La economía WTF*

El futuro que nos espera y por qué depende de nosotros

BESTSELLER DEL *WALL STREET JOURNAL*

Tim O'Reilly

El fundador y CEO de O'Reilly Media nos propone un viaje al mañana de la tecnología

* What The Fuck

DEUSTO

La economía WTF

El futuro que nos espera y por qué depende de nosotros

TIM O'REILLY

Traducción de Rebeca Bouvier

EDICIONES DEUSTO

Título original: *WTF*
Publicado por Harper Collins Publishers, Nueva York, 2017

Deusto es un sello editorial de Centro Libros PAPF, S. L. U.
Grupo Planeta
Av. Diagonal, 662-664
08034 Barcelona

www.planetadelibros.com

ISBN: 978-84-234-2932-5
Depósito legal: B. 7.752-2018
Primera edición: mayo de 2018
Preimpresión: gama, sl
Impreso por Romanyà Valls, S.A.

Impreso en España - *Printed in Spain*

Sumario

1

Ver el futuro en el presente

En los medios, a menudo se me tacha de futurista. Yo no me veo así. Más bien me considero un cartógrafo, esto es, trazo un mapa del presente que ayuda a ver las posibilidades del futuro. Los mapas no son meras representaciones de ubicaciones y caminos físicos, sino cualquier sistema que nos ayude a ver dónde nos encontramos y adónde intentamos ir. Una de mis citas favoritas es de Edwin Schlossberg: «Saber escribir consiste en crear un contexto en el que otras personas puedan pensar».[23] Este libro es un mapa.

Usamos mapas —abstracciones simplificadas que representan una realidad subyacente— no solo para ir de un lugar a otro, sino en todos los aspectos de nuestras vidas. Cuando caminamos por nuestra casa a oscuras sin necesidad de encender la luz, es porque hemos interiorizado un mapa mental del espacio, la distribución de las habitaciones, la ubicación de cada silla y cada mesa. De manera similar, cuando un emprendedor o un inversor de riesgo va a su trabajo todos los días, utiliza un mapa mental del panorama tecnológico y empresarial. Disponemos el mundo en categorías: amigo o conocido, aliado o competidor, importante o irrelevante, urgente o trivial, futuro o pasado. Hemos creado un mapa mental para cada categoría.

Pero tal como nos recuerdan las tristes historias de gente que siguen religiosamente su GPS por un puente que ya no existe, los mapas pueden equivocarse. En los negocios y en la tecnología, a menudo no vemos claramente lo que tenemos delante porque navegamos utilizando mapas viejos y a veces incluso mapas malos —mapas que obvian detalles fundamentales sobre nuestro entorno o quizás incluso lo distorsionan a propósito.

La mayoría de las veces, en campos rápidamente cambiantes como la ciencia y la tecnología, los mapas son incorrectos sencillamente por-

que hay demasiadas cosas que se desconocen. Todo emprendedor, todo inventor, es a la vez un explorador que intenta dar sentido a lo que es posible, a lo que funciona y a lo que no, y a cómo seguir adelante.

Piensa en los emprendedores que trabajaron para desarrollar el ferrocarril transcontinental de Estados Unidos en el siglo XIX. Se propuso por primera vez en 1832, pero no estuvo claro que el proyecto fuera factible hasta la década de 1850, cuando la Cámara de Representantes proporcionó financiación para una exhaustiva serie de exploraciones del oeste norteamericano que precedieron a la construcción. Tres años de exploración —entre 1853 y 1856— dieron como resultado los *Pacific Railroad Surveys*, una colección de doce tomos de datos sobre 650.000 kilómetros cuadrados del oeste norteamericano.

No obstante, todos esos datos no despejaron completamente el camino. Se produjo un debate encarnizado sobre la mejor ruta, un debate que no consistió tan solo en los méritos geofísicos de las rutas del norte en comparación con las del sur, sino también en cuestiones que atañían a la esclavitud. Incluso cuando en 1863 se decidió la ruta y empezó la construcción, hubo problemas inesperados: un grado de pendiente más pronunciada de lo indicado, que una locomotora no podía salvar, condiciones meteorológicas que hacían ciertas rutas intransitables durante el invierno... No se podía simplemente trazar una línea sobre un mapa y esperar que todo fuera a la perfección. Se tuvo que perfeccionar y redibujar el mapa con más y más capas de datos esenciales que se fueron añadiendo hasta que resultó lo suficientemente claro como para poder utilizarlo. Los exploradores y topógrafos siguieron muchos caminos equivocados antes de decidirse por la ruta final.

Crear el mapa correcto es el primer reto al que nos enfrentamos al intentar comprender las tecnologías WTF actuales. Antes de poder entender cómo abordar la IA, las aplicaciones a la carta y la desaparición de los empleos de la clase media, y cómo todo ello puede integrarse en un futuro en el que queramos vivir, hemos de asegurarnos de que no nos cieguen ideas anticuadas. Hemos de ver patrones que cruzan viejas barreras.

El mapa hacia el futuro es como un rompecabezas al que le faltan muchas piezas. Puedes ver el esbozo de una imagen en un extremo y otro esbozo en otro extremo, pero hay muchos vacíos y te es difícil hacer las conexiones. Y entonces un día alguien vuelca otra serie de piezas sobre la mesa y de repente la imagen se vuelve clara. La diferencia entre un mapa de un territorio desconocido y un rompecabezas es que en el segundo caso nadie conoce la imagen entera por adelantado. No existe

hasta que la vemos... es un puzle cuya imagen va surgiendo conforme avanzamos, una imagen tanto inventada como descubierta.

Encontrar el camino al futuro es un acto de colaboración y cada explorador coloca piezas fundamentales que permiten a los demás seguir adelante.

