LAS LECTURAS SIMPLISTAS EN LAS QUE EL CAMBIO TECNOLÓGICO RESULTA EN UNA REDUCCIÓN DEL EMPLEO OCULTAN LOS PROCESOS COMPLEJOS DE DIVERSIFICACIÓN Y CAMBIO ESTRUCTURAL QUE SE DERIVAN DE ESTE. EN EL PRESENTE ARTÍCULO ANALIZAMOS CUATRO TENDENCIAS GENERALES EN UNA (MUY BREVE) HISTORIA DE LA INFORMÁTICA PARA DESARTICULAR DISCURSOS SOBRE EL FUTURO DEL TRABAJO.

LA PROGRAMACIÓN, ¿UNA CLAVE PARA DESARTICULAR LOS DISCURSOS APOCALÍPTICOS SOBRE EL FUTURO DEL TRABAJO?

por VERÓNICA ROBERT. *Centro de Estudios Económicos del Desarrollo. Conicet-IDAES/UNSAM.*

por NICOLÁS MONCAUT. *Centro de Estudios Económicos del Desarrollo. Conicet-IDAES/UNSAM*

"La mejor forma de predecir el futuro es inventarlo".
Alan Kay

Introducción

Desde la revolución industrial en adelante, la relación entre trabajo y progreso tecnológico ha sido compleja. El fantasma de la desocupación en masa provocada por la emergencia de nuevas tecnologías ha sido recurrente. Y no sin razón. De hecho, cientos, miles de tareas y puestos han desaparecido por el cambio tecnológico. Sin embargo, el trabajo sigue siendo un recurso valioso para la producción capitalista.

Frente al avance de las tecnologías de información y comunicación (TICs) con adelantos que reorganizan la producción, tales como la robotización, la inteligencia artificial, el aprendizaje automatizado (*machine learning*), y a transformaciones organizacionales tales como la *Gig Economy* (una nueva modalidad de trabajo basada en plataformas, conocida también economía de los pequeños encargos), han reaparecido las predicciones apocalípticas sobre el futuro del trabajo. Lanzan pronósticos desoladores sobre ocupaciones que desaparecerán ("el 47% de los puestos trabajos serán reemplazados por máquinas") y plantean una gran incertidumbre en torno al futuro ("que aún no existen ni sabemos cómo serán los empleos de las futuras generaciones").

Más allá de los pronósticos, en este artículo nos interesa destacar algunas reacciones a tales expectativas como: (i) las que utilizan el eventual desempleo tecnológico como velo de las crisis económicas derivadas de problemas de insolvencia de la demanda o de la deslocalización de la producción; o (ii) las que aprovechan la ocasión para impulsar transformaciones en las instituciones del mercado de trabajo, que significan un cercenamiento de los derechos laborales.

En este artículo se pretende desarmar estos discursos demostrando que la relación entre trabajo y tecnología es compleja y contradictoria. Para ello utilizaremos la historia de la informática, porque es uno de los sectores señalados como el principal responsable de la crisis del empleo actual y el resurgimiento de discursos sobre el futuro del trabajo. Pero también porque nos permite ver cómo a lo largo de la historia en el sector han convivido tendencias hacia el ahorro de costo salarial y hacia la ampliación y complejización de la demanda de trabajo, que resultan en el crecimiento de la masa salarial. La limitación de este ejercicio está en que no podremos ver mecanismos de compensación intersectoriales, tan característicos del proceso de cambio tecnológico. Pero demostramos que si estas tensiones están incluso presentes dentro de un mismo sector, será más fácil prever que ocurran en la economía como un todo.

Software como problema, software como solución

Hoy pocas personas quedan al margen del nuevo mundo digital, cada vez más ubicuo, ya sea porque utilizan redes sociales, plataformas de compra, sistemas digitales de pago, comunicaciones en un sentido amplio o ¡incluso electrodomésticos! Sin embargo, tan sólo el 0,33% de la población mundial sabe programar, lo que implica un crecimiento contante de los costos laborales.

La historia de la programación es una historia de expansión incesante en la que se expresan de forma contradictoria la búsqueda a la baja de los costos laborales, a través del uso de la tecnología para la gestión y el control del proceso de trabajo, y el aumento exponencial de la demanda de trabajo, que además de expandirse se diversifica y complejiza en una red creciente de competencias y habilidades específicas.

