Sepa por qué los programadores le contestan "no se puede" cada vez que usted pide algo razonable y sencillo

Daniel Mordecki

BIBLIOTECA Concreta

Tabla de Contenido

Introducción

Hace ya unos cuantos años, por 1996, vino a Uruguay a dar una charla Alain Jorda, un español experto en temas de Internet. Yo trabajaba en aquella época como consultor en el área de e-business, y un cliente al que recién le habíamos construido un sitio de comercio electrónico nos invitó a la programadora del sitio y a mí a una reunión con este experto y el equipo de trabajo del sitio. Contra mis resquemores, Jorda resultó no solamente una persona muy agradable de tratar, sino un excelente asesor con opiniones muy interesantes sobre los temas más variados de Internet. En el medio de la reunión, decidió hacer en vivo y en directo una compra en el sitio de comercio electrónico que habíamos desarrollado: fue literalmente imposible. La programadora se levantó al menos 4 veces para indicarle cómo registrarse, cómo buscar, cómo seleccionar, cómo pagar y cómo hacer muchas otras cosas, pero a pesar de eso, un experto en Internet, con su tarjeta de crédito en la mano, su notebook conectado a Internet y la asistencia de la programadora del sitio fueron incapaces de comprar cualquier cosa que se vendiera allí. Probablemente con media hora más de intentos, hubiéramos podido realizar la transacción, pero los 15 minutos que le dedicamos simplemente no fueron suficientes: cuando quería comprar, el sitio le pedía que se registre, cuando se registraba, el sitio ya no se acordaba de lo que había seleccionado para comprar, cuando hacía ambas cosas, faltaba incluir en el registro la dirección de envío, y así sucesivamente hasta agotar la paciencia, que en esa ocasión tuvo la cota de 15 minutos.

Al salir de la reunión mi forma de ver el software y los sitios Web había cambiado, no tenía ninguna respuesta, ni siquiera tenía formulada la primera de las preguntas que tendría que contestar, pero había percibido por primera vez en mi vida que los programas que los consultores, ingenieros, analistas y programadores creamos para los usuarios están a una distancia sideral de lo que ellos, desde su propio punto de vista, esperan recibir. Al resto de los participantes no les pareció tan grave el problema y creo que en definitiva para la mayoría pasó desapercibido; al fin de cuentas, con un poco más de esfuerzo hubiéramos terminado la transacción. A mí esto me preocupó más todavía: no solo no creábamos los programas que los usuarios esperaban sino que además éramos absolutamente miopes para ver el problema. Mi decisión fue terminante,

no tenía otra opción que cambiarme de bando: pasarme al lado de los usuarios. Y así lo hice.

Desde aquella fecha he buscado y rebuscado en cuanto material e información pasa por mis manos todo lo que pueda referir a otros informáticos y no informáticos que sientan lo mismo. Con alegría constaté que son muchos más que los que yo pensaba y que la elaboración teórica y la práctica para entender, atacar y resolver este tipo de problemas es amplia y sólida. Con el tiempo, he sistematizado un conjunto de ideas que parten fundamentalmente de esos materiales y tienen en mi propia experiencia algún complemento o matiz. Son esas las ideas que terminaron plasmadas en este libro. No me queda muy claro si este será el libro que los demás quieran leer, pero no tengo dudas de que es el libro que quería escribir.

Los programadores

A lo largo de este libro se menciona reiteradamente el término programador, por lo que vale la pena definirlo antes de comenzar y hacer algunas precisiones al respecto. El término programador englobará a todos aquellos informáticos que trabajan en la creación de aplicaciones y sitios Web: Ingenieros de sistemas, arquitectos de software, analistas de sistemas, programadores, técnicos de soporte y cualquier otro título o función que integre el organigrama de un equipo de desarrollo o área de sistemas.

De aquí en adelante, el término programador engloba a todos aquellos para los que la palabra "cliente" representa la primera parte de la combinación Cliente/Servidor y no una persona de carne y hueso que pone el dinero del que sale nuestro sueldo. Si jamás oyó mencionar la expresión Cliente/Servidor o le parece que todo este párrafo carece de sentido, relájese, usted está muy lejos de ser un programador.

Por mi parte, me declaro programador confeso. Adoro la programación, las computadoras, los autos y todos los aparatos en general, cuanto más complicados mejor. A pesar de que ya no desarrollo software en forma profesional, sigo haciéndolo como hobby para mantener www.mordecki.com, para procesar un archivo de texto plano o para hacer

cualquier cosa en la que pueda programar un poco. Mi último hallazgo es Perl, un lenguaje de programación para scripts de Internet que todavía se programa en editor de texto, con una sintaxis entreverada y una potencia milagrosa al que por lo demás, los programadores Visual Basic no comprenden fácilmente: una verdadera joya. En realidad no programo mucho, pero conservo intactos todos los vicios de los programadores, es decir, prefiero perder el triple del tiempo para encontrar como hacer algo con una macro, antes que teclear delete, delete, enter[1] ciento cincuenta veces, jamás elijo las opciones default de ningún programa, siempre instalo personalizado y antes de usar nada reviso todas y cada una de las opciones disponibles. Todavía abro a diestra y siniestra ventanas del viejo y querido DOS y sigo utilizando comandos inexistentes para los simples mortales como attrib, fc o xcopy: los programadores de ley saben exactamente de qué hablo. Reviso los equipos de todos los que están a mi alrededor, me hago rogar un poco, pero puedo pasarme un día entero para resolver un problema menor si alguien me hace una pregunta: la única ley es que cuanto más difícil mejor. Leo más manuales y libros técnicos que novelas, inclusive de aplicaciones que ni uso ni voy a usar. El único vicio que perdí es el de llevar la cuenta de los procesadores de los PC, me quedé en el Slot II, hace algunos años. A pesar de todo, cada uno de los días que siguieron a aquel día de la reunión con el experto español he intentado ponerme en los zapatos de mis víctimas, como los Lubolos de las comparsas[2].

A quién está dirigido este libro

El libro está dirigido a todos aquellos que, sin pertenecer al mundo informático, a diario viven la dura tarea de interactuar con los departamentos de sistemas y los programadores para que sus proyectos cobren vida. Ya sea dentro de las empresas que desarrollan software con fines comerciales, o en los propios departamentos internos de sistemas de organizaciones con y sin fines de lucro, hay miles de personas que día tras

---

[1] *Usar los nombres de las teclas en Inglés, ¡otro vicio!*

[2] *En el carnaval del Uruguay, se llama Lubolos a los blancos que tocan el tamboril junto a los negros en las comparsas. Los Lubolos se pintan todo el cuerpo y la cara de negro, para poner de manifiesto la vergüenza que un blanco debe sentir por la esclavitud y poder así ser aceptados en el rito carnavalero del candombe codo a codo con sus hermanos de color.*

día, reunión tras reunión, esperan y desesperan para que los sistemas que serán soporte de sus proyectos "se pongan en producción".

Para estos simples mortales la informática resulta inescrutable, así como le resultan incomprensibles las explicaciones de qué fue lo que atrasó el proyecto esta vez, o de por qué el cambio que solicitan es absolutamente imposible o por qué para cambiar un campo hay que desarrollar todo el sistema de nuevo. Les da lo mismo si la solución es migrar todo a Java o prenderle una vela a San Jorge: si funciona, que lo hagan de una vez por todas.

Las personas que trabajan en las áreas comerciales, de marketing, operaciones, al igual que los gerentes que tienen la responsabilidad de dirigir una empresa, dependen actualmente de los sistemas como del oxígeno que respiran: prácticamente nada importante es posible hoy en día en una empresa sin un sistema informático. Como contrapartida, los sistemas y los departamentos de sistemas se han transformado en la peor clase de problema: aquel que es inmune a todo tipo de solución, que absorbe inmutable todo el dinero, todo el esfuerzo y todas las iniciativas de cambio.

Este libro está escrito desde la primera hasta la última letra para ellos, las personas que lidian un día sí y otro también con nosotros los programadores. Si al menos uno de ellos consigue a partir de este libro gerenciar un proyecto que implique interacción con el departamento de sistemas, generando menos angustias que en sus proyectos anteriores, entonces mi objetivo estará cumplido a satisfacción. Se que este libro despertará la ira de la mayoría de los programadores que lo lean, pero si consigo mi objetivo, el precio a pagar habrá sido justo.

La estructura del Libro

El libro se divide en dos: una primera parte que plantea el diagnóstico de la situación actual en lo que respecta a las aplicaciones informáticas, al software y a los sitios Web y una segunda parte que propone una metodología de diseño que tiene como objetivo resolver los problemas expuestos en la primera parte.

Varios amigos que generosamente leyeron y criticaron este libro me hicieron notar, entre numerosas correcciones y defectos, que la primera parte que abarca los capítulos 1, 2 y 3 es más amena y divertida que la segunda parte. Tienen razón sin lugar a dudas. Es que la realidad cuasi delirante en la que nos sumerge el software permite poner alguna pizca de humor al diagnóstico. La segunda parte, que propone una metodología de trabajo, sin aspirar al tono duro de un libro técnico, está obligada a comportarse con más seriedad y menos humor (negro).

Primera Parte

Así está el mundo, amigos.

8.

Te amo, te odio, ¡dame más!
Peperina, Charlie García

Capítulo 1

¡Las computadoras me odian!

Te amo...

Sin ánimo de entrar en discusión sobre la historia de la computación, podemos asumir que hace 50 años, tal vez un poquito más, las computadoras vieron la luz por primera vez. Y lo hicieron para quedarse. Despreocupadas, comenzaron a tomar parte del trabajo de científicos, gobiernos y empresas, hasta que treinta años después, exactamente en 1981, la aparición del PC consolidó los intentos de fabricar un computador de uso personal. Allí comenzaron a transformarse en un objeto cotidiano también para los simples mortales. De ahí a la proliferación de microprocesadores y software en todo tipo de equipamiento y dispositivos hay un paso. Hoy desde la heladera al automóvil, recorriendo la lista de los objetos más diversos, inclusive algunos inverosímiles (¿recuerda el calzado deportivo con lucecitas?, pues bien, tenía adentro un chip y software) han llevado la relación hombre-computadora a ser parte de nuestra experiencia diaria.

La vida de la mayoría de los ciudadanos de este planeta es prolífera en interacciones con chips, interfaces y software. Está regada generosamente de teclados, sonidos, timers, sensores, displays, luces, diodos, cables y líneas telefónicas, toda una serie de objetos que se entrecruzan unos con otros para inmiscuirse amablemente en nuestros quehaceres hasta ser parte de la propia rutina.

El impacto de la combinación *hardware digital + software*[3] en el diario vivir es asombroso, y prácticamente no tiene parangón en la historia. Ninguna tecnología anterior fue tan disruptiva, tan omnipresente y representó un cambio tan significativo en tan poco tiempo. La capacidad de percibir el contexto a través de dispositivos de lectura, procesar digitalmente la información y hacer que ésos u otros dispositivos de salida actúen en función de los resultados del procesamiento, parece ser la fórmula mágica a ser aplicada a cualquier disciplina para avanzar. Ahora la batidora puede detectar la consistencia de la masa para determinar la fuerza que debe hacer, el motor puede saber si el conductor aceleró de forma vehemente para mantener las revoluciones a la hora de cambiar de velocidad, y la cámara fotográfica puede reconocer el iris del fotógrafo

[3] *Llamaremos a lo largo del libro "productos basados en software" a los productos que contienen la combinación hardware digital + software.*

para adaptarse a sus preferencias. Todo esto condimentado por una baja frenética de costos: la ley de Moore, que propone que cada 24 meses se duplica la capacidad de procesamiento por unidad de costo, viene cumpliéndose empecinadamente desde hace más de dos décadas, haciendo viables rápidamente ideas que hace apenas dos o tres años eran un sueño.

La invasión de los chips y software es tan ubicua, tan omnipresente que parece que las otras industrias sólo pueden avanzar digitalizándose. Revise los últimos avances en la industria automotriz: el encendido electrónico, la inyección electrónica, el control electrónico de los frenos (ABS), distribución electrónica de la fuerza de frenado (EBV), el control electrónico de tracción (ASR), bolsas de aire de seguridad con control electrónico (Airbag), bloqueo electrónico del encendido, computadora de a bordo, caja de cambios electrónica (Triptronic), posicionamiento satelital electrónico (GPS), computadora para asistir al tallerista. El automóvil es hoy una computadora sobre ruedas. Los ingenieros mecánicos y diseñadores han tenido el celo necesario para que manejar sea una tarea cada vez más sencilla y placentera: quien maneja el auto no necesita saber lo complejo que es el software que controla cada uno de los mecanismos que lo asisten en la conducción. De hecho, para la mayoría aplastante de los conductores, la idea de que cuando mueven el volante de su auto o pisan el freno levemente producen el mismo efecto que tecleando un enter en su PC no se les pasa siquiera remotamente por su cabeza. De la evolución de la barra de dirección puramente mecánica a la dirección servoasistida controlada por software, perciben las ventajas sin necesidad de gastar tiempo en comprender los detalles.

Hemos vivido años de una primavera cautivante y permanente, donde las aplicaciones, los dispositivos y las novedades se suceden unas a otras en una forma incesante y vertiginosa, tal como florecen todas y cada una de las plantas cuando termina el invierno.

Te odio...

El confort y placer que nos brindan los dispositivos con los que convivimos tiene a la vez un matiz de problema, una faceta de insatisfacción. Muchas veces nos sentimos como cuando en el Picnic los

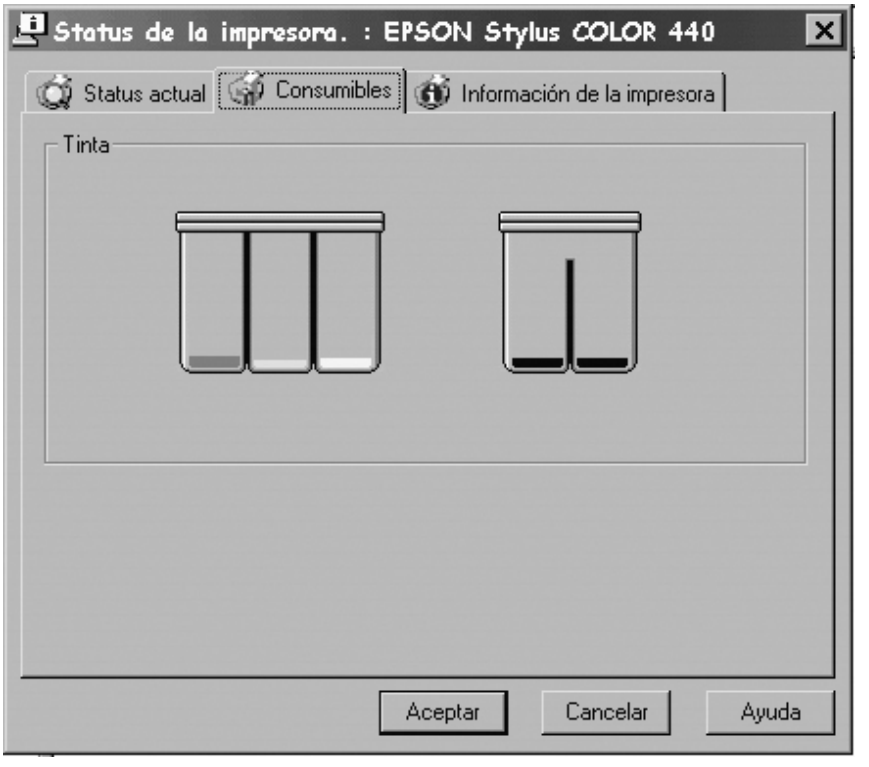
sandwiches se ensuciaron con un poquito de arena. Al principio creemos que no importa, que es lo mismo, que siguen exquisitos como al principio. Pero después de los primeros bocados comenzamos a percatarnos que esas pequeñas migas de arena incomodan más de lo previsto, hasta que el almuerzo queda sin terminar. Los productos basados en software, con el computador personal como rey de la selva digital, nos provocan a todos o a la mayoría, una sensación de impotencia, de incomprensión, que a veces aflora inclusive en forma de angustia y bronca y que se constituye en la otra cara de la moneda del confort y placer que los mismos productos nos aportan.

La impresora

Las impresoras son un caso patético. Tienen la osadía de confabularse en nuestra contra para imprimir lo que no queremos, tener siempre configurado A4[4] cuando el papel es Carta, y Carta cuando el papel es A4, negarse a imprimir cuando estamos más apurados y estropear la última hoja que nos quedaba a las 2 de la mañana. Mi impresora Epson Stylus 440, digno representante de la estirpe, tiene dos cartuchos: uno para la tinta negra y otro para la tinta de color. Tiene un utilitario fabuloso que me indica cuánta tinta hay en cada cartucho, separada por color, con un gráfico que representa los cartuchos y su nivel de tinta, no solo fácil de entender sino extremadamente simpático. Sin embargo, cuando un cartucho se queda sin tinta, toda la simpatía desaparece y es sustituida por un mensaje que dice: "su impresora está fuera de línea, verifique la conexión" y da la opción de Reintentar o Cancelar la impresión. En ese momento es imposible saber si los cartuchos tienen tinta, si falta papel o si hay algún otro problema. El utilitario de la impresora arroja ese, y solamente ese mensaje hasta que arreglemos el error. No hace falta decir que con dos cartuchos negros y dos color se paga el costo completo de la impresora, por lo que la alternativa de empezar a probar cartuchos para ver si puedo imprimir es desagradable y cara. ¿Por qué me castiga? ¿Por qué en el momento que yo más necesito saber cuánta tinta hay en los cartuchos se niega a decírmelo? ¿Dónde dejó la simpatía de antaño?

---

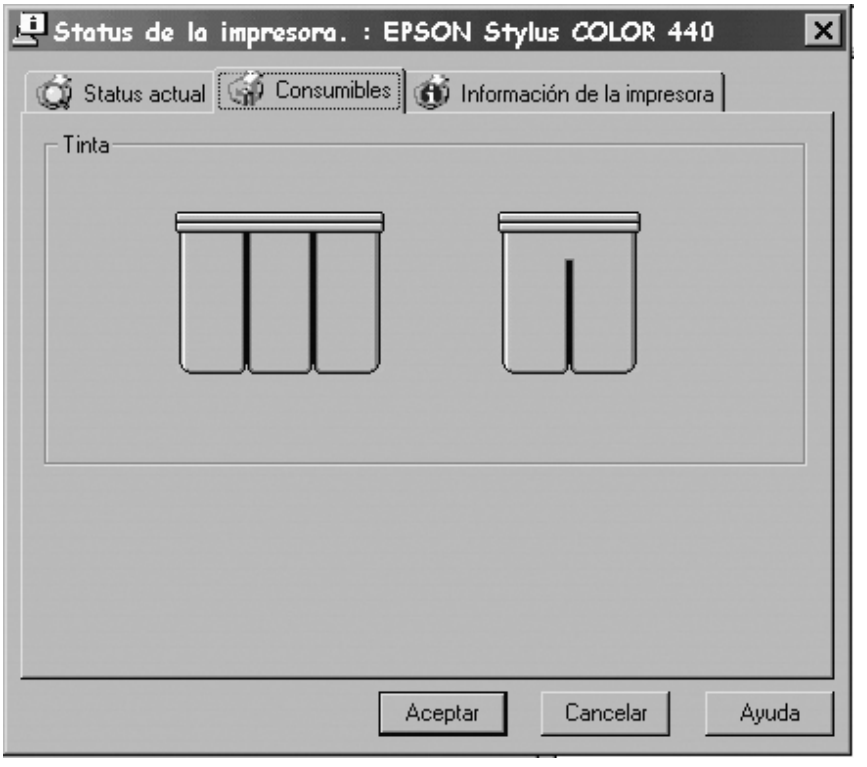
[4] De paso: ¿De dónde salió el famoso papel A4? Todos éramos felices en nuestra ignorancia hasta que apareció el papel A4 en nuestras vidas



Lo que veo mientras ambos cartuchos tienen tinta

Increíblemente, en este caso la explicación se remonta al nacimiento del PC, 20 años atrás. La impresora que acompañaba al PC original era de matriz de puntos, en la mayoría de los casos una Epson MX 80. La impresora movía el rodillo del papel con un motor de pasos (como todas las impresoras hasta nuestros días), que oficia a la vez de propulsor y freno: hace avanzar o retroceder el papel en pequeñísimos escalones y cuando está quieto frena el carro, dejando el papel quietecito en su lugar. La impresora contaba también con una perilla que permitía mover el carro en forma manual, pero antes había que liberar el motor presionando el famoso botón "Off Line". De esta manera, la impresora quitaba la corriente del motor de pasos, el freno se soltaba y el carro se podía mover libremente. De lo contrario, si no presionaba el botón "Off Line" había dos opciones: romper el motor o romper la perilla plástica. Cuando la impresora estaba "Off Line" o fuera de línea, el PC perdía control, dado que era imposible mover el motor y la perilla a la vez. Veinte años

después mi impresora Epson no tiene perilla, ni botón "Off Line", sin embargo sigue comportándose como si lo tuviera, y ante el menor problema deja de funcionar y me echa en cara "estoy fuera de linea, resuelva sus problemas usted solo".



Lo que veo cuando se termina uno de los dos cartuchos. Adivina, adivinador ¿Cuál de los dos cartuchos se terminó?

La ley universal dice que la impresora hace siempre lo que quiere, independientemente de su conocimiento, dedicación y paciencia: si están bien los márgenes está mal el tamaño de papel, si hay tinta de color, falta tinta negra. Si imprime con calidad, queda toda la hoja empastada de tinta. Si no imprime con calidad, quedan los textos borrosos. Si cancelo un trabajo, sigue imprimiendo, si clickeo el botón de imprimir se mueven todos los iconitos pero la impresora no imprime. Y cuando acertamos todos los parámetros, salen esas malditas hojas llenas de jeroglíficos incomprensibles.

La impresión digital es un avance impactante. No es que ahora se hace lo mismo que antes pero más rápido, sino que se trata de un nivel cualitativo distinto, donde se pueden crear documentos impresos que antes eran impensables, inimaginables. Pero hacemos todo eso sin que la impresora consiga cautivarnos, y una vez que asumimos que la impresión digital es un derecho adquirido, comenzamos a preocuparnos de todos esos inconvenientes con los que nos importuna.

El Cajero Automático

Imagine que usted entra a un bar, se sienta en una mesa y llega el mozo:

- ¿Qué se va a servir?
- Un café, por favor.
- ¿En pocillo o en vaso?
- En vaso...
- ¿Fuerte, suave o intermedio?
- Este..., cualquiera, da lo mismo
- Le traigo intermedio entonces, ¿está bien?
- Si, si, tráigame por favor el café.
- ¿Lo quiere bien caliente, o no tanto?
- ¿Me va a traer el café?
- Pero es que no puedo traerle el café si no me dice si lo quiere bien caliente o no tanto, después se quema y dice que es mi culpa...
- Tráigame el café bien caliente, ¡pero por favor tráigamelo ya!
- ¿Azúcar o edulcorante?
- Lo que sea, ¡siempre que no me pregunte nada más!
- Señor, no se moleste, pero si usted fuera diabético, y yo le traigo azúcar en vez de edulcorante, menudo problema tendría...
- Está bien, está bien, tráigame edulcorante.
- Me va a disculpar, pero no tenemos café

Así se comporta el cajero automático. Después que contesté puntualmente todas y cada una de sus inquisiciones certeras necesarias para retirar dinero, me informa que no tiene dinero en su billetera. ¿Por qué no me lo dice al comienzo, cuando pulso “extracción”?, o mucho mejor, ¿por qué directamente no agrega al botón extracción una aclaración de que no hay efectivo disponible y anula la opción? Lo que es una descortesía

inaceptable en un humano, es también una descortesía inaceptable en un equipo.

Esta no es la única forma en que me irrita el cajero automático. Siempre que quiero retirar dinero el Cajero Automático me pregunta: "¿caja de ahorro o cuenta corriente?". Yo creía que se trataba de una pregunta importante, al fin de cuentas el cajero tiene dentro mi dinero y es bueno que se cuide bien antes de dárselo a alguien equivocado. Pero un día descubrí que en realidad, me lo pregunta sólo para ver si consigue que me equivoque tontamente: él sabe que yo tengo cuenta corriente y no tengo caja de ahorro, a pesar de lo cual me pone el señuelo ahí cada vez que entro. Una vez por mes me equivoco y ahí pega el zarpazo: "cuenta inválida". Yo podría tolerarlo, ya dijimos que tiene mi dinero dentro y ese es un buen motivo para tolerar una chanza, pero el Cajero Automático se burla de mí con alevosía y premeditación, tal como el mozo: me deja seguir adelante tranquilo, digitar si quiero pesos o dólares, el importe, y todos los demás datos y recién cuando terminé todo mi trabajo contesta con el mensaje de error y me manda nuevamente al principio. Es eso lo que me enfurece.

El Cajero Automático es un dispositivo electrónico digital de extrema utilidad, que genera un consenso general de insatisfacción. En Uruguay llegó a votarse una ley para prohibir que se paguen las jubilaciones utilizando cajeros automáticos. Muchos vieron en esta ley una burla al progreso y tenían razón. Puede interpretarse además como el reflejo de un sentimiento de ingratitud hacia un aparato ingrato.

El diseño de sus aplicaciones, desarrolladas hace más de 30 años, responden a las necesidades de un pequeñísimo procesador y una comunicación de 300 baudios, de un elevado costo. Hoy el cajero cuenta con una línea dedicada, que pasa el 99% del tiempo inactiva, con un enorme procesador, que pasa el 99% del tiempo inactivo, con una cantidad importante de memoria RAM, que pasa el 99% del tiempo vacía y un gigantesco disco duro, que está mayormente desocupado. No se ha evolucionado un milímetro en la forma de relacionarse con sus visitantes de todos los días, que tienen que contestar una y otra vez las mismas preguntas, para recibir una y otra vez las mismas respuestas. Las personas lo utilizan a diario, y valoran su utilidad. Es seguro, cómodo y práctico.

Pero el cajero no los cautiva, no los enamora, sino que más bien los intimida.

La alarma del auto

Cada tanto, me detengo unos minutos en el estacionamiento del supermercado para verificar si todo sigue igual y sistemáticamente obtengo la confirmación: en diez minutos suenan al menos 4 alarmas de auto. No se trata de que roben 4 autos cada diez minutos, sino que los dueños de los autos no consiguen abrir la puerta sin que suene la alarma. Verifíquelo usted mismo, alcanza con diez minutos.

Las alarmas para automóviles no son otra cosa que una computadora, un dispositivo de hardware digital controlado por software. En su carrera por vender han incorporado decenas de funciones que previenen situaciones riesgosas, activan y desactivan las trancas de las puertas del auto, difieren el bloqueo, etc. Esta combinación ha llegado a tal límite, que a algunos conductores les es absolutamente imposible entender su funcionamiento y lo único que atinan a hacer es abrir la puerta del auto con la llave, dejar que la alarma suene y poner cara de yo no fui.

Si hace una encuesta entre sus allegados, verificará que no son pocos los que desconectaron definitivamente la alarma porque no conseguían dominarla. Es que la sensación que nos proporciona estar parados al lado del auto con la alarma sonando y sin poder hacer nada, con la impresión de que todo el planeta tiene la vista fija en nosotros, es más desagradable que el temor a que nos roben el auto.

Lo más interesante es que el funcionamiento básico y elemental de la combinación alarma/bloqueo no tiene fisuras: cuando se presiona el botón del control remoto, si las puertas están todas cerradas, se bloquea el auto y se prende la alarma. Si lo presiona nuevamente, se desbloquea el auto y se apaga la alarma. Lo desafío en que encuentre una situación en la que la alarma suena por otro motivo que un robo.

La alarma para autos es un avance incuestionable: cualquier ladrón preferirá siempre robar un auto sin alarma que uno con alarma, las estadísticas lo demuestran en forma contundente. Pero después de uno y otro intento fallido de mantener el control de la situación, muchos de sus

usuarios desisten de su protección y rezan para que el seguro pague los daños.

Mi programa de correo electrónico

Como muchos de los usuarios de este planeta, utilizo Outlook para recibir y enviar correo en la oficina. Mientras Outlook esté abierto, independientemente de la aplicación que se esté ejecutando, inclusive si esta es una presentación configurada para deshabilitar todo lo que pueda interrumpir, éste insiste en mostrar el siguiente mensaje 5 o 6 veces al día:

Como amante de la informática, entiendo todas y cada una de las explicaciones del mensaje, algo que asumo no le sucede a los no-informáticos. Sin embargo, en tres años de investigación, preguntándole a todo quien se cruza frente a mí, comenzando por el administrador, no he conseguido eliminar este molesto mensaje que me condena a "aceptar" esta ridícula situación como única opción.

El correo electrónico es una aplicación maravillosa, increíble, que ha cambiado la forma de relacionarse de las personas. Uno se pregunta: ¿Cómo era que uno trabajaba cuando no había correo electrónico? Pero inclusive toda esa magia no justifica que el software se pase importunándome con sus necesidades y requerimientos.

Deleite y pesar

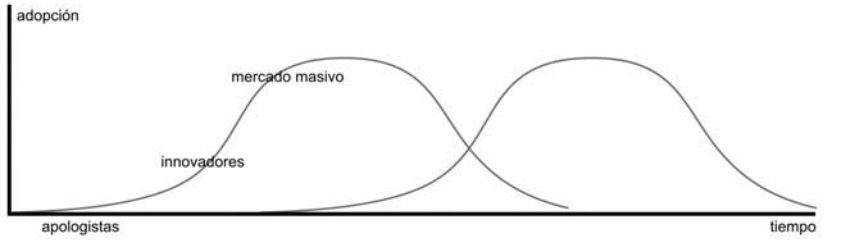
En el mismo paquete que las maravillas que los productos basados en software nos aportan, vienen los problemas y las insatisfacciones. Los programadores y las empresas que comercializan software dicen que es un mal necesario, y que el problema es que los usuarios no leen los manuales, no se capacitan y por tanto no saben utilizar los programas. Ven las causas en los usuarios, en los clientes, que deberían capacitarse

más, ser más cuidadosos y prestar más atención. A partir del convencimiento de que las causas son otras totalmente distintas y que no residen en los usuarios sino en el propio software, el objetivo de este libro es plantear un enfoque distinto y por tanto, un camino de solución radicalmente diferente.

¡Dame más!

La curva de adopción

Los productos tecnológicos tienen un proceso, una curva de adopción particular, que se repite una y otra vez. Es extremadamente difícil cuando nace una tecnología predecir si llegará a ser exitosa, pero si lo logra, será sin duda siguiendo algunos pasos repetidos, previsibles. La dificultad de pronosticar el futuro en tecnología no tiene que ver con la curva de adopción, sino con la selección de las tecnologías que serán las estrellas dentro de diez años.



La curva de adopción de la tecnología

La curva de adopción de una nueva tecnología comienza con un período de introducción, de larga duración, donde sólo los apologistas aceptan lidiar con las enormes dificultades que la tecnología implica para obtener sus menguados beneficios. Así, de Leonardo a los hermanos Wrigth, fueron numerosos los pilotos que arriesgaron su propia vida con tal de poder volar, aunque fuera unos segundos, con una máquina más pesada que el aire. Y desde que los hermanos Wrigth lo consiguieron, hasta que los usuarios más innovadores lo adoptaron, pasaron unos 20 largos años. El mercado de la aviación comercial de pasajeros se constituyó como masivo pasada la segunda guerra mundial, 50 años después de la primera

hazaña. Y también en la aviación sucedió algo propio de la curva de adopción de la tecnología: cuando el mercado se hizo masivo, el avance fue aplastante y vertiginoso. En la propia década del 50, en apenas una decena de años las travesías transoceánicas de pasajeros por mar se incorporaron a la historia. Lo que fue un anhelo durante cientos de años y creció como una realidad tangible pero embrionaria durante 50 años, en apenas 10 le quitó la supremacía a sus competidores.

Lo mismo vale para el automóvil, el teléfono, el fonógrafo y la luz eléctrica. Todas estas innovaciones tecnológicas siguen el modelo de adopción: período de introducción muy largo, más problemas que realidades, solo los apologistas se animan. Período de nacimiento comercial, avance lento, se acercan los innovadores. Explosión, todo lo abarca y todo lo puede: mercado masivo.

Apologistas del software[5]

La industria del software vive la curva de adopción de sus tecnologías y productos en un formato comprimido. Lo que en otras industrias llevó decenas o cientos de años, al software le llevó lustros, lo que otras tecnologías consiguieron en lustros, el software lo consigue en meses. La curva es la misma y los períodos de gestación también son extensos, pero cuando un producto de software supera la etapa de los innovadores y llega a la madurez, con una capacidad de reproducción infinita y un costo marginal casi nulo se dispara al estrellato a la velocidad de la luz. Alcanza con recordar que Internet tiene más de 30 años de antigüedad. Los primeros 20 no pasó de ser una preciosidad tecnológica recluida en el ámbito académico y militar. A partir de que se libera su uso comercial, entre 1991 y 1992, la explosión es tan fantástica que ningún otro medio de comunicación jamás alcanzó un desarrollo similar. Llegar a 60 millones de usuarios, lo que la radio consiguió en algo así como 40 años, el World Wide Web lo consiguió en cinco.[6]

La velocidad vertiginosa con la que literalmente explotan los productos basados en software es más visible aún si se toma nota de lo rápido que un producto sucede al siguiente. El primer Boeing 747 que despegó del

---

[5] *La división en apologistas y sobrevivientes fue introducida por Alan Cooper en el libro Presos de la Tecnología, aunque con un perfil un poco distinto del planteo utilizado en nuestro libro. Alan Cooper - Presos de la Tecnología, Cap. 2 Página 29 - Pearson Education, México 2001*

[6] *Fuente: Morgan Stanley Research, para IBM Consulting Group.*

suelo en un vuelo comercial, lo hizo el 22 de enero de 1970. Ningún equipo informático diseñado en esa época tiene cabida hoy en otro lugar que en un museo ¡El PC original, con 16k de memoria, disquetera de 160k, procesador de 4.88 MHtz es del 81!. El avión de pasajeros más moderno de la Boeing, el 777, voló por primera vez en un vuelo comercial en 1995. Windows 95 dejó de ser comercializado en 1997 y en el 2003 Microsoft anunció que dejó de soportar el vetusto sistema operativo. El tiempo que en otras industrias es considerado de introducción de un producto, inclusive en industrias de aplicación super-intensiva de la tecnología como la aeronáutica, en la industria del software alcanza para el ascenso, brillo y ocaso de cualquier producto.

El PC origina de 1981, en versión con 2 disketeras

Con semejante éxito detrás, los apologistas del software no han podido resistir a envalentonarse un poco más de la cuenta. Y no es para menos. Todos los números son aplastantemente positivos: sus productos crecieron en un abrir y cerrar de ojos, sus empresas son las más cotizadas del planeta, todas las otras industrias dependen de ellos. El planeta los mira con respeto y un poco de envidia. La carne es débil, y no es cuestión de encadenarse a la falsa humildad: los apologistas del mundo están gozando de su éxito.

Transformando al otro en sobreviviente

El problema cardinal y objetivo final de este libro, es entender qué ha sucedido en esta historia con el resto de los mortales no apologistas: los hombres y mujeres de a pie, aquellos recelosos de la tecnología, que no sienten ni necesidad ni placer en adoptarla cuando aún está en la cuna.

Si bien el éxito alcanzado hasta el momento es arrasador, las computadoras y los programas de software todavía no han conseguido cautivar el mercado realmente masivo, el público del mercado universal que va mas allá de innovadores y apologistas. En medio del disfrute, los apologistas están tan ocupados en gozar su éxito que han perdido la cuenta de que el otro que está enfrente ya no es un apologista, sino alguien no iniciado en los intríngulis informáticos, que no se enamora fácilmente de este feature (característica), aquella función o aquel otro parámetro de configuración.

