Interfaz: Ventana del siglo XXI
La importancia de la interfaz de software hoy

Nicolás Correa Morales

Preguntarse por la historia particular de la interfaz de software y la forma en que se organiza internamente podría parecer una selección azarosa, pero una de las razones que me motiva a dedicarle atención es que se trata de la cara visible de los programas que en buena medida moldean el mundo y gestionan cómo lo vemos o interactuamos con él, mientras que a la vez su centralidad en nuestra historia actual queda invisibilizada por ser un instrumento de paso que no provoca intrigas y es sobreentendida en el cotidiano. Mencionarla es redundante.

Alexander Galloway señaló en *The Interface Effect* el efecto codificador de la interfaz en el plano social; y por su parte, Marshall McLuhan (al menos en la etapa de su teoría que inscribe en *Understanding Media*) aborda los medios en general en tanto elementos que tienen en sí un significado, o sea, en tanto transmisores de efectos tras cuya observación del aparato se pueden deducir las implicancias sociales y técnicas que los concibieron, más allá del mensaje que porten. Así, para McLuhan un planisferio no solo nos transmite información sobre los territorios de las naciones, sino que, a través del mismo mapa en tanto medio, vamos a poder observar los intereses que prefirieron tal proyección por sobre otras, o las formas de representación de las fronteras y sus motivos, o el material utilizado para la impresión, etc.

Por su parte, Carlos Scolari (2018) propone una serie de leyes que ajustan y definen a la interfaz en diversos grados. Las conexiones entre objetos y sujetos que plantea nos permiten entender la interfaz como un medio.

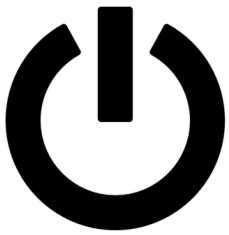
Así, si entendemos la interfaz como un medio comunicador a veces entre objetos y a veces entre sujetos, podemos asegurar que tanto Galloway como McLuhan y Scolari están de acuerdo en las potentes implicancias de esta, tanto si hablamos

de la interfaz como producto o en tanto productora de efectos sociales y técnicos. Enfrentando ambas afirmaciones nos encontramos con que la interfaz de software en términos generales es sensible de ser el punto en el que confluyen las interacciones humanas con los sistemas internos, ya sea en sentido humano-sistema o al revés.

Presentaré a continuación las dos principales formas en que nos llega a nosotros la interfaz de los softwares que usamos.

Gramática de intercambios: construcción centralizada de interfaz de software

Scolari (2018) realizó una serie de definiciones útiles para delimitar a qué significado de interfaz haré referencia. En primer lugar, nombra a cada noción de la interfaz con una metáfora. La noción de interfaz en tanto hardware es identificada con la metáfora "interfaz como intercambio de información" y refiere a la interfaz como la parte física de una computadora que posibilita la correcta conexión con otro aparato, facilitando el intercambio de información entre ambos. También llama "interfaz de superficie" a aquella que figura en la pantalla y que con sus ventanas, íconos, menús y cursores[1], debe resultar al usuario fácil de leer, debe serle familiar, y también remitir a interfaces que ya haya visto.



La pregunta "¿qué ser humano del siglo XXI se cuestiona sobre si este es el botón indicado para encender el software o el aparato?" es superficial. La pregunta correcta sería más bien "¿son estas las caras visibles más conocidas por nosotros de las tecnologías que habilitan el 90

Podríamos historizar específicamente de dónde viene el símbolo de encendido o quién fue el primero en emplear los menús en el lado superior de la ventana y

[1] En computación: windows, icons, menus and pointers (WIMP).

los porqués de cada elemento inicial que se tome, pero delimitaremos nuestro interés a la interfaz en términos generales, entendiéndola como resultante de una confluencia de opciones, cara visible de elementos y botones con los que interactúa directamente el usuario.