Estas tendencias aparentemente contradictorias cobran sentido a la luz de la competencia capitalista, que opera vía el conflicto distributivo entre capital y trabajo, pero también vía el conflicto entre capitales que conduce a la exploración de nuevos conceptos tecnológicos y a una constante diferenciación. Es decir, un conflicto entre lo viejo y lo nuevo, o, como lo describe Schumpeter, el proceso de destrucción creativa:

I- Puja distributiva, como reflejo de la competencia entre capitales homogéneos, orienta el cambio tecnológico a reducir los costos laborales.

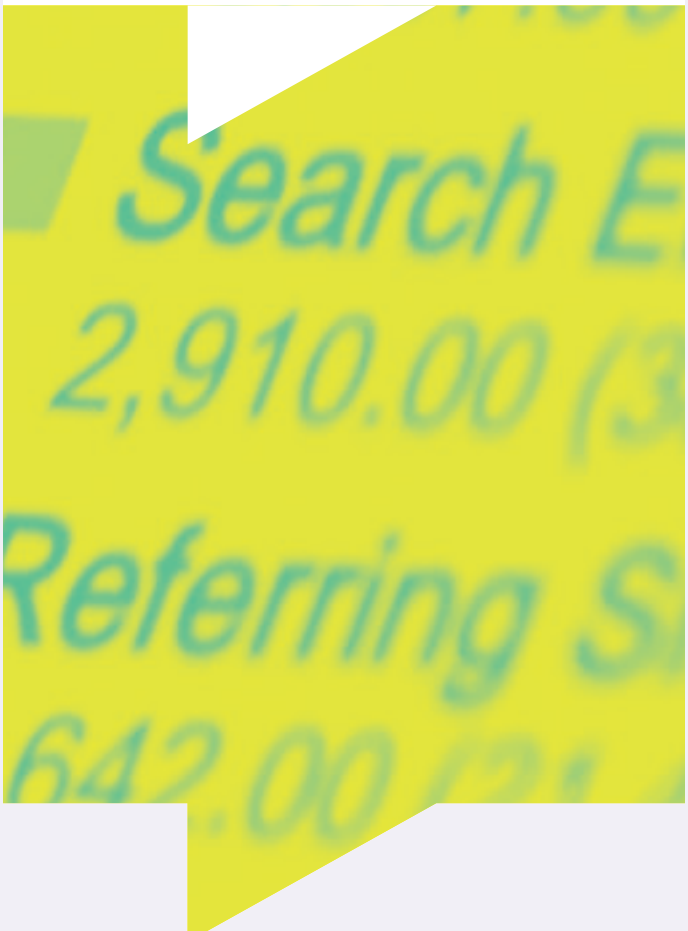
II- Esto puede lograrse a través dos estrategias diferentes, vía la incorporación de más personas a la fuerza de trabajo, al reducir los costos de reproducción, o vía la automatización.

III- Con la difusión del cambio tecnológico se exacerba la competencia capitalista que –en búsqueda de nuevas fuentes de cuasi-rentas– explora nuevos conceptos tecnológicos y alienta la diferenciación.

IV- Este proceso alimenta la demanda de trabajo y la diversifica en diferentes perfiles laborales asociados a los nuevos conceptos tecnológicos que suponen diferentes costos de reproducción. De este modo la búsqueda por reducción de costos laborales coincide con expansión de la masa salarial.

En las siguientes secciones analizaremos estas tendencias en una breve historia de la informática.

Los lenguajes de alto nivel permitieron reducir los tiempos de formación requeridos y expandir la oferta de programadores, lo cual contrarrestaba parcialmente al crecimiento de los costos laborales.



