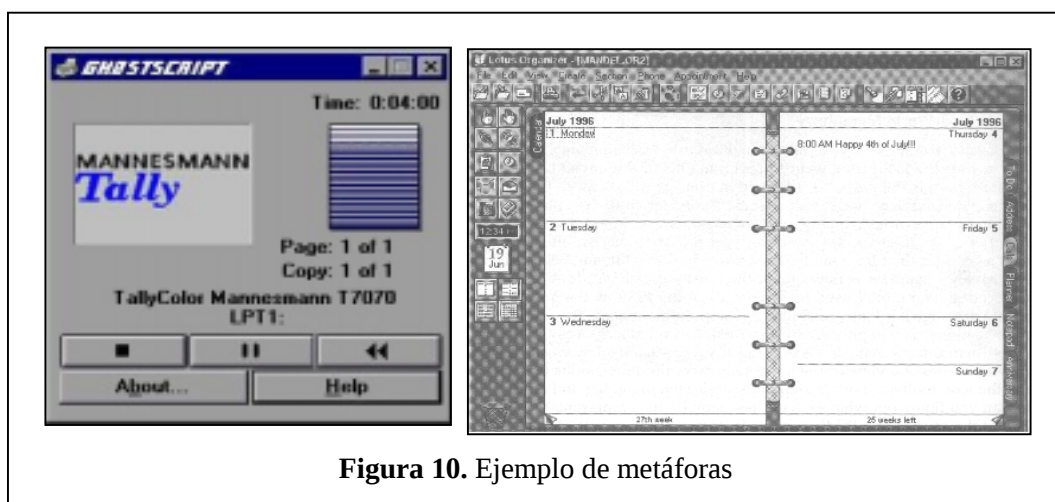


Figura 10. Ejemplo de metáforas

En la Figura 10 se compara la aplicación de metáforas en el desarrollo de una IU. En el primer caso, se utiliza incorrectamente la metáfora de una cámara de video para representar el procesamiento de un documento por una impresora. Se puede observar que el botón  carece de sentido, ya que no se puede volver atrás un trabajo que ya ha sido impreso. En el segundo caso, la metáfora de la agenda es utilizada correctamente para la implementación de una agenda electrónica.

Curva de Aprendizaje

El aprendizaje de un producto y su usabilidad no son mutuamente excluyentes. El ideal es que la curva de aprendizaje sea nula, y que el usuario principiante pueda alcanzar el dominio total de la aplicación sin esfuerzo.

Reducción de Latencia

Siempre que sea posible, el uso de tramas (multi-threading) permite colocar la latencia en segundo plano (background). Las técnicas de trabajo multitarea posibilitan el trabajo ininterrumpido del usuario, realizando las tareas de transmisión y computación de datos en segundo plano.

Protección del Trabajo

Se debe poder asegurar que el usuario nunca pierda su trabajo, ya sea por error de su parte, problemas de transmisión de datos, de energía, o alguna otra razón inevitable.

Auditoría del Sistema

La mayoría de los navegadores de Internet (browsers), no mantienen información acerca de la situación del usuario en el entorno, pero para cualquier aplicación es conveniente conocer un conjunto de características tales como: hora de acceso al sistema, ubicación del usuario en el sistema y lugares a los que ha accedido, entre otros. Además, el usuario debería poder salir del sistema y al volver a ingresar continuar trabajando en lugar dónde había dejado.

Legibilidad

Para que la IU favorezca la usabilidad del sistema de software, la información que se exhiba en ella debe ser fácil de ubicar y leer. Para lograr obtener este resultado se deben tener en cuenta algunas como: el texto que aparezca en la IU debería tener un alto contraste, se debe utilizar combinaciones de colores como el texto en negro sobre fondo blanco o amarillo suave. El tamaño de las fuentes tiene que ser lo suficientemente grande como para poder ser leído en monitores estándar. Es importante hacer clara la presentación visual (colocación/agrupación de objetos, evitar la presentación de excesiva información.

