Artículo científico
ISSN 2965-1557
http://doi.org/10.1590/2965-1557.036.e202430897

Revista de Filosofia Aurora
periodicos.pucpr.br/aurora
PUCPRESS

Los modos de existencia de los objetos computacionales

On the modes of existence of computational objects

Javier Blanco [a]
Córdoba, Argentina
[b] Universidad Nacional de Córdoba

Como citar: BLANCO, Javier. Los modos de existencia de los objetos computacionales. *Revista de Filosofia Aurora*, Curitiba: PUCPRESS, v. 36, e202430897, 2024. DOI: http://doi.org/10.1590/2965-1557.036.e202430897.

Resumen

En este trabajo, me propongo recuperar la relevancia de la propuesta filosófica de Simondon para analizar las tecnologías computacionales, resaltando la vigencia de esas ideas formuladas en un contexto tecnológico disímil, tratando a la vez de delinear algunas divergencias originadas por la aparición de novedades radicales en la tecnología misma. En este sentido, la idea es replicar el método o gesto de Simondon para comprender los objetos técnicos, oyendo qué tienen para decir, estableciendo un diálogo con ellos e intentando construir una mirada conceptual que sea fiel a su singularidad. En el caso de las computadoras y la computación es posible que no nos encontremos ya frente a (solo) un nuevo objeto técnico, sino que sus peculiaridades las convierten en algo mucho más multifacético y complejo. Otro de los objetivos de esta indagación será evaluar esta posibilidad.

Palabras-clave: Simondon. Tecnología. Computación. Programación. Machine learning.

[a] PhD em Informática pela Technische Universiteit Eindhoven, e-mail: javier.blanco@unc.edu.ar

Abstract

In this work, I attempt to recover the relevance of Simondon's philosophical approach to analyze computational technologies, highlighting the validity of those ideas formulated in a different technological context, trying at the same time to outline some divergences caused by the appearance of radical novelties in the technology itself. In this sense, our idea is to replicate Simondon's method or gesture to understand technical objects, listening to what they have to say, establishing a dialogue with them and trying to build a conceptual view that is faithful to their uniqueness. In the case of computers and computing, it is possible that we are no longer faced with (only) a new technical object, but rather their peculiarities turn them into something much more multifaceted and complex. Another objective of this investigation will be to evaluate this possibility.

Keywords: *Simondon. Technology. Computation. Programming. Machine learning.*

La apuesta por una cultura técnica en el siglo XXI

Ya en la década de 1950 Simondon señalaba el desacople entre la evolución técnica y la cultural, la creciente brecha en las respectivas velocidades que hoy suele conducir a actitudes tecnocráticas y en muchos casos a una reacción humanista conservadora. Habiendo transcurrido casi setenta años, la necesidad y urgencia de desarrollar una cultura técnica como apuesta política y antropológica solo se ha incrementado. Podría argumentarse que la humanidad pudo sobrevivir o hasta progresar sin que esto haya ocurrido, lo cual probaría que un desarrollo profundo y amplio de una cultura técnica no habría sido tan urgente ni tan necesario, pero eso sería un signo más de incomprensión de este tiempo, o un síntoma más de la disrupción encarnada en el actual momento tecnológico. Es posible que muchos de los principales problemas sociales y políticos del presente tengan algunas de sus causas en esta ausencia, o, dicho de otra manera, que muchas de estas cuestiones encontrarán nuevas respuestas en la necesaria conformación y evolución de dicha cultura, proceso que suponemos incipientemente en marcha.

El momento tecnológico analizado por Simondon se ha transformado drásticamente, no obstante consideramos que muchas de sus ideas y conceptos son relevantes para dar cuenta del presente y de los futuros posibles. Claro que esto requiere por un lado poner a prueba la pertinencia de sus análisis en este nuevo contexto, y por otro revisar las ideas y las preguntas que articularon sus teorías, reconsiderando la importancia relativa de cada noción y construyendo otras que den cuenta de las múltiples transformaciones tecnológicas. Todo esto es particularmente difícil, ya que Simondon construye un andamiaje conceptual de amplio espectro, que va desde los procesos de individuación, desde una novedosa concepción ontogenética, hasta las formas de existencia de los objetos técnicos y la drástica transformación cultural que encarnan. Poder usar esta perspectiva en el presente no puede ser una simple aplicación de sus nociones a nuevos objetos técnicos, lo que entraría en contradicción con su propio método de trabajo, sino que implica construir o encontrar los propios conceptos que emerjan del análisis de estas nuevas formas técnicas.

En este trabajo, me propongo recuperar la relevancia de la propuesta filosófica de Simondon para analizar las tecnologías computacionales, resaltando la vigencia de esas ideas formuladas en un contexto tecnológico disímil, tratando a la vez de delinear algunas divergencias originadas por la aparición de novedades radicales en la tecnología misma. En este sentido, la idea es replicar el método o gesto de Simondon para comprender los objetos técnicos, oyendo qué tienen para decir, estableciendo un diálogo con ellos e intentando construir una mirada conceptual que sea fiel a su singularidad. En el caso de las computadoras y la computación es posible que no nos encontremos ya frente a (solo) un nuevo objeto técnico, sino que sus peculiaridades las convierten en algo mucho más multifacético y complejo. Otro de los objetivos de esta indagación será evaluar esta posibilidad.

Explorar los conceptos simondonianos volviendo a considerar sus alcances y límites para dar cuenta de los sistemas computacionales permite también una mirada meta-teórica diferencial que, creemos, ilumina a la vez la acelerada evolución tecnológica y la accidentada evolución filosófica del último siglo. El (des)acople entre ambas formas de evolución es un fenómeno fundamental para analizar el presente y, sobre todo, para orientar futuros. La pregunta meta-teórica acerca de la posibilidad misma de comprender nuestro propio momento tecnológico adquiere cada vez más relevancia mientras que su respuesta se vuelve más incierta.

