

GRADO EN INGENIERÍA INFORMÁTICA

Módulo de Formación Básica

INTERFACES USUARIO - COMPUTADOR

Dra. D.^a Alexandra La Cruz Puente





Este material es de uso exclusivo para los alumnos de la VIU. No está permitida la reproducción total o parcial de su contenido ni su tratamiento por cualquier método por aquellas personas que no acrediten su relación con la VIU, sin autorización expresa de la misma.

Edita

Universidad Internacional de Valencia

Grado en

Ingeniería Informática

Interfaces Usuario - Computador

Módulo de Formación Básica

6ECTS

Dra. D.^a Alexandra La Cruz Puente

Índice

INTRODUCCIÓN	9
TEMA 1. INTERFAZ DEL USUARIO (UI).....	13
TEMA 2. USABILIDAD	15
2.1. Errores de diseño que afectan a la usabilidad	15
2.2. ¿Por qué es importante la Usabilidad?.....	18
TEMA 3. PRINCIPIOS GENERALES DE LA USABILIDAD	21
3.1. Facilidad de Aprendizaje	23
3.1.1. Mente Humana	24
3.1.2. Reconocimiento vs Recuerdos	25
3.1.3. Estilos de Interacción que afectan a la capacidad de Aprendizaje.....	25
3.1.4. Modelo de Usuario vs Modelo de los Sistemas.....	27
3.1.5. Principios de la facilidad de aprendizaje vs Diseño de Patrones.....	28
3.2. Visibilidad.....	30
3.2.1. Acciones Visibles.....	31
3.2.2. Estados Visibles.....	31
3.2.3. Feedback	31
3.3. Eficiencia.....	31
3.3.1. Procesamiento humano de la Información	32
3.3.2. Principios prácticos y patrones de diseño para hacer una interfaz más eficiente.	33
3.3.3. Evaluación Predictiva	35
3.4. Errores y Control de Usuario	37
3.4.1. Error humano	37
3.4.2. Principios de Diseño.....	37
3.4.3. Deshacer (Undo).....	37
TEMA 4. DISEÑO CENTRADO EN EL USUARIO	39
4.1. Modelos para el Desarrollo de software.....	40
4.1.1. Modelo de Cascada	40
4.1.2. Modelo Iterativo.....	40

4.1.3. Modelo de Espiral	41
4.2. Procesos de Diseños Centrados en el Usuario	41
4.2.1. Pautas del diseño centrado en el Usuario	41
4.2.2. La implementación o el desarrollo del diseño	42
4.2.3. Evaluación del Diseño Centrado en el Usuario	43
TEMA 5. ARQUITECTURA DE SOFTWARE PARA EL DISEÑO DE LA INTERFAZ DE USUARIO	45
5.1. Modelo de Vista Jerárquico.....	46
5.1.1. Separación de Funciones de las Estructuras Jerárquicas	46
5.2. Manejador de Eventos	46
5.3. Modelo Vista Controlador (MVC)	47
5.3.1. Ventajas del MVC.....	48
TEMA 6. DISEÑO GRÁFICO	49
6.1. Simplicidad.....	49
6.1.1. Reducción.....	50
6.1.2. Consistencia	50
6.1.3. Doble función.....	51
6.1.4. Fragmentación	51
6.2. Contraste y Variables Visuales	51
6.2.1. Contraste	51
6.2.2. Variables Visibles	52
6.2.3. Selectividad	53
6.2.4. Asociatividad.....	53
6.2.5. Técnicas para lograr el Contraste	53
TEMA 7. VISUALIZACIÓN DE LA INFORMACIÓN.....	55
7.1. Algunas formas de Visualizar los datos.....	56
7.1.1. Gráficos Tradicionales	56
7.1.2. Diagramas de Dispersión (Scatterplot).....	56
7.1.3. Treemap.....	57
7.1.4. Nube de etiquetas.....	57
7.2. Algunos de los diez mejores diagramas de visualización de la información a modo de ejemplo. .	57
7.3. Herramientas de Visualización	59

TEMA 8. ACCESIBILIDAD E INTERNACIONALIZACIÓN	61
8.1. Accesibilidad	62
8.1.1. Tipos de discapacidad	62
8.1.2. Tecnología Asistida	62
8.1.3. Guías de accesibilidad	64
8.2. Internacionalización y Localización	64
8.2.1. Retos de Diseño	65
8.2.2. Técnicas para la implementación	66
TEMA 9. EXPERIENCIA DE USUARIO (UX)	69
9.1. Metodología de Diseño de Experiencia de Usuario	71
9.1.1. Etapas	72
9.1.2. Actividades	73
9.1.3. Técnicas	73
9.1.4. Herramientas	73
TEMA 10. EVALUACIÓN HEURÍSTICA	75
10.1. Las 10 reglas heurísticas de Nielsen	76
10.2. Principios de Norman	76
10.3. Las 8 Reglas de Oro de la Usabilidad de Ben Schneiderman	77
10.4. Cómo realizar la Evaluación Heurística	78
GLOSARIO	79
ENLACES DE INTERÉS	81
BIBLIOGRAFÍA	85
Referencias bibliográficas	85
Bibliografía recomendada	86

Leyenda



Glosario

Términos cuya definición correspondiente está en el apartado "Glosario".



Enlace de interés

Dirección de página web.

Introducción

Este curso trata principalmente sobre los elementos que se deben tomar en cuenta para hacer un buen diseño de la interfaz de usuario (**UI**). Es esa parte del sistema informático que interactúa con una persona, usando entrada y salida, y la propiedad que más nos preocupa acerca de esas interfaces de usuario es la **usabilidad**. En el área de la interacción humano-computadora, la usabilidad es el término más común para este campo de estudio.

La interacción humano-computador es un campo multidisciplinario (ver Figura 1) que tiene que ver con la ergonomía del hardware, usabilidad del software, la experiencia del usuario, entre otras disciplinas que tienen que ver con el comportamiento humano, la reacción del humano ante un dispositivo o sistema al momento de interactuar. Este último está muy relacionado con la usabilidad.

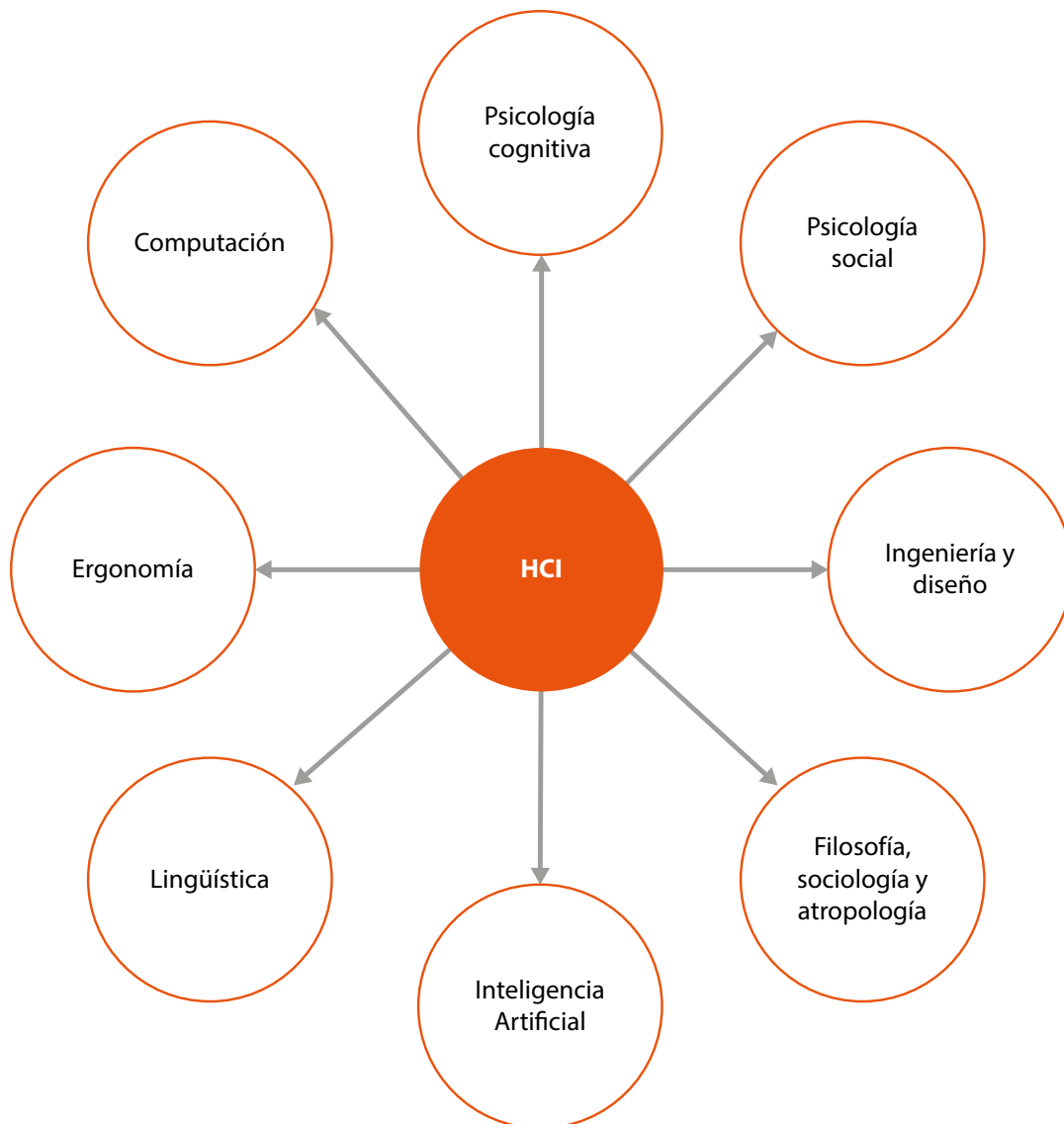


Figura 1. HCI, por sus siglas en inglés, Human Computer Interaction. Fuente: Profesora Adelaide Bianchini, de su curso Interfaces con el Usuario <https://ldc.usb.ve/~abianc/materias/ci4325/>

Las áreas de psicología cognitiva y social, así como lingüística, filosofía, sociología y antropología están muy relacionadas directamente con el comportamiento humano, con las acciones y reacciones de los usuarios frente a un sistema o dispositivo con el cual va a interactuar con un computador. La ingeniería y diseño, así como el área de la computación e inteligencia artificial, se relacionan más bien con las herramientas tecnológicas existentes para desarrollar o implementar las estructuras o modelos de diseño de interacción humano-computador.

Muchas personas adoptan un enfoque más amplio y holístico para pensar sobre el usuario y su relación no solo con interfaces de usuario, sino también con los productos, sistemas y procesos de los que forman parte esas interfaces de usuario. Este campo se llama **diseño de experiencia** del usuario (a menudo llamado **UX**). Esto es más amplio de lo que trataremos en este curso, pero muchos de los principios que discutiremos (diseño centrado en el usuario, diseño iterativo, simplicidad y usabilidad) son generalmente aplicables a UX. De hecho, son aplicables a muchos tipos de diseño, considerados

como el problema general de crear nuevos objetos que resuelven problemas o satisfacen necesidades en ciertos contextos y bajo ciertas restricciones. Hay una moda llamada "pensamiento de diseño" que involucra una perspectiva muy similar a la que tomaremos en este curso: recopilar información, hacer prototipos, hacer mejoras iterativas. El diseño de la interfaz de usuario es un caso especial de diseño.

Dedicaremos una buena parte del curso a definir y estudiar la usabilidad, luego veremos los principios de diseño que afectan positivamente a la usabilidad de los sistemas. Después, describiremos cuáles son esas técnicas de diseño que permiten analizar las tareas, crear prototipos y hacer evaluaciones de usuarios. Finalmente, conoceremos cómo podemos implementar dichas técnicas. Para este curso no hablaremos mucho sobre ergonomía, esto sería tema de otro curso.

Al final del curso se espera que Usted pueda conocer el término usabilidad y sus principios generales, conociendo las capacidades humanas y el proceso de construcción de una interfaz gráfica de usuario (**GUI**).

Tema 1.

Interfaz del Usuario (UI)

La interacción humano-computador se refiere a los medios que se suelen utilizar para interactuar y lograr obtener una comunicación bidireccional entre el humano y el computador. Tiene que ver con el **hardware** y el **software**. En cuanto al hardware, se refiere a los dispositivos de entrada y salida que se pueden utilizar en la interacción: el teclado, dispositivos sensoriales para personas con discapacidad, el ratón, etc. Mientras que, cuando nos referimos al software, se requiere del diseño de una interfaz que permita al humano interactuar con los sistemas computacionales a través de acciones y eventos utilizando los dispositivos de entrada y salida.

Algunos autores señalan que los términos Interfaz de Usuario (UI), interacción (**IxD**) y experiencia de usuario (UX) (ver Figura 2), se entienden como interfaz del usuario indistintamente, sin embargo, cada uno es una parte de la interfaz de usuario con fines distintos. La Interfaz de usuario se refiere más a la forma o presentación de los objetos o elementos de la interfaz que son utilizados para comunicar al usuario con el computador o el sistema que, además, son consistentes con el diseño funcional de la aplicación. Es importante señalar que el diseño gráfico no es diseño de interfaz, una interfaz de usuario puede incluir o no diseño gráfico, el diseño gráfico tiene que ver más con la presentación y estética. La Interfaz de usuario se relaciona más con la manera en la que el usuario comunicará de la mejor forma sus solicitudes de acción al sistema o aplicación, mientras que la experiencia del usuario tiene que ver más con la satisfacción del usuario antes, durante y después de usar la aplicación, está

más relacionado con el usuario. Para lograr una mejor experiencia del usuario es necesario conocer bien al usuario, sus motivaciones y necesidades. La interacción de diseño permite integrar UI y UX y combina la experiencia del usuario con los elementos de Interfaz más adecuado para realizar las acciones pertinentes a la funcionalidad del sistema o aplicación.

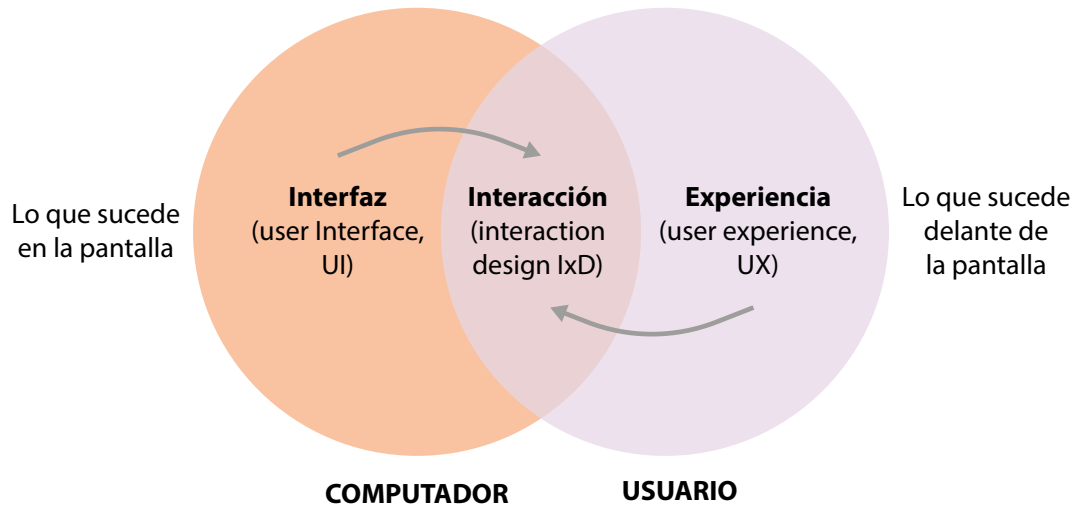


Figura 2. Diferencias entre UI (interfaz de usuario), IxD (interacción de diseño) y UX (experiencia de usuario). Fuente: <http://www.luisan.net/blog/>

La persona encargada del UI se va a centrar en diseñar el sistema o aplicación en función de los elementos de interfaz que ofrezca la herramienta de programación de interfaz que se esté utilizando: formularios, botones, estructuras de diseño visual, colores, tamaños, etc. Estará encargado de la parte visual y puede o no incluir diseño, como se ha mencionado anteriormente. Una interfaz de usuario debe ser sencilla, fácil de usar y fácil de aprender a usarla, lo ideal es que sea intuitiva, atractiva para usuario y, por supuesto, útil. Combinar todas estas características no es una tarea fácil, ni mucho menos es algo que se pueda diseñar de manera rápida, de ahí la importancia de que en el momento de diseñar la interfaz se mantenga comunicación constante, sin ser invasivo, con el usuario final. No necesariamente el usuario final es el cliente.

La persona encargada de UX va a evaluar, no solamente la interfaz de acuerdo a la usabilidad y facilidad de uso, principalmente se va a ocupar un poco más de la funcionalidad y de la percepción de los usuarios mientras usan la interfaz, qué sensaciones les genera, satisfacción, rechazo, etc.

Tema 2.

Usabilidad

El término usabilidad en nuestro contexto se refiere a crear interfaces de usuarios efectivas, usables, útiles. No tiene sentido crear interfaces en un sistema o aplicación que, por más bonitas o visualmente estéticas que sean, los usuarios no quieran usarlas, bien sea porque puede crear confusión en vez de facilitar la interacción o porque les haga perder tiempo intentando entender cómo interactuar con la aplicación. Una presentación bonita de la interfaz no necesariamente implica que haya usabilidad en ella.

Lores, Granollers y Lana (2001) definen usabilidad como “la medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado” (p.16).

2.1. Errores de diseño que afectan a la usabilidad

En el sitio Hall of shame (<http://hallofshame.gp.co.at>) podemos encontrar varios ejemplos de los errores más comunes cometidos en el diseño de interfaces. Ejemplos clásicos en el uso erróneo de los controles o *widgets*, uso de botones del tipo **checkboxes** en vez de usarlos como **radiobutton**. En la Figura 3 se muestra un mal ejemplo de una ventana de una aplicación para una tienda de zapatos para mujeres. Queriendo evitar problemas quizás de exclusión, usaron opciones de selección multiple

(checkboxes) para la selección del género, en vez de usar elementos de selección simple (radiobutton). Lo más interesante es que el lema de la empresa es: "Exclusivamente para mujeres, hecho por mujeres".

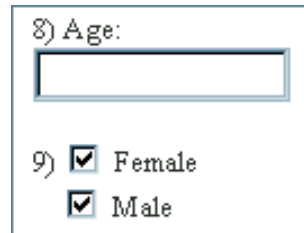


Figura 3. Ejemplo erróneo del uso del widget checkboxes. Fuente: <http://hallofshame.gp.co.at>

Así mismo, podemos encontrar muchos ejemplos del uso incorrecto de elementos interactivos humano-computador. Este tipo de errores, más que facilitar al usuario y hacer que la interfaz sea usable, hacen que este se confunda y no sepa cómo avanzar.

Otro ejemplo (ver Figura 4) es una aplicación que facilita el diseño y creación de certificados personalizados, mostrando una serie de diseños o plantillas, en los que simplemente se puede escoger uno de los diseños con el ratón (mouse) e imprimir. Es bastante sencillo porque no hay que memorizar nada ni escribir comandos complejos y lo que se ve es lo que se recibe (**WYSIWYG-what you see is what you get**), el usuario puede visualizar antes de imprimir. Por el lado de utilidad funcional no habría mayor problema, sin embargo, podemos indicar varios detalles. El mensaje de ayuda que está en el lado izquierdo es demasiado largo y, realmente, no es necesaria tanta explicación para una tarea tan simple como escoger uno de los diseños. Por otro lado, la barra de desplazamiento no ayuda o facilita la búsqueda o elección del diseño, es muy fácil perderse, no existe guía para ir de diseño en diseño, solo se puede mover usando la barra de desplazamiento. No existe información adicional acerca de la cantidad de diseños disponibles, sí existe algún orden, qué tanto se puede desplazar para ir al siguiente diseño. Se hace pesado y poco práctico usar esta interfaz.

