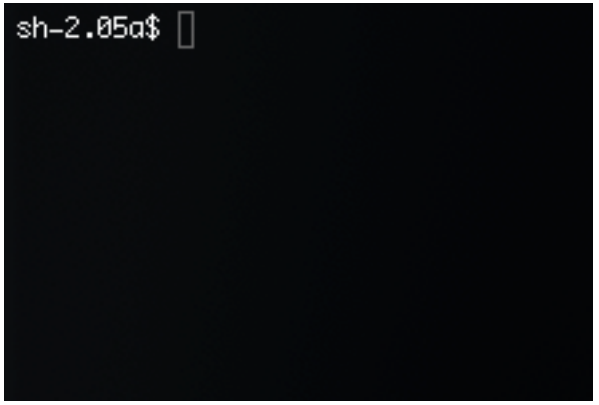
Interfaces gráficos

Dr.Daniel Morató
Area de Ingeniería Telemática
Departamento de Automática y Computación
Universidad Pública de Navarra
daniel.morato@unavarra.es
Laboratorio de Interfaces de Redes
http://www.tlm.unavarra.es/asignaturas/lir

CLI

- El diseñador del interfaz hace poco más que permitir al usuario hacer lo que quiera
- El usuario debe memorizar los comandos
- Una vez conocido es fácil de utilizar, pero no es fácil de aprender a utilizar
- Pregunta-respuesta:
  - No hace falta memorizar opciones, el programa pregunta por cada una de ellas
  - No puedo cambiar el orden. Ejemplo: especificar nombre de fichero



Satisfacción/frustración

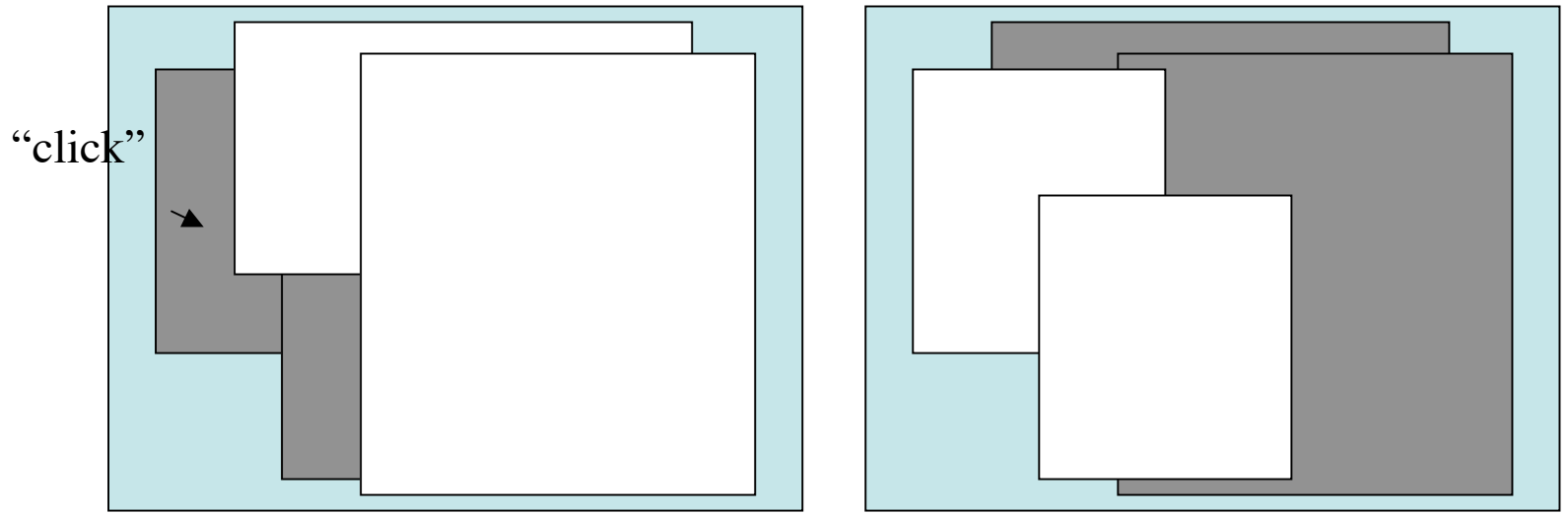
- Las pequeñas frustraciones se acumulan
- Más feliz cuanto más se sienta en control del entorno
- Desagrado cuando sucede algo que no se puede controlar
- Un interfaz de usuario está bien diseñado cuando el programa se comporta exactamente como el usuario espera que se comporte
- El usuario no quiere pensar en cómo funciona. El interfaz será mejor cuanto menos haga pensar al usuario.