Hay algunos trabajos y conferencias que abordan la pregunta por la computadora o los programas desde la perspectiva de la filosofía de Simondon, pero a mi entender el tema esta lejos de haber encontrado respuestas adecuadas y menos aún exhaustivas. Podría darse el caso de que un pensamiento desarrollado a mediados del siglo XX no pueda trasladarse al presente, y que intentar un análisis simondoniano de la computación sea un ejercicio intelectual ocioso o condenado al fracaso. Creo, por el contrario, que muchos de los problemas que abordó entonces siguen siendo cruciales hoy, incluso más que entonces: la alienación, la incomprensión del objeto técnico, la escisión creciente entre cultura y técnica al mismo tiempo que aumenta la interdependencia entre ambas, la pregunta por el lugar del humano en los conjuntos técnicos

(u hoy en los ambientes o plataformas). El recorrido teórico necesario para reformular estas preguntas es largo, complejo y de resultado incierto. Involucra también la formulación de otras preguntas específicas al acelerado desarrollo de las tecnologías cognitivas actuales.

Tres libros que abordan con distintos énfasis la cuestión del software desde la perspectiva de Simondon son los de Coline Ferrarato (2020), Bernhard Rieder (2020) y Simon Mills (2016). Hay muchas ideas valiosas en los tres, exploran las ventajas y dificultades de analizar al software como objeto técnico revisando y adecuando sobre todo las ideas de MEOT (Simondon, 2007) en este nuevo contexto, aunque también refieren a algunos conceptos de otros trabajos, como la idea de individuación, de información y de allagmática. No buscamos sin embargo aquí establecer un diálogo con estos trabajos sino con las tecnologías computacionales en sí, comprender conceptualmente las diferentes dimensiones técnicas implicadas en su desarrollo, tratando de esbozar cómo podría pensarse un mecanología del siglo XXI. Para ellos buscaremos, tal como propone Simondon (2009) en la introducción, individuar el pensamiento junto a la veloz individuación actualmente en marcha de los sistemas computacionales, acoplarse a su ontogénesis con la expectativa, más necesaria en este contexto que nunca, de construir una epistemología allagmática, un conocimiento de y desde las operaciones.

Una crítica, sin dudas válida, planteada por Bernhard Rieder a los estudios de cultura técnica aplicada a la programación, señala un foco casi exclusivo en los trabajos de Turing y Shannon para dar cuenta de la idea de computación y de información respectivamente, sin dedicarle casi consideraciones a la evolución de las técnicas computacionales (e informacionales) sobre todo desde la década del '80 al presente. En particular el desarrollo de las computadoras personales y las redes de interconexión transformaron no solo la integración social de las computadoras sino también los desarrollos algorítmicos y las tecnologías ciber-físicas. Quizá esa fascinación con los trabajos fundacionales se sostiene en que los desarrollos posteriores estaban, en un cierto sentido, previstos como derivas teóricas esperables de la idea de máquina universal. Incluso ya entre 1947 y 1950 Turing propuso la idea de máquinas que aprendan e incluso que modifiquen su propia programación. Por supuesto que la realización concreta de estas ideas introdujo muchas novedades conceptuales que ameritan un estudio tecnológico y cultural propio que se encuentra en pleno desarrollo. Al mismo tiempo, sigue siendo fértil seguir explorando derivas teóricas que van de la mano con las trayectorias tecnológicas, en tanto condiciones de comprensión de su evolución y de formulación del abanico de posibilidades futuras. En cualquier caso, la compleja interacción entre ideas matemáticas profundas y heterodoxas -sobre todo las desarrolladas a partir de los trabajos de, entre otros, Alan Turing y Kurt Gödel- y desarrollos tecnológicos concretos y situados, parece ser una área inagotable de investigación y creación.

De computadoras y computaciones

Para poder caracterizar la ontología de lo computacional/digital partimos de una paradoja aparente: por un lado, parecería adecuado pensar en una ontología plana, dada la homogeneidad constitutiva de los objetos digitales todos finalmente reductibles a secuencias de ceros y unos; por otro, los diferentes niveles de abstracción que pueden conformarse en el universo digital no conocen cotas en su tamaño o complejidad. De hecho, como bien muestra Bratton (2016) en The Stack, el mundo presente alcanza una complejidad históricamente inédita a partir de las múltiples capas, mayoritariamente digitales, que lo constituyen. Más aún, no podría afirmarse que cada capa puede reducirse causalmente a capas inferiores, siendo necesario pensar espacios de causalidad intermedia, muchas veces definidos antes que sus implementaciones. Pueden pensarse también las implementaciones como interpretaciones semánticas, dando lugar a una ontología necesariamente multi-nivel o directamente jerárquica, con operaciones internas complejas. Esta dualidad aparentemente contradictoria es en realidad la base constitutiva y la potencia de las tecnologías computacionales. Disponer de un medio de desarrollo de estructuras de complejidad arbitraria a partir de unos pocos principios constitutivos habilita una nueva

forma evolutiva. En palabras de Dennett, el trabajo de Turing permitió la construcción de un enorme espacio de diseño de procesamiento de la información, en el cual pueden construirse recorridos que van desde la ignorancia absoluta hasta variadas formas de la inteligencia artificial (Dennett, 2017, p. 66).

Es importante destacar acá que el uso del sintagma "inteligencia artificial" no implica un compromiso, al menos en el caso de Dennett, de tomar a la inteligencia humana como paradigma a imitar o reproducir. De hecho Dennett opera una inversión "darwiniana" de ciertas concepciones del trabajo de Turing, ya que no lo considera como una invención top-down de sistemas inteligentes, sino por el contrario como un espacio donde pueden emerger diferentes formas de inteligencia donde solo hay en principio cálculo mecánico. La máquina universal de Turing es una de las nociones fundamentales para justificar esta posibilidad.