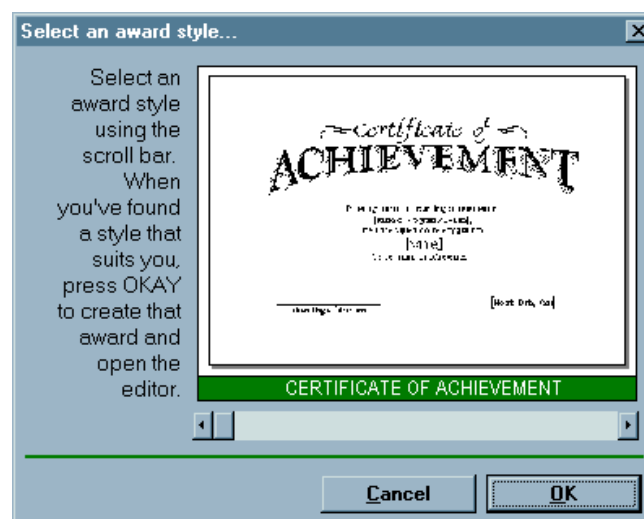


Figura 4. Ejemplo de un mal diseño de interfaz. Fuente: <http://hallofshame.gp.co.at>

Por lo general, una barra de desplazamiento horizontal debajo de una imagen, documento o algún otro contenido está diseñada para ir justamente desplazando el contenido horizontalmente de forma

continua, no de manera discreta. En este caso, no se trata de una película o vídeo o de algo continuo, sino más bien de una muestra discreta en la que se requiere ir de plantilla en plantilla. Debemos tener claro que hay elementos de diseño de interfaz que hablan por sí mismos. En el mundo real, por ejemplo, en relación al pomo de las puertas, todos saben que deben girarse para abrir; en una puerta con manguito, se debe empujar o tirar de ella para abrirla. Lo mismo ocurre con los elementos diseñados para la interacción humano-computador, hay elementos de diseño de interfaz que hablan por sí solos como, por ejemplo, la selección de opciones múltiples del ejercicio anterior u opciones exclusivas, claro que en muchos casos dependerá de la experiencia del usuario.

En el ejemplo visto en la figura 4, un usuario inexperto o poco frecuente ve la barra de desplazamiento, puede que lea la ayuda, asume que funciona y puede que simplemente lo ignore. Un usuario frecuente posiblemente querrá usar una plantilla que ya ha usado antes, pero seguramente no tendrá ni idea del nivel en la barra de desplazamiento donde se encontraba. Por lo que un tercer detalle que presenta este ejemplo sobre un mal diseño de interfaz es el hecho de que no le muestra ningún acceso rápido al usuario frecuente. En este caso, una mejora podría ser que le muestre las plantillas más recientemente seleccionadas, por ejemplo.

En el ejemplo de las puertas con manguito, en algunos casos se tiene el letrero empuje o tire como elemento de ayuda. En la Figura 4 el texto de ayuda a la izquierda podría servir al usuario inexperto, sin embargo, la ayuda dice “*press OKAY*”, y no existe ningún botón que diga *OKAY*, esto crea confusión, siendo otro problema de diseño que afectará la usabilidad del sistema. De hecho, esta interfaz (ver Figura 4) toma lo que debería ser un acceso aleatorio y lo transforma en un proceso lineal. Cada usuario tiene que mirar a través de todas las opciones, incluso si ya sabes cuál quiere. Adicionalmente, por la forma en que se colocó la ayuda, sugiere que el diseñador de la interfaz se dio cuenta del problema y lo resolvió con un parche. Sin embargo, la manera correcta de resolver los problemas de diseño es a través del rediseño, no se puede parchear y resolver.

El diseño de la interfaz debe ser parte del proceso de diseño general del software o de la aplicación. Muchos desarrolladores tenemos problemas con la interfaz, pues pensamos más en la funcionalidad que en la interacción con el usuario y, realmente, es una de las particularidades o atributos de los sistemas el hecho de tener una interfaz que permita una interacción con el usuario que sea simple, sencilla y justamente haga a la aplicación realmente usable para el usuario y que el usuario aprenda y quiera usar la aplicación o el software.

Un buen ejercicio, tal como lo muestra en el mismo sitio en el cual se presenta el problema y la solución, sería ¿cómo podría rediseñarse el cuadro de diálogo de la Figura 4 para resolver algunos de estos problemas de diseño que podrían mejorar la usabilidad? Una manera podría ser colocar la lista con nombres de las diferentes plantillas, en vez de la ayuda a la izquierda, eliminar la barra de desplazamiento y mostrar cada plantilla según la selección del usuario (ver Figura 5). Esto facilitaría la elección, tanto para el usuario experto como para el inexperto.

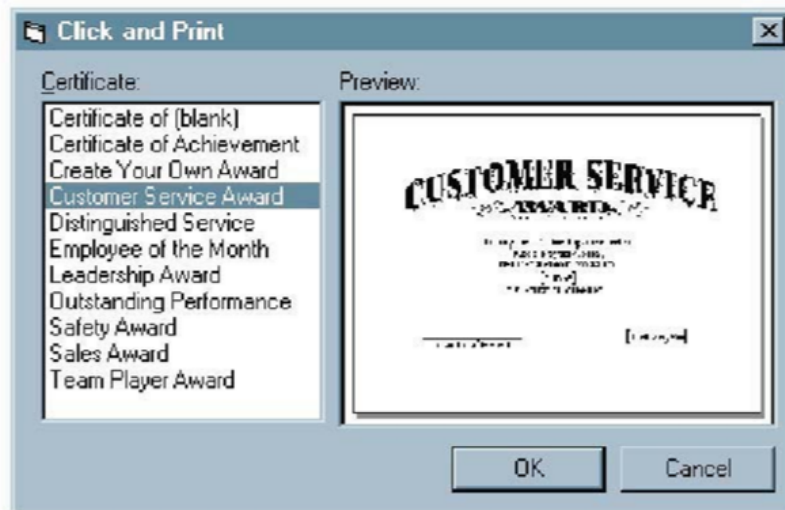


Figura 5. Rediseño de la interfaz de la Figura 4 que mejora la usabilidad de la misma. Fuente: <http://hallofshame.gp.co.at>

Estos ejemplos los podemos encontrar en el sitio <http://hallofshame.gp.co.at>, así como muchos ejemplos más de casos erróneos que muestran el problema de diseño y reducen la usabilidad de los sistemas o aplicaciones. Allí, se muestra una posible solución en algunos casos.

2.2. ¿Por qué es importante la Usabilidad?

El uso de los principios de diseño basados en la usabilidad ha tenido como consecuencia:

- La reducción de los costos de producción.
- La reducción de los costos de mantenimiento y soporte.
- La reducción de los costos de uso.
- La mejora en la calidad del producto.

Como humanos, tenemos la capacidad extraordinaria de aprender y adaptarnos rápidamente. Es posible que aun con la peor interfaz podamos adaptarnos a su uso, sin embargo, no estaremos muy contentos con ello y, seguramente, usaremos dicha interfaz lo menos posible, a no ser que estemos obligados a hacerlo. Un ejemplo, serían las aplicaciones administrativas para oficinistas que no se sientan muy felices de usar dichas aplicaciones y hará que lo usen lo menos posible, afectando entonces así a la usabilidad y productividad.

Por ello, existen razones prácticas por las que es importante tener especial cuidado en el diseño de las interfaces en nuestros sistemas o aplicaciones. La usabilidad afecta fuertemente a la percepción del usuario con respecto al sistema debido a que la interfaz es el medio por el cual el sistema se presenta al mundo. Existen principalmente dos características que afectan a la usabilidad de los sistemas:

1. Fácil de usar
2. Fácil de aprender

El atributo **“fácil de usar”**, tomado en cuenta en un buen diseño de interfaz, afectará positivamente a las recomendaciones de boca en boca, comentarios del sistema e incluso podría influir en las ventas. Hoy en día, podríamos hablar del número de seguidores y usuarios, un sistema es fácil de usar si realiza la tarea para la cual fue diseñado de manera sencilla, eficiente e intuitiva.

Un sistema **“fácil de aprender”** permite al usuario realizar la tarea en el menor tiempo posible (intuitivo) sin mayor complicación (cometiendo pocos errores porque la interfaz te lleva de manera intuitiva) y generando satisfacción en los usuarios. Una aplicación usable es aquella que permite que el usuario se concentre más bien en la tarea que desea realizar que en cómo usar la aplicación.

Hoy en día, existen muchas plataformas de tipo **e-commerce** por lo que, si los usuarios compradores no tienen manera de conseguir rápidamente los productos que están buscando o no entienden muy bien el proceso para adquirir el producto o resulta engorroso, simplemente no lo usarán y se irán a otros sitios.

Como ejemplos de sitios e-commerce tenemos www.mercadolibre.com o las tiendas basadas en la plataforma www.shopify.com o www.magento.com, solo por mencionar algunos ejemplos. Cada uno de esos sitios de e-commerce tiene segmentos de usuarios diferentes, sin embargo, en este caso es necesario tomar en cuenta otros factores ya relacionados directamente con la tecnología de e-commerce, aunque estas cuestiones no son tema de este curso.

Desafortunadamente, la percepción del usuario acerca de la usabilidad del sistema generalmente es algo que pasa desapercibido. Una interfaz atractiva puede parecer **amigable** aun cuando no sea realmente usable. Esto puede deberse a que los usuarios inexpertos pueden sentirse culpables cuando cometen un error al usar los sistemas sin darse cuenta de que seguramente podría resolverse si se crean interfaces **“fáciles de aprender”**. Es así como el término usabilidad es diferente de otros atributos del sistema como son: confiabilidad, rendimiento o seguridad. Si el sistema resulta lento, se cierra de repente o se queda enganchado, generalmente culparemos al propietario del sistema y reportaremos los problemas. Sin embargo, si el sistema no es usable para el usuario, puede no resultar tan grave, sobre todo si son usuarios inexpertos que luego se acostumbrarán y puede que entonces no reporten el problema.

Los diseñadores de software debemos tomar en cuenta varios atributos que se deben cumplir en el momento de desarrollar, algunos de ellos son, por ejemplo:

- Funcionalidad
- Rendimiento
- Costos
- Seguridad
- Usabilidad
- Tamaño

- Confiabilidad
- Estandarización

Generalmente, en el momento de realizar el diseño de la interfaz se debe tratar de equilibrar de alguna manera la presencia de cada uno de estos atributos. El atributo usabilidad no es quizás el más importante y por sí solo no hace un buen diseño de software y, por supuesto, dependerá mucho de la funcionalidad o la razón de ser del software. Por ejemplo, los sistemas bancarios requerirán ser más confiables y seguros, quizás la seguridad en estos sistemas es más importante que la facilidad de su uso. Otro ejemplo, pueden ser los sistemas militares deben ser más seguros que fáciles de usar al inicio. Quizás los astronautas pueden preocuparse más por la confiabilidad de su sistema de navegación que por la usabilidad. De preferencia, en términos generales, se deberían diseñar sistemas que sean confiables, seguros y usables.

Las dimensiones de usabilidad no son uniformemente importantes para todas las clases de usuarios o para todas las aplicaciones. Esa es una de las razones por las que es importante comprender a sus usuarios y saber qué cosas se deberían optimizar. Un sitio web utilizado solo una vez por millones de personas como, por ejemplo, el registro nacional de llamadas telefónicas no solicitadas, tiene una necesidad tan grande de facilidad de aprendizaje y de cero en aprendizaje, que supera con creces otras preocupaciones. Un programa de negociación bursátil usado diariamente por operadores expertos para quienes los segundos perdidos se traducen en dólares perdidos, debe poner la eficiencia por encima de todo lo demás. Pero los usuarios tampoco pueden clasificarse simplemente como novatos o expertos, para algunas aplicaciones (como el comercio de acciones), sus usuarios pueden ser expertos en el dominio, tener un profundo conocimiento sobre el mercado bursátil y, aun así, ser novatos en su aplicación particular. Incluso los usuarios con una larga experiencia en el uso de una aplicación pueden ser novatos o usuarios poco frecuentes en lo que respecta a algunas de sus funciones.

Es así como podemos aplicar algunos principios para mejorar la usabilidad de los sistemas. Estos sistemas los podemos ver en el siguiente tema.

Tema 3.

Principios Generales de la Usabilidad

Según Dix (1993), existen algunos principios generales para mejorar la usabilidad de los sistemas y se describen a continuación.

- 1.** Facilidad de Aprendizaje. Como comentamos anteriormente, un sistema fácil de aprender se mide en función de la cantidad de errores cometidos, la satisfacción del usuario y el tiempo de realización de la tarea para la cual está diseñada el sistema. Un sistema fácil de aprender debe ser según Dix (1993):
 - a.** Sintetizable. Que el usuario sea capaz de darse cuenta de los cambios de estados en el proceso de ejecución de las tareas.
 - b.** Familiar. Tiene que ver con la correlación de otras aplicaciones, es posible que, si el usuario es experto, sea muchísimo más fácil relacionar las interfaces por su experiencia con otras aplicaciones y, si es inexperto, igual pueda aprender a usar rápidamente el sistema.
- 2.** Consistencia. Esta característica tiene que ver con los mecanismos de estilo, presentación o formas que son usados en el sistema. Para ello, se recomienda tener en cuenta lo siguiente:

- a. Seguir una guía de estilo previamente diseñada, de esta manera si diferentes desarrolladores trabajan en el sistema, pueden mantener el mismo estilo y no que cada uno imponga su estilo particular.
 - b. Diseñar un mismo “**look and feel**”.
 - c. Hacer modificaciones necesarias.
 - d. Añadir nuevos elementos en vez de modificar las ya existentes, manteniendo el mismo estilo entre versiones del sistema.
3. Flexibilidad. Tiene que ver con la manera en que el usuario y el sistema intercambian información (Input/Output), para lo cual, el sistema debe cumplir los siguientes parámetros:
- a. Darle el control al usuario. Que el usuario pueda cancelar o editar o modificar cuando él así lo requiera, darle la posibilidad de deshacer alguna actividad o modificación y recuperar lo que recientemente se haya perdido por alguna razón.
 - b. Migración de tarea. Permitir que el usuario pueda compartir con el sistema la realización automática de tareas. Un ejemplo de esto son los correctores automáticos de los editores, es una tarea que es compartida por el sistema, pero el usuario tiene el poder de aceptarla o modificarla.
 - c. Capacidad de sustitución. Este parámetro tiene que ver con la capacidad de utilización de diferentes sistemas de medición, usar centímetros o pulgadas, por ejemplo, y que el usuario pueda cambiar estos atributos sin mayor problema.
 - d. Adaptabilidad. Tiene que ver con la capacidad de adecuación automática del sistema al usuario, reconociendo si se trata de un usuario experto o novato.
4. Robustez. Tiene que ver más con la funcionalidad del sistema, si realmente hace las tareas que se requieren y el soporte de las mismas.
5. Recuperabilidad. Tiene que ver con la capacidad de corregir una acción una vez reconocido el error por el usuario. Para ello, se puede tener una estrategia para prevenir posibles errores de usuario.
6. Tiempo de respuesta. Es importante que los tiempos de respuesta de los procesos involucrados en las distintas tareas sean soportables por el usuario, esto puede variar dependiendo de la experiencia del usuario en el uso de los sistemas.
7. Adecuación de tareas. Este parámetro tiene que ver con cómo se comprenden las distintas tareas que ejecute el usuario y la capacidad de cambiar de ejecución de las tareas.
8. Disminución de la carga cognitiva. No dejarle al usuario tener que aprender cómo va ejecutando las tareas o tener que memorizar códigos, etc.

La siguiente sección habla un poco de la capacidad de aprendizaje que pueden ofrecer las interfaces a través del uso de las metáforas. Las metáforas constituyen una forma de permitir que el usuario aprenda de manera más fácil.

3.1. Facilidad de Aprendizaje

Una interfaz que le permita al usuario inexperto aprender rápidamente a usar la aplicación o el sistema y a los usuarios de uso frecuente recordar los procesos es una interfaz que mantiene el atributo de facilidad de aprendizaje. Para ello, debemos hablar entonces acerca de la ciencia cognitiva, observando cómo funciona la mente humana. Luego veremos cómo han ido evolucionando los diseños de interfaces gráficas de usuario *GUI* (por sus siglas en inglés, *Graphic User Interfaces*) desde el punto de vista del aprendizaje. Examinaremos tres estilos que se han usado durante mucho tiempo y que aún se siguen usando. Hablaremos de cómo los usuarios aprenden sobre una interfaz formando un modelo mental de sus partes y sus comportamientos. Finalmente, hablaremos de algunos principios de diseño que se pueden aplicar si la capacidad de aprendizaje es un atributo importante en el momento de hacer el diseño de la interfaz.

Antes, debemos ser conscientes de que la gente no aprende instantáneamente, tampoco podemos esperar que los usuarios recuerden instrucciones si las mismas son demasiado explicativas o confusas (ver Figura 6).

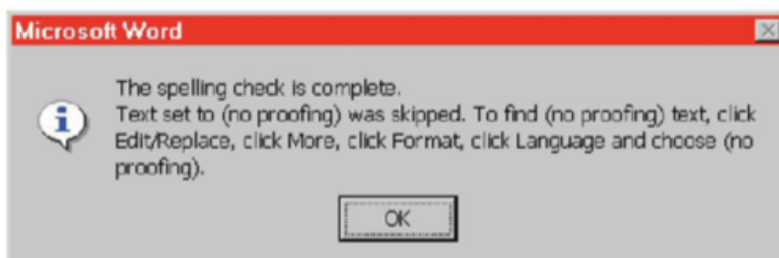


Figura 6. Ejemplo de un modelo de diálogo difícil de memorizar y aprender (© Microsoft. Derechos reservados).

Fuente: <http://hallofshame.gp.co.at>

El cuadro de diálogo de la figura 6 es un cuadro de diálogo modal, por lo que el usuario no puede seguir sus instrucciones hasta después de hacer clic en OK, pero luego las instrucciones desaparecen de la pantalla y el usuario seguramente no podrá recordarlo. Una solución obvia a este problema sería un botón que simplemente ejecutara las instrucciones directamente. Este mensaje podría ser claramente un parche de última hora para resolver un problema de usabilidad.

En nuestra área, debemos tener en cuenta las propiedades de los sistemas computacionales tales como: velocidad del procesador, tamaño de la memoria, disco duro, sistema operativo y la interacción entre sus componentes. La interfaz de usuario es el medio por el cual se comunica el humano con el computador, así que es importante conocer ciertos aspectos del ser humano para poder hacer una buena interacción humano-computador. En cuanto a este tema, hay mucho más de lo que podemos dar en este curso, así que solo destacaremos algunos aspectos importantes que deben ser tomados en cuenta en el momento de realizar un diseño de interfaces en un sistema o aplicación.

3.1.1. Mente Humana

Uno de los aspectos importantes a tener en cuenta para el diseño de interfaces es la mente humana y su capacidad de memorizar o guardar trozos o eventos en la memoria. Tiene dos componentes:

1. Memoria operativa o de trabajo
2. Memoria a largo plazo

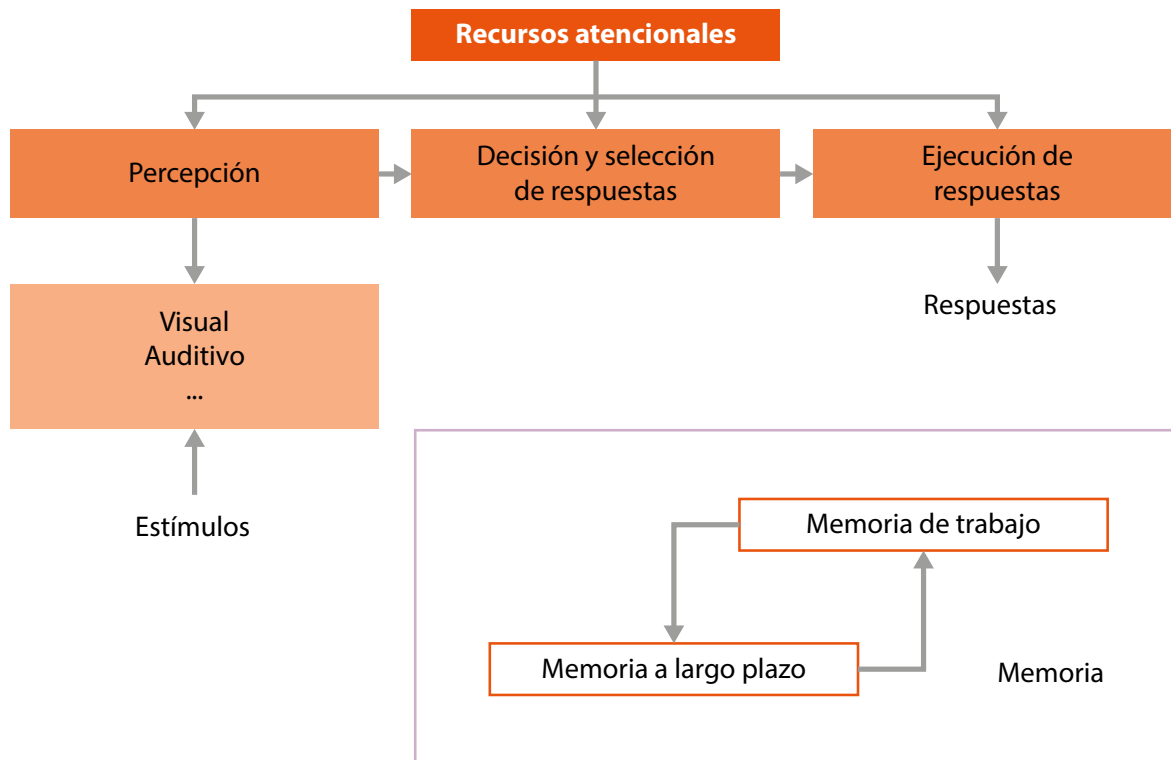


Figura 7. Modelo cognitivo General. Wickens C. D. (1992).

