Controversias y Concurrencias Latinoamericanas
ISSN: 2219-1631
revistacyc.alas@gmail.com
Asociación Latinoamericana de Sociología
Uruguay

Trabajo y Metodologías ágiles[1]

Amorim, Henrique; Reis Grazia., Mauricio

Trabajo y Metodologías ágiles [1]
Controversias y Concurrencias Latinoamericanas, vol. 11, núm. 20, 2020
Asociación Latinoamericana de Sociología, Uruguay
Disponible en: https://www.redalyc.org/articulo.oa?id=588663787012

Dossier

Trabajo y Metodologías ágiles [1]

Trabalho e Metodolgias [2]

Henrique Amorim
Universidade Federal de São Paulo, Brasil
henriqueamorim@hotmail.com

Mauricio Reis Grazia.
Universidade Federal de São Paulo, Brasil
mauricio_reis_grazia@hotmail.com

Redalyc: https://www.redalyc.org/articulo.oa?id=588663787012

Recepción: 06 Febrero 2020
Aprobación: 07 Marzo 2020

Resumen:

El propósito de este artículo es analizar críticamente las metodologías ágiles, presentes en la producción de software. Con este objetivo, nos preguntamos en qué medida esa producción y formas de organización del trabajo reproducen viejas formas de organizar el trabajo, especialmente los Taylor-Fordistas y Toyotistas. Contrariamente a las tesis que presentan el surgimiento del trabajo inmaterial como un momento paradigmático de ruptura con la producción industrial, nuestro objetivo es problematizar el carácter de "novedoso" por el cual se presentan las formas de organización del trabajo en el siglo XXI, aún más, si estas mismas formas no se configurarían como adaptaciones de aquellos procesos de organización laboral característicos del fordismo y del toyotismo, pero que profundizan una nueva frontera productiva poco o nada explorada por el capital en los siglos XIX y XX, la producción inmaterial.

Palabras clave: trabajo (In) material, Metodologías ágiles, industria del software, Toyotismo.

Resumo:

A proposta desse artigo é analisar criticamente as metodologias ágeis, presentes na produção de software. Com este objetivo, nos questionamos em que medida tal produção e formas de organização do trabalho reproduzem antigas formas de organização do trabalho, sobretudo, as taylor-fordistas e as toyotistas. Contrariamente as teses que apresentam a emergência do trabalho imaterial como um momento paradigmático de ruptura com a produção industrial, temos por objetivo debater em que medida o "novo", presente nessas formas de organização do trabalho no século XXI, se configurariam como adaptações do taylor-fordismo e do toyotismo a uma nova fronteira produtiva pouco ou nada explorada pelo capital nos séculos XIX e XX, a produção imaterial.

Palavras-chave: Trabalho (I) material, Metodologias Ágeis, Indústria de Software, Toyotismo.

Introduccion

Las teorías de la sociedad postindustrial parecen guiar parte de la sociología del trabajo al analizar nuevas formas de trabajo y producción, como las circunscritas al sector de la tecnología de la información. A partir de la tesis desde la cual viviríamos en una sociedad estructuralmente distinta a la del industrialismo, se proyectan nuevos tipos de profesiones, trabajos y formas de organización del trabajo que envasan y dan razón a la existencia contemporánea de las sociedades posindustriales.

Sin embargo, trataremos de abordar la organización social contemporánea del trabajo distintamente a como lo hace las teorías postindustriales, y lo haremos porque, en primer lugar, el trabajo inmaterial o la producción inmaterial como un todo – comprendida como la producción de conocimiento, información y comunicación, constituyendo así su materia prima – no evidencia en su determinación que constituya una sociedad postindustrial (o incluso sociedad de conocimiento, inteligencia o información); en segundo lugar, porque en una producción y trabajo inmateriales, ya sea en sus formas más precarias, (como los centros de atención telefónica), o aún en contextos más sofisticados en los que la adquisición de información se hace

imprescindible (como lo es para la programación y el desarrollo de software), presentan en realidad más evidencias de características de una organización típicamente industrial, (generalmente presentados como "nuevas formas"), que de una superación del modelo industrial.