Hay una serie de discusiones y argumentos contra la posibilidad de reproducir la inteligencia humana, o al menos la conciencia humana. Una forma argumental esgrimida entre otros por Roger Penrose parece operar de la siguiente manera: Primero se muestra por algún argumento Gödeliano – erróneo – que no es posible que haya una mente implementada en una computadora. Asumiendo esto, se proponen buscar características no-digitales de los cerebros (supuestos fenómenos "continuos" o incluso "cuánticos"), y suponer (sin ninguna evidencia) que en esos fenómenos radicaría la posibilidad de inteligencia o de conciencia. No vamos a analizar aquí ninguna de esas propuestas, pero es importante dejar claro qué es lo que nos permite sostener la hipótesis mecanicista de Turing. Aunque asumamos que efectivamente haya fenómenos cerebrales irreductibles a lo digital (es difícil y sujeto a debate entender qué significaría esto), lo que sabemos hoy es que es suficiente con lo digital para producir comportamientos de una variedad infinita, establecer patrones de complejidad arbitraria, construir formas variadas y precisas de reflexividad y auto-conciencia, construir formas evolutivas (o co-evolutivas) para programas, en general en ambientes también digitales.

No abundaremos aquí en una caracterización de la noción de computación de Turing (1936) ya que hay suficiente literatura al respecto. Es necesario sí considerar que esta noción conceptual de computación también tiene su historia, antes y después de la confluencia de ideas que cobró forma en 1936 (GANDY, 1988). Una parte importante de la historia posterior incluye el desarrollo de una teoría de la programación, que puede pensarse como una novedosa aplicación a gran escala de la lógica matemática, lo que también dio nacimiento a una nueva perspectiva sobre los sistemas formales. Ya no serán éstos sólo formulaciones precisas de propiedades o estructuras matemáticas previamente existentes, sino un espacio de diseño, indagación y exploración de nuevas metodologías de trabajo dentro de los mismos sistemas. Esto contrasta con el uso que se daba a la lógica en matemática, donde importaban siempre más las propiedades meta-lógicas y nadie seriamente trabajaba haciendo desarrollos formales en el mismo sistema, excepto cuando era necesario para demostrar algún meta-teorema. Este pecado original de la lógica matemática es quizá una de las causas por las que muchos filósofos (Lucas, Searle, Penrose entre otros) expresan un desprecio por las propiedades sintácticas, asumiendo también una línea de defensa antropológica en la supuesta incapacidad semántica de los sistemas computacionales.

Ya Edsger Dijkstra (1982) indicaba que una de las novedades radicales que introducía la computación era la inédita complejidad combinatoria que debía ser resuelta con una misma tecnología: la programación. Desde los bits hasta las complejas aplicaciones computacionales del presente hay alrededor de diez ordenes de magnitud, y esa complejidad solo sigue creciendo. La multiplicidad de niveles semánticos involucrados en cualquier sistema computacional genera una complejidad inédita para la que no se dispone aún de herramientas conceptuales adecuadas. Además, tanto las abstracciones como las herramientas que permiten crearlas y usarlas, como las herramientas que se usan para construir esas herramientas, y así siguiendo, tiene el mismo status ontológico, y sus multifacéticos y polimórficos vínculos se constituyen con una misma forma tecnológica. En ese mismo artículo, Dijkstra advertía que las novedades radicales son poco bienvenidas en casi cualquier ámbito. El sentido común deja de funcionar

con ellas, las metáforas o alegorías usuales se vuelven inadecuadas y suelen ser tan perturbadoras que la mayoría prefiere ignorarlas, hacer de cuenta que no existen o que son irrelevantes. Vemos aún hoy con frecuencia este tipo de actitud de negar o de intentar minimizar la profundidad de las transformaciones en todos los ámbitos de la vida producidas por la computación ubicua.

Ya en ese momento, mucho antes de la aparición de Internet y la popularización de las computadoras personales, advertía Dijkstra que la transformación que la computación traía al mundo no podía ser comprendida con los conceptos, metáforas o analogías disponibles hasta entonces. Si bien han habido importantes en la recientemente desarrollada filosofía de la computación (Primiero, 2019), (Turner, 2018), la acelerada evolución de estas tecnologías presenta continuamente nuevos problemas a resolver y en algunos casos vuelve obsoletos o irrelevantes algunos de los análisis previos. Igualmente seguimos a Dijkstra en su idea de que el desafío intelectual que presentan las computadoras terminará siendo mucho más relevante que la miríada de aplicaciones que impactan diariamente en nuestras vidas. Esta idea, que incipientemente exploramos en este trabajo, pone en una perspectiva diferente el necesario análisis del enorme impacto de las tecnologías computacionales actuales, en particular de las que caen hoy bajo el confuso rótulo de "inteligencia artificial". De manera precautoria, será necesario distinguir las características propias de las tecnologías actualmente existentes de las posibilidades intrínsecas del mundo algorítmico.

Esta distinción va de la mano – aunque no se identifica – con dos perspectivas diferentes -ambas necesarias- acerca de qué es una computadora y de qué es el software. Por un lado, la noción conceptual de computación que ya describimos. Por otro, la rica historia de los desarrollos de software con sus múltiples invenciones desde mitad del siglo XX hasta el presente, que abordaremos en la siguiente sección.

Para concluir este recorrido por computadoras y programas, podemos considerar a la computadora misma como individuo técnico, constituido en un medio asociado propio e inscripto en un linaje técnico. Incluso desde esta perspectiva ingenieril, y sin poner en foco la enorme novedad del software, las computadoras ponen en tensión la noción de individuo técnico de Simondon. Una pregunta a resolver cuando hablamos de computadoras es si estamos pensando en un linaje técnico o en varios. Los principios de funcionamiento electrónico de las computadoras desde la década de 1940 hasta el presente han sido extremadamente variados y han evolucionado muy rápidamente. Una computadora actual no comparte ningún elemento técnico on una computadora de hace 50 años. Sin embargo, desde otra perspectiva, ambas computadoras son una encarnación de una máquina universal, y más precisamente son una realización de la arquitectura de von Neumann, que describe funcionalmente a las computadoras en términos de sus componentes y las interacciones entre estos. Ninguno de estos componentes, que podríamos considerar los elementos técnicos que constituyen el individuo técnico computadora, funciona de la misma forma en los diferentes momentos históricos, ni siquiera con los mismos principios. Pero en tanto realización del esquema de funcionamiento propuesto por von Neumann, las diferentes realizaciones pueden ser consideradas parte de una trayectoria tecnológica o incluso de un proceso de concretización.