Estos dos componentes se comportan de manera muy diferente de la memoria de la computadora, y el aprendizaje (el proceso de poner información y procedimientos en la memoria a largo plazo) no es un proceso de almacenamiento simple. La memoria operativa es donde haces tú pensamiento consciente. La ciencia cognitiva sostiene que la memoria operativa o de trabajo no es en realidad un lugar separado en el cerebro, sino más bien un patrón de activación de elementos en la memoria a largo plazo. Se dice que la capacidad de la memoria de trabajo es de aproximadamente 7 ± 2 cosas (técnicamente llamado trozos). Se puede decir que los trozos son el elemento de la memoria que puede percibir la mente humana. Son trozos muy pequeños que se guardan en la memoria de trabajo, que puede aumentarse con la práctica si el usuario, conscientemente, utiliza técnicas mnemotécnicas que convierten datos arbitrarios en fragmentos memorables. Sin embargo, no se puede esperar a que el usuario haga eso, una buena interfaz no debe exigir mucho ni contar con la memoria temporal o de trabajo del usuario.

La información que se almacena en la memoria temporal desaparece en poco tiempo, segundos, tal vez. Las distracciones pueden simplemente eliminar todo lo que en algún momento haya llegado a

la memoria temporal o de trabajo y la práctica de repetición puede evitar esto, sin embargo, requiere de atención.

La memoria a largo plazo, en cambio, tiene una capacidad, si se quiere, infinita y muestra poca descomposición. Los recuerdos a largo plazo no son borrados tan fácilmente ni aun intencionalmente, generalmente se vuelven inaccesibles. El mecanismo usado en este caso para mantener estos recuerdos memorables parece ser un mecanismo elaborado, que realiza conexiones con trozos existentes. Utiliza técnicas mnemotécnicas para asociar cosas que necesita recordar con eventos familiares, sucesos de la infancia. Sin embargo, estas técnicas requieren de mucho trabajo y atención por parte del usuario. Una clave para tener esa capacidad de aprendizaje es hacer conexiones tan fáciles como sea posible.

3.1.2. Reconocimiento vs Recuerdos

Es importante hacer la distinción entre el reconocimiento y el recuerdo. El reconocimiento tiene que ver con recordar algo con la ayuda de alguna pista que se asocia con el conocimiento del mundo de la persona. El recuerdo tiene que ver con recordar algo sin la ayuda del mundo exterior, es algo que viene de tu cabeza. Es mucho más fácil recordar un menú (por reconocimiento) que unas líneas de comando, por ejemplo.

Podemos pensar en la capacidad de aprendizaje (y la visibilidad y la eficiencia) en términos de un modelo cognitivo generalizado de cómo las personas usan un sistema computacional, tema tratado por primera vez por Donald Norman (2002) en *"The Design of Everyday Things"*, un libro muy recomendado.

Una interfaz fácil de aprender tendría pequeños abismos en referencia a la ejecución y evaluación del sistema, de modo que los objetivos del usuario estarían más cerca de la representación del sistema en cierto sentido: más fácil de ejecutar y más fácil comparar.

La capacidad de aprendizaje no es la única propiedad de usabilidad que se puede entender con este modelo. Visibilidad y retroalimentación se refieren más a la evaluación del sistema. La eficiencia es una medida de todo el ciclo: la velocidad de ejecución y percepción.

3.1.3. Estilos de Interacción que afectan a la capacidad de Aprendizaje

Que las interfaces sean fáciles de aprender ha sido uno de los mayores objetivos en la evolución de las GUI en los últimos años. Podemos mencionar tres tipos de estilos de interfaces gráficas usados en computadores de escritorio (conteniendo, mínimo, un computador, un teclado y un ratón (mouse)), que han sido usados hasta ahora. Estos estilos han evolucionado hacia un mayor uso de la capacidad de aprendizaje por el uso de los mismos en los sistemas.

Por líneas de comando:

El uso de líneas de comando, desde el **prompt de CMD de Windows** o el **Shell de Linux**.

- El shell de UNIX

```
$ ls -la
$ mv mmail
```

- MSDOS

```
C:>pwd
C:>ls -la *.c
C:>dir *.c 1)    > ls -s *.ini
```

- El uso de una dirección URL, por ejemplo, www.google.com
- **Query** en motores de búsqueda

Se puede decir que fueron las primeras formas o estilos de interfaces de usuario. A pesar de ello, sigue siendo utilizado, especialmente por los usuarios expertos, luego, estos estilos evolucionaron a usar más bien menús y formularios. La interfaz por línea de comandos es la forma más directa de darle instrucciones al computador. Sin embargo, requiere conocer un lenguaje y una serie de comandos para poderlos utilizar eficientemente.

Menús y Navegación:

Una interfaz con menús y formularios o diálogos fue lo más avanzado de la Web 1.0. Todavía, muchas interfaces gráficas mantienen este tipo de interacción, por ejemplo: la barra de menú (la cual puede implicar árboles de menús) y las cajas de diálogos (pequeños formularios).

Manipulación Directa:

Los usuarios interactúan con la representación visual de los objetos. La manipulación directa se define por tres principios:

1. Representación visual continua. Iconos representando archivos, carpetas representando directorios en el escritorio del computador, por ejemplo. La representación visual continua puede ser verbal o icónica, pero está presente constantemente, y no cuando se solicite.
2. Acciones físicas o botones. Una acción física podría ser hacer clic sobre un objeto y moverlo de un lado al otro, cambiar su tamaño, por ejemplo. Así mismo, se puede permitir ejecutar acciones por comando presionando sobre algún botón visible.
3. Los efectos de las interacciones deben ser rápidos (visibles tan rápido como sea posible), incrementales (se pueden ver cambios incrementales, el tamaño por ejemplo o el uso de uno o varios objetos), reversibles (poder deshacer alguna acción) y ser inmediatamente visibles.

La manipulación directa explota las capacidades motoras en la interacción humano-computador, depende menos de habilidades lingüísticas que por interfaces con comandos o menú/diálogos,

siendo una interacción más natural en el sentido de que aprendemos a manejar el mundo antes de aprender a hablar, leer o escribir.

Comparación de estilos de Interacción:

Los lenguajes de comandos requieren más aprendizaje, requiere que el usuario aprenda mediante lectura, práctica y entrenamiento. Los menús y formularios y la manipulación directa son más fácil de aprender que los estilos de lenguajes de comandos, son más intuitivos y requieren más conocimiento del mundo que conocimiento adquirido por el usuario.

3.1.4. Modelo de Usuario vs Modelo de los Sistemas

Modelo de un sistema. Se refiere a cómo funciona el sistema. Un modelo describe las partes del sistema y cómo esas partes interactúan para hacer lo que se supone que hace el sistema para lo que fue creado.

Modelos de interfaces. Es el modelo que presenta el sistema para interactuar con el humano. Estos deben ser simples, apropiados, refleja el modelo de usuario para la tarea por la cual fue creado el sistema y debe comunicar bien el resultado de las acciones.

Modelo del usuario. Se refiere a cómo el usuario cree que funciona el sistema.

	Por líneas de comando	Menús y Formularios	Manipulación Directa
Fácil de aprender	Requieren mucha más atención al aprendizaje por medio de la lectura, práctica y entrenamiento	Son más intuitivos Usan más el conocimiento del mundo que el conocimiento aprendido por el usuario	
Mensajes de error	Generalmente, se generan frecuentemente mensajes de error por omisión de pasos del usuario		
Eficiencia	Los usuarios expertos pueden ser muy eficientes usando lenguaje de comandos	Requiere de definición de atajos	Solo si la manipulación directa es apropiada para la acción que se desea
Tipos de usuario	Generalmente, funciona muy bien con usuarios expertos	Es mejor para los usuarios novatos o esporádicos	
Sincronía	Son estilos síncronos. Se escribe el comando, luego se ejecuta y el sistema responde. En el caso del diálogo, primero es completado el diálogo, luego ejecutado y el sistema da su respuesta		Es una acción asíncrona, se manejan más bien por eventos del ratón (mouse)

	Por líneas de comando	Menús y Formularios	Manipulación Directa
Dificultad de programación	Es relativamente fácil de programar.	Requiere de una herramienta de soporte	Es bastante complicado de programar. Se debe manejar la ocurrencia de varios eventos constantemente.
Accesibilidad	Son más visibles, por lo tanto más accesibles.		Es más difícil para el usuario darse cuenta de este estilo de interfaz, a menos que sea un usuario experto.

Tabla 1. Tabla comparativa de los distintos estilos de interfaces en función de varias características de las interfaces. Fuente: Elaboración propia

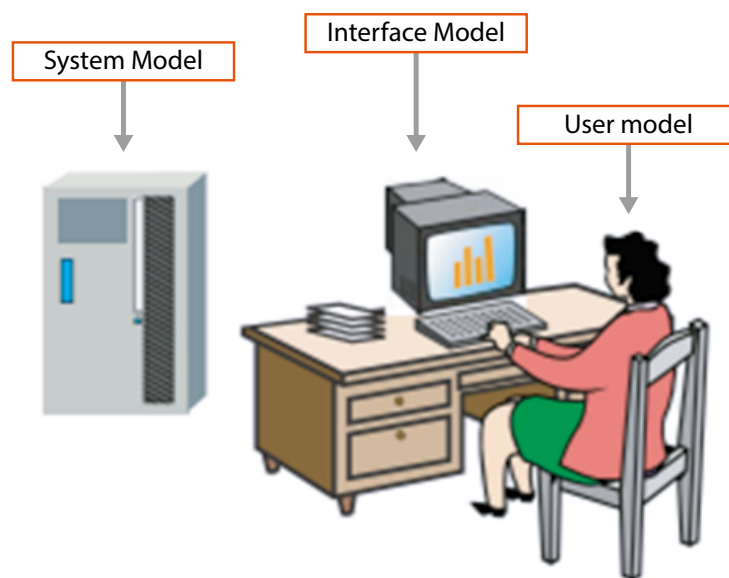


Figura 8. Modelos del sistema y el modelo de usuario. Fuente: MIT OpenCourseWare.

3.1.5. Principios de la facilidad de aprendizaje vs Diseño de Patrones

En términos generales, Donald Norman (2002) presenta dos principios claves que permiten realizar diseños de interfaces fáciles de aprender. El primero, tiene que ver con elementos de comunicación entre el modelo de usuario y el modelo del sistema y que, generalmente, se usan para la interacción con los objetos físicos. Por ejemplo, el pomo de la puerta tiene un tamaño que permite ser manipulado por la mano de una persona adulta, la forma de la tijera. Estos objetos permiten mostrar el modelo mental de cómo funcionan. El segundo principio tiene que ver con la consistencia, las interfaces de usuario son más fáciles de aprender si le son familiares a los usuarios. Para cumplir con estos principios se deben tener en cuenta varias características.

Affordance:

Esta palabra en español se traduce como ofrecimiento y es manejado como tal por algunos autores en esta área de interacción hombre-computador, sin embargo, no refleja su significado real. Es un término

más bien usado en psicología cognitiva, por lo que preferiría usar el término en inglés directamente en este contexto. El término *affordance* se refiere a la facilidad de descubrir cómo ejecutar una acción en la interfaz (Donald Norman, 2002). Esta característica percibe las propiedades de los objetos que determinan cómo funcionan. En el diseño gráfico de las interfaces (GUI), por ejemplo, existen objetos que se describen por sí solos, el usuario al verlos sabe cómo funcionan. Generalmente, el usuario al ver un botón sabe que, para hacerlo funcionar, el botón debe ser presionado. Lo mismo pasa con las barras de desplazamiento, al verlas el usuario sabe que debe desplazarlas con el ratón de forma continua para visualizar el contenido.

El término *affordance* en GUI es la forma en que una interfaz se comunica de forma no verbal con el usuario, es decir, la forma de cómo operar. Claro, que no es algo innato en el usuario y se adquiere con la experiencia. Lo mismo sucede con los objetos, sabemos que una silla es para sentarse por experiencia, un bebé no lo sabe hasta que, por experiencia, aprende que la silla es para sentarse. Ahora bien, hay dos formas de mostrar esta característica: una es lo que se percibe al ver el objeto y, otra, lo que realmente hace. Por ejemplo, la puerta con una manija puede halarse o empujarse para abrir, si no se coloca una señal de ayuda, todos cometerán el error de empujar en vez de halar o viceversa, según sea el caso. Es así como entonces se debe tener en cuenta lo que realmente hace.

Mapeo Natural:

El mapeo natural tiene que ver con el arreglo físico de los controles con respecto a las funciones, es decir, si tengo un área de contenido de texto y el texto excede del espacio asignado, lo más natural es que se tenga una barra de desplazamiento del lado derecho para poder desplazarse en el contenido del texto. No es natural colocarlo debajo ni por encima del área de texto.

Visibilidad:

La visibilidad es un principio esencial, probablemente el más importante, para comunicar un modelo al usuario. Si el usuario no puede ver un control importante, debe entonces adivinar que existe y donde está. Las áreas y controles relevantes deben ser visibles para el usuario. Se requiere de un esfuerzo importante para hacer visibles las partes de una interfaz de computadora que dejarlas invisibles. Por ejemplo, arrastrar y soltar objetos en una interfaz gráfica es un control invisible y dependerá mucho del nivel del usuario. Sin embargo, esta técnica tiene tan poca visibilidad que muchos usuarios simplemente no se dan cuenta de cuándo arrastrar y soltar es posible, y cuándo no lo es. Como resultado, esta increíble técnica de manipulación directa es, a menudo, secundaria, un acceso directo utilizado solo por usuarios expertos que lo conocen, mientras que otros son menos utilizables (a menudo, la interfaz de estilo y forma de estilo) es utilizada por la mayoría de usuarios principiantes y casuales.

Capacidad de Respuesta o Feedback:

El principio último de una comunicación a través de la interfaz usuario-computador es la capacidad de respuesta del sistema o feedback, y tiene que ver con lo que el sistema hace cuando el usuario ejecuta una acción. Cuando el usuario realiza satisfactoriamente una acción, se debe obtener una respuesta. No siempre debe ser visual, puede ser auditiva o de tipo háptica, utilizando, por ejemplo, la vibración del ratón o en una pantalla táctil, en la cual presionar suficientemente puede indicar una

función, como hacer clic en el segundo botón del ratón. Es importante tener en cuenta también el tiempo de respuesta que se obtenga del sistema, aun cuando sea quizás algo más relacionado con la funcionalidad del mismo.

Consistencia:

Cosas similares deben verse y actuar de manera similar. Así mismo, diferentes cosas deben ser visiblemente diferentes. Hay tres tipos de consistencia:

- Consistencia interna, dentro del mismo sistema.
- Consistencia externa, con otras aplicaciones en la misma plataforma.
- Consistencia metafórica, con la utilización de metáforas que asemejan objetos del mundo real.

Consistencia de diseño. Tiene que ver con dónde se muestran los controles en la pantalla. Consistente es que el menú principal aparezca en la parte superior izquierda, como generalmente funcionan todas las aplicaciones. Mostrarlo de manera diferente le llevará un tiempo al usuario aprender a usarla.

Consistencia de términos. Es importante mantener los mismos términos, mismos conceptos o nombres de los objetos. Usar nombres diferentes de los objetos en partes diferentes de la interfaz confunden al usuario. Esto tiene que ver también con el lenguaje que se use para comunicar al usuario, si se usan palabras simples, sencillas y no muy técnicas, de tal manera que no se forje al usuario a aprender nuevas palabras, cuando se podrían usar palabras más comunes.

Los programadores tendemos a usar sustantivos y pocos verbos, el lenguaje que se use debe ser entendido y bien escrito. Si se trabaja con distintos idiomas, cada diálogo en los distintos idiomas debe estar bien escrito. En cuanto a este tema, existen laboratorios de investigación trabajando en la adaptación de los sistemas a culturas e idiomas distintos, aún es un problema abierto por resolver en el área de diseño de interfaces de usuario gráficas (GUI), hacer el reconocimiento automático del lenguaje y la cultura.

Una buena estrategia para mantener la consistencia en los sistemas es tomar de muestra la interfaz utilizada por plataformas estándar, tales como las guías de interfaces humanas de Apple, Windows Vista, GNOME, KDE, entre otros.

Metáforas:

Las metáforas son una forma de usar objetos del mundo real para ser usadas en la interfaz. Un ejemplo clásico es mostrar un icono con la puerta de salida para indicar que, haciendo clic allí, se sale del sistema. También, el uso de un cubo de basura para indicar que si se quiere borrar un objeto, se puede arrastrar hacia la basura y así ser eliminado. Las ventajas y desventajas se muestran en la Tabla 2.

3.2. Visibilidad

Tal como se ha comentado anteriormente, la visibilidad es una característica importante de las interfaces gráficas del usuario dado que el usuario no puede ver o tiene que adivinar qué existe y dónde

está. Este tema de la visibilidad tiene que ver mucho con la conducta cognitiva del individuo, como el modelo de atención, la percepción, implicaciones prácticas de los tiempos de respuesta y de las respuestas como tal, entre otros factores. Un buen diseño de interfaz de usuario requiere comprender las propiedades y limitaciones de los seres humanos y es tan importante como entender cómo programar la computadora. Aunque no es tema de este curso, más adelante iremos mencionando algunos de los aspectos más importantes relacionados con la conducta cognitiva del individuo.

Ventajas	Desventajas
Fácil de aprender cuando se usa adecuadamente	Es difícil para el diseñador encontrar la metáfora que mejor cale en la acción
Encaja en el modelo mental del usuario muy fácilmente	Puede ser engañoso
	Puede ser limitante
	La metáfora se pierde en algún momento
	El uso de las metáforas es excusa de los malos diseños

Tabla 2. Ventajas y Desventajas del uso de Metáforas. Fuente: Elaboración propia.

3.2.1. Acciones Visibles

Las acciones visibles son las acciones que el usuario puede hacer en la interfaz. Anteriormente, hemos hablado acerca del término *affordance*, el cual tiene que ver con las propiedades reales y percibidas de un objeto que indican cómo debe ser operado. Si el usuario no puede percibir cómo funciona un elemento gráfico de la interfaz, no se podrá comunicar de manera efectiva con el sistema. Un ejemplo de ello son los **hiperlinks** y los botones. Los botones son un objeto metafórico del mundo real pero los *hiperlinks*, a pesar de que no hacen ninguna referencia metafórica, ambos objetos llevan al usuario a la acción, el usuario sabe qué hacer cuando los visualiza, claro que depende mucho de la experiencia del usuario. Otro ejemplo es el cambio de la forma del apuntador del ratón cuando pasa sobre algún objeto, el cambio de forma podría indicar que hay una acción referente al objeto como, por ejemplo, obtener mayor información.

3.2.2. Estados Visibles

Forma en que se visualiza el estado actual del sistema.

3.2.3 Feedback

La visibilidad no siempre es visual, puede ser auditiva o háptica. El feedback auditivo tiene que ver con la forma en que se muestra el resultado de ejecutar una acción por parte del usuario.