Si empatía es la capacidad de entender los problemas desde el punto de vista de otra persona, de "ponerse en sus zapatos", podemos decir que anti-empatía es lo que reciben los usuarios comunes por parte de sus pares apologistas, lo que los ha llevado a transformarse en sobrevivientes de la tecnología. El término puede parecer un poco duro, pero es necesario sacudirnos ese halo de sabelotodos para llegar a comprender lo que sienten quienes usan las aplicaciones que los apologistas desarrollamos para ellos.

Según Po Bronson, en sus "Siete Hábitos de Gente Altamente Ingenierizada" uno de los atributos principales de los apologistas de la informática es ser generosos en su egoísmo.[7] Nada más justo. Los apologistas respondemos a las caras de incomprensión de nuestros compañeros de trabajo, de nuestros usuarios y clientes con más siglas, más jerigonza y más funciones, lo que a nosotros nos parece imprescindible para resolver cualquier problema y no lo que ellos desde su perspectiva necesitan y reclaman. Somos como aquel caudillo que gritaba "este pueblo será feliz, aunque tenga que matarlos a todos para conseguirlo".

---

[7] *Los siete hábitos de la gente altamente ingenierizada son:*
1. Son generosos en su egoísmo
2. La ceguera mejora su visión
3. No sólo morderán la mano de quien les da de comer, sino la suya propia
4. Harán el máximo esfuerzo para preservar la imagen de que les importa muy poco su imagen
5. Tratarán de arreglar insistentemente algo que no está descompuesto hasta que se descomponga
6. "No me equivoqué al responder, tú hiciste la pregunta equivocada."
7. Consideran la ausencia de crítica como un cumplido
Se cita en "Presos de la tecnología", no pudimos conseguir la cita original.

Para muestra basta un botón

Alguien puede argumentar que las afirmaciones son exageradas, que si bien es cierto que hay problemas, no se trata de problemas estructurales de industria sino de las rencillas normales que afectan a cualquier proyecto. Discrepo con esta visión. Los sobrevivientes se están tomando revancha de los desaires: están votando con los pies. No renuevan su software, siguen utilizando el mismo sistema operativo, la misma planilla de cálculo o el mismo procesador de texto que hace muchos años. Es lo mismo ser sobreviviente con la versión 95 o 98 que con la 2000 o la 2003: para un sobreviviente no vale gastar dinero para obtener el mismo resultado.

El ejemplo más contundente de esta revancha es el cambio en las formas de licenciamiento de Microsoft. El coloso del software no ha conseguido tentar a sus clientes desde Windows 95 a esta parte: a fines del año 2003, un 39%[8] de los PC corrían aún Windows 95 o 98, versiones del sistema operativo que tienen al menos 5 años de antigüedad. A pesar de que hay una nueva versión de sistema operativo y/o de suite de oficina cada 18 meses, el tiempo promedio de renovación en las empresas es de 3 a 5 años y en el hogar es más aún. La respuesta a la falta de ventas es una nueva forma de licenciar el software que castiga duramente a aquellos que deciden renovarlo en períodos mayores de 2 años. Microsoft, impotente ante la indiferencia de los sobrevivientes para con sus productos, aplica todo el peso al que su monopolio lo habilita.

Nuevos usuarios, nuevo enfoque

La concepción con que han creado y crean hoy los apologistas sus productos en la industria del software genera un circulo vicioso que separa aún más a los apologistas de los sobrevivientes. Es cada vez mayor la distancia entre los amantes de la tecnología informática y el gran torrente de personas que buscan, para cumplir sus objetivos, herramientas que no generen complicaciones, que sean efectivas y fáciles de usar y que

---

[8] *news.com: Users cling to old Microsoft operating systems http://news.com.com/2100-1016_3-5121458.html?tag=st_rn*

tengan un precio razonable por el valor que reciben, sin tener la más mínima preocupación por qué es lo que tienen adentro.

Las empresas de software intentan resolver el problema pisando aún más fuerte el acelerador, proponiendo mucho más de lo mismo. La apuesta suena razonable, por lo menos para los apologistas. Cuando fundaron sus empresas y llevaron al mercado sus primeros productos, se trataba de productos con pocas funciones y numerosas limitaciones. Los usuarios de la primera línea de combate, apologistas por definición, entre los que se cuentan por supuesto los propios programadores y los comentaristas de las revistas especializadas, ávidos de tecnología y dispuestos a cualquier costo con tal de conseguirla, transformaron esos primeros productos en un incipiente éxito.

El flujo de dinero por las ventas y la simpatía de los inversores ante el éxito, permitieron invertir en más ingeniería, mejorar la tecnología y con ello, agregarle nuevas funciones a los productos para capturar nuevas porciones de mercado hasta completar la primera fase de introducción de producto: los apologistas e innovadores adoptaron el producto a nivel planetario y se lo impusieron a una parte de los congéneres que tenían a su alrededor.

Ahora se enfrentan a la tarea de capturar a un nuevo segmento de mercado, el mercado masivo: llegar a cada hogar, a cada habitante, conseguir los niveles de penetración del televisor, del teléfono, del lavarropas o del microondas. Hoy las empresas se han transformado en grandes empresas, con una enorme capacidad de invertir en ingeniería y desarrollo para generar versiones de sus productos que contienen más tecnología y una cantidad gigantesca de funciones. También disponen de un enorme departamento de Marketing, con un generoso presupuesto, dispuesto a comunicar a los cuatro vientos las virtudes de las polifuncionales versiones de sus productos.

Sin embargo, esta estrategia empeora el problema. El nuevo segmento de mercado a cautivar y conquistar no está compuesto de apologistas. Se trata de clientes más pragmáticos, menos osados pero más sensatos en lo que a tecnología respecta. Esperan productos sencillos, fáciles de usar, sin complejidades y a un costo razonable por lo que ofrecen. A diferencia de los apologistas, no les importa lo que los productos tienen adentro sino cómo se comportan hacia afuera. Son los sobrevivientes y en un número

mucho mayor, el mercado de aquellos que aún no utilizan las computadoras.

Este nuevo segmento es decenas o tal vez cientos de veces mayor que el de los innovadores que ya adoptaron la tecnología. Para conquistarlo no se requiere de un nuevo producto sino de un nuevo enfoque, una nueva estrategia que comprenda los objetivos personales de los usuarios y los objetivos específicos que persiguen cuando desempeñan la tarea que exige utilizar una aplicación de software determinada.

Los objetivos personales

Todos tenemos objetivos personales, dentro de los cuales se cuenta como uno de los más importantes el de no sentirse ni tontos ni humillados. Apologistas y sobrevivientes intentan conseguir este objetivo por caminos distintos.

Para los apologistas, amantes de la tecnología, las dificultades que presenta la utilización de software son un desafío. Están dispuestos a leer voluminosos manuales, a gastar interminables horas, llegar hasta los límites de su capacidad para salir triunfantes frente a cualquier dificultad. Esta carrera de obstáculos no solo no amedrenta a los cruzados, sino que por el contrario valoriza el premio que significa el triunfo.

Los sobrevivientes no están dispuestos a los sacrificios que la utilización del software implica, no porque no tengan capacidad o no sean inteligentes, sino porque no disfrutan resolviendo configuraciones o estudiando manuales. Como ya aprendieron que sin ese esfuerzo no conseguirán los resultados, saben a priori que el software que los apologistas crearon para ellos ganará la partida y por ello lo sobreviven con indiferencia, con desidia que a veces se transforma en ira. Sobreviven las aplicaciones y las computadoras. Sobreviven la memoria y el disco duro. Sobreviven los drivers, las direcciones IP y los plugins. Sobreviven archivo, nuevo, abrir, cerrar, guardar, guardar como y siempre perder la información y los documentos. Y por sobre todas las cosas sobreviven a los mensajes de error que por cientos se abren informando de problemas tan graves como incomprensibles y sugiriendo soluciones tan complejas como inútiles.

La nueva estrategia para la creación de software debe comprender que para cautivar al mercado masivo, el software deber reducir enormemente su complejidad de uso. Y no se trata de agregar un sistema de ayuda más sofisticado o con descripciones más largas. Tampoco se resuelve con más wizards donde un usuario presiona cuatro o cinco veces "siguiente" para contestar a preguntas que no entiende, con respuestas que no leyó. "¿Desea una instalación típica, máxima, mínima o personalizada?". Señor fabricante: deseo escribir una carta o pasar un rato jugando carreras de autos. No sólo no se si preciso una instalación típica, máxima, mínima o personalizada, sino que en general no deseo ni necesito instalación alguna.

El software es humillante: pregunta cosas que no entiendo ni sé, mientras me oculta las que él sabe y que a mí me interesan, me interrumpe permanentemente con mensajes que no me importan y que a pesar de que dicen ser extremadamente importantes no deben serlo, porque los cierro sin leer y sigo trabajando. Me hace perder enormes cantidades de tiempo cuando se cae, se olvida permanentemente de lo que le digo, no confía en mí y me echa la culpa de sus errores. A los apologistas esta situación los excita, a los sobrevivientes los aburre.

Humillar y aburrir a los clientes está en las antípodas de la comprensión de sus objetivos personales.

Los objetivos específicos

Cuando una persona utiliza una aplicación de software para ejecutar una tarea, es imprescindible entender qué objetivos específicos persigue con esa tarea. Las personas manejan por la ciudad no porque les guste el desafío de encontrar un auto a contramano en cada esquina, o porque disfruten esperando que cambie la luz de un semáforo, sino por ejemplo, porque tienen que llegar a una hora determinada al lugar hacia el que viajan.

Los apologistas, que aman la tecnología en sí misma, sienten pasión por ella y ven en la tarea de lidiar con la tecnología un objetivo en sí mismo. Para ellos utilizar la última versión de un producto de software es una necesidad. Para los sobrevivientes es una imposición. Utilizar una

determinada tecnología no participa ni participará jamás de sus objetivos específicos. Unos disfrutan manejando, los otros no. En el mercado de los automóviles hay un segmento especial para los apologistas de la conducción de automóviles: el segmento de los autos deportivos particularmente rentable y lucrativo. Pero por cada auto deportivo que se comercializa, se venden cientos o miles de autos de todo tipo: utilitarios, económicos, de lujo, espaciosos, y muchos otros, orientados a cumplir mejor el objetivo específico de quien los utilice y no a dar placer a quien los maneja por el solo hecho de manejar.

El software actual, creado y desarrollado por Apologistas a su propia imagen y semejanza, pone el foco en la tarea misma, complementando esta visión del mundo con la idea de que cuantas más tareas sea capaz de resolver, más feliz va a estar quien lo use. El software del futuro deberá entender los objetivos específicos que quien lo utiliza desea alcanzar, focalizarse en ellos y quitar del camino todo aquello que pueda entorpecerlos.

La evolución de la fotografía

La fotografía es una tecnología ejemplar para ilustrar el problema que el software sufre hoy en día. En sus comienzos, la tecnología era tan rudimentaria y exigía tanta dedicación de fotógrafo y fotografiado, que compitió durante largas decenas de años con la pintura antes de conseguir imponerse como el medio idóneo para retratar personas, paisajes o momentos. Durante ese período es difícil decir que existiera un mercado vinculado a la fotografía. Los fotógrafos, además de sacar las fotografías, eran constructores de sus propias cámaras y desarrollaban la alquimia de sus películas, papeles y revelados.

Una vez que la tecnología se consolidó, comenzaron a aparecer cámaras fotográficas y servicios de revelado en el mercado. Los innovadores, aficionados a la tecnología de la fotografía, comenzaron a encontrar productos con una tecnología superior que les permitió mostrar sus habilidades y superar definitivamente la disputa con los retratos pintados. Nació el mercado de la fotografía, al que se sumaron los usuarios de primera línea, hasta constituir un grupo compacto, capacitado para sacar

fotografías de calidad a pesar de lo rudimentario de los equipos que utilizaban y las dificultades que ello implicaba.

Pero fue recién cuando la tecnología evolucionó lo suficiente para que sacar una fotografía fuera sólo "hacer clic" que el mercado abarcó las grandes masas de usuarios pragmáticos, menos dispuestos a adoptar y utilizar nuevas tecnologías hasta que éstas no prueben su utilidad y la justicia de su costo. Los equipos que utilizan no sacan las fotografías más sofisticadas, no permiten mayor control sobre los parámetros que determinan el enfoque, la cantidad de luz, el tiempo de exposición y la profundidad de campo, pero cumplen con su objetivo específico de guardar una imagen para recordar una persona, un momento, un paisaje y cumplen con el objetivo personal de no hacerlos sentir tontos: prácticamente todas las fotos que sacan salen bien, con una calidad más que aceptable, con apenas hacer clic. El mercado cobró entonces el tamaño gigantesco que hoy detenta, donde la penetración de la cámara fotográfica es prácticamente a nivel de cada hogar.

Es importante hacer notar que la profesión de fotógrafo no solo no desapareció sino que ha cobrado mayor importancia que en el pasado. Los equipos que utilizan son extremadamente sofisticados y no se parecen en nada a las que utilizan las personas comunes, excepto que tienen una lente y una película (dejemos para otra oportunidad la fotografía digital). Pero el foco de la industria fotográfica cambió, y se dirigió hacia esos usuarios comunes, llegando a generar el producto exacto que cumpliera sus objetivos tanto personales como específicos para entonces hacer estallar el mercado. Esa es la brecha que debe cubrir hoy la industria del software.

¿Soberbia?

Una de las 22 leyes inmutables del Marketing, de Al Ries y Jack Trout dice que "el éxito conduce a la soberbia, y la soberbia al fracaso". La industria del software, a pesar de su juventud, es indudablemente la más exitosa de los últimos tiempos. Ha crecido hasta niveles inimaginados, y penetra todas y cada una de las actividades humanas. Ya dijimos que hoy hay software en la batidora, en el reloj, en el sofá, en la estufa y en los zapatos. Pronto habrá en la ropa, en el inodoro, en la pared y en general

en todo lo que nos rodea sin excepciones, incluyendo la posibilidad de que esta lista incluya nuestro propio cuerpo.

¿Será que ese éxito se ha transformado en soberbia? La industria del software no consigue cautivar a sus nuevos clientes y usuarios, que reaccionan con indiferencia y desprecio ante lo que consideran un mal necesario. Ningún negocio tiene larga vida sin clientes satisfechos. Los clientes, por más monopolio y barreras de salida que protejan a una empresa, más tarde o más temprano pagarán los costos necesarios para librarse de ella. La soberbia enceguece y no deja ver el problema que está delante de nuestras narices.

Tal vez surjan nuevas empresas que capturen esta oportunidad y crezcan sobre las cenizas de las actuales. Tal vez algunas de las empresas actuales capten el problema y den un giro que les permita saltar hacia el mercado realmente masivo. Tal vez sea el software libre quien capture la oportunidad, cambiando completamente el mapa del mercado y de la industria. Tal vez sea un poco de cada cosa, y de algunas más que hoy nos cuesta imaginar. Pero lo que es seguro es que cuando el software y la informática den el salto definitivo hacia un mercado realmente masivo, será porque alguien habrá comprendido a cabalidad la brecha entre apologistas y sobrevivientes y habrá construido un puente firme para franquearla.

El homo sapiens sapiens: un animal explicativo

Forma parte de las habilidades de la especie humana la capacidad de obtener conclusiones a partir de información fragmentaria sobre la realidad. Componer la historia completa a partir de retazos es una habilidad extremadamente útil, que nos permite enfrentar con éxito la vida a pesar de ser el único animal que tropieza dos veces con la misma piedra. Así como no necesitamos que una película muestre todos y cada uno de los minutos de la vida de cada uno de los personajes para entender la historia, no necesitamos mirar permanentemente la torta en el horno para saber si está pronta, ni necesitamos ver a cada minuto a nuestro hijo en la cuna para saber si está pasando una buena noche. Alcanza con un

pequeño ruido, un olor particular, algunas señales, para componer toda la historia.

El corolario natural de dicha habilidad es la necesidad inapelable de explicar todo lo que no encaja en una historia. Ante cualquier hecho que se desvía del fluir normal de los acontecimientos, comenzamos inmediatamente a proponer o suponer posibles causas para tal anomalía, relacionando los elementos que tenemos a la vista de modo de concebir un nuevo flujo, una nueva historia, en la que las anomalías tengan explicación y en la que los descarriados vuelvan a ser parte del rebaño. Convertirnos en homo sapiens sapiens nos transformó en animales explicativos.

En el mundo físico, estas relaciones causa/efecto con las que explicamos el comportamiento de los objetos domésticos que nos rodean son en general directas y sencillas. Empujo y se mueve, arrastro y se raya, golpeo y se parte, presiono y se achata. En el mundo de la electricidad y los motores las relaciones son un poco más complejas, pero nuestra larga experiencia con ellos nos permite formarnos ideas útiles sobre el porqué de la interrupción en el comportamiento normal de una aparato o equipo.

Pero el mundo de las computadoras es completamente distinto: la relación causa/efecto del comportamiento de una computadora es inescrutable para alguien a quien no le confiaron los secretos de la programación. Y debido a ello, los animales explicativos fracasan estrepitosamente a la hora de entender por qué su computadora tomó tal o cual decisión.

Por ejemplo, si una persona se sienta frente a su máquina de escribir mecánica, una serie de varillas une la tecla "q" con el sello que estampa la letra "q" en el papel, por lo que al apretar la tecla, probablemente pueda observar con sus propios ojos como la serie de varillas se mueve llevando el sello hacia el papel. Con una máquina de escribir eléctrica, probablemente la serie de varillas no sea visible, se agreguen algunos motores, relés y switches, pero el comportamiento es aproximadamente idéntico, salvo que se necesita menos fuerza para apretar la letra "q" y la respuesta es más rápida. Pero al apretar la tecla "q" en el teclado de la computadora, las cosas cambian dramáticamente. Supongamos que estamos usando Windows 2000 y que está desplegado el escritorio, si hay alguna aplicación cuyo nombre empiece con q, entonces el foco irá hacia ella al presionar la tecla q, pero si no hay ninguna aplicación cuyo nombre

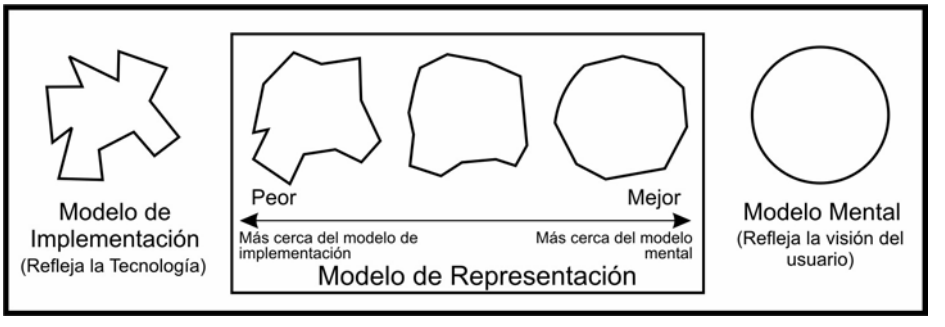
empiece con q, entonces sonará una campana, siempre y cuando no hubiéramos configurado otro sonido para ese tipo de error. Pero si estuviera abierta una aplicación, como Word, probablemente hubiera aparecido una q en la pantalla, siempre y cuando no estuviera presionada otra tecla a la vez y dependiendo de qué tecla fuere. Si es la tecla "Alt" y Word está en español, entonces sonará la campanilla, en otros idiomas habrá tal vez un comando asociado, si fuera la tecla "Ctrl" entonces se ejecutaría el comando alinear a la derecha, salvo que hubiéramos configurado Word para que ese comando esté asociado a otra combinación de teclas. Pero si a su vez el teclado está en español de Latinoamérica, entonces el "Alt" de la izquierda tendrá campanilla, pero el de la derecha producirá la @ algo que no sucede si el teclado está en Español, pero con la configuración Española o Tradicional. Ahora, esto siempre y cuando no tengamos ningún utilitario que capture la tecla y lo haga comportarse de otra manera. Pero siempre dentro de Word, si antes de la q apretamos "Alt" y luego la "c", soltamos ambos y presionamos "q", entonces aparecerá el Asistente Para Esquemas. Podríamos seguir infinitamente, explicando cómo dependiendo de la aplicación que tiene foco (un concepto propiamente informático), de las aplicaciones visibles e invisibles que están ejecutando en la computadora y de cómo está configurada cada una de ellas, después de ejecutar millones de instrucciones y decenas de programas, la computadora reacciona a la acción de presionar la letra "q".

El hombre común, habituado a formarse modelos en su cabeza que explican el comportamiento de los objetos que lo rodean, cuando se sienta ante su computadora se expone a un mundo incomprensible. Las pistas que el software le da no le alcanzan para producir modelo mental adecuado, y ello lo condena a generar explicaciones erróneas, a perder la orientación sobre qué hacer en cada momento y en general, a que su trabajo se transforme en una absurda carrera de obstáculos.

El modelo mental y el modelo de representación

En este contexto, la perplejidad del animal explicativo es superlativa. Las tripas de la computadora se comportan de una forma inescrutable con respecto a su comportamiento externo: la relación causa/efecto está

completamente quebrada. Esto propone a los diseñadores de las aplicaciones un desafío adicional al que tienen los diseñadores de los objetos y equipamientos del mundo físico: el de devolver a quién utiliza el software la habilidad de explicar los porqué del comportamiento de la computadora, no en función de lo que pasa dentro de la computadora sino de lo que se ve desde fuera. Los apologistas en general y los programadores en particular son incapaces de entender esta necesidad de los sobrevivientes: no solamente comprenden perfectamente las causas internas del comportamiento de su computadora, sino que las situaciones que no entienden son un desafío interesante que abre la oportunidad de entretenerse y conseguir una nueva victoria de la inteligencia humana sobre el silicio inerte.



El Modelo de Representación, según Alan Cooper

De este modo, hay un desafío para los diseñadores de la interacción de las aplicaciones, que es el de generar un modelo externo para el software al que Alan Cooper[9] bautizó como Modelo de Representación, que habilite al usuario de la aplicación a formarse un modelo mental que le permita explicar en sus propios términos el comportamiento del software. Para ello, es necesario esconder la problemática de la implementación. No importa si detrás de la pantalla hay un cluster de bases de datos relacionales o un millón de lineas de assembler: esto debe ser un problema exclusivamente interno de la implementación, el software debe mostrarse o representarse hacia afuera de una forma distinta a la de la

[9] *El trabajo de Alan Cooper se basa en la idea expuesta por Donald A. Noman en el libro "The design of Everyday Things", donde lo bautizó como "system image".*

Donald A. Norman: The design of everyday things, Página 16. Basic Books, Nueva York, USA -1988

Alan Cooper: About Face 2.0, Página 22. Wiley Publishing Inc. Indianapolis, USA - 2003

implementación que le permita al usuario crearse un modelo mental sólido y coherente que le restituya sus habilidades lógicas, permitiéndole cumplir a cabalidad su rol de animal explicativo.

La implementación y la Web

Para las empresas que desarrollan software, los ingresos provienen en una mayoría absolutamente aplastante de lo que venden a otras empresas. La cruda realidad es que el software que se usa en el hogar en nuestras latitudes es en general pirata, es decir, no genera ingresos para las empresas que lo desarrollan.

Los usuarios empresariales tienen la imperiosa necesidad de desarrollar sus tareas para no ser despedidos y no tienen gran probabilidad de ser considerados si dicen que la aplicación que lleva contabilidad es muy difícil de usar y que por esa causa no les fue posible terminar con su trabajo a tiempo. Si algún empleado osara hacer semejante planteo, sería inmediatamente aplastado por dos grupos: los ingenieros de sistemas con el gerente de sistemas a la cabeza, que lo tildarían de incapaz y perezoso, por no aprender a usar un sistema tan sencillo y eficiente por un lado, y por quienes autorizaron el gasto para implementar la aplicación de contabilidad por otro, que se van a negar sistemáticamente a reconocer que gastaron dinero en una aplicación con la que los empleados de la empresa están lejos de conseguir los resultados esperados. En las empresas, la facilidad de uso de las aplicaciones no está en el menú de temas a discutir, independientemente de las enormes pérdidas que genera una aplicación cuyos usuarios no consiguen entender, sumada al efecto de contar con empleados a los que su trabajo no hace felices.

Pero la Web es un mundo distinto: los navegantes no están adormecidos por el miedo a ser despedidos, ni por la arrogancia de un gerente apologista. En la Web, los navegantes dedican unos segundos, tal vez unos minutos a evaluar un sitio, que no es otra cosa que una aplicación preparada para ejecutar su Interface en un navegador, y si la evaluación es negativa, no lo visitan nunca más. No importa lo encumbrado de los estrategas de marketing, ni lo sofisticado de los programadores, ni lo avanzado de la tecnología, ni lo atractivo del diseño gráfico. La ley es una, y solamente una: si el sitio no me gusta me voy y no vuelvo nunca

más. De esta manera, a nadie le cuesta mucho reconocer que un porcentaje muy importante de los sitios Web no son visitados por nadie y por tanto no sirven absolutamente para nada.

La Web subvirtió el poder y ahora son los sobrevivientes los que definitivamente mandan: si usted quiere que su sitio tenga éxito, tendrá que morder el polvo, tragar la arrogancia y arrodillarse ante los visitantes potenciales del sitio, comprender sus objetivos y construir un sitio que conduzca de una forma eficiente y eficaz a esos visitantes potenciales hacia esos objetivos. Si no lo hace, podrá argumentar, explicar, comentar, e inclusive insultar, pero no tendrá visitantes y su proyecto será apenas otro fracaso más.

Un diseño contra sensus

Imagínese un docente en un curso de diseño Web. Después de un año entero de dar clases de http, html, flash, javascript, procesamiento de imágenes, Dreamweaver, rollover, menúes descolgables y todos los trucos y técnicas que un buen diseñador Web debe conocer, un alumno le presenta su trabajo final: un logo, una barra de edición de texto y un botón, centrados en el medio de una enorme página completamente en blanco y con convicción y una sonrisa afirma:

- Este es el diseño de la home page del sitio que en los próximos meses acaparará la atención de navegantes, programadores, comentaristas de la Web y por sobre todas las cosas, de los inversores. Con este sitio, vamos a ser la nueva estrella de Internet.

El resultado no es más que la crónica de una muerte anunciada: el docente irritado se tira de los pelos y se pregunta: ¿para qué tanto esfuerzo? ¿Qué hice para merecer esto? Y castiga a su pobre alumno, incapaz de asimilar

toda la sabiduría del diseño con la peor nota que le pueda aplicar. Cuando el alumno se va, como única forma de reivindicar su honor, le grita desde lejos al profesor: ¡le vamos a poner Finder, ya lo va a leer en los diarios!

No es una historia real, pero perfectamente podría serlo: antes de que Google fuera una realidad, cualquier alumno hubiera perdido un examen de diseño Web con su diseño absolutamente minimalista, espartano. Y probablemente hoy también lo perdería. Una recorrida de las técnicas, recomendaciones y principios de diseño Web que propone el 90% de la bibliografía sobre el tema apunta exactamente en la dirección contraria. La explicación tiene sus causas en que los mismos apologistas que antes diseñaban las aplicaciones ahora diseñan los sitios, que no son otra cosa que aplicaciones que se utilizan desde un navegador, y por tanto son incapaces de entender a los sobrevivientes que hay del otro lado de la pantalla. A esto se suma la irrupción de las huestes de los publicistas, con su fascinación por la TV y la dedicación obsesiva para llamar la atención.

La home page de Google, tal como luce en abril de 2004

Pero ahora es distinto, y en un espacio como la Web, donde los navegantes dedican apenas unos minutos a juzgar si deben utilizar o no

una aplicación, y si la respuesta es negativa no vuelven nunca más al sitio Web, la comprensión de los objetivos de los visitantes y el diseño de una interacción centrada en esos objetivos tiene premio.

Así es que Google se ha transformado en un suceso. Sus principios son sencillos y todo lo que se hace en el sitio tiene un alineamiento obsesivo con ellos: que las búsquedas sean rápidas, aplastantemente rápidas y que los resultados sean exactamente los que el navegante espera. Para que una página descargue rápido en una Internet que es hoy lenta, muy leeeeeeenta, hay que sacrificar todos los elementos de la página que no sean imprescindibles y por eso el diseño gráfico es absolutamente minimalista.

Lo más increíble es que con ese mismo principio, Yahoo! se transformó en el buscador más exitoso de la Web. Cuando pasó de un directorio, es decir de una lista ordenada de sitios, en un buscador, AltaVista ya exisitía y tenía una enorme cantidad de sitios registrados. Yahoo! seleccionó a Inktomi para proveerle la tecnología de búsqueda y su objetivo no fue ni ser el buscador que tenía más sitios registrados, ni el más completo, sino ser el más rápido y el de resultados más claros[10]. Y valga la redundancia, rápidamente desplazó a AltaVista del lugar preferido de las búsquedas. Sobre el tráfico que generaba el buscador construyó el portal, tal vez el más exitoso que hay en la Web. Para mejorar su capacidad de búsqueda, terminó su contrato con Inktomi y contrató como nuevo motor de búsqueda a una empresa llamada... ¡Google! y el arma mató al inventor: Google lo desplazó del lugar uno de las búsquedas aplicando su propia medicina. Hoy Yahoo! adquirió Inktomi con el objetivo de retomar un camino de tecnología de búsquedas independiente de Google, que le permita recuperar aunque más no sea parte del terreno perdido.

Cuando se eliminan las trabas y los usuarios pueden elegir libremente qué aplicaciones utilizar, los resultados son aparentemente sorprendentes, y muy distintos de los que los apologistas hubieran previsto: los usuarios votan sin dudarlo por las aplicaciones que de una forma más fácil y rápida les permiten conseguir sus objetivos.

---

[10] Como parte de esta apuesta, el nombre del sitio que originalmente se llamó "Jerry's Guide to the World Wide Web" fue cambiado por el de Yahoo! como acrónimo de "Yet Another Hierarchical Officious Oracle" (Apenas otro directorio jerárquico oficioso), ironizando con la idea de "somos apenas uno más"

Los usuarios usando

Si los apologistas que construyen las aplicaciones vieran a los usuarios usándolas, estoy convencido que dejarían de ser apologistas, y tendríamos la oportunidad de tener aplicaciones mejor diseñadas y por lo tanto mucho más útiles y económicas.

A pesar de que ver a los usuarios usando las aplicaciones es algo extremadamente fácil de alcanzar, prácticamente ninguna empresa de nuestra región lo hace. Conseguir 4 o 5 clientes que naveguen por el sitio Web corporativo, pedirles que cumplan 2 o 3 tareas sencillas cada uno, como por ejemplo registrarse, buscar un precio y bajar un manual de un producto, requiere apenas un día de trabajo. La recompensa será enorme: veremos por ejemplo que 4 de los 5 clientes no entienden la organización de los productos, en software, impresoras y consumibles. Ellos buscan encontrar aquel cartucho junto con las impresoras, y la palabra consumibles no les dice nada. O veremos que los 5 clientes tienen problemas con el buscador, que no entienden la opción de buscar en el Rubro, en la Categoría o en todo el Catálogo.

Mirar a los usuarios usar las aplicaciones es sencillamente fascinante. Después de semanas y en la mayoría de los casos meses de trabajar en una aplicación, los programadores y diseñadores adquieren un contexto con respecto a la misma, que se hace omnipresente y los anestesia frente a los problemas. Los usuarios que no poseen ese contexto, y que además no están vinculados a la organización y jerga internas de quienes desarrollan el sitio, desnudan rápidamente las incongruencias y problemas cuando se sientan frente al monitor y comienzan a sufrirlos en carne propia. Alcanza con salirse de la postura de Apologista y observarlos sin prejuicios para detectar estos problemas rápidamente, y con ello acercar enormemente las aplicaciones a sus usuarios potenciales.

Apologistas y sobrevivientes

Hasta aquí nos ha traído la historia de la informática, de la mágica combinación del hardware digital y el software: nos ha dotado de las

herramientas más poderosas jamás conocidas, a la vez que nos ha dividido.

Por un lado quedaron los Apologistas, que aman la tecnología en general y la informática en particular, en sí misma. Cuando tenían 10 años desarmaron el teléfono, hoy saben programar el video para que grabe un programa y entienden la diferencia entre un archivo y su acceso directo. Los apologistas son un grupo selecto, pequeño en comparación con el resto de los mortales, de personas que comprenden la tecnología y que la disfrutan.

Para ser un verdadero apologista no alcanza solo con el amor por la tecnología, es necesario además una actitud de proselitismo militante en su favor. No solo utilizar el último aparato, sino contarlo en cada oportunidad que se cruce, tratando en cada instancia de conseguir un nuevo adepto. Son verdaderos evangelistas del uso de la tecnología y la informática.

Los apologistas crearon el software y la industria informática a su imagen y semejanza. Crean los nuevos productos, las nuevas versiones, los nuevos features y disfrutan enormemente con ellos. Su última creación magistral son las versiones beta de los productos: versiones a las que acceden inclusive pagando, que les permiten usar un producto plagado de errores y problemas por el puro placer de conocerlo antes que el resto.

Por otro lado quedaron los sobrevivientes. Serlo no tiene ninguna connotación social, ni profesional. No se trata de individuos que han fracasado en ningún aspecto particular de la vida. Sencillamente, han fracasado en el intento de utilizar a plenitud y satisfacción los sistemas y aplicaciones que los Apologistas han creado para ellos. Es cierto que los utilizan, pero a diferencia del televisor, la estufa o la afeitadora, usarlos es una tarea frustrante, compleja, llena de incongruencias, de vías muertas, de tareas sin sentido repetidas una y otra vez.

Llegó la hora de cambiar las formas de relacionamiento entre unos y otros, en base a un cambio radical en el proceso de creación del software y los aplicativos. No será un proceso ni fácil ni instantáneo, pero transformará a los sobrevivientes del software en clientes satisfechos, haciendo posible que la computadora alcance los niveles de penetración del teléfono o del televisor, algo que significaría un cambio social y

económico de tal magnitud que sus derivaciones y consecuencias son difíciles de imaginar.

40. ¡Las computadoras me odian!

Capítulo 2

Pensar primero

Una industria inmadura

Comparada con disciplinas milenarias como la arquitectura, la ingeniería civil o la mecánica, la informática es una disciplina aún en pañales. Pero ninguna disciplina consiguió en 50 años invadir prácticamente todos los aspectos de la vida sobre la tierra. Además de las computadoras, hoy todo tiene microprocesadores y software dentro: el televisor, el auto, la heladera, el reloj y el termómetro. Lo que no tiene software aún, sin duda pronto lo tendrá. Y todo comienza a comportarse de esa forma incómoda, insensata, que nos hace sentir torpes y tontos.

Se trata de un grandote inmaduro, recién convertido en nuevo rico, que en su crecimiento arrollador toma prestado, compra o roba de aquí y de allá metodologías, técnicas y soluciones, sin darse tiempo a generar las propias en numerosas áreas, abriendo así flancos enteros a los problemas y el descontrol.

La situación se agrava porque el software es tal vez uno de los elementos más complicados con los que le ha tocado interactuar a la mente humana. Se trata de un intangible basado en un modelo lógico matemático estricto, cuya expresión exterior nada tiene que ver con su estructura interna y que no responde a las leyes físicas a las que los humanos estamos acostumbrados.

Esto crea problemas nunca antes afrontados, como por ejemplo la revisión del trabajo de sus subordinados. En cualquier área, un gerente es capaz de supervisar a sus subordinados, verificar que hacen sus tareas correctamente. Sin embargo, lleva más tiempo entender el código generado por un tercero que el que llevó escribirlo o directamente codificar el programa nuevamente. Eso hace que, en la práctica, nadie revise el código que se escribe para un proyecto. Se diga lo que se diga, se negocie lo que se negocie, sus programadores conservarán siempre la libertad de escribir el código de la forma que deseen, sin que nadie vaya jamás a controlarlos.