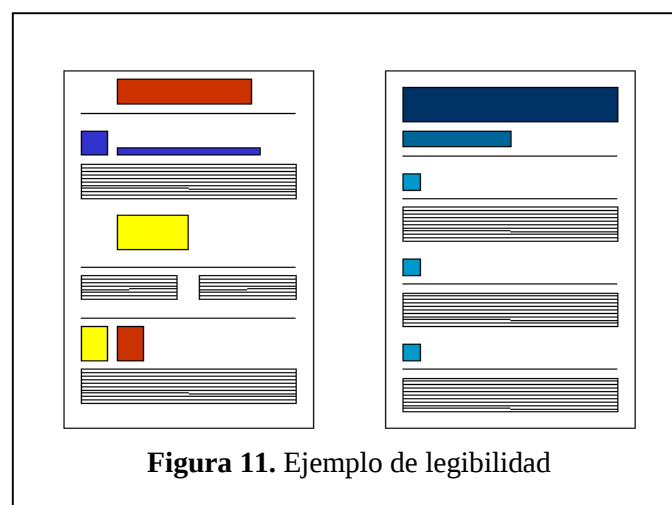


Figura 11. Ejemplo de legibilidad

En la Figura 11 se describe una comparación de disposición de los objetos en pantalla. La figura de la izquierda, combina una disposición asimétrica de la información con un conjunto de colores que no facilita la lectura. La figura de la derecha realiza la presentación de la información utilizando una gama de colores homogénea y una alineación del texto que favorece a la legibilidad del mismo.

Interfaces Visibles

El uso de Internet, ha favorecido la implementación de interfaces invisibles. Esto significa que el usuario siempre ve una página específica, pero nunca puede conocer la totalidad del espacio de páginas de Internet. La navegación en las aplicaciones debe ser reducida a la mínima expresión. El usuario debe sentir que se mantiene en un único lugar y que el que va variando es su trabajo. Esto no solamente elimina la necesidad de mantener mapas u otras ayudas de navegación, sino que además brindan al usuario una sensación de autonomía.

3. Utilización de Prototipos en la Implementación de IU

Niveles de Prototipado

Se puede hacer una clasificación de los principales tipos de prototipos, variando su grado de complejidad, de acuerdo a las características que consideren y a su operabilidad para realizar simulaciones.

- *Prototipos Estáticos:* son aquellos que no permiten la alteración de sus componentes, pero sirven para identificar y resolver problemas de diseño. En esta categoría se incluyen las presentaciones sobre reproductores, papel u otro medio de visualización.
- *Prototipos Dinámicos:* permiten la evaluación de un modelo del sistema sobre una estación de trabajo o una terminal. Estos prototipos involucran aspectos de diseño mas detallados que los prototipos estáticos, incluyendo la validación del diseño del sistema en términos de requerimientos no funcionales, por ejemplo de performance.
- *Prototipos Robustos:* deben ser relativamente completos en la simulación de las características dinámicas de la interfaz (presentación de mensajes de error, entrada y edición de datos, etc.). Esta categoría puede ser utilizada para validar los objetivos de diseño.

El nivel de sofisticación del prototipo debería incrementarse a lo largo del proceso de diseño de interfaces de usuario. La información recolectada durante las tareas de análisis del sistema y la especificación de los requisitos del usuario constituyen los datos clave para el proceso de prototipación.

4. Heurísticas para la Evaluación de IU

Las heurísticas ayudan a poder analizar las IU y localizar problemas que afecten la utilización de las mismas.