3.3. Eficiencia

En este caso, la eficiencia no tiene que ver con la funcionalidad de los algoritmos detrás del procesamiento de los datos del sistema, de las tareas que el sistema deba realizar y para lo cual fue

creado, nos referimos a eficiencia en términos de los canales de comunicación que se están usando en el diseño de la interfaz gráfica, en la interacción usuario-computador, es decir, con qué rapidez puedo obtener información e instrucciones acerca de la interfaz o con qué rapidez puedo llegar a entender cómo usar el sistema. Son preguntas que podemos responder, pero primero debemos entender cómo el humano puede procesar la información, conocer el sistema cognitivo humano. A partir de allí, podremos mencionar los principios prácticos y patrones de diseño para hacer una interfaz más eficiente.

3.3.1. Procesamiento humano de la Información

Card, Moran y Newell (1983) presentan un modelo cognitivo usado para calcular cuánto le cuesta al humano hacer una tarea usando el computador. Relaciona mucho la computadora con la mente humana, es por eso que en el modelo se pueden ver procesadores y memoria. La Figura 9 es una versión modificada del modelo realizada y divulgada por el MIT (Curso de Interfaz de Usuario Gráfica - <https://ocw.mit.edu>). El modelo describe diferentes tipos de memoria y procesadores.

La memoria de corto plazo (*Short-term sensory store*) es el primer sitio de almacenamiento con los primeros estímulos que recibe el usuario del computador, estímulos que se reciben a través de los sentidos (*senses*), estímulos visuales, auditivos o hápticos, según cual sea el caso. **El procesador perceptual** (*perceptual processor*) recibe las entradas sensoriales almacenadas e intenta reconocer los símbolos: caracteres, palabras, imágenes, iconos, fonemas, sonidos. Ayudado, por supuesto, por lo que se tiene almacenado en la **memoria a largo plazo** (*Long-term memory*), en la cual se encuentran almacenados por la experiencia del usuario todos estos símbolos y es capaz de reconocerlos. **El procesador cognitivo** (*cognitive processor*) toma los símbolos reconocidos por el procesador perceptual, hace comparaciones y toma decisiones. El procesador cognitivo puede coger y colocar información en la **memoria operativa o de trabajo** (*working memory*), la cual es una memoria a corto plazo también. **El procesador motriz** (*motor processor*) recibe una acción del procesador cognitivo y ordena a los músculos ejecutar la acción. Hay un *feedback* implícito entre los sentidos y los músculos, dado que los sentidos observan y reciben nuevos estímulos. **La atención** (*attention*) sería como el componente que mantiene el control sobre los procesadores humanos de la información y mantiene la atención y concentración en las tareas que se ejecuten. Cada uno de los procesadores descritos tiene un ciclo de tiempo que varía de persona en persona y las condiciones externas. Por ejemplo, se estima que estos tiempos están en los rangos siguientes:

Tp = Tiempo de procesamiento perceptual ~100ms [50-200 ms]

Tm = Tiempo de procesamiento motor ~70ms [25-170 ms]

Tc = Tiempo de procesamiento cognitivo ~70ms [30-100 ms]

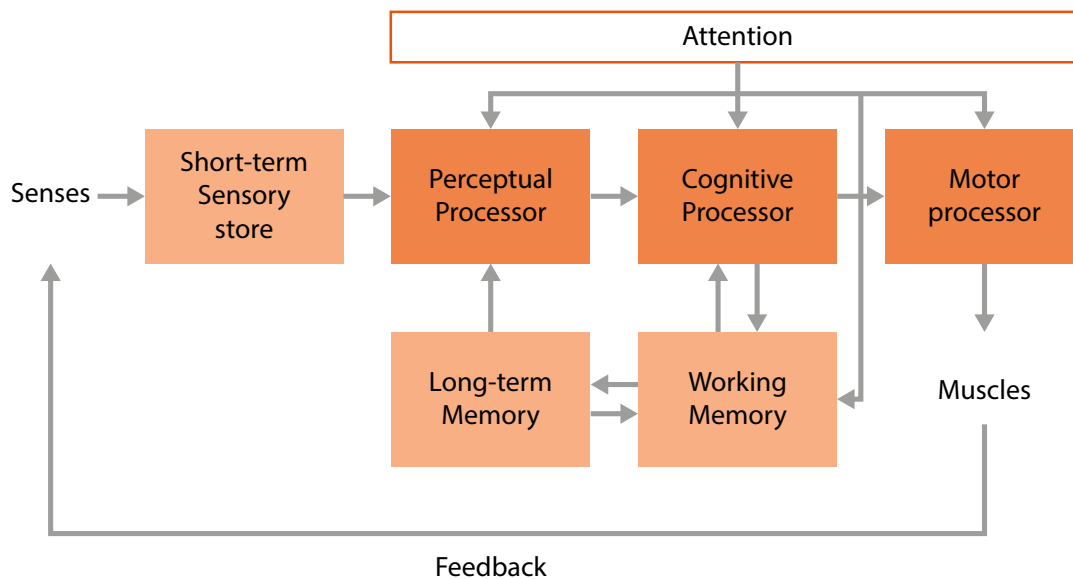


Figura 9. Modelo del procesador de información humano definido por Card et al. (1993). Fuente: Elaboración Propia.

Estos tiempos son estimados de esta manera, sin embargo, pueden variar dependiendo del individuo o de las condiciones. Los tiempos de respuesta motora o perceptuales suelen ser diferentes cuando se está conduciendo o jugando a videojuegos que cuando se está leyendo, por ejemplo, así como los tiempos del procesador cognitivo. Dentro del procesador perceptual podemos encontrar el efecto que se denomina fusión perceptual, y esta se da cuando se relacionan dos acciones o dos estímulos recibidos en instantes de tiempos distintos. Este efecto puede afectar a la percepción de casualidad, por ejemplo, si en la interfaz un evento ocurre después del otro, es posible que pensemos que el primer evento o acción originó el segundo, y no siempre es así.

3.3.2. Principios prácticos y patrones de diseño para hacer una interfaz más eficiente.

Mejorando el uso del ratón (mouse):

Para mejorar la acción de tener que mover con el ratón objetos de un lado al otro, es recomendable tener los objetos relacionados en la acción lo más cerca posible.

Atajos por teclado – Shortcuts:

Usar combinaciones de teclas para realizar una acción sin tener que ir al menú utilizando el ratón y así reducir el tiempo para ejecutar una acción (Ctrl+S, Ctrl+N, Ctrl+Q, etc.), como se muestra en la figura 10. El acceso por teclado afecta positivamente a la accesibilidad del sistema. Otra manera de hacer interfaces más eficientes es agrupar acciones de tal manera que en una misma acción se pueda hacer más de una.

Por defecto, histórico, autocompletación:

Tener disponible configuraciones de acciones por defecto, dejando activas las más usada por los usuarios o mantener un histórico de los últimos documentos utilizados, como se tiene en aplicaciones

como Word de Microsoft, por ejemplo. La figura 10 es un ejemplo de histórico en la aplicación Inkscape. Auto-completar facilita bastante también la tarea de introducir textos completos, aunque a veces resulta molesto, por lo que debemos tener cuidado con esto.

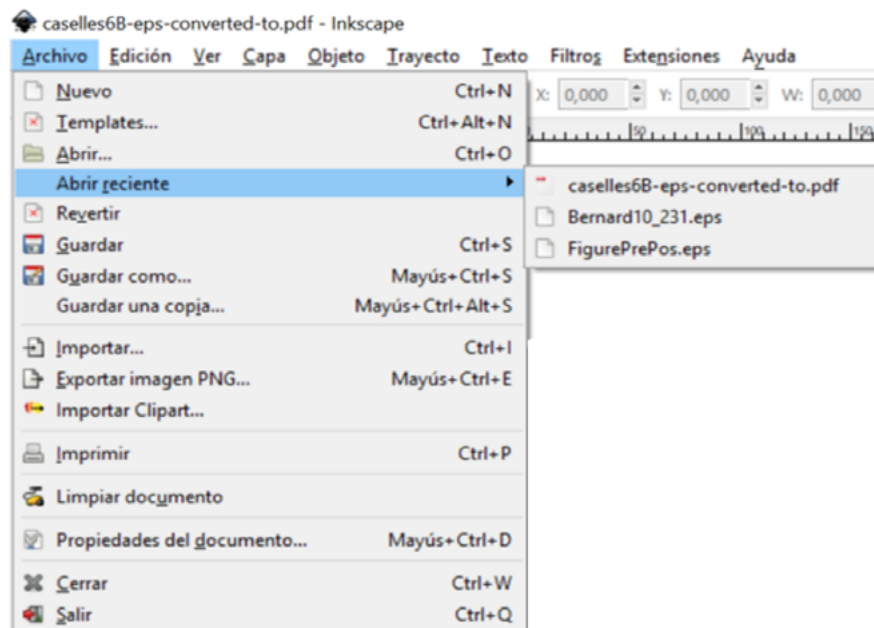


Figura 10. Muestra del uso de Shorcuts del teclado en la aplicación Inkscape. Adicionalmente, muestra también la lista de histórico de los archivos más recientemente usados. Fuente: Elaboración propia.

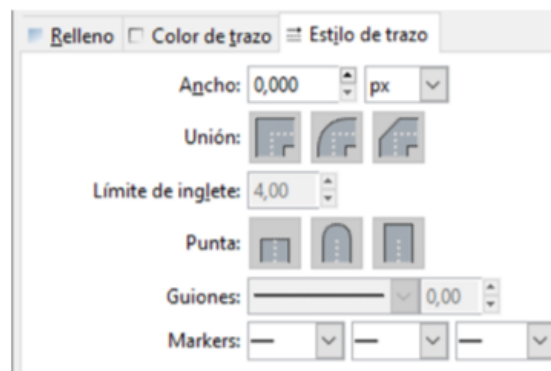


Figura 11. Muestra del uso de agrupamiento de acciones de la aplicación Inkscape (programa opensource). Fuente: Elaboración propia.

Anticipación:

Anticiparse en una interfaz es permitir en un mismo espacio tener toda la información y herramientas necesarias al alcance del usuario. Por ejemplo, la ventana de apertura de archivos de Windows. Anticipa poder cambiar de directorio, tener la lista de los tipos de archivos para guardar el archivo o visualizar un mismo tipo de extensión de archivo, tener los botones abrir, cancelar, etc., es decir, todo lo necesario para ejecutar la acción que es abrir archivo. En la figura 11 se observa la muestra de la aplicación *inkscape* para agrupar opciones en un mismo espacio relacionado.

3.3.3. Evaluación Predictiva

La evaluación predictiva permite usar la psicología cognitiva del humano para predecir la usabilidad del sistema, en este caso, predecir la eficiencia. Nos permite predecir la eficiencia del diseño de la interfaz antes de construirla. La técnica de predicción requiere de un modelo abstracto, de cómo el usuario interactúa con el sistema. El modelo debe ser cuantificable, podremos usar, por ejemplo, las estimaciones de tiempo que requiere cada procesador, cognitivo, motor y perceptible. Aproximar estos valores usando experimentos de usuarios con otras aplicaciones, por ejemplo, menú en cascada o diferente uso de objetos de interfaces. Entre las ventajas de hacer una evaluación predictiva se encuentran:

- No requiere usuarios reales.
- No es necesario construir un prototipo inicial de Interfaz del usuario (UI). Se pueden comparar distintos diseños antes de implementarlos.
- La teoría provee explicación de los problemas de UI, así con una evaluación predictiva, estaremos apuntando hacia las áreas donde el diseño puede ser mejorado.

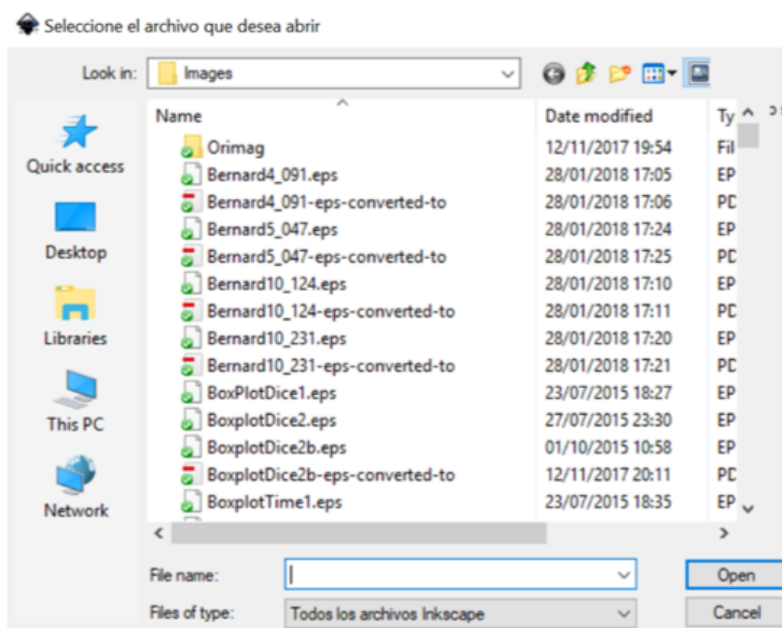


Figura 12. Ventana de apertura de archivos de Windows. Fuente: Inkscape.

GOMS (Card, Moran y Newell, 1986), por sus siglas en inglés, es un conjunto de objetivos (Goals), operadores (Operators), métodos (Methods) y reglas (Selection rules), para escoger entre métodos compitiendo por un mismo objetivo. Es un método usado por los especialistas en usabilidad para predecir cuantitativamente tiempos de realización de una acción o cumplimiento de un objetivo, técnicas usadas en el análisis para la eficiencia de las interfaces de usuario de los sistemas. Existen dos modelos de evaluación predictiva, sin embargo, son usados más bien para evaluar a los usuarios expertos.

1. Modelo KLM (Keystroke Level Model):

En términos generales, el modelo KLM (Card, Moran y Newell, 1986) estima tiempos por cada elemento evaluado, lo suma y obtiene un tiempo estimado en miles de segundos de lo que puede tardar un usuario en realizar una acción.

Utiliza varios operadores, midiendo el tiempo que se tarda en presionar una tecla o grupo de teclas, presionar el ratón (mouse), moverse entre el mouse y el teclado, trazar una línea con el mouse (mover el mouse con uno de los botones del mouse presionado), apuntar con el cursor del mouse y, uno de los más importantes, la preparación mental del usuario. El operador más subjetivo es el de preparación mental, el cual es difícil de medir en tiempo pero, generalmente, se hace una estimación. Por ejemplo: borrar una palabra en un editor de texto. Puede haber varias maneras de ejecutar esta acción:

- Hacer clic sobre la palabra, mover el mouse sobre la palabra y presionar la tecla Delete.
- Hacer clic en el inicio de la palabra, presionar la tecla shift al final de la palabra, seleccionar la palabra y presionar la tecla Delete.
- Hacer clic en el inicio y seguir presionando clic hasta borrar todos los caracteres de la palabra.
- Hacer doble clic sobre la palabra y presionar la tecla Delete.

Cada una de estas opciones requiere un tiempo, es un ejemplo simple para evaluar cuál de ellas supone más o menos tiempo. Con este método, se cuenta el número de clic y tiempo de movimiento del mouse, aunque cada clic o movimiento va a depender muchísimo del usuario. Sin embargo, existen tablas estimadas con las cuales se puede hacer una estimación de estos valores (Card et al. 1980).

2. Modelo CPM-GOMS:

Este modelo consiste en evaluar los tiempos referentes al procesador perceptual, cognitivo y procesadores de motricidad en paralelo, en la figura 13 denotamos el procesador paralelo de la mano derecha, mano izquierda y de los ojos.

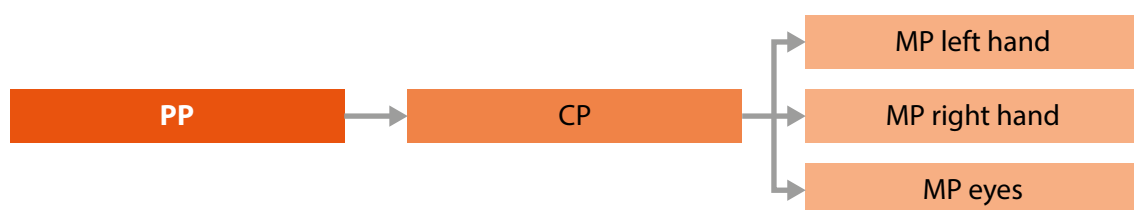


Figura 13. Modelo de evaluación predictiva CPM-GOMS. Fuente: MIT OpenCourseWare.

El problema surge cuando se solapan las tareas, el procesador perceptual activa el procesador cognitivo, primero se percibe y luego se decide ejecutar la acción, la cual puede activar en paralelo cualquiera de los procesadores motrices con los movimientos de las manos y ojos.

3.4. Errores y Control de Usuario

3.4.1. Error humano

Los errores humanos son caracterizados por desatención, similitud de tareas, frecuencia de uso del sistema o aplicación.

3.4.2. Principios de Diseño

De los principios de diseño debemos tener en cuenta ciertas ideas para evitar o prevenir el error humano:

1. Prevenir errores en la medida de lo posible.
2. Escribir buenos y correctos mensajes de error.
3. Darle el control al usuario sobre los diálogos, toda operación debería tener la opción de cancelar.
4. Darle el control al usuario sobre los datos, permitirle hacer CRUD (*create, read, update and delete*), crear, leer, modificar o actualizar y eliminar.

3.4.3. Deshacer (Undo)

La tarea de deshacer es más complicada de lo que parece. No es una tarea sencilla deshacer, es necesario almacenar las tareas realizadas, estatus, cadena de acciones, etc. Normalmente, se tiene un límite por la capacidad de almacenamiento temporal de las acciones que se pueden deshacer.

Tema 4.

Diseño Centrado en el Usuario

En esta sección desarrollaremos el tema de diseño de las interfaces de usuario considerando la forma de los procesos y principios que deberíamos aplicar para construir una UI. Mencionaremos el modelo de cascada y el modelo iterativo. Este último quizás es la mejor práctica para desarrollar UI. El modelo iterativo es una especialización del modelo espiral descrito por Boehm (1986) para el desarrollo en ingeniería de software (ver figura 14). Nos enfocaremos en el diseño iterativo llamado diseño centrado en el usuario.

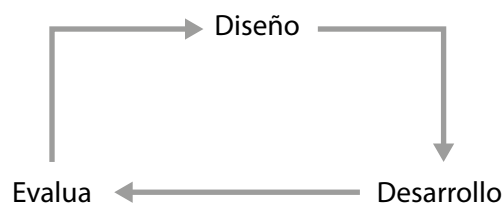


Figura 14. Modelo iterativo en ingeniería de software. Fuente: Elaboración propia.

Generalmente, el diseño de la interfaz no es tan sencillo y la usabilidad es una propiedad difícil de garantizar. Uno de los objetivos de este curso es facilitar las herramientas para poder aplicar ingeniería de la usabilidad en el diseño de las interfaces. El diseño iterativo es un proceso cíclico entre el diseño,

el desarrollo o la implementación y la evaluación. Se realiza un diseño de la interfaz, se desarrolla un prototipo y se evalúa. Dependiendo de los resultados de la evaluación, se puede hacer un rediseño, se desarrolla y se vuelve a evaluar, y así sucesivamente seguimos iterando hasta lograr un buen diseño que cumpla mínimamente con las necesidades y requerimientos del cliente y del usuario, sin dejar de lado la aplicación de los principios de la usabilidad. El desarrollo de la interfaz del usuario (UI) siempre es un riesgo, sin embargo, siguiendo las guías de diseño, es posible reducir el número de iteraciones. Es importante tener en cuenta que a la primera es imposible lograr un buen diseño.

4.1. Modelos para el Desarrollo de software

A continuación, se describirán algunos de los modelos de desarrollo de software a fin de compararlos y tener una visión de por qué el modelo iterativo permite tener un mejor diseño de interfaz.

4.1.1. Modelo de Cascada

Los modelos de Cascada (ver figura 15) tienen sentido para proyectos de bajo riesgo. En este modelo cada módulo tiene su proceso de evaluación, el diseño se valida en función de los requerimientos, la codificación en función del diseño, y así sucesivamente. Las validaciones no siempre se dan en cada paso del modelo de cascada, a veces se validan en el siguiente paso. El problema es cuando se comete un error en una etapa temprana y el feedback se obtiene en etapas posteriores. Un error podría acarrear costos y trabajos innecesarios.