Una computadora física es también una máquina abstracta, en un sentido a ser adecuadamente dilucidado (trabajándolo posiblemente desde una ontología de los artefactos). El funcionamiento de una computadora es descripto en términos de la circulación de bits entre sus diferentes componentes (memoria principal, procesador, memoria secundaria, etc.). Pero ¿qué es un bit? Claramente es también una entidad abstracta, un bit en memoria RAM difiere sustancialmente a nivel físico de un bit en el procesador central, en un disco o circulando por la red. La proliferación de diferentes tecnologías de almacenamiento muestra que ni siquiera están acotadas las diferentes realizaciones de esta idea. Si bien todo artefacto funciona en parte transmitiendo información entre los diversos componentes, este debe ser uno de los pocos casos donde todo su funcionamiento puede explicarse en esos términos. Desde esta perspectiva, los bits también son elementos constitutivos de las computadoras y evolucionan técnicamente. Concebir abstractamente a la computadora como formas de manipular bits independientemente de su realización física es una manera de poner en el mismo linaje a la EDVAC y la MacPro.

Cuando Simondon se refiere a las "máquinas de cálculo", refiere al margen de indeterminación que ofrecen y las posibilidades de adaptación y de constitución de conjuntos o redes. Esta propiedad que aumenta el grado de tecnicidad, es explorada por Simondon en una mirada más general de la técnica tomando como paradigma a las máquinas o, como las llama con un grado de generalidad mayor, los individuos técnicos.

Simondon sostiene que la actividad técnica puede ser analizada a partir de tres instancias: los elementos, individuos y conjuntos técnicos. Los individuos técnicos son aquellos que vinculan diferentes órdenes de magnitud para cumplir determinada actividad. Así, hasta el siglo XVIII, el rol de individuo técnico, sobre todo en la actividad artesanal y productiva, fue cumplido por el cuerpo humano: "debajo" de él se encontraban las herramientas y los instrumentos (herramientas de percepción), y "sobre" él estaban los talleres u otras sedes de la actividad técnica. Ya en el MEOT Simondon (2007) entendía que había un potencial diferencial en las incipientes computadoras:

> Las calculadoras modernas no son puros autómatas; son seres técnicos que, por sobre sus automatismos de adición (o de decisión por funcionamiento de basculadores elementales), poseen vastísimas posibilidades de conmutación de circuitos, que permiten codificar el funcionamiento de la máquina restringiendo su margen de indeterminación (Simondon 2007, p. 34).

Podemos encontrar, a grandes rasgos, dos maneras de relacionar lenguajes o proto-lenguajes formales y mecanismos. Por un lado, a medida que las máquinas fueron volviéndose más complejas, aparecieron notaciones cada vez más sofisticadas para caracterizar sus partes, formas, estructuras y funcionamientos. Charles Babbage desarrolló una notación mecánica para caracterizar de manera precisa la estructura y comportamiento de una máquina, fue la base de sus diseños de máquinas de cálculo, y también le permitió describir otras máquinas, como la máquina sueca de cálculo, de Scheutz. También Franz Reuleaux desarrolló un cálculo simbólico para expresar acoplamientos cinemáticos que permitían describir la topología de una gran variedad de mecanismos e incluso diseñar nuevos.

La irrupción histórica del software como constitutivo de casi toda forma tecnológica multiplicó las posibilidades de invención y achicó drásticamente los tiempos y costos de desarrollo tecnológico. Una de las principales razones de esta transformación, aún en curso, es el vínculo causal que se establece entre lenguajes y mecanismos, no ya en clave descriptiva sino como parte integrada del funcionamiento, donde la intervención humana en tanto intérpretes de las descripciones o prescripciones lingüísticas se vuelve innecesaria e ineficiente. Si bien en su artículo fundacional Alan Turing (1936) llama "computer" a un humano siguiendo reglas de cálculo, queda claro ya en ese trabajo que las máquinas aritméticas propuestas podían ser realizadas por mecanismos no-humanos, y que apelar a un computador humano tenía más que ver con la solución al Entscheidungsproblem que con el futuro desarrollo de la idea de computación.

La tarea desarrollada por el computer, humano o no, puede resumirse bajo la noción computacional de intérprete, esto es un programa o mecanismo particular que toma como datos de entrada otros programas en algún lenguaje de programación y sus datos propios y produce los resultados de ese programa. El computer hace eso con las reglas que definen una máquina de Turing, la máquina universal también puede pensarse como un intérprete de otro nivel, toma una codificación de cualquier maquina y le comporta como ella. El hardware de una computadora puede entenderse como un intérprete del lenguaje de máquina, cualquier lenguaje de programación tiene su propio intérprete (o un compilador que lo traduce a algún otro lenguaje que finalmente será interpretado, en última instancia por el hardware. Así los sistemas computacionales quedan caracterizados por dos propiedades fundamentales: la efectividad y la programabilidad. Ambas necesarias para caracterizar las computadoras y la noción misma de computación, aunque ninguna de las dos es extensional, es decir, que puede haber sistemas efectivos y/o programables que se comporten igual (para algunos casos) con otros que no lo sean (Blanco; García, 2022).

Esta idea de intérprete computacional como constitutivo de la idea de computación puede servir para un desarrollo más preciso y extenso del programa allagmático de Simondon, ya que justamente su razón de existencia es vincular causalmente a las codificaciones con los comportamientos, es decir, puede entenderse como una manera particularmente rica y precisa de establecer una ciencia de las operaciones (Priestley, 2011).