Captar las rimas

Mark Twain tiene fama de haber dicho: «La historia no se repite, pero a menudo rima».[24] Estudia historia y encontrarás pautas. Esta es la primera lección que aprendí sobre cómo pensar acerca del futuro.

La historia sobre cómo se desarrolló, perfeccionó y adoptó el término *software de código abierto* a principios de 1998 —lo que nos ayudó a entender la naturaleza cambiante del software, cómo ese nuevo entendimiento cambió el curso de la industria y lo que predijo del mundo por llegar— muestra de qué forma los mapas mentales que utilizamos limitan nuestra manera de pensar, y cómo revisar el mapa puede transformar las decisiones que tomamos.

Antes de profundizar en lo que ahora ya es historia antigua, necesito que retrocedas a 1998.

El software se distribuía en cajas retractiladas, las actualizaciones llegaban como mucho una vez al año, a menudo cada dos o tres. Solo el 42 por ciento de los hogares estadounidenses disponían de un ordenador personal, en comparación del 80 por ciento actual que dispone de un *smartphone*. Solamente el 20 por ciento de la población estadounidense tenía un teléfono móvil. Internet era algo que emocionaba a los inversores, pero aún era diminuta, con apenas 147 millones de usuarios en todo el mundo, en comparación con los 34.000 millones de ahora. Más de la mitad de los usuarios de la red en Estados Unidos accedían a ella a través de AOL. Amazon y eBay habían sido lanzadas tres años antes, y Google se fundó en septiembre de ese año.

Microsoft había convertido a Bill Gates, su fundador y consejero delegado, en el hombre más rico del planeta. Era la empresa definitoria de la industria tecnológica, con una posición casi monopolística en software para ordenadores personales, ventaja que había aprovechado para destruir a un competidor tras otro. El Departamento de Justicia

de Estados Unidos inició una investigación antitrust contra la empresa en mayo de aquel año, igual que había hecho casi treinta años antes contra IBM.

En contraste con el software exclusivo que hizo de Microsoft una empresa tan exitosa, el software de código abierto se distribuye bajo una licencia que permite a cualquiera estudiar, modificar y aportar libremente a dicho software. Entre varios ejemplos de software de código abierto se cuentan los sistemas operativos Linux y Android; navegadores como Chrome y Firefox; lenguajes de programación populares como Python, PHP y JavaScript; herramientas modernas de *big data* como Hadoop y Spark; y juegos de herramientas punteras de inteligencia artificial como Tensorflow de Google, Torch de Facebook y CNTK de Microsoft.

En los primeros tiempos de los ordenadores, la mayoría del software era de código abierto, pero no se llamaba así. El ordenador venía con un software operativo básico, pero gran parte del código que hacía que el ordenador fuera útil era software propio, escrito para resolver problemas específicos. El software escrito por científicos e investigadores en concreto a menudo se compartía. A finales de la década de 1970 y comienzos de la de 1980, sin embargo, las empresas se dieron cuenta de que controlar el acceso al software les proporcionaba ventaja comercial y empezaron a bloquearlo utilizando licencias restrictivas. En 1985, Richard Stallman, un programador del Massachusetts Institute of Technology publicó *The GNU Manifesto*, donde estableció los principios de lo que él llamó «software libre» —no en lo que se refería a precio, sino a libertad: libertad para estudiar, redistribuir y modificar software sin permiso.[25]

La ambiciosa meta de Stallman era crear una versión completamente libre del sistema operativo Unix de AT&T, originalmente desarrollado en Bell Labs, la rama de investigación de AT&T.

Cuando Unix se desarrolló a finales de la década de 1970, AT&T funcionaba como un monopolio legal con enormes beneficios procedentes de servicios telefónicos regulados. En consecuencia, AT&T no tenía permiso para competir en el sector informático, entonces dominado por IBM, y de acuerdo con el decreto de consentimiento de 1956 con el Departamento de Justicia, había autorizado el uso de Unix a grupos de investigación informática con condiciones ventajosas. Programadores de universidades y empresas de todo el mundo respondieron aportando elementos clave al sistema operativo.

Pero después del decisivo decreto de consentimiento de 1982, por el cual AT&T acordaba ser dividida en siete empresas más pequeñas (las

«Baby Bells») a cambio de poder competir en el mercado informático, AT&T intentó que Unix estuviera protegido. Demandaron a la Universidad de California, Berkeley, que había creado una versión alternativa de Unix (la Berkeley Software Distribution, o BSD), y finalmente lograron cerrar la organización colaborativa que había ayudado a crear el sistema operativo desde el principio.

Mientras el Unix de Berkeley era paralizado por los ataques legales de AT&T, el proyecto GNU de Stallman (llamado así por las siglas de «Gnu's Not Unix», es decir, GNU no es Unix) había duplicado todos los elementos de Unix excepto el núcleo, el código central que actúa como una especie de policía de tráfico para el resto del software. Ese núcleo fue proporcionado por un estudiante de informática finlandés llamado Linus Torvalds, cuya tesis de su máster de 1990 consistió en un sistema operativo minimalista parecido a Unix que podría ser transportable a arquitecturas informáticas diferentes. Llamó a este sistema operativo Linux.

Durante los años siguientes, hubo una actividad comercial virulenta en la que los emprendedores aprovecharon las posibilidades de un sistema operativo completamente libre combinando el núcleo de Torvalds con la recreación del resto del sistema operativo Unix de la Fundación por el Software Libre. El objetivo ya no era AT&T, sino Microsoft.