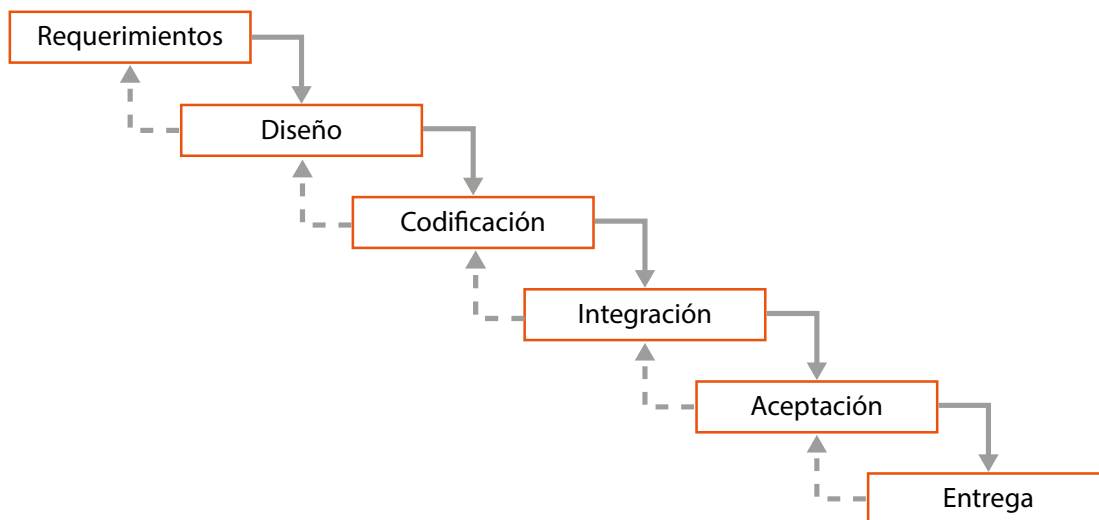


Figura 15. Modelo de cascada en ingeniería de software. Fuente: Elaboración propia.

Este modelo de desarrollo de software no es muy útil cuando el usuario no se involucra desde las etapas tempranas del desarrollo, es muy importante la constante validación en las etapas tempranas a fin de cometer la mínima cantidad de errores posibles.

4.1.2. Modelo Iterativo

El modelo iterativo o espiral es bastante útil cuando los requerimientos no son conocidos o son inciertos. Los procesos de diseño de UI centrados en el usuario son procesos iterativos, basados en el

desarrollo de un prototipo, conllevan una evaluación constante y se enfocan tempranamente en el usuario y las tareas.

4.1.3. Modelo de Espiral

El modelo espiral (ver figura 16) es un modelo iterativo con un ciclo espiral, inicia en el diseño, se desarrolla, se evalúa y el resultado de la evaluación puede requerir un rediseño o actualización o modificación del diseño. La forma de espiral corresponde al costo que acarrea cada paso del ciclo o lo que equivale, además, a su fidelidad y precisión. Por ejemplo, un primer diseño puede ser en papel, lo cual es de poca fidelidad.

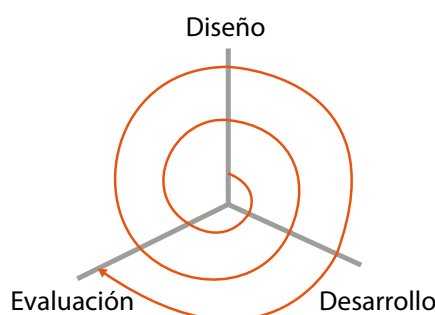


Figura 16. Modelo de cascada en ingeniería de software. Fuente: Elaboración propia.

El diseño iterativo de la interfaz con prototipos sencillos y económicos podría presentar alternativas de diseño e ir trabajando en ellas, de forma paralela, hasta acordar el mejor diseño de la interfaz que se adecúe a las necesidades del cliente y del software. A medida que se va desarrollando, el riesgo de tener un mal diseño disminuye. A mayor número de iteraciones, se obtiene un mejor diseño, sin embargo, se debe tener cuidado con los costos que acarrea cada iteración.

4.2. Procesos de Diseños Centrados en el Usuario

El diseño iterativo es, como hemos visto, crucial para en el proceso de diseño de interfaces centrado en el usuario y es el más utilizado. Este proceso tiene tres características:

1. El diseño iterativo usa construcción rápida de prototipo. Prototipos sencillos, de construcción rápida.
2. Su enfoque inicial es el usuario (quiénes son los usuarios) y las tareas (qué tareas van a realizar).
3. La evaluación es constante, involucra a los usuarios, consultores y los diseñadores. Cada modificación del prototipo es evaluado constantemente.

4.2.1. Pautas del diseño centrado en el Usuario

El primer paso en el proceso de diseño de interfaz centrado en el usuario es, entonces, conocer quiénes son los usuarios: ¿quiénes son?, ¿qué nivel tienen?, ¿cómo es su entorno?, ¿cuáles son sus objetivos?, ¿qué información necesitan, ¿qué pasos están involucrados en el logro de sus objetivos?

En algunos casos es necesario realizar entrevistas a los usuarios, observar cómo realizan las tareas normalmente. Es importante siempre tener en cuenta las pautas de diseño que ya hemos tratado en las secciones anteriores sobre aplicar los principios de usabilidad. Todo ello, ayuda a evitar errores y costos de desarrollo. Recordemos que las pautas son generalmente heurísticas y no reglas rígidas, más bien son flexibles. La flexibilidad se debe, en parte, a que inicialmente pueden no tenerse muy claro los requerimientos del sistema, por lo que se debe ser flexible a la hora de realizar un diseño de prototipo, por ejemplo. Una de las pautas es que el usuario debe tener el control pero, al mismo tiempo, tenemos que prevenir errores. Otra pauta hace referencia a la visibilidad, pero hacer que todo sea visible puede afectar negativamente a la sencillez. Las pautas de diseño nos ayudan a analizar las alternativas de diseño con sensatez, pero no dan todas las respuestas.

Las pautas son:

1. Conocer al usuario y las tareas que debe realizar.
2. Tomar en cuenta los principios de diseño:
 - a. Fácil de Aprender.
 - b. Visibilidad, sin afectar a la sencillez.
 - c. Eficiencia.
 - d. Prevención y manejo de errores.
 - e. El usuario debe tener el control y la libertad, previniendo los errores de usuario.

4.2.2. La implementación o el desarrollo del diseño

Una manera de predecir la usabilidad del sistema es con la construcción de un prototipo, mientras más económico mejor. Luego, tener en cuenta cómo se va a presentar la interfaz gráfica, según las especificaciones del sistema.

Prototipo:

Inicialmente, un prototipo en papel o en diagrama, a pesar de ser de baja fidelidad, puede dar una primera visión de lo que se construirá más adelante. Luego, en un próximo paso, se puede hacer un diseño con mayor fidelidad, usando algún lenguaje de programación.

Implementación de la interfaz gráfica del usuario (GUI):

Para la implementación de la interfaz gráfica se debe tener en cuenta:

- a. Tipo de entrada y salida.
- b. Presentación. Si es una aplicación de escritorio o requiere un diseño de interfaz Web, aunque hoy en día la tendencia es usar sistemas Web y móviles.

- c.** Existen herramientas que facilitan los diseños GUI, escoger la que mejor se adecúe a las necesidades del usuario y del cliente, además del presupuesto.

4.2.3. Evaluación del Diseño Centrado en el Usuario

Se pueden aplicar varios tipos de evaluación:

- a.** Evaluación predictiva. Como hemos visto anteriormente, donde el usuario se simula para predecir eficiencia y, en parte, la usabilidad del mismo.
- b.** Evaluación experta. Usando heurísticas de evaluación, lo veremos en detalle más adelante.
- c.** Evaluación empírica. Observando cómo el usuario usa el sistema y la aceptación que tenga el sistema hacia el usuario. Esta forma de evaluación es la que más se usa generalmente.

Tema 5.

Arquitectura de Software para el Diseño de la Interfaz de Usuario

En esta sección tocaremos el tema de la arquitectura de software más utilizada para el desarrollo de las interfaces gráficas de usuario. Antes, entendamos por diseño de patrones cómo los modelos a seguir para el diseño de dichos patrones han sido utilizados por otros desarrolladores y han permitido resolver problemas de interfaz usuario-computador. Es la razón por la que se recomienda seguir los patrones de diseño para las interfaces, a fin de cometer el menor número de errores posible en el momento de su desarrollo. Tres de los patrones de diseño de las interfaces más usadas son:

- 1.** Modelo de Vista Jerárquico
- 2.** Manejador de Eventos
- 3.** Modelo-vista-controlador

5.1. Modelo de Vista Jerárquico

Una vista es un objeto de interfaz que cubre un área determinada de la pantalla, generalmente delimitada por un área cuadrangular llamada cuadro, *frame* o ventana, en algunos casos. Dependiendo de la herramienta de programación para interfaces que estemos usando, puede tener diferentes nombres: *widgets*, controladores o interactivos.

Estas vistas están contenidas en una jerarquía de vistas (o widgets). Algunas vistas contienen otras vistas (según como sea implementado). Estas vistas contenedoras de otras vistas pueden ser: ventanas, paneles, barra de herramientas. Este modelo de vista jerárquico no es solo una jerarquía arbitraria sino espacial. Las vistas contenedoras están anidadas dentro del cuadro delimitador de su vista padre.

En la mayoría de las herramientas de desarrollo de interfaz gráfica, las propiedades de las vistas padres son heredadas por las propiedades de las vistas contenedoras. Algunas acciones de la vista contenedora no afectan al padre pero al revés sí; por ejemplo, cerrar o salirse de una vista padre saca de la escena las vistas contenedoras igual.

5.1.1. Separación de Funciones de las Estructuras Jerárquicas

Prácticamente todas las herramientas de desarrollo de GUI tienen algún tipo de estructura jerárquica. Este modelo presenta una idea de estructuración poderosa, entre sus funciones:

- **Salida.** Las vistas son responsables de mostrarse a sí mismas, y la jerarquía de vistas dirige el proceso de despliegue de vistas. Las GUI cambian su salida cambiando las vistas.
- **Entrada.** Las vistas pueden tener manejadores de entrada y manejar los controladores del mouse y el teclado.
- **Diseño.** La estructura jerárquica controla cómo se distribuyen las vistas contenedoras dentro de su área o cuadro de vista. Un algoritmo de diseño automático calcula automáticamente las posiciones y tamaños de las vistas.

5.2. Manejador de Eventos

El manejador de eventos es el elemento que permite escuchar de los dispositivos de entrada las acciones a ser ejecutadas (llamados oyentes); desde el mouse, el teclado.

El manejo de eventos de entrada GUI es una instancia del modelo de manejador de eventos (también conocido como Observador). En el modelo de manejador de eventos, una fuente de eventos permite generar una secuencia de eventos discretos que corresponde a un estado de transición en la fuente. Uno o más manejadores de eventos registran la secuencia de eventos, que proporcionan una función o acción que se ejecutará cuando se produce un nuevo evento. En este caso, el mouse es el origen del evento y los eventos son cambios en el estado del mouse: su posición x, y o el estado de sus botones (ya sea que estén presionados o liberados). Los eventos a menudo incluyen información adicional sobre la transición (como la posición x, y del mouse), que podría agruparse en un objeto de evento

o pasado como parámetros. Cuando ocurre un evento, el origen del evento se distribuye a todos los oyentes suscritos, llamando a sus funciones que ejecutarán la acción correspondiente.

5.3. Modelo Vista Controlador (MVC)

Según el patrón modelo-vista-controlador (ver figura 17) (Gamma, Helm, Johnson, Vlissides, 1995; Burbeck, 1987), originalmente creado para la interfaz de usuario Smalltalk-80, la idea original era separar la interfaz gráfica del funcionamiento de la aplicación, de tal forma que cambios en el diseño de la interfaz no requirieran cambios en la capa de negocio. Esto influyó mucho en los lenguajes de Programación Orientada a Objeto (POO), luego ha sido aplicado en el ámbito de la interfaz del usuario. Este modelo separa el frontend (Modelo de entrada del sistema, vista al usuario), del backend (capa de negocio, implementación de la funcionalidad del sistema) y el controlador (manejado a través de los eventos). En este caso, la entrada es manejada por el controlador y la salida por la vista (ver ejemplo de esquema MVC para una aplicación web).

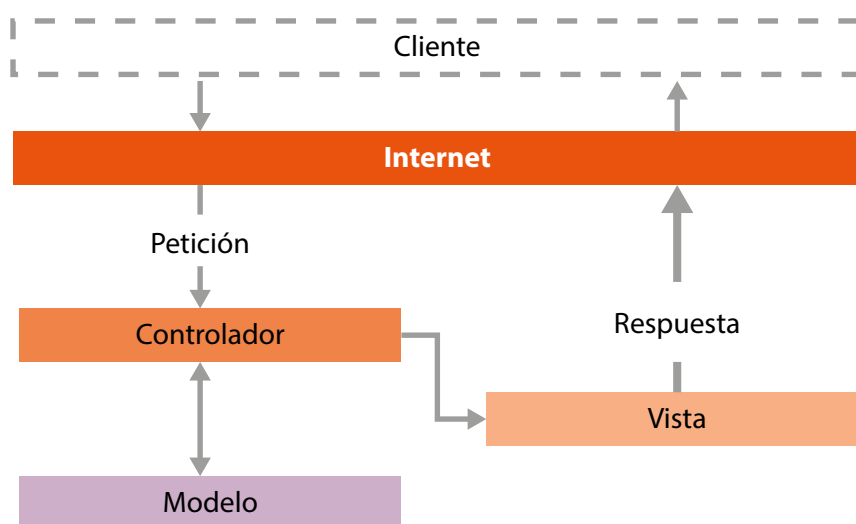


Figura 17. Modelo Vista Controlador. Fuente: <http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/122>

En términos generales, los componentes del Modelo Vista Controlador están compuesto por tres objetos:

1. **Vista:** representa la interfaz de usuario y todas las herramientas con las cuales el usuario hace uso del programa.
2. **Modelo:** es donde está toda la lógica del negocio, la representación de todo el sistema, incluida la interacción con una base de datos, si es que el programa así lo requiere.
3. **Controlador:** este componente es el que responde a la interacción (eventos) que hace el usuario en la interfaz y realiza las peticiones al modelo para pasar estos a la vista.

El modelo es responsable de mantener los datos específicos de la aplicación y proporcionar acceso a esa información. Los modelos son a menudo mutables, y proporcionan métodos para cambiar el estado de forma segura, preservando su representación invariante. Un modelo también debe notificar

a sus clientes cuando haya cambios en sus datos, de modo que las vistas dependientes puedan actualizar sus pantallas y los controladores dependientes puedan responder apropiadamente. Los modelos hacen esta notificación utilizando el patrón de oyente en el que las vistas y los controladores interesados se registran como oyentes para los eventos de cambio generados por el modelo.

Los objetos de vista son responsables de la salida. Una vista ocupa un pedazo de la pantalla, generalmente un área rectangular. Básicamente, la vista consulta el modelo de datos y dibuja los datos en la pantalla. Escucha los cambios del modelo para que pueda actualizar la pantalla y reflejar esos cambios. Finalmente, el controlador maneja la entrada, recibe eventos de teclado y mouse, e instruye al modelo a cambiar en consecuencia a algún evento.

5.3.1. Ventajas del MVC

1. Una de las principales ventajas del modelo es la separación de responsabilidades de cada objeto, los datos, las vistas y el controlador.
2. Las vistas y el modelo pueden ser reutilizados por otras aplicaciones.
3. Múltiples vistas pueden usar el mismo modelo.
4. Las vistas pueden ser reutilizadas por otros modelos.
5. Las vistas y el modelo pueden cambiar independientemente.

Tema 6.

Diseño Gráfico

En esta sección vamos a estudiar algunas de las pautas más importantes a tener en cuenta a la hora de hacer el diseño de la interfaz que, aunque se trate de un tema de otra carrera, como lo es el diseño gráfico, es importante para nuestro campo conocer los lineamientos básicos de diseño gráfico que se deben seguir cuando se esté haciendo el diseño de la interfaz del usuario.

Para mayor información y referencia sobre el tema de diseño gráfico se pueden consultar los libros de textos de Mullet K. y Sano D. (1995), Tufte E. (2001), Ware C. (2004). El libro de Mullet & Sano es anterior a la web, pero los principios que describe siguen siendo relevantes para cualquier medio visual. El libro de Ware es mucho más técnico que Mullet & Sano o Tufte. Ware discute las bases psicológicas y anatómicas de la percepción, mientras se relaciona con los principios prácticos de diseño.

6.1. Simplicidad

El diseño debe ser simple, pero al mismo tiempo completo. El diseño por simplicidad es un proceso de eliminación y no de agregación de pequeñas cosas. La simplicidad está en constante presencia con el análisis de tareas, precondiciones de información y otras directrices de diseño, que de no tenerlo en cuenta, terminaría acumulando más y más elementos en un diseño, "por las dudas". La simplicidad te

obliga a tener una buena razón para todo lo que agregas y quitar todo lo que no pueda sobrevivir al escrutinio riguroso.

Algunos lemas que se deben tener en cuenta cuando se realiza un diseño gráfico son los siguientes:

- Menos es más.
- Si hay dudas, mejor quítalo.
- Mantenerlo simple, estúpido (KISS keeping it simple, stupid).
- "La perfección no se alcanza cuando no hay nada más que añadir, sino cuando no hay nada más que quitar", Antoine de Saint-Exupéry.

Algunas de las técnicas para mantener la simplicidad se discuten en las próximas secciones.

6.1.1. Reducción

Cuando se habla de reducción en diseño gráfico, se habla de eliminar aquello que no es necesario. Esta técnica tiene tres pasos:

1. Decidir las cosas esenciales que deben estar ya que, sin ellas, no tiene sentido el diseño.
2. Examinar cada elemento: etiqueta, control, color, fuente, grosor de la línea, etc., para decidir si tiene un propósito esencial.
3. Eliminarlo si no es esencial. A veces puede parecer esencial, entonces se debe probar a eliminarlo y ver si el diseño se pierde o no influye mucho.

Los iconos demuestran bien el principio de la reducción (ver figura 18). Una fotografía de un par de tijeras no puede posiblemente funcionar como un ícono de 32x32 píxeles; en cambio, tiene que ser una imagen cuidadosamente dibujada que incluya los detalles mínimos de forma del objeto.



Figura 18. Iconos de los procesadores de palabras de ©Microsoft. Fuente: Elaboración propia.

6.1.2. Consistencia

Los elementos deben ser consistentes en cuanto al formato, misma fuente, color, ancho de línea, dimensiones, orientación, etc. Inconsistencias en el formato, fijarán la atención del usuario donde hay inconsistencia y, es posible, que no sea lo que se quiere que el usuario haga. Por el contrario, si el diseño es consistente, los elementos que se quieran resaltar y de los cuales se quiere que se destaquen al usuario, se destacarán mejor. La consistencia se logra usando patrones regulares y limitando variaciones que no sean esenciales entre los elementos, a menos que se quiera resaltar o destacar para llamar la atención del usuario.

6.1.3. Doble función

Para hacerlo simple y reducir redundancia se pueden combinar elementos haciendo que cumplan múltiples roles en el diseño. Hablamos sobre esta idea de "doble función" desde el bosquejo del diseño. Por ejemplo, la barra de desplazamiento puede servir a tres roles, permite el arrastre, indica la posición de la ventana de desplazamiento relativa al documento completo e indica la fracción del documento que es visible en el espacio de la ventana. Del mismo modo, la barra de menú en una aplicación desempeña varias funciones (ver figura 19): etiqueta conjunto de funciones (Archivo, Inicio, Insertar, etc.), indicador de activación y ubicación para botones de control de conjunto de funciones.