Pensar primero, hacer después

Si miramos la metodología de trabajo de las disciplinas ingenieriles y constructivas maduras que mencionábamos más arriba, veremos que todas comparten un patrón común: primero se piensa en el problema, se elabora una solución, generando al mismo tiempo las herramientas para que todos los participantes del proyecto conozcan y comprendan dicha solución, y recién después se comienza a construir la solución propiamente dicha.

Miremos por ejemplo la Arquitectura: primero se piensa la mejor solución para el desarrollo de, digamos, un centro comercial. Se analiza el tráfico interno y externo de personas, mercaderías y vehículos, cómo se integra a su contexto, cómo impacta el ambiente. La circulación, la iluminación, el acondicionamiento térmico y sonoro. Se analizan todos los ítems aplicando en algunos casos técnicas genéricas y en otros específicas. Junto con este trabajo, se van desarrollando descripciones de las soluciones en la forma de planos, maquetas, memorias descriptivas, animaciones, fotomontajes, dibujos, cronogramas y flujos de caja. Dependiendo de la envergadura del proyecto podrían probarse materiales, hacer cateos o construir pequeños pilotos. Todo esto **antes** de comenzar siquiera a hacer el pozo de la obra. La construcción de un puente o la fabricación de un nuevo automóvil sigue el mismo criterio.

Pero el desarrollo de software no sigue este patrón, a pesar de ser una típica actividad ingenieril. Apenas se decide que se va a producir un determinado producto o aplicación, se escribe una lista de las cosas que el software debe hacer y se comienza a programar. Sobre la marcha se va modelando la Interface, ensamblando las distintas partes y resolviendo los problemas que pudieran surgir. El resultado es como una de esas casas de balneario, hechas por un constructor idóneo, donde no hay siquiera una pared en escuadra, el agua se empoza en el techo, las llaves de la luz están atrás de las puertas y la puerta del baño da a la cocina. Con el agravante de que se consumió un 180% de presupuesto y se terminó en el doble del tiempo que el previsto.

Sin embargo este no es el único camino posible. Existe un camino que aprende de las otras disciplinas constructivas la metodología de pensar

primero, hacer después, o más específicamente diseñar primero, programar después.

La lista de features

El primer intento por poner orden al problema de pensar primero y programar después, es la creación de una lista de características que el software deberá tener, lo que en la industria se conoce como la lista de features. El departamento de sistemas invita a participar a todos los interesados y se crea una lista que enumera una por una las funciones que deben estar incluidas en el software. Así aparecerán en la lista elementos como manejo multimoneda, seguridad a nivel de registro, multiempresa, y sin duda, aparecerá como una característica ineludible "fácil de usar". La lista se escribe y se hace firmar por todos los participantes.

Los programadores ven la lista de features como una excelente herramienta de especificación porque se adecua exactamente a sus necesidades. Un feature se transformará a la hora de codificar la aplicación en una función o subprograma y por lo tanto una lista de features no es otra cosa, desde el punto de vista del programador, que el documento de análisis de más alto nivel.

Creer que una lista de features resuelve el problema del diseño, es como creer que una lista de ingredientes es lo mismo que una torta. No importa el nivel de detalle: se puede especificar el diámetro de los huevos y el grano máximo aceptable para el azúcar e igual estamos muy lejos de describir la torta. Con los mismos ingredientes se pueden realizar varias tortas distintas, todas excelentes, pero sobre todo se puede realizar una infinidad de sancochos impresentables.

En los pasos inmediatamente posteriores a la elaboración colectiva de la lista de features, los programadores asignan tiempos de desarrollo a cada uno de los features, y los ordenan en el tiempo asignado para desarrollar la aplicación. El resultado es que los features más aburridos de programar, los rutinarios, los que no presentan desafíos, tendrán sin duda abultados tiempos de desarrollo y serán enviados al final de la lista. Así el sistema de reportes y listados, la documentación y el sistema de ayuda, siempre ganan el honor de estar últimos en el cronograma.

Lo que piensa el programador	La realidad
Cuantas más funciones, mejor es el producto	Cuantas más funciones, mayor complejidad
Las funciones son siempre buenas, por lo tanto más es siempre mejor	Hay funciones buenas y malas, diseñar es elegir las adecuadas
Diseñar es listar y describir las funciones	No alcanza con listar y describir las funciones para Diseñar
Los usuarios adoran las funciones	Los usuarios adoran unas pocas funciones, ignoran la mayoría y sufren el resto.

Cuando el proyecto comienza a retrasarse y comienza a vislumbrarse que la torta va a devenir en sancocho, los participantes de la reunión original empiezan a discutir los cambios y las modificaciones. Allí los programadores mostrarán la lista de features y afirmarán con convicción: no hay derecho a reclamo, nuestro programa tiene todo lo que usted pidió, el problema es que usted no sabe lo que quiere. El resultado es una negociación en la que, a cambio de la introducción de modificaciones y reprogramación de algunos features ya terminados, se recorta el alcance: en este proceso mueren los ítems más costosos en tiempo del final de la lista, que no son otros que aquellos que los programadores pusieron allí por aburridos, o por considerarlos innecesarios o inadecuados. Dependiendo de su afinidad por el humor negro, estas negociaciones le resultarán divertidas o penosamente patéticas.

La lista de features no resuelve el problema. Se pueden ver batallas sórdidas y sumamente agresivas entre las distintas áreas del proyecto en torno a ella, y el resultado es el mismo que en cualquier guerra: desazón, bronca y mutilaciones. Su único efecto positivo es que hacen que todas las partes vean que el proceso de desarrollo de software está completamente fuera de control y que la forma en que se realiza no conduce a los resultados esperados.

Los usuarios no saben diseñar

El argumento más fuerte esgrimido por los programadores frente a los usuarios en base a la lista de features es "ustedes no saben lo que quieren". Y ésta es sin lugar a dudas una gran verdad.

La causa es sencilla: los usuarios no saben diseñar. Un individuo puede disfrutar más o menos de utilizar un auto, pero ni una situación ni la otra tienen relación alguna con su capacidad de diseñarlo, lo normal es que el individuo no tenga ni la más insensata idea de cómo se diseña el auto. Diseñar un auto es una tarea que requiere conocimientos y habilidades distintas que las de conducir. El propio Michael Schumacher no diseña los autos en los que corre: tiene un equipo de diseñadores que trabajan con él, que entienden sus requerimientos, los interpretan y los transforman en funciones mecánicas que se plasman luego en el auto con el que Schummy gana un título tras otro. Con el software debería pasar lo mismo: un equipo de diseñadores debería trabajar con los usuarios para definir cómo debería ser el software para satisfacer sus necesidades a plenitud. Y atención que el objetivo no es "qué debería hacer" sino "cómo debería ser". La diferencia es sutil pero sustancial. Lo primero es una lista de características o features. Lo segundo es una descripción comprensiva de cómo interactúa el software con el usuario para que éste último consiga sus objetivos. Y lo mismo sucede en todas las disciplinas: no es lo mismo saber si una torta es rica, que saber hacer una torta rica, ni saber que un vestido es bonito, que saber hacer un vestido bonito.

Los programadores no saben diseñar

Los objetivos de la programación y del diseño de la interacción van por carriles separados, por lo que no es viable que quien está programando sea capaz de diseñar. No sólo se trata de tareas disjuntas en el tiempo, dado que el diseño antecede a la programación, sino que las metodologías, objetivos y requisitos son completamente distintos.

El diseño de la interacción está focalizado en las necesidades de los seres humanos, la programación está centrada en las necesidades de las computadoras. Unas y otras son bien distintas. Mientras que los humanos somos imprecisos, nos equivocamos, somos ambiguos y cambiamos de opinión y gusto, las computadoras son exactas, precisas, inequívocas,

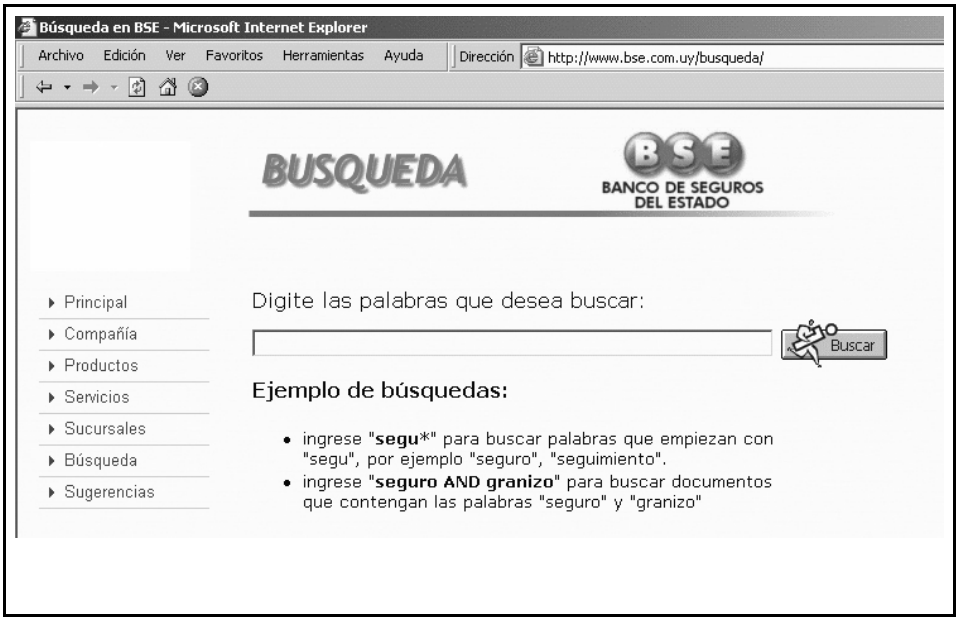
lógicas y repetitivas. Una persona no puede pensar a la vez en necesidades tan contrapuestas.

La programación es una tarea muy absorbente, que requiere de altos niveles de concentración y de una enorme capacidad de razonamiento lógico. Los programadores, por lo menos los buenos programadores, están entrenados para realizar estas tareas tan exigentes: para ello se necesita a veces pensar de una forma radicalmente distinta a la del resto de los mortales. Por ejemplo, si alguien pide a un programador que seleccione de la base de datos a los clientes que tienen 40 y 41 años, el programador deberá en realidad seleccionar a los clientes que tienen ó 40 ó 41 años, porque ningún cliente puede tener 40 y 41 años a la vez. El usuario recibirá el listado solicitado, sin enterarse jamás que el programador en realidad para cumplir con el requerimiento extrae de la base de datos algo totalmente distinto de lo que él le solicitó.

Del mismo modo, un buen programador pasará la mitad del tiempo destinado a la programación previendo el comportamiento del software en las situaciones límites o de borde, algo que los usuarios notarán solamente si dichas situaciones están mal resueltas y su software naufraga sin previo aviso. ¿Qué pasa cuando el nombre del archivo tiene más de 1024 caracteres? ¿Qué pasa si calculo el promedio de cero registros? ¿Qué pasa si configuro el ancho de página como un número negativo? La cantidad de situaciones es casi ilimitada y un programador debe garantizar que el software que crea pase airoso esas situaciones, no pierda el trabajo hecho hasta ahora y genere al menos algún camino para seguir adelante. Si la programación es de calidad, el usuario siquiera notará que está todo previsto, pero si la programación es deficiente el software se caerá a cada paso.

El diseñador, por el contrario, gastará su tiempo en que la interacción sea sofisticada y sutil en las tres o cuatro tareas centrales que el usuario utilizará el 80% de su tiempo, dedicando una porción muy pequeña a las situaciones de borde, que se presentan muy esporádicamente y que no determinan la satisfacción del software con la aplicación (salvo que estén mal resueltas a nivel de la programación, como dijimos más arriba). Junto a eso, el diseñador de la interacción se preocupará de que el software hable el lenguaje de quien lo va a usar y no el lenguaje de la computadora.

Un ejemplo de como los programadores diseñan mal está plasmado en las versiones avanzadas de los buscadores de los sitios de Internet. En la mayoría aplastante de los casos no se trata de otra cosa que una Interface elemental para introducir sentencias SQL directamente o para aplicar lógica booleana a las búsquedas. Tanto el lenguaje SQL como la lógica booleana son herramientas que los programadores adoran y los simples mortales desconocen, por lo que los usuarios comunes y corrientes, según muestran las pruebas de usabilidad de los sitios, jamás utilizan la mayoría de los buscadores avanzados.



¿Digito con comillas o sin comillas?

El Diseño de la Interacción

La solución al problema pasa por aplicar en la industria del software la metodología que las industrias constructivas maduras han acuñado a lo largo de cientos de años: primero se piensa y se especifica el producto a construir, luego se construye, y por último se decora y se refinan las terminaciones.

La disciplina que en la industria del software cumple el rol de pensar y especificar la solución del problema se llama Diseño de la Interacción[11]. El Diseño de la Interacción es el proceso de análisis y creación tanto de la interacción de los sistemas de computación con los seres humanos que los usan, como de la experiencia de éstos al utilizarlos. El Diseño de la Interacción se ocupará de estudiar la problemática de los usuarios potenciales, entender sus objetivos, planificar cómo pueden alcanzarlos mejor con una herramienta informática, y generará una especificación completa y detallada de cómo deberá comportarse dicha herramienta.

Diseño y Análisis de Sistemas

Los programadores argumentarán inmediatamente que esa es la tarea del Análisis de Sistemas, pero eso no es exacto. El Análisis de Sistemas parte de una descripción completa de la aplicación a programar y la desmenuza aplicando una estricta metodología técnica, hasta definir cada una de las partes que los programadores codificaran. Estas partes, constituidas básicamente por procedimientos y funciones, se unirán al final del proceso de desarrollo para dar vida a la herramienta de software o aplicación que se está construyendo. El error está en que la definición de si la solución será o no útil para el usuario no forma parte de la metodología del Análisis informático. Éste debe partir del problema completamente resuelto desde el punto de vista del usuario, ya que sus herramientas y etapas solamente toman en cuenta las necesidades y capacidades de los equipos que ejecutarán la aplicación y de los programadores que la construirán.

---

[11] La primera referencia que conozco al Diseño de la Interacción es del libro "Designing for the User with OVID", de Dave Robert, Dick Berry, Scott Isensee, John Mullaly, Macmillan Technical Publishers, USA - 1998. La evolución del concepto de Diseño de la Interacción tal como se utiliza en este libro fue introducida por Alan Cooper en el libro Presos de la tecnología, Pearson Education, Mexico - 2001

Tal como su nombre lo indica, cualquier metodología de *análisis* es una metodología de disección y reconstrucción. Para "dividir y conquistar", consigna emblema del Análisis de Sistemas, se debe partir del todo. Por el contrario, el diseño es una actividad de creación, no de construcción. No disecciona el problema al que se enfrenta, sino que genera una solución por la acumulación ordenada de información sobre el mismo. Para diseñar un sistema informático, es necesario recolectar información sobre el problema que intenta resolver y sobre esta base crear una solución. En otras palabras, el diseño parte de la hoja en blanco para crear la solución, el análisis parte de la hoja completa, para implementar la solución allí dibujada.

Vale la pena aclarar que el Análisis de Sistemas no deja por ello de ser una actividad creativa, que requiere creatividad desde el punto de vista de quien la desarrolla. Mientras que la creatividad que se aplica en el diseño está relacionada con la relación que tendrá el usuario con el software, en la etapa de análisis la creatividad se aplica a los problemas estrictamente técnicos.

Diseño de la interacción y diseño de la Interface

Sin intención de polemizar sobre qué es exactamente la Interface[12], preferimos utilizar la expresión Diseño de la Interacción, debido a que en general está asumido en la industria del software, que la Interface es algo que se construye al terminar de programar. Lo que se hace al terminar de programar es apenas decorar el programa, o decorar la Interface. Si el constructor dejó una columna en el medio de su dormitorio, o hizo la puerta del baño de su casa directamente hacia la calle, no hay decorador de interiores que pueda corregir estos problemas: su casa es y será un desastre, independientemente del color de las cortinas. Lo mismo pasa con las aplicaciones: no importa lo sofisticado de los gráficos, o lo llamativo de los íconos, si la aplicación no es más que el estricto modelo de implementación entregado en su versión más cruda al usuario, ni el más hábil y experiente de los diseñadores gráficos podrá salvarlo.

---

[12] *De hecho, ni la palabra "Interface" ni "Interfase" están incluidas en el diccionario de la Real Academia Española.*

El Diseño de la Interacción debe obligatoriamente realizarse antes de que se escriba la primera línea de código. Al momento de escribir la primera línea de código, prácticamente todas las aplicaciones que resuelven un problema dado tienen un costo y un tiempo de programación razonablemente equivalente. En cambio, cuando se escribe la primera línea de código los programadores ya han pasado por la etapa de análisis, la estructura lógica de funcionamiento de la aplicación estará determinada y por ende el modelo de datos también estará completamente especificado. Esos son los cimientos del edificio, y con los cimientos en su lugar, el edificio está definido: será ese y no otro. Construir cualquier edificio de 10 pisos con un nivel de calidad dado tiene un costo prácticamente idéntico por metro cuadrado. No importa el tamaño de los dormitorios o la ubicación de los sanitarios: todos los edificios son viables en el papel del arquitecto por el mismo costo. Pero una vez que se colocaron los cimientos en el pozo, la suerte del edificio está echada. Será ese y no otro, a pesar de que los transeúntes que pasan por al lado no ven más que un gran pozo en la tierra, con unas columnas de concreto que no les dicen nada. Los transeúntes son incapaces de ver o imaginar en esas columnas el edificio terminado. Con el software pasa lo mismo. Una vez que se escribe la primera línea de código, la suerte está echada, la aplicación completamente definida, a pesar de que los usuarios no puedan verla y crean que todavía son posibles cambios radicales. Recuerde, el código es como el hormigón: una vez que está escrito, la estructura endurece y los cambios se volverán inviables, a pesar de que usted no vea ni pueda imaginar qué es lo que se erigirá sobre esos cimientos.

Antecedentes del Diseño de la Interacción

Sin tener una relación causa-efecto lineal, hay numerosas contribuciones que pueden citarse como antecedente de la idea de Diseño de la Interacción. No se trata de que la metodología del Diseño de la Interacción tome directamente partes de cada una de ellas, sino que diversos autores han aportado a la práctica de la construcción de aplicaciones un diagnóstico común: los usuarios del mercado masivo no están satisfechos con las aplicaciones que utilizan, que los frustran y eso se está transformando en un techo para el salto que debe dar la industria hacia mercados masivos comparables a los de la televisión o los de la telefonía.

Las soluciones propuestas comparten también dos ideas rectoras: se necesita una metodología nueva, que especifique la aplicación a construir estrictamente desde el punto de vista del usuario y esa metodología precede el análisis de sistemas y la programación.

Dentro de estos antecedentes se puede incluir a uno de los libros que sientan las bases del diseño centrado en el usuario, a pesar de que no es un libro de diseño informático, sino de diseño en general, orientado al diseño de objetos de la vida diaria: El diseño de las cosas de todos los días[13], de Donald A. Norman. También figuran en esta lista dos trabajos de Jakob Nielsen: Ingeniería de la Usabilidad[14] y Usabilidad, Diseño de sitios Web[15]. El primero es el manual de cabecera de las tareas de análisis de la facilidad de uso del software, desde hace más de 10 años. El segundo, es un detallado análisis de qué es lo que funciona y es fácil de usar y qué es lo que no funciona y es difícil de usar en la Web, basado en la experiencia práctica de usuarios reales.

Hugh Beyer y Karen Holtzblatt proponen en Diseño Contextual[16] una metodología completa para el análisis de las necesidades de los usuarios potenciales de una aplicación, cómo especificarla y cómo a partir de ellas construir aplicaciones. Por su parte Alan Cooper, en sus libros About Face[17] y Presos de la Tecnología[18] hace un aporte invalorable que incluye la propuesta de la propia metodología del Diseño de la Interacción. About Face es un análisis pormenorizado y detallado de cómo perciben los usuarios cada uno de los elementos de una Interface basada en ventanas en un computador personal. La versión 2.0 incorpora capítulos específicos sobre Diseño de la Interacción. Por su parte, Presos de la Tecnología propone un análisis de la problemática de la industria del software hoy, y el Diseño de la Interacción como una solución a ella.

Por último, el libro de Steve Krug No Me Hagas Pensar[19] propone un enfoque basado en el sentido común para analizar cómo los usuarios

---

[13] Donald A. Norman: The Design of Everyday Things (También publicado como The Psychology of Everyday Things). Basic Books, Nueva York, USA -1988
[14] Jakob Nielsen: Usability Engineering. Academic Press, San Diego, USA - 1993
[15] Jakob Nielsen: Usabilidad, Diseño de Sitios Web. Pearson Education, Madrid - 2000
[16] Hugh Beyer y Karen Holtzblatt: Contextual Design. Academic Press, San Diego, USA - 1998
[17] Alan Cooper: About Face. Hungry Minds, New York, USA - 1995
[18] Alan Cooper: Presos de la Tecnología. Pearson Education, Mexico, 2001
[19] Steve Krug: Don't Make Me Think. New Riders, Indianápolis, USA, 2000

perciben los sitios Web y navegan en ellos, como punto de partida para construir los propios sitios.

Probablemente hayan decenas o cientos de propuestas similares a éstas, tal vez algunas más importantes y otras de mejor calidad, pero en mi opinión, estas obras brindan un panorama suficientemente amplio y claro de la necesidad de cambiar el enfoque y la metodología con la que se construye el software, tanto de aplicativos tradicionales como de sitios Web hoy en día.

Vale la pena diseñar

Para que la idea de "diseñar primero, programar después" sea aceptada, primero hay que contestar de una forma convincente la pregunta de si vale la pena, esto quiere decir si la relación costo/beneficio de la nueva propuesta supera a las prácticas comunes de hoy día. Y la respuesta es un contundente sí: "diseñar primero, programar después" es menos costoso, genera software de mejor calidad y usuarios más satisfechos, todo lo que se traduce en una relación costo/beneficio excelente comparada con las metodologías que se utilizan actualmente.

Software menos costoso

Si un cliente le contratara a Ud. el desarrollo de una aplicación, al día siguiente de la firma del contrato lo llamará por teléfono y le preguntará: "¿Y? ¿Ya comenzaron a programar?" Está absolutamente asumido a lo largo y ancho de la industria que cuanto antes se comienza a programar, más temprano se termina de programar.

Como vimos antes, la primera línea de código determina en forma prácticamente completa el resultado que obtendrá, independientemente de su capacidad para visualizarlo así en ese instante. Realizar cambios, modificaciones o rediseños sobre un desarrollo que está en marcha, es equivalente a pedirle a los albañiles que derriben una parte de la estructura del edificio para construirla nuevamente: algo que no sucede casi nunca en la vida real. Todos los que alguna vez programamos pasamos por la dura experiencia de mostrar el primer avance de nuestro trabajo a un cliente y que éste, en vez de felicitarnos, con cara de

perplejidad nos diga: "yo pensé que tal o cual cosa iba a ser distinta". La historia continúa siempre igual: nosotros argumentaremos que la lista de features indicaba lo que nosotros hicimos y el cliente que es obvio que debe ser de otra forma. El cambio es ahora extremadamente costoso comparado con el costo que hubiera representado hacerlo distinto desde el principio.

Pedro mira por primera vez la pantalla de ingreso de clientes y dice: nombre y apellido deberían ir separados. Zas, la batalla comenzó: Manuel el programador, mientras habla, comienza mentalmente a pensar en todos los lugares en los que es necesario hacer cambios para separar el nombre y el apellido. "No es lo que estaba en la especificación: allí decía que 'el alta de clientes deberá incluir entre los datos el nombre completo, la dirección, etc.' y eso es exactamente lo que hicimos". Pedro argumenta que "es OBVIO que el nombre completo es el nombre y el apellido separados" a lo que Manuel contestará que "no hay nada obvio, que el nombre completo es así y si te equivocaste no es mi problema". Sea cual sea el resultado, el final no es feliz. La diferencia en costos *antes* de comenzar de una opción frente a la otra era exactamente cero, sale absolutamente lo mismo utilizar un campo para nombre y apellido juntos que dos campos separados, siempre y cuando la decisión se tome en el momento adecuado, es decir, antes de escribir la primera línea de código.

En estas situaciones siempre se acepta hacer al menos algún cambio, lo que no sólo implica repetir la tarea, sino que hace nacer malherido al propio código. Los cambios se plasmarán en parches al código que, a pesar de que desarrollan la función solicitada por el cliente, aumentarán la probabilidad de introducir errores y dificultarán enormemente en el futuro el mantenimiento de la aplicación. La calidad del código es vital a la hora de determinar el costo de una aplicación durante su vida útil, porque determina el costo de agregar nueva funcionalidad, y la facilidad para detectar y corregir errores. En resumen: programamos dos veces lo mismo y empeoramos la calidad del código. Mal negocio si los hay.

A esto se agrega de que el equipo de diseño es mucho más reducido que el equipo de programadores: una relación de 5 a 1 es razonable, pudiendo llegar en algunos casos a 10 programadores por cada diseñador. Si el resultado fuera exactamente el mismo, sería más barato siempre diseñar

primero, ya que esto reduciría la cantidad de horas/hombre total dedicadas a la creación de la aplicación.

Software de mejor calidad

Tal como expresa el párrafo anterior, no diseñar antes de programar empeora la calidad del resultado final. Y esto tiene múltiples causas.

En primer lugar, diseñar reduce la cantidad de cambios a realizar sobre la marcha, porque la solución del problema es más adecuada a todos los involucrados, incluyendo a los usuarios en primer lugar y porque se generaron herramientas que le permiten a cada uno de los participantes visualizar esta solución antes de que comience a construirse. Siempre surgirán cambios a partir del cambio de la propia realidad en la que el proyecto se desarrolla y de las necesidades comerciales de la empresa. También se detectarán errores en el diseño y en la programación. Pero desaparecerán los numerosos cambios que tienen su causa en malentendidos entre las partes y errores de interpretación.

En segundo lugar, los programadores podrán concentrarse exclusivamente en su tarea: programar. No tendrán que gastar ni un instante de su tiempo en diseñar, o en suponer que es lo que van a decir cuando vean el programa terminado. Un programador amigo llena sus programas de parámetros y opciones ocultas, que le permiten que se comporte de múltiples maneras, para poder realizar más rápido los cambios que los usuarios aún no le pidieron, pero que él está seguro que le van a pedir. Cuando le pregunto por qué gasta tiempo y esfuerzo en esa tarea, en mi opinión absolutamente prescindible, contesta con convicción: "yo conozco a mi ganado". Se programa más y el software es más complejo porque se asume a priori que no conocemos que es lo que los usuarios realmente necesitan. Sería sin duda más económico entender con precisión qué es lo que los usuarios tienen como objetivo, diseñar una aplicación capaz de ayudarlos a alcanzarlos, y que luego los programadores la desarrollen concentrados exclusivamente en esa aplicación y no en el abanico de sus posibles variaciones.

Por último, el software con menos parches, con un modelo de datos más refinado, con menos opciones ocultas, desarrollado por programadores

focalizados 100% en la tarea, sin duda será más compacto y eficiente, y por lo tanto tendrá menores requerimientos de equipamiento.

Usuarios más satisfechos

El proceso de Diseño de la Interacción aplicado antes de la programación, generará sin dudas usuarios más satisfechos: porque el resultado será más adecuado a sus objetivos y expectativas, y porque el propio proceso de desarrollo será menos estresante para él en el caso que participe o para sus representantes en la empresa (marketing, negocios, etc.) en el caso que el usuario no participe directamente.

La satisfacción de un cliente no es el resultado abstracto de un producto "bueno", sino la relación entre su percepción del producto que recibe y las expectativas que tenía antes de recibirlo. Un cliente quedará absolutamente insatisfecho si su nuevo Porsche 911 no alcanza los 250 kilómetros por hora, pero no hubiera sido así si se tratara de un Fiat Uno. Los autos son absolutamente distintos, pero también las expectativas de quienes los usan son absolutamente distintas. Si ambos autos colman las expectativas de quienes los compran, sin duda estos quedarán igualmente satisfechos.

El proceso de Diseño de la Interacción permite a la vez conocer los objetivos y expectativas de los usuarios y trabajar sobre ellas, a través de herramientas que le ayudan a visualizar cómo será el producto acabado desde su punto de vista. Esto hace que el usuario pueda adaptar sus expectativas o modificar el producto antes de recibirlo, de modo que cuando lo reciba, las malas noticias sean mínimas. Cuando el cliente ve el avance del programa y no lo satisface, el problema no necesariamente es que lo que le muestran no sea bueno, o no esté bien hecho, sino que sus expectativas al respecto eran otras. El Diseño de la Interacción cierra esta brecha, minimizando el peligro de generar insatisfacciones y por ende, consiguiendo que los usuarios estén más satisfechos. Y usuarios satisfechos son sinónimo de buenos negocios.

Capítulo 3

Los objetivos del Diseño

Como complemento a la definición del Diseño de la Interacción, es interesante analizar sus objetivos. De esta manera se puede cumplir con dos propósitos en forma simultánea: complementar la propia definición y generar un mecanismo para determinar si el Diseño de la Interacción, una vez finalizado, es realmente bueno.

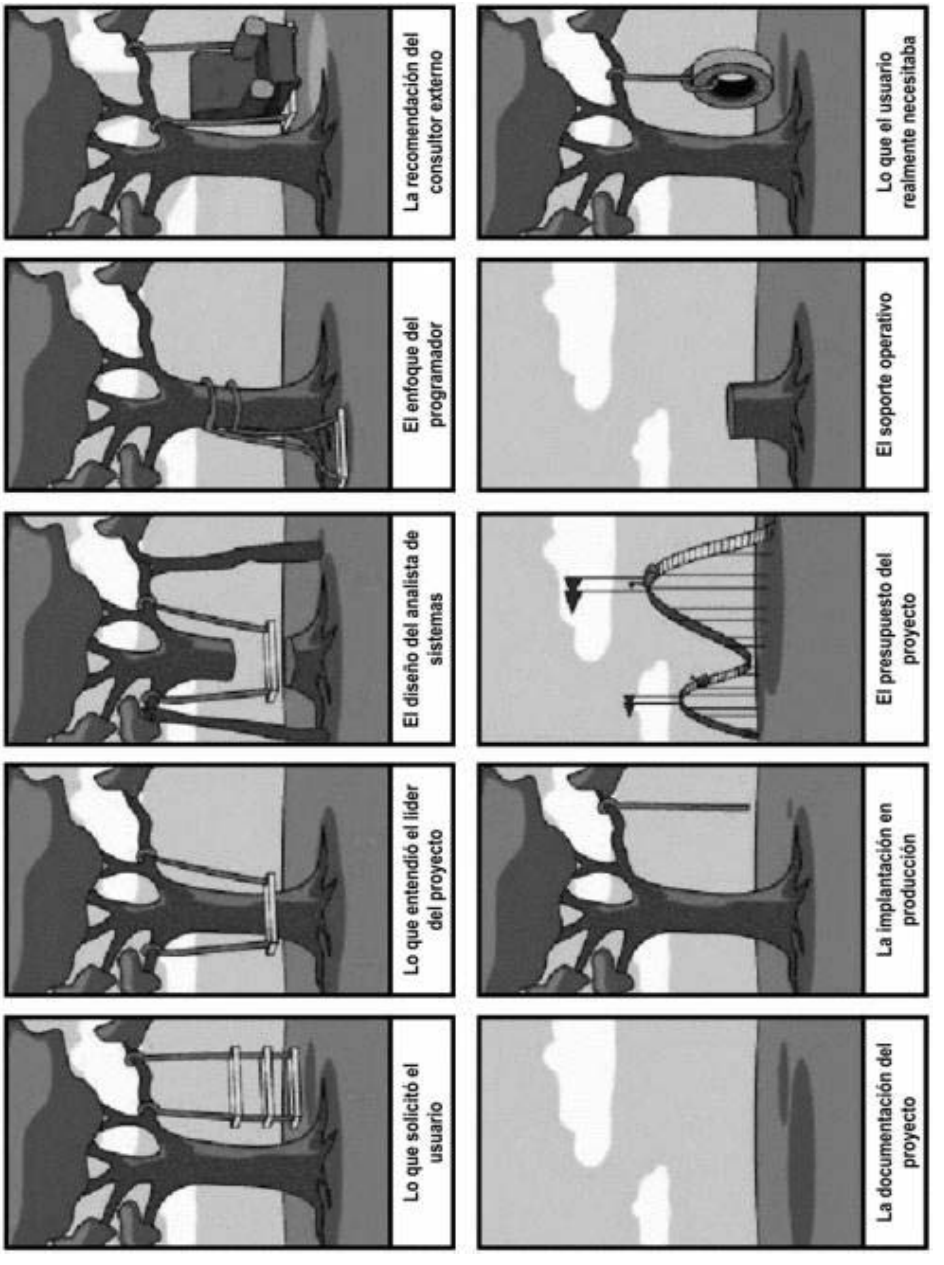
El conocimiento a priori de objetivos genéricos para el diseño de sitios y aplicaciones, más allá del objetivo específico y trivial de diseñar dicha aplicación o sitio, nos permite saber si vamos o no por buen camino. Preguntando por cada objetivo en concreto podemos determinar con seguridad si estamos o no diseñando en forma acertada.

En este marco podemos determinar cinco objetivos para el Diseño de la Interacción:

- ✓ Definir el producto final
- ✓ Acotar y minimizar los costos
- ✓ Poner foco en el usuario
- ✓ Sacar la presión que el diseño implica para el equipo de programación
- ✓ Hacer creíbles y "cumplibles" los cronogramas

Definir el producto final

Circula por Internet una dibujo que ironiza acerca de lo que espera cada uno de los participantes de un proyecto de software. Sería muy difícil definir el problema con más acierto que en ese dibujo. La diferencia de enfoque y de expectativas es radical entre los distintos participantes y en muchos de los casos incompatible. Definir el producto final antes de empezar a programar y alinear las expectativas de todos los participantes tras esa definición es la única medida profiláctica para este problema, algo que resulta tan obvio en la teoría como difícil de encontrar en la práctica. Es, sin duda, el objetivo más importante que debe perseguir el Diseño de la Interacción.



Lo que espera cada participante de un proyecto de software.

Como ya dijimos más arriba, otras disciplinas más maduras que la informática, como por ejemplo la arquitectura y la ingeniería civil, trabajan el tiempo necesario para cumplir con este objetivo antes de comenzar con las tareas de campo. Todas las herramientas son válidas: maquetas, planos, memorias descriptivas, animaciones, cronogramas, presupuestos, flujos de caja y un bagaje enorme de técnicas permiten dar visiones completas del producto terminado a los clientes, a los inversores, a los constructores, a los albañiles, subcontratistas y todos los participantes de la obra sin necesidad de poner el primer ladrillo. El ejemplo es válido para trasladarlo a la industria informática. En ésta, apenas se conocen los primeros esbozos del sistema, se comienza a programar, con la psicodelia de que programar y diseñar en paralelo ahorra tiempo y dinero. Muchas veces inclusive, es considerado entre los programadores una viveza, un rasgo de picardía, obviar el trabajo previo de diseño y documentación.

Si pensáramos en sentido inverso y aplicáramos los usos y costumbres informáticos a la ingeniería civil, entonces después de una reunión de dos o tres horas, donde le indican al gerente de proyecto que se va a construir un edificio con 30 apartamentos de dos dormitorios, cocina, living comedor y otras facilidades, de $75m^2$ cada uno, con 15 cocheras y de categoría media, éste se dirige al terreno y sentencia: “muchachos, vayan haciendo un pozo acá de más o menos 5 metros como la vez pasada, creo que con eso va a alcanzar, así adelantamos mientras yo hago los planos. Si después hay alguna diferencia vemos”.

Definir correctamente el producto final antes de comenzar a programar tiene dos consecuencias altamente positivas: permite trazar el camino y permite manejar las expectativas de los clientes.

Trazar el camino

Recién al definir el producto final, es posible trazar el camino, desarrollar un plan del proyecto. Sin un destino, sin saber a dónde nos dirigimos, no se puede marcar un camino. Tal como dice Lewis Carroll en Alicia en el País de las maravillas: “todos los caminos son buenos para quién no sabe a dónde va”. Parece haber sido escrito especialmente para esta ocasión.