En síntesis, la propuesta de este artículo es, apuntar cómo, por medio del análisis de las denominadas metodologías ágiles aplicadas a la producción de software, aún se sigue en las viejas formas de organizar el trabajo en el ámbito de la producción inmaterial, contrariamente al énfasis que los análisis suelen hacer ante las novedades de las formas contemporáneas de organización de la producción y trabajo y, por ello, aspectos que caracterizarían una supuesta ruptura para con el tradicional modelo de organización industrial.

Es, por lo tanto, una elección metodológica [3] que incurre, como todos las demás, en riesgos. Sin embargo, volver la mirada hacia lo que se reproduce y se preserva en las formas de organización del trabajo en relación con la producción Taylor-Fordista, más específicamente toyotista, cuando analizamos la producción de software, nos permite aclarar, desde la perspectiva del trabajo, las condiciones bajo las cuales están sujetos los sectores de trabajo de tecnología de la información. Es decir, nos permite analizar cuáles serían los factores condicionantes de la organización industrial del trabajo inmaterial con respecto, por ejemplo, a la intervención y la participación de los trabajadores en las decisiones productivas y en la dinámica de sus propias actividades, sobre todo, en un período histórico en el que los trabajadores son designados como colaboradores y, aparentemente, se les requiere una participación creativa e inteligente en las más variadas etapas y frentes de este tipo de producción.

Castells (1999; 2004) pudo observar aspectos característicos de la sociedad de la información que podrían justificar, al menos para una parcela de los trabajadores involucrados en la producción de información, cierta autonomía, posibilitada por la necesidad de adquirir un alto grado de educación formal y que, dada la naturaleza de este tipo de trabajo cognitivo y de producción, serían llevados exhaustivamente a un aprendizaje continuo.

Las observaciones que hace Castells califican nuestro argumento a medida que, y diferentemente, consideramos este aprendizaje, tanto educativo-formal como técnico, como una forma de entrenar al trabajador, tal como lo conceptualiza Gramsci (2011) y que Braverman (1980), y más tarde Días (1997), desarrollarán. Por ello, se hace necesario especificar la naturaleza del conocimiento y la información utilizada en los procesos de trabajo y producción. Es decir, si partimos desde la perspectiva del capital, el conocimiento y la información, que sirven como materia prima para la producción de bienes inmateriales, pueden ser interpretados y reinterpretados, creados y recreados a todo momento como una necesidad del proceso mismo, y cuyo objetivo es aumentar su productividad.

Sin embargo, si uno comienza desde el punto de vista del trabajo, uno debe, ante todo, cuestionarse por la naturaleza de este conocimiento, información y saberes. ¿Serían estas calificaciones educativas algo que de hecho resultaría positivo en las condiciones de trabajo y de vida de los grupos de trabajo o serían calificaciones profesionales como formas de capacitación, es decir, de control para lograr una mayor eficiencia? ¿Y en cuanto a la eficiencia productiva dirigida a reducir la autonomía de estos colectivos para aumentar la productividad? En segundo lugar, sería necesario analizar críticamente qué es lo que se expresa a través de la aparente novedad de los "nuevos tipos de organización del trabajo" que están en gran medida caracterizados por su mayor creatividad e inteligencia, y que exigirían un mayor conocimiento técnico, científico, e incluso emocional, y todo ello en constante actualización. Estos elementos se presentan, en particular, como el objetivo a abordar en nuestro texto.

Cuando analizamos críticamente el trabajo de los programadores y desarrolladores de software, es posible, desde nuestra perspectiva, ir al centro de estos problemas de investigación. Esto se debe a que, además de ser vistos como ejemplos de trabajadores, con autonomía productiva y toma de decisiones en sus actividades laborales, dado su perfil de alto nivel de educación, formación científica, inteligencia y creatividad, estos trabajadores no tendrían una rutinización como característica de su trabajo.