Algunas pautas para evaluar una IU son:

- Visibilidad del estado del sistema
- Semejanza del sistema al mundo real
- Control y libertad por parte del usuario
- Consistencia y estandarización
- Prevención de Errores
- Reconocimiento de acciones y opciones
- Flexibilidad y eficiencia en el uso
- Estética y diseño minimalista
- Reconocimiento de errores, diagnóstico y recuperación
- Ayuda y documentación

Para establecer medidas que indiquen la severidad de los problemas en el uso de las interfaces, se deben conocer los factores que determinan el grado de un problema:

- La frecuencia de ocurrencia.
- El impacto que causa la ocurrencia del problema.
- La persistencia del problema.
- El impacto en el mercado.

Medidas de severidad de un problema en la IU:

- 0: No puede llegar a considerarse un problema.
- 1: Es un problema “cosmético” que no necesita ser corregido a menos que se disponga tiempo extra en el proyecto.
- 2: Es un problema menor y su corrección puede tener baja prioridad.
- 3: Es un problema mayor y su corrección debería tener alta prioridad.
- 4: Es una catástrofe para la utilización de la aplicación y es imperativo corregir el error.

Para la evaluación de los problemas en las IU es conveniente que contar con mas de un evaluador; de esta forma los resultados son mas confiables.

5. Caso Práctico

Para complementar los aspectos teóricos, se realiza una ejemplificación de una de las actividades propuestas por la metodología Métrica Versión 2 para el diseño de interfaces de usuario.

La metodología Métrica Versión 2 contempla la simulación de diálogos de pantalla para la actividad EFS 4 “Definir Interfaces de Usuario”, a partir de las principales funciones interactivas, eventos y consultas identificados en la fase de análisis.

Seguindo la metodología, la tarea 4.1 prescribe las siguientes acciones:

- Definir los formatos individuales de las pantallas utilizadas.
- Describir de modo detallado los diálogos entre pantallas y el encadenamiento entre las mismas.
- Determinar los diálogos que se consideran críticos.
- Realizar una maqueta dinámica.

Asimismo, la tarea 4.2 indica:

- Obtener una definición de los formatos de los informes generados.
- Definir los formularios utilizados (si fuera necesario).
- Verificar si los diseños realizados están de acuerdo con los estándares de la unidad y obtener la validación y conformidad de los usuarios.

Para acotar el presente trabajo, se realiza un desarrollo del primer punto de la tarea 4.1 propuesta en la metodología, exponiendo algunos de los productos resultantes.

- Tarea 4.1.1. Definir los formatos individuales de las pantallas utilizadas.

Caso Práctico: Construcción de un sistema de Gestión de Alquiler de Automóviles (AGA 2000).
Requisitos de interfaces de usuario:

- La interfaz con el usuario se hará mediante pantallas con menús desplegables.

Se describe a modo de ejemplo, el formato de la pantalla principal del sistema AGA 2000 y sus componentes (Figura 12).

Descripción de componentes

Barra de Título: se utiliza para desplegar el título de la pantalla desplegada. Si la ventana está activa, la barra de título tendrá un color diferente al resto de las ventanas desplegadas.

Menú Principal: contiene un conjunto de botones que permiten desplegar la totalidad de las pantallas del sistema.

Usuario del Sistema: indica el nombre del usuario que está utilizando el sistema, el cual ha sido previamente ingresado con una contraseña como requisito para acceder al sistema.