Modelos

- El usuario no es una *tábula rasa*. Al empezar a utilizar un programa ya cuenta con unas expectativas de cómo funciona
- Esto es el
- Cómo funciona el programa es el
- Un interfaz de usuario está bien diseñado cuando el modelo del programa se ajusta al modelo del usuario
- Debemos ajustar el modelo del programa al modelo del usuario
- Ejemplos:
  - Documentos en blanco
  - Editores de páginas HTML vs editores de texto.

¿Cómo es el modelo del usuario?

- Pregúntale a él/ellos
- Opiniones diferentes
- Habrá una opinión mayoritaria
- Basta con probarlo con unos pocos usuarios
- El usuario asume el modelo más simple posible
- Ejemplo:
  - Cambio de aplicación activa con aplicaciones con varias ventanas



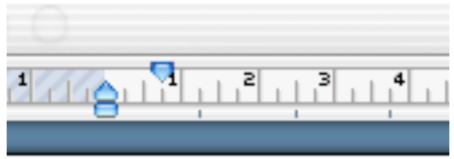
Opciones

- Cada vez que se ofrece una opción se le está pidiendo al usuario que tome una decisión
- El problema está en pedir decisiones sobre algo que al usuario no le importa. Esas decisiones son tarea del diseñador
- Al usuario le interesan las decisiones relacionadas con la tarea que desea completar
- Muchas veces no se saben tomar las decisiones en el diseño y el resultado son cuadros de Opciones o Preferencias plagados de alternativas
- Ejemplos:
  - Búsqueda de ayuda en Windows
  - Toolbars y menú flotante
  - Cambios de aspecto sin cambios de comportamiento.

Metáforas

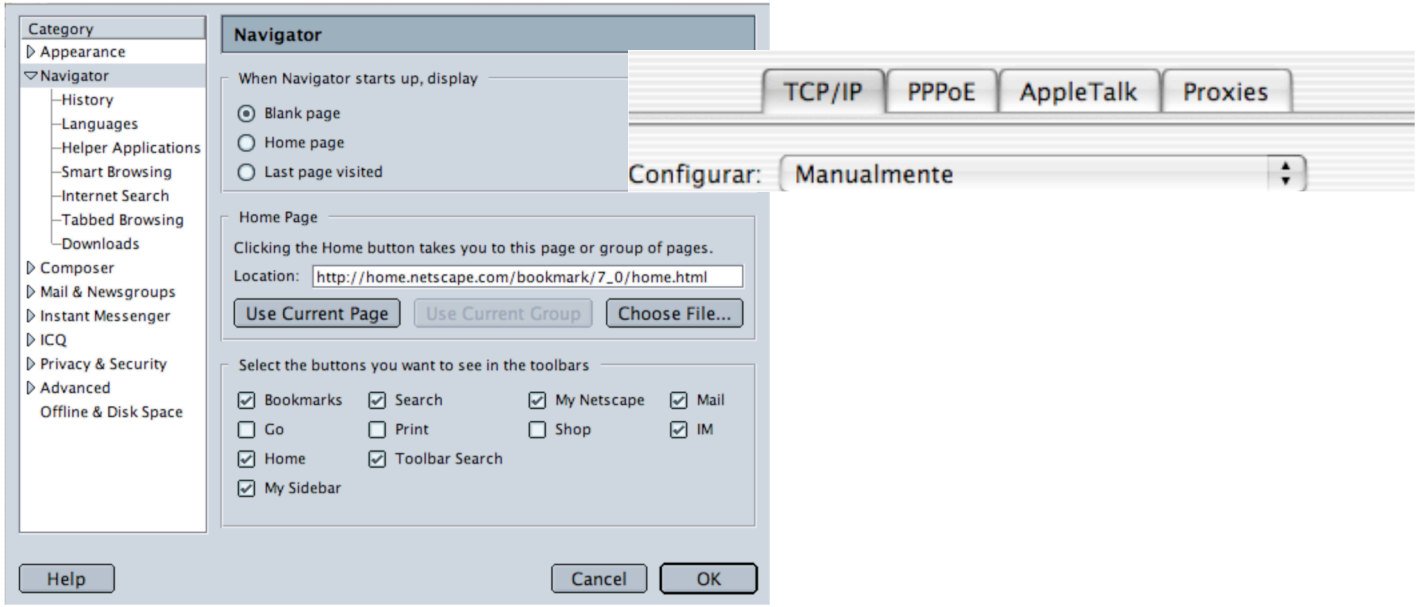
- Cuando el modelo de usuario está incompleto el programa puede emplear metáforas para enseñarle su modelo
- Enseñarle al usuario cómo funciona el programa dándole pistas mediante metáforas
- Deben comportarse de forma predecible, como los objetos de la vida real, si no sólo añaden confusión
- Ejemplos (buenos y malos):
  - Escritorio
  - Zoom
  - Papelera
  - Marca de sangrado
  - Carpetas dentro de carpetas.