En los principios de la industria de los PC, IBM y un creciente número de proveedores de ordenadores personales clónicos como Dell o Gateway ofrecían el hardware, Microsoft proporcionaba el sistema operativo y una serie de empresas de software independientes ofrecían las «aplicaciones asesinas» —procesamiento de texto, hojas de cálculo, bases de datos y programas de gráficos— que impulsaron la adopción de la nueva plataforma. El sistema operativo de disco (DOS, por sus siglas en inglés) de Microsoft fue una parte clave del ecosistema, pero en absoluto lo controlaba todo. Eso cambió con la introducción de Microsoft Windows. Sus interfaces de programación de aplicaciones (API, por sus siglas en inglés) hicieron más sencillo el desarrollo de aplicaciones, pero anclaban a los desarrolladores a la plataforma Microsoft. Los sistemas operativos competidores para PC como el OS/2 de IBM fueron incapaces de romper el predominio. Y al poco tiempo, Microsoft utilizó su primacía del sistema operativo para privilegiar sus propias aplicaciones —Microsoft Word, Excel, PowerPoint, Access y, más tarde, su navegador Internet Explorer (ahora Microsoft Edge)— y llegó a acuerdos de paquetes con grandes compradores.

El sector del software independiente para ordenadores personales murió lentamente mientras Microsoft se adueñaba de una categoría de aplicaciones tras otra.

Este es el patrón de rima que advertí: el sector del ordenador personal había empezado con una explosión de innovación que rompió el monopolio de IBM en la primera generación de la informática, pero había acabado en otro monopolio al estilo «el ganador se lo lleva todo». Busca patrones de repetición y pregúntate cuál será la siguiente iteración.

Por entonces, todo el mundo se preguntaba si la versión de escritorio de Linux podría cambiar el curso de la partida. No solo las empresas emergentes sino también grandes compañías como IBM, en un intento por recuperar su posición de liderazgo, apostaron que podrían.

Con todo, con Linux se buscaba algo más que tan solo competir con Microsoft. Estaba escribiendo las normas de la industria del software de una forma que nadie esperaba. Se había convertido en la plataforma sobre la que muchos de los grandes sitios web del mundo —en aquel momento, en concreto, Amazon y Google— se estaban creando. Pero también estaba redefiniendo la manera en que se estaba escribiendo el software.

En febrero de 1997, en el Congreso de Linux de Würzburg, Alemania, el hacker Eric Raymond publicó un trabajo titulado «The Cathedral and the Bazaar», que electrizó a la comunidad Linux.[26] Expuso una teoría del desarrollo de software basada en reflexiones sobre Linux y las propias experiencias de Eric con lo que más tarde vino a llamarse desarrollo de software de código abierto. Eric escribió:

> ¿Quién hubiera pensado, incluso cinco años atrás, que un sistema operativo de categoría mundial pudiera fusionarse como por arte de magia, a base del hackeo a tiempo parcial realizado por varios miles de desarrolladores distribuidos por todo el planeta, conectados únicamente por finas hebras de internet? [...] La comunidad Linux parecía un gran bazar balbuceante de programas y enfoques distintos (acertadamente simbolizados por los sitios archivos de Linux, que aceptaban solicitudes de todo el mundo) del que supuestamente, y solo a base de una sucesión de milagros, podía surgir un sistema coherente y estable.

Eric estableció una serie de principios que se han convertido, a lo largo de las últimas décadas, en parte del evangelio de desarrollo de software: que el software debe ser lanzado pronto y a menudo, mejor en un estado inacabado a la espera de ser perfeccionado; que los usuarios deben ser tratados como «codesarrolladores» y que, «dado un número suficiente de ojos, todo error se vuelve obvio».

Hoy, tanto si desarrollan software de código abierto como protegido, los programadores utilizan herramientas y enfoques que primero empleó la comunidad de código abierto. Pero más importante aún, cualquiera que utilice hoy software de internet ha experimentado estos principios. Cuando entras en sitios como Amazon, Facebook o Google, estás participando en el proceso de desarrollo de una manera desconocida en la era de los PC. No eres un «codesarrollador» al modo que imaginó Eric Raymond —no eres otro hacker que ofrece sugerencias sobre funciones y aporta código—. Pero sí eres un «beta tester» —alguien que prueba software que evoluciona constantemente, que no está acabado y que ofrece comentarios— a una escala nunca imaginada. Los desarrolladores de software de internet actualizan constantemente sus aplicaciones, probando características en millones de usuarios, midiendo su impacto y aprendiendo conforme avanzan.

Eric se dio cuenta de que algo estaba cambiando en la manera de desarrollar software, pero en 1997, cuando publicó *The Cathedral and the Bazaar*, aún no estaba claro que los principios que articuló se fueran a propagar más allá del software libre, más allá del desarrollo de software mismo, dando forma a sitios de contenidos como Wikipedia y a la larga permitiendo una revolución en la que los consumidores se convertirían en cocreadores de servicios como transporte (Uber y Lyft) y alojamiento (Airbnb) a la carta.

Cuando fui invitado a dar una charla en la misma conferencia de Würzburg, titulada «Hardware, Software, and Infoware», mi exposición fue muy diferente.[27] No solo me fascinaba Linux, sino también Amazon. Esta empresa se había edificado sobre la base de varias clases de software libre, incluido Linux, pero en lo que se refería a carácter me parecía fundamentalmente diferente a las clases de software que habíamos visto en las eras informáticas anteriores.

Hoy resulta obvio para todo el mundo que los sitios web son aplicaciones y que la web se ha convertido en plataforma, pero en 1997, la mayoría de la gente consideraba que el navegador era una aplicación. Si sabían algo más sobre la arquitectura de la web, es posible que consideraran que el servidor y el código y los datos asociados eran la aplicación.

El contenido gestionaba el navegador, de la misma manera que Microsoft Word gestiona un documento o Excel te permite crear una hoja de cálculo. Por el contrario, yo estaba convencido de que el contenido mismo era una parte esencial de la aplicación, y que la naturaleza dinámica de ese contenido estaba impulsando un patrón de diseño arquitectónico enteramente nuevo para la siguiente etapa más allá del software, lo que entonces llamé «infoware».