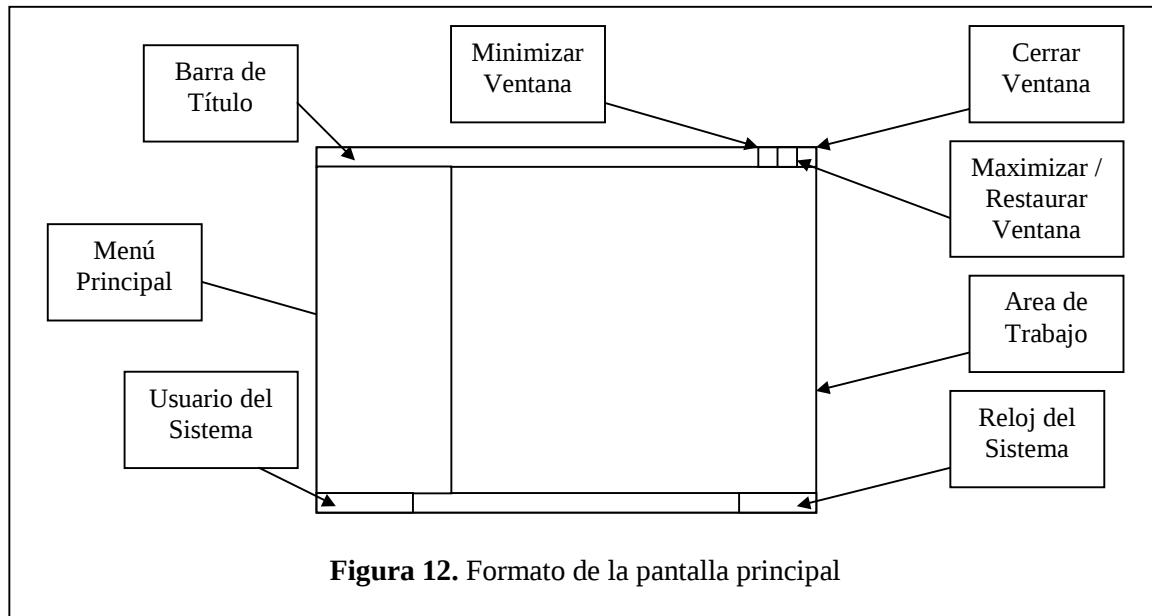
Reloj del Sistema: indica la hora actual del sistema.

Area de Trabajo: es el lugar donde se despliegan las pantallas que son activadas a través del Menú Principal.

Maximizar/Restaurar Ventana: botón que se utiliza para ampliar o reducir el tamaño de la pantalla.

Minimizar Ventana: control que se utiliza para quitar de primer plano de trabajo una ventana, sin cerrarla.