No hay computación sin software

El marco general propuesto por Turing que presentamos en la sección anterior sigue siendo fundamental para comprender, analizar e inventar nuevas formas computacionales y ofrece orientaciones indispensables en cualquier trabajo especulativo acerca del inagotable universo computacional. Sin embargo, como bien señala Bernhard Rieder, en muchos casos la referencia a los trabajos fundacionales de la ciencia de la computación omite analizar el rico desarrollo de las tecnologías computacionales sobre todo a partir de la década de 1970, cuando el software concreto cobra un interés en sí mismo, irreductible a los principios generales de la computación. No todo desarrollo de software puede ni necesita ser entendido como un capítulo de la concretización de las máquinas de Turing. En este sentido, el libro de Rieder (RIEDER, 2020) puede verse como el mejor intento hasta el presente de desarrollar una mecanología con el software en el centro en tanto objeto técnico.

Las nociones de elemento, individuo y conjunto técnico pueden ser recuperadas en el contexto de las tecnologías computacionales, pero requieren ser reelaboradas y contextualizadas. La multiplicidad y complejidad de niveles de abstracción que caracterizan a los sistemas computacionales hace más difícil identificar los diferentes componentes, y a veces un cambio de perspectiva (o de nivel de abstracción) cambia el rol que un objeto cumple. Los pocos trabajos que abordan esta cuestión buscan restablecer la tríada simondoniana en los contextos computacionales. En ocasiones, los elementos se identifican con funciones, los individuos con programas, y los conjuntos con plataformas o redes (Ferrarato, 2020). En otras, los elementos equivalen a las técnicas algorítmicas, los individuos a los programas, y los conjuntos a las plataformas (Rieder, 2020). No obstante, desde nuestra posición, todos esos intentos no dan cuenta de la serie de capas de abstracción cuyos niveles de funcionamiento se entrelazan. Como una propuesta heurística proponemos despegarnos de la noción de los objetos técnicos y pensar en términos de ambientes técnicos, en tanto un ecosistema de operaciones que transforman estructuras (Hörl, 2013)

El medio asociado a un objeto digital es en general también digital (o híbrido) y desde una perspectiva más abstracta éste existirá también en su propio medio asociado. Este tipo de secuencia de abstracciones no es una excepción sino que es intrínseco a la existencia del software en el mundo, conformando una "pila" (Bratton, 2016) de programas de complejidad creciente acoplados de maneras diversas. La ciencia de la computación, en particular la rama conocida como ingeniería del software, ha desarrollado múltiples herramientas para lidiar con este tipo de complejidad que pueden también ser útiles a la hora de comprender conceptualmente el software actualmente existente, sus interrelaciones, las plataformas como marcos a la vez técnicos y organizacionales. La manera simondoniana de presentar los linajes técnicos será también útil para analizar y dimensionar los acelerados procesos evolutivos del software.

Tampoco parece claro cómo extender la idea de concretización al software. Una de las ideas fuerza para programar sistemas medianos o grandes es la de separación de incumbencias (separation of concerns), que indica, en lo posible, abordar por separado cada cuestión a resolver y tratar de modularizar todo lo posible los programas, para mayor comprensión y facilidad con la localización de errores. Un software más concreto no sería entonces el que pueda incorporar más funciones en cada componente, sino por el contrario el que pueda aislar lo más posible cada función y resolverla por separado, aprovechando también las bibliotecas de software disponibles. En este sentido, creo que es completamente factible desarrollar una teoría de la concretización del software, aprovechando de hecho las múltiples ideas de la ingeniería del software sobre la calidad de los programas. Dicha teoría tendría que poder extender las ideas de Simondon a sistemas técnicos cuyos criterios de calidad no son necesariamente análogos a los planteados en MEOT.

Este desarrollo aportaría tanto a mejorar y refinar la noción de concretización como a poner a poner a prueba y extender las concepciones heredadas de décadas de estudio del desarrollo de programas. Pensar a los programas como objetos técnicos o como artefactos (Turner, 2018) conlleva incorporar nuevos criterios de validación de los programas, también en contextos sociales de uso o de desarrollo (Mills, 2016).

La ciencia de la computación suele poner como su principal objeto de estudio a la programación. Para Edsger Dijkstra puede considerarse una computadora como un procesador general de símbolos que se convierte en un procesador particular, específico, por la introducción de un programa (sin cambiar un solo cable). Esta idea se condice con la noción de Simondon mencionada antes, de entender a la programación como restringiendo el grado de indeterminación de una máquina de cálculo. Sin embargo, aclara Dijkstra, es más interesante enfocarse en los programas, y pensar al revés, es decir considerar a un programa como un manipulador abstracto de símbolos que se transforma en un procesador concreto de símbolos por la introducción de una computadora. Esta idea es coherente con pensar al software como objeto técnico. La idea de volverlo más concreto no parece directamente relacionada con la concretización en Simondon, pero será necesaria una reflexión más detallada sobre eso.

Agrega Dijkstra: entonces, la tarea de las programadoras se enmarca de manera precisa: tienen que construir dichos programas. ¿Y cómo se construyen los programas? Manipulando símbolos, disponiendo de principios científicos o matemáticos para hacerlo de manera correcta y estratégica. Para Dijkstra -y para nosotros- la ciencia de la computación es, y será siempre, un interjuego entre la manipulación mecánica de símbolos llamada computación y la manipulación humana de símbolos llamada programación.

Volviendo a Rieder, este da como primer ejemplo de técnica algorítmica, es decir de elemento técnico en su propuesta, a la recursión. No deja de ser revelador el ejemplo, también por los problemas que acarrea, sobre todo por el nivel de generalidad. Podríamos decir que hay muchas recursiones, en múltiples diferentes niveles. Aquí la idea de elemento técnico tiene que ser extendida, ya que la recursión puede verse también como una técnica estructurante de un programa. Si tomamos como un elemento, por ejemplo, un algoritmo de ordenación, en ese nivel de abstracción es irrelevante si dicho algoritmo es recursivo o iterativo, si usa listas enlazadas (¿otro elemento técnico?) o arboles binarios, etc. Es decir, en ciertos contextos, la recursión funcionaría como elemento técnico, es decir como técnica algorítmica, pero en otros podría tener otros roles.