Mientras Eric se centraba en el éxito del sistema operativo Linux, y lo veía como alternativa a Microsoft Windows, a mí me fascinaba especialmente el éxito que tenía el lenguaje de programación Perl, que posibilitaba este nuevo paradigma en la web.

Perl fue creado inicialmente por Larry Wall en 1987 y distribuido libremente por las incipientes redes informáticas. En 1991, publiqué el libro de Larry, *Programming Perl* y en el verano de 1997 arranqué con la Conferencia Perl. La inspiración para esta conferencia me llegó por la conjunción fortuita de comentarios de dos amigos. A comienzos de 1997, Carla Bayha, suministradora de textos sobre informática para la cadena de librerías Borders, me había dicho que la segunda edición de *Programming Perl*, publicada en 1996, había sido uno de los libros top 100 para todas las categorías en Borders. Me pareció curioso que, a pesar de ese hecho, apenas había nada escrito acerca de Perl en ninguna publicación especializada en informática. Dado que tras Perl no había ninguna empresa, era prácticamente invisible para los entendidos que se interesaban por el sector.

Entonces, Andrew Schulman, autor del libro *Unauthorized Windows 95*, me dijo algo que me pareció igualmente curioso. En aquella época, Windows estaba emitiendo una serie de anuncios de televisión sobre la forma en que su nueva tecnología, llamada Active/X, «activa internet». Según Andrew, las versiones de demostración en aquellos anuncios habían sido realizadas en su mayoría con Perl. Así que me quedó claro que era Perl, no Active/X, lo que estaba en el meollo de la manera en que se suministraba el contenido web dinámico.

Me indigné. Decidí que necesitaba montar un poco de jaleo sobre Perl. De modo que, a principios de 1997, anuncié mi primera conferencia como ardid publicitario para atraer la atención de la gente. Y de eso también había ido a hablar en el Congreso de Linux de Würzburg.

En el ensayo que redacte más tarde basado en la charla, escribí: «Perl ha sido llamado la "cinta aislante de internet", e igual que la cinta aislante, se usa de maneras muy diferentes. Igual que un plató de rodaje que se sostiene con cinta aislante, un sitio web se monta y se desmonta

en un día, y requiere herramientas ligeras y soluciones rápidas, pero eficaces».

Consideré el enfoque de la cinta aislante como propiciador esencial del paradigma «infoware», en que el control sobre los ordenadores se realizaba a través de una interfaz de información, no una interfaz de software propiamente dicha. Un enlace web, tal como describí en su momento, era una manera de incorporar órdenes a un ordenador en documentos dinámicos escritos en lenguaje humano corriente, en lugar de, digamos, un menú desplegable, que incorporaba pequeñas porciones de lenguaje humano en un programa de software convencional.

La siguiente parte de la charla se centró en una analogía histórica que me iba a obsesionar durante los siguientes años. Me fascinaban los paralelos entre lo que el software de código abierto y los protocolos abiertos de internet estaban haciéndole a Microsoft, y la manera en que, años antes, habían desplazado a IBM Microsoft y un sector independiente del software.

Cuando entré en esta industria en 1978, esta se estaba quitando de encima el monopolio de IBM, no muy distinto a la posición que Microsoft ocuparía veinte años más tarde. El control de IBM sobre el sector se basaba en sistemas informáticos integrados en los que el software y el hardware estaban estrechamente acoplados. Crear un nuevo tipo de ordenador significaba inventar tanto hardware nuevo como un nuevo sistema operativo para controlarlo. Las pocas empresas independientes de software que existían debían elegir de qué proveedor de hardware iban a ser satélites, o transferir su software a múltiples arquitecturas de hardware, igual que muchos desarrolladores de teléfonos hoy día necesitan crear versiones separadas para iPhone y Android. Excepto que el problema era mucho peor. A mediados de la década de 1980, recuerdo hablar con uno de los clientes de mi negocio de consultoría de documentación, el autor de una librería de gráficos para servidores llamada DISSPLA (*display integrated software system and plotting language*), que me aseguró que debía mantener más de doscientas versiones diferentes de su software.

El ordenador personal de IBM, lanzado en agosto de 1981, lo cambió todo. En 1980, al darse cuenta de que se les escapaba el nuevo mercado de los microordenadores, IBM estableció un proyecto experimental en Boca Ratón, Florida, para desarrollar la nueva máquina. Tomaron una decisión crucial: para reducir gastos y acelerar el proceso, desarrollarían una arquitectura abierta utilizando componentes acordes con las normas del sector, incluido software con licencia de terceras partes.

El PC, como iba a llamarse al poco tiempo, fue un éxito inmediato cuando se lanzó en el otoño de 1981. Las proyecciones de IBM indicaban la venta de 250.000 unidades en los primeros cinco años.[28] Se rumoreaba que el primer día habían vendido 40.000,[29] y, en dos años, más de un millón llegaron a manos de los clientes.

Sin embargo, los ejecutivos de IBM no supieron entender las consecuencias de sus decisiones. En aquel momento, el software tenía un papel reducido en la industria informática, una parte necesaria pero menor de un ordenador integrado, a menudo combinado con este en lugar de vendido por separado. De modo que cuando llegó el momento de proporcionar un sistema operativo para la nueva máquina, IBM decidió obtener la licencia de Microsoft, lo que le dio a esta empresa el derecho de revender el software al segmento del mercado que IBM no controlaba.

El tamaño de aquel segmento estaba a punto de subir como la espuma. Debido a que IBM había publicado las especificaciones para la máquina, su éxito fue seguido por el desarrollo de decenas, y luego cientos de clones compatibles de PC. Los obstáculos para entrar en el mercado eran tan pocos que Michael Dell construyó su empresa del mismo nombre cuando todavía era un estudiante en la Universidad de Texas, ensamblando y vendiendo ordenadores desde su dormitorio en la residencia universitaria. La arquitectura del ordenador personal IBM se convirtió en el estándar, con el tiempo desbancó a otros diseños de ordenador personal y, en las siguientes dos décadas, a los miniordenadores y servidores.