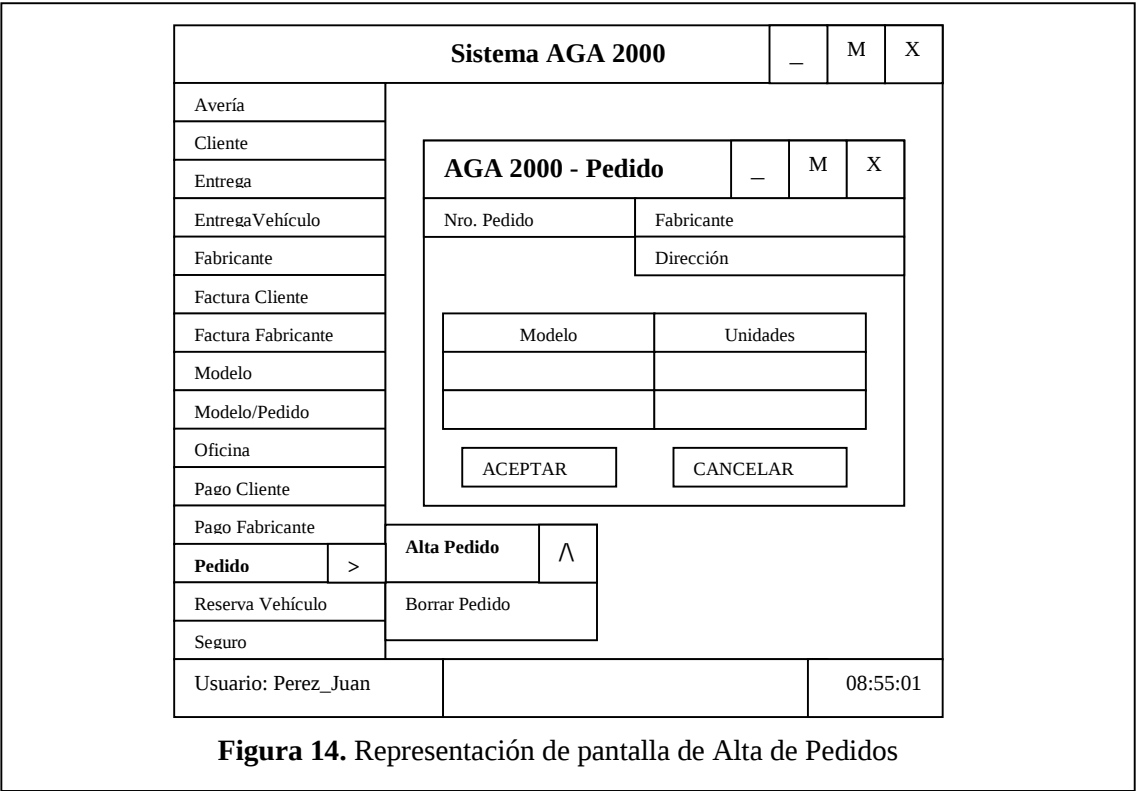
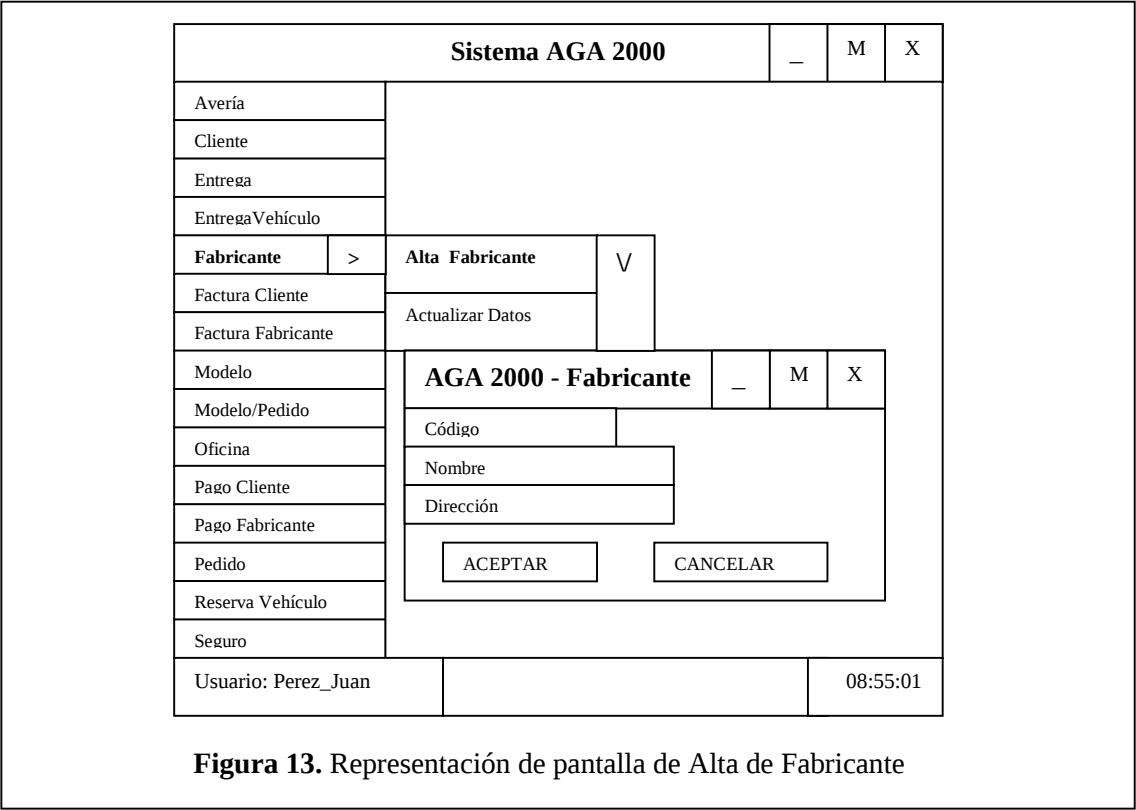
Cerrar Ventana: control que se usa para cerrar una ventana.