Dado que el desarrollo de código, que se basa en la definición previa de las estructuras y modelos de datos, congela el proyecto, la supuesta posibilidad de que sobre la marcha se pueden ir modificando las características de los sistemas es apenas una ilusión óptica que desconoce la realidad de que el diseño de software comprende decisiones sobre un cúmulo de equilibrios y balances, y que para escribir la primera linea de código es necesario haber tomado todas y cada una de esas decisiones. Esto se puede hacer en forma explícita, separando el diseño de la programación o en forma implícita, asumiendo que “programamos un poco y después vemos”.

Un corolario interesante de la falta de definición del producto final, es que sin ésta es imposible saber cuándo el producto está terminado. Como al partir no se conoce el punto de llegada, el desarrollo del proyecto es más parecido a una expedición de descubrimiento que a una tarea ingenieril. Y la mayoría de las veces pasa lo que le pasó a Colón: cuando creen que llegaron están exactamente en la antípoda de lo que buscaban. Tristemente muchas veces, como también le pasó a Colón, los héroes mueren sin percatarse jamás de este hecho.

Es frecuente que después de varias postergaciones en las fechas de finalización de un proyecto de desarrollo de una aplicación, se defina arbitrariamente una fecha en la que el producto saldrá sí o sí, y para ello se intensifica la negociación de qué features dejar y cuáles recortar, nace la versión 2.0, a la que se adjudican automáticamente todas aquellas funciones que el tiempo no alcanza para programar, y se recorta salvajemente todo aquello que no sea absolutamente imprescindible para transformar las funciones que ya están implementadas en una versión "estable".

Un programador se defendía a diestra y siniestra: "pero salimos en fecha ¿eh?, te acordás: los listados no daban nada, las facturas estaban mal, modificábamos todo a mano, pero ¡ahí estábamos! Nos pasamos seis meses hasta que los números empezaron a dar algo". Nada más paradigmático: él estaba convencido que terminaron en fecha cuando en realidad la aplicación se puso en el aire tal como estaba en el momento en que alguien con suficiente poder les dijo basta. Su ignorancia lo hacía enormemente feliz. Todavía faltaban 6 meses de trabajo, pero eso es un detalle menor, del que solamente se puede preocupar un malintencionado.

Sin definir completamente el producto final, es imposible trazar el camino. No es un problema de voluntad: es materialmente imposible trazar un camino que no va a ninguna parte. Nos subimos al ómnibus de la programación y salimos más o menos hacia allá: ¡Albricias! ¡Estamos viajando a toda velocidad! Cuando se termine el combustible no habrá lugar a duda alguna, estaremos exactamente en el lugar al que merecemos llegar.

Manejar las expectativas de los clientes

Todos nosotros queremos que nuestros clientes queden contentos, satisfechos con el producto que les entregamos, sean estos clientes internos, externos o ambos.

No es posible tener clientes satisfechos, más allá de algún golpe de suerte, si no somos capaces de manejar las expectativas de dichos clientes. La idea vulgar de que un cliente queda satisfecho cuando el producto es "bueno" es esencialmente falsa. Un cliente queda satisfecho cuando su percepción sobre el producto que se le entrega es igual o mejor que las expectativas que tenía.

La acumulación de features en una lista no permite manejar las expectativas de cliente alguno. Siguiendo con el ejemplo de la Arquitectura, la definición de features de una vivienda dice bastante, pero no alcanza para generar expectativas: "Apartamento de 3 dormitorios, 2 baños, 110 m^2, vista al frente, piso de cerámica importada, calefacción central" abarca una cantidad enorme de apartamentos completamente distintos. Por eso, entre otras cosas, la arquitectura brinda herramientas adicionales para que sus clientes se formen una idea mucho más acabada del producto que recibirán. En el diseño de sitios y aplicaciones ocurre exactamente lo mismo. Mientras que una "excelente búsqueda" para uno puede significar una generosa dotación de operadores lógicos, de cercanía, y una tabla de sinónimos, para otro puede significar velocidad para encontrar la palabra buscada. Para un tercero puede implicar capacidad de distinguir entre los idiomas de diferentes documentos y para un cuarto capacidad de búsqueda no solo en textos, sino también en imágenes, sonidos, videos y otros formatos de archivos. Sin diseño, es decir, sin un correcto manejo de las expectativas de los clientes, sólo la

suerte puede salvarnos de que se defrauden cuando vean nuestra búsqueda terminada.

El diseño debe evitar el momento fatídico en el que mostramos el trabajo por primera vez a un cliente interno o externo, ese programa en el que invertimos tantas horas, del que estamos casi enamorados y el cliente con cara de perplejidad dice: yo creí que iba a ser distinto. Todos los que programamos pasamos muchas veces por esta desagradable situación, cuyo único remedio está en manejar las expectativas del cliente, para que cuando vea el producto terminado, éste colme dichas expectativas sin sorpresas desagradables para ambas partes.

Acotar y minimizar los costos

Las leyes de Murphy rezan que un proyecto informático consume el 90% de los recursos en el primer 90% del trabajo y otro 90% de recursos en el 10% restante. Un corolario afirma que no vale presupuestar 180% para evitar el problema, ya que el resultado será consumir un 360% del presupuesto. Palabra del señor Murphy.

La lista de desvíos en los presupuestos y cronogramas de proyectos informáticos es extensísima y abarca desde pequeños proyectos hasta proyectos enormes, míticos, como el retraso de más de 2 años en el lanzamiento de la versión 4 de Windows, que terminó saliendo al mercado bajo el nombre de Windows 95.

Las dificultades en determinar costos reales para el desarrollo de los sistemas, derivan principalmente de la falta de diseño. La carencia de diseño determina que no se pueda trazar un camino, un cronograma confiable, y sin éste no hay presupuesto. La situación se agrava con los problemas que generan cuando los clientes (internos y externos) se ven defraudados al ver los resultados del trabajo a medida que van apareciendo. Se imponen así cambios que se suman a las definiciones tardías del sistema, que van contra las definiciones tomadas implícitamente al empezar a programar. Las pulseadas y roces que supone este proceso de interacción y sucesivas aproximaciones, genera en algunos casos verdaderas batallas campales en la interna empresarial. Cuando los costos comienzan a dispararse y el departamento de sistemas

comienza con los misiles tierra-email al resto de la empresa por las indefiniciones, el resto de la empresa se esconde detrás de barricadas de firewalls y replica con cañones de 55 milibytes dirigidas a las 10 cabezas más buscadas de sistemas por los retrasos. El daño ya está hecho.

Dado que el presupuesto se construye con la base de la lista de features, sobre la que se determinan las horas hombre que lleva cada feature y el precio de dichas horas, los programadores determinan unilateralmente y en forma implícita, a falta de mejor definición, qué partes son más importantes y cuáles no lo son. Cuando los costos se disparan, comienza la hora de los recortes y allí se eliminan sin piedad los features más costosos. Este proceso perverso, lejos de ayudar, solo agrega leña al fuego, perjudicando enormemente la calidad del producto final y avivando la discusión de las distintas áreas que promovieron en su momento la inclusión de los features que ahora se eliminan.

Es muy importante reiterar que cuando todavía no se escribió la primera línea de código, prácticamente todas las soluciones a un problema tienen un costo similar, que se enmarcan todas dentro de un rango de costos. La dificultad no está en estimar el costo de implementación de una de esas soluciones, sino en manejar todos los factores que generaran desviaciones en los costos al construirla. El costo no se dispara porque fue mal estimado, sino porque se estimó el costo de una solución muy distinta a la que se terminará implementado.

La medicina para este mal es entonces preventiva, y no curativa: hay que hacer coincidir la aplicación que se construye con la que quienes participan del proyecto creen que se está construyendo, lo que tiene como exigencia previa que los que participan del proyecto crean todos al unísono que están construyendo la misma aplicación. El diseño de la interacción permite definir adecuadamente cada una de las áreas de interacción de la aplicación con los futuros clientes y usuarios, dar a estos una visión completa de lo que obtendrán cuando el desarrollo esté concluido y en base a esto priorizar las áreas funcionales del sistema. También el diseño de la interacción permite desarrollar el camino desde la situación actual hacia el destino conocido, determinar el trabajo de cada parte, no sólo del área de sistemas, sino de todas las áreas involucradas y en base a esto desarrollar un presupuesto óptimo, y además realista.

Y si no cree todo esto, pero tiene la piel curtida de ir una y otra vez a confesarle a su jefe que todavía falta mucho pero que en la alcancía queda poco: ¿por qué no prueba hacer las cosas distinto?

Poner foco en el usuario

Salvo en la proclama, el usuario no está representado en el proceso de desarrollo de sitios y sistemas. En definitiva el usuario es a los sistemas, como el ciudadano es a la política: mucho en el estrado y poco en la cocina, similitud que nos hace perder un poco las esperanzas, pese a lo cual insistiremos hasta el final.

Usuario y programador participan del proyecto con objetivos distintos y cuotas de poder distintas. Tal como electores y elegidos. Y para que los usuarios puedan captar una cuota de poder, es necesario plasmar sus necesidades en el proyecto a través del diseño antes que los programadores plasmen las suyas. Diseñar es marcar las fronteras que las necesidades de los usuarios imponen a los programadores. Y si en política es dudoso el poder de los electores para controlar la ejecución de los programas preelectorales de los elegidos[20], en el ambiente del desarrollo de sitios y sistemas esto es perfectamente viable.

Los folletos gritan que la Interface es intuitiva y el sistema fácil de usar. Creaciones enérgicas y convincentes como “Interface humana”, “interacción natural", "relación hombre-máquina" y similares plagan las cajas, las descripciones y listas de features de los distintos productos. Sin embargo, los supervivientes, usuarios de a pie, sienten que las computadoras no los comprenden, no los toman en cuenta, se mofan de ellos, y más tarde o más temprano responden con indiferencia, desidia y hasta odio hacia ellas.

La disociación entre intenciones y realidad tiene en las carencias de diseño para la creación de sitos y aplicaciones su causa más profunda. La intención de poner foco real en el usuario es sincera, pero ante la presión de los plazos, la presión de los costos y la presión de la competencia, los desarrollos terminan recorriendo caminos que los alejan de las buenas

---

[20] *Parece que el diablo ha metido su cola: lo que los políticos presentan a sus electores no es otra cosa que programas. Y si los que hacen programas son programadores, entonces ¡los políticos son programadores!*

intenciones y los acercan a la realidad: faltan las definiciones, el tiempo se termina y cada vez el desarrollo se parece más a un "vale todo". Prueba contundente de esta realidad es la documentación: en general, cuando está disponible, es pésima. Apenas colma la línea del contrato que decía: "documentación para el usuario", con cosas de este tipo:

> ***Cantidad de Buffers***: el parámetro Cantidad de Buffers del sistema le permitirá configurar la cantidad de buffers que desea para el funcionamiento del sistema.
>
> *Información detallada*: modificando el valor asignado al parámetro cantidad de buffers usted podrá adaptar el comportamiento del sistema a sus requerimientos y preferencias. Es muy importante que asigne el valor adecuado a este parámetro, de modo de aprovechar al máximo el rendimiento de su equipo en función de sus necesidades.

Puede ser que dude que los usuarios odian la computadora, pero no creo que vaya a discutir que odian la documentación.

Determinar con la mayor claridad posible el producto final, y con esta definición determinar el camino a seguir, previene de las desviaciones que las presiones imponen y acerca los sistemas al usuario. Siempre es posible torcer el camino hacia un derrotero erróneo, alcanza apenas con tomar una o dos decisiones equivocadas durante el proceso de desarrollo. Mientras que el diseño no garantiza que el usuario se va a sentir a gusto con el sitio o sistema, que va a alcanzar sus objetivos, la inversa sí es garantizada: la ausencia de diseño de la interacción genera sitios y sistemas hechos a la medida de las computadoras, los programadores, las presiones y las pulseadas entre los distintos departamentos.

Sacar la presión que el diseño implica para el equipo de programación

Quien alguna vez trabajó como programador conoce la desesperación de tener que desarrollar un sistema con un esquemita hecho en una servilleta a la hora del almuerzo. Las primeras armas como programador nos hacen creer que eso es una oportunidad para brillar, la vida nos enseña luego

que es una carta blanca para criticar impunemente nuestro trabajo, incluir funcionalidad que no estaba prevista, obligarnos a hacer cambios enormes que de otro modo no hubieran sido necesarios.

Los emails, apuntes de las reuniones y todas las armas que tengamos serán insuficientes: tal o cual función es obvio que debió ser incluida y a pesar de que en su momento nadie oyó nuestras preguntas, nadie puso interés en las reuniones, nadie llenó los formularios, nosotros tenemos que programar. Los programadores asumen así, por omisión del resto del equipo, la pesada tarea de diseñar: una tarea para la que no están capacitados y que se les obliga a desempeñar en las peores condiciones.

El diseño tiene teoría, objetivos, fundamentos, técnicas y metodología distintas que la programación. Es más, en muchos casos, los caminos son opuestos. Por ejemplo, mientras que la programación supone el análisis permanente y exhaustivo de la situaciones de borde, el diseño de la interacción prácticamente las ignora. Diseñar primero y programar después debería ser casi casi el sueño de los programadores: trabajar sin usuarios.

El diseño actúa como traductor español-jerga informática, jerga informática-español, eliminando la necesidad de las agotadoras discusiones sobre cómo programar una determinada característica, que no sólo son muy aburridas, sino que hacen que se pierdan horas preciosas de programación en los momentos menos indicados. Los documentos de especificación de diseño deben contener toda la información necesaria para que los programadores realicen su trabajo. Los errores en esos documentos, las omisiones y los problemas surgidos por los cambios en el contexto ajeno a las posibilidades de control del equipo de trabajo no necesitan ahora ser manejadas directamente con los usuarios, sino con el equipo de diseño, que comprenderá las restricciones que impone el avance realizado hasta el momento en que se detecta un error.

Obligar a los programadores a diseñar es en primer lugar injusto y en segundo lugar extremadamente contraproducente: se diseña muy mal y se quita tiempo a los programadores para que hagan el trabajo que realmente saben, quieren y tienen que hacer.

Hacer creíbles y cumplibles los cronogramas

Si bien este ítem podría haberse incluido dentro del ítem de costos por su parentesco, merece un párrafo aparte por sus consecuencias.

Las causas son prácticamente las mismas, pero las consecuencias son distintas: mientras que los problemas de costos impactan en la rentabilidad de los proyectos, la credibilidad es un intangible que cala mucho más hondo y los problemas de credibilidad tienen consecuencias mucho más profundas para las empresas. Las desviaciones en costos se arreglan con más dinero, la falta de credibilidad no.

El proceso que ya describimos, donde se comienza a programar sin tener claro el producto final, donde se toman implícitamente decisiones de diseño que congelan la capacidad de incorporar funcionalidad que luego será considerada imprescindible, genera márgenes de incertidumbre abusivos con respecto a los que sería razonable asumir. Los propios cambios y definiciones tardías agravan este proceso, con riesgo de que se vuelva literalmente infinito. De esta forma es frecuente encontrar decisiones para que en una fecha determinada el producto salga al aire "tal como está" dando una muestra elocuente de que hace mucho que el proceso de desarrollo está fuera de control.

Conformarse con el anhelo de que éste sea un problema exclusivo del Área de Sistemas es miopía. En una empresa de hoy, sin sistema no hay negocio. Hace 25 años primero se ponía el nuevo producto a la calle y luego se pedía a sistemas que desarrolle los programas para soportarlo. El plazo de desarrollo era importante, pero el ciclo de negocios era independiente del ciclo de desarrollo de los sistemas. Hoy el orden es el inverso, primero se implementa el sistema y luego se sale con el producto a la calle, tanto que muchas veces la funcionalidad del software es parte vital del propio producto. El ciclo de negocios será en el tiempo igual o mayor que el ciclo de desarrollo y por tanto las incertidumbres en este último dejaron de ser un problema de costos para transformarse en un problema que afecta el corazón del negocio mismo. La falta de credibilidad en los cronogramas de desarrollo de sistemas deviene falta de

credibilidad en la capacidad de la empresa para desarrollar los negocios que la mantienen viva.

El diseño como arma competitiva

La incertidumbre y falta de credibilidad que el desarrollo de sitios y sistemas impone a las empresas, la impotencia ante el empantanamiento de los proyectos, que de herramienta de cambio se vuelven lastre que mantiene el status quo, que de arma competitiva se vuelven desventaja, merecen un cambio profundo en la forma de concebir y encarar estos procesos.

Superar las carencias en Diseño de la Interacción aparece como uno de los caminos para revertir esta tendencia. No es el único. Tal vez ni siquiera sea el mejor. Pero está al alcance de la mano, con objetivos claros, técnicas definidas y e implementaciones realizables. No supone gastos sino todo lo contrario: no nos cabe duda que el Diseño de la Interacción es para la empresa una opción eficiente y rentable. Un gran negocio.

Segunda Parte

¿Será posible?

72.

Capítulo 4

Los Personajes y el Reparto

El Usuario ha muerto, ¡Viva el Usuario!

Para comenzar a corregir los problemas que ocasiona el mal diseño o la ausencia definitiva del mismo, el primer paso es eliminar la palabra "usuario" del vocabulario. El concepto de usuario es tan elástico, tan adaptable a las necesidades del código, que en vez de ser una herramienta para determinar si el software será útil a la hora que un ser humano de carne y hueso lo use en vivo y en directo, es solamente una forma de justificar porqué es una gran idea incluir este o aquel feature. Siempre habrá un hipotético usuario dispuesto a sentirse feliz por la inclusión de un determinado feature, lo que implica que el concepto de usuario no nos ayudará a distinguir antes de construirla, qué aplicación será adecuada y cuál no, ya que para todas encontraremos un supuesto usuario dispuesto a aplaudir.

Si le tocó asistir a una reunión para determinar las características a incorporar a un producto de software, le resultará familiar una discusión en la que un grupo de profesionales integrado por programadores, gente de marketing, diseñadores gráficos y el gerente de proyecto discuten encarnizadamente por ganar el puesto de defensor número uno del usuario, afirmando con vehemencia que el usuario preferirá esta solución, o aquella otra, o tal vez ésta de acá. En realidad cada uno ve un usuario completamente distinto, extremadamente parecido a sí mismo, a un amigo o a un conocido, y discute encarnizadamente para imponer **su** visión de lo que es mejor.

Supongamos que en una determinada aplicación, llegó el momento de discutir la implementación de un corrector ortográfico. El Gerente de Proyecto piensa en un corrector común y corriente, parecido al que él mismo tiene a disposición en Word: un diccionario default, al que se pueden ir sumando palabras en la medida en la que se utiliza el producto. El programador en realidad odia el corrector ortográfico de Word, que no se da cuenta de qué palabras están en inglés y qué palabras están en español, por lo que sus textos, regados generosamente de términos técnicos en inglés, están siempre llenos del maldito subrayadito rojo. Propone por tanto el corrector ortográfico de sus sueños, con múltiples

diccionarios en múltiples idiomas todos activados a la vez, que además de darse cuenta en qué idioma está cada palabra, tenga la posibilidad de bajar de Internet de forma automática diccionarios técnicos para médicos, abogados, arquitectos y por supuesto, para programadores. En cambio el experto en Marketing piensa en su madre: desde que su hermana vive en el exterior y a pesar de sus 68 años, aprendió a usar el e-mail. El teclado le es extraño, y escribe con muchos errores, más que faltas de ortografía. Le da mucha vergüenza enviar los emails llenos de errores, pero no puede con el corrector ortográfico: todavía no entendió la diferencia entre omitir y agregar. No hay caso, llena el diccionario de cosas mal escritas y cambia las que no hay que cambiar: un desastre. Se imagina un corrector ortográfico tipo Instantmatic que corrija todo con un solo botón, sin las opciones que confunden a las personas como su madre.

La discusión empieza a acalorarse: el programador dice que con un solo botón va a corregir cualquier cosa, al final si en vez de año puso ano, el corrector no se va a dar cuenta ja, ja, ja y el experto en marketing furioso le contestará que él no sabe nada de los usuarios reales: que los usuarios reales para empezar, de noche, duermen, y que cuando dicen "le dije" no hacen el gestito "así con las manos como si estuvieran tecleando", que lo que los usuarios necesitan es un corrector mucho más sencillo. El Gerente de proyecto intentará terciar diciendo que los usuarios más avanzados dominan los correctores comunes, como el de Word, que usan miles de usuarios y que no hay por qué inventar la rueda. El programador replica que Word es una basura, y su corrector una porquería acorde y que él no ve por qué van a perder la posibilidad de desarrollar un corrector realmente bueno, que además en todo caso, si escribe textos en un solo idioma es lo mismo porque el corrector anda igual y que de todos modos va a haber una opción de configuración para forzar el idioma por lo que es lo mismo. Y de nuevo el experto en Marketing va a replicar que los usuarios reales no son expertos y que el producto le tiene que causar una buena primera impresión, y que si mucho después se da cuenta que se podía configurar no sirve de nada. Y la discusión sigue infinitamente. Para ser más exacto, sigue hasta la fecha límite, en la que el producto debe estar listo sea como sea.

El problema está en que un corrector ortográfico no es mejor en abstracto, sin que es mejor para alguien en concreto. Y lo que deberían hacer, antes

de discutir como será el corrector ortográfico, es definir quién es ese alguien.

Los personajes

Existe una herramienta para saldar esta discusión: los personajes.[21] La idea es que el resultado de la discusión sea la definición de una persona ficticia, pero con una descripción detallada que pueda contestar efectivamente las preguntas de qué necesita el usuario.

Un personaje único

El ejemplo de más arriba muestra que es muy difícil, sino imposible satisfacer todos los gustos y necesidades simultáneamente. Intentar satisfacer a todos al unísono da como resultado una de esas palas de campamento, que son pala, hacha, martillo y destornillador a la vez, y que en realidad no hacen ninguna de las cuatro cosas decentemente. Hay situaciones en las que nos vemos enfrentados al desarrollo de sitios o programas de un uso tan general que hacen que el problema sea extremadamente difícil de resolver, pero en el 90% de los casos es posible encontrar un personaje que permita resolver los conflictos.

La experiencia muestra que es más efectivo diseñar para un personaje que tratar de diseñar para varios a la vez, o para el promedio de todos ellos, siempre y cuando el personaje esté bien elegido: se trata de aquel que equilibra el hecho de ser realmente representativo del grupo de personas que utilizarán el programa y que a la vez "lastima" menos a aquellos usuarios no representados directamente.

Elegir un personaje es una decisión de compromiso, precisamente porque diseñar es tomar decisiones de compromiso. De allí surge su valor como herramienta: en la elección se están descartando en forma implícita los diseños que no haremos. Al elegir el personaje, estamos en definitiva eligiendo en forma indirecta el diseño, y creando a la vez una herramienta para evaluarlo, combinación poderosa si las hay.

---

[21] *Alan Cooper: Presos de la Tecnología. Pearson Education, Mexico, 2001 - Capítulo 9, página 123*

Un personaje ficticio

No se trata de elegir una persona. No sirve la mamá del experto en Marketing, ya que el personaje es una herramienta de diseño, y transformarlo en una persona real le hará perder su valor. Si decidimos hacer una aplicación para una persona real, terminaremos saldando las discusiones preguntándole a esa persona qué es lo que prefiere, y sustituyendo la tarea de diseño por un cúmulo de preferencias y opiniones particulares.

Visto desde cierto punto de vista, la metodología de los personajes es análoga a la segmentación que se utiliza en Marketing, pero en vez de definir el segmento por rangos de valores o datos sociodemográficos, se lo define por un representante prototípico.

Una descripción útil

A la hora de definir el personaje, es necesario hacer una descripción con un nivel de detalle tal que ayude a contestar las preguntas de diseño, pero no encasille las ideas o cierre puertas. El nivel de detalle es también una importantísima definición de diseño. Lo que en un caso puede ser irrelevante en otro caso puede ser definitorio. Lo único que no debe faltar jamás es el nombre, que es lo que habilita a eliminar definitivamente la palabra "usuario" de su vocabulario.

Para el sitio Web de un diario Uruguayo con cobertura nacional, el lugar donde vive Federico es irrelevante siempre y cuando viva en Uruguay: que viva en Montevideo, Las Piedras o Salto, es prácticamente lo mismo: accede a casi los mismos canales de TV, a los mismos diarios impresos y a una selección muy parecida de radios. Vibra con los mismos eventos deportivos y participa de los mismos hechos políticos. Tal vez los hechos sociales tengan algunos matices menores. Pero si Mariano es un uruguayo que vive en Toronto la cosa cambia radicalmente: una cosa es generar un producto periodístico para alguien que está inmerso en el contexto informativo/político/deportivo/cultural/social del país y otra muy distinta para alguien que está inmerso en otro contexto pero quiere informarse sobre lo que pasa en su país natal. En este caso, el sexo, el nivel socioeconómico, y la profesión, por ejemplo, son datos mucho menos relevantes que el lugar donde vive.

Un diseño a la medida del personaje, no a la inversa

Otra tentación en la que no se debe caer es la de definir un personaje a la medida de la solución que pensamos, poner la carreta delante de los bueyes. Si queremos incluir un corrector bilingüe definimos un personaje que escribe permanentemente textos en dos idiomas, si queremos un corrector sencillo, definimos un personaje que adora las aplicaciones que tienen cuadros de diálogo con 2 botones. El personaje debe definir el diseño, no al revés.

Es muy común toparse con esta deformación: es un paso sutil, pero mortífero para la eficacia de la metodología. Mientras le plantean al diseñador el problema, va imaginándose una solución y luego crea un personaje a imagen y semejanza de ese diseño preestablecido. La tautología es perfecta, pero inútil. El secreto está en definir el personaje mientras el papel está en blanco, sin un preconcepto del diseño al que queremos llegar.

¿Cómo saber si un personaje está bien definido?

Un personaje está bien definido, cuando es capaz de contestar qué es necesario incluir en el diseño y qué cosas deben quedar fuera. Recuerde que el concepto de usuario es incapaz de dejar nada fuera: siempre hay un usuario al que le servirá incluir un feature dado. Si su personaje está bien definido entonces le ayudará a elegir entre las opciones, lo que significa seleccionar una de esas opciones y descartar el resto. Lo otro es falta de criterio: ponerlas todas y que el usuario elija sobre la marcha.

Supongamos que estamos diseñando un buscador para un sitio Web de un diario. En general asumimos que para un buscador así, el que podamos poner rango de fechas, limitar secciones del diario donde buscar, definir cuántos resultados se muestran y cómo se ordenan, definir más de una palabra de búsqueda, si deben estar todas o alguna de ellas y la prioridad al encontrarlas, serán todas opciones deseables.

Qué opciones incluir y cuáles no debería quedar claro con la definición de nuestro personaje. Por ejemplo:

Jorge tiene 42 años. Es jefe administrativo de una empresa importadora. Utiliza el paquete contable de la empresa y Excel diariamente, así como Lotus Notes para enviar y recibir emails. La computadora no es un enemigo, pero tampoco es uno de sus amigos dilectos: en general aprende a utilizar las funciones sólo cuando son estrictamente necesarias, y después que las aprende, jamás busca nuevas opciones para hacer lo mismo. Todas las mañanas, llega al trabajo, se prepara un café, lee los emails recibidos y ojea en la Web la edición del diario.

Sin duda los departamentos contables de las empresas están repletos de Jorges. También es claro que Jorge no utilizará la búsqueda sofisticada. Jorge necesita una búsqueda sencilla, probablemente una barra de búsqueda y un botón al lado que diga "Buscar" sea suficiente. Esto está determinado por dos elementos: Jorge ojea el diario, no lo estudia. Jorge no aprende una función hasta que no es estrictamente necesaria y una búsqueda sofisticada nunca es necesaria a la hora de ojear el diario, por lo tanto, Jorge jamás va a aprender a utilizarla.

Sumado a esto, Jorge nos dice mucho más sobre cómo encontrar que sobre cómo buscar, cómo deben mostrarse los resultados: de una manera que sea fácil ojearlos. Si la frase de búsqueda es por ejemplo "Batlle y Lula" los resultados serán...

Búsqueda sofisticada:

http://www.midiario.com/noticias/20031109.html?title="ref245234"&top
Presidentes del Mercosur se reúnen en Montevideo
Coincidencias: 11 - Puntaje: 98/100 - Fecha: 20040409 - 23:43 GMT

Búsqueda para Jorge:

Presidentes del Mercosur se reúnen en Montevideo **(ayer)**
... en la reunión participaron el presidente de Uruguay, Jorge Batlle y el presidente de Brasil, Luis Inacio Lula da Silva

La diferencia es sutil, pero crucial. Uno de los problemas más graves del buen diseño es su parecido con la famosa historia del Huevo de Colón[22]. Después que sus colegas vean un buen diseño, afirmarán que es obvio e intrascendente: no se desanime. Cuanto mejor es el diseño, menos se nota, y por lo tanto menos elogios recibirá. Nadie dice "qué buena esta taza para tomar el té", y sin embargo la gente se queja cuando la taza está mal diseñada, cuando el asa no tiene la forma correcta y al agarrar la taza uno se quema los dedos con el calor del contenido. El buen diseño de la interacción produce aplicaciones y sitios Web sencillos, directos, de comprensión inmediata: no recibirá ningún elogio por ellos, salvo de algún experto que conozca lo difícil que es alcanzar un resultado de calidad.

Cualquier programador, al ver el ejemplo de arriba, se afiliará a una de las 2 siguientes teorías:

- ✓ *Da lo mismo*: hay usuarios que prefieren una y usuarios que prefieren la otra. Es cierto que puede haber un Jorge a quien no le interese la URL, pero también es cierto que puede haber un usuario más avanzado, que tenga interés en la URL, o en la fecha exacta en que se subió la nota al sitio.
- ✓ *Es lo que quise decir*: En realidad yo puse eso para simplificar, pero en el código final iba a implementar lo otro. (o sea, el viejo y querido Huevo de Colón)

No es ni lo uno, ni lo otro, sino todo lo contrario. A diferencia de lo que piensa el común de la gente, el secreto no está en las grandes funciones, sino en los detalles. La gente que no trabaja en el diseño de los productos, está convencida que un producto es bueno cuando hace bien lo que tiene que hacer. Sin embargo, un producto que no "hace lo que tiene que hacer" directamente quedará fuera de la categoría. Un producto será considerado bueno cuando sobre la base de que hace lo que tiene que hacer, tiene detalles y elementos secundarios resueltos de una forma particularmente

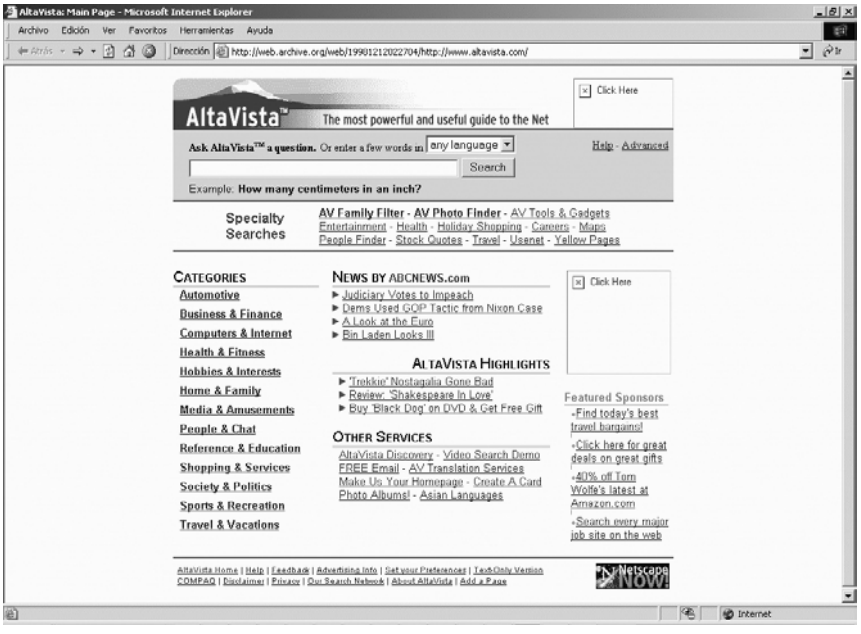
[22] *Se cuenta que Colón desafió a los cortesanos de Castilla a que pararan un huevo sobre la mesa, para mostrar que él era capaz de resolver problemas que nadie podía resolver. Ninguno pudo. Colón cascó suavemente la cáscara y paró el huevo sobre la mesa. La reacción fue inmediata: los cortesanos no aceptaron la demostración, afirmando que la solución era obvia, a pesar de que breves instantes antes habían sido incapaces de dejar el huevo parado sobre la mesa.*

buena. Un auto es un vehículo que tiene cuatro ruedas, asientos, un motor que lo auto-propulsa y alguna forma de conducirlo. Sin embargo, nadie compra un auto porque tiene asientos, motor y ruedas, esa es apenas la condición que le permite participar de la categoría de producto "auto". Los clientes elegirán un auto determinado porque tiene una línea especial, o porque tiene aire acondicionado, o porque tiene un gran portaequipajes, o porque es extremadamente seguro. Todos detalles, desde el punto de vista de la definición genérica de "auto".

Del mismo modo, un buscador es un programa que busca entre una cantidad enorme de páginas Web. Pero no es lo mismo AltaVista que Google y el secreto no está en la función principal. Cuando surgió Google ambos buscaban rápidamente entre una cantidad impresionante de páginas aquellas que contienen las palabras que usted escribió: pero mientras que Google lo hacía en décimas de segundos, AltaVista lo hacía en dos o tres segundos. Mientras que la página de Google descargaba en dos o tres segundos, la de AltaVista lo hacía en diez o doce. Y mientras que en Google siempre me daba cuenta por qué cada vínculo de la primera página de resultados merece estar allí, en AltaVista muchas veces hojeo la página completa y no llego a entender por qué el buscador no encuentra lo que yo quiero encontrar y encuentra millones de páginas que yo no quiero buscar. Todos detalles, que hacen que Google sea utilizado cientos de veces más que AltaVista.

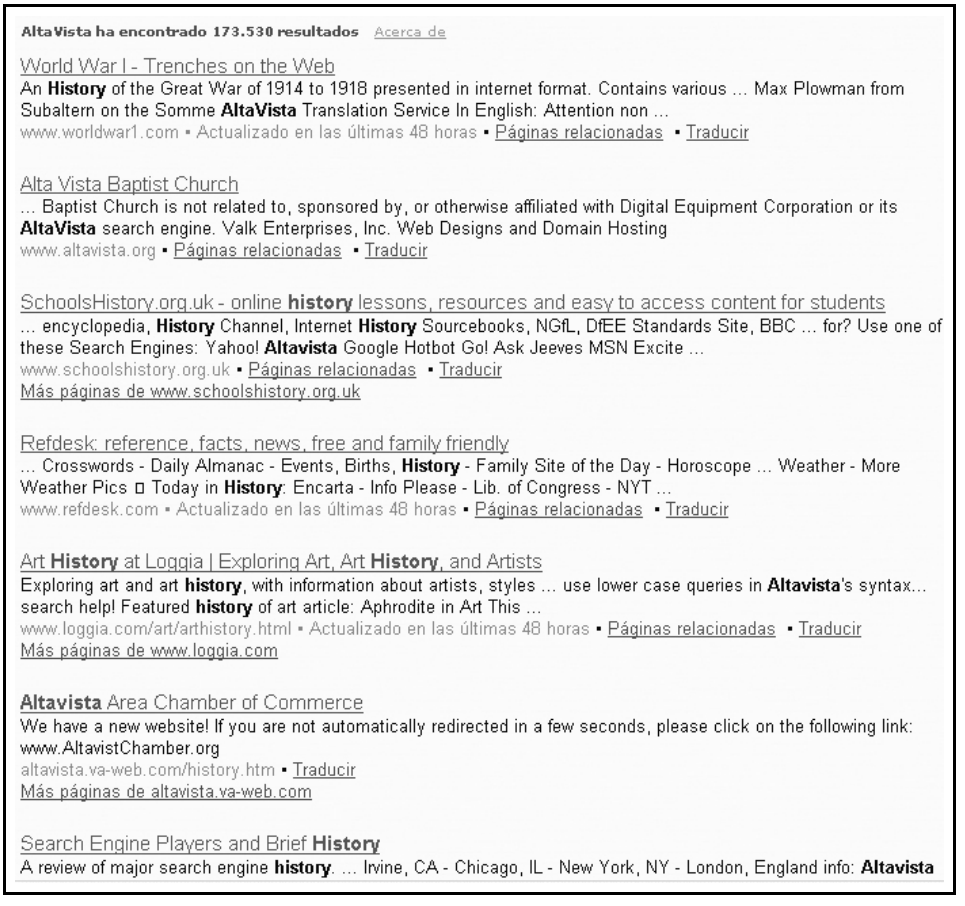
Hoy, AltaVista desistió de su propio camino e intenta ser una copia de Google, tal como lo muestran las imágenes de las páginas siguientes. Sin embargo, sus búsquedas siguen sin convencer. Por ejemplo: ninguno de los resultados que caben en la primera pantalla para la búsqueda de los términos "AltaVista History" se refieren a la historia de la propia AltaVista, algo que Google resuelve perfectamente.