A medida que los ordenadores personales eran construidos por cientos de fabricantes grandes y pequeños, IBM, sin embargo, fue perdiendo el liderazgo en el nuevo mercado. El software se convirtió en el nuevo sol alrededor del cual giraba la industria, y Microsoft se había convertido en la empresa más importante del sector informático.

Intel también se adjudicó un papel privilegiado a base de tomar decisiones valientes. Para poder estar segura de no asfixiarse con un solo proveedor, IBM había exigido que cada componente de la arquitectura de hardware abierta estuviera disponible por parte de, al menos, dos proveedores. Intel había acatado esta orden: dio la licencia de sus chips 8086 y 80286 a la rival AMD, pero, en 1985, con el lanzamiento del procesador 80386, tomaron la audaz decisión de hacer frente a IBM. Para ello, confiaron en que el mercado de los clones fuese lo suficientemente grande como para que la voluntad de IBM fuera invalidada por el mismo mercado. Pat Gelsinge, antiguo director tecnológico de Intel, me explicó cómo sucedió: «Votamos las cinco personas del comi-

té de gerencia. Hubo tres votos en contra y dos a favor. Pero Andy [Grove, consejero delegado de Intel] era uno de los dos, de modo que lo hicimos igualmente».

Esta es otra lección sobre el futuro. No se limita a suceder. La gente hace que suceda. Las decisiones individuales importan.

En 1998, la historia se había repetido en gran parte. Microsoft había utilizado su posición como proveedor único del sistema operativo para PC con el fin de establecer un monopolio de software de escritorio. Las aplicaciones informáticas se habían hecho cada vez más complejas y Microsoft levantaba barreras contra sus competidores. Ya no era posible que un programador único o una empresa pequeña tuvieran impacto en el mercado del software para PC.

Ahora, el software de código abierto y los protocolos abiertos de internet estaban desafiando ese dominio, por lo que los obstáculos para entrar en este mercado se estaban derrumbando. La historia puede que no se repita, pero sí que rima.

Los usuarios podían probar un producto nuevo de manera gratuita... e incluso más: podían crear su propia versión a medida y también gratuita. Era posible revisar el código fuente a gran escala por pares independientes, y si a alguien no le gustaba una característica, podía sumarle algo, sacarle algo o volver a implementarla. Si devolvían la característica corregida a la comunidad, podía adoptarse de forma muy amplia y rápidamente.

Y, además, debido a que los desarrolladores (al menos al principio) no estaban intentando competir por el lado comercial, sino que se centraban sencillamente en resolver problemas verdaderos, hubo espacio para la experimentación. Como ya se ha dicho a menudo, el software de código abierto te permite satisfacer el deseo de resolver un problema. Debido al paradigma de desarrollo distribuido y las nuevas características añadidas por los usuarios, los programas de código abierto «evolucionan» tanto como se diseñan. Y tal como escribí en el ensayo titulado *Hardware, Software and Infoware*, «la evolución no genera un solo ganador, sino diversidad».

Esa diversidad fue la razón de que las semillas del futuro se encontraran en el software libre y en internet, en lugar de en tecnologías ahora ya instituidas, ofrecidas por Microsoft.

Casi siempre ocurre que, si quieres ver lo que el futuro deparará, no tienes que observar las tecnologías ofrecidas por la corriente dominante, sino las de los innovadores que se salen del camino marcado.

La mayoría de las personas que lanzaron el sector del software de ordenador personal cuatro décadas atrás no eran emprendedores, eran chavales para quienes la idea de tener su propio ordenador era absurdamente emocionante. Programar era como una droga... No, era mejor que una droga, o que formar un grupo de rock, y por descontado era mejor que cualquier trabajo que pudieran imaginar. Y lo mismo sucede con Linux, el sistema operativo de código abierto utilizado hoy en día como sistema operativo de los PC por 90 millones de personas, y por miles de millones, ya que es el sistema operativo que hace que funcionen la mayoría de los sitios de internet grandes, y es el código que sustenta todos los teléfonos Android. ¿Sabes el título del libro de Linus Torvalds sobre cómo desarrolló Linux? *Just for fun*.[30]

La World Wide Web empezó de la misma manera. Al principio, nadie se lo tomó en serio como un lugar donde ganar dinero. Se trataba de la satisfacción de compartir nuestro trabajo, la descarga de adrenalina que significaba hacer clic sobre un enlace y conectar con otro ordenador al otro lado del mundo, y construir destinos similares para nuestros iguales. Todos éramos entusiastas. Algunos de nosotros también emprendedores.

Sin duda, son aquellos emprendedores —gente como Bill Gates, Steve Jobs y Michael Dell en la época de los ordenadores personales; Jeff Bezos, Larry Page, Sergey Brin y Mark Zuckerberg en la era de la web— quienes vieron que este mundo impulsado por la pasión por los descubrimientos y el intercambio podía convertirse en la cuna de una nueva economía. Encontraron financiadores, transformaron el juguete en una herramienta y construyeron negocios que transformaron un movimiento en una industria.

La lección está clara: trata la curiosidad y el asombro como la guía hacia el futuro. Ese asombro puede significar que aquellos locos entusiastas están viendo algo que tú no ves... aún.

La enorme diversidad de software que había crecido entre software libre se reflejaba en los éxitos de ventas que impulsaban mi editorial.

Perl no era el único. Muchos de los libros sobre tecnología más exitosos de la década de 1990, libros que yo publiqué y con títulos que solo a un programador le gustarían —*Programming Perl, Learning the Vi Editor, Sed & Awk, DNS and Bind, Running Linux, Programming Python*— trataban de software escrito por personas y distribuido libremente por internet. La web misma pasó a ser de dominio público.