Elementos autoexplicativos

- Ejemplos:
  - Chapa de empujar para abrir puerta
  - Cajas de CDs y DVDs
  - Botones 3D
  - Cuadro de opciones de preferencias del Netscape vs pestañas...



Consistencia vs creatividad

- Consistencia entre diferentes programas los hace más fáciles de utilizar para el usuario
- Básicamente el modelo del programa se adapta al modelo aprendido del usuario
- Empleada inteligentemente nos permite evitar “reinventar la rueda”

Pruebas de usabilidad

- No hace falta mucha gente
- No son tan importantes los resultados estadísticos como detectar fallos en el interfaz
- Miden lo fácil que es aprender a usar el programa
- Generalmente el software no está terminado, lo cual genera sus propios problemas
- El entorno no es el habitual del usuario y además se siente “examinado”
- Gente falla el test por no saber ni por dónde empezar
- Por ello se inventaron los wizards o los asistentes:
  - No resuelven el problema de usabilidad, lo enmascaran
  - Son como las preguntas de los programas antiguos pero hacen que tenga éxito el test
  - Si el usuario quiere hacer algo diferente vuelve el problema
  - Ejemplo: balloon sobre el botón de inicio en windows

Diseñar para situaciones extremas

- ¿Entiende y puede usar el interfaz un experimentado informático?
- ¿Entiende y puede usar el interfaz un anciano con vista cansada y parkinson?
- Diseñándolo para casos extremos será más sencillo de utilizar en situaciones normales

El usuario no sabe leer

- Puede que el usuario no tenga el manual
- Aunque tuviera el manual, el usuario no lo leería
- De hecho tampoco va a leer lo que aparezca en el interfaz si puede hacer lo que quiere sin leerlo
- Si tiene que leerlo para hacer lo que quiere será una molestia
- Puede que incluso en ese caso no lo lea y deje de intentar hacer lo que quería o cambie de programa
- Cuanto más aparezca para leer menos probabilidades hay de que lo lea
  - Los usuarios expertos asumen que saben lo que se les intenta explicar y no tienen tiempo para leerlo
  - Los inexpertos se saltan las instrucciones esperando que las selecciones por defecto les valgan
  - Lo pocos inexpertos que lo leen pueden acabar confundidos por los conceptos que se les intentan explicar
- Ejemplo: usuario intimidado por los ordenadores que intenta salir de un programa.

El usuario no sabe usar el ratón

- Una barra de menú en la parte superior de una ventana tiene menos de 1cm de anchura
- Una barra de menú en la parte superior del escritorio tiene menos de 1cm de anchura pero “una milla” de altura
- ¿Cuáles son los 5 puntos de la pantalla más fáciles de acertar con el ratón?...
  - Las 4 esquinas
  - El punto donde ya está el ratón
- Ejemplo: El botón de *Inicio* de Windows 95
- Los usuarios no saben usar el ratón:
  - Puede que empleen malos dispositivos: trackpad, trackball
  - Las condiciones no sean óptimas: bola sucia, mala alfombra para óptico, en un tren que se mueve
  - El usuario inexperto con ordenadores no sabe utilizar un ratón
  - Hay personas con problemas motrices
  - Doble-click sin mover el ratón es complicado para muchas personas
  - En general tener que mover con cuidado el ratón requiere energía y concentración que el usuario prefiere emplear en realizar mejor su tarea

El usuario no recuerda

- Ejemplo: nombre del fichero a abrir
- Los menús sustituyen a los comandos de CLI, ya no hay que memorizarlos.

Relatividad temporal

- Los días son segundos:
  - Partes de un interfaz que se tarda días en desarrollar solo serán experimentados por el usuario durante unos segundos
  - El programador pasa mucho más tiempo delante del interfaz. Lo que para él es evidente no lo será para alguien que solo está unos segundos ante él
- Los meses son minutos:
  - En otra escala, el desarrollo de un software completo lleva meses o años
  - Durante ese tiempo se desarrollan muchos conceptos y se incluyen muchas opciones en el programa
  - El resultado final puede ser un programa con una gran cantidad de opciones que el usuario debe aprender a usar en muy poco tiempo
  - Los mejores diseños suelen tener el menor número de opciones
- Los segundos son horas:
  - Si el programa parece lento los usuarios sienten que pierden el control
  - Trucos:
    - Responder inmediatamente a la acción del usuario. Aunque no se complete que sepa que está en curso
    - Dividir el procesamiento en pequeños bloques de forma que sean despreciables para el usuario
    - Unir los procesamientos en un gran bloque que permita al usuario cambiar de actividad (Ejemplo: preguntar todo lo necesario antes de empezar instalación lenta, no preguntas en medio).