Ahora que Google es una realidad incontenible, todos los buscadores tratan de parecérsele: pero parecido no es lo mismo. Como dice Antoine de Saint-Exupery en El Principito, "lo esencial es invisible a los ojos". AltaVista en su intento de recuperar el terreno perdido copia lo que ve: la gráfica, los botones, la presencia o ausencia de alguna función.



Pagina original de AltaVista

Pagina actual de AltaVista (abril de 2004) ¿Mimetización con Google?



AltaVista ha encontrado 173.530 resultados Acerca de

World War I - Trenches on the Web
An **History** of the Great War of 1914 to 1918 presented in internet format. Contains various ... Max Plowman from Subaltern on the Somme **AltaVista** Translation Service In English: Attention non ...
www.worldwar1.com ▪ Actualizado en las últimas 48 horas ▪ Páginas relacionadas ▪ Traducir

Alta Vista Baptist Church
... Baptist Church is not related to, sponsored by, or otherwise affiliated with Digital Equipment Corporation or its **AltaVista** search engine. Valk Enterprises, Inc. Web Designs and Domain Hosting
www.altavista.org ▪ Páginas relacionadas ▪ Traducir

SchoolsHistory.org.uk - online **history** lessons, resources and easy to access content for students
... encyclopedia, **History** Channel, Internet **History** Sourcebooks, NGfL, DfEE Standards Site, BBC ... for? Use one of these Search Engines: Yahoo! **Altavista** Google Hotbot Go! Ask Jeeves MSN Excite ...
www.schoolshistory.org.uk ▪ Páginas relacionadas ▪ Traducir
Más páginas de www.schoolshistory.org.uk

Refdesk: reference, facts, news, free and family friendly
... Crosswords - Daily Almanac - Events, Births, **History** - Family Site of the Day - Horoscope ... Weather - More Weather Pics □ Today in **History**: Encarta - Info Please - Lib. of Congress - NYT ...
www.refdesk.com ▪ Actualizado en las últimas 48 horas ▪ Páginas relacionadas ▪ Traducir

Art **History** at Loggia | Exploring Art, Art **History**, and Artists
Exploring art and art **history**, with information about artists, styles ... use lower case queries in **Altavista**'s syntax... search help! Featured **history** of art article: Aphrodite in Art This ...
www.loggia.com/art/arthistory.html ▪ Actualizado en las últimas 48 horas ▪ Páginas relacionadas ▪ Traducir
Más páginas de www.loggia.com

Altavista Area Chamber of Commerce
We have a new website! If you are not automatically redirected in a few seconds, please click on the following link: www.AltavistChamber.org
altavista.va-web.com/history.htm ▪ Traducir
Más páginas de altavista.va-web.com

Search Engine Players and Brief **History**
A review of major search engine **history**. ... Irvine, CA - Chicago, IL - New York, NY - London, England info: **Altavista**

Resultados de buscar "altavista history" en AltaVista: recién el séptimo resultado tiene que ver con la historia del buscador

Pero el éxito de Google se basa en una interacción distinta, que tiene una determinada expresión exterior, pero también una implementación interior, que AltaVista no consigue percibir.

La diferencia sutil ubica a éstos y aquellos de un lado u otro de la línea que separa el fracaso del éxito. Un personaje bien definido es una herramienta de diseño que permite dilucidar esas diferencias por un método menos costoso que el ensayo y error y más efectivo que la adivinación.

AltaVista - a **History** (from WebSerch)
Back to Search Engine Specifications, **AltaVista** link. **...** End-1999. **AltaVista** replaced by Inktomi in powering MSN Search. **AltaVista** had **...**
www.clubi.ie/webserch/engines/altavist/history.htm - 31k - Cached - Similar pages

NetHistory: Search
... Although brief, this site provides a reasonable **history** of the early search engines, up until **Altavista** in 1997. **... AltaVista History**. **...**
nethistory.urldir.com/search.php - 10k - Cached - Similar pages

NetHistory: Companies
... akebono.stanford.edu. **AltaVista History**. **... AltaVista** Site White Paper. White paper details the inner workings of **AltaVista**, as well a **history** of the search engine. **...**
nethistory.urldir.com/companies.php - 15k - 10 Jan 2004 - Cached - Similar pages

World War I - Trenches on the Web
... You can access it from any page by clicking on the Search button. An Internet **History** of The Great War. **... AltaVista** Translation Service. **...**
Description: An Internet **history** of the Great War. This is an evolving project, updated frequently with new material.
Category: Society > History > ... > World War I > General Accounts
www.worldwar1.com/ - 18k - Cached - Similar pages

Search Engine Players and Brief **History**
... www.**altavista**.com Office Locations San Mateo, CA - Irvine, CA - Chicago, IL - New York, NY - London, England info: **Altavista** Company **History** Current number of **...**
www.searchengineworld.com/engine/players.htm - Similar pages

AltaVista - a brief **history** of the Alta Vista search engine

Resultados de buscar "altavista history" en Google: de los 6 primeros resultados, 5 son acerca de la historia del buscador, incluyendo los 3 primeros

Lamentablemente, los personajes adolecen del mismo problema que el propio diseño: la mayoría de la gente no capta la diferencia entre un personaje bien definido y uno mal definido. ¿Qué importancia tiene que Jorge lea el diario, ojee el diario o lea atentamente el diario? A los ojos de sus compañeros de equipo, al principio, todos los personajes parecerán iguales. Es más, algunos de ellos jamás llegarán a captar la diferencia entre uno bueno y uno malo y muchos de ellos jamás llegarán a formular una definición útil de personaje. No se desanime. El esfuerzo y la desazón de los primeros momentos tiene premio a la hora de los resultados. Eso sí, no intente explicar que el éxito se debe al excelente diseño, todos le contestarán al unísono: "¿esas paginitas con dos o tres gif son un 'excelente diseño'?".

El Reparto

Habitualmente, un sitio Web o una aplicación no está destinada a un tipo de usuario en particular sino a un grupo de usuarios que desempeñan distintos roles. Por ello, habitualmente la definición de un único personaje no es suficiente, y como en una obra de teatro, es necesario poner sobre las tablas un reparto.

No se trata de definir personajes para distintos tipos de usuarios, como usuarios principiantes, intermedios y avanzados. Se trata de roles distintos, personas que se relacionarán con áreas independientes de funcionalidad en la aplicación o sitio Web que estamos diseñando. Para el sitio Web de una empresa, probablemente el área para clientes sea distinta que el área de proveedores. Los objetivos de los clientes y de los proveedores son disjuntos, así como los objetivos de la empresa respecto de ellos. Por lo tanto el sitio necesitará en ese caso dos personajes distintos para diseñar la interacción con cada uno de esos roles.

Personajes principales

La definición más importante en torno al reparto tiene que ver con los personajes principales y cuántos de ellos son necesarios. Las reglas básicas a seguir son dos:

1. Cada personaje principal debe tener una Interface propia
2. Cuantos menos personajes, mejor.

1. Cada personaje principal debe tener una Interface propia

Si define más un personaje principal, es porque no consiguió determinar un personaje que represente a la mayoría de los usuarios, sin lastimar al resto. Dicho de otra manera, cuando analiza el público objetivo descubre dos grupos homogéneos y bien diferentes entre sí[23]es que necesitará dos personajes. En el ejemplo del diario Uruguayo, si no consigue conciliar la

[23] *En términos matemáticos, un personaje es el representante canónico de una clase, en un conjunto dividido en clases de equivalencia.*

propuesta de diario para quien vive en el propio país, con quien vive en Melbourne o Estocolmo, se requieren dos diarios Web distintos y eso determina la creación de dos personajes distintos.

Esta definición lo obliga a construir dos diarios, uno para Federico que vive en Montevideo y otra para Mariano, que hace 9 años emigró a Toronto. Crearlas separadas implicará el trabajo adicional de realizar dos diseños, más un plus: ¿cómo unificarlos? Su sitio tiene una sola home page, o al menos eso suponía usted antes de comenzar. Las soluciones son múltiples: puede crear dos sitios con URL distintas, dos home pages, puede intentar identificar a los usuarios por su dirección IP. Puede poner un botón que diga: "Mi Diario para los que viven fuera de Uruguay " o desarrollar una Home Page partida al medio. Puede pensar otra decena de opciones todas válidas, pero haga lo que haga estará generando un diseño más complejo que el que hubiera creado si tuviera un solo personaje, lo que nos trae a la premisa numero dos:

2. Cuantos menos personajes, mejor

Cada personaje adicional agrega de forma inevitable una enorme cantidad de complejidad adicional. No se trata solamente de trabajo para usted, sino de trabajo para sus futuros usuarios. Recuerde que los visitantes de su sitio o los usuarios de su aplicación, salvo que sean apologistas de la tecnología, no disfrutan entendiendo las sutilezas implícitas de su diseño: resolver problemas complejos cargará al usuario con esfuerzo adicional para llegar a lo que necesita y eso es siempre negativo, salvo que se trate del juego de la búsqueda del tesoro[24].

El diario El Observador de Uruguay (www.observa.com.uy), por ejemplo, maneja tres conceptos con interacciones distintas pero entreveradas en una única home page: El Observador Digital, que es una imagen digital del diario en papel, observa.com.uy, que es la versión digital del diario y miO (derivado de mi Observa) que es el acceso a las partes pagas de la versión digital del diario y del El Observador Digital. A mí me llevo mucho trabajo llegar a esta deducción y no conozco a ninguna persona no vinculada al diario que sea capaz de definir cada uno de los conceptos. El resultado es una ensalada mixta, que complica enormemente la

[24] *Los juegos son una categoría de aplicaciones muy particular, donde diseñar una interacción directa significa en general estropear el juego y con ello ir en contra del objetivo del usuario de divertirse.*

interacción, empeorado por la decisión de marcar el paso de una parte del sitio a la otra abriendo ventanas, lo que hace que una visita de media hora al sitio deje diez o doce ventanas abiertas en el escritorio.[25]

Por tanto, cuando piense en agregar un personaje, sobre todo si se trata de un personaje principal, piénselo dos veces.

Ese personaje está de más

Como corolario, podemos ver el problema desde el punto de vista inverso: si dos personajes no generan una interacción distinta, entonces uno de los dos está de más. Lo que al principio puede parecer una necesidad de diferenciar, a lo largo de la construcción de la interacción del sitio puede resultar en complejidad adicional innecesaria. Tal vez en un sitio de modas sea lógico pensar en un personaje Hombre y un personaje Mujer. Sin embargo, puede que a pesar de que el hombre y la mujer requieran de una gráfica distinta, o de textos distintos, no requieran precisamente de una interacción distinta, y resulte exactamente lo mismo la interacción para mostrar una colección de modas a un hombre que a una mujer. Así como la colección de primavera utilizará una paleta de colores distinta que la de invierno, una colección de hombres utilizará una gráfica distinta que la de mujeres, pero todas pueden compartir una interacción común. No se trata de una receta y cada caso es distinto, pero si tiene más de un personaje principal, no deje nunca de preguntarse si alguno de ellos no está de más.

Personajes Secundarios

Además de los personajes principales, probablemente su sitio o aplicación interactúe con otras personas, que no son los destinatarios finales, pero que también tienen su relevancia. El diario Web, además de lectores, tendrá redactores, editores, diagramadores, etc. Muchos de ellos interactuarán directamente con el backstage del sitio y requieren por tanto una interacción completamente distinta que la del lector. Para diseñar estas interacciones será necesario definir personajes secundarios.

---

[25] *En el tiempo en el que se corregía el libro, afortunadamente Observa ha detectado el problema y ha introducido un nuevo diseño que corrige numerosos errores, entre ellos el de las ventanas.*

La definición de personajes secundarios debe seguir los mismos criterios que la definición de personajes principales: cada personaje que defina tendrá una interacción particular y menos es siempre mejor.

Para un sitio pequeño, como el sitio institucional de una empresa, alcanza en la mayoría de los casos con un personaje principal. Sitios más complejos requieren hasta 3 personajes principales y hasta 5 personajes secundarios. Esto no es una ley física que se cumplirá sí o sí en todos los casos, como la ley de la gravedad, pero salvo que esté diseñando el sitio multilenguaje de la CNN, si tiene más de 5 o 6 personajes, entre principales y secundarios, lo más probable es que esté haciendo las cosas mal.

Marketing de Reparto

Una vez definido el reparto es necesario hacer que todo el equipo de trabajo lo incorpore, de otro modo su utilidad será nula. Si su equipo de trabajo nunca utilizó una herramienta semejante, probablemente lo miren a usted como a un marciano, y con cara de compasión le den la oportunidad de tirar toda esa porquería a la basura simulando que nunca vieron nada y propongan hacer las cosas como se debe: nos ponemos a programar y después vemos. Una vez más, no se desanime, los resultados valen el esfuerzo.

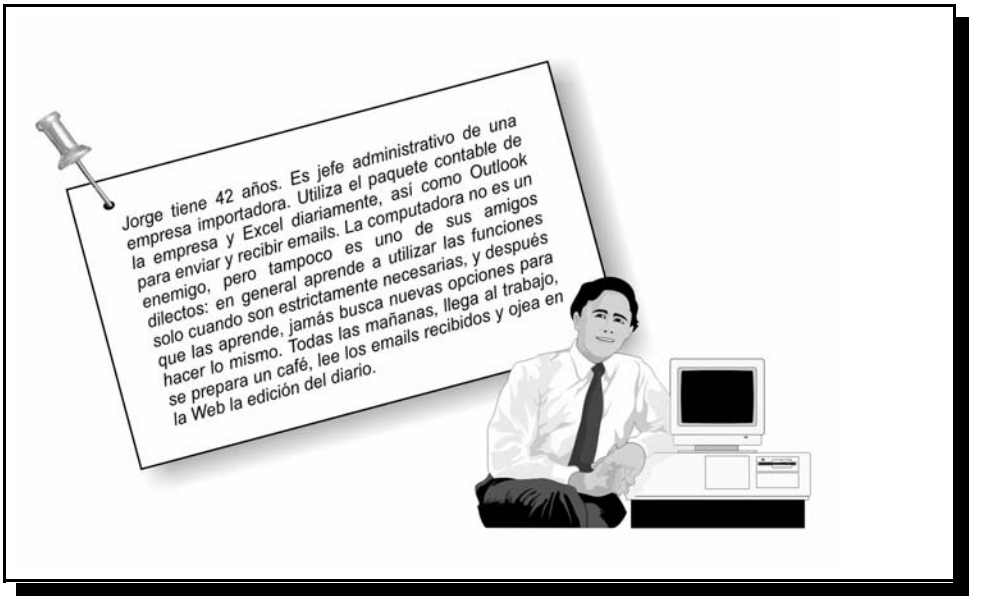
Como de todos modos, haga lo que haga, lo mirarán como a un bicho raro, una de las formas más eficientes para promocionar el reparto es ir en la dirección contraria de la explicación racional, y hacer de cuenta que todos lo aceptan. Parece extraño, pero funciona. Si intenta justificar racionalmente el valor del Diseño de la Interacción y por ende la necesidad imperiosa de adoptar el reparto como herramienta de diseño, perderá horas argumentando en contra de los argumentos más contundentes que la razón humana puede producir:

- ✓ Siempre lo hemos hecho así
- ✓ Nosotros ya sabemos de diseño
- ✓ Eso es una payasada

Es verdad que hay que tener la cabeza abierta a las opiniones, pero también es cierto que para opinar hay que saber. No es esperable que por

intuición uno genere opiniones válidas sobre física cuántica. Para emitir una opinión válida, primero debería estudiar al menos los fundamentos de la física cuántica. Si asumimos que el problema que estamos atacando es la ausencia del diseño como disciplina en la industria informática, no es arriesgado pensar que la mayoría de sus compañeros de equipo no saben nada de diseño. El Diseño de la Interacción es una disciplina técnica, y tiene como tal una base teórica, metodología y una experiencia acumulada surgida a partir de su aplicación sistemática.

Hasta que su gente se habitúe a trabajar diseñando primero y programando después, la adopción de los personajes como destinatarios de la aplicación o sitio a construir pasa por hacer campaña de sus personajes como si fueran candidatos en la próxima elección:



Promueva la utilización de los personajes como herramienta de diseño

Una campaña de publicidad para promover a los personajes como candidatos a usuarios. Busque una foto para cada personaje, coloque su nombre y descripción abajo, fabrique volantes y afiches, póngalos en su escritorio, en las paredes de su módulo u oficina, en el protector de pantalla. Cree una dirección de email para el personaje y envíe las invitaciones a las reuniones en su nombre. Como en publicidad, la

creatividad combinada con los medios de comunicación adecuados llevarán a sus personajes a la victoria.

En las reuniones, cuídese de no utilizar jamás la palabra usuario y reformule cada planteo que le hacen en función del reparto. Volviendo al ejemplo de la búsqueda en el diario Web, si alguien afirma:

- Lo mejor es incluir una búsqueda avanzada, con operadores lógicos y posibilidad de incluir varias palabras clave de búsqueda a la vez.

La respuesta tradicional sería:

- Nuestros usuarios en general ojean el diario en la Web, por lo que es mejor tener una búsqueda sencilla.

Lo que obtendría una contestación contundente y definitiva:

- De todos modos, si hay usuarios que prefieren una búsqueda sencilla, ponen una sola palabra y funciona igual. Nosotros lo incluimos y si un usuario no quiere usarlo no lo usa, pero si otro usuario más avanzado lo quiere usar lo usa. En definitiva es lo mismo.

Ya sabemos que no es lo mismo. Intente algo parecido a esto:

- En realidad Jorge no lee el diario en profundidad sino que lo ojea, lo repasa. A eso se suma que aprende algo solamente cuando lo necesita imperiosamente, por lo que lo que Jorge no va a usar una búsqueda muy compleja si no la necesita. Precisa una búsqueda donde poner una frase, con cuadro de unos 35 caracteres y un solo botón, que diga Buscar, resultados claros con el título como link y muy poco más.

Ahora la respuesta "De todos modos, si hay usuarios que prefieren una búsqueda sencilla, ponen una sola palabra funciona igual ... etc." deja de ser inapelable. No estamos diseñando para un usuario ni sencillo ni avanzado, sino para Jorge. Y la búsqueda que usted está proponiendo es sin duda significativamente mejor para Jorge que la supuesta búsqueda avanzada.

Momento de decisión

Si no consigue promover el reparto, si sus personajes pierden la elección, y se sigue diseñando para el usuario elástico, multipropósito, tal vez llegó la hora de buscar un nuevo trabajo. Pero tenga confianza de que encontrará rápidamente aliados. Salvo en las huestes mas ortodoxas del apologismo, en todas partes hay sobrevivientes encubiertos dispuestos a apoyarlo, que le confesarán en secreto que el sitio que se estaba haciendo le parecía un despropósito, pero que no sabían ni cómo decirlo ni a quién.

Capítulo 5

Los objetivos

Objetivos vs. Tareas

Los objetivos de las personas mueven el mundo. Los hay de todo tipo: personales, globales, sociales, colectivos, también los hay altruistas, egoístas, éticos e inmorales. Independientemente de la religión, opinión política, nacionalidad o cualquier otra característica étnica, sociológica, religiosa o demográfica, los seres humanos nos movemos detrás de nuestros objetivos. Cada uno de nosotros, en forma consciente, inconsciente o semiconsciente ha construido un sistema de objetivos e intenta alinear sus acciones de todos los días en torno a ellos, para que estas acciones viajen sobre una ruta que conduzca precisamente al cumplimiento de esos objetivos.

Las tareas de todos los días no son otra cosa que granitos de arena que ponemos, con la esperanza de que algún día la montaña alcance el volumen necesario para que nuestros objetivos se hagan realidad. Hay una relación profunda entre tareas y objetivos. Los objetivos se desdoblan en tareas. Realizamos tareas, anhelamos objetivos.

A la vez que hay una relación profunda, hay una serie de diferencias importantes entre tareas y objetivos. Los objetivos tienden a ser permanentes en el tiempo, a perdurar inclusive toda nuestra vida. Las tareas tienden a ser momentáneas, pasajeras. Los objetivos son enormemente gratificantes cuando los alcanzamos, las tareas no necesariamente son gratificantes y en muchos casos es cansador o desagradable desarrollarlas. Los objetivos tienden a ser pocos, las tareas por el contrario tienden a ser muchas, en la mayoría de las veces repetidas una y otra vez en pos de un objetivo.

Desde hace decenas o cientos de años, las familias tienen como objetivo tener un hogar agradable para vivir. Este objetivo supone una cantidad importante de tareas entre las que se encuentra mantener ese hogar limpio. Barrer, quitar el polvo, eliminar la basura, lavar la ropa, lavar la vajilla, son tareas tediosas, repetitivas, que las familias realizan un día sí y el siguiente también tras el objetivo de tener un hogar agradable para vivir.

Los objetivos tienden a ser permanentes	Las tareas tienden a ser momentáneas
Los objetivos gratifican a las personas cuando los alcanzan	Las tareas requieren esfuerzo, muchas veces son tediosas y molestas
Los objetivos tienden a ser pocos	Las tareas tienden a ser muchas y repetitivas
Los objetivos impulsan a los individuos a actuar	Las tareas no son un incentivo en sí mismas
El diseño se centra en los objetivos	Los features representan tareas

La relación entre objetivos y tareas

Una última diferencia entre los objetivos y las tareas es que mientras los primeros son más abstractos, intangibles, las segundas son concretas y fuertemente vinculadas a la tecnología. Hace tal vez cientos de años que se barre el polvo de las casas, pero lo que antes se hacía con ramas de paja atadas en forma casera a un tronco delgado, luego se hizo con escobas fabricadas industrialmente, y luego con aspiradoras. Hoy existen además todo tipo de detergentes, ceras y productos anti-polvo. La tarea de eliminar el polvo de la casa es radicalmente distinta hoy que en el año 1900. El objetivo es exactamente el mismo, la familia trabaja en pos de tener un hogar agradable y limpio para vivir.

El diseño se centra en los objetivos, la lista de features en las tareas.

El diseño basado en objetivos

El problema fundamental del desarrollo de sitios y aplicaciones a través de la lista de features, es que se quita el foco de los objetivos de los futuros usuarios, ya que los features se relacionan directamente con las tareas. Es necesario reposicionar el foco y pasar de software que hace cosas a software que ayuda a quien lo usa a conseguir objetivos.

No es que haya una división absoluta entre el software orientado a tareas y el software orientado a objetivos, ni que el primero siempre es un desastre y el otro siempre se lleva todas las palmas. El problema es más complejo. Una aplicación sencilla, que permita desarrollar una tarea puntual en forma eficaz y eficiente, probablemente sea exitosa mientras la tecnología en la que se basa tenga vigencia. Así Lantastic fue un producto muy exitoso para redes pequeñas entre pares (sin un servidor dedicado) mientras los protocolos de comunicaciones fueron propietarios. Permitía conectar los PC rápidamente y ver el disco de la otra computadora como un disco adicional de mi PC, en base a una capa de software de comunicaciones diseñada especialmente. El día que TCP/IP saldó la discusión sobre los protocolos de comunicaciones propietarios, la tarea "comunicar algunos PC" cambió de soporte y Lantastic desapareció del mercado.

Cuando las aplicaciones y sitios Web son más complejos, entonces deben permitir desarrollar una cantidad mayor de tareas. A medida que se mantienen en el mercado, o, si fueron desarrolladas en casa, que se mantienen en producción en la interna de la empresa, aparecen sucesivas versiones, incorporan cada vez más funciones, más features y más tareas. Pero la acumulación de features debe seguir una lógica, tener una lógica interna que le permita a los usuarios navegar en la complejidad, aprender con facilidad, mantener los niveles de productividad. Acumular features sin tomar en cuenta los objetivos de los usuarios, es como agregar ingredientes y condimentos a un plato de comida sin un objetivo preciso, creyendo que cuanto más ingredientes y más condimentos, más gustosa será la comida y más satisfechos estarán los comensales. Los platos más sofisticados, que mezclan muchos ingredientes y condimentos, inclusive algunos aparentemente contradictorios, requieren de una mano maestra que mantenga la cordura, algo más difícil de alcanzar cuantos más ingredientes y condimentos se incorporen. Los objetivos de los usuarios potenciales deben jugar el rol de equilibrar y amalgamar las funcionalidades del software en un todo único. El diseño basado en objetivos cumple el rol de garantizar que esto sea así. El software siempre es un conjunto de herramientas para desarrollar tareas, no es allí que radica la diferencia entre una buena y una mala aplicación, sino en la organización, cohesión y coherencia que este set de herramientas tiene.

Si Microsoft no consigue vender una cantidad adecuada a sus intereses de licencias de Office 2003, es porque a pesar de que tiene literalmente miles de funciones más que Office 97 o 2000, los clientes no sienten que ninguna de esas funciones los acerquen a sus objetivos. De hecho, pregunte a su alrededor y comprobará que salvo los vendedores de Microsoft y los profesores que dan cursos de Office, los demás mortales, inclusive los apologistas más ortodoxos, son incapaces de nombrar de memoria 10 diferencias entre la versión 97, 2000 y 2003 de Word. Para un usuario de Office, que ahora soporte XML para poder desarrollar Web Services sobre los documentos es intrascendente, inexistente. Aquí entra un apologista y afirma: pero hay usuarios más avanzados que sí lo necesitan y por lo tanto lo valoran. El argumento no parece muy poderoso, porque las empresas y clientes hogareños no adoptan Office al ritmo de sus nuevas versiones, sino al ritmo de la renovación de sus PC, cuando la renovación de hardware literalmente los obliga.

Office es tal vez la aplicación más masiva que existe en el mercado, y es también la muestra más contundente de que agregar funciones y más funciones porque hay algún grupo de usuarios con características particulares que lo valora es un método agotado. Las funciones agregan complejidad en forma exponencial, rompen con la coherencia lógica de la aplicación y en este caso no aportan valor adicional alguno para los clientes, porque no están en línea con los objetivos que los usuarios de Office intentan alcanzar al utilizarla. Agregando más y más funciones, Office no acelerará ni el ritmo de adopción ni el ritmo de migración para transformarse en un producto tan masivo como el celular o el microondas.

No es lo más probable que usted tenga que enfrentar las disyuntivas de quienes desarrollan y comercializan Office. Por el contrario, es muy probable que se enfrente a las disyuntivas de desarrollo de aplicaciones con un espectro de usuarios más acotado, más homogéneo, más manejable, tanto si está vinculado al software comercial o si trabaja junto al departamento de sistemas de su empresa. Y también es muy probable que usted no tenga detrás el presupuesto billonario de desarrollo de aplicaciones de Microsoft. Nos enfrentamos por tanto a un problema más acotado, más manejable, pero tenemos una sola bala en el cargador. Será lo mejor ir por terreno firme: entender los objetivos de los clientes

potenciales y desglosarlos en un conjunto de tareas adecuado, coherente, completo y compacto.

Los objetivos y los personajes

Diseñar es encontrar dentro del espectro de objetivos de nuestros personajes el nivel justo de detalle que los comprenda y describa de forma útil para el desarrollo de la aplicación. Podemos asumir que todos los personajes tendrán un objetivo de máximo nivel que podemos denominar "Ser Feliz". Sin embargo, Ser Feliz no nos aporta nada al diseño de ninguna aplicación. El diseñador debe navegar entre los objetivos de sus personajes, de modo de encontrar aquellos para los cuales la aplicación que estamos diseñando será una herramienta útil a hora de alcanzarlos.

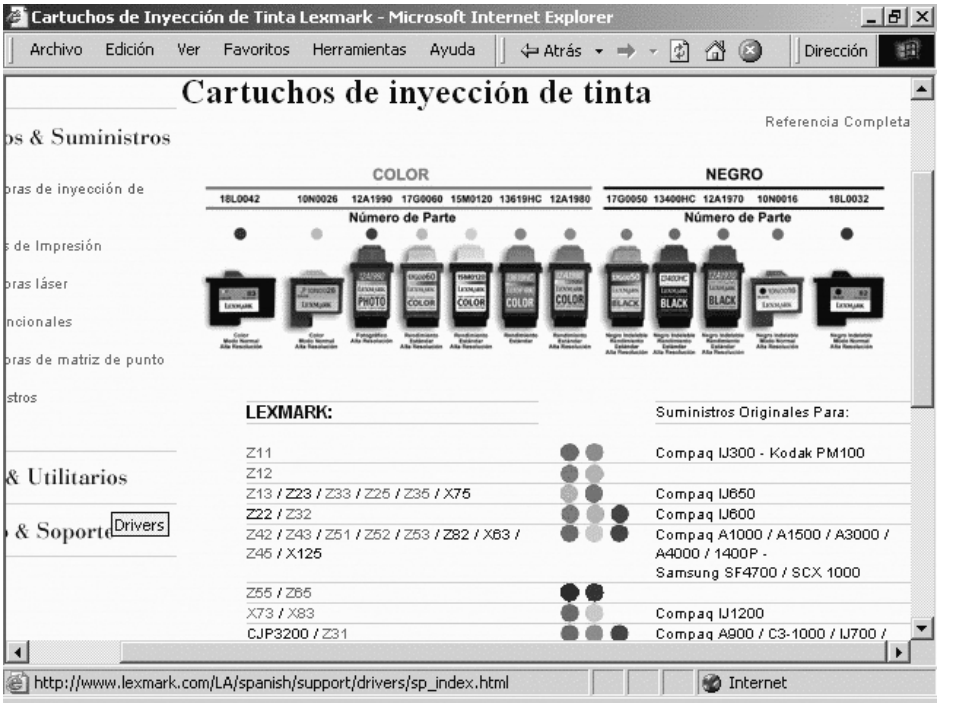
Es apabullante la cantidad de sitios Web que exhiben impúdicos su incomprensión de los objetivos de los visitantes, que muestran entre coloridos gráficos y tipografía esmerada las tripas de la implementación y la interna empresarial. Desde el punto de vista de una empresa que fabrica impresoras, el área de comercialización de impresoras seguramente está separada del área de venta de cartuchos, que en la jerga de los informáticos se denominan "suministros" y estas dos están a su vez separadas del área de desarrollo y soporte de drivers. Para Juan, que revende productos de mercería, su impresora de chorro de tinta, los cartuchos carísimos que consume y todos los problemas y trucos para imprimir gastando lo menos posible son una entidad única e indivisible. El objetivo de Juan es enviar propuestas a sus clientes con una presentación profesional, y su impresora es una herramienta para ello, dolores de cabeza incluidos. Lexmark no tiene en cuenta a Juan, sino a su organigrama interno, y se lo hace saber en la home page:

Cuando Juan ubica su impresora, se percata que aún no ganó terreno alguno para saber qué cartucho comprar o para averiguar si hay un nuevo driver que resuelva ese problemita por el que la impresora se tranca a veces. En la página de la impresora no hay más que vínculos genéricos que lo obligan a empezar de nuevo. Por ejemplo el vínculo "suministros" en la página de su impresora lo conduce a una tabla, donde Juan debe comenzar a buscar nuevamente desde el principio entre todos los cartuchos disponibles, que se muestran organizados de una forma distinta que las impresoras, en el medio de la jerigonza interna con que Lexmark bautiza sus productos.

La página de la impresora Lexmark de Juan, no hay pistas sobre los cartuchos, sino un link a la página general de suministros

HP por el contrario tiene un sitio de impresoras diseñado a la medida de los objetivos de Juan. Una vez que encuentra su impresora, puede directamente desde allí encontrar fácilmente todos los cartuchos necesarios, junto con el abanico de todos los accesorios disponibles para su impresora. También puede acceder a soporte técnico de esa impresora, área donde encontrará los drivers, documentación técnica y consejos útiles, sin tener que volver a recorrer toda la estructura jerárquica del sitio.

No se trata de que Lexmark tenga peor tecnología detrás de su sitio Web, ni que quienes lo desarrollaron sea menos inteligentes o capaces que los que desarrollaron el sitio de HP o cualquier otro sitio Web. Se trata de que Lexmark aún no consiguió alejarse del modelo tradicional de desarrollo y su competidor, en este caso HP, sí lo hizo.



Los suministros de Lexmark, una jerarquía nueva, con nombres y lógica también nueva

Los dos sitios tienen las mismas funciones, los mismos features: los dos muestran las impresoras y sus productos relacionados, pero mientras en uno el ojo que mira es el de Juan, en el otro el ojo que mira es el del programador, lo que obliga a Juan a recorrer tres árboles jerárquicos[26] distintos si quiere encontrar para su impresora información general, un cartucho y un driver.

---

[26] *Los arboles jerárquicos patas para arriba, con la raíz arriba y las ramas que crecen hacia abajo, son una estructura que los programadores adoramos y los sobrevivientes nunca terminan de entender. Ni que hablar de recorrerlos recursivamente. "¿Me podes explicar otra vez eso de que dentro de la carpeta pueden haber documentos u otras carpetas que a su vez pueden tener documentos u otras carpetas?"*

Para la impresora HP de Juan, un solo lugar reúne toda la información y opciones necesarias

Dicho en términos técnicos, mientras que el modelo de representación del sitio de Lexmark muestra el modelo de implementación, el modelo de representación del sitio de HP refleja el modelo mental de Juan.

Juan formará su modelo mental del problema a partir de sus objetivos y de su experiencia anterior a la interacción con el sitio. Enviar propuestas a sus clientes con una presentación profesional existía como objetivo antes de comprar la computadora. Por su parte la impresora, sus cartuchos y sus

drivers existían antes de entrar al sitio. Juan, que como homo sapiens sapiens es un animal explicativo, entiende que al imprimir de alguna manera lo que imprime "pasa" por el driver, de allí al cable y de éste a la impresora. En su modelo mental imprimir, driver, cable, impresora y cartucho son una unidad, un todo. Por el contrario, para la empresa los drivers son un costo muerto: se regalan. Las impresoras son sólo un medio para vender cartuchos: apenas salvan el costo. Los cartuchos son una vaca lechera que mantiene a los otros dos y genera las ganancias: cada uno vale una fortuna. Es posible imponer el modelo interno de la empresa en el sitio y forzar a Juan a entenderlo, reformulando su modelo mental. Pero es imposible hacerlo sin que Juan gaste energías en ese proceso. Será entonces más redituable diseñar el sitio o la aplicación para que responda al modelo mental que Juan trae cuando lo visita: le permitirá gastar todas sus energías en alcanzar sus objetivos.