Puede utilizarse para la descripción de las pantallas, las operaciones que se especifican en los diagramas de la Historia de Vida de las Entidades del sistema anteriormente presentado (AGA 2000). Cada operación representa una pantalla diferente, por lo que se expondrán algunas de ellas (Figuras 13 y 14) con el objeto de ilustrar el diseño de las mismas.

Opciones de Menú	Operaciones
Avería	Alta avería, Actualización período Avería, Borrado de avería.
Cliente	Alta cliente, Modificar dirección, Emisión tarjeta, Cancelación tarjeta, Renovación tarjeta.
Entrega	Alta entrega, Borrar Entrega
Entrega/Vehículo	Alta entrega / vehículo, Borrar entrega / vehículo
Fabricante	Alta fabricante, Actualizar datos
Factura Fabricante	Alta factura fabricante, Borrar factura fabricante
Modelo	Alta modelo, Borrar modelo
Modelo/Pedido	Alta modelo/pedido, Borrar modelo/pedido
Oficina	Alta oficina, Borrar oficina
Pago Cliente	Alta pago cliente, Borrar pago cliente
Pago Fabricante	Alta pago fabricante, Borrar pago fabricante
Pedido	Alta pedido, Borrar pedido
Reserva vehículo	Alta reserva vehículo, Borrar reserva vehículo
Seguro	Alta seguro, Modificar porcentaje seguro, Borrar seguro

Ejemplos de pantalla para la operación de Alta de Fabricante y Alta de Pedidos



El encadenamiento de las pantallas está determinado a partir de la pantalla principal del sistema, permitiendo desplegar cualquiera de las pantallas utilizadas para las operaciones anteriormente descritas. Dichas pantallas pueden ser activadas o cerradas en forma independiente.

6. Conclusiones

Las integración de los principios, prototipos y heurísticas de evaluación durante el proceso de diseño de IU permite la creación de interfaces que satisfacen las expectativas del Modelo del Usuario, el cual es el punto de vista mas importante para garantizar la aceptación de un sistema.

Entre las características que contribuyen a construir una interfaz sencilla de utilizar, sobresale la utilización de metáforas como ayuda para simplificar al usuario en la operación del sistema.

La prototipación es un proceso de uso frecuente durante el diseño de IU. A través diferentes niveles evolutivos de prototipos se pueden lograr simulaciones del sistema a construir con un alto grado de detalle, pudiendo validar los requisitos de diseño que se hayan propuesto.

Las heurísticas de evaluación de IU permiten identificar problemas que interfieran en la operación del sistema, resultando de ellas un diagnóstico que puede ser utilizado como retroalimentación para una nueva versión optimizada de la interfaz de usuario.

7. Bibliografía

- Guía de Estudio del Módulo III “Metodología de Construcción de Sistemas de Software” de la Maestría en Ingeniería del Software, Escuela de Posgrado, Instituto Tecnológico de Buenos Aires.
- Molich, R., y Nielsen, J., “*Improving a human computer dialogue*”, Communications of the ACM 33, 3 (March), pp 338-348, 1990.
- Molich, R., y Nielsen, J., “*Heuristic evaluation of user interfaces*”, Proceedings of the ACM CHI’90 Conference, pp. 249-256, 1990.
- Molich, R., y Nielsen, J., “*Enhancing the explanatory power of usability heuristics*”, Proceedings of the ACM CHI’94 Conference, pp. 152-158, 1994.
- Durrett, H.J., “*General Approach to Rapid Prototyping*”, <http://swt.edu/~hd01/4326/PROTOTYP.htm>, 1997.