Quizá una de las características distintivas de la programación en tanto forma técnica, es la diversidad de roles de sus distintos componentes. La recursión puede funcionar como un elemento técnico, como una forma de estructurar un programa en tanto individuo técnico, como herramienta conceptual para dar el marco a un programa (por ejemplo para la descripción de una estructura sintáctica que será implementada por un parser generador), etc. La noción teórica de que todo programa también puede ser un dato de otro (meta)programa obliga a sostener cierta plasticidad conceptual.

Es por esto que sin abandonar la idea de programación como vínculo privilegiado entre humanos y sistemas computacionales, es necesario elaborar cómo funciona ese interjuego entre manipulación humana y mecánica de símbolos. Dijkstra pone el énfasis en el trabajo sobre sistemas formales, los que funcionan como andamiajes tecno-conceptuales para la construcción de los programas. Su principal herramienta técnica para construir programas era su lapicera. Pero también esos vínculos están mediados por otros (meta)programas, indispensables para el trabajo con las computadoras, los cuales Dijkstra da por supuestos y no tematiza. Esta otra dimensión específica de las tecnologías computacionales es que los métodos y estrategias de concretización se distribuyen de diversas maneras entre programadoras y software. Desde que, entre otros, Grace Hopper comenzó a concebir herramientas intermedias entre programadoras y computadoras, una variedad de herramientas permite mejorar esa distribución: ensambladores, macro-expansión, intérpretes, compiladores, sistemas de tipos, super-compiladores, optimizadores. En esta genealogía de herramientas para programar pueden incluso incluirse hoy los

sistemas de construcción de clasificadores estadísticos conocidos como machine learning, que abordaremos en la próxima sección.

Datos, algoritmos, información

Cuenta Luciano Floridi (2014) que se estimaba que la humanidad había producido un total de 12 exabytes (un exabyte es 10^{18} bytes) de información en toda su historia hasta la aparición de las computadoras personales, pero que para 2006 habría llegado a los 180 exabytes, o sea que en unas pocas décadas se había producido 15 veces más información que en todo el resto de la historia. Las estimaciones actuales hablan de una producción de 328 exabytes ¡por día!, es decir que en un día se produce casi 30 veces la cantidad de información que la humanidad produjo hasta avanzado el siglo XX. ¿Cómo es posible esta gran aceleración, y qué consecuencias sociales, epistemológicas y políticas tiene?

La aparición de Internet a inicios de la década de 1990 aceleró el proceso de datificación. También puede situarse ahí el cambio de siglo, el comienzo de una nueva era donde las mediaciones digitales pasarían a establecer las condiciones para la cognición y la percepción, distribuidas técnicamente y en una evolución acelerada.

Otro gran salto en la cantidad de datos disponible lo produce el desarrollo masivo de los llamados algoritmos de aprendizaje maquínico o machine learning. Éstos constituyen una realización técnica específica que es posible gracias a que todo programa es también un dato, es decir es procesable por otros programas. La construcción de programas clasificadores usando técnicas de machine learning es una aplicación no demasiado sofisticada de meta-programación, es decir de construir un programa que produzca otros programas. En todas sus variantes, se dispone de un programa de entrenamiento que toma un conjunto grande de datos y a partir de ellos construye otro programa, un clasificador, capaz de reconocer en datos nuevos cuáles son suficientemente semejantes a los datos de entrenamiento. De alguna manera infiere las propiedades implícitas que satisfacen los datos originales, propiedades que en general no responden a formas humanas de categorizar (es por esto que Stiegler (2020) habla de máquinas categoriales, creadoras de nuevas formas de categorizar). Es importante remarcar acá que las categorías creadas usualmente no admiten una expresión comprensible y comunicable. Podemos pensar que estos programas de entrenamiento comprimen la información contenida en los datos de entrenamiento y la operativizan.

En diversos trabajos y notas periodísticas se alude correctamente a la falta de capacidad de inferencia de los sistemas de machine learning (referidos como "inteligencia artificial"), indicando que su principal virtud está en encontrar correlaciones estadísticas, incluso entre variables a priori desconocidas. La posibilidad de reconocer patrones en base a una gran potencia de cálculo y la abundancia de datos permite pensar a esta tecnología como una nueva forma de percepción, pudiendo comprenderse como un instrumento de magnificación del conocimiento, un nooscopio (Pasquinelli; Joler, 2020). La metáfora de entender a estos clasificadores estadísticos como lentes sugiere que además de permitir ver lo que sería invisible de otros modos, también introducen sus propias distorsiones, a veces llamadas "sesgos" y en este caso, sin ningún modelo óptico-geométrico que las acote.

Hay buenos trabajos, además del ya mencionado nooscopio, que dan cuenta de esta manera de comprender a los programas de machine learning (O'neil, 2016), (Crawford, 2021). Solo me interesa referir aquí a una falacia de composición que es necesario analizar para evitar algunos sesgos teóricos y las posiciones fatalistas que se siguen de ellos. En realidad se trata de dos falacias análogas: por un lado, considerar que las propiedades de los algoritmos de machine learning que se satisfacen hoy son válidas para cualquier desarrollo de esta tecnología. El contexto de producción, los criterios a maximizar cuando se construyen los modelos y la propia historia técnica imprimen a los programas ciertas características que son bien descriptas en los libros mencionados. Ahora bien, el machine learning es también una meta-tecnología, es decir que es un marco en el que pueden desarrollarse tecnologías específicas, por ejemplo reconocimiento de imágenes médicas, jugadores de go o de ajedrez, procesamiento de lenguaje natural,

plegado de proteínas... la lista puede extenderse casi indefinidamente. Algunas de las características de cada una de esas formas técnicas suelen ser tomadas como válidas para cualesquiera otra, incluso para las que aún no se han desarrollado. Eso conlleva muchos prejuicios y suele restringir la imaginación para constituir mejores marcos tecnológicos, desde perspectivas políticas, culturales o cognitivas. El estado actual de concentración monopólica de recursos computacionales y datos, que imprime su propio sesgo a los desarrollos más grandes del momento, es contingente y pueden pensarse alternativas más comprometidas con crear condiciones para un desarrollo y uso más justo e igualitario.