Hay veces que el sitio Web o la aplicación están tan mal diseñados, en las que hay tanta distancia entre el modelo mental del usuario y el modelo de representación, que el esfuerzo que debe hacer el usuario para reformular su modelo mental antes de ser capaz de usar las funcionalidades del software se torna absolutamente desproporcionado frente al propio beneficio que brindan al usarlas. Dicho de otra forma: el costo de aprender a utilizar una aplicación es tan grande comparado con el beneficio, que el usuario directamente desiste de usarlas. La Web ha llevado este problema a límites antes desconocidos, porque la definición del nivel de dificultad con que me enfrentaré al utilizar un sitio Web se evalúa en la mayoría de los casos en algunos segundos, tal vez minutos. En la Web, el premio por acercar el modelo de representación de la aplicación al modelo mental de nuestros usuarios es sensiblemente mayor que en las aplicaciones tradicionales.

La teoría del magazine de balas

Imagine que el visitante de su sitio Web es un soldado que tiene en su rifle un magazine con 30 balas. Cada tarea que usted le presente, requerirá al menos una bala. Si es más compleja, más balas. Cuando se le terminen las 30 balas, el soldado se enfrentará a una de estas dos situaciones: habrá vencido y usted tendrá un cliente satisfecho o estará lejos de sus

objetivos, solo y desarmado, con lo que seguramente optará por escaparse rápidamente del sitio.

Cuando diseña su sitio Web, o eventualmente una aplicación pero en particular un sitio Web, tenga en mente que cada decisión de diseño le costará a su visitante una o más balas, y que cuando se termine el magazine, habrá terminado su oportunidad de convencerlo. Si lo obliga a gastar balas inútiles, su chance baja. Hay por ejemplo una fuerte discusión sobre el color que deben tener los links en las páginas. Es cierto que el azul subrayado no encaja con muchos diseños gráficos estilizados, pero también es cierto que no ponerlos azul subrayado obligará a su visitante a gastar una bala para encontrarlos. No tiene nada de malo, ni su sitio fracasará porque los links no sean azul subrayados, pero si a esa bala le suma la bala que hay que gastar para que la página descargue en más de un minuto, otra derivada de lo difícil que resulta leer textos celestes sobre una foto clara, y la bala que hay que gastar para entender una estructura de productos distinta a la que el visitante tiene en mente, y muchas otras balas, entonces las chances de éxito serán mínimas.

La teoría del magazine de balas es extremadamente útil al diseñar. Obliga a mantener el foco en el modelo mental del futuro visitante del sitio, partir de los objetivos para mantener a raya el modelo de implementación y dosificar las desviaciones. Inclusive, la búsqueda tajante de un diseño en el que el visitante no deba utilizar ninguna bala, suele ser muy exitoso: ¿cuántas balas es necesario gastar para entender el sitio de Google?

Distintos tipos de objetivos

Cada persona tiene un sistema de objetivos, donde se combinan múltiples niveles a partir de su interrelación con el contexto económico y social. Para el diseño, es importante tomar en cuenta tres de estos niveles: los objetivos personales, los objetivos corporativos y los objetivos prácticos.[27]

---

[27] *Alan Cooper: Presos de la Tecnología. Ibidem. Página 154*

Los objetivos personales

Independientemente del medio en el que una persona esté inserto, y del tipo de tarea que esté desempeñando, siempre tendrá en mente sus objetivos personales e intentará actuar para alcanzarlos. Las situaciones que fuerzan a las personas a actuar contra sus objetivos personales son extremadamente violentas y el software debería evitar a toda costa obligar a sus usuarios a caer en este tipo de situaciones.

Todos los individuos manifiestan de alguna manera su necesidad de autoestima, su objetivo de ser útil, su necesidad de crecer como persona. Estos objetivos personales se manifiestan en otros más concretos como el objetivo de no parecer tonto, cometer pocos errores, mostrar capacidad y competencia ante sus pares y superiores. Los niveles de complejidad innecesaria en las aplicaciones, el software que obliga a descargar el magazine completo de balas apenas empezamos o los programas con los que nunca damos en el clavo, atentan contra esos objetivos personales y generan por tanto situaciones violentas en la relación usuario-software. Es frecuente encontrar gente insultando duramente a su computadora, pegando puñetazos a la mesa o arrugando con furia los papeles recién impresos. Es menos frecuente pero lamentablemente no muy difícil de encontrar, gente que golpea el teclado, el monitor o la computadora misma, o gente que directamente se levanta de su escritorio totalmente entregado.

Un empleado que cuando va a buscar su trabajo a la impresora del piso descubre bajo la mirada atenta de sus compañeros que en vez de las dos hojas de texto se imprimieron decenas de hojas con chirimbolos incomprensibles, u otro que frente a su jefe no consigue cambiar el formato de una tabla o un estudiante que intenta convencer a su profesor que hizo el trabajo, pero que la computadora literalmente lo tragó, son personas que ven ultrajados sus objetivos personales por culpa del software. Si su software está bien diseñado, minimizará el riesgo de que sus usuarios caigan en este tipo de situaciones extremas, lo mismo que en otras menos dramáticas pero más frecuentes.

Habitualmente no es necesario escribir para cada personaje los objetivos personales, ya que son genéricos y comunes a todos ellos, pero si su equipo de desarrollo es novato en el Diseño de la Interacción o si está

rodeado de numerosos apologistas, no dude en promover los objetivos personales de sus personajes con todos los recursos que tenga a su alrededor. Por lo demás, gaste hasta la última gota de sus energías para combatir cualquier decisión que atente fuertemente contra los objetivos personales de los futuros usuarios.

Los objetivos corporativos

El software comercial está dirigido en la mayoría de los casos a satisfacer las necesidades de las empresas. Los folletos lo promueven en base a la promesa de que aumentará la eficiencia organizacional, reducirá los costos, permitirá ganar participación de mercado o reducirá los ciclos de negocios.

Ninguna empresa adquirirá o desarrollará software con una expectativa distinta que la de acercarse a sus objetivos empresariales, por lo que diseñar software que no tome en cuenta estos objetivos es un suicidio. Este es el tipo de objetivos más analizado y considerado dentro de la industria informática, y a la corta o a la larga, las aplicaciones que se mantienen en el mercado han conseguido que sus empresas clientes sientan que son una herramienta válida para lograr sus objetivos corporativos.

Sin embargo, que las empresas alcancen con sus aplicativos los objetivos corporativos, no implica que quienes utilizan dichos aplicativos en la vida real se sientan satisfechos al utilizarlos. Dentro de las empresas es muchas veces una obligación utilizar aplicaciones complicadas, trabajosas y desagradables, que molestan enormemente a quienes las usan todos los días. Recuerde: por más que las empresas actúen en pos de sus objetivos corporativos, las decisiones las terminan tomando individuos. Si esos individuos no están satisfechos con su aplicación, harán todo lo que esté a su alcance para dejarlo fuera del negocio, más allá de que su aplicación sea potencialmente capaz de lograr alguno de esos objetivos corporativos.

Los objetivos prácticos

Mientras que los objetivos personales son demasiado genéricos para definir la interacción de una aplicación y los objetivos empresariales son demasiado vagos, el tercer grupo de objetivos se erige como el candidato.

Los objetivos prácticos son aquellos que unen los objetivos corporativos con los objetivos personales.

Mientras que el objetivo corporativo "ganar participación de mercado" no nos dice demasiado, y los objetivos personales de "no sentirse tonto, cometer pocos errores y realizar una cantidad adecuada de trabajo" tampoco nos ayudan, el objetivo de "procesar en el día todos los pedidos recibidos" o "enviar todas las propuestas en menos de 48 horas" brindan las pistas necesarias. Se trata de los objetivos prácticos, que actúan como bisagra de unión entre los objetivos personales y los objetivos corporativos.

El sistema de objetivos se debe estructurar de modo que la aplicación toda alcance los objetivos corporativos, mientras que la utilización por parte de cada usuario alcance los objetivos prácticos navegando a favor de los objetivos personales. Hay una interrelación muy fuerte entre los tres niveles de objetivos, pero son los objetivos prácticos los que guían el Diseño de la Interacción, ya que son éstos los que se alcanzan con la interacción real hombre/computadora de cada día. Los objetivos corporativos y personales actúan como marco de referencia: los primeros para definir la estructura lógico funcional de la aplicación toda, los segundos para garantizar la satisfacción de los usuarios al utilizar el software.

Los objetivos y el reparto

Existe una relación muy estrecha entre los objetivos y el reparto. Los objetivos a definir no son objetivos en abstracto sino objetivos para los personajes del reparto. Si analizando la aplicación y los personajes no consigue entender sus objetivos, o le parece que la aplicación ayuda a alcanzar objetivos para personas distintas de sus personajes, habrá que barajar y repartir nuevamente.

Como en el caso de la definición del propio reparto, encontrará sitios o aplicaciones muy grandes, o extremadamente complejas que podrían requerir múltiples objetivos para algunos de sus personajes. En más del 90% de los casos uno o dos objetivos por personaje es lo correcto. Como

complemento, cada personaje del reparto, sea este principal o secundario, debe tener al menos un objetivo.

Desarrollar el software en torno a los objetivos de sus potenciales clientes lo ayudará a cruzar la estrecha línea entre el software que hace cosas y el software que cautiva a quien lo usa.

Capítulo 6

Los escenarios

El último paso en el proceso del Diseño de la Interacción es la descripción de los escenarios de uso del sistema. La función de los escenarios es vincular los objetivos del usuarios con las acciones concretas que desarrolla al utilizar la aplicación.

Llegó la hora a las tareas

Una vez definidos los objetivos para todos y cada uno de los personajes del reparto, entonces sí llegó la hora de poner la atención en las tareas. El próximo paso, el primero en la determinación de los escenarios, es desglosar cada uno de los objetivos de los personajes en tareas.

Ahora el objetivo "mantener el hogar limpio" debe dividirse en barrer, sacar el polvo, lavar la loza y limpiar los vidrios. Cada uno de los objetivos debe ser analizado minuciosamente para encontrar la lista completa de las tareas que hay que realizar para alcanzarlo. Recuerde que los objetivos tienden a ser pocos, pero las tareas tienden a ser muchas.

Para cada tarea, será necesario ponderar algunos atributos que nos permitirán mas adelante ordenarlas y ubicarlas en su lugar correcto. Utilizaremos para ello una escala de uno a cinco (uno más importante, cinco menos importante), pero en realidad lo importante no es la escala, sino la ponderación de los atributos en concreto. Los atributos a ponderar son:

- ✓ *Nivel de importancia*: cuán importante es la tarea para alcanzar el objetivo. Si tengo que ir a mi trabajo, tomar el ómnibus es una tarea muy importante. Leer los titulares de los diarios en el quiosco es una tarea que realizo todos los días cuando voy al trabajo. Hacen el recorrido un poco más placentero, pero desde el punto de vista del objetivo "ir al trabajo" es una tarea de importancia 5. El nivel de importancia nos va a ayudar a priorizar la programación de features en el proceso de desarrollo (recuerde que hay una relación directa entre un feature y una tarea).
- ✓ *Nivel de dificultad de implementación*: El nivel de dificultad para implementar una tarea no está directamente vinculada a su importancia o a algún otro atributo. Hay tareas muy difíciles, pero

no tan vitales para alcanzar un objetivo determinado. En combinación con el nivel de importancia, el nivel de dificultad nos va ayudar a ponderar el esfuerzo de diseño y programación que hay que poner en una tarea determinada. Por ejemplo, las tareas muy importantes y muy complejas requerirán de un esfuerzo de diseño y programación considerable. Pero eso no será necesariamente así para una tarea de importancia mínima y muy difícil de realizar.

- ✓ *Frecuencia*: La frecuencia con que repetimos una tarea tiene una importancia vital para el diseño. En primer lugar, las tareas repetidas muy frecuentemente tienen el efecto de la gota que horada la piedra: cualquier defecto menor de diseño, que se repite una y otra vez termina agotando nuestra paciencia. El mismo defecto, si apareciera en forma muy esporádica, no llegaría siquiera a molestarnos. Por otra parte, las tareas muy frecuentes tienen ciclos de aprendizaje muy rápidos, y eso determina la necesidad de un diseño muy sofisticado, para brindar aceleradores, atajos y herramientas más sutiles que permitan al usuario sacar provecho de su experiencia.

Puede que su diseño requiera otros atributos adicionales de ponderación de las tareas, como por ejemplo el nivel de importancia que le atribuyen los clientes, la presencia de las tareas/features en el software de sus competidores u otros. Proceda en ese caso de la misma forma que con los ítems anteriores.

Cuando concluya, más que una lista tendrá frente a usted un sistema de tareas, agrupadas por objetivos y ponderadas por los atributos que determinarán la importancia que tienen a la hora de encarar el diseño de la interacción de cada una de ellas con los personajes para los que estamos diseñando.

Qué son los escenarios

Para entender qué son los escenarios, es muy bueno pensar que una aplicación de software es como su casa. Piense en ella: está repartida en habitaciones, cada una de las cuales está destinada a cumplir un objetivo

práctico particular, a lo sumo dos. Cada habitación cuenta con todos los muebles, utensilios y elementos necesarios, ordenados de una forma coherente y adecuada para realizar las tareas que permitan alcanzar el o los objetivos que tiene asociada. La cocina es para cocinar, los dormitorios para descansar, el comedor para comer y el living para la recreación y el ocio. También podemos tener un cuartito del fondo para guardar trastos, un escritorio, un garaje, un lavadero y muchas otras opciones que dependen de las preferencias y posibilidades de cada uno de nosotros. Pero la esencia se mantiene: nuestra casa es un sistema ordenado de habitaciones diseñadas de modo que podamos en ellas desarrollar las tareas que nos acerquen a un objetivo práctico concreto.

Ocasionalmente podemos realizar las tareas que habitualmente se realizan en una habitación en otra, así podemos comer en la cocina, dormir en el living o lavar la ropa en el baño. Si estas tareas se realizan en forma permanente en una habitación distinta que la original, veremos a la nueva habitación anfitriona transformándose para albergar los utensilios, herramientas y elementos necesarios para desarrollar la nueva tarea. Si el comedor se utiliza para estudiar, los libros, cuadernos, carpetas y lápices pasarán a la corta o a la larga a formar parte permanente del decorado.

Nosotros nos movemos por la casa en la medida en que nos proponemos nuevos objetivos prácticos, pero no es sensato ni cómodo tener que cambiar de habitación para desarrollar las tareas necesarias para un solo objetivo. No es razonable sacar las cosas de la heladera en el living, mezclar los ingredientes en el comedor, cocinarlos en la cocina, y lavar los utensilios de cocina en la pileta del fondo, desplazándose permanentemente de un lugar a otro en la medida que vamos preparando el almuerzo.

La situación a la que arribamos en el instante en que terminamos de mudarnos es sumamente ilustrativa: la casa tiene todas las habitaciones y tenemos todos los muebles, herramientas, accesorios y utensilios dentro, pero embalados en cajas. Nos llevará un tiempo importante llegar nuevamente al equilibrio que teníamos en la casa anterior. Es cierto que al principio no sabemos siquiera en cual de las malditas cajas hay un plato y una olla, pero después de un primer reparto grueso de muebles y herramientas hay un proceso de acomodamiento sutil hasta que cada cosa

encuentra su lugar definitivo y nuestro nuevo hogar tiene el equilibrio funcional necesario.

Los escenarios son al software como las habitaciones son a la casa. Son áreas funcionales de una aplicación o sitio Web que deben contar con todas las herramientas necesarias para que uno de nuestros personajes alcance uno de sus objetivos prácticos. Si facturar una venta es un objetivo práctico de uno de nuestros personajes, entonces debe haber un área del software donde pueda buscar el cliente y modificar sus datos si es necesario, buscar los artículos y modificar sus precios, calcular los totales e imprimir la factura. Pedro debe poder facturar sin salir de la habitación, es decir, sin tener que moverse del módulo de facturación al modulo de clientes y luego regresar al de facturación para continuar. Internamente, una factura consulta y actualiza tablas de datos distintas: los clientes, los productos, los impuestos, los precios. Desde el punto de vista de la programación es lógico y conveniente que cada una de estas tablas de datos tengan sus rutinas de actualización propias. Para dar de alta un producto, no es necesario tener en cuenta la tabla de clientes. Para dar de alta un impuesto, no es necesario tener en cuenta la tabla de productos, siempre desde el punto de vista de la programación. Resulta entonces lógico para el programador definir primero el modelo de datos, es decir definir las tablas, programar luego las rutinas para insertar, modificar o eliminar datos en las tablas, lo que en la jerga se define como ABM (de altas, bajas y modificaciones) y por último construir las funciones de más alto nivel, cuidando la integridad referencial de la base de datos: una factura tiene que estar emitida a un cliente válido, tiene que incluir productos válidos, que tengan al menos un precio en la lista de precios y asignado un código de impuesto pre-definido. Cuando Pedro se enfrenta al sistema para hacer una factura, debe salir de la habitación de facturación y dirigirse a la habitación de clientes para modificar la dirección del cliente y debe ir luego otra habitación, la de productos, para hacer una modificación en el precio del artículo que está vendiendo.

Aquí viene otra vez el Huevo de Colón: un apologista explicará que se puede vincular el ABM de clientes para que sea invocado desde la pantalla de facturación, lo mismo que la rutina de modificación de precios de modo que siquiera se note que son tres cosas distintas. Sí, se puede, es más, probablemente sea la solución óptima al problema que estamos planteando, pero la mayoría aplastante del software comercial no lo hace,

sencillamente porque no fue diseñado pensando en Pedro, sino pensando en la base de datos, la integridad referencial y la lista de features. Es posible hacerlo, aparece como obvio cuando nos lo dicen, pero se necesita un minucioso trabajo de diseño de la interacción para que se haga realidad.

No existe una relación matemática biunívoca entre personaje/objetivo y un escenario. Hay una base lógica de correspondencia, pero es necesario probar, abrir la mente, proponer ideas, antes de alcanzar un conjunto de escenarios adecuado. Definir los escenarios no es una tarea mecánica, sino una tarea de diseño y por tanto requiere a la vez de técnica y creatividad.

Los escenarios son el esqueleto del modelo de representación, que luego llenaremos de botones, menúes, combos, campos de edición y todo el bagaje de artefactos de software con que cuentan las aplicaciones y sitios Web. Definir correctamente los escenarios garantizará que la aplicación que verán en la pantalla los futuros usuarios estará adaptada a su modelo mental, y no al modelo de implementación.

Tipos de escenarios

Escenarios de uso diario

Dentro de los escenarios que terminará teniendo una aplicación, hay probablemente uno o dos, a lo sumo tres, que serán aquellos en que los futuros usuarios desarrollarán la mayoría aplastante de sus tareas. Cuando utilizo Word, paso el 99% del tiempo enfrentado a la pantalla de edición. Puede que utilice a veces la función de combinar correspondencia, o la de crear nuevos estilos, pero haga lo que haga, el escenario de uso diario de Word será el de la pantalla de edición.

Una curiosidad: los protectores de pantalla son consecuencia de estas característica. Dado que las aplicaciones están en general todo el día en una pantalla determinada, el fósforo del monitor se desgasta más rápido en los pixeles que están siempre prendidos. Como solución nacieron los protectores de pantalla, que de simples "oscurecedores" del monitor,

devinieron en las sofisticadas aplicaciones que muestran fotos, animan muñequitos y otras veleidades por el estilo.

El tiempo que los futuros usuarios pasarán en estos escenarios, elevan su importancia en forma exponencial frente al resto de los escenarios. Cada interacción, cada definición de diseño cobra importancia especial por formar parte de un escenario de uso diario. Por ejemplo, a pesar de que la mayoría de sus compañeros de trabajo sean incapaces de nombrar 5 de las miles de diferencias entre Word 97 y Word 2000, pruebe mencionarles ésta: en Word 2000, deshacer continúa funcionando aún después de guardar el archivo, algo que antes no sucedía. Comprobará que más de la mitad de ellos notaron este detalle y lo valoran enormemente. Se trata de una sutileza, donde el sistema de archivos imponía su rigidez al modelo de representación. El hecho de que forme parte del escenario de uso diario en el que usted pasa el 99% del tiempo cuando utiliza Word amplifica este detalle hasta hacerlo visible a todos los usuarios de la aplicación.

En los escenarios de uso diario, sus usuarios aprenderán rápidamente los trucos para dominarlos y los recordarán de una visita a la siguiente, ansiosos de descubrir nuevos atajos. Las deficiencias y defectos los lastimarán hasta hartarlos. Las sutilezas, tanto las positivas como las negativas serán siempre descubiertas. Los escenarios de uso diario son la quintaesencia del diseño de aplicaciones: es allí donde los usuarios serán infinitamente felices o rabiosamente desdichados.

Escenarios de uso periódico

Los escenarios de uso periódico son aquellos que responden a algún objetivo de algún personaje, pero no son utilizados a diario sino en forma esporádica. Puede existir una aplicación que no tenga escenarios de uso diario, sino solamente escenarios de uso periódico, ya que por sus características implicará que no sea utilizada frecuentemente. Un ejemplo de este tipo de aplicaciones podría ser un programa para respaldos de un PC, que los usuarios utilizan cada uno o dos meses para respaldar su disco o un programa compactador de emails viejos.

Si consideramos a Word, que sí tiene un escenario de uso diario que roba el 99% del tiempo de trabajo, la función combinar correspondencia que mencionábamos es un buen ejemplo: para la mayoría aplastante de los

usuarios, no es una funcionalidad que utilicen más de una vez al mes. Inclusive es probable que para quién deba realizar mailings a diario, la funcionalidad que le brinda Word no sea lo suficientemente sofisticada: no está pensada como escenario de uso diario.

A diferencia de lo que sucede en los escenarios de uso diario, el uso esporádico hace que los usuarios se enfrenten a ellos cada vez como si fuera la primera. Se olvidan de lo que aprendieron la vez pasada y no tienen un incentivo especial para aprender esta vez, dado que saben que no volverán hasta dentro de mucho tiempo. Su foco es el de despachar la tarea lo más rápido posible y por lo tanto el tiempo en el que están en un escenario de este tipo no alcanzará para disfrutar de detalles y sutilezas.

Escenarios de uso necesario

Los escenarios de uso necesario tienen las mismas características que los de uso periódico, con una diferencia fundamental: no existen para satisfacer necesidades de los usuarios, sino necesidades de la aplicación. Su exponente canónico, más representativo, lo constituyen las pantallas de configuración.

La dificultad principal tiene su causa en que el modelo mental de nuestros usuarios jamás incluye a priori ninguna de las tareas que tendrá que desarrollar en un escenario de uso necesario. Esto se agrava por el hecho de que en la mayoría de los casos no tenemos otra solución que exponer en forma impúdica el modelo de implementación. En general, en las pantallas de configuración exponemos a los usuarios al sistema de archivos y a las características del equipo: memoria, disco, tarjeta de video, impresora, interrupciones, puertos y a otros detalles técnicos totalmente ajenos a los objetivos reales de nuestro reparto.

Hay dos reglas básicas con respecto a los escenarios de uso necesario. La primera dice que trate lisa y llanamente de eliminarlos. Tome decisiones valientes por sus usuarios. Programe para averiguar la información que necesita en vez de preguntársela. Brinde valores por omisión adecuados. Si no puede o no se anima a cumplir con esta regla, entonces aplique su versión light: el software debe funcionar razonablemente bien sin necesidad de pasar por ningún escenario de uso necesario. Si no consiguió convencer a sus colegas de eliminar esa pantalla de configuración, coloque unos valores por defecto tales que el usuario pueda usar su

aplicación sin hacer nada que no contribuya directamente a cumplir sus propios objetivos. Con el tiempo, los apologistas descubrirán la pantalla de configuración y podrán cambiar los parámetros. Si por el contrario, su aplicación abre un cuadro de diálogo de configuración antes de poder comenzar a usarla, donde el usuario debe decidir en qué carpeta colocar los archivos temporales, el directorio para las plantillas de estilo y si va a utilizar o no separadores para la impresión: ¡queme inmediatamente este libro!

Joel Spolsky[28], en su sitio www.joelonsoftware.com propone como el premio máximo a la estupidez informática el siguiente cuadro de diálogo:

A quién le importa si minimiza el tamaño de la base de datos o maximiza la capacidad de búsqueda, cuando además el cuadro ni siquiera tiene la delicadeza de informar, ya que estamos, cuántos megabytes de diferencia hay entre una opción y la otra. Como el programador quería garantizar que iba a ganar el premio propuesto, nos regaló otro cuadro de diálogo

---

[28] *Joel Spolski, User Interface Design for Programmers Chapter 3: Choices - año 2000. http://www.joelonsoftware.com/uibook/chapters/fog0000000059.html*

que no tiene desperdicio:

¡Bravo! ¡Excelente! Haga click en buscar para buscar, y en Finalizar para empezar. Recuerde siempre la primera regla: trate de eliminar los escenarios de uso necesario. Y si bien es cierto que estos dos cuadros de diálogo llevan el premio, hay que estar siempre alerta y no perder de vista el viejo refrán que reza "si ves las barbas de tu vecino arder, pon las tuyas en remojo".

La segunda regla dice: jamás mezcle los escenarios de uso necesario con los escenarios de uso diario o uso periódico. Ya que son un mal necesario, valga la redundancia, al menos manténgalos separados. Si proveyó valores por omisión adecuados, confíe en que la mayoría de los usuarios no visitarán jamás los escenarios de uso necesario. Si los mezcla con los que sí visitarán entonces expondrá inútilmente a sus usuarios a las complejidades de la implementación.

El sistema de archivos

Todo lo que implique interacción directa con el sistema de archivos será probablemente un escenario de uso necesario, salvo que se trate de una aplicación técnica para manejo del propio sistema de archivos y destinada

a usuarios técnicos. Si quiere que sus clientes pasen de ser sobrevivientes a personas realmente felices busque la forma de eliminar las opciones Abrir, Guardar, Guardar como, defragmentar, comprimir y en general todas las tareas vinculadas a la administración directa de carpetas y archivos. Las pruebas de usabilidad muestran que el sistema de archivos de las computadoras modernas, si bien es excelente desde el punto de vista técnico, es absolutamente inescrutable para los no apologistas. Conceptos como las estructuras jerárquicas profundas, el almacenamiento recursivo (carpetas que tienen carpetas), los sistemas de nombres y extensiones, las denominaciones de las unidades, y la diferencia entre almacenar en RAM y en un dispositivo magnético son absolutamente inescrutables para la gente común. No es una disquisición teórica o una opinión genérica, sino la constatación práctica de las pruebas de usabilidad de las aplicaciones, que lo reafirman una y otra vez.

Además de las decenas de propuestas de interfaces donde el sistema de archivos no muestra sus intimidades a todo el que se sienta frente al monitor, como por ejemplo ZoomWorld propuesta por Jef Raskin[29], existe una aplicación que utilizamos todos los días sin inconvenientes que resolvió este problema: el email. En una aplicación corriente de email no existen las opciones abrir, cerrar, guardar, guardar cómo, y los usuarios no se quejan de ello. Esta es la prueba más contundente de que es posible desarrollar aplicaciones que oculten a los usuarios las necesidades y complejidades del sistema de archivos de las computadoras en las que estas aplicaciones se ejecutan.

Las pantallas de configuración

Contra lo que creen los apologistas, la mayoría aplastante de los sobrevivientes del mundo informático no sólo no utiliza las pantallas de configuración, sino que no llega a entender jamás las sutilezas que hacen tan felices a los apologistas. Algunos ejemplos interesantes: MSN y Netscape reciben millones de hits diarios producto de que los usuarios no cambian la página default del navegador que utilizan[30]. O el navegador venía con el equipo, u otra persona se los instaló. No es que no sepan como cambiarlo, ni siquiera saben que se puede cambiar. Cambiarán el navegador cuando cambien el equipo, y con este cambio adoptarán la

[29] *Jef Raskin: The Huma interface: new directions for designing interactive systems - Addison Wesley 2000. Página 152*

[30] *Jakob Nielsen, Usabilidad, diseño de sitios Web, página 287*

nueva página default que el navegador traiga. Ejemplo dos: Yahoo!, Google y otros buscadores reciben por día cientos de miles de búsquedas de URLs completas[31]. Se trata decenas de miles de usuarios que acceden a los sitios Web poniendo www.amazon.com en la barra de búsqueda, para luego ir precisamente a www.amazon.com. No entendieron la lógica de las URL, y probablemente no la entiendan jamás mientras dispongan de un método tan eficiente como el que usan.

Los apologistas piensan que los escenarios de uso necesario son imprescindibles, pero esto no es cierto. En la medida en que cualquier industria madura, se van eliminando los escenarios de uso necesario de modo que los clientes se enfrenten a menos dificultades. Las primeras radios a válvula tenían numerosos escenarios de uso necesario: debían calentar, había que orientar permanentemente la antena, la sintonía estaba repartida en sintonía fina y sintonía gruesa. Las radios de hoy en día, inclusive muchas muy económicas, no solo funcionan sin demora apenas se conecta la alimentación, sino que ademas utilizan tecnología digital que hace que la función de sintonización haya desaparecido: solo hay que encenderlas, adecuar el volumen y seleccionar el programa preferido. El software debe seguir un proceso similar en el que los escenarios de uso necesario sean eliminados uno a uno de modo de separar definitivamente la tarea del chofer de la tarea del mecánico.

Los escenarios de caso límite

El resto de los escenarios no incluidos en las tres categorías anteriores representa la solución a situaciones límite a las que se enfrentará el sistema: qué sucede cuando se llena el disco, qué pasa cuando el nombre de un documento tiene más de 1024 caracteres, qué hacer si el modem no encuentra línea. Se trata de situaciones de borde, situaciones en las que el funcionamiento de la aplicación intenta rebasar la frontera de las capacidades y hay que tomar una decisión al respecto.

Desde el punto de vista de la programación estas situaciones son vitales, representan un trabajo enorme y una de las dos o tres características que diferencian la buena programación de la mala. Un programador profesional no solo tiene la capacidad de resolverlas, sino

[31] *Este es un ejemplo citado en forma recurrente. Como ejemplos: Don't make me think - Steve Krug, página 27 y el artículo sobre Google "Buscador de Crecimiento" de Keith H. Hammonds, Revista Gestión, Volumen 8 - Número 5 - Setiembre/Octubre de 2003, página 47.*

fundamentalmente de detectarlas, lo que es sin duda una habilidad prodigiosa. Encontrar en los requerimientos aquellas inconsistencias que harán tambalear la aplicación que estamos construyendo hasta tumbarla, es una atributo de mucho valor para los integrantes del equipo de programación.

Desde el punto de vista de los usuarios, estas situaciones tienen relevancia mínima mientras que las soluciones cumplan con dos requisitos: conserven el trabajo realizado hasta el momento y den la oportunidad de llamar a un técnico que resuelva el problema.

Los enfoques de desarrolladores y usuarios con respecto a los escenarios de caso límites son diametralmente opuestos. Supongamos que se llena el disco. Conservar el trabajo realizado hasta el momento por el usuario con el disco lleno requiere sin duda del programador una destreza y maestría considerable. Santiago, por el contrario, recibió el PC que utiliza en su oficina hace tres años y jamás tuvo el menor cuidado del espacio en disco. Cuando el procesador de texto dio el mensaje error: disco lleno, salvó el documento, cerró el editor, vació definitivamente la papelera de reciclaje y volvió a abrir editor y documento, como si nada hubiera pasado. Para Santiago, la situación límite fue intrascendente, ignora que quien programó el editor hizo maravillas para que cuando él apretara el botón guardar, el documento se guardara íntegro como todos los días, a pesar de que no había en el disco espacio suficiente para ello. El buen diseño debe separar estos enfoques, mostrando hacia afuera lo que Santiago espera encontrar, y permitir que hacia adentro el equipo de programación invierta el tiempo necesario para resolver en forma adecuada estas situaciones, de modo de construir aplicaciones estables y confiables.

Repartiendo el presupuesto de diseño

Siempre se podrían hacer mejor las cosas si fuéramos más, si hubiéramos tenido más tiempo y más dinero o si nos hubieran prestado más atención, pero la realidad es que nuestro trabajo será diseñar con el presupuesto que tenemos en el tiempo que tenemos, y no con el que hubiéramos deseado tener. Proponer metodologías que indican cómo hacer las cosas bien en la

Tierra de NuncaJamás no tiene utilidad práctica, siempre nos enfrentaremos a problemas concretos, con presupuestos concretos y otras restricciones también concretas. El desafío es proponer metodologías que optimicen el resultado en el marco de esas restricciones. Los escenarios cumplen ese rol en la metodología de Diseño de la Interacción.

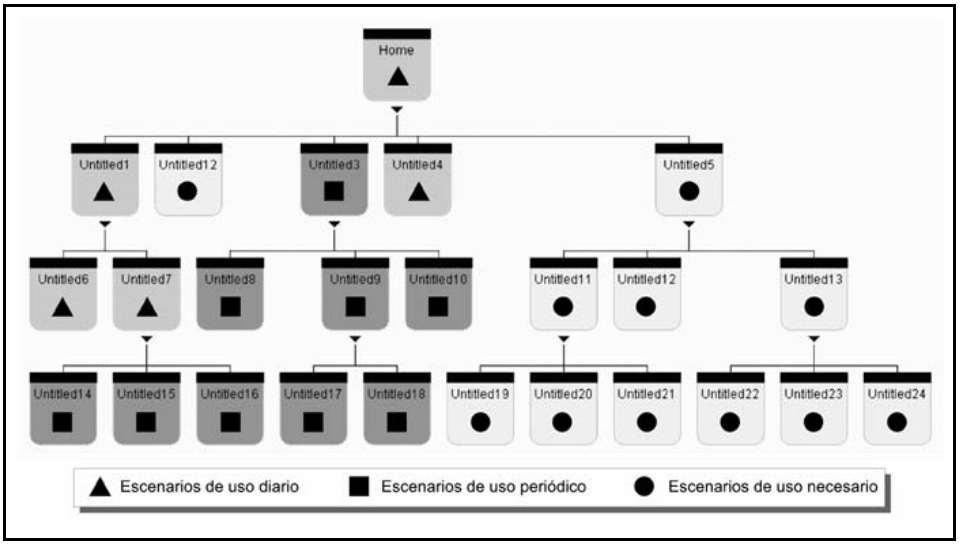
Suponga que usted tiene que diseñar un sitio que va a terminar teniendo 200 páginas. No importa cuál sea su presupuesto y el tiempo que disponga para diseñarlo, de alguna forma tendrá que dividir ese presupuesto entre las 200 páginas. Si no hace una decisión sensata y consciente antes de empezar a consumir el tiempo y el presupuesto, la vida tomará cruelmente la decisión por usted.

Cuando no se sigue la regla de diseñar primero y programar después, el tiempo de diseño es proporcional al esfuerzo de programación, algo que resulta natural ya que se diseña y se programa a la vez, y ambas tareas las desarrolla la misma persona, el programador. Sin duda se trata de un despropósito, que arroja un resultado nefasto, ejemplo de lo cual son las pantallas de configuración sofisticadísimas, llenas de botones, menúes descolgables y otras maravillas de las interfaces gráficas modernas. Las opciones de configuración llevan muchas veces una porción muy importante del tiempo destinado al desarrollo y el programador entiende entonces que vale la pena que la cara visible de ese trabajo tenga el maquillaje adecuado, algo que la mayoría de los usuarios probablemente no tengan siquiera la oportunidad de ver, ya que jamás abrirán la ventana de configuración.

La regla a seguir debe basarse en el siguiente criterio: 75% del esfuerzo para los escenarios de uso diario, 20% para los de uso periódico, 5% para los de uso necesario y nada para los de caso límite.

Sus futuros clientes, aquellos sobrevivientes que utilicen la aplicación, pasarán su vida en los escenarios de uso diario, donde cada detalle bien resuelto, cada atajo para aprender, cada información brindada a tiempo, los llenará de satisfacción. Por el contrario, cada omisión, cada contrariedad por más mínima que sea, terminará enojándolos enormemente. Cuando se abre el cuadro de diálogo para guardar el documento que estoy editando, quiero que esté exactamente en la carpeta que yo esperaba encontrar. Si todos los días, dos o tres veces por día, tengo que recorrer todo el sistema de archivos para encontrar la carpeta,

terminaré por molestarme. Pero si me pasa lo mismo con las carpetas donde almaceno los estilos, algo que probablemente utilice una vez o dos durante la vida útil de mi editor de texto, olvidaré rápidamente el percance. Un buen diseñador que cuenta con 8 hs de trabajo para resolver ambos problemas, debe dedicar 7 hs 50 minutos al primer problema y 10 minutos al segundo.