La serie de elementos que facilitan la conversación (en términos de Scolari) usuario-software responden, como dice Scolari, a "una gramática que regula los intercambios con los sujetos que las utilizan" (2018, p.31). Esta serie de elementos y su gramática son el resultado de largos e intrincados recorridos de los desarrolladores de software en su conjunto; lo que llega a nuestras pantallas son las conclusiones de una serie debates acerca de un diseño ideal de los botones y menús, debates que hoy en día siguen en vigencia y nos entregan periódicamente el grado más actualizado de esas conclusiones, que por su naturaleza son siempre temporales. Vemos que continúan estos debates cuando un software lanza un nuevo parche para solo cambiar de lugar elementos o también (y este es un caso más explícito) cuando el software es online y no requiere de la descarga estricta de una actualización, como nos pasa a los usuarios de los programas online de Google cuando nos advierte la misma plataforma con una ventana emergente que tales o cuales herramientas han cambiado de posición. Yo, usuario, leo, asiento y cliqueo en el botón de "Entendido" en la esquina inferior derecha del mensaje sin tener conciencia total de que estoy accediendo a un nuevo estadío de la interfaz que conozco: la evolución de esta sigue su marcha.

En 1977, un año después de que apareciera la primera computadora personal de Apple, la compañía realizó un documento titulado *Apple Human Interface Guidelines*, un compendio con una auténtica gramática para la interacción con el humano, lo que Alan Turing llamaría interacción computadora-usuario y, Scolari, diseñador-usuario. Esta documentación, que también hoy continúa desarrollándose (según el especialista en Procesos Manabu Ueno este documento hoy sorprende con sus constantes cambios), incluyó, en esos primeros compendios, secciones cuyos títulos pueden resultan raros a simple vista si recordamos que es un manual técnico del campo de las ciencias de la computación: "intuición", "familiaridad", "la metáfora". Con respecto a este último título, resulta oportuno mencionar que Scolari se ocupa de ahondar en el concepto de metáfora antes de aplicarla para caracterizar sus tipos postulados de interfaz:

> La metáfora permite comprender una idea o un dominio conceptual

> en términos de otro. No es sólo un ornamento poético del lenguaje [...]. Como escribieron George Lakoff y Mark Johnson en su clásico *Metaphors We Live By*, las metáforas juegan un papel central en la puesta del discurso y comprensión del mundo que nos rodea, incluido el de los diseñadores y usuarios de las interfaces. (2018: p.21)

Así, los diseñadores se ocuparon de captar la forma más intuible para resumir complejos menús y opciones en una palabra o incluso símbolos, el "+" tomado de la matemática para referirnos a aumentar el tamaño de algo, el globo de diálogo prestado de las viñetas para comentar un documento, el balde de pintura para "pintar" una superficie con un color, una flecha curva señalando hacia la izquierda para deshacer un cambio (basada en la estudiada idea de la línea de tiempo, inconsciente intuición de las culturas occidentales de que para la izquierda se retrocede al pasado y para la derecha se avanza hacia el futuro).

El código abierto: construcción comunitaria de interfaz de software

Si bien las diversas gramáticas desarrolladas por las grandes firmas para llegar a las interfaces robustas y efectivas de hoy abrieron el paso, hay diversas dinámicas que parecen representar un eco de las posibilidades actuales que tiene un individuo de trabajar en conjunto con otros de otras partes del mundo sin la interferencia de algún gobierno o autoridad superior.

La modalidad de desarrollo de software de *open source* o código abierto es una de las más relevantes e interesantes para estudiar dentro de las enormemente diversas dinámicas que emplea la programación. Muchos de los procedimientos de desarrollo de software cuando una empresa es su dueña pueden resumirse en: equipo que construye el software, empresa dueña que busca satisfacer una demanda de usuario y chequeos para confirmar que el software sí se condice con la necesidad del usuario. Se ha demostrado lo efectivos de esos métodos, pero no por eso son perfectos.

Con el código abierto, la construcción se da en desorden, pero de una forma igual de efectiva. No se trata de un grupo de desarrolladores trabajando para la firma de software en un código que se mantiene cerrado, hiperespecializado, sino que son multitudes de ojos que observan el código. Esto es gracias al pilar de su funcionamiento que es la comunidad, cuyas ventajas pueden resumirse en dos: la multiplicidad de voces (son tantos más los miembros de una comunidad

en relación a un grupo de desarrolladores de una empresa, que es difícil pensar en algún rincón con un defecto que sea pasado por alto), y la semiabstracción (no se especializan en eso, lo que otorga la visión panorámica de ser también usuario).