Por lo tanto, enfatizamos que los objetivos de nuestro texto consisten en el análisis y la problematización exclusiva de aquello que, según nuestra opción metodológica, se limita a lo que se preserva y se reproduce en las formas de organización del trabajo, algunas de las llamadas metodologías ágiles. Como maneras específicas de organización y explotación del trabajo en la producción de software, intentaremos demostrar que, contrariamente a las tesis que entienden el uso del conocimiento, una novedad paradigmática, el nuevo presente en las formas contemporáneas de gestión del trabajo, del mismo modo se las puede analizar sobre una doble dimensión: sea como lo nuevo que actualiza formas anteriores de explotación y dominación, necesarias para la dinámica capitalista de renovar sus fuerzas productivas, sea, a un mismo tiempo, como lo nuevo que actualiza estas prácticas de explotación y dominación del trabajo a medida que reproduce estructuralmente la dinámica de producción y trabajo, pero travestido de algo mucho más inteligente, creativo, participativo y emprendedor. Así, las metodologías ágiles caracterizarían la reproducción del carácter parcial y serial de la producción Taylor-Fordista y, al mismo tiempo, un proceso de auto-Taylorización de los grupos de trabajo, ya que se busca, en función de sus prácticas de gestión, la cooptación de la subjetividad del trabajo.

El software Fábricas

Como ya lo habíamos sugerido, nuestro objetivo es analizar críticamente las metodologías ágiles, considerándolas como formas de radicalizar la explotación y organización de la fuerza laboral (in) material. En este sentido, rehacer una historia, incluso resumida, de cómo se desarrolla el trabajo conjuntamente al desarrollo industrial, reproduciendo su subordinación social, basada en la imposición de diversos y diferentes dispositivos tecnológicos, formas de gestión y capacitación técnica y científica, puede llevarnos a formas contemporáneas de organización del trabajo.

Sin embargo, esto nos llevará no hacia una novedad impulsada por las aparentes revoluciones informacionales y/o tecnológicas (basadas en las nuevas tecnologías de información y comunicación - NTIC) que sostengan las sociedades de información, conocimiento y/o inteligencia, sino a la comprensión de un conjunto de prácticas sociales que renuevan antiguas estructuras de dominación, y que Antonio Gramsci (2011) lo llamaría desrevolución pasiva, en que todo se transforma para que nada cambie.

Al reflexionar sobre esta subordinación del trabajo al capital, Karl Marx (2013) presenta, de manera metafórica, su comprensión acerca de la importancia del trabajo para la reproducción del capital como una relación social. Es en este sentido que afirma el autor: "El capital es trabajo muerto, que, como un vampiro, vive solo de succión del trabajo vivo, y vive tanto más cuanto más lo pueda succionar" (Marx, 2013, p. 307, traducción propia). Desde el punto de vista del análisis histórico, Marx nos ofrece una descripción de la transición de la manufactura hacia la maquinaria teniendo como eje central, no un determinismo de la tecnología, sino su determinación por lo social, lo histórico. Preocupado por demostrar cómo cambian los colectivos laborales en función de a las demandas históricas de expansión de capital, Marx indica, además de un proceso gobernado por el desarrollo de fuerzas productivas, que la transición de la manufactura a la maquinaria sirve a un conjunto de intereses sociales ya impuestos.