Coloreando las páginas a diseñar para un sitio Web,
se puede tener una definición clara de dónde y cómo priorizar
la aplicación de los recursos disponibles

Si los escenarios que definió no lo ayudan a dividir el tiempo disponible y el presupuesto de su equipo de diseño, o de usted mismo si es un equipo unipersonal, no están bien definidos. Si está diseñando un sitio Web, dibuje un recuadro para cada página que tendrá que diseñar, verifique que están todas, y pinte con un color distinto cada página según pertenezca a un escenario de uso diario, periódico, necesario o de caso límite. Verificará con sorpresa que va a dedicar el 70% del tiempo en el diseño del 10% de las páginas.

Manos a la obra

Si llegó hasta esta etapa del proceso del Diseño de la Interacción, está pronto para comenzar a trabajar en concreto sobre el diseño de las pantallas de su aplicación o páginas de su sitio. No está dentro de los objetivos de este libro abarcar ese tema. Hay decenas de libros excelentes y miles de libros pésimos al respecto, sobre todo si de diseño Web se trata. Dentro de los excelentes vale la pena nombrar otra vez About Face, de Alan Cooper en lo que se refiere a aplicaciones para PC, y Don't make me think, de Steve Krug junto con Usabilidad, diseño de sitios Web de Jakob Nielsen para el diseño de sitios.

Una vez completadas las tres etapas: personajes, objetivos y escenarios, su equipo de trabajo tendrá una visión de qué es lo que debe construir, y cómo debe estar organizado desde el punto de vista de quienes van a utilizar las aplicaciones. Queda ahora especificarlo para luego comenzar a programar. Construya su aplicación ciñéndose estrictamente a las definiciones que tomó y comprobará que los resultados compensan con creces el esfuerzo.

Capítulo 7

Cómo especificar el diseño de un sitio Web

La tarea de especificar la Interface de una aplicación basada en una GUI[32], si bien no se trata de una tarea fuera de nuestro alcance, tiene un nivel de complejidad importante. En contraposición, especificar un sitio Web, independientemente del tamaño del sitio, es relativamente sencillo. No estamos hablando de diseñarlo, sino de detallar de forma completa, comprensible y sin ambigüedades cómo es ese diseño una vez que lo terminamos. Dejaremos entonces a autores más experientes con propuestas completas y de calidad la tarea de especificar interfaces basadas en GUI, tal como Hugh Beyer y Karen Holtzblatt lo hacen en su libro Diseño Contextual y encararemos la segunda tarea, más adecuada a nuestras posibilidades y al alcance de este libro.

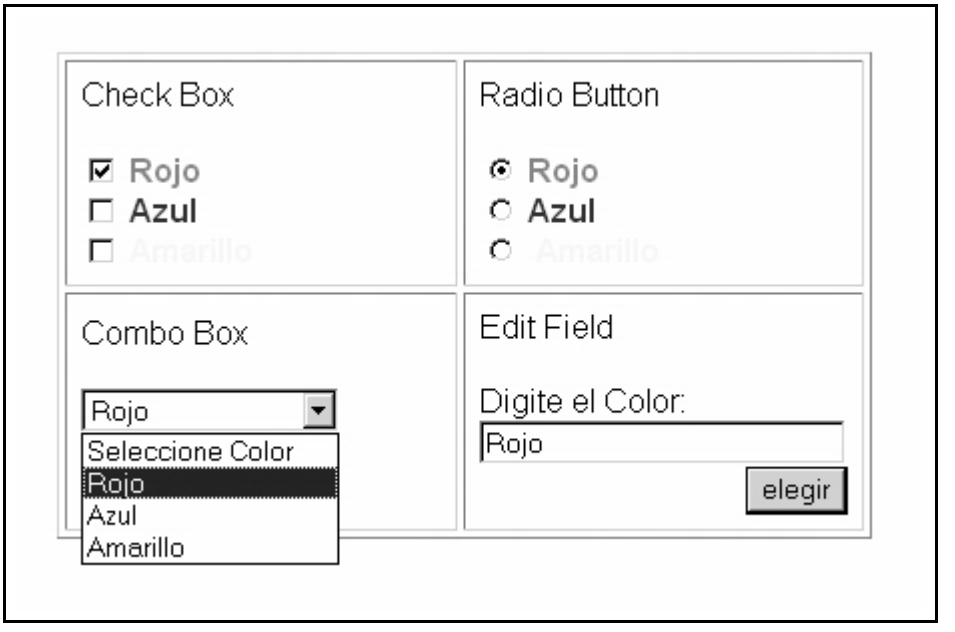
¿Qué es especificar una Interface?

Una vez completado para nuestro sitio Web el proceso de definición de nuestros personajes, descriptos sus objetivo, desglosados éstos objetivos en tareas que se agrupan en escenarios, estamos en condiciones de especificar cómo se presentará el sitio ante los ojos de los usuarios. Especificar la Interface es dar el paso siguiente y determinar unívocamente, sin ambigüedad alguna, cómo será cada uno de los elementos de la interacción del sitio.

Para especificar la Interface de su sitio Web es necesario conocer las opciones disponibles, lo que en la Web se reduce a las opciones que usted ve en los sitios que visita. Si no conoce los nombres científicos, alcanzará con una conversación de 15 minutos con un diseñador Web para aprenderlos. Si definimos que el color de una prenda del catálogo se seleccionará de una lista de opciones, ahora estaremos en condiciones para especificar si esta lista será un combo box, un check box, un edit field o un radio button, ya que llegó la hora de tomar esa decisión.

---

[32] *GUI - Graphic User Interface: Interface Gráfica de Usuario. Hoy en día, sinónimo de Interface basada en ventanas, creada en Xerox Parc, implementada en el sistema operativo de la Mac y omnipresente gracias a Windows.*



Distintas opciones de selección que provee el estándar HTML

A diferencia de sus primas GUI, las interfaces de los sitios Web son extremadamente rudimentarias. Derivan de este hecho consecuencias contradictorias. Por un lado, la Interface básica de una página Web presenta dificultades mínimas de aprendizaje y recordación. Un amigo decía que sus cursos de utilización de la Web duraban menos de 5 minutos: enseñaba a abrir el navegador, en el que configuraba Yahoo! como home page y daba las tres instrucciones básicas:

- ✓ Escriba acá (y señalaba con el dedo la barra de búsqueda) un texto relacionado con el material que quiera ver y luego clickee el botón.
- ✓ En las páginas sucesivas, si se pone la manito, entonces al hacer click va a ir a una página más o menos como dice el texto azul subrayado, si no se pone manito, mueva el Mouse hasta que se ponga.
- ✓ Terminó el curso!!!

Por otra parte, las páginas Web está a años luz de brindar las herramientas que proporcionan las aplicaciones tradicionales en un PC. Baste solo

como ejemplo que en una página Web la manipulación directa es prácticamente inexistente. Cambiar el ancho de una celda arrastrando uno de sus bordes es algo que hacemos todos los días en nuestras aplicaciones tradicionales y una maravilla tecnológica que todavía esperamos en una página Web.

Como no somos nosotros los encargados de saldar esta discusión, aprovechamos la escasez de herramientas de las páginas Web para transformarlas en una facilidad a la hora de resolver nuestro problema: hacer sencilla la tarea de especificar un sitio Web desde el punto de vista de la interacción, permitiendo que esta tarea sea desarrollada por el diseñador sin necesidad (o con necesidad mínima) de asistencia por parte de un programador.

Es cierto que existen tecnologías como JavaScript, ActiveX y aledaños, y que en la medida en que se desarrollan, la brecha comienza a cerrarse lentamente, muy lentamente. Pero a los efectos de especificar el sitio, alcanza con entender el funcionamiento de los elementos básicos provistos por la especificación del HTML.

¿Por qué especificar?

Para que el diseño esté terminado debe estar especificado: de lo contrario habrá perdido todo su tiempo. La especificación es el entregable del diseño y hará que todos los actores involucrados con el proceso de creación e implementación del sitio Web puedan generar expectativas válidas sobre qué esperar al final del camino. En particular, la especificación será el documento que los programadores utilizarán para la codificación del sistema. Los personajes, los objetivos y los escenarios son las herramientas de diseño, utilizando los cuales llegará a un diseño para su sitio Web. La especificación es el destino último de ese diseño, el plano sin el cual los constructores jamás podrían llevar a cabo la obra adecuadamente.

Sin una especificación los objetivos del diseño quedarán por el camino. El objetivo número uno, la definición del producto final, permitir visualizar cómo es el final del camino antes de emprenderlo se concreta con el documento de especificación, ésta es precisamente esa imagen de cómo

será el producto final cuando esté construido. La posibilidad de alcanzar los demás objetivos deriva naturalmente de su capacidad para definir el producto final, a excepción del cuarto, sacar la presión que el diseño implica para el equipo de programación. En este caso también es la especificación del diseño la clave, porque es este documento el que da las indicaciones concretas a cada programador de cómo tendrá que ser el comportamiento de las rutinas y programas que escriba para la aplicación, lo que constituye el punto de partida para su trabajo.

Un arma contundente

Podría ser que usted se maneje dentro de un equipo que funcione armoniosamente, y valore enormemente el rol del diseño, así como de la importancia de diseñar primero y programar después. En ese caso, cuando concluya con la especificación del diseño habrá concluido la parte fuerte de su tarea, y comenzará la parte fuerte de la tarea de programación, en la cual usted participará de las reuniones de avance y de la solución de los problemas puntuales que vayan surgiendo. Pero también podría ser, y apuesto diez a uno a que es, que usted tenga que demostrar en un proyecto tras otro, peleando con uñas y dientes, que es mucho mejor diseñar la interacción antes de programar, ante una turba de apologistas enardecidos que quieren sacarlo del medio y expertos en Marketing que no encuentran mejor forma de "acercar a las partes" que proponer que su metodología tan buena y creativa se aplique en el próximo proyecto porque en éste estamos apurados, lo que traducido al español antiguo quiere decir nunca.

En un proyecto que hoy está en el aire diseñamos y especificamos un sitio completo que daba soporte a una promoción en una tienda, página por página de la primera a la última, antes de hacer el llamado de precios para proveedores de la programación. Después de asignado el proveedor y de haber éste trabajado durante más de un mes, nos mostraron orgullosos los primeros avances del sitio funcionando: no tenía absolutamente nada que ver con nuestra especificación. La nueva interacción era tan mala que ni siquiera ellos eran capaces de manejarse con naturalidad, y juro que no exagero un épsilon. Había ejemplos paradigmáticos, como la incorporación del concepto de usuario. Nuestro diseño no manejaba usuarios, para simplificar la interacción y el soporte, dado que el sistema iba a tener un uso muy esporádico y nuestro personaje era único dentro de

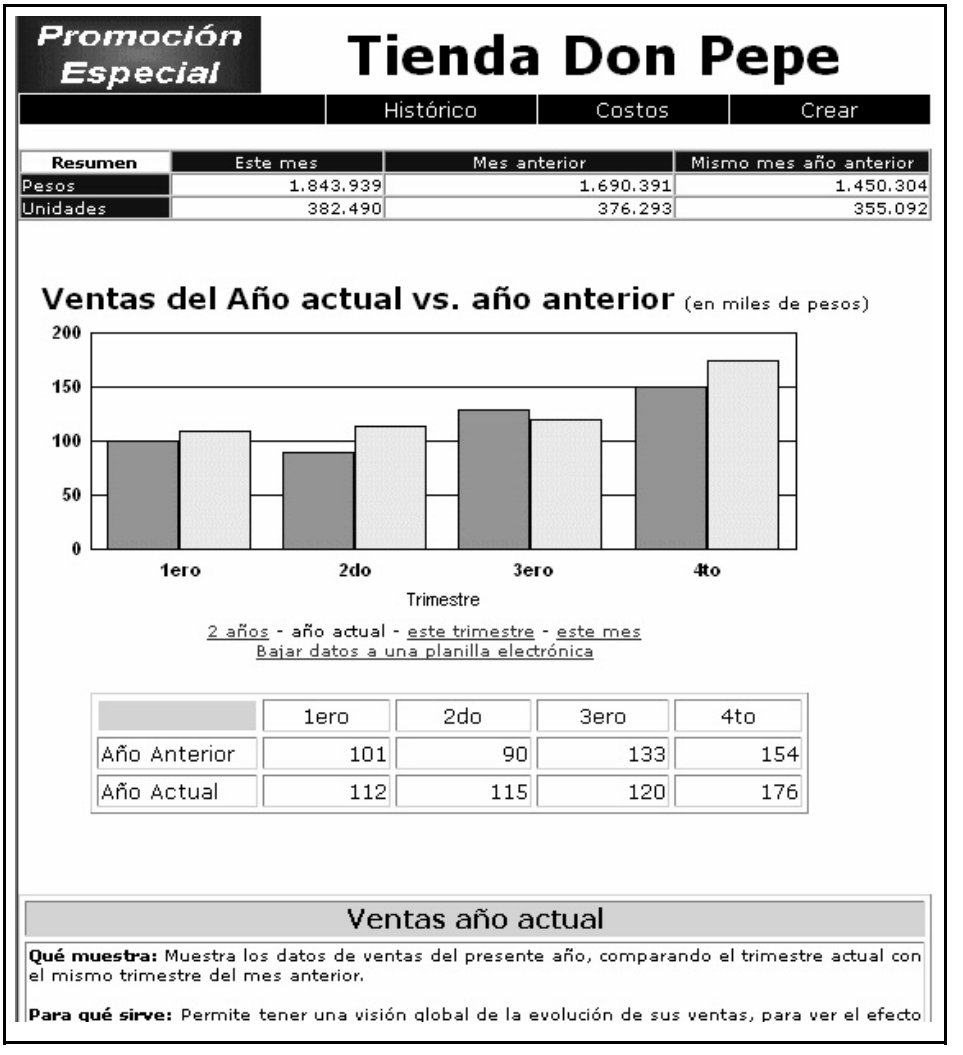
la tienda, por lo que lo usaba siempre él. Los programadores agregaron la creación, administración y eliminación de usuarios al sistema, algo que les es muy afín y natural, y no conseguían entender por qué nosotros queríamos eliminarla si en definitiva, ya estaba hecha y seguramente en alguna situación futura podría llegar a ser útil. Lo mismo para las promociones: nuestro sistema manejaba las promociones de a una, por lo que siempre se actuaba sobre la próxima promoción, simplificando la administración de promociones: de hecho no existía. Los programadores agregaron el manejo de infinitas promociones simultáneas e incorporaron amablemente al menú las opciones necesarias. Tampoco entendían porqué nos empecinábamos en eliminar un feature ahora que ya lo habían hecho.

La discusión con los programadores duró un rato hasta que el responsable del proyecto, sin titubear, asumió el compromiso de modificar la programación para cumplir con la especificación original. A pesar de que externamente quedó exactamente como lo especificamos, internamente conserva hasta el día de hoy problemas, producto de que las modificaciones no fueron una reescritura sino un parche al código ya hecho. Los usuarios desaparecieron de los menúes, pero no del código o del modelo de datos, por lo que cada tanto el sistema arroja un mensaje de error del tipo: "usuario invalido: usted no tiene permiso para acceder a tal función".

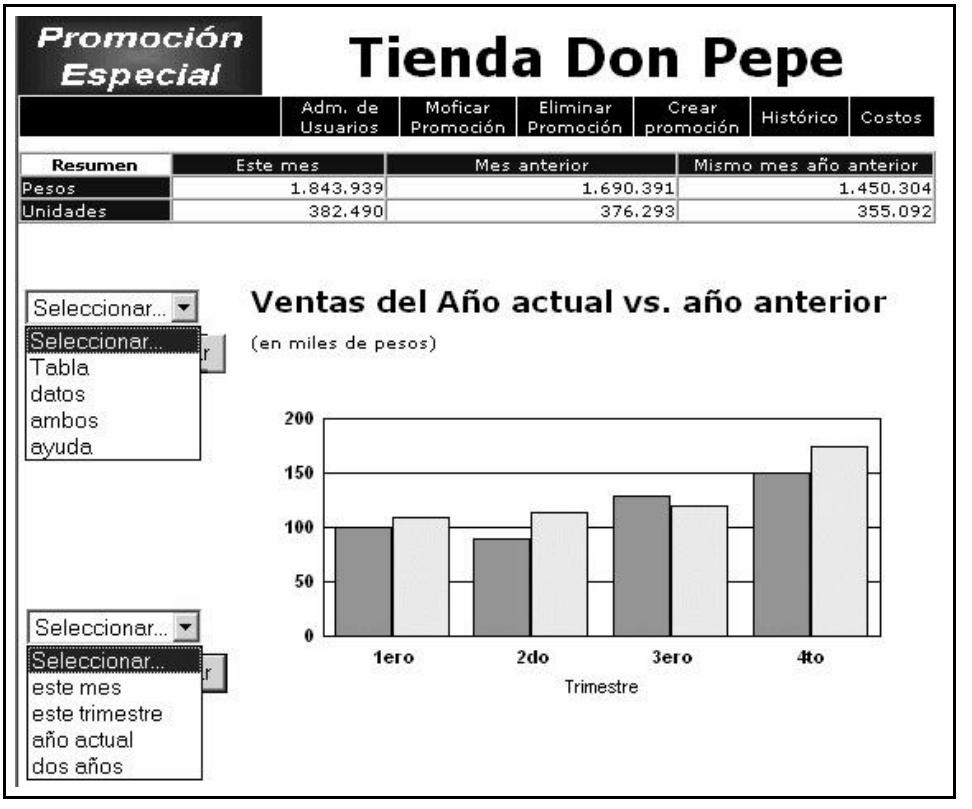
El ejemplo es ilustrativo. Sin una adecuada especificación del diseño jamás se hubiera podido demostrar que lo realizado era distinto que lo solicitado, porque lo uno y lo otro cumplían a cabalidad con la lista de features.

Los programadores entienden el diseño que usted les entrega como una sugerencia, un camino posible dentro de los que imaginan en su cabeza como viables. Su práctica de años les indica que mientras se programa se va poblando la Interface, y así intentará hacerlo una vez más. Cada brecha, cada resquicio que deje su especificación será utilizado sin piedad para salirse del camino y adaptar la interacción a la implementación, en la que el programador se siente naturalmente más cómodo. Si usted no cuenta con un documento escrito, con el que pueda demostrar sin lugar a la más mínima ambigüedad o duda que lo solicitado es distinto a lo que se intenta programar, el proyecto irá alejándose lentamente del diseño original hasta que queden en el resultado apenas unas trazas de lo que fue

en su momento el punto de partida. Recuerde que el código es como el concreto: una vez que está allí y fragua es para siempre, jamás se da marcha atrás. (¡A lo sumo se emparcha!)



Bosquejo de nuestra propuesta, nótese la Interface "todo en uno", con el sistema de ayuda incorporado a la propia página

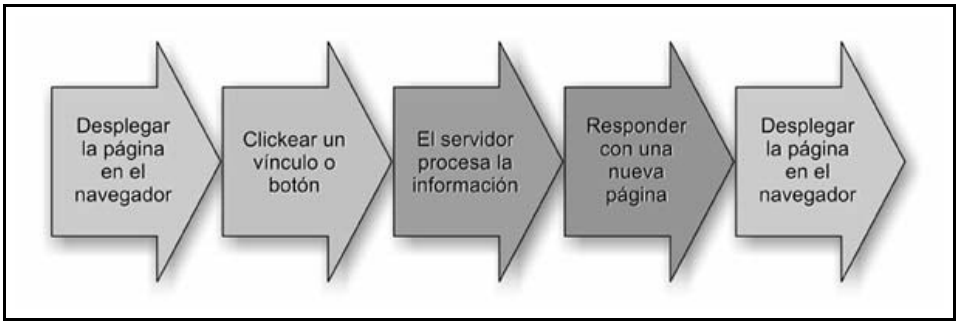


Bosquejo de la propuesta de los programadores, la Interface "todo en uno" desaparece y se incorporan dos menúes, con default en "seleccionar", que cuando no están desplegados (la imagen está forzada, pero en el navegador se ven comprimidos) no aportan ninguna pista sobre su contenido. En el menú principal aparecen nuevas opciones y nuevos conceptos, como los usuarios y su administración. En resumen, el diseño contiene estos y otros errores, por lo que es sensiblemente peor que el original

Considerando que el buen diseño de la interacción no recoge muchos elogios, la tarea a la que usted se enfrenta será más ardua aún. Para tener éxito, hay que estar convencido de la utilidad de la metodología y convertirse en un evangelista perseverante de sus virtudes y ventajas.

¿Cómo especificar el sitio?

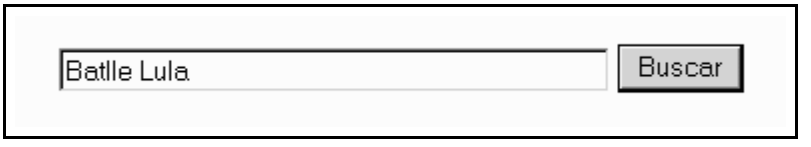
El funcionamiento de un sitio Web permite, de una forma económica, especificar el diseño de la interacción con mucha precisión. La clave está en que el funcionamiento de un sitio Web cumple un ciclo sencillo: la página se despliega dentro del navegador, el usuario clickea un vínculo o un botón, esa información se envía al servidor, que la procesa y responde con una nueva página que se desplegará dentro del navegador para que el ciclo vuelva a comenzar.



El procesamiento de la información en el servidor, para generar la nueva página que será enviada como respuesta, puede abarcar desde la tarea más elemental de cargar de una página previamente almacenada en el disco del propio servidor, hasta la ejecución de programas con niveles de complejidad muy elevados. Independientemente de la complejidad, en todos los casos esas acciones son invisibles para el usuario que está mirando la pantalla de su navegador, con la única excepción de la demora para cargar la página, y esta última en general es insignificante comparada con el tiempo de descarga producto de las demoras en las comunicaciones. Por lo tanto, especificaremos con total precisión el sitio si creamos las páginas tal como se verán en el navegador. Luego los programadores harán su tarea desarrollando la parte invisible que se ejecuta en el servidor, y los diseñadores gráficos culminarán la estética del sitio. Llamaremos al resultado de esta metodología Esqueleto Estático del sitio.

Si estamos diseñando la página de resultados de la búsqueda en el sitio, tal como se discutió en el capítulo 4, incluiremos el campo para el ingreso

de la o las palabras a buscar y el botón de búsqueda en una página, y luego la página de resultados. Ambas páginas tendrán un contenido fijo, cuyo objetivo es explicar cómo se interactúa con la búsqueda, no cómo debe implementarse ésta internamente. La programación que hay detrás de la búsqueda puede ser complejísima. De hecho el software para búsquedas es de las tecnologías en las que más esfuerzo se invierte hoy en día en la Web, con los casos de Google y Yahoo! a la cabeza y MSN corriéndolos de atrás. Los dos cuadros que se adjuntan, si bien no son aún una especificación terminada, dan una buena idea de cómo debe funcionar esa búsqueda.



Presidentes del Mercosur se reúnen en Montevideo (publicado **ayer**)
... en la reunión participaron el presidente de Uruguay, Jorge Batlle y el presidente de Brasil, Luis Inacio Lula da Silva

El Presidente Jorge Batlle habló sobre el Alca (publicado hace **2 días**)
... en referencia a las declaraciones del Presidente de Brasil, Lula da Silva, Jorge Batlle manifestó dudas sobre la viabilidad ...

El único requerimiento técnico para poder completar uno mismo el esqueleto estático del sitio es poder manejar mínimamente una herramienta para diseño de sitios, entre las que se cuentan NetObjects Fusion, Dream Weaver, Front Page y Netscape Composer entre otros. Su manejo básico es extremadamente sencillo y nuestra misión no es ni generar código HTML de calidad ni diseñar gráficamente la estética y el look and feel del sitio, sino simplemente establecer cuáles son los componentes que plasman la interacción de los futuros visitantes al sitio con el mismo.

El Esqueleto Estático

El Esqueleto Estático es una colección de páginas Web vinculadas entre sí que simulan el funcionamiento que tendrá el sitio una vez que esté terminado, definiendo a la vez, en forma unívoca, la interacción del sitio con sus visitantes.

Es estático porque no tiene ningún programa detrás, sino que las páginas preexisten siempre al visitante. Tal como se dijo, cuando un sitio está en el aire, muchas de las páginas son generadas en el momento (on the fly) para cada visitante a partir de información que está almacenada en la base de datos, información que el propio visitante ingresó y/o algún otro dato o parámetro recogido por el sistema. Esas páginas no existían cuando el visitante ingresó al sitio. En el esqueleto estático del sitio esta complejidad se omite, incluyendo un 100% de páginas estáticas. El objetivo es mostrar a cabalidad el comportamiento del sitio, por lo que el nivel de detalle, la cantidad de páginas construidas y la cantidad de ejemplos son una decisión más de diseño. En general, vale el criterio de que ante la duda, construyamos un ejemplo más.

De esta manera, cuando termine el proceso de diseño de la interacción, especificación incluida, podrá entregar al equipo de programación un conjunto de archivos que se verán como si fueran el sitio.

Podríamos decir que se trata de un prototipo del sitio, pero probablemente no es lo más conveniente. Los prototipos tienden a tener algunas propiedades que una especificación no debe tener. En primer lugar, no son completos. Implementan alguna parte del sistema, de la que se tienen más dudas, la que es más innovadora, de la que hay más discusión. El esqueleto estático del sitio debe contener el 100% de las páginas que serán programadas, eso forma parte de su esencia, independientemente de lo innovador, problemático o discutido que haya sido cada una de ellas. Si no están todas, el valor de la especificación se reduce en forma exponencial.

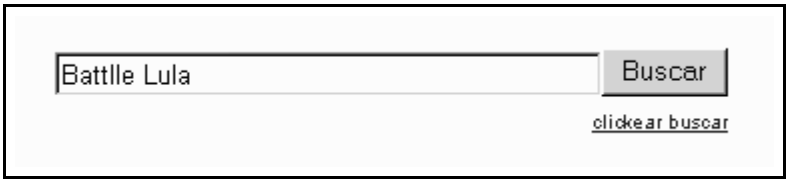
En segundo lugar, los prototipos dejan una inmensa cantidad de detalles para después, se concentran en las funciones básicas. El diseño de la interacción debe necesariamente contemplar todos los detalles, además de

las funciones básicas, por lo que su especificación debe plasmarlos uno por uno.

Por último, los prototipos exitosos tienden a ser usados como punto de partida para el desarrollo del propio producto. La especificación del diseño de la interacción no tiene programación alguna, no tiene cuidado en la calidad de la construcción y no debería ser incluida en el producto final, sino utilizada exclusivamente como guía para su desarrollo.

Botones y Links

Para dar "vida" al esqueleto estático, es necesario linkear todo lo linkeable. Esto a veces es posible y a veces no dado que, por ejemplo, para que un formulario funcione adecuadamente se necesita un programa que lo atienda en el servidor[33]. Agregue en esos casos links comunes, que simulen el funcionamiento del botón, y que conduzcan al navegar a la página ejemplo del resultado del formulario. Este criterio se extiende a otras situaciones, como por ejemplo el caso de algunos menúes, que resultaría muy complicado construir completamente.



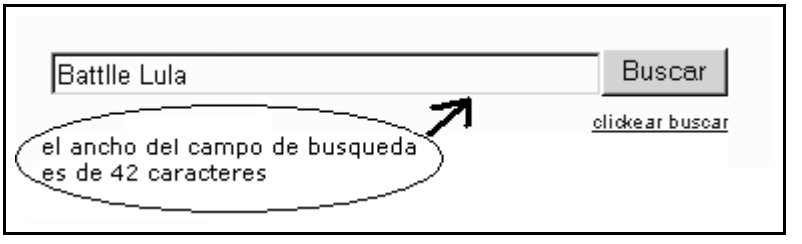
Información complementaria

Junto con las páginas que conforman el Esqueleto Estático del Sitio, es necesario aportar una importante cantidad de información complementaria, que incluye todo tipo de atributos y detalles del comportamiento: entran en esta categoría desde los valores default hasta los pixeles que deben separar dos imágenes y las cosas que queremos que el sitio NO haga. No omita jamás ninguna de estas piezas de

[33] *Esto no es técnicamente exacto, ya que hay formas de hacer funcionar un formulario en un sitio estático y desconectado, pero en general los programas para el diseño de páginas no lo contemplan, y aprender este tipo de trucos técnicos no agrega nada al proceso de diseño y especificación.*

información, y utilice todas las herramientas a su alcance para incluirlas en el propio sitio. Van acá algunas opciones

- ✓ *Capturar las pantallas y escribirles arriba:* junto con una página que requiere comentarios, adjuntar otra que sea una imagen de la pantalla capturada. Alcanza con un manejo rudimentario de Paint para esta tarea. La página con los comentarios se vincula abajo de la original con un link del tipo : Atención: tener en cuenta los comentarios sobre esta página



- ✓ *Comentarios al margen*: Utilizar todo tipo de comentarios al margen en la propia página, ubicando en áreas que no molesten textos, imágenes, tablas, links a archivos y todo lo que sea necesario para entender sin ambigüedades qué es lo que la página indica y qué es lo que no indica.
- ✓ *Páginas específicas para comentarios:* no dude en incluir todas las páginas adicionales que crea convenientes para los comentarios, datos, indicaciones y especificaciones que estime necesarias. Nunca estarán de más.

El trabajo intensivo que realizará para diseñar la interacción del sitio y luego especificarla, generarán un conjunto completo de pre-supuestos y sobreentendidos que desaparecen a la hora en que los programadores miran el esqueleto estático para programar las funciones que le darán vida. Todos estos pre-supuestos y sobreentendidos deben ser detectados e incluidos como comentarios y aclaraciones al propio esqueleto.

El esqueleto estático y el diseño gráfico

El diseño gráfico tiene sin duda relevancia para la interacción, no solo porque hace que el sitio sea visualmente agradable, sino porque los

colores y elementos gráficos agrupan, explican, indican, dan pistas, forman parte integral e indivisible de la interacción del sitio. A pesar de esto, son minoría los casos donde es imposible especificar la interacción del sitio sin definir completamente el diseño gráfico.

Existen casos donde la gráfica será parte indisoluble de la propia esencia del sitio, y es necesario tomarla en cuenta inclusive antes de plantearse la problemática de la interacción. Si estamos desarrollando un sitio de música tecno, la gráfica jugará un rol preponderante, porque la propia música tecno está asociada a una estética particular: si el sitio no consigue que sus visitantes lo vean dentro de esa estética, fracasará. Si por el contrario estamos desarrollando el sitio de una empresa, lo más probable es que sus lineamientos de comunicación trabajen sobre los logotipos, slogans, tipos de letra y colores, tal vez la determinación de un estilo más formal, más informal o en el medio. En el caso de enfrentarnos a un sitio donde el arte no forma parte integral de la propuesta, si fuera necesario podríamos especificar el 100% del sitio indicando con imágenes "de relleno" dónde van los logos, las fotos de los productos, etc.

Siempre es mejor poder contar con un diseñador gráfico al especificar la interacción del sitio, pero esto no es posible en todos los casos, ya sea por presupuesto o porque usted decidió que si no llama al diseñador, tendrá un contendiente menos para discutir. En esos casos, salvo que se encuentre frente a un sitio muy particular, avance sin miedo hasta el final de la especificación, creando el esqueleto estático del sitio con placebos gráficos e incluya todos los comentarios necesarios sobre estos elementos, en lo que tiene que ver con su comportamiento en el sitio (¿deben moverse? ¿cuántos colores tienen? ¿tienen sentido o son abstractos?) así como el rol que deben desempeñar.

Especificaciones Light

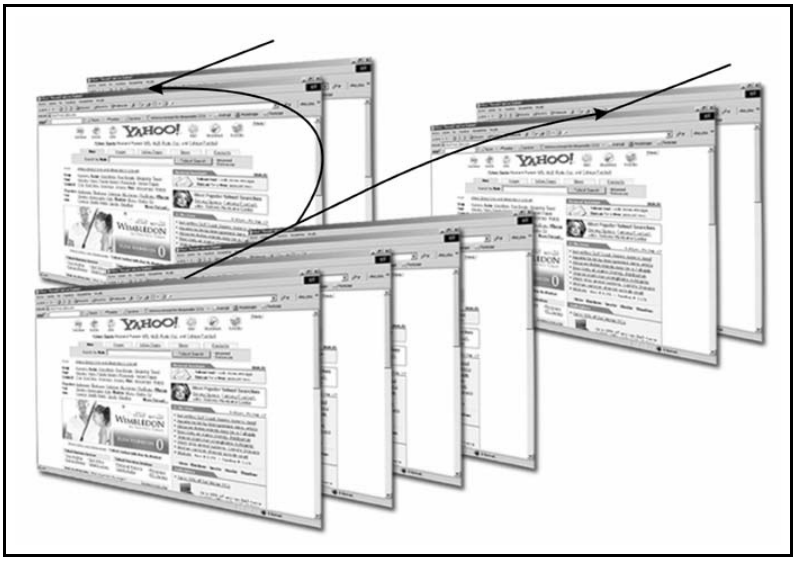
Si su habilidad para lidiar con herramientas para generar páginas Web es limitada, existen alternativas menos precisas pero más sencillas para especificar un sitio. En definitiva, si concebimos un continuo que tiene en un extremo una especificación completa, de calidad y en el otro un sitio Web completamente carente de especificación, cualquier técnica que mueva su proyecto hacia la especificación completa alejándolo del otro

extremo, aportará valor, reduciendo la incertidumbre y aumentando la eficacia de su inversión en diseño de la interacción.

StoryBoard

La técnica de StoryBoard es prestada del ámbito de la publicidad televisiva y el cine. Se trata de crear imágenes que muestran en secuencia como se verán las distintas pantallas del sitio.

Las pantallas se pueden dibujar con cualquier herramienta, sea ésta de dibujo como Corel y Paint, software para presentaciones como PowerPoint y Freelance Graphics o cualquier software con el que usted tenga habilidad suficiente para dibujar las pantallas. Una estudiante de arquitectura hizo sus primeras armas en diseño de la interacción especificando las páginas en Excel. Y en el caso de que no se sienta confiado con ninguna herramienta informática, entonces ponga un viejo y querido lápiz en su mano y saque unas hojas de papel blanco de la bandeja de la impresora.



En un StoryBoard se dibuja cada pantalla en una lámina, de modo de simular el comportamiento del sitio

El StoryBoard tiene algunas ventajas: es mucho más rápido de crear y sobre todo mucho más rápido de modificar, inclusive sobre la marcha en medio de una reunión. Es más fácil crear varias versiones distintas de la

misma página, para estudiar cuál de ellas tiene una interacción mejor. También tiene contras: la primera es que es menos precisa que el esqueleto estático, por la propia naturaleza de láminas fijas, con la que es más difícil representar interacción. Un segundo problema es la naturaleza lineal del StoryBoard, con la que es difícil mostrar la navegación aleatoria y sin un camino prefijado propia del hipertexto.

Tal como en el caso del Esqueleto Estático, todo lo que no surja en forma exacta de las láminas del StoryBoard, debe ser especificado aparte en anotaciones al margen, texto, imágenes, o el medio que crea conveniente.

No deje nada librado al azar

La especificación culmina el proceso de diseño de la interacción eliminando las incertidumbres e indefiniciones de la Interface entre los futuros usuarios y la aplicación. El Esqueleto Estático y el StoryBoard son dos técnicas válidas y probadas para especificar un sitio Web, pero no son las únicas. De hecho, puede haber otras técnicas que usted considere mejores o más útiles en cada caso, inclusive pueden mezclarse las técnicas, si la situación lo amerita.