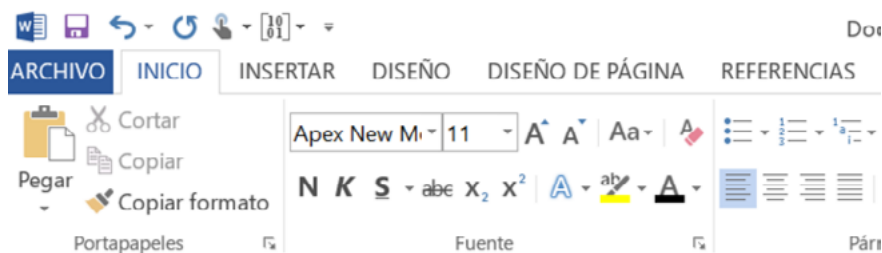


Figura 19. Barra de menú de ©Microsoft Word. Fuente: Elaboración propia.

6.1.4. Fragmentación

La forma de presentar la información se puede simplificar, nuestra memoria y percepción es a trozos, como vimos cuando hablamos sobre el aprendizaje. Una lista de cuatro (4) elementos o menos es algo sencillo para que nuestra memoria organice y perciba, pero listas de cosas mayores a eso es más difícil, por ello, la lista larga de cosas debe organizarse.

Nuestra capacidad para formar trozos en la memoria operativa depende en gran medida de cómo la información es presentada. Si tiene que mostrar un número largo o un código largo, es mejor dividirlo en trozos más cortos para presentar la información. 3-4 caracteres por porción es una buena regla empírica que funciona bastante bien en la mayoría de los casos.

6.2. Contraste y Variables Visuales

6.2.1. Contraste

El contraste se refiere a diferencias perceptibles en una dimensión visual, como el tamaño o el color. El contraste en diseño gráfico es la irregularidad en un diseño que comunica información o hace que los elementos se destaquen. La simplicidad dice que debemos eliminar inconsistencia de elementos y eliminar los que no son esenciales.

Cuando se decide agregar contraste en algún elemento del diseño, entonces se debe pensar en la dimensión y el grado de contraste de tal manera que la diferencia con respecto a otros elementos sea notable, fácilmente perceptibles y apropiadas para la tarea. Berlin, Berg y Scott (1981) señalan la importancia de tener en cuenta siete variables visuales para agregar contraste en los elementos del

diseño. Ellos son: el color, el tono de color, texturas, formas, posición, orientación y tamaño (ver figura 20).

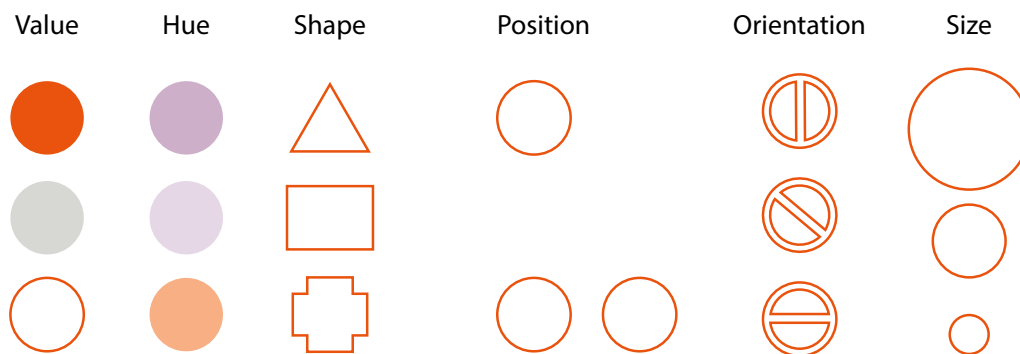


Figura 20. Siete variables visuales. Berlin et al. (1989).

Berlín et al. llamaron a estas dimensiones **variables retinales** porque se pueden comparar sin esfuerzo sin procesamiento cognitivo adicional, como si la retina estuviera haciendo todo el trabajo.

6.2.2. Variables Visibles

Las variables visuales se utilizan para la comunicación, permiten codificar y simplificar la información. Sin embargo, antes de elegir una variable visual para expresar alguna distinción, debe asegurarse de que las propiedades de la variable visual coincidan con su comunicación. Por ejemplo, puede mostrar una temperatura usando cualquiera de las dimensiones: posición en una escala, longitud de una barra, color (rojo), o la forma de un icono (un sol feliz o un copo de nieve). La elección de la variable visual afecta fuertemente a cómo los usuarios podrán percibir y usar los datos mostrados.

Dos características de las variables visuales son el tipo de escala y la longitud de la escala. Una escala puede ser nominal, de orden o cuantitativa.

- Escala nominal es una lista de categorías, la comparación entre escalas nominales se puede hacer por los símbolos de igualdad.
- Una escala de orden agrega una orden a los valores de la variable: posición, tamaño, valor y, hasta cierto punto, la granularidad de la textura.
- Una escala cuantitativa puede percibir el nivel de diferencia en el orden. La posición es cuantitativa, puedes ver dos puntos en un gráfico y decir que uno es dos veces más alto que el otro. El tamaño es también cuantitativo, pero hay que tener en cuenta que tenemos más facilidad para percibir las diferencias cuantitativas en una dimensión (es decir, la longitud) que en dos dimensiones (área).
- Escalas no cuantitativas. El tono de color, por ejemplo, no es un valor cuantitativo, no podemos percibir fácilmente que un tono es el doble de oscuro que otro.

La longitud permite distinguir niveles. La longitud de una variable es la cantidad de valores distinguibles que se pueden percibir. Podemos reconocer una variedad casi infinita de formas, por lo que la variable

de forma es muy extensa, pero puramente nominal. La posición igual puede ser grande o corta. La orientación, por el contrario, es muy corta, solo un puñado de orientaciones diferentes se pueden percibir en una pantalla.

6.2.3. Selectividad

La percepción selectiva puede variar la atención de los usuarios. La selectividad es el grado en que se puede seleccionar un único valor de la variable a partir de todo el campo visual. Variables selectivas pueden ser: posición, tamaño, orientación, tono de color, valor de la textura. La mayoría de las variables son selectivas, por ejemplo, puede ubicar objetos verdes de un vistazo u objetos pequeños. La forma, sin embargo, no es selectiva en general. Es difícil distinguir triángulos en medio de un mar de rectángulos.

6.2.4. Asociatividad

La asociatividad se refiere a lo fácil que es ignorar la variable, dejando que desaparezcan todas las distinciones a lo largo de esa dimensión. Las variables con mala asociatividad interfieren con la percepción de otras dimensiones visuales. En particular, el tamaño y el valor no son asociativos, ya que los objetos pequeños o débiles son difíciles de distinguir.

6.2.5. Técnicas para lograr el Contraste

Para lograr un buen contraste en los elementos de una interfaz, primero se elige el elemento al cual se le quiere dar un mayor contraste, luego se escoge la variable visual más adecuada que permita comunicar al usuario las tareas correspondientes asociadas al elemento gráfico. Por ejemplo, la estructura de un libro de texto: título, capítulo, sección, texto del cuerpo, nota al pie. Los datos requieren una variable visual ordenada, una variable puramente nominal como la forma (por ejemplo, usar diferentes fuentes) no sería capaz de comunicar el orden jerárquico. Pero colocar, quizás, el título en una fuente más grande o en diferente color, enumerar los capítulos o secciones permite tener un mejor contraste y da una percepción de jerarquía sobre los componentes.

Todo elemento visual que se agregue debe cumplir con la simplicidad y la asociatividad. Una vez que haya elegido una variable visible, se usa la mayor cantidad posible de la variable. Es decir, si se escoge el tamaño, se deben usar tantos niveles de tamaño como sea posible para representar los datos, pero que al mismo tiempo los tamaños sean distinguibles entre ellos y perceptibles al usuario. Las diferencias exageradas pueden ser útiles, en ocasiones, especialmente cuando se dibujan iconos, el dibujante tiene que darle a los objetos proporciones exageradas para que sean fácilmente reconocibles.

La **prueba de estrabismo** es una técnica conocida que simula el procesamiento visual temprano para que pueda ver si los contrastes que ha intentado establecer son evidentes. El ejercicio consiste en cerrar un ojo y entrecerrar el otro para interrumpir la concentración. Cualquier distinción que se pueda realizar será visible de un vistazo. Esta estrategia o prueba de estrabismo se puede usar para probar qué tan distinguible es un color o un elemento en la interfaz.

Tema 7.

Visualización de la Información

Las técnicas de visualización de la información tienen como finalidad analizar y comunicar la información de tal manera que el usuario pueda pensar y entender visualmente lo que los datos están diciendo. Una correcta visualización de los datos permite analizar y presentar los datos de tal manera que puedan ser interpretados correctamente. Visualizar los datos gráficamente o visualmente permite obtener patrones y relaciones entre los datos que, de otra forma, no sería posible. Nuestro sistema perceptivo contribuye al proceso de pensamiento cuando se visualizan los datos. Por la misma razón, la visualización puede ser una excelente manera de comunicar información a otras personas, pensar juntos y tomar una decisión compartida.

La visualización permite:

- Analizar. Pensar con el sistema perceptual del individuo, jugar con los datos y entenderlos.
- Presentar. Permite compartir, colaborar, resaltar hechos importantes, patrones, relaciones y tomar decisiones.

7.1. Algunas formas de Visualizar los datos

La visualización de los datos debe permitir hacer un buen uso del espacio, algunos son interactivos, lo cual los hace más usables y representativos de los datos.

7.1.1. Gráficos Tradicionales

Los tradicionales gráficos de barras, gráficos de líneas y gráficos de torta tienen la ventaja de que son fáciles de aprender: se enseñan en las escuelas y se ven en una variedad de medios. Son bien conocidos y fáciles de interpretar.

7.1.2. Diagramas de Dispersión (Scatterplot)

Permite mapear par de atributos en los espacios $x-y$ (ver figura 21). Una forma más genérica de gráfico, ampliamente utilizada en la visualización de la información, es un diagrama de dispersión, que representa cada elemento de datos como una marca cuya posición (x, y) está determinada por dos de sus atributos. Por lo general, estos dos atributos están relacionados de alguna manera, por lo que la nube de puntos tendrá un aspecto visible y un patrón útil de alto nivel que transmite información, aunque no necesariamente cierto, dependerá siempre de los datos. Se usan otras variables visuales para mapear otros atributos de datos en cada punto. Cuando la pantalla se llena demasiado y los puntos caen uno encima del otro, un enfoque es la fluctuación o desplazamiento del punto, moviendo los puntos de superposición ligeramente separados, sacrificando la precisión posicional por una visualización más precisa de la densidad (Ellis, 2007).

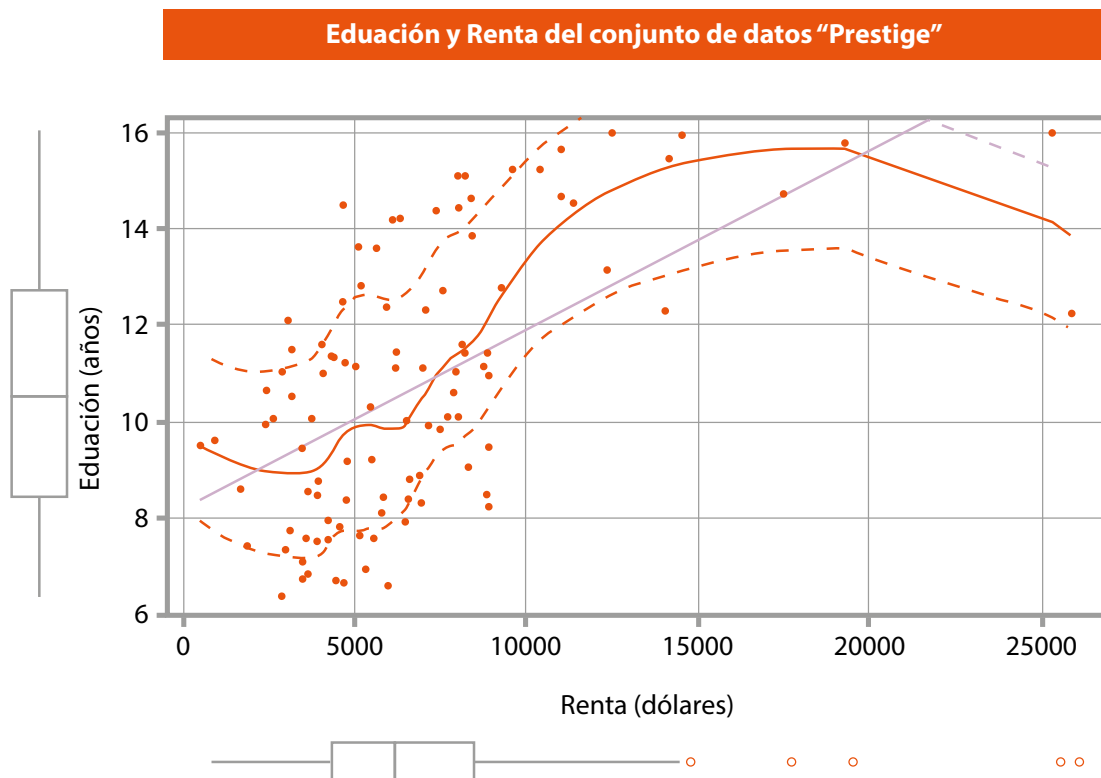


Figura 21. Diagrama Scatterplot. Fuente: <http://nubededatos.blogspot.com>

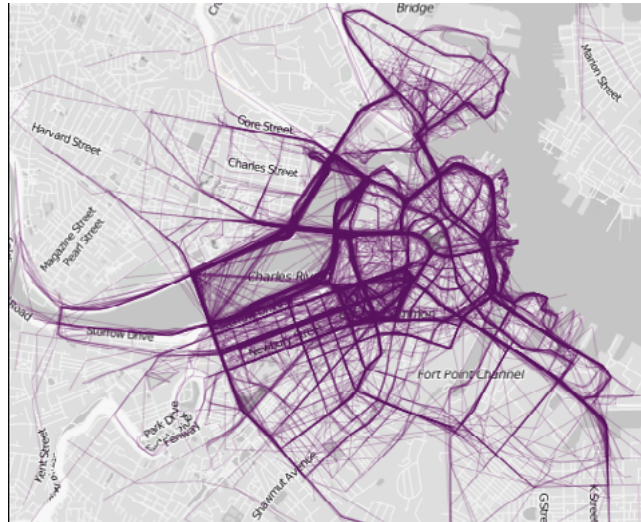


Figura 24. Ruta de Corredores. Fuente: <http://cristinaaced.com/blog/2016/06/08/nubes-etiquetas-2/>

Rutas de Vuelo de un día en Londres:

Se visualizan más de 6000 vuelos cada día en esta ciudad (ver figura 25).



Figura 25. Ruta de Vuelo de un día en Londres. Fuente: <https://elbauldelprogramador.com/14-ejemplos-visualizacion-datos/>

Historia de las medallas olímpicas desde 1896 hasta 2016:

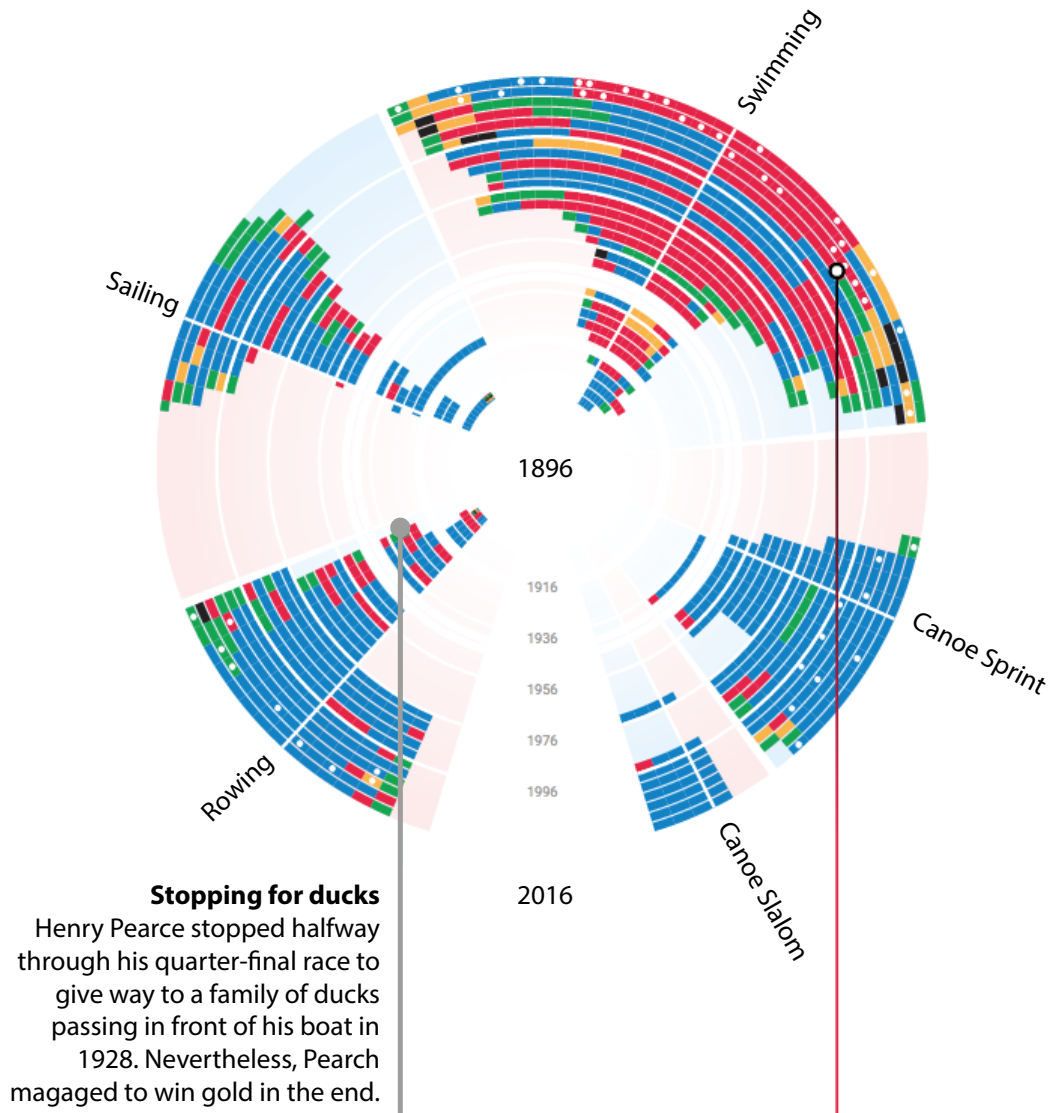


Figura 26. Muestra la visualmente la historia de todas las medallas de oro en los juegos olímpicos desde 1896 hasta 2016.

Fuente: <https://elbauldelprogramador.com/14-ejemplos-visualizacion-datos/>

7.3. Herramientas de Visualización

Algunas de las herramientas que permiten usar los modelos de visualización más comunes son:

1. Flot, Flotr: utilizan Javascript y canvas, permite crear visualizaciones convencionales de datos.
2. ProtoVis: basado en Javascript y SVG, es flexible, permite usar los modelos de visualización más comunes que pudimos ver en esta sección.

Otra de las herramientas consiste en realizar las visualizaciones programando y usando algunos de los lenguajes gráficos. Por ejemplo, en las visualizaciones de las mejores visualizaciones de la

información, seguramente los datos fueron leídos y, luego, combinando estrategias de visualización para poder visualizar la imagen que se genera de resultado.

Tema 8.

Accesibilidad e Internacionalización

La accesibilidad se refiere a cómo hacer posible que usuarios con ciertas discapacidades, motoras o físicas puedan acceder a la interfaz y poder interactuar con el computador o dispositivo. En algunos casos, se requieren dispositivos adicionales, por ejemplo, teclados para personas con discapacidad visual. Es decir, tiene que ver tanto con la tecnología de hardware como de software usada para mejorar la accesibilidad de los usuarios a los sistemas computacionales y permitir la interacción humano-computador en caso de discapacidad. Por ello, estaremos hablando de algunas de las tecnologías usadas y algunas pautas a seguir para hacer diseños de UI accesibles.

Parte de la característica de la accesibilidad es la internacionalización, es decir, permitir que el sistema o aplicación esté disponible en diferentes lenguajes y culturas. La parte más difícil en cuanto a este tema de la internacionalización de las interfaces es justamente las culturas. Generalmente, hoy en día es posible utilizar herramientas ontológicas y objetos de interfaz multilenguaje, algunos ya son universales y no tenemos mayor problema, sin embargo, sí tenemos aún un problema de investigación por resolver en esta área y es la internacionalización ya que tiene cuenta no solo la parte del lenguaje, sino también la cultural.