La historia

Los orígenes

Los orígenes de la informática se remontan muy atrás en el tiempo, con la máquina de Babbage inspirada en los telares Jacquard en la que la programación de la máquina era independiente de los procesos mecánicos que la ejecutaban. Sin embargo, la primera máquina programable utilizada en la automatización de tareas de cálculo de forma masiva fue la ENIAC (Electronic Numerical Integrator And Computer) desarrollada por la Universidad de Pensilvania, EE.UU. La ENIAC, de acuerdo con muchos historiadores, fue de hecho la primera computadora de propósito general. Construida con el objetivo de estimar trayectorias de misiles durante la Segunda Guerra Mundial, permitió ahorrar miles de horas de cálculos manuales. Cientos de matemáticas mujeres estaban abocadas a esta tarea en el Departamento de Defensa de EE.UU., entre ellas, solo seis fueron seleccionadas para programar la ENIAC. La **automatización** del cálculo iba a desplazar a miles de trabajadores de cuello blanco y, aun así, una nueva profesión había nacido.
Aún sin lenguajes, las primeras programadoras de la historia escribían sus códigos abriendo y cerrando puertas lógicas de forma manual (es decir, "escribían" las instrucciones en una cadena de ceros y unos) para que la ENIAC ejecutara luego una rutina.
Si bien la ENIAC fue un desarrollo tecnológico impulsado por el sector público con motivos de defensa, pronto se expandió su uso a fines civiles con la **difusión** de la informática orientada a la investigación científica y a los negocios, abriendo **nuevos negocios y diversificando mercados** en un proceso de competencia schumpeteriana por capturar cuasi-rentas tecnológicas.

Los primeros lenguajes

En las décadas de los '50 y '60 prosperaron prototipos y modelos variados de computadoras como la ENIAC, así como calculadoras mecánicas y electrónicas y otros dispositivos de uso específico. Para la programación de algunas de estas máquinas ya se disponía del Lenguaje Assembler, que ofrecía un conjunto de instrucciones básicas listas para utilizar sin necesidad de su traducción al código máquina (de cero y unos). Esto implicó una primera separación entre la tarea de programar y el diseño de la arquitectura del computador, que facilitó el desarrollo de la actividad, aunque todavía estaba bajo el control de los productores de hardware. Los programas de base, como sistemas operativos o controladores, eran desarrollados por empresas de hardware en asociación con laboratorios privados (Bell Labs, de

AT&T, o Palo Alto, de Xerox) y universidades (como el MIT). Las personas orientadas al diseño de las computadoras, la arquitectura del software, así como a la programación de las máquinas propiamente dichas, eran científicos altamente calificados y las tareas que desarrollaban requerían de una elevada creatividad e inventiva. Las nuevas actividades y los nuevos puestos de trabajo crecían a un ritmo lento, pero con altos salarios.

La ingeniería de software

En la década de los '70, las mejoras tecnológicas sustantivas en materia de microprocesadores y la fuerte expansión de las computadoras ampliaron significativamente la demanda de soluciones informáticas y por lo tanto de los costos laborales. La respuesta fue ampliar la oferta de trabajo de programadores (o **reducir sus costos de reproducción**) a través del desarrollo de lenguajes de alto nivel como el C, creado por el Bell Labs, que permitieron la **estandarización de las competencias laborales** para la programación y de su difusión con la creación de carreras universitarias como la computación científica y la investigación operativa y más tarde con la ingeniería de software. Por otra parte, estos lenguajes fueron el primer paso a una mayor independencia del desarrollo de software respecto del hardware en el que este sería ejecutado.

La consolidación de ingeniería del software como disciplina tecnológica permitió estandarizar las metodologías de desarrollo de software y planificar los procesos asignando tareas y plazos, lo que, entre otras cuestiones, volvía más predecible el desarrollo de software y menos sujeto a competencias y saberes específicos. Así, se introduce el control sobre la programación orientado a **incrementar la productividad vía la intensificación del trabajo**. Por otra parte, los lenguajes de alto nivel permitieron reducir los tiempos de formación requeridos y **expandir la oferta de programadores**, lo cual contrarrestaba parcialmente al crecimiento de los costos laborales.

Este proceso consolidó aquello que se había iniciado con los Lenguajes Assembler: la separación entre el software y el hardware. La emergencia de nuevos modelos empresariales como el de Microsoft, basado en el desarrollo de software a medida (OS/DOS) para empresas de hardware (IBM), monetizable a través de la venta de licencias junto al hardware, es el mejor exponente de esta tendencia. La estrategia de separar el desarrollo de software de la producción de hardware se orientaba a **explotar ventajas de producción en escala y tercerizar la costosa actividad de desarrollo de software**, en el contexto de competencia entre empresas de hardware.

El software libre

Previo a la consolidación de los modelos de negocios de las empresas de software, muchos programas y rutinas circulaban entre los usuarios (especialmente académicos). Estas prácticas de compartir código entraron en conflicto con la comercialización del software y dieron lugar a la emergencia de modelos alternativos de desarrollo basados en la libertad de circulación y apertura del código. Las asociaciones que propiciaban estos movimientos, como la Free Software Foundation, sostenían que las modalidades descentralizadas de desarrollo lograban programas de mejor calidad técnica a la vez que alentaban el aprendizaje autónomo y la innovación. Algunas empresas se opusieron férreamente al modelo abierto (como Microsoft), pero otras lo alentaron ya que entendieron que estas prácticas, aunque contrarias a la comercialización del software y hasta cierto punto idealistas en materia de propiedad intelectual, favorecían la **reducción de los costos de desarrollo del software y de construcción de competencias** al desplazar de la esfera productiva a la esfera privada la actividad de la programación (en muchos casos ejercida como *hobby*).

Hoy en día, con la difusión de las capacidades de programación, los foros y la comunidad de software libre tienen un impacto significativo en la producción de software a nivel global.

El proceso de competencia en clave schumpeteriana describe una carrera loca por reducir costos laborales al tiempo que complejiza la demanda de trabajo.

GUI

Por mucho tiempo la informática estaba restringida a especialistas: académicos, oficinas públicas o grandes empresas, y a un creciente grupo de entusiastas hasta cierto punto atraídos por la cultura *geek* y los videojuegos. Fuera de este círculo, la informática era un terreno ajeno a la mayoría de la población porque, a pesar de que los lenguajes ya estaban considerablemente lejos del código máquina, aún era una disciplina considerada difícil y árida. La emergencia de las interfaces gráficas de usuario (GUI) permitió que el uso de computadoras personales se popularizara, en esta etapa es cuando llega una computadora personal a cada hogar, lo que provocó que se extendieran y multiplicaran las demandas de aplicaciones.

La interfaz gráfica que expandió la demanda de programadores también ayudo a expandir la oferta, ya que la aparición de lenguajes gráficos (como Visual Basic) **amplió el conjunto de personas con habilidades para programar**.

La Internet y el outsourcing global de software

En los '80 y los '90, las redes de computadoras, primero privadas y luego públicas, bajo los protocolos de comunicaciones de Internet ampliaron todavía más los usos que ahora se extendían al terreno de la comunicación ya sea a través del correo electrónico, como a través de páginas estáticas y dinámicas pronto llegaría a lugares hasta entonces insospechados como el de las redes sociales. **Internet dio un nuevo impulso a las necesidades de programación**, mientras que su **difusión facilitó el desarrollo de software a la distancia**, a través de la implementación de diferentes modelos de *outsourcing* y *offshoring* global de software. El mercado de trabajo se extendió y entraron nuevos jugadores globales como India, con la consiguiente reducción de los costos de programación.

La ley de Moore, del otro lado del tablero

A lo largo de todo este proceso, la expansión en la capacidad de procesamiento de los microchips, almacenamiento en diferentes soportes (discos rígidos, memorias, etc.) y transmisión a través de fibra óptica y redes inalámbricas como 3G, 4G y la inminente 5G siguió creciendo a tasa exponencial. Claro que duplicar la capacidad de procesamiento del microprocesador de INTEL 4004 de 1971 (2.300 transistores) no tendría el mismo impacto que duplicar la de los micros más recientes que cuentan con más de 700 millones de transistores. La miniaturización de componentes permitió su expansión. Hoy la barrera más grande para poder continuar con esta expansión no es la imposibilidad

física de colocar más transistores en un espacio finito, sino la de reducirlos aún más de tamaño. Aun así, los nuevos desarrollos en materia de computación cuántica permiten prever un futuro con mayor capacidad de procesamiento todavía. Esta dinámica tecnológica, junto con la extensión global de la Internet, masificaron el acceso a nuevos dispositivos interconectados de propósito general (como teléfonos inteligentes) que permitieron la aparición de nuevas aplicaciones y tecnologías como la internet de las cosas y del *Big Data*.

Todos estos factores fortalecen minuto a minuto a nuevas demandas de programadores. Muy distintas, claro, a las de las 6 mujeres pioneras de la programación, o a la de los investigadores de Bell Labs y Palo Alto como Alan Kay, o incluso a la de los programadores de Visual Basic con el *boom* de la programación gráfica, pero de una magnitud muy superior.

Capacidades y plataformas

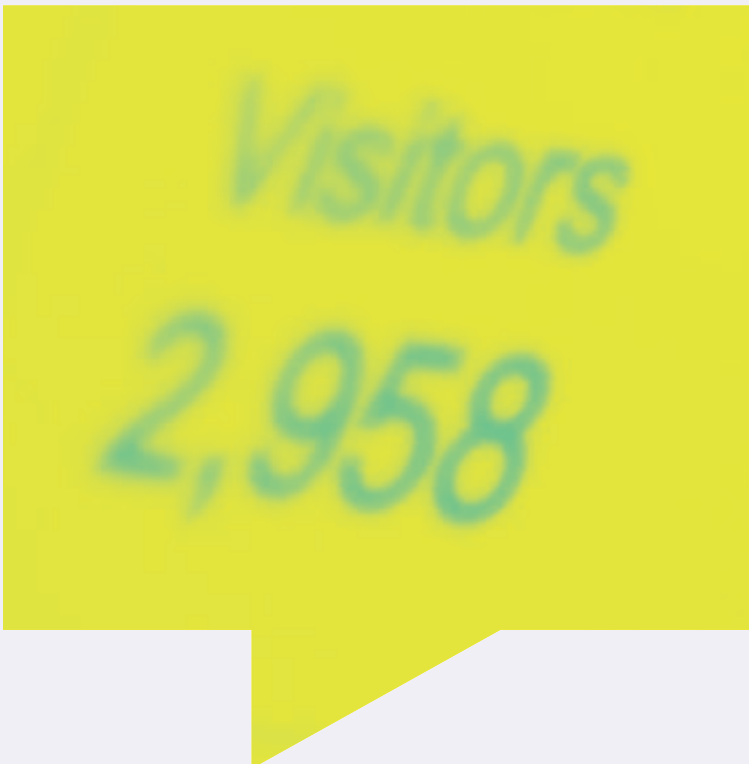
Los nuevos programadores disponen de un conocimiento que es cada vez más genérico y difundido. Adquieren sus habilidades de programación en la educación básica y media y en tecnicaturas, aunque también a través de juegos, videos de YouTube, o en comunidades de práctica. Es decir, **los ámbitos de formación se diversifican frente a tareas que se simplifican para reducir los costos**.

Pero esta tendencia a la homogenización de competencias simples contrasta con la diversidad de metodologías y tecnologías que se superponen como capas geológicas que se corresponden a su vez con las capas geológicas del software ya desarrollado que requiere mantenimiento y soporte.

Una nota especial requiere la cuestión de las plataformas. La demanda de programadores (así como de otras ocupaciones) comienza a ser gestionada por plataformas globales, como UpWork o Freelancer, que seleccionan, contratan y controlan el proceso de trabajo a la distancia y de forma automática. Ya sin credenciales de competencias y habilidades, estas plataformas evalúan las capacidades antes de contratar a través de un análisis de la reputación, de la complejidad de las tareas realizadas en el historial e incluso a través de ejercicios y pruebas *ad-hoc*. Las metodologías de trabajo, embebidas en las plataformas, **ejercen control sobre el proceso de forma automatizada,** de modo que las instituciones, normas y reglas de trabajo quedan cosificadas en las reglas de la plataforma que son inflexibles ante la variedad.