Independientemente de la técnica que elija, cuide de no dejar ningún aspecto del diseño de la interacción del sitio librado al azar. La utilidad del diseño depende en gran medida de su completitud, de la capacidad de abarcar desde las líneas fundamentales hasta los detalles menores. La diferencia entre un gran diseño y un pésimo diseño no está en los trazos fundamentales sino en los detalles. Las características principales colocan a un producto dentro o fuera de la categoría, pero no son las que lo hacen un gran producto. Tal como vimos en el capítulo 4, en el fondo todos los buscadores buscan y encuentran miles de sitios para mis palabras de búsqueda. Lo que hace grande un buscador como Google son apenas unos segundos menos en la búsqueda y un orden mejor que el del resto en los resultados. La diferencia en el orden no es una diferencia aplastante, es una diferencia sutil, pero que hace, valga la redundancia, una diferencia aplastante en la satisfacción de los navegantes de la Red. La especificación del diseño debe llegar en forma capilar hasta cada uno de los detalles para construir esa diferencia.

Capítulo 8

Los usuarios usando

La usabilidad

Las personas utilizan los programas, las aplicaciones y los sitios Web de una forma muy distinta de la que los que creamos esas aplicaciones esperamos. Si pudiéramos ver con sus ojos la pantalla del monitor, sentir como mueven sus manos y leer a la vez sus pensamientos, constataríamos la distancia asombrosa entre lo que nosotros esperábamos que ellos pensarían mientras hacíamos el programa y lo que en realidad perciben cuando lo usan efectivamente. Precisamente éste es el desafío de la Usabilidad: entender cómo ven el software quienes lo utilizan, qué piensan cuando lo usan, qué pistas le otorgan a los usuarios las respuestas que el software les da, entender por qué toman sus decisiones cuando se deciden por una u otra opción.

En términos de los conceptos manejados en este libro, la Usabilidad tiene como objetivo analizar el nivel de coincidencia que el modelo de representación de una aplicación o sitio Web expresa a través de su Interface, con el modelo mental del usuario que está sentado frente a ella. Cuando un aplicativo se muestra hacia afuera exactamente como el usuario espera encontrarlo, el modelo de representación y el modelo mental son idénticos y por tanto la dificultad de uso será nula: ese es el objetivo teórico a alcanzar. En la práctica, es necesario testear para encontrar las divergencias entre el modelo de representación que efectivamente construimos al diseñar la interacción de la aplicación y el modelo mental que los usuarios construyen al utilizarla, producto de su experiencia anterior y la propia experiencia que adquieren al usar el aplicativo.

La intención es entonces buscar los caminos para capturar las acciones, actitudes y pensamientos de los usuarios usando el software. Esto puede parecer complicado, y en realidad puede llegar a serlo. En algunos casos será necesario desarrollar prototipos, aplicaciones que capturen las teclas y clicks del Mouse, midan tiempos, filmar los movimientos del usuario y analizar qué parte de la pantalla miran sus ojos en cada momento. Como contrapartida, en la mayoría aplastante de las situaciones, y en mayor medida aún si se trata de sitios Web, alcanza para conseguir los primeros resultados con pedirle a alguien que no participó en la cocina del proyecto

que mientras lo observamos realice dos o tres tareas con nuestra aplicación a la vez que repite en voz alta lo que va pensando. ¡Manos a la obra ahora mismo! Salga al pasillo, tome al primero que pasa, siéntelo en su escritorio y pídale que busque en el sitio de la empresa el precio de un repuesto para el producto XYZ. (... 5 minutos para la prueba ...) ¡Felicitaciones! ya recibió su bautismo en Usabilidad.

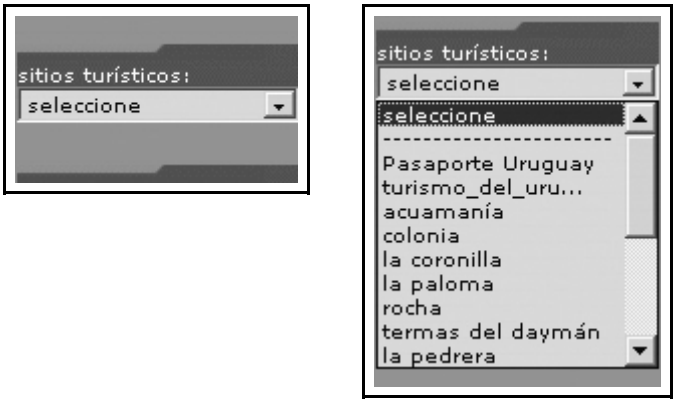
No es que las pruebas de usabilidad en realidad se realicen así nomás y que cualquier cosa puede ser considerado una prueba de usabilidad decente. El punto es que en general se dan dos elementos complementarios que hacen que inclusive una prueba de usabilidad mínima tenga resultados formidables: por una parte las aplicaciones cuya interacción no fue diseñada sistemáticamente suelen tener errores garrafales de usabilidad, por otra los equipos de trabajo están tan ligados a las aplicaciones que son absolutamente miopes para detectar esos errores. Es entonces que una prueba mínima, hecha realmente con el primer sobreviviente que encontramos en el pasillo, nos desasna de repente de algo terrible y obvio que estuvo plantado frente a nuestras narices en el medio de la pantalla durante los últimos dieciocho meses sin que fuéramos capaces de darnos cuenta.

Qué esperar de la Usabilidad

La Usabilidad produce habitualmente dos entregables. El primero es el análisis de una aplicación concreta, tratando de encontrar las debilidades que complican a los usuarios, que los alejan de sus objetivos cuando interactúan con ella.

Si visitan www.turismo.gub.uy, se encontrarán con un menú titulado *sitios turísticos*; las pruebas de usabilidad muestran de forma inequívoca que los usuarios entienden que ese menú contiene destinos turísticos, pero en realidad contiene sitios Web turísticos. Al observar a los usuarios navegando y escuchar sus razonamientos se constata inmediatamente que muchos de ellos no comprenden hacia dónde los llevan las opciones que seleccionaron en ese menú, inclusive después de un largo rato de ir para atrás y para adelante. Si el Ministerio de Turismo hubiera invertido una hora/hombre de su presupuesto en una prueba de usabilidad, hubiera descubierto rápidamente este problema y hallado una corrección sencilla,

eliminando la necesidad de que sus visitantes gasten dos o tres balas de su magazine en esta tontería.



Plegado Desplegado

El menú descolgable tal como se ve en el sitio del Ministerio de Turismo

El segundo entregable es la sistematización de todo lo aprendido en las distintas pruebas de usabilidad, para encontrar patrones de problemas, guías de usabilidad basadas en el análisis empírico de las distintas situaciones. Un ejemplo clásico de este segundo entregable es el análisis de las pantallas de bienvenida, las animaciones que aparecen antes de mostrar la home page de un sitio. Se pueden encontrar decenas de disquisiciones teóricas sobre su utilidad, conveniencia y atractivo. Inclusive se pueden leer sesudas argumentaciones a favor y en contra de la opción "saltar presentación". Las pruebas de usabilidad[34] muestran de forma contundente que la mayoría aplastante de los usuarios detestan las presentaciones. No se trata simplemente de que no surten el efecto esperado, o que no llaman la atención, o de que no son leídas, se trata de que las odian, producen rechazo, los visitantes se enfurecen con ellas y con el sitio que las muestra. Además se puede constatar que el nivel de furia aumenta en las sucesivas veces que se ingresa al sitio y aumenta exponencialmente cuando la navegación lleva al visitante a la home page, y en vez de la home page vuelve a aparecer otra vez la maldita pantalla de

[34] *Existen numerosas pruebas sobre las pantallas de bienvenida, una de ellas es la de Jakob Nielsen en Usabilidad, Diseño de Sitios Web, ibídem página 176.*

bienvenida. Esta verificación práctica deriva en una guía de usabilidad: elimine las páginas de bienvenida, siempre.

La vida es la que manda

Ni los resultados del análisis de un sitio, ni las guías de usabilidad resuelven problema alguno por sí solas. Los test de usabilidad no prueban una tesis a partir de una hipótesis como en un teorema, sino que constituyen un insumo vital, que sumado a la experiencia, conocimientos y sentido común del diseñador, le permitirán perfeccionar el comportamiento interactivo de una aplicación. Rara vez una prueba de usabilidad permite decidir si la opción A es definitivamente mejor que la opción B, y en general en esos casos un diseñador razonablemente experiente es capaz de prever el resultado de la prueba sin realizarla. Las pruebas de usabilidad pondrán al descubierto las virtudes y defectos tanto para la opción A como para la opción B, y será a partir de ello que el diseñador proponga una solución al problema que muy probablemente no será ni la A ni la B que fueron punto de partida. Inclusive las guías de usabilidad más universales, como la de eliminar las pantallas de bienvenida tienen excepciones y deben ser utilizadas con sentido común. Es razonable que una página para adultos, ya sea de sexo o con fotos muy fuertes sobre una guerra, contenga una portada aséptica que filtre a los usuarios, advirtiéndoles del contenido del sitio que hay detrás de ella.

Corolario: un cambio de actitud

Las pruebas de usabilidad sensibilizarán de forma rápida y contundente al equipo de trabajo frente a las necesidades y problemas de los usuarios reales. En los equipos de desarrollo que jamás realizaron pruebas de usabilidad, la distancia entre lo que los programadores piensan de los usuarios y lo que los usuarios piensan de los programadores es tan inconmensurable, tan dispar, que el impacto de las primeras pruebas de usabilidad deviene en un shock para la actitud de todo el equipo.

En una oportunidad discutimos largamente con un programador sobre la necesidad de modificar un programa, con una Interface de texto tipo Telnet, que entre otros problemas serios incluía la siguiente "ayuda":

Se trataba de una de esas largas, tediosas e inútiles discusiones sobre lo que piensan los usuarios, de su obligación eternamente incumplida de leer las ayudas y los manuales, y todos los etcéteras del caso. Una prueba de usabilidad de 5 minutos demostró que ninguno de los 5 usuarios que conseguimos dentro de la empresa pasó con éxito la utilización de la pantalla que incluía las opciones derecha/izquierda, a pesar de que 3 de ellos tenían larga experiencia en la utilización de la aplicación. Miraban la pantalla, pensaban, y siempre erraban. El programador se mordía la lengua para no insultar a sus compañeros de trabajo. Usted podrá decir que es un ejemplo demasiado trivial y yo coincido, sin embargo es real y la prueba de usabilidad saldó definitivamente la discusión. Los programadores son gente inteligente, desarrollan una tarea extremadamente exigente, que les requiere muchas veces jornadas muy largas de trabajo bajo la presión de cronogramas y fechas límites. No se empecinan en mantener interfaces pobres porque sí, sino porque en el acierto o en el error están absolutamente convencidos que son las más adecuadas. Las pruebas de usabilidad son uno de los pocos caminos para revertir estas opiniones cuando están equivocadas y probablemente sean la de mejor relación costo/beneficio.

El impacto que provoca ver a un usuario sentado sufriendo con nuestro programa, sin acertar en sus intentos por manejar las distintas herramientas y opciones que le ofrecemos en la Interface, nos producirá angustia y unas fuertes ganas de gritarle "¡Tarado! no ves que el botón buscar está ahí al lado", "¡Estúpido! no te das cuenta que es el otro menú y no ese, ¡pero hay que ser nabo realmente!". Se trata de una sensación desesperante. Las pruebas de usabilidad no solo convencen: exasperan, irritan, enfurecen. En los cursos en la Universidad, siempre pido a los alumnos que hagan una prueba de usabilidad breve y la filmen, para después analizar los resultados en clase. En general la dificultad para el usuario de la prueba no pasa de encontrar tal o cual cosa en un sitio Web dado. Más de la mitad de los grupos no son capaces de abstenerse de interrumpir al usuario real de la prueba y decirle: "por ahí no", "es el otro botón" y en este caso ellos no tienen vínculo emocional alguno con el

sitio que están testeando, no fueron ellos quienes lo construyeron. Nunca faltan los usuarios que siquiera pueden poner la URL del sitio, como en el caso de una señora que no conseguía ingresar a www.uruguaynatural.com. Puede parecer ridículo, pero es tan solo la más cruda y pura realidad. Están filmados éste y muchos otros casos. La señora estuvo 5 minutos peleando: cuando lo escribía bien, agregaba .uy al final e iba a dar a otro lado, cuando sacaba el .uy, borraba algo más y obtenía una página de error. Así una vez y la siguiente también. Una de las alumnas que hacían el trabajo era su hija y la otra filmaba alternativamente a la madre y a la hija: la primera yendo para adelante y para atrás, con una sensación de impotencia y ridículo, la segunda furiosa con su madre, hasta que lanzó un "ta, basta, mamá sos una pelo...." y se cortó la filmación. Cuando se enteraron que no habían borrado la cinta se querían morir, pero su trabajo tenía un valor enorme, ya que no hay nada más ilustrativo: una persona adulta, educada y con la mejor disposición no es capaz de desentrañar los misterios de las estructuras jerárquicas de los nombres de la Web. Apenas los sobrevive a los tumbos. Los diseñadores gráficos y programadores que pusieron el sitio en el aire jamás se imaginaron siquiera que el problema existe. El modelo mental de la señora dice: sitio uruguayo termina en .uy. Uruguay Natural es uruguayo, entonces termina en .uy. Si hay una revista y un sitio que se pelean por el nombre, lo que lleva a que la revista sea .uy y el otro sitio .com a secas es ajeno a su modelo mental. La señora descargó todo el magazine de balas y no había rasguñado aún al primer enemigo: las balas no le alcanzaron para ver siquiera la home del sitio.

En la misma prueba hubieron varios casos ilustrativos más: un señor cometió el mismo error, agregó el .uy y realizó toda la tarea en otro sitio obviamente sin éxito, pero sin darse cuenta que ese no era el sitio de Uruguay Natural, sino de la revista homónima. Una señora, novata en las lides de la computación, estuvo 10 minutos clickeando el botón derecho del Mouse en vez del izquierdo: no lo notó ella y tampoco los alumnos del grupo. Un muchacho de unos 22 años, se sentó, saludó a las cámaras vanidoso, y emprendió la tarea convencido que en uno o dos minutos saldría con toda la gloria: estuvo más de media hora yendo y viniendo, enterrándose en una rama del sitio primero y luego en la siguiente, sin encontrar lo que estaba aparentemente "en sus narices". El 40% de los

usuarios no consiguió sobrevivir al intento: desistió sin poder encontrar en el sitio de Uruguay Natural la dirección de un hotel en La Paloma.

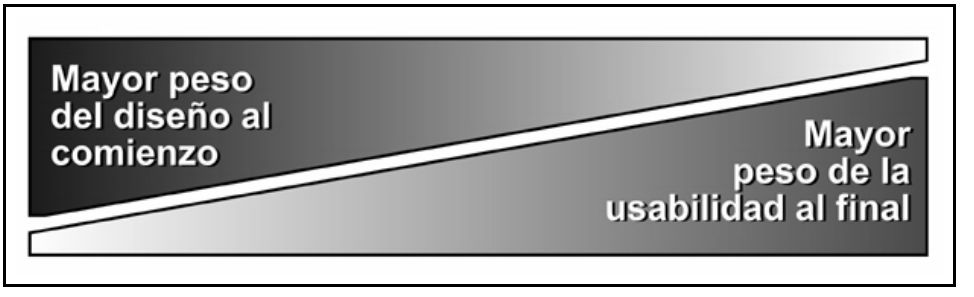
Mientras los que hacemos los sitios discutimos sobre el rollover de los iconos y si es más conveniente utilizar los botones por defecto del navegador o sustituirlos por imágenes, las pruebas de usabilidad nos muestran que para muchos usuarios, inclusive aquellas cosas de las que nosotros ni nos preocupamos, que damos por hechas, por obvias, son una carga demasiado pesada de llevar que llegan a transformarse en un calvario para los usuarios con crucifixión incluida. Si su equipo de desarrollo no se sensibiliza viendo estas situaciones con sus propios ojos, entonces tienen el corazón de piedra.

Usabilidad y diseño de la interacción

Los expertos en usabilidad y los diseñadores de la interacción mantienen en sus trabajos una disputa sobre cuál de las dos disciplinas es la realmente importante, algo así como la eterna disputa entre arquitectos e ingenieros civiles. Las disciplinas tienen foco en problemáticas distintas, y por lo tanto su utilización y conveniencia son distintas.

La usabilidad es la prueba de fuego de cualquier proyecto. Nada más crudo y conveniente que testear los sitios y aplicaciones con usuarios reales, de carne y hueso, en el entorno de uso real o lo más parecido posible a éste. Allí se encuentran los problemas grandes, medianos y pequeños con los que los usuarios se enfrentarán cuando el producto esté en la calle. Allí se pueden testear opciones, analizar posibles soluciones, valorar ventajas y desventajas. Ningún diseñador, por más experiente que sea puede sustituir las pruebas de usabilidad. Es más, la rutina de diseñar y testear a veces nos vuelve perezosos, y comenzamos a creernos que no es tan necesario testear, o por lo menos que no es tan imprescindible hacerlo tan seguido. Y allí ¡pum! nos chocamos una vez más contra la realidad real, terca, empecinada, que se obstina en mostrarnos una y otra vez que por más experiencia, conocimientos y teoría, siempre cometemos decenas de errores que jamás hubiéramos detectado sin las pruebas de usabilidad.

Pero para hacer pruebas de usabilidad se necesita primero tener algo para probar. No hay usabilidad que valga cuando la hoja está en blanco. El diseño de la interacción es la herramienta para construir un sitio o una aplicación desde cero. La usabilidad testeará lo que el diseño construyó primero. Y las distintas soluciones a los problemas son producto del diseño, la usabilidad encuentra los problemas pero no los resuelve, esta es tarea del diseño de la interacción.



Peso relativo del diseño y de la usabilidad,
según avanza un proyecto

Lo razonable es mantener un equilibrio entre ambas disciplinas: el diseño pesa fuertemente en el comienzo, donde ya se pueden comenzar a testear los primeros materiales, los primeros storyboards, las primeras partes de los esqueletos de un sitio. A medida que el proceso avanza, la interacción comienza a tomar forma, a concretarse y se hace más intensivo el proceso para pulir detalles, encontrar hasta el más pequeño defecto, agregar sutileza y sofisticación a los escenarios de uso diario, lo que hace tomar más peso a la usabilidad.

Probar, probar y volver a probar

El proceso de diseño es un proceso iterativo: diseñar, testear, rediseñar, volver a testear, ampliar el diseño, testear lo que agregamos, hasta el final, y después también. Aquí lleva ventaja el diseño de la interacción frente a otras disciplinas constructivas: es mucho más fácil y económico probar, testear, confirmar la usabilidad de cada una de nuestras decisiones de diseño para una aplicación basada en software que testear las

decisiones que toma un ingeniero para diseñar un puente o las de un arquitecto para un edificio de diez pisos.

En el caso de los sitios Web, la facilidad para especificar el sitio final en un esqueleto estático, permite utilizarlo de modo que los test de usabilidad se realicen en situaciones prácticamente equivalentes a las condiciones de uso reales, sin esfuerzo adicional de programación. En el caso de aplicaciones tradicionales, tal vez sea necesario trabajar un poco para poder realizar las pruebas, pero este pequeño esfuerzo adicional vale la pena. Inclusive los test de usabilidad en base a StoryBoards son extremadamente productivos, ya que para preguntarle a un usuario qué espera que suceda cuando clickee un botón no requiere más que un dibujo de la pantalla. Testear la usabilidad del diseño primero y de la aplicación después es un proceso continuo, no de grandes hitos sino de pequeños incrementos.

Al respecto de los test de usabilidad, valen algunos comentarios de Steve Krug[35]

- ✓ *Testear un usuario es 100% mejor que no probar ninguno*: Testear siempre funciona. Inclusive el peor test con el usuario incorrecto mostrará cosas que mejorarán su sitio Web o aplicación
- ✓ *Testear un usuario temprano en el proyecto es mejor que testear 50 al final*: Testear no debe ser un proceso aparatoso, ni debe ser visto como una especie de prueba integral del sistema que se hace al final, para ver si todo funciona bien. Recuerde que el código es como el concreto, y encontrar errores tarde es muchas veces casi lo mismo que no encontrarlos.
- ✓ *La importancia para reclutar testers está en general sobrevalorada*: Testear usabilidad es completamente distinto que realizar focus groups. La mayoría de los problemas tienen que ver con el sentido común y con la visión de la interacción de la aplicación para quienes no están inmersos en el contexto del desarrollo del proyecto. Este tipo de problemas los puede detectar cualquier persona, el usuario perfecto no es tanto mejor que uno elegido al azar.

---

[35] *Don´t make me think - Steve Krug, Ibídem página 142. (Excepto lo que está encomillado, la cita no es textual)*

- ✓ *El objetivo del testing no es probar o desaprobar algo. El objetivo es permitir a los diseñadores tomar decisiones informadas*: Ya dijimos más arriba que es muy difícil que un test de usabilidad determine sin dudas que la opción A es mejor que la opción B, y en general en estos casos los diseñadores profesionales tomarán esta decisión sin dificultades. El testeo de usabilidad proveerá al equipo de diseño de ventajas y desventajas para cada opción estudiada, lo que sumado a su experiencia y restricciones del proyecto les permitirá tomar una mejor decisión en cada caso.
- ✓ *Nada es mejor que una reacción de la audiencia en vivo*: "Una razón por la que las películas de los Hermanos Marx son tan fantásticas, es porque antes de comenzar a filmarlas, realizaban un tour por el circuito del vaudeville y ponían en escena las partes del film, con cinco shows al día, improvisando constantemente y notando cuales líneas recogían las mejores risas. Inclusive después de acordar sobre una línea, Groucho insistía probando pequeñas variaciones para comprobar si podía ser mejorada."

Testear la usabilidad del diseño primero y del sitio o aplicación real después no es una cosa de una vez y para siempre. Inclusive un producto que presentaba un nivel excelente de usabilidad cuando salió al mercado, puede decaer en la medida en que surgen otros productos competitivos, los usuarios se hacen más expertos en su uso, surgen nuevas áreas de aplicación o se agregan nuevas versiones.

Un test de usabilidad casero

Si no cuenta con presupuesto para contratar un experto en usabilidad, afronte usted mismo la tarea. Recuerde que un test, inclusive uno casero es siempre mejor que ningún test de usabilidad. Asumamos para lo que sigue, que usted cuenta con el esqueleto estático del sitio resultado del diseño, del que hablamos en el capítulo anterior, y que se dispone a testearlo.

Reclute a los participantes

Los expertos afirman que entre 5 y 8 participantes es lo adecuado. Después de 7 u 8 participantes, las pruebas comienzan a ser idénticas una tras otra, sin generar hallazgos nuevos. De todos modos, tres participantes le aportarán una ayuda invalorable, y recuerde que siempre un test es 100% más que ninguno.

De acuerdo a sus posibilidades, defina cómo reclutar a los participantes. Tal vez compañeros de trabajo no involucrados en el proyecto, clientes de la empresa, amigos o familiares. Siempre es mejor que pertenezcan al segmento objetivo del sitio que estamos testeando, pero salvo que se trate de un sitio sobre un tema extremadamente específico, la diferencia entre testear con el candidato ideal y el auxiliar contable de la empresa es mucho menor de lo que usted imagina.

Si tiene dudas, aplique el sentido común. El auxiliar contable de la empresa está ahí, disponible, a la espera de que usted le pida que lo ayude. Realice la prueba con él y si no le satisface, entonces sí encare la tarea más difícil de encontrar un candidato perteneciente al segmento objetivo.

La infraestructura

Cargue el sitio en un PC normal. Si fuera posible, cárguelo en un servidor Web real y utilice un PC conectado a Internet para la prueba, de modo de tener tiempos de respuesta más parecidos a los que tendrá una vez que el sitio real esté en el aire. Si no cuenta con un servidor propio, puede conseguir uno por 10 dólares mensuales, y si eso es muy oneroso acceda a un servidor gratis a través de sitios Web del tipo de Geocities.[36] Coloque el PC en una habitación que contenga la menor cantidad de elementos que puedan distraer o poner nervioso a los usuarios de prueba.

Solo falta decidir si filmar o no filmar. La mayoría de este tipo de filmaciones no tienen gran utilidad, nadie las mira. Por otra parte, la filmadora es muy invasiva e intimida a los usuarios mientras trabajan. Si está dispuesto a gastar media hora más en ver nuevamente un video, consiga un usuario más y gaste ese tiempo en probar nuevamente, será mas productivo. Vale la pena la intrusión de una filmadora si el auditorio

[36] *En www.thefreesite.com puede encontrar una lista completa de todo tipo de servicios gratuitos en Internet, incluyendo hosting, email, listas de distribución entre otros.*

para la prueba es muy grande. Más de dos personas junto al usuario es demasiado, y es muy bueno que todos los desarrolladores del equipo, e inclusive invitados vinculados al proyecto, vean en vivo y en directo los test. Para ello sí vale la pena poner una cámara sobre un trípode, y reproducir filmación y audio en la pieza de al lado, de modo que puedan observar sin molestar y en tiempo real todos los que lo deseen.

El plan de la prueba

Recorra el esqueleto del sitio, y anote en un cuaderno los puntos que desea testear, dedicando un renglón a cada uno. Considere que una prueba razonable para un usuario puede durar de 45 minutos a una hora, muchas veces ni tanto, por lo que una lista de 8 renglones estará bien.

Verifique personalmente que todo funciona bien

Realice usted mismo la prueba, para verificar que el sitio que conforma el esqueleto estático está accesible, que los links funcionan, que las imágenes se cargan correctamente. Verifique fehacientemente que no perderá ni su tiempo ni el de sus equipo ni el de sus usuarios de prueba. A la vez, el recorrido le permitirá ir pensando cómo llevar adelante la prueba, qué preguntar, cómo guiar a su usuario durante la misma.

Reciba al primer usuario

Este momento es crucial. En los primeros 3 a 5 minutos debe conseguir que su usuario de pruebas se distienda, a la vez que usted desarrolla los tres pasos de bienvenida, vitales para que el resto de la prueba sea de utilidad:

- ✓ *Busque empatía con el usuario de prueba.* Salude amablemente e intente en todo momento colocarse al mismo nivel que su invitado. Es mejor jugar el rol de alumno que el de profesor. Vale la pena hacerle dos o tres preguntas de cortesía, sobre su ocupación, sobre el barrio en el que vive o sobre el tiempo. Si no consigue quebrar el hielo al comienzo, las probabilidades de que la prueba sea útil se reducen.
- ✓ *No es una prueba al usuario sino al sitio*. Expréselo explícitamente. Debe decirle al usuario algo así: "En XYZ estamos construyendo un sitio Web. Hemos trabajado tanto en él que ahora

ya no nos damos cuenta de si navegar es fácil o difícil y queremos que nos ayude a probarlo. No es una prueba de su inteligencia, ni de su habilidad, sino de nuestro sitio. Pensamos que puede tener algunos errores, y probablemente usted pueda ayudarnos a encontrarlos."

- ✓ *Pídale que piense en voz alta*. Es imprescindible que a medida que navega, el usuario de la prueba nos explique por qué toma cada decisión. Para ello alcanza con que piense en voz alta, lo que terminará en algo por el estilo de: "Esta oreja dice 'configuración', acá creo que se pueden cambiar los colores" o "si escribo 'Batlle' acá y clickeo en 'Buscar' me imagino que me va a traer una cantidad de links de páginas que tienen Batlle"

Manos a la obra

Pídale al usuario de prueba que recorra un poco el sitio al azar durante 3 a 5 minutos, para tener una primera impresión, siempre repitiendo en voz alta lo que piensa, lo que le parece aquello que ve en la pantalla. Luego comience una a una con la lista de dudas de su cuaderno, teniendo en cuenta las siguientes recomendaciones: [37]

- ✓ *Proteja a los participantes:* no interrumpa inútilmente a su usuario de prueba, déjelo que se enfrente solo a los problemas, pero esté atento para que no caiga en situaciones engorrosas o ridículas. Usted tiene la responsabilidad de mantener el equilibrio en el nivel de dificultad al que se enfrenta, para que la prueba sea útil sin que su participante sienta frustración.
- ✓ *Busque siempre saber qué tiene el usuario de prueba en su cabeza:* Como si fueran los globos sobre los personajes de las historietas, es necesario leer el pensamiento de su usuario de prueba. Persiga siempre el objetivo de entender porqué hace tal o cuál cosa, más que registrar exactamente qué hace. La facilidad de uso es un problema de modelo mental vs. modelo de representación. La pruebas de usabilidad intentan entender el modelo mental y éste reside dentro de la cabeza de sus invitados.

---

[37] *Basado en Don´t make me think - Steve Krug, Ibídem página 155*

- ✓ *No le de pistas*. por más desesperado que usted esté, no le de pistas a su usuario de prueba. Las pistas y ayudas hacen que la prueba no valga de nada. Si está probando la pantalla A y se tranca, déjelo que pruebe una o dos opciones, y tres o cuatro también. Solamente cuando crea que debe proteger a su usuario frente a una situación inconveniente, ayúdelo a salir, pero en ese momento la prueba de la pantalla A habrá finalizado y habrá que pasar a probar la pantalla B.
- ✓ *Mantenga sus instrucciones y preguntas simples*. Cuanto usted más habla, más tiempo pierde y menos prueba. Un buen test requiere por parte del moderador de una introducción breve y de frases y preguntas concisas.
- ✓ *Investigue, sondee*: los usuarios no dicen todo lo que piensan, al menos no lo que nosotros queremos saber. Sin ser cargoso, cuando encuentra una pista investigue, pregunte, al final ese es el objetivo del test. Por ejemplo si un usuario le dice: "no clickeo este link porque me parece que acá no está la foto que busco", sería bueno preguntarle "¿qué le parece que aparecerá cuando se clickea ese link?".
- ✓ *No tenga miedo de improvisar:* los test de usabilidad le proporcionarán sorpresas. Cambie los planes sin miedo para interpretarlas, para investigarlas. Si los dos primeros usuarios se trancan fuertemente con la búsqueda, y usted ya se hizo una idea de cómo arreglarla, elimine la búsqueda del tercer test, y ponga foco en otro nuevo problema que detectó el segundo usuario.
- ✓ *No se preocupe si un usuario resulta poco experiente, o se tranca demasiado.* Por el contrario, de los usuarios súper expertos no aprenderá mucho ya que entienden todas las metáforas y conocen todos los trucos. Se descubren problemas, se detectan posibles soluciones con los usuarios que no consiguen seguir adelante con facilidad. Su objetivo son los sobrevivientes, los apologistas sobrevivirán sin necesidad de su ayuda.

Que pase el que sigue...

Una vez realizada la prueba, escriba un párrafo o dos con los resultados. No cometa ninguno de los dos errores más habituales. Dejar para escribir después con más tiempo hace que nos olvidemos de detalles importantes. El mejor momento para escribir los comentarios es exactamente ese, apenas el usuario terminó el test. Escribir informes demasiado largos es contraproducente, nadie los lee. Cada uno de los que está observando la prueba, usted incluido, debería escribir uno o dos párrafos con las conclusiones y comentarios de lo que encontró en la prueba con el usuario. Y listo para que pase el señor que está esperando amablemente junto a la puerta.

La reunión de análisis y resumen

Una vez que finalizó la prueba del último usuario, junte al equipo y discutan sobre los hallazgos, los problemas detectados y las posibles soluciones. No deje esta reunión para después, a la mañana siguiente el valor de la reunión y de sus resultados no alcanzará ni a la mitad del valor de la reunión realizada en caliente, a continuación de la última prueba.

¡Gracias!

Si llegó hasta aquí, no me queda más que agradecerle su paciencia y dedicación. Si además encontró al menos una idea que piensa que le resultará útil, el agradecimiento es doble.

Tal como dice la Introducción, este libro está dedicado a mejorar la calidad del proceso con el que se concibe y diseña el software, dotando a quienes participan en ese proceso de herramientas que hagan más productivo su trabajo, y cuando toque, hacer más distendida la vida de quienes conviven con nosotros los programadores. Es duro reconocerlo, pero son demasiadas las veces que nuestro complicado mundo de siglas y protocolos, sumado a numerosos "no se puede", transforma la tarea de nuestros compañeros de proyecto no-técnicos en una actividad frustrante. Si en su próximo proyecto nota alguna mejoría, por mínima que sea, resultado de la aplicación de algún concepto o idea expuesta aquí, entonces este libro habrá sido un éxito. Si el costo a pagar es el enojo de algún programador, por mi parte estoy dispuesto a pagarlo con gusto.

Todo este libro fue escrito con la ilusión de que es posible transmitir la pasión por el Diseño. Diseñar es el desafío de crear equilibrios, de establecer balances. Diseñar es la tarea de decidir que hay que incorporar y que hay que eliminar. Diseñar es elegir unos caminos y renunciar a otros. Diseñar es una técnica, teoría condimentada por la práctica colectiva, la experiencia individual y el sentido común. Diseñar es planificar en detalle qué vamos a hacer, antes de hacerlo. Diseñar es pensar primero.

Montevideo, 31 de octubre de 2004

No deje de enviar sus comentarios sobre el libro a través de www.mordecki.com/libro

158. Los usuarios usando

Glosario

Término	Concepto
Programa	Conjunto de instrucciones en el lenguaje de la computadora, que permiten utilizarla para realizar una tarea, o un conjunto de ellas.
Programadores	Ingenieros de sistemas, Analistas de Sistemas, Desarrolladores, Programadores y en general todos los técnicos que desarrollan, implementan y dan soporte a las aplicaciones informáticas
Usuarios	Todas las personas vinculadas a las aplicaciones informáticas, que no son programadores
Aplicación	Conjunto de programas para resolver un problema utilizando una computadora o un conjunto de ellas. Si bien el límite entre Aplicación y Programa puede ser difuso, en general se habla de una Aplicación para problemas más grandes y complejos que los que resuelve un programa.
Sitio Web	Conjunto de páginas Web que tienen un tema común. Programa o aplicación a la que se accede utilizando un Navegador Web
Parámetro	Definición que modifica el comportamiento de una aplicación, pero que no forma parte de los datos básicos de la misma. Por ejemplo el archivo para utilizar como fondo en el escritorio.
Configuración	Conjunto de valores que tienen en un momento dado todos los parámetros de una aplicación.
Default	Valor que se asigna a un parámetro cuando se instala la aplicación, sin necesidad de que el usuario tome ninguna decisión al respecto

Sistema	Conjunto de computadoras, equipos de comunicaciones, cables, sistemas operativos, utilitarios y aplicaciones que componen la estructura de tecnología de la información de una organización.
Hardware[38]	En los sistemas, todos los componentes físicos como computadoras, servidores y equipos de comunicaciones.
Software	En los sistemas, todos los intangibles, como programas, aplicaciones, sistemas operativos y bases de datos
Microprocesador o Chip	El corazón de una computadora, pastilla de Silicio que ejecuta millones de operaciones lógicas por segundo para permitir que los programas se ejecuten.
Digital	Equipo o dispositivo que funciona en base a un chip y software.
Interface	La parte de los sistemas, tanto de hardware como de software que interactúan directamente con los usuarios
Feature	(Característica) Un atributo particular que tiene una aplicación o programa, una tarea que es capaz de resolver, como por ejemplo: convierte múltiples monedas, calcula automáticamente los impuestos o quita los ojos rojos de las fotos.
Función	Parte de un programa que resuelve un problema concreto. En particular, es razonable pensar que un Feature terminará implementándose como una función.

[38] Una definición de Hardware y Software que circula entre los programadores dice que Hardware es todo lo que puedes partir con el hacha, mientras que Software es aquello que solo puedes insultar.

Graphic User Interface (GUI)	Interface de Usuario Gráfica - Interface de usuario basada en elementos gráficos como ventanas, punteros y cursores. La más conocida es la de Windows
Navegador Web	Programa que permite acceder a los Sitios Web. Por ejemplo Netscape, Internet Explorer, FireFox