La máquina o el sistema maquinario, al mismo tiempo que descalifica a la fuerza laboral aún artesanal y presente en las manufacturas, expresa como síntesis los intereses de las clases dominantes. Al profundizar el control sobre las masas trabajadoras, al pasar de una relación de subordinación "externa", mediada por los sueldos, hacia una subordinación "interna" al proceso de producción, se reestructuran las formas de subordinación del trabajo en relación con la dominación del capital. Esto, lejos de ser solo un aparato tecnológico que intensifica el ritmo de producción, es también, y conjuntamente, una forma de dominación, control y subordinación política de los grupos de trabajo por las clases dominantes. Por lo tanto, la transición de la manufactura hacia la maquinaria no rompe de todo con las relaciones de producción capitalista, ni siquiera sirven de base para la constitución de una nueva sociedad.

Al igual que las estrategias de control de la fuerza laboral en la industria de los siglos XVIII y XIX se profundizan con la maquinaria, de modo semejante se lleva a cabo con el taylorismo y el fordismo a fines del siglo XIX y principios del XX. Sin embargo, en este último, el proceso es aún más radicalizado, ya sea por la introducción de preceptos racionalizadores del manejo de Taylor, o por la introducción de la cinta mecánica y un nuevo estilo de vida con Fordismo (Gramsci, 2011, p. 237-282).

Las transformaciones tecnológicas iniciadas en el último cuarto del siglo XX con las nuevas tecnologías de información y comunicación son la base de una serie de debates sobre el papel del trabajo inmaterial en el momento actual de la producción capitalista. La producción inmaterial, entendida aquí como un microcosmos de producción de software, a menudo se señala como el epicentro de una ruptura entre las formas de producción industrial y las "nuevas" formas de producción inmaterial. Por el contrario, entendemos que la producción de conocimiento o información, pese sus diferencias con la producción de bienes físicos, obedece a los mismos principios rectores desde un punto de vista organizacional y, a menudo, como demostraremos, que se los puede utilizar con frecuencia de los estándares de Taylor y Ford, sin dejar espacio, en caso de aumento de la competencia en el mercado, de hacerse uso de la base empresarial tecnológica, organizativa e ideológica del toyotismo.

Si tenemos en cuenta el proceso de subordinación del trabajo al capital con respecto al control y la explotación de los procesos de trabajo, veremos que la industria del software, a pesar de haber surgido sólo en la segunda mitad del siglo XX, presenta una dinámica similar a la de la industria tradicional, sobre todo, en el sentido de que también tuvo una fase "artesanal", sin embargo superada por la producción fundada en la subsunción real del trabajo.

De principio, en la década de 1960 hasta mediados de la década de 1970, la industria del software, especialmente la estadounidense, presentó, según Cusumano (1989), un método "artesanal" de desarrollo de software. La comparación del autor entre el desarrollo de software y la producción artesanal se debe a la aparente ausencia de racionalización de la producción. En la producción artesanal, el software se desarrolló en un proceso de prueba y error que requirió, sobre todo, capacitación técnica para desarrolladores de software y un intenso trabajo creativo. Algo que ayudó a construir una mística de que el trabajador en el sector de la tecnología, específicamente los programadores de software, serían altamente creativos y que ocuparían un lugar diferenciado de los demás trabajadores en el proceso de producción (Xavier, 2008). Estos programadores de software se caracterizaron como trabajadores especiales, ya sea por su alto grado de conocimiento técnico o por las habilidades creativas y emocionales que se hacen imprescindibles para una industria todavía vías de formación.

Sin embargo, la aparente autonomía de estos trabajadores parece estar más vinculada al hecho de que, al principio, la producción de software era pequeña y manufacturada, por lo que no representaba una industria a gran escala. A medida que se desarrolla la producción capitalista, y el software gana más espacio, la autonomía de los primeros desarrolladores es reemplazada por la prescripción de tareas realizadas y la gestión de los procesos de trabajo.

Esto sucedió ya que la producción de software artesanal carecía de un método estructurado o etapas formalizadas del proceso de trabajo que resultaban en procesos de desarrollo caóticos, en los cuales hubo una gran pérdida de recursos y poca eficiencia de los programas, dando lugar, muchas de las veces, a programas de difícil manutