Guía práctica de diseño de la interacción con el usuario

No es el usuario el que tiene que adaptarse al interfaz , sino que hay que diseñarlo de modo que resulte tan intuitivo y natural que lo aprenda y utilice.

1. Diseño centrado en el usuario.

- Diseño centrado en el usuario: su **comportamiento** simple y claro.
- Conocer al usuario: las características de las clases de usuarios.

Los diseñadores necesitan **entender a los usuarios del sistema.** Pueden comenzar *entrevistándolos* (cuestionarios) y *observándolos* trabajar, antes y durante el diseño de la interacción: análisis del usuario, análisis de tareas, análisis del flujo de la información, estudios de tiempo-movimiento. Por ejemplo, interesa saber ¿cuáles son las características cognitivas, de comportamiento, antropométricas, ergonómicas, de actitud del usuario? Qué **tareas** necesita realizar el usuario con y sin ordenador.

- Implicar al usuario haciéndolo participar en el diseño.

A los diseñadores les ayuda entender qué tareas necesitan realizar los usuarios , cuáles realizan habitualmente, en qué condiciones,... Informan al diseñador de cómo **desearían** realizar las tareas o sugerir otras nuevas. Ayuda a hacer más fácil la automatización.

- Prevenir los errores del usuario.

Anticiparse a los posibles errores del usuario. Por ejemplo, desactivar elecciones de menú o botones que no estén disponibles.

- Optimizar las operaciones del usuario.

Por ejemplo, usar acelerador (teclas combinadas).

Macros y abreviaturas. Los macros -incluidos los definidos por el usuario- permiten a los usuarios definir con un nombre singular una secuencia de acciones o tareas utilizadas frecuentemente, de modo que no haya que teclear todos los pasos de una secuencia completa cada vez que se quieran realizar esas tareas.

- Dar el **control** al usuario.

Que el usuario se sienta con el control en la mano.

Dar mensajes no culpabilizadores, por ej., en vez de "introduzca el comando siguiente" (enter next command) poner "preparado para el comando siguiente" (ready for next command). la computadora está preparada para responder a la demanda del usuario.

Tiene un impacto psicológico.

- **Ayudar** al usuario a hacerse con el sistema.

Ayuda a tiempo, cuando se necesita.

Resumen:

- Diseño centrado en el usuario.
- Conocer al usuario.
- Implicar al usuario haciéndolo participar en el diseño.
- Prever los errores del usuario.
- Optimizar las operaciones del usuario.
- Buscar dónde conviene dar el control al usuario.
- Ayudar al usuario a hacerse con el sistema.

2. Modelo del sistema.

- Dar al usuario un **modelo del sistema** basado en las tareas del propio usuario.

Esta es una de las causas de los interfaces pobres.

Un modelo del sistema establece la **estructura de la arquitectura** de un sistema.. Normalmente representa el flujo de datos y operaciones realizadas con esos datos. Esto se aplica al modelo conceptual, que es la visión de la secuencia y la funcionalidad típicas que el desarrollador (y, por tanto, el sistema) ofrece al usuario. Esto se traduce al modelo mental del usuario: cómo percibe al sistema (Mayhew, 1992). El modelo mental gobierna cómo el usuario entiende e interactúa con el sistema.

Un modelo mental **consistente** del usuario, basado en las tareas que el usuario realiza, guiará de un modo general al usuario para que complete las tareas. Por ejemplo, en un interfaz de manipulación directa, un usuario realiza tareas seleccionando primero un objeto (sea un icono), y lego realiza una acción sobre este objeto (sea abrirlo). Este usuario aprende rápidamente este paradigma objeto-acción, y lo aplicará en todo el interfaz

Resumen:

- Modelo del sistema.
- Modelo mental consistente.

3. Consistencia y simplicidad.

- Ser **consistente**. A veces se lo llama "principio de epatar lo menos posible".

Si algo se hace de cierta manera en el interfaz (por ejemplo, una tarea con nombre específico, un icono con una cierta apariencia o comportamiento, etc.), el usuario espera que la misma cosa funcione de la misma manera en el resto del interfaz: ***cosas similares se espera que se hagan de maneras similares***.

Evitar que un criterio pueda entrar en conflicto con otro.

La consistencia influye mucho en la usabilidad.

- **Simplicidad**.

Dividir las tareas en subtareas más simples, es decir, reducir las tareas a iconos, acciones, palabras, ... naturales al usuario.

Resumen:
- Consistencia del interfaz.
- Simplicidad.

4. Memoria humana.

- Tener en cuenta las limitaciones de la **memoria** humana. La memoria a corto plazo dura entre 30 segs y 2 minutos.

Hay que limitar, pues, el número de items que el usuario tiene que tratar en un momento. La información tiene que organizarse sin scrolling, sin buffer...

¿Cómo manejamos las interrupciones cuando se realizan tareas? Por ejemplo, con el correo electrónico. ¿Cómo restaurar el entorno anterior de trabajo?

Las pilas mentales funcionan a corto plazo: hay una pérdida de información en periodos relativamente cortos.

Abrir subtareas por niveles, de abajo-arriba, para que la interrupciones tengan menor impacto.

Secuencias lineales cortas para las acciones del usuario.

- Mejor **reconocer** que recordar.

Por ejemplo, buscar (reconocer) una elección de un menú es mucho más fácil para el usuario que tener que recordar todas las elecciones posibles y luego escribir una de ellas como comando. Al reconocer el campo de la memoria el reconocimiento reduce la posibilidad de cometer (escribir) errores.

Resumen:
- Que el usuario pueda recordar las tareas.
- Que el usuario reconozca y no tenga que recurrir a la ayuda.

5. Aspectos cognitivos.

- Seguir la orientación **cognitiva**.

La orientación cognitiva implica minimizar las transformaciones mentales que el usuario tiene que hacer, disminuir el esfuerzo del usuario al realizar tareas. A ello ayudan las técnicas mnemónicas de memoria, por eje., las teclas de aceleración.

Usar letras significativas para el menú de selección, por ej., "cut", "copy",... Utilizar signos visuales: iconos, flechas,... Que los objetos y el movimiento del cursor siga los movimientos mentales.

- Sacar **analogías** del mundo real.

Utilizar situaciones, palabras, imágenes y metáforas naturales y conocidas por los usuarios, de modo que las expectativas de los usuarios y la dirección cognitiva crezcan. Como sucede en las hojas de cálculo y en la metáfora de los ficheros (desktop) del Macintosh.

Mímica del mundo real.

A veces interesa añadir etiquetas a los iconos para identificarlos mejor. Hacer ejercicios de creación de iconos. A veces se identifican mejor los iconos fotográficos.

Resumen:
- Minimizar las transformaciones mentales que el usuario tenga que hacer.
- Trazar analogías con el mundo real.

6. Feedback.

- Utilizar feedbacks **informativos**.

Un bucle cerrado es mejor que uno abierto.

Un buen feedback permite a los usuarios controlar (medir) mejor sus acciones.

Los usuarios necesitan feedbacks sintácticos (articulatorios) y semánticos. Un feedback **sintáctico o articulatorio** dice a los usuarios que sus **manos** trabajan correctamente. Por ej., cuando un usuario selecciona "open" de un menú pull-down, la elección puede parpadear brevemente antes de que el menú desaparezca, es un feedback articulatorio que indica a los usuarios que pincharon la selección pretendida. Luego, en la caja de diálogo, aparecerá la lista de los ficheros posibles que puede abrir, este es un feedback **semántico**, que confirma que la elección pretendida era correcta para la tarea que el usuario deseaba realizar.

Normalmente se utilizan soportes visuales -textuales o gráficos- para los feedbacks. Muestran a los usuarios los efectos de las acciones y pueden mostrarles lo que está disponible en un momento dado.

- Proporcionar al usuario **indicadores de estado** apropiados.

Siempre que el sistema esté realizando un proceso potencialmente largo, debe darse al usuario un feedback, por ej., desplegar un reloj de arena, un reloj de agujas que se muevan, o simplemente "working...".

En general, el sistema debería tener una respuesta de tiempo adecuada para cada tarea. Por ej., para escribir, mover el cursor y apretar el ratón, se recomienda una respuesta de tiempo de 50 a 150 msgs. Para tareas simples y frecuentes, la respuesta debería ser menor de 1 sg, para que el usuario no se aburra. Para tareas comunes, se recomienda de 2 a 4 sgs. En tareas más complejas, los usuarios toleran hasta unos 12 sgs.

Para respuestas mayores de 2-4 sgs, el usuario debería tener un indicador de estado, sobre todo si el usuario no puede interactuar con el sistema, mientras el sistema procesa. Al terminar el proceso, el indicador de estado desaparecerá automáticamente. EL indicador de estado debe permanecer en la pantalla al menos 2 sgs para que el usuario pueda leerlo.

En otras situaciones, como buscar en una red, conectar, formatear un disco, transferir ficheros, etc., lo útil es mostrar al usuario un indicador de la parte de la tarea que queda por realizar (por ej., el porcentaje). Cuando la tarea supera algunos segundos, esto es lo que hay que hacer.

Resumen:
- Utilizar feedback informativo.
- Dar al usuario indicadores de estado.

7. Mensajes del sistema.

- En los mensajes utilizar palabras **orientadas al usuario**. No mensajes crípticos. Comunicarse con los usuarios en términos de sus tareas y en palabras familiares a ellos. Por ej., no epatar con " 505 hex 0001F9 doublewords of storage were not recovered".
- Utilizar en los mensajes **frases positivas**.

Como los mensajes forman parte de la interactividad del sistema tienen un gran impacto en el usuario. Es siniestro decir: "fatal error, run aborted". El usuario necesita ayuda y ánimo.

- En los mensajes **de error**, utilizar términos específicos y constructivos.

Frases como "syntax error", o "incorrect data", son poco específicos. El sistema debe analizar los errores y proporcionar al usuario la información que tenga.

Deben ser constructivos. Por ej., "invalid entry", como respuesta a un input erróneo para un campo, es inútil para el usuario. Una respuesta más específica podría ser "Inventory part number is out of allowable range", o mejor, "Inventory numbers range from 0000 to 9999".

Ser breves y concisos, evitando verborrea, y, si hace falta, añadir una ayuda con información adicional.

- Echar la **culpa** de los errores al sistema (no al usuario).

Es diferente decir "illegal command" que "unrecognized command".

Los diseñadores de interacción novatos se sorprenden de la cantidad de trabajo que da escribir mensajes positivos, específicos, constructivos y orientados al usuario.

Resumen:
- Mensajes informativos orientados al usuario.
- Mensajes de error positivos, constructivos y específicos.
- Evitar mensajes culpabilizadores.

8. Antropomorfización.

- **No antropomorfizar**, o sea, no atribuir características humanas a objetos no humanos (por ej., a las computadoras). A veces se ponen caras que mueven los ojos, como si miraran, o los labios, como si

hablaran ("Buenos días, Pepe. ¿Cómo estás hoy?"). Sí vale en los sistemas de entrenamiento, pero utilizado de un modo juicioso y cuidadoso (schneiderman, 1992).

El problema es que la antropomorfización puede causar irritación o, por el contrario, producir en enganche a una falsa impresión de inteligencia, donde no la hay. Incluso el término "amigable" es confusamente antropomórfico.

- Utilizar **"agentes".**

Resumen:

- No antropomorfizar sin motivo y cuidado.

- Crear "agentes".

9. Modalidad y acciones reversibles.

- Utilizar los **modos** con cuidado.

Un modo es un estado del interfaz en el que la acción del usuario tiene un significado (y un resultado) diferente al que se le da en otro estado.

Dar una señal virtual para distinguir los modos: por ej., en un editor gráfico, la forma del cursor puede cambiar para indicar si el editor está en modo de crear círculos o líneas. Que el usuario conozca en todo momento los modos activos.

El modo 'preemptive' es aquel en que el usuario tiene que completar una tarea antes de iniciar otra. Por ej., en una caja de diálogo en 'modo preemptivo' el usuario tiene que terminar cierta tarea mostrada en la caja antes de que la caja se vuelva inactiva, o que desaparezca; mientras la caja esté desplegada, el usuario no podrá realizar ninguna acción fuera de la caja.

En general, el 'modo preemptivo' debe evitarse, excepto cuando el usuario está obligado a dar una respuesta (por ej., salvar un fichero o no) antes de seguir con la tarea.

- Hacer fácilmente **reversibles** las acciones del usuario.

Un ejemplo es el comando "undo" en los sistemas gráficos.

También se aplica la reversibilidad a las acciones de navegar por el sistema, por ej., permitiendo al usuario volver a niveles previos o pantallas (al menos a la pantalla previa). Debe poder cancelarse una tarea sin necesidad de completarse (scape, exit,...). Cuidar que el usuario sepa lo que anula con el "undo" . En ventanas o menús conviene que el usuario pueda volver al menos a lo anterior.

Esto anima a los usuarios a investigar el sistema (estando prevenidos de acciones desastrosas).

Resumen:

- Cuidar los modos dando al usuario información de su estado.

- Facilitar la reversibilidad de las acciones.

10. Ganar la atención del usuario y mantenerla.

Pero ganarla juiciosamente.

a) Para el **texto**, la regla es utilizar sólo 2 niveles de intensidad en una sola pantalla. Ser parco con el subrayado, bold y vídeo inverso, para orientar la vista al objeto. En pantallas de texto, en general, no utilizar más de 3 fuentes diferentes en una sola pantalla, y no más de 4 tamaños diferentes. Las fuentes con serif son más fáciles de leer (el ojo se desliza mejor). No utilizar sólo mayúsculas, porque relentizan la velocidad de lectura un 10%.

b) Controlar mucho el **parpadeo**, sino irrita.

c) **Audio**.

Para hechos importantes y redundantes.

Utilizarlo cuando lo que aparece en la pantalla no es suficientes para llamar la atención del usuario, o porque la pantalla esté demasiado llena. Es un canal suplementario. Se pueden utilizar beeps o sonidos suaves como confirmaciones positivas de que una acción se realizó con éxito, o, por ej., que la impresora no tiene papel. Tonos más estimulantes para emergencia, o para captar la atención inmediatamente.

Cuidar no saturar de sonido o irritar.

d) **Color**.

Como ejercicio, comenzar a diseñar en blanco y negro (monocromo).

Generalmente no utilizar más de 4 colores diferentes en una pantalla, especialmente si la mayor parte del contenido es texto, y no más de 7 en toda la aplicación. En general, el azul o el negro son los mejores para el background, especialmente para el texto con caracteres blancos o amarillos, respectivamente. El azul no debe ser utilizado para texto, pues es uno de los más difíciles de leer porque

los receptores del ojo (los conos) son muy insensibles al azul. Por ello, el azul es un buen color para el background y para campos amplios, no para los detalles.

Un background brillante (por ej., blanco) al cabo del tiempo fatiga los ojos. Importa que el background y el foreground contrasten.

El color sirve para codificar, pero debe utilizar de un modo conservador y redundante. Sirve para relacionar objetos y campos en la pantalla. Por ej., ventanas o pantallas para tareas pueden relacionarse por su color. También pueden llamar la atención sobre información importante, o sobre cambios. Y dan más realismo a los iconos. Combinar forma y color para distinguir clases de items.

Hay convenciones culturales sobre el color. Por ej., en USA, el verde, amarillo y rojo tienen connotaciones especiales. Cuando los norteamericanos ven un objeto verde, automáticamente piensan. "adelante" (go) o "vale" (O.K.). El amarillo les sugiere precaución. Y el rojo les produce la idea de "stop" o "algo está equivocado". El rojo se reserva para los mensajes de emergencia o iconos críticos.

Como "para gustos se hicieron colores" es bueno, dejar al usuario customizar el color: darles su control, salvo que se quiera dar al color un significado especial en el diseño.

Resumen:
- Retener razonablemente la atención del usuario
- Usar adecuadamente los recursos de letra, color, sonido, etc.

11. Despliegue (pantallas).

- Mantener el despliegue.

Cambiar lo menos posible las pantallas.

Los objetos estáticos, como los botones, frases e iconos, deben aparecer en todas las pantallas en el mismo sitio, por razones de consistencia. Que el usuario no tenga que ajustar los ojos a nuevas pantallas: evitar fatigarlo.

La inercia es buena en el lugar, forma y tamaño de los objetos, pero no necesariamente en las etiquetas, indicadores por defecto, etc.

- Organizar las pantallas para gestionar la complejidad.

Simplificar la pantalla eliminando la información innecesaria. Frases concisas, mensajes e iconos fáciles de reconocer.

Minimizar la densidad de la pantalla total, especialmente el texto, y las subáreas. Es difícil leer iconos muy próximos, y pueden oscurecer la información que el usuario necesita para realizar sus tareas.

Hay que evitar tener mucha información arriba, abajo, a la izquierda y a la derecha de la pantalla.

La tendencia inicial equivocada es poner demasiada información arriba o abajo en la pantalla.

Conviene imaginar la pantalla dividida en **cuadrantes**, y poner más o menos la misma información en cada cuadrante.

Dejar bastante espacio en blanco alrededor del texto. No dejar menos del 25% de espacio en blanco (Tullis, 1988); y, en el texto, se recomienda un 50%.

Agrupar lógicamente en la pantalla la información relacionada y utilizar formas de letra e iconos familiares al usuario. Alinear columnas y filas. Evitar caracteres especiales.

Resumen:
- Mantener la inercia del display de una pantalla a la siguiente.
- Organizar la pantalla para gestionar la complejidad.

12. Los usuarios.

Acomodarse a la experiencia y diferencias de los usuarios individuales. Por muy experto que sea un usuario, resulta un novato cuando usa por primera vez un sistema.

Las diferencias de usuarios dependen de su experiencia, aptitudes técnicas particulares, edad, dominio del problema y conocimiento de él. También la habilidad de visualización, vocabulario, y razonamiento lógico.

Hay que permitir a los usuarios tomar decisiones sobre el interfaz: elegir versiones cortas o largas de menús y ayuda y mensajes; menús cortos con funciones simples, y largos con otras más complejas.

Que el usuario customice cambios en los comandos o nombres de menús, en el lugar de los objetos interactivos, etc.

- Adaptarse a los niveles de experiencia del usuario.

Al menos hay 3 niveles de experiencia de los usuarios (Shneiderman, 1992):

a) **Novatos**.

El novato es el no tiene conocimiento sintáctico del sistema, y sólo un poco de conocimiento semántico. Estos usuarios se suelen acercar al sistema con ansiedad o pánico. Necesitan que el interfaz sea claro y simple, con un pequeño número de funciones significativas, mensajes de error lúcidos, y feedback informativo. También pueden necesitar manuales comprensibles y tutoriales "online" y demos. Esto además sirve para los otros niveles.

b) **Intermitentes**.

Los intermitentes son los que mantienen un conocimiento semántico del sistema a lo largo de del tiempo, pero pierden conocimiento sintáctico. Para el interfaz, tales usuarios prefieren comandos simples y consistentes, secuenciación de los pasos significativos, funciones y teclas fáciles de recordar, algún recordatorio de asistencia y ayuda, y manuales concisos.

c) **Frecuentes.**

Los frecuentes son los que tienen conocimiento semántico y sintáctico del sistema. Estos son los típicos usuarios "potentes". Desean interacción rápida, potentes comandos y, tal vez, con teclado, teclas reducidas, breves mensajes de error con acceso a detalles para poder investigarlo, feedback conciso, y customización de su propio interfaz.

Resumen:
- Adaptarse a las diferencias y experiencias del usuario.
- Clasificar y jerarquizar a los usuarios.

Referencias

Open Software Foundation, 1990: OSF/Motiv Style Guide. Englewood Cliffs, N.J.: Prentice Hall.
Algunos libros básicos:

- Powell, J.E. (1990), Designing User Interface. San Marcos, CA: Microtrend Books.
- Mayhew, D.J. (1992), Principles and guidelines in Software User Interface Design. Englewood Cliffs, N.Y.: Prentice Hall.
-Marcus, A. (1992), Graphic Design for Electronic Documents and User Interfaces. N.Y.: ACM Press/Addison-Wesley.
- Schneiderman, B. (1992), Designing the User Interface: Strategies for Effective Human - Computer Interaction. Reading, MA: Addison-Wesley.