¿Han observado Uds. los diseños de las páginas web en China, por ejemplo? Culturalmente, el uso de los colores puede influir positiva o negativamente en las personas y puede variar dependiendo de las

culturas. ¿Cómo podemos entonces atacar este problema? Es un tema interesante tener la posibilidad de ofrecer la interfaz en varios lenguajes permite apoyar a los usuarios que hablan diferentes idiomas y tener diferentes convenciones culturales en cuanto al lenguaje. Hablaremos de algunas de las razones por las que la internacionalización puede ser difícil y analizaremos parte del apoyo que existe en los *toolkits* para desarrollar las GUI y hacerlo más fácil.

8.1. Accesibilidad

8.1.1. Tipos de discapacidad

En esta sección solo nos enfocaremos en tres tipos de discapacidades: la visual, auditiva y la motora.

Discapacidad Visual:

La discapacidad visual puede ser total, parcial o existir discapacidad para percibir los colores, por ejemplo, el daltonismo, o la discapacidad de enfocar claramente por agudeza visual deteriorada. Para la mayoría de las personas, los problemas de agudeza visual pueden corregirse con gafas o lentes de contacto, pero en algunos casos estos defectos no pueden ser corregidos.

Discapacidad Auditiva:

Las deficiencias auditivas afectan a la capacidad de detectar la intensidad del sonido y varían en un espectro de sensibilidad. Las deficiencias auditivas, a menudo, dependen de la frecuencia del sonido, se puede dar el caso que una persona puede escuchar bien las frecuencias más bajas, pero no las frecuencias más altas.

Discapacidad Motora:

Las discapacidades motoras vienen en muchas formas diferentes y tienen muchas causas diferentes. Las víctimas de parálisis cerebral experimentan temblores y espasmos incontrolables, por lo que es difícil controlar los movimientos. La distrofia muscular y la esclerosis múltiple pueden debilitar los músculos y los pacientes pueden cansarse fácilmente cuando hacen movimientos musculares repetidos o grandes. El daño neuronal puede causar completa parálisis de las extremidades.

Las causas de estas discapacidades son muy variadas. La discapacidad visual, auditiva y motora puede darse por la edad. Otras discapacidades pueden darse por la sobre-exposición, por ejemplo: la capacidad auditiva puede perderse a temprana edad por la sobre-exposición a sonidos muy altos durante mucho tiempo. También puede deberse a situaciones naturales de nacimiento, o por algún accidente, como puede ser el caso de las discapacidades motoras.

8.1.2. Tecnología Asistida

El diseño universal es una escuela de pensamiento que busca pensar en el diseño de las cosas que funcionen para todos los usuarios, para lo cual, toma en cuenta varias características:

- Uso equivalentes. Todos los usuarios deberían tener la misma interfaz, no siempre es posible, pero se busca que sea así en la medida de lo posible.
- Flexibilidad en el uso.
- Simple e intuitivo.
- Con información perceptible.
- Que sea tolerante al error.
- Que requiera para su uso el menor esfuerzo posible.
- Con tamaño y espacio adecuado para su uso.

No siempre está muy claro cómo conseguir este diseño, pero se busca, dentro de lo posible, que pueda cumplir con estas características.

La tecnología asistida permite ayudar a algunos usuarios con discapacidades, dicha tecnología puede ser a nivel de hardware o de software. Por ejemplo:

1. Ampliación de la pantalla para facilitar la lectura, lo que ayuda a los usuarios que tengan una agudeza visual reducida pero que no son totalmente ciegos.
2. Los lectores de pantalla ayudan a los totalmente ciegos, leyendo el contenido de la pantalla en voz alta como discurso. Para usuarios totalmente ciegos que conocen Braille, un lector de pantalla se puede conectar a una pantalla Braille, que les permite leer la pantalla de forma privada (y silenciosa) y probablemente más rápido también.
3. Para usuarios con problemas de audición, las interfaces gráficas de usuario plantean menos problemas porque es mucho menor la información transmitida por señales auditivas. Los sonidos del sistema pueden traducirse a pantalla flash y los vídeos pueden incluir subtítulos.
4. En el lado de la entrada, hay dispositivos apuntadores alternativos. La mirada o el seguimiento de la pose de la cabeza pueden mover el cursor del mouse alrededor de la pantalla sin el uso de las manos.
5. Dispositivos de soplar y succionar (en los que el usuario sopla o succiona un tubo) se puede utilizar para hacer clic en un botón, a menudo en combinación con un dispositivo impulsado por la boca como palanca de mando.
6. Los usuarios con discapacidades motoras menos extremas pueden usar *touchpads* o *trackballs*, que son más accesibles que los ratones porque requieren movimientos más pequeños. El cursor del mouse también se puede mover alrededor por las teclas del teclado, Windows y Mac tienen esta función incorporada.
7. Los usuarios totalmente ciegos, por lo general, usan un teclado exclusivamente, así que no se les puede ofrecer pantallas de lectura, quizás como soporte para algún acompañante.

8. Para la entrada de teclado por usuarios con problemas motrices graves, se puede usar un dispositivo señalador que se pueda combinar con un teclado en la pantalla.
9. El software de controlador del teclado se puede ajustar para que sea más fácil de usar desactivando la auto-repetición para que las teclas no tengan que ser liberadas rápidamente o modificando el uso de las teclas (Ctrl+Shift) para que el usuario no tenga que presionar varias teclas a la vez. El reconocimiento de voz ofrece otra forma para accionar algunos comandos y hacer entradas de texto sin utilizar las manos.

8.1.3. Guías de accesibilidad

Ahora veamos algunas pautas específicas para crear interfaces accesibles. La mayoría de estas directrices están orientadas a hacer que su interfaz sea compatible con la tecnología de asistencia, por ejemplo, ayudando al lector de pantalla hacer un mejor trabajo de traducir la pantalla en texto.

Las pautas que siguen se resumen a partir de dos fuentes. La Sección 508 es una accesibilidad estándar para sitios web y software creado por agencias del gobierno de EE. UU. o contratistas del gobierno. Cualquiera que quiera vender software al gobierno de EE. UU., debe seguir las reglas de la Sección 508, que cubre tanto el software de escritorio como los sitios web.

La Iniciativa de Accesibilidad W3C es un grupo en el World Wide Web Consortium que ha producido una lista de pautas de accesibilidad (voluntarias) para los sitios Web. No las discutiremos en esta sección, pero sirven de referencia.



Enlace 1

Reglas de la Sección 508.

<http://www.section508.gov/>



Enlace 2.

Pautas del W3C:

<http://www.w3.org/TR/WAI-WEBCONTENT/>

8.2. Internacionalización y Localización

Las interfaces con poblaciones de usuarios internacionales, tales como Microsoft Word, Windows, programas de los sistemas operativos MAC, por ejemplo, deben ser cuidadosamente diseñadas para facilitar su adaptación a otros idiomas y culturas. El proceso de hacer una interfaz de usuario lista para la traducción se llama internacionalización. Esencialmente, la **internacionalización** separa las partes de la interfaz específicas del idioma del resto del código para que esas partes puedan ser reemplazadas fácilmente. La traducción generalmente se hace no por programadores, por lo que su trabajo es más fácil si los mensajes de texto están separados del código. Actualmente, hacer esta traducción para un idioma y cultura en particular se llama **localización**.

Una forma de entender la diferencia entre estos dos términos técnicos es por analogía a la portabilidad a través de plataformas de sistemas operativos. Si se desarrolla un programa cuidadosamente para

que no dependa de características específicas de un sistema operativo o procesador, se dice que es un programa portable e internacionalizado.

Lamentablemente, la localización es mucho más difícil que simplemente saber qué palabras sustituir y tampoco se puede depender de miembros multilingües en su equipo de trabajo.

8.2.1. Retos de Diseño

Algunos de los retos en la internacionalización de las aplicaciones son:

1. Todos los mensajes de cara al usuario deben ser traducidos, tanto los de ayuda, los botones, mensajes de error, advertencias, textos de estatus, etc.
2. Los textos en otros lenguajes podrían cambiar de tamaño, por lo que es importante tener en cuenta los espacios para dichos textos. Una palabra en alemán puede ser mucho más larga que una en inglés o español, por ejemplo. Se debe tener especial cuidado con los caracteres especiales.
3. Riesgo de las traducciones. Si la persona que traduce los textos no conoce el lenguaje o usa traductores automáticos como el traductor de Google, es posible que la traducción no sea la correcta y puede que dé un mensaje distinto, confuso o contrario.
4. Los caracteres y forma de escritura, por ejemplo, los sistemas chinos o árabes. Los alfabetos difieren y el sentido de la escritura también, como es el caso del árabe.

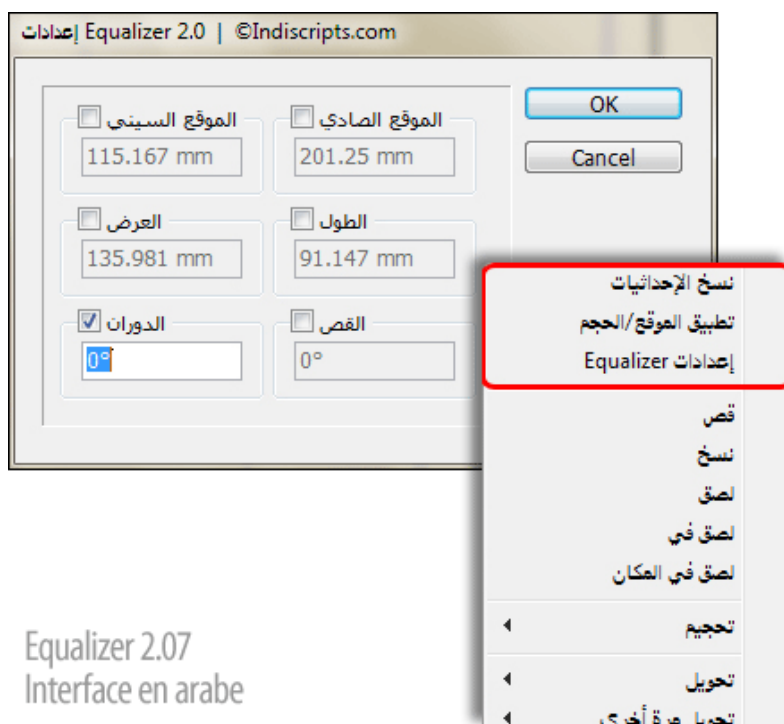


Figura 27. Caracteres de interfaz Window para el idioma árabe. Fuente: <http://www.indiscripts.com/>

5. Formato de fechas y horas. Dependiendo del país y cultura, pueden cambiar de forma de presentación.

USA: 02/26/2018 (Mes/Día/Año)

El resto: 26/02/2018 (Día/Mes/Año)

6. Cantidades, números:

USA/Reino Unido: 45,225.55

Francia: 45 225,55

Alemania: 45.225,55

7. Los colores pueden tener diferentes significados en culturas distintas. En el este de Asia, por ejemplo, el blanco está asociado con muerte y, por lo general, es el tema de color usado en los funerales. Por el contrario, en el oeste el blanco es símbolo de pureza y es usado en las bodas, así como el uso del rojo que, en ciertas culturas, es símbolo de suerte.
8. El uso de iconos, en algunas culturas también difieren. Un ejemplo es el símbolo de *stop*, es un icono bastante universal, sin embargo, la forma difiere en Japón, por ejemplo.



Figura 28. Símbolo del Stop en EE. UU. Fuente: <https://vinilos-stica.es>



Figura 28.1. Símbolo del Stop en Japón. Fuente: <http://japanbybike2012.blogspot.com>

8.2.2. Técnicas para la implementación

Hoy en día, se cuenta con *toolkits* de desarrollo de UI que proveen un buen soporte para la internacionalización, lo cual hace que sea sencillo de implementar.

1. Mensajes:

Los mensajes pueden ser traducidos usando *tokens* para facilitar el manejo de mensajes dinámicos. Sin embargo, hay que tener cuidado con los errores gramaticales al hacer este tipo de personalización o mensajes dinámicos. De ser necesario, es importante usar traductores

humanos, lo cual es lo ideal o, al menos, revisar las traducciones con el experto en el lenguaje al cual se está traduciendo.

2. Usar los Encoding maps para los números:

ASCII, Latin-1, UCS-2, UTF-8

3. Y para caracteres:

ASCII, Latin-1, Unicode (Latin-1+Greek, Cyrillic, etc.)

4. Bidireccionalidad de los textos. Java, por ejemplo, soporta la bidireccionalidad de la escritura para el uso de la lengua árabe.

5. Para el formato de las fechas y números, se recomienda usar librerías de formato si las herramientas de interfaces lo soportan.

6. Es necesario separar la estructura de la presentación:

Por ejemplo, el uso de imágenes o iconos puede requerir de traducción o cambios según la cultura o lenguaje, entonces es importante separarlos de la parte de la estructura. Ocurre lo mismo con el tipo de *font* del carácter ya que algunos caracteres no pueden ser representados o dibujados en la interfaz usando un tipo de *font* particular, es necesario poner atención a esto. Igualmente, es importante prestar atención al uso de colores particulares, por lo que hemos comentado anteriormente.

Tema 9.

Experiencia de Usuario (UX)

El Diseño de Experiencia del Usuario (UX, por sus siglas en inglés) es ya reconocido hoy en día como una carrera. Hay profesionales especializados en Experiencia del Usuario y se posicionan en la tecnología como los expertos en UI/UX, UI por especialistas en interfaces de usuario y UX por especialistas en Experiencia de Usuario. En este contexto, la interfaz de usuario o interacción hombre-computador se refiere a la tecnología y metodología utilizada para el desarrollo de la interfaz de usuario, la manera de comunicación del usuario con la aplicación, software o sistema; mientras que la experiencia de usuario, se refiere a cómo integrar de forma eficiente justamente la UI con el usuario. Muchos de los principios que rigen una buena interfaz de usuario son parte también de la UX.

Se pueden confundir los términos usabilidad, experiencia de usuario e interfaz de usuario. Los conceptos entre estos dos últimos está un poco más clara, la experiencia de usuario agregado a la UI hace una mejor sinergia que permite realizar mejores diseños. La usabilidad es parte de los principios fundamentales de la interfaz del usuario. Sin embargo, algunos expertos utilizan indistintamente los términos usabilidad y UX como si se tratara de lo mismo.

Veamos los conceptos de usabilidad y experiencia de usuario. La usabilidad está presente en una interacción hombre-dispositivo cuando permite facilidad de uso y facilidad de aprendizaje en su uso; mientras que experiencia de usuario se define como el conjunto de factores referentes a la

interacción del usuario y el dispositivo, cuyo resultado se refleja en una percepción positiva (negativa) del producto. Con experiencia de usuario uno de los aspectos más importante es la **percepción**.

La diferencia se puede ver en un ejemplo sencillo, una puerta de madera con pomo o la puerta de vidrio que dice hale o empuje. Con el ejemplo de la puerta de madera o vidrio con pomo o manilla, el usuario al verla, y dada su experiencia claramente, sabe que para pasar al otro lado debe rotar el pomo de la puerta o halar o empujar la manilla. Esta interfaz es fácil de aprender y el usuario la ve y sabe qué debe hacer para abrirla. Las puertas que normalmente vemos en edificios modernos, clínicas o aeropuertos, que se abren cuando detectan la presencia de una persona u objeto cerca de la misma, generan una percepción positiva del usuario y, antes de llegar a la puerta, no tiene ni que pensar en cómo abrirla, la puerta se abre sola. En este caso, se tiene un resultado positivo de percepción, de experiencia de usuario.

Para algunos autores UX es una filosofía de diseño que tiene por objetivo la creación de productos que resuelvan necesidades concretas de sus usuarios finales, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo. Hoy en día, para muchos UX puede ser un término nuevo, sin embargo, ha estado presente siempre en el desarrollo de las interfaces centradas en el usuario. Ojo, la interfaz centrada en el usuario no necesariamente es lo mismo que la experiencia de usuario.

Generalmente, la metodología utilizada para el Diseño Centrado en el Usuario se sigue por el esquema siguiente:

- Conocer a fondo a los usuarios finales, normalmente usando investigación cualitativa o investigación cuantitativa.
- Diseñar un producto que resuelva sus necesidades y se ajuste a sus capacidades, expectativas y motivaciones.
- Poner a prueba lo diseñado, usando test de usuario.

A decir verdad, la experiencia de usuario siempre se ha aplicado en el momento de desarrollar interfaces que permitan la interacción humano computador, desde tiempos remotos hasta la actualidad. Sin embargo, en los últimos años, me atrevo a decir que los últimos 5 años, el desarrollo de aplicaciones ha ido evolucionando en el desarrollo de interfaces muchísimo más simples, más funcionales y más fáciles de usar, que hasta un niño o anciano puede manejarla fácilmente. Así mismo, en el desarrollo de aplicaciones, se ha tendido a realizar diseños que agraden mucho más a los usuarios finales, es decir, que les de satisfacción al usarlo. La experiencia de usuario involucra investigación, arquitectura de la información y diseño de interfaces de usuario, es por eso que han surgido como carrera los profesionales UX.

En secciones anteriores, hemos definido el proceso que sigue el desarrollo de software, arquitectura y diseño. Hemos visto que la arquitectura de diseño de software y el diseño de la interfaz, prácticamente siguen la misma arquitectura, que en el caso del desarrollo de interfaces centradas en el usuario se recomienda seguir un proceso iterativo, entre el diseño, implementación y evaluación. Basados en esta arquitectura, el rol del especialista en experiencia de usuario involucra un proceso comunicativo

constante con el usuario, el cliente y el software. El objetivo final del especialista UX es lograr ese equilibrio entre el cliente y el usuario para así lograr un mejor producto. Generalmente, las metodologías usadas en el diseño de interfaces que involucra UX son bastante flexibles.

Rol de diseñador dentro del proceso de creación

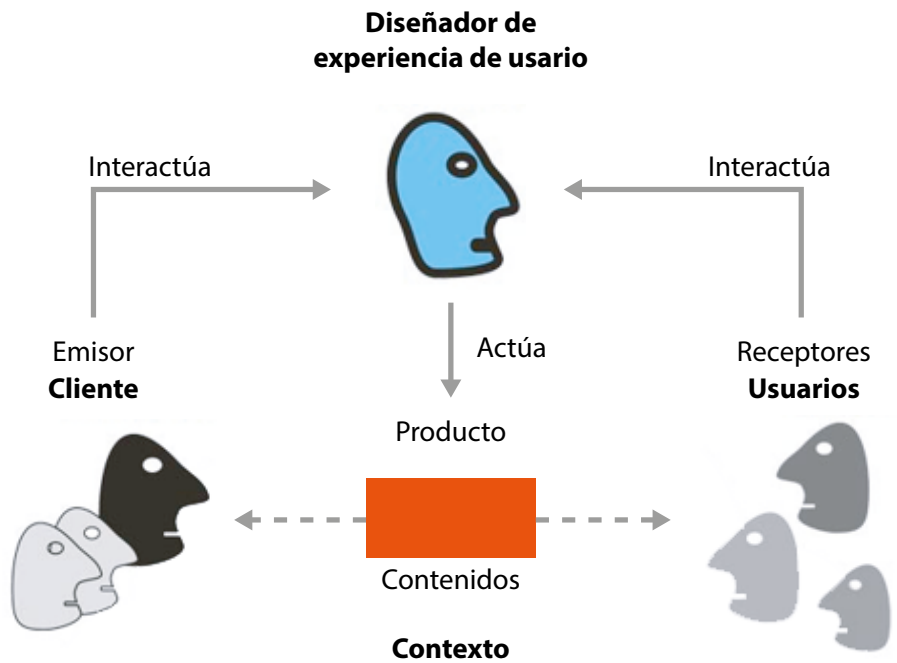


Figura 29. Rol del experto en experiencia de usuario (UX). Fuente: Rodrigo Ronda León (2013), <http://www.nosolousabilidad.com/articulos/uxd.htm>

En esta sección estudiaremos la metodología de diseño de experiencia de usuario que propone Ronda León (2013). ¿Por qué este autor y no otro? Las metodologías pueden variar ligeramente entre un autor u otro, sin embargo, las mismas dependerán mucho de diversos factores que tienen que ver con la percepción humana, la parte cognitiva y cómo individualmente cada persona desarrolla o percibe la idea. Podemos decir que es una metodología reciente propuesta por expertos en UX.