Me di cuenta de que muchos de los autores de estos programas no se conocían. La comunidad del software libre que había confluido alrededor de Linux no interactuaba mucho con la gente de internet. Debido a mi posición como editor tecnológico, me movía entre ambos círculos, así que decidí reunirlos, pues pensaba que necesitaban verse a sí mismos como parte del mismo relato.

En abril de 1998, organicé un evento que llamé en un principio «Conferencia del Freeware» para reunir a los creadores de programas de software libre más importantes.

El momento fue ideal. En enero, la destacada empresa de Marc Andreessen, Netscape, creada para comercializar un navegador de internet, había decidido ofrecer el código fuente de su innovación como proyecto de software libre bajo el nombre de Mozilla. Debido a la presión competitiva por parte de Microsoft, que había construido su propio navegador y lo había cedido gratuitamente (pero *sin* código fuente) para «dejar sin aire a Netscape», estos no tuvieron más remedio que volver a las raíces del software libre de la web.[31]

En el encuentro, que tuvo lugar en el Stanford Court Hotel (ahora llamado Garden Court) de Palo Alto, reuní a Linus Torvalds, Brian Behlendorf (uno de los fundadores del proyecto de servidor Apache), Larry Wall, Guido van Rossum (creador del lenguaje de programación Python), Jamie Zawinski (desarrollador jefe del proyecto Mozilla), Eric Raymond, Michael Tiemann (fundador y consejero delegado de Cygnus Solutions, una empresa que comercializaba herramientas de programación de software libres), Paul Vixie (autor y mantenedor de BIND [Berkeley Internet Name Daemon], el software tras el sistema de nombres de dominio de internet) y Eric Allman (autor de Sendmail, el software que enviaba la mayoría del correo electrónico de internet).

En la reunión, uno de los temas que surgió fue el nombre de «software libre». El movimiento de software libre de Richard Stallman se había granjeado muchos enemigos con su propuesta aparentemente radical de que todo código fuente debe entregarse libremente, porque es inmoral no hacerlo. Aún peor, mucha gente había interpretado que los

desarrolladores de software libre eran contrarios a su uso comercial. En el encuentro, Linus Torvalds observó: «No me di cuenta de que *free* tiene dos significados en inglés: "libre" y "gratis"».

Linus no era el único que tenía diferentes conceptos sobre lo que significaba *free*. En otra reunión, Kirk McKusick, jefe del proyecto Berkeley Unix, que había desarrollado muchas de las características y utilidades clave de Unix que se habían incorporado a Linux, me había dicho: «A Richard Stallman le gusta decir que el *copyright* (los derechos de autor) es el mal, de modo que necesitamos una nueva interpretación llamada *copyleft*. Aquí en Berkeley utilizamos *copycentral*, es decir, le decimos a la gente que vaya a Copy Central [la copistería local] y hagan copias». El proyecto Berkeley Unix, que me había acercado al sistema operativo en 1983, formaba parte de una larga tradición académica de intercambio de conocimientos. El código fuente se ofrecía para que la gente pudiera sumar, y eso incluía el uso comercial. El único requisito era la atribución.

Bob Scheifler, director del proyecto Sistema de Ventanas X de MIT, seguía la misma filosofía. El Sistema de Ventanas X había arrancado en 1984, y cuando me tropecé con él en 1987, se estaba convirtiendo en el sistema de ventanas estándar para Unix y Linux, adoptado y adaptado por prácticamente cada proveedor. Mi empresa desarrolló una serie de manuales de programación para X que usaban las especificaciones de MIT como base, reescribiéndolas y ampliándolas, y luego concediendo licencias a empresas que entregaban nuevos sistemas Unix y basados en X. Bob me animó: «Esto es exactamente lo que queremos que las empresas hagan —me dijo—. Estamos sentando las bases y queremos que todos construyan encima».

Larry Wall, creador de Perl, fue otro de mis mentores en la reflexión sobre el software libre. Cuando le pregunté por la razón por la que había hecho que Perl fuera software libre, me explicó que el trabajo de otros había generado tanto valor que se sintió obligado a devolverlo de alguna manera. Larry también citó una variación de la clásica observación de Stewart Brand: «La información no quiere ser libre. La información quiere ser valiosa». Como muchos otros autores de software libre, Larry había descubierto que una forma de hacer más valiosa su información (es decir, su software), era ofreciéndola. Fue capaz de aumentar su utilidad no solo para sí mismo (porque otros que la retomaron hicieron cambios y mejoras que él podía usar), sino también para cualquiera que la utilice, porque cuando el software se hace más ubicuo, puede asumirse como base para un trabajo adicional.

Sin embargo, también me quedaba claro que los creadores de software exclusivo, incluidos aquellos como Microsoft, que eran considerados inmorales por la mayoría de los defensores del software libre, habían descubierto que, al restringirla, podían hacer que su información resultase valiosa. Microsoft había generado un enorme valor para sí misma y para sus accionistas, pero era al mismo tiempo la facilitadora clave de la computación personal ubicua, una precursora de las redes de computación globales de la actualidad. Y esto se traducía en valor para toda la sociedad.

Observé que Larry Wall y Bill Gates tenían mucho en común. Como creadores (si bien con una multitud de contribuidores) de un corpus de trabajo intelectual, habían tomado decisiones estratégicas sobre cómo maximizar su valor. La historia ha demostrado que ambas estrategias pueden funcionar. La cuestión clave para mí fue cómo maximizar la creación de valor para la sociedad, en lugar de que una persona o una empresa se limiten a capturar valor. ¿Cuáles eran las condiciones para que ofrecer software libre fuera una mejor estrategia que mantener su exclusividad?