Machine learning

Otra nota debe hacerse en relación con el aprendizaje automatizado o *machine learning*, que implica que a partir de la exploración de grandes datos, las computadoras puedan hacer inferencias que no fueron programadas sino que fueron "aprendidas" de forma autónoma. Por lo tanto, ***machine learning*** **implica, entre otras cosas, la automatización de la tarea de programación**. Por ejemplo, *Google translate* solía tener más de un millón de líneas de código, con minuciosas instrucciones, y hoy apenas tiene 500, el resto es aprendizaje automático. Sin embargo, la programación de redes neuronales, la ciencia de datos, la inteligencia artificial, tienen la extraña cualidad de reducir los costos de programación y a la vez elevarlos, ya que no requieren programadores que trabajen como escribas del Medioevo todas las instrucciones posibles, sino que ideen creativos algoritmos evolutivos con capacidad de mejora automática. Mientras las tareas simples de programación se aproximan a un *commodity* estandarizado y transado internacionalmente al menor precio posible, emergen nuevas tareas y competencias de una naturaleza distinta sobre la que un nuevo tipo de recursos escasos fijan altos salarios.



Análisis y conclusiones

La historia de la informática que hemos contado hasta aquí es una historia en la que el proceso de competencia en clave schumpeteriana describe una carrera loca por reducir costos laborales al tiempo que complejiza la demanda de trabajo, reemplaza competencias y por lo tanto también los incrementa. Esta carrera amplía el acceso a las competencias mínimas para el uso y el desarrollo de programas de computación. Hay una tendencia en este proceso a la simplificación de las formas de programar y de aprender a programar. Al reducir las barreras a la entrada aumenta la base para la competencia, la emergencia de nuevos bienes y servicios y nuevos actores en el mercado. Las grandes empresas de software de los '70 (Xerox, Texas Instruments...) fueron reemplazadas por las de los '80 y '90 (Microsoft, Oracle...) que hoy son disputadas por las de los 2000 (Amazon, Facebook, Google...). Si bien algunos casos permanecieron (IBM, Mac), lo han hecho sobre la base de redefinir sus modelos de negocios, reorientar su mix de servicios y productos y competir en los nuevos segmentos que se abren de forma continua. Hay dos claves que soportan este proceso. Por un lado, los nuevos desarrollos se apoyan sobre los desarrollos tecnológicos previos, y por el otro, encuentran una demanda creciente explicada por menores costos de acceso y difusión de capacidades de uso.

Como dijimos al comienzo de la nota, solo una ínfima parte de la población puede programar, pero cada vez son más los que utilizan la programación en diferentes ámbitos laborales, sociales o culturales.

La automatización, por lo tanto, desplaza trabajo a la vez que da lugar a la creación de nuevas ocupaciones. Pero la posible aparición de nuevas ocupaciones no debe tranquilizarnos en exceso. Podemos descreer de los pronósticos apocalípticos, pero los motores tecnológicos que están en marcha son reales. Los robots, físicos y virtuales, con capacidad de aprendizaje autónomo, avanzan sobre las tareas manuales, intelectuales e incluso aquellas que involucran habilidades emocionales o capacidades creativas.

Frente a esta amenaza, las reacciones que más atención han recibido son las que ocultan detrás del desempleo tecnológico las crisis sistémicas de sobreproducción, asociadas a la concentración del ingreso y la compresión de las clases medias. Frente a esto promueven como solución mercados de trabajo más flexibles, que recortan derechos e incrementan la desigualdad, so pretexto de que las nuevas tecnologías requieren (y habilitan) agilidad y versatilidad en los procesos de producción.

Probablemente, este camino bloquearía los procesos de diversificación y ampliación de mercados mencionados en este artículo, ya que conducían a una concentración mas que distribución del excedente derivado del cambio tecnológico. La historia que hemos recorrido en el artículo nos muestra que los salarios altos actúan como motor de búsqueda del cambio tecnológico y son a su vez su principal resultado.

Como dice Alan Kay, la mejor forma de predecir el futuro es inventarlo. No hay trayectorias inevitables, sino caminos alternativos de la organización de la producción y de la producción de conocimiento que llevan a diferentes escenarios en materia de cambio tecnológico y distribución de sus excedentes. Institucionalidades alternativas deberían abrir debates entre ciencia abierta y regímenes cerrados de propiedad intelectual y su impacto sobre el cambio tecnológico, o debates sobre distribución del ingreso, incluyendo instituciones laborales, meritocracia e ingreso básico universal.

No hay nada inevitable en el cambio tecnológico más que enfrentarnos al debate sobre el tipo de sociedad en el que queremos vivir.