9.1. Metodología de Diseño de Experiencia de Usuario

Ronda León (2013) propone una metodología para seguir e integrar el diseño basado en experiencia del usuario, basado en cuatro elementos:

- **Etapas.** Este elemento permite definir y concretar los tiempos de ejecución de las distintas etapas involucradas en el desarrollo de interfaces basadas en UX.
- **Actividades o tareas.** Se definen las tareas que se deben llevar a cabo en las distintas etapas.
- **Técnicas.** Tiene que ver con las distintas técnicas conocidas o nuevas que pueden aplicarse para llevar a cabo las tareas o cumplir con las diferentes etapas, todas las técnicas aplicadas van, por supuesto, más relacionadas con el producto.

- **Herramientas.** Permite definir las herramientas de desarrollo más adecuadas que se recomiendan usar para desarrollar una aplicación UX.

9.1.1 Etapas

Las etapas consisten en cuatro fases, la primera es la etapa de investigación en donde se concentra la parte, si se quiere, más importante, porque es la etapa en la que se recogerá la mayor información posible acerca de las necesidades del usuario y la forma en que el usuario realiza sus tareas y del sistema, en general. Luego, esta información se organiza y procesa, en esta fase se convierte la información recopilada en producto. En la etapa de diseño se genera el diseño del producto a partir de la información organizada. Y, finalmente en la prueba se realiza una comprobación de la calidad del diseño.

Según el autor, estas etapas pueden verse tal cual en las etapas de desarrollo de software, en las que se pueden ver de dos formas: tanto lineal como iterativa (Ronda León, 2013). Muchos autores recomiendan la iterativa por la ventaja de ofrecer, en una etapa temprana del proyecto, un bosquejo de lo que será el diseño de la interfaz y, luego, se puede ir iterando junto con la implementación y las pruebas hasta llegar a un buen diseño de la interfaz, teniendo en cuenta que se mantiene de forma separada la capa de negocio de la capa de la interfaz del usuario a fin de que, al generar cambios importantes o significativos en la interfaz, no se produzcan cambios importantes en la capa de negocio.

Investigación:

En esta parte del diseño se generan todas las preguntas posibles necesarias para capturar tanto del cliente como del usuario y organización de los datos pertinentes al sistema. Es donde se definen, justamente, todas las tareas a realizar en las etapas definidas anteriormente y que permitirán hacer el diseño y las pruebas. Las actividades que se generen deben estar relacionadas con el proyecto, el usuario, el diseño y el contenido.

Organización:

En esta fase se relaciona la información recopilada en las etapas anteriores con la parte artística, aquí el rol del diseñador es muy importante, dado que se debe simplificar lo mejor posible y organizar toda la información obtenida en las fases anteriores. Es la etapa donde se definen flujos, contenidos, formas en que se presentarán los contenidos a los usuarios, haciendo corresponder tanto las necesidades del cliente como las del usuario, etc.

Diseño:

En esta etapa ya se definen los elementos tanto visuales como estructuras y flujo de procesos que harán las tareas y actividades para lo cual se crea el sistema. Esta última tiene que ver con la funcionalidad del sistema. Una de las partes del diseño es seleccionar los elementos visuales de cara al usuario que permitirán hacer las tareas pertinentes al sistema, aplicación o producto que satisfaga tanto las necesidades del usuario como las del cliente y la organización.

Prueba:

Finalmente, en esta fase se realizan las pruebas pertinentes a la interfaz como la funcionalidad completa del producto, entre las pruebas se encuentra las de prototipo y funcionalidad, según el enfoque iterativo de diseño de software.

9.1.2 Actividades

Durante esta fase se generan las actividades y tareas que se deben realizar en cada una de las etapas de investigación, organización, diseño y pruebas.

9.1.3. Técnicas

Las técnicas están relacionadas con el todo del proyecto, no necesariamente con las actividades o las etapas, más bien se centra en aplicar técnicas que tengan que ver, por ejemplo, con realizar actividades y generar una tormenta de ideas, referentes por supuesto con el producto que se quiere lograr. Algunas de las técnicas pueden ser: entrevistas con usuarios, clientes, expertos del tema o del área, realizar un mapa de ideas. Estas técnicas pueden realizarse a través de foros de discusión, relaciones persona a persona, etc. Consiste, básicamente, en aplicar técnicas para el manejo de la información y generación de ideas.

9.1.4. Herramientas

Las herramientas a utilizar varían según el presupuesto, se pueden usar herramientas de acceso libre o pago por licencia. Hoy en día, existen muchas herramientas de trabajo colaborativo que permiten una fluidez comunicacional bastante amplia en las que se puede tener un espacio virtual de **co-working** o colaborativo para compartir ideas, tareas, dificultades encontradas en el medio, etc. Muchas son de uso gratuito y otras requieren pago. Las herramientas que se usen van a variar dependiendo de las posibilidades económicas.

Tema 10.

Evaluación Heurística

Según la RAE, **heurística** proviene del griego *heuristikein* y significa “hallar”, “inventar”. La define como:

- Técnica de la indagación y del descubrimiento.
- Búsqueda o investigación de documentos o fuentes históricas.
- En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos como por tanteo, reglas empíricas, etc.

La interfaz de usuario es un área bastante flexible dado que depende de muchos factores variables desde la psicología cognitiva y la forma tan individual como lo es la capacidad de percibir, hasta el cliente, ambiente u organización. Para diseñar una interfaz se requiere de un conjunto de pautas que son elegibles, por lo que prácticamente cada diseñador o especialista en usabilidad tiene su propio conjunto de heurísticas, sin embargo, la mayoría de estas pautas se superponen. Generalmente, los expertos no están en desacuerdo con los elementos, principios o enfoques que constituyen una UI, pero no están de acuerdo en cómo organizar estos principios en un conjunto de reglas operacionales.

La heurística se puede usar de dos maneras: durante el **diseño**, para ayudarlo a elegir entre diseños alternativos; y durante la **evaluación heurística**, para encontrar y justificar problemas en las interfaces. Antes vamos a conocer algunas reglas heurísticas definidas por varios expertos.

10.1. Las 10 reglas heurísticas de Nielsen

Las 10 reglas de Nielsen (1993) constituyen una guía que permite identificar posibles problemas de usabilidad, dichas reglas establecen los criterios más importantes a mantener, ellos son:

1. Estado del sistema visible. Se debe mantener informado al usuario acerca del estado del sistema.
2. Fácil de aprender. Utilizar el mismo lenguaje de los usuarios, en la medida de lo posible, usando términos conocidos por los usuarios.
3. Ofrece opciones de corrección de errores causados por el usuario, con la opción de deshacer acciones.
4. Aplica las normas y convenciones de los sistemas conocidos.
5. Prevé errores. Más que informar sobre posibles errores, lo mejor es prevenirlos antes de que ocurran.
6. Simplicidad al evitar que el usuario necesite memorizar datos para realizar una acción.
7. Que el usuario sea capaz de personalizar la interfaz, ofreciendo opciones de personalización.
8. Simplifica la interfaz lo más posible, evitando la información irrelevante o decorativa.
9. Usa mensajes de error que expliquen qué falló y cómo solucionarlo.
10. Entrega documentación fácil de encontrar que explique las tareas más importantes.

Si observamos las heurísticas de Nielsen, básicamente intenta mantener los principios básicos de usabilidad.

10.2. Principios de Norman

Donald Norman (2002) propone algunos principios universales del diseño que pueden ser aplicados para cualquier diseño. Estos son:

1. Visibilidad. Parece ser un principio bastante obvio, mientras más visibles son las funciones más probabilidad de ser usada por los usuarios. Sin embargo, debemos hacer un balance entre visibilidad y simplicidad, como lo vimos en capítulos anteriores.

2. Retroalimentación o *feedback*. El diseño o prototipo debe ser capaz de ofrecerle respuestas al usuario por cada acción que realice, lo que no se dice no existe. No habrá manera para el usuario de saber si se realizó una acción o no correctamente si no obtiene respuesta.
3. Mapeo natural. Que haya una perfecta relación entre los controles o botones y la acción que permite hacer.
4. Claridad. Que le permita al usuario saber cómo usar el sistema, ofreciendo la percepción correcta de los elementos visuales.

10.3. Las 8 Reglas de Oro de la Usabilidad de Ben Schneiderman

Schneiderman (1985), otro de los gurús de la usabilidad, describe lo que él denomina **las ocho reglas de oro de la usabilidad** que deben ser tenidas en cuenta tanto para el diseño como para la evaluación heurística. Estas 8 reglas de oro fueron usadas por Apple y ya conocemos sus resultados.

1. Consistencia. Esta regla se refiere a que haya una consistencia entre los colores, iconos u objetos visuales y lenguaje. Esto debe realizarse en función de los usuarios, de esta manera será más fácil de usar y aprender.
2. Permitir atajos (*shortcuts*). A medida que el usuario se va volviendo experto, él solo quiere llegar rápido a la acción que le permite realizar la tarea que desea, así que si se le ofrecen atajos es mucho más fácil ejecutar las tareas.
3. Retroalimentación o *feedback*. Tiene que ver con mantener informado al usuario acerca el estado del sistema y del resultado de cada acción ejecutada.
4. Diálogo. Diseñar textos de diálogo para cerrar procesos que le permitan al usuario confirmar que la acción fue ejecutada exitosamente.
5. Manejo de errores. Ofrece una forma sencilla de corregir errores.
6. Permite la facilidad de regresar sobre sus propios pasos. Posibilidad de revertir acciones.
7. Da el control a los usuarios, esto le da poder y confianza al usuario.
8. Memorización mínima. Nuestra capacidad de memoria es corta, entre 3 o 5 objetos se pueden fácilmente memorizar, más de eso es imposible de recordar. Para nuestra mente es más fácil reconocer algo que memorizar, podemos valernos de esta capacidad para hacer objetos más reconocibles.

La atención humana es limitada y solo somos capaces de mantener 5 objetos/ideas en nuestra memoria a corto plazo al mismo tiempo. Por esto, la interfaz debe ser lo más sencilla posible y con una jerarquía de información evidente. Elige reconocimiento en vez de recuerdo. Reconocer algo es más fácil que recordar algo porque el reconocimiento incluye claves que nos ayudan a recordar objetos almacenados en nuestra basta memoria. Por ejemplo, es común que prefiramos preguntas de opción

múltiple a preguntas abiertas, ya que, en la primera, podemos reconocer la respuesta y no saberla de memoria como en el segundo caso.

10.4. Cómo realizar la Evaluación Heurística

Generalmente, se crea una especie de lista tipo *checklist* y se marcan qué principios de usabilidad se están aplicando en el diseño, cuáles tienen fallos o carencias y se da una solución de cómo podría mejorarse. Normalmente, esto lo hace un experto en usabilidad.

La evaluación heurística es un proceso de inspección de usabilidad, originalmente fue creado por Nielsen (1998), quien ha realizado una serie de estudios para evaluar su efectividad. Esos estudios han demostrado que la heurística favorece la relación costo-beneficio; el costo de encontrar problemas de usabilidad en una interfaz es generalmente más barato que los métodos alternativos. La evaluación heurística es un método de inspección, lo realiza un experto en usabilidad, alguien que conoce y entiende la heurística que acabamos de comentar y ha usado y visto muchas interfaces.

Los pasos básicos son simples:

1. Evaluar la interfaz a fondo, juzgando la interfaz sobre la base de la heurística que acabamos de analizar.
2. Hacer una lista de los problemas de usabilidad encontrados, las formas en que los elementos individuales de la interfaz se desvían de la heurística de usabilidad, marcando en un *checklist* elementos de la interfaz o principios de usabilidad que no se cumplan.
3. Volver a inspeccionar la interfaz al menos dos veces y enfocarse en elementos particulares de la interfaz, siempre sobre las bases de las reglas de evaluación heurística.
4. No limitar el uso de las reglas de evaluación heurísticas.

Glosario

Checkbox

Elemento visual de interfaz gráfica que le permite al usuario seleccionar una o varias opciones de una lista de opciones.

Co-working

Modalidad de trabajo colaborativo interdisciplinario.

e-commerce

Se refiere a plataformas de comercio electrónico.

GUI

Iniciales en inglés de *Graphic User Interface*, Interfaz gráfica de usuario.

Hardware

Componente físico o dispositivo electrónico que compone un computador o interactúa con el computador a través de los elementos físicos de entrada y salida.

Hiperlink

Los hiperlink son textos que permiten enlazar con otros documentos desde la web.

IxD

Iniciales para denotar diseño de interacción entre el UI y UX.

Query

Consulta a una base de datos o motores de búsqueda.

MIT

Massachusetts Institute of Technology.

Radiobutton

Elemento visual de interfaz gráfica que le permite al usuario seleccionar una opción de una lista de opciones.

Shell de Linux

Lugar donde se dan las instrucciones por línea de comandos usando el sistema operativo Linux.

Software

Programa que se ejecuta en el computador.

UI

Iniciales en inglés de *User Interface* para denotar el tema de interfaz del usuario.

Usabilidad

Término que se refiere a la capacidad que tiene un sistema o aplicación de ser preferido en su uso por el usuario.

UX

Iniciales en inglés de *User Experience* para denotar el tema de experiencia de usuario, disciplina muy mencionada recientemente y que se tiende a confundir con los términos de UI y diseño de interfaz centrado en el usuario.

Widget

Elemento visual o gráfico que permite al usuario interactuar con la aplicación, software o sistema; por ejemplo: ventana, botón, menú, etc.

WYSIWYG

En inglés *what you see is what you get*. Estrategia para representar texto en la pantalla de forma exacta a cómo se ve en apariencia cuando se imprima.

Enlaces de interés

Curso de Interfaz del Usuario de la profesora Adelaide Biachini

Contenido y recursos del curso dictado por la Profesora Adelaide Biachini en la Universidad Simón Bolívar, de Caracas, Venezuela.

<https://ldc.usb.ve/~abianc/materias/ci4325/>

LN Creatividad y Tecnología

Blog de propósito general en el área de creatividad y tecnología.

<http://www.luisan.net/blog/>

Hall of Shame

Proyecto que recoge las buenas y malas interfaces diseñadas y que permite orientar al diseñador de interfaces sobre qué cosas no hacer y porqué, así como qué cosas se pueden mejorar y cómo.

<http://hallofshame.gp.co.at>

Mercado Libre

Plataforma e-commerce para la compra y venta de cosas nuevas y usadas.

www.mercadolibre.com

Shopify

Plataforma e-commerce que permite la creación de tiendas personalizadas, orientadas al estilo del vendedor, está más centrado en el usuario.

www.shopify.com

Magento

Plataforma e-commerce que permite la creación de tiendas personalizadas, orientadas al administrador de las ventas.

www.magento.com

Cursos Abiertos del MIT (Massachusetts Institute of Technology)

Sitio web que ofrece diferentes cursos en el área de tecnología que ofrece el MIT.

<https://ocw.mit.edu>

Madeja Junta de Andalucía

Es un proyecto cuya misión es proporcionar un entorno que permita a todos los implicados en el desarrollo y en la explotación del software una referencia clara de cuáles son las directrices que han de guiar esta actividad, así como dar a conocer los recursos y herramientas que están a su disposición.

<http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/122>

Nube de Datos

Página dedicada al análisis de datos en R, Excel, Access, VBA y SQL.

<http://nubededatos.blogspot.com>

TIBCO

TIBCO Spotfire es un proyecto de divulgación de la información en el área de manejo de datos. Facilita el acceso, el análisis y la creación de informes dinámicos sobre el manejo de datos. Ofrece un valor inmediato tanto si se es un investigador de mercado, un representante de ventas, un científico o un ingeniero de procesos, ya que le permite identificar rápidamente tendencias y patrones en sus datos críticos de negocio.

<https://docs.tibco.com>

El Baúl del Programador

Proyecto que pretende mantener recursos y artículos de interés en el área de la tecnología para fines académicos y comerciales.

<https://elbauldelprogramador.com/14-ejemplos-visualizacion-datos/>

Reglas de la Sección 508

Define un conjunto de reglas de accesibilidad que un producto de software debe cumplir si desea ser utilizado por algún organismo del gobierno de los Estados Unidos de América.

<http://www.section508.gov/>

Pautas de Accesibilidad de la Web

Pautas establecidas y sugeridas por el consorcio del W3C. W3C es un consorcio compuesto por una comunidad internacional, cuyos miembros se dedican a tiempo completo y mediante trabajo público a desarrollar los estándares de la Web.

<http://www.w3.org/TR/WAI-WEBCONTENT/>

Indiscripts

Es un proyecto de divulgación de la información referente al diseño gráfico. Ofrece conocimiento y herramientas de diseño.

<http://www.indiscripts.com/>

Blog No Solo Usabilidad

Revista sobre personas, diseño y tecnología en torno a interfaces del usuario.

<http://www.nosolousabilidad.com/articulos/uxd.htm>

Bibliografía

Referencias bibliográficas

- Boehm, B., (1986). *A Spiral Model of Software Development and Enhancement*, ACM SIGSOFT Software Engineering Notes, ACM, 11(4):14-24, Agosto.
- Burbeck, S., (1992). *Applications Programming in Smalltalk-80 (TM): How to use Model-View-Controller (MVC)*. Recuperado de <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>
- Card, S. K., Moran, T. P., Newell, A., (1986) The Model Human Processor: An Engineering Model of Human Performance. In K. R. Boff, L. Kaufman, & J. P. Thomas Eds., *Handbook of Perception and Human Performance*. Vol. 2: Cognitive Processes and Performance.
- Card, S. K., Moran, T. P., Newell, A., (1983). *The Psychology of Human Computer Interaction*. Lawrence Erlbaum Associates.
- Card, S. K., Moran, T. P., Newell, A., (1980). *The keystroke-level model for user performance time with interactive systems*. Communications of the ACM. 23 (7): 396–410.
- Dix, A., (2007), *Taxonomy of Clutter Reduction for Visualización de la información*, IEEE Transactions on Visualization & Computer Graphics (IEEE).
- Dix, A., (1993) *Human computer interaction*. Prentice Hall Englewood Cliffs, NJ.
- Gamma, Erich and Helm, Richard and Johnson, Ralph and Vlissides, John M., (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*, 1ª ed. Addison-Wesley Professional.
- Lores J., Granollers T., Lana S., (2001). *Introducción a la interacción persona-ordenador*. Universitat de Lleida. Libro electrónico. Recuperado de <http://www.aipo.es/>
- Norman, D., (2002), *The Design of Everyday Things*. Basic Books, Inc. New York, USA.
- Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.
- Ronda León, R., (2013). Diseño de Experiencia de Usuario: etapas, actividades, técnicas y herramientas. En: *No Solo Usabilidad*, nº 12, 2013. Recuperado de: <http://www.nosolousabilidad.com>
- Shneiderman, B., (1986). *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley Longman Publishing Co.
- Wickens C. D., (1992). *Engineering Psychology and Human Performance*, 2ª ed. Harper Collins, Nueva York, NY.

Bibliografía recomendada

Baecker, R., Grudin, J., Buxton, W. & Greenberg, S., (1997). *Readings in Human-Computer Interaction: toward the year 2000*, 2ª ed. Morgan-Kaufmann.

Berlin J., Berg W., Scott P., (1981). *Graphics and Graphic Information Processing*. De Gruyter.

Laurel, B., (1990). *The Art of Human-Computer Interface Design*. Addison-Wesley.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., Carey, T., (1994). *Human-Computer Interaction*. Addison-Wesley.

Tidwell, J., (2006). *Designing Interfaces - Patterns for Effective Interaction Design*. Ed. O'Reilly.

Tufte, E., (2001). *The visual display of quantitative informations*, 2ª ed. Graphics Press.

Ware C., (2004). *Information Visualization: Perception for Design*. Morgan Kaufmann Publisher.

Wilbert O. G., (2007). *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*, 3ª ed. Wiley Publishing, Inc.

Agradecimientos

Autora

Dra. D.ª Alexandra La Cruz Fuente

Departamento de Recursos para el Aprendizaje

D.ª Carmina Gabarda López

D.ª Cristina Ruiz Jiménez

D.ª Sara Segovia Martínez