Esta cuestión se ha repetido, más ampliamente, a lo largo de toda mi carrera: ¿cómo puede un negocio crear más valor para la sociedad del que captura para sí mismo?

¿Qué hay en un nombre?

Mientras lidiábamos con el nombre «software libre», se propusieron varias alternativas. Michael Tiemann dijo que Cygnus había empezado a utilizar la palabra *sourceware*. Pero Eric Raymond se mostró a favor de «código abierto», un nuevo término acuñado tan solo seis semanas antes por Christine Peterson del Foresight Institute, un *think tank* de nanotecnología, durante una reunión convocada por Larry Augustin, consejero delegado de una empresa filial de Linux llamada VA Linux Systems.

A Eric y a otro desarrollador y activista del software libre, Bruce Perens, les había entusiasmado tanto el nuevo término de Christine que habían formado una entidad sin ánimo de lucro llamada Open Source Initiative para acomodar en una especie de metalicencia las diversas licencias de software libre que se estaban utilizando. Pero, por el momento, el término era bastante desconocido.*

* *Open source* es el término inglés para código abierto (*N. del T.*).

No a todo el mundo le gustaba: «Suena demasiado a "open sores"»,* comentó un participante, pero todos estuvimos de acuerdo en que había serios problemas con el nombre «software libre» y que la adopción generalizada de un término nuevo podría ser un importante paso adelante. De modo que lo sometimos a votación. «Código abierto» ganó cómodamente a *sourceware* y todos acordamos utilizar en adelante la nueva expresión.

Fue un momento importante, porque había organizado para el final del día una conferencia de prensa con reporteros de *The New York Times, The Wall Street Journal, San Jose Mercury News* (en aquel momento el diario de Silicon Valley), *Fortune, Forbes* y muchas otras publicaciones nacionales. Debido a que había establecido relaciones con muchos de aquellos periodistas durante la época, a comienzos de la década de 1990, en que promovía la comercialización de internet, se presentaron a la conferencia a pesar de no saber cuál iba a ser la noticia.

Coloqué a todos los participantes del encuentro en fila, delante de los reporteros, y expliqué una historia que ninguno de ellos había oído jamás. Fue algo así:

> Cuando oyes la expresión «software libre», crees que se trata de un movimiento rebelde, hostil hacia el software comercial. He venido aquí a explicar que todas las empresas de gran tamaño —incluidas las vuestras— ya utilizan a diario software libre. Si tu empresa tiene un nombre de dominio —digamos nytimes.com o wsj.com o fortune.com—, ese nombre únicamente funciona gracias a BIND, el software escrito por este hombre, Paul Vixie. El servidor que utilizas es probable que sea Apache, creado por un equipo cofundado por Brian Behlendorf, ahí sentado. El sitio web también hace un uso intensivo de lenguajes de programación como Perl y Python, escritos por Larry Wall, aquí, y Guido van Rossum, aquí. Si envías un correo electrónico, seguro que fue transferido a su destino por Sendmail, escrito por Eric Allman. Y todo esto sin tener en cuenta Linux, del que todos habéis oído hablar y que fue escrito por Linus Torvalds, aquí también.
>
> Y he aquí lo increíble: todos estos tipos tienen una cuota de mercado dominante en categorías importantes de software de internet sin que ningún inversor de riesgo les dé dinero, sin tener detrás ninguna empresa, tan solo por la fuerza que da crear un software superior y ofrecerlo a cualquiera

* *Open source* se pronuncia exactamente igual que *open sores*, «heridas abiertas» en castellano (*N. del T.*).

que quiera utilizarlo o que esté dispuesto a sumar con sus aportaciones al proyecto.

Debido a que el término «software libre» tenía unas connotaciones negativas, nos hemos reunido aquí hoy con el objetivo de decidir adoptar un nuevo nombre: «software de código abierto».

Durante las siguientes semanas, me entrevistaron varias decenas de veces y expliqué que todas las piezas fundamentales de la infraestructura de internet eran de «código abierto». Aún recuerdo la incredulidad y sorpresa que pude constatar en los entrevistadores, pero en pocos días el término había sido aceptado, con lo que se configuró un nuevo mapa. Es más: hoy en día nadie se acuerda siquiera de que el evento se llamó inicialmente Conferencia del Freeware, pues poco después la gente se refirió al encuentro como «la conferencia del código abierto».

Esta es una lección clave sobre cómo ver el futuro: reunir a gente que ya vive en él. El escritor de ciencia ficción William Gibson observó, como es sabido: «El futuro ya está aquí. Lo que pasa es que aún no está distribuido uniformemente».[32] Los primeros desarrolladores de Linux e internet vivían en un futuro que ya estaba en vías de llegar al resto del mundo. Reunirlos fue el primer paso para redibujar el mapa.

¿Estás mirando el mapa o la carretera?

Otra lección: entrénate para detectar que, efectivamente, estás mirando el mapa, en lugar de la carretera. Compara constantemente los dos y presta especial atención a todas las cosas que veas y que faltan en el mapa. Así es como fui capaz de advertir que la narrativa sobre el software libre propuesta por Richard Stallman y Eric Raymond había ignorado el software libre más exitoso de todos, el que sustenta internet.

Tu mapa debería ayudarte a ver, no debería sustituir el acto de ver. Si sabes que viene una curva, podrás estar preparado para cuando llegue. Si la curva no llega cuando la esperas, quizás estés en la carretera equivocada.

Mi propio adiestramiento en mantener la vista en la carretera empezó en 1969, cuando tenía tan solo quince años. Me hermano Sean,

que contaba con diecisiete, conoció a un hombre llamado George Simon, que iba a tener un papel importante en mi vida intelectual. George era líder de tropa de los Explorer Scouts, el nivel adolescente de los Boy Scouts —nada más y, sin embargo, nada menos—. El punto fuerte de la tropa, a la que se unió Sean, era la comunicación no verbal.