La otra falacia que suele ocurrir en los análisis sociales de las tecnologías computacionales, es tomar las características propias de los sistemas de machine learning como extensibles a cualquier sistema algorítmico. Incluso el uso común de esta palabra – hasta hace poco desconocida en el habla general – hoy queda asociada a los clasificadores estadísticos, que si bien en tanto programas en funcionamiento puede decirse que implementan un algoritmo, lejos están de parecerse a los algoritmos paradigmáticos que se han desarrollado históricamente. Por cierto, y en tensión con lo dicho anteriormente, los sistemas de machine learning sí tienen algunas características específicas que no se extienden a cualquier sistema algorítmico. Entre las más destacadas se encuentra la opacidad de las representaciones que usan los modelos, la incapacidad de manipular su propio código de manera explícita, la poca composicionalidad. Todas estas propiedades dan cuenta de una baja tecnicidad y de una necesidad de buscar nuevas ideas (Marcus; Davis, 2019). Respecto de la idea de sesgo, es indudable que es un tema interesante y necesario de analizar, pero la formulación misma del problema parece sugerir que podría haber programas de machine learning sin sesgos, lo cual es una tontería. La limitada capacidad humana de mitigar los propios sesgos tiene un origen cultural y se logra por feedback social, sobre todo negativo. Los programas de machine learning operativizan sesgos en su propia definición, que por supuesto pueden ser debatidos y cambiados, pero no podría existir un clasificador estadístico sin sesgos, la idea misma es auto-contradictoria.

Estas falacias se reflejan en muchos de los conceptos con los que se busca dar cuenta de las consecuencias sociales de la expansión de las tecnologías computacionales. Ejemplo de esto, es el concepto, útil e interesante, de "gubernamentalidad algorítmica" (Rouvroy; Berns, 2013). Los análisis de esta noción foucaultiana, muy pertinente para dar cuenta del presente político, conllevan también, al menos en una lectura rápida y dada la fuerza y novedad del adjetivo usado, a concluir que toda mediación algorítmica en política adolecería de los problemas descriptos. Por el contrario, creemos que hay buenos caminos para la construcción de mediaciones computacionales que favorezcan valores colectivos, reflexivos y creativos.

Para concluir esta sección remito a una discusión que venimos teniendo con Manolo Rodríguez y Darío Sandrone: ¿es posible caracterizar a las tecnologías de machine learning usando una nueva tríada simondoniana: datos-algoritmos-plataformas?

Así como la gran novedad técnica del siglo XIX fue que las máquinas comenzaron a ser los nuevos individuos técnicos, esto es, sistemas de portación de herramientas, desplazando a los cuerpos humanos y, en todo caso, integrándolos como elementos técnicos dentro de su propia actividad, encontramos en el siglo XXI un esquema novedoso. La analogía puede servir para pensar la nueva infraestructura digital a partir de la serie datos-algoritmos-plataformas, que estarían en la base de un nuevo tipo de maquinaria correspondiente a una nueva etapa del capitalismo, que algunos llaman "capitalismo de plataformas" (Srnicek, 2016). Esta manera de concebir el modo de existencia de los programas de machine learning actuales, no se adecua de manera completa al esquema simondoniano, ya que no podría decirse que los datos son elementos constitutivos de los algoritmos. Hay un proceso complejo de mediación entre ambos en el cual solo quedarían los patrones implícitos inscriptos en los procesos de clasificación. Las plataformas tampoco son meramente un conjunto de algoritmos sino que tienen una estructura compleja tanto organizativa como tecnológica (Bratton, 2016) y están sometidas a un proceso constante de (re)estructuración. Pese a todas estas diferencias y necesidades de adecuación conceptual, la elaboración

de esta tríada puede ser una buena apuesta para la construcción de una cultura técnica para los inicios del siglo XXI.

Aquí encontramos otro ámbito de trabajo donde una relectura de Simondon podría abrir nuevas vías interesantes de investigación. Una de las razones fundamentales de la hiperabundancia de datos es que se dispone ahora de una tecnología capaz de operativizarlos eficazmente. Es decir, podemos preguntarnos qué hace que algo sea un dato. En principio eso es simple en programación, un dato siempre es tal relativamente a un programa dado. En el caso del machine learning, se llama datos a los que sirven como entrada del programa de entrenamiento, y luego a los que serán clasificados por el programa obtenido.

Curiosamente, entrada ya la tercera década del siglo XXI, las principales teorías de la información en uso se desarrollaron antes de la década de 1970, es decir cuando aún no había computadoras personales y, como vimos, la cantidad de información producida en toda la historia humana hasta entonces era menos que la que se produce hoy en una hora. Dos de los principales exponentes son la hoy llamada teoría estadística de la información de Shannon y la teoría algorítmica de la información de Kologorov, Solomonoff y Chaitin. Ambas intentan medir la cantidad de información y establecer cotas para la posibilidad de procesarla y almacenarla. Más allá de la indudable utilidad de ambas en sus respectivos ámbitos de aplicación, no es claro que den cuenta de los principales fenómenos informacionales actuales.

La noción de información de Simondon también apareció antes de la computación ubicua e Internet, y como tal estudia fenómenos informacionales diversos, sin ningún énfasis especial en las tecnologías computacionales. Sin embargo, dado el nivel de generalidad conceptual de su teoría, es posible adecuarla a la nueva escena de producción acelerada de datos. La teoría de Shannon calcula (im)probabilidades a partir de un código ya constituido mientras que la teoría algorítmica asocia la información a la longitud del programa más corto que la produce. Pero ninguna de ellas parece adecuada para dar cuenta de la información contenida en los datos de entrenamiento de un sistema de *machine learning*. La idea de Simondon, en cambio, de que la información es relativa a un receptor dado, y que orienta la operación de individuación a partir del estado metaestable de dicho receptor, parece directamente aplicable. Esto llama a un desarrollo teórico y matemático de esta idea para volverla operativa.

