# El código como lienzo: La importancia de la estética del software en la era de la mercancía

**Justin Vasquez Tellez**

**Bryan Morales Calderón**

En la formación académica de la ingeniería, se nos condiciona a pensar en el software bajo una lógica principalmente utilitaria. Se nos enseña que un buen programa es aquel que optimiza recursos, reduce tiempos de ejecución, resuelve un problema con alta precisión y que carece de bugs. Bajo esta premisa, el código se percibe como una herramienta invisible, un medio para un fin comercial o funcional que pierde su valor una vez completada su tarea, ya que ese es su único propósito. Sin embargo, esta visión es profundamente reduccionista y omite una dimensión fundamental de nuestra labor: la capacidad del software para ser un medio para la expresión creativa y una obra cultural en sí misma.

Si entendemos el arte como una manifestación que organiza la experiencia humana y genera significado, es imposible ignorar que, hasta cierto punto, tiene relación con la programación. Al programar, no solo estamos dando instrucciones a una máquina, sino que estamos construyendo estructuras lógicas, diseñando según nuestras diferentes formas de pensamiento y a la vez moldeando la forma en que las personas entienden e interactúan con la realidad digital. El software deja de ser una simple mercancía para convertirse en una extensión de la literatura y la arquitectura moderna, donde la elegancia o simpleza de un algoritmo funcional puede llegar a conmover como la métrica de un poema.

## La elegancia como criterio de calidad: El legado de Donald Knuth

Para mantenernos en la idea del software como arte, es importante recurrir a la figura de Donald Knuth. En su histórica conferencia *Computer Programming as an Art* presentada originalmente en 1974, Knuth (1974/2007) cuestionó la visión de la programación como una actividad puramente técnica. Según su planteamiento, la programación aspira a la belleza a través del estilo y la claridad. Knuth sostiene que un programador que se preocupa por la estética de su código

no solo produce un software más robusto, sino que, al crearlo, experimenta una satisfacción creativa similar a la de un músico componiendo una sinfonía.

Desde nuestra perspectiva, la belleza en el software se manifiesta en la legibilidad y la armonía que se encuentra dentro del sistema. Hay algo, por así decirlo, artístico en un código que, siendo complejo en su función, logra ser simple en su lectura. Cuando nos enfrentamos a un proyecto donde las funciones están conectadas de manera lógica y fluida, estamos ante una obra de artesanía digital, en comparación a cuando el código es un desastre desordenado e inentendible (aunque funcional). La utilidad de mercado suele premiar la rapidez del lanzamiento, pero la calidad artística del software se sostiene en aquello que no siempre es visible para el usuario: la arquitectura del código. Un programa "bello" es aquel que respeta a ambos, tanto al lector humano como a la máquina que lo realiza.

**El software como arquitectura de la experiencia: Videojuegos y arte generativo**

Para comprender el software como una obra cultural, no podemos ignorar partes en donde la funcionalidad y la estética son inseparables. Un ejemplo que en la actualidad y en el futuro va a seguir siendo importante es el desarrollo de videojuegos independientes (*indies*), donde el código no solo sostiene las mecánicas, sino que tiene un papel importante en la parte emocional del creador. En proyectos como estos, el programador actúa como un arquitecto de mundos, y como estos videojuegos son creados por un grupo limitado de personas, es algo muy personal, en comparación a un trabajo hecho por una gran empresa. Como señala Christiane Paul (2023) en su análisis sobre el arte digital, el software permite crear instalaciones interactivas donde el espectador ya no es un sujeto pasivo, sino un co-creador de la experiencia a través de los algoritmos de respuesta.

Desde nuestro punto de vista, esto demuestra que el código tiene un *estilo* propio que se puede comparar al de una película. Así como un director de cine elige un ángulo de cámara para transmitir una emoción, un programador elige un

algoritmo para transmitir su idea, ya sea de forma simple o compleja. La belleza no está solo en el producto final, sino en la lógica o procedimiento que lo sustenta.

**El diseño por números: La estética de la limitación**

Podríamos considerar también la obra de John Maeda (1999), quien en *Design by Numbers* explora cómo la programación permite a los diseñadores romper las barreras de las herramientas comerciales tradicionales. Al crear nuestras propias herramientas mediante el código, recuperamos esa parte artística.

En lugar de usar un software cerrado que impone sus propias reglas, se crea el software para imponer las nuestras. Consideramos que esta libertad es lo que vuelve a la programación tan interesante.

**La analogía literaria: El código como narrativa**

Otra forma de expandir nuestra visión del software es compararlo con la literatura. Existe una relación entre la figura del escritor y la del programador: escribimos para ser leídos por otros humanos. Un sistema de software bien documentado y estructurado cuenta una historia sobre cómo se resolvió un problema, cuáles fueron las decisiones del autor y qué prioridades tenía en ese momento histórico.

Cuando dejamos de ver el código como algo que se hace solo para cumplir su tarea, se siente desechable, pero si lo tratamos como una obra que merece ser preservada, estamos otorgándole un valor personal. El software de los sistemas operativos pioneros hoy se estudia como piezas históricas no solo por lo que hicieron, sino por cómo fueron escritos bajo limitaciones extremas. Eso demuestra la creatividad que tuvieron los programadores al conseguir crear el código a pesar de tener herramientas tan básicas; esa es la verdadera importancia: dejar una huella de pensamiento humano en un medio digital que, de otro modo, sería puramente comercial.