Más tarde, George acabó dando talleres en el Esalen Institute, que era al movimiento del potencial humano de la década de 1970 lo que Googleplex o el Infinite Loop de Apple es al Silicon Valley de hoy. Yo enseñé en Esalen con George cuando apenas había salido del instituto y desde entonces sus ideas han influido profundamente en mi pensamiento.

George tuvo la idea, aparentemente disparatada, de que el lenguaje en sí mismo era una especie de mapa. El lenguaje da forma a lo que somos capaces de ver, y a cómo lo vemos. George había estudiado la obra de Alfred Korzybski, cuyo libro *Science and Sanity*, publicado en 1933, había vuelto a ponerse de moda en la década de 1960, en gran parte gracias al trabajo de S. I. Hayakawa, estudiante de Korzybski.

Korzybsky creía que la realidad es fundamentalmente incognoscible, dado que *lo que es* siempre se ve mediado por nuestro sistema nervioso. Un perro percibe un mundo muy distinto del que percibe un ser humano, e incluso diferentes seres humanos experimentan el mundo de forma muy diversa. Pero, igualmente importante, nuestra experiencia viene determinada por las palabras que usamos.

Tuve una experiencia al respecto años más tarde, cuando me mudé a Sebastopol, una pequeña ciudad del norte de California, donde tenía caballos. Antes de vivir allí, cada vez que miraba un prado, veía algo que llamaba «hierba». Pero con el tiempo, aprendí a distinguir entre avena, centeno, dáctilo y alfalfa, así como otros tipos de forraje como la algarroba. Ahora, cuando veo un prado, veo todos los tipos, así como otros cuyos nombres desconozco. Disponer de lenguaje para las hierbas me ayuda a verlas más plenamente.

El lenguaje también puede engañarnos. A Korzybski le gustaba demostrar a la gente cómo las palabras determinaban su experiencia del mundo. En una famosa anécdota, compartió con la clase una lata de galletas envueltas en papel marrón.[33] Mientras todos engullían las galletas (algunos incluso repitieron), sacó el papel de una de las que había repartido y mostró que eran para perros. Varios estudiantes salieron corriendo de clase para ir a vomitar. La lección de Korzybski: «Acabo de demostrar que la gente no solo come comida, sino también palabras, y que el sabor de estas últimas a menudo supera el de la primera».

Korzybski argumentó que muchas aberraciones psicológicas y sociales pueden verse como problemas del lenguaje. Pongamos por ejemplo el racismo: se basa en términos que niegan aspectos básicos de la humanidad de la gente a la cual describe. Korzybski pidió a sus oyentes que fueran visceralmente conscientes del proceso de abstracción, por el cual una serie de afirmaciones sobre la realidad —mapas que nos pueden guiar, pero que también nos pueden engañar— se transforman en la realidad.

Comprender esto parece especialmente importante frente a acontecimientos como el que se produjo con las *fake news*, o noticias falsas, que influyeron notablemente en las elecciones presidenciales de Estados Unidos en 2016. No se trató tan solo de los ejemplos más notorios, como la red de niños esclavos que supuestamente dirigía la campaña de Hillary Clinton desde una pizzería de Washington D. C., sino la selección sistemática y cada vez más algorítmica de noticias para satisfacer y amplificar las ideas preconcebidas de la gente. Sectores enteros de la población están ahora dirigidos por mapas enormemente divergentes. ¿Cómo vamos a resolver los problemas más urgentes del mundo si ni siquiera intentamos crear mapas que reflejen la carretera real que tenemos delante, sino que nos dirigimos hacia metas políticas o empresariales?

Después de trabajar con George durante unos años, aprendí a detectar casi instintivamente cuándo me enfrascaba en las espirales de palabras que utilizamos para la realidad y cuándo prestaba atención a lo que estaba experimentando en verdad, o más aún, cuándo iba más allá de lo que estaba experimentando y alcanzaba la meta en sí. En el momento en el que nos enfrentamos a lo desconocido, cierta receptividad cultivada, una apertura de miras hacia lo ignoto, conduce a mejores mapas que simplemente intentar superponer mapas anteriores sobre lo que es nuevo.

Es precisamente esta capacidad de mirar al mundo directamente, y no limitarnos tan solo a reorganizar los mapas, lo que se encuentra en el núcleo de la labor científica original, y, como intento argumentar en este libro, también en la actividad empresarial y la tecnología.

Como relata en su autobiografía *¿Está usted de broma, Sr. Feynman?*, el conocido físico Richard Feynman se quedó consternado al ver cuántos estudiantes de una clase que visitó durante su año sabático en Brasil eran incapaces de aplicar lo que habían aprendido. Inmediatamente después de una charla sobre la polarización de la luz, con demostraciones en las que empleaba tiras de película polarizadora, formuló una pregunta cuya respuesta podía contestarse mirando, a través de la

película, la luz reflejada en la bahía que se veía fuera. A pesar de su capacidad para recitar la fórmula correcta si se les preguntaba directamente (el ángulo de Brewster), nunca se les ocurrió que esta proporcionaba una manera de responder a la pregunta planteada. Habían aprendido los símbolos (los mapas), pero no eran capaces de relacionarlos suficientemente con la realidad subyacente para poder utilizarlos en la vida real.[34]

«No sé qué le ocurre a la gente. No aprenden entendiendo. Aprenden de otra manera. De memoria, supongo —escribió Feynman—. Su conocimiento es tan frágil.»[35]

Reconocer cuándo uno se encuentra encallado en las palabras, mirando el mapa en lugar de la carretera, es algo sorprendentemente difícil de aprender. La clave es recordar que esta práctica se basa en la experiencia: no se aprende leyendo un libro; hay que practicarlo. Como veremos en el próximo capítulo, esto es lo que hice en mi constante esfuerzo por entender la relevancia del software de código abierto.