Conclusión: condiciones para una co-evolución

Entender a los programas, del tipo que sean, como objetos técnicos permite pensar desde la filosofía de Simondon el tipo de relaciones que se establecen con ellos. Si la relación se da desde el desconocimiento, se producen formas insidiosas de alienación. En general para los sistemas computacionales el conocimiento será siempre parcial, en parte por la enorme complejidad combinatoria que ostentan. Para el caso particular de la actual tecnología de *machine learning*, la opacidad de sus representaciones y la forma estandarizada de programación establecen por defecto relaciones con un alto grado intrínseco de alienación, incluso para quienes programan esos sistemas. Para quienes los usan, la alienación aumenta, volviéndose incluso en algunos casos, de condicionamiento comportamental.

Sin embargo, la posibilidad de establecer mejores relaciones con los programas está siempre presente, siendo la programación uno de los vínculos más creativos y dinámicos. Por supuesto que ciertas formas tecnológicas específicas dentro de la meta-tecnología computacional favorecen mejores vínculos. Esto puede ser indicativo de qué formas de concretización permiten mejores individuaciones futuras, constituyendo criterios tecno-políticos para eso.

En este sentido, la programación puede ser vista como una dimensión particularmente promisoria para los procesos de subjetivación, pero para que funcione de manera eficaz e integrada al desarrollo colectivo serán necesarios ambientes técnicos reflexivos y versátiles. Pueden desarrollarse formas de acoplamiento que maximicen el grado de indeterminación, o que al menos ofrezcan un abanico de

posibilidades extenso y variado. En contra de ciertos prejuicios, la computación no es una forma de automatización, por el contrario, es condición para orientar procesos de desautomatización.

Creemos que una manera de conformar mejores criterios evolutivos, puede basarse en la propuesta de Simondon de la constitución de una allagmática, una ciencia de las operaciones. Para ello, es necesario plantear con claridad definiciones operacionales, en las que las entidades no permanecen ontológicamente fijas como elementos, individuos o conjuntos, como si se trataran de estructuras, sino que puedan dinámicamente asumir diferentes posiciones operatorias los diferantes niveles de análisis. En esta línea, una aproximación allagmática al esquema datos-algoritmos-plataformas debe poder dar cuenta del modo en que son formalizadas, patronizadas y predichas las actividades sociales que se desarrollan, cada vez más, a través de las mediaciones algorítmicas.

Referencias

BLANCO, J.; GARCÍA, P. Efffectiveness and Programmability. In defense of a relational notion of computation. In Dov Gabbay (editor): Mathematical foundations of software engineering—essays in honour of Tom Maibaum on the occasion of his 70th birthday and retirement, 179–191, Coll. Publ., [London], 2022.

BRATTON B. *The Stack*: On software and sovereignty. Cambridge: MIT Press, 2016

CRAWFORD, K. *The Atlas of AI*: Power, Politics, and the Planetary Costs of Artificial Intelligence. Connecticut: Yale University Press, 2021.

DENNETT, D. From Bacteria to Bach and Back: The Evolution of Minds. England: Penguin Books, 2017.

DIJKSTRA, E. On the cruelty of really teaching computing science. EWD, 1036, 1988.

FERRARATO, C. *Prospective Philosophy of Software*: A Simondonian Study. New Jersey: John Wiley & Sons, 2020.

FLORIDI, L. *The Fourth Revolution*: How the Infosphere is Reshaping Human Reality. Oxford: OUP, 2014.

GANDY, R. The Confluence of Ideas in 1936. In: R. Herken (ed.). *The Universal Turing Machine*: A Half-Century Survey, Oxford: Oxford University Press, pp. 55-111, 1988.

HÖRL, E. A Thousand Ecologies: The Process of Cyberneticization and General Ecology. translated by Jeffrey Kirkwood, James Burton, and Maria Vlotides. In: DIEDERICHSEN, Diedrich; FRANKE, Anselm. *The Whole Earth*: California and the Disappearance of the Outside. edited by. Diedrich Diederichsen and Anselm Franke, 121–130. Berlin: Sternberg Press, 2013.

MARCUS, G.; DAVIS, E. *Rebooting AI*: Building Artificial Intelligence We Can Trust. US: Knopf Doubleday Publishing Group, 2019.

MILLS, S. *Gilbert Simondon*: Information, Technology, and Media: Rowman & Littlefield International, 2016.

O'NEAL, C. *Weapons of Math Destruction*. New York: Crown Books, 2016.

PASQUINELLI, M.; JOLER, V. El Nooscopio de manifiesto. *laFuga*, v. 25, 2021. Disponible en: http://2016.lafuga.cl/el-nooscopio-de-manifiesto/1053.

PRIMIERO, G. *On the Foundations of Computing*. Oxford: Oxford University Press, 2019

RIEDER, B. Engines of Order: A Mechanology of Algorithmic Techniques. Netherlands: Amsterdam University Press, 2020.

ROUVROY, A. ; BERNS, T. "Gouvernementalité algorithmique et perspectives d'émancipation : Le disparate comme condition d'individuation par la relation?" *Réseaux*, v. 40, n. 177, 163–196, 2013.

SIMONDON, G. *El modo de existencia de los objetos técnicos*. Buenos Aires: Prometeo, 2007.

SIMONDON, G. *La individuación a la luz de las nociones de forma e información*. Buenos Aires: Cactus, 2009.

SIMONDON, G. *Communication et information*: Cours et conferences. Chatou: Les Éditions de La Transparence, 2010.

SRNICEK, N. *Platform Capitalism*. New Jersey: John Wiley & Sons, 2016.

STIEGLER, B, et le Collectif Internation (Eds.). Bifurquer, il n'y a pas d'alternative: Paris: Les Liens qui Libèrent, 2020

TURING, A. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. London Math*, s2-42(1), pp. 230-265, 1936.

TURNER R. *Computational Artifacts*: Towards a Philosophy of Computer Science. Springer, 2018.

RECEBIDO: 24/10/2023
APROVADO: 19/06/2024

RECEIVED: 10/24/2023
APPROVED: 06/19/2024