Heurísticos

- Se intenta adivinar lo que el usuario quiere hacer
- El programa *apuesta* por lo que considera más probable
- Ejemplo: se empieza a escribir el nombre de un fichero y lo completa
- Ejemplo: :-) en word
- Se pueden volver molestos con facilidad
- Problema: cada vez que el heurístico falla crea más desagrado en el usuario que muchos aciertos
- ¿Cuánto? Depende de lo que cueste deshacer la acción del heurístico
- Un buen heurístico es obvio cómo adivina, fácil de deshacer y acierta con una alta probabilidad.

La Web

- Se diseñó para ir mostrando documentos uno a uno
- Un interfaz con mucha interactividad con procesamiento en el servidor se enfrenta al *lag*
  - El tiempo que transcurre entre que el usuario realiza una acción y obtiene la respuesta es largo por culpa de la red
  - Ejemplo: webmail, borrar correos uno a uno
  - En la web cada click del usuario implica al menos un RTT lo cual reduce la usabilidad. Se debe diseñar para minimizar el número de RTTs de espera experimentados
- No dispone de muchos elementos de UIs
  - Ejemplo: no hay menús, si se implementa con JavaScript hay problemas de compatibilidad y si se hacen con `<select>` se supone que seleccionar la opción no debería generar un cambio de URL (no lo espera el usuario)
  - Ejemplo: no se puede sacar un cuadro de diálogo. Sacar otra ventana no es igual y se confunde con publicidad
  - Ejemplo: no hay un buen campo de edición de texto. `<textarea>` no soporta grabar por si se cierra la ventana
- Un problema de usabilidad es muy grave porque la competencia “está a un *click* de distancia”.

Modelo del usuario

- Logo en la esquina superior izquierda
- Click en él lleva a la *home page*
- Los enlaces son azules y subrayados y todo lo que es azul y subrayado es un enlace
- Los botones parecen botones
- El usuario trae ya aprendido el modelo de la prensa impresa: encabezados, secciones, etc.

Comportamiento del usuario

- El usuario no lee las páginas, las escanea, busca el enlace interesante y lo pulsa
- No ve todas las opciones y escoge la mejor sino que escoge la primera *razonable* que ve
  - Puede que tenga prisa
  - Si se equivoca puede volver atrás
  - Es más satisfactorio
- Muchas veces en realidad no entiende cómo funcionan las cosas sino que se hace su propia idea de cómo funcionan. Al menos mientras le sirva. Ejemplo: creer que google o AOL son Internet.

Navegación

- Es muy importante que el usuario vea con facilidad cómo navegar por el sitio web
- En comparación con el mundo real el usuario no tiene información espacial de dónde está. Se le debe indicar con claridad (logo, sección). Ejemplo: breadcrumbs

CNET > Downloads > Mac > **Multimedia & Design**

- El usuario no puede saber cuánto le queda por explorar, no hay sentido de la escala. Debe poder ver las opciones
- Puede llegar hasta ahí de muy diferentes formas, en vez de siguiendo un camino jerárquico puede ir con un enlace directo.

Top Ten Web Design Mistakes of 2005 (Jakob Nielsen)

1. Problemas de legibilidad con tipos de letra demasiado pequeños, de tamaño fijo, o con poco contraste con el fondo.
2. Enlaces no estándar: no subrayados, que no distinguen enlaces visitados de no visitados, JavaScript…
3. Flash.
4. Contenido que no ha sido escrito para la web.
5. Búsquedas que funcionan mal.

Top Ten Web Design Mistakes of 2005 (Jakob Nielsen)

6. Incompatibilidad con ciertos navegadores.
7. Formularios demasiado largos y complicados.
8. Falta de información de contacto o sobre la empresa en general.
9. Diseños rígidos con ancho fijo.
10. Mal uso de la ampliación de fotos; si vas a dejar que los usuarios vean una imagen más de cerca, que sea a un tamaño lo suficientemente grande para permitirle apreciar los detalles, nunca le enseñes la misma imagen o una versión sólo un poco mayor.

Resumen

- El objetivo es poner en línea el modelo del usuario y el modelo del programa
- Debemos saber como espera el usuario que funcionen las cosas para adaptar el modelo del programa
- Podemos enseñarle el modelo del programa al usuario con metáforas
- Aunque suene duro:
  - El usuario no lee
  - El usuario no sabe usar el ratón
  - El usuario no tiene buena memoria
- Diseñar para el caso mas desfavorable hará que el interfaz sea más manejable en todos los casos.

Lecturas interesantes

- *User interface design for programmers.* Joel Spolsky

- *Don’t make me think: A common sense approach to web usability.* Steve Krug

- *Designing web usability: The practice of simplicity.* Jakob Nielsen

Próximo día

Proyecto final