## El código como discurso de expresión social

Avanzando en esta reflexión, el software no solo es una obra estética, sino también un acto político. Como argumentan Cox y McLean (2013) en su obra *Speaking Code*, la programación es una práctica que organiza posibilidades e impone estructuras en el mundo. El acto de escribir código nunca es neutral; cada decisión técnica conlleva una carga política sobre cómo debería funcionar la sociedad, ya que define qué es posible, qué optimizar, qué modelar y en última instancia, quiénes son los sujetos habilitados a ser el público meta para interactuar con el programa.

No solo la importancia recae en que un código funcione; también importa la accesibilidad, la privacidad o la facilidad para el usuario. Poniendo de ejemplo a arquitectos que logran diseñar una ciudad o lugar hermoso, en el caso de que no tenga accesibilidad o no tomara en cuenta las necesidades de los usuarios con necesidades especiales, solo ciertas personas podrían disfrutarla, ya que, por ejemplo, si hay muchas escaleras, no hay rampas, un edificio enormemente alto sin ascensor, los usuarios no tienen cómo transportarse libremente y, por ende, no tienen cómo disfrutar esa obra realmente.

Con base en esta información, podemos decir que el movimiento del software libre se presenta como una forma de resistencia cultural. Cuando el código se libera, deja de ser una propiedad privada. Aquí, la autoría se vuelve colaborativa y el estilo de un proyecto se define por la comunidad que lo mantiene. Ver el código como una obra digna de ser preservada cambia nuestra relación con la tecnología: dejamos de ser simples consumidores para ser partícipes de una cultura tecnológica duradera.

## La tensión con la inteligencia artificial y la nueva autoría

Uno de los debates más urgentes en la actualidad es la intervención de la inteligencia artificial en la creación de software. ¿Puede algo como un código generado por un modelo de lenguaje ser considerado arte? Nuestra opinión es que, si bien la IA puede generar fragmentos de código funcionales, basarse completamente en su ayuda para hacer el código completo carece de la intención

que define a la obra artística y, además, ¿qué diferenciaría a un programador que se basa completamente en la ayuda de la IA para su código de una persona común haciendo lo mismo? El arte requiere de una visión humana que busque comunicar algo.

La IA es una herramienta técnica avanzada, pero ¿se puede considerar que la autoría se mantiene en quien forma el concepto y le dice a alguien más que lo haga? Además, dado que los creadores o dueños de las IA tienen influencia en el contenido que se genera, en este caso, si la herramienta no permite hacer todo lo que usted quiere crear, ¿realmente se tiene la misma libertad creativa si alguien más limita lo que se puede hacer? En este sentido, dejar el trabajo de la creación total a la IA es renunciar a nuestra voz, creatividad personal y expresión artística, y reducir el software a una simple mercancía automatizada y sin alma.

**La ingeniería con imaginación**

En la formación de un ingeniero es común escuchar que su producto o trabajo tiene que ser lo más eficiente posible; aunque eso es cierto y siempre se debe tender a mejorar, no debe reducirse solo a eso. La verdadera innovación o diferencia entre un ingeniero y otro está en su toque personal. En si su producto está pensado y hecho con cariño, en vez de simplemente realizarlo para entregar o para “hacer lo mínimo aceptable” como un producto comercial, eso es lo que lo diferencia. Como sugiere Malcolm McCullough (1996), la capacidad de pensar críticamente sobre el impacto humano de lo que construimos es lo que separa a un técnico de un verdadero ingeniero con visión artística.

El rigor y la creatividad no son opuestos; se necesita de ambos para crear sistemas o productos que no solo sean resolver un problema o terminar un trabajo; también deben tener un significado profundo, que se note el cariño con el que está hecho y el empeño que ha tomado hacerlo.

En conclusión, reconocer el software como una forma de arte es un acto de rebeldía contra la mercantilización absoluta de la tecnología. Al valorar el código por su estilo, su elegancia y su capacidad expresiva, estamos humanizando la ingeniería. Como futuros profesionales, tenemos la responsabilidad de no ver

nuestras creaciones simplemente como productos con fecha de caducidad, sino como piezas de un rompecabezas cultural que define nuestra época. El software es un espejo de nuestra persona, sociedad y, como tal, merece ser tratado con el respeto y la dedicación que se le otorga a cualquier obra maestra de la humanidad.

**Referencias**


Cox, G., & McLean, A. (2012). *Speaking code: Coding as aesthetic and political expression*. MIT Press. https://doi.org/10.7551/mitpress/8193.001.0001

Knuth, D. (2007). *Computer programming as an art. AMC.* (Obra original publicada en 1974). https://doi.org/10.1145/1283920.1283929

Maeda, J. (1999). *Design by numbers*. MIT Press. https://archive.org/details/designbynumbers0000maed/page/8/mode/2up

McCullough, M. (1996). *Abstracting craft: The practiced digital hand*. MIT Press. https://archive.org/details/abstractingcraft0000malc

Paul, C. (2023). *Digital art* (4th ed.). Thames & Hudson. https://openlibrary.org/books/OL27504948M/Digital_Art