Los programas de código abierto son los que ponen al alcance del público todo su código fuente, es decir, el programa está expuesto totalmente, permitiendo que un usuario acceda a él y lo altere a voluntad en su versión de escritorio para agregar una característica que crea que le falte. En caso de requerirlo la comunidad, el usuario publica sus cambios locales dando lugar a la nueva versión del programa, cuyos cambios son arduamente revisados y comprobados.

En este sentido, y en contra de quienes afirman que la interfaz es por definición opaca, las interfaces creadas con código abierto son transparentes totalmente para aquellos que manejen el código con el que está programada.

Este nivel de maniobrabilidad implica un compromiso de la comunidad que rodea a esta interfaz y significa también un interés en *ser parte* del programa y no solo usuario. Claro que en adición existe una parte de la comunidad que no participa de forma directa en la generación de soluciones pero disfruta de sus beneficios.

Y no son casos marginales, los software de código abierto son muy populares y tienen una larga historia. Actualmente, aunque la mayoría de los proyectos se caracterizan por ser propiedad de la comunidad, los hay con respaldo de grandes empresas como Google, que impulsa 2000 proyectos open source actualmente[2],o Apple, que además de contar con muchos proyectos, lanzó en 2014 el hoy popular lenguaje de programación Swift, de código abierto.

Ante la aparición de gigantes representantes del software no transparente como estos en escena, surgiría la pregunta de quién valida fehacientemente que un proyecto esté del todo desligado a los servidores y repositorios de información mundial que pertenezcan, directamente o no, a estas firmas. Para esto hay organizaciones como Free Software Foundation (1985) y Open Source Initiative (1998) que auditan este tipo de proyectos y a los que la comunidad internacional se refiere actualmente.

Por mencionar proyectos célebres de código abierto, se encuentra el sistema operativo Linux, empleado hoy ampliamente en los sectores de análisis masivos

[2] https://opensource.google/

de datos y en las ciencias no necesariamente computacionales por la enorme versatilidad y amigabilidad de su diseño. Otro ejemplo es el popular navegador Mozilla Firefox, que nació en 1998 como un proyecto de una comunidad de código abierto, o el caso de una de las hijas más omniscientes de la era de la computación, Wikipedia.

Las desventajas de este tipo de código son muy pocas, y son atropelladas bajo la estampida de mejoras que implica esta filosofía de trabajo a la que podemos poner muchas etiquetas: software libre, programas susceptibles de fusión con otros programas, actualización constante, comunidad, transparencia o, (¿por qué no?) comunismo computacional.

¿Por qué la interfaz de software?

El diseño de una paleta de botones y herramientas dispuestas a ser usadas o a desplegarse para ramificarse en subherramientas más específicas puede no solo ser resultado de gramáticas que busquen lo intuitivo y lo sencillo, sino que puede ser además condicionante de la forma en que el usuario asiduo de estas herramientas entiende luego ciertos aspectos del mundo.

La creación de estas interfaces es resultado, a veces, del capital de una firma, y otras, del trabajo sinérgico de una comunidad desinteresada que trabaja el código abierto. Las decisiones tomadas en uno u otro escenario dan forma a la estructura y a las herramientas de la interfaz de software y por lo tanto repercuten en la manera en que el software mismo trabaja. Si a esto agregamos que saber usar ciertas interfaces es en sí un capital intelectual porque tiene directa relación con la velocidad y la calidad de un resultado, esta interfaz podría no ser tan fácilmente discernible de la sustancia de los software, que incluye, especialmente en el tiempo postpandémico, absolutamente todos los ámbitos: estudiantil, laboral, del entretenimiento, social, sexual.

Bibliografía

Galloway, A., *The Interface Effect*. Polity Press (2012).

Kittler, F., "La ciudad es un medio", en *La verdad del mundo técnico*. (2018 [2013]).

McLuhan, M., *Comprender los medios de comunicación*. MIT University press (1964 [1996]).

Scolari, C., *Las leyes de la interfaz: Diseño, ecología, evolución, tecnología*. Gedisa Editorial (2018).

—-

Apple Human Interface Guidelines <Recuperado en 2021 de aquí>

Google Open Source <Recuperado en 2021 de aquí>

Free Software Foundation <Recuperado en 2021 de aquí>

Open Source Initiative <Recuperado en 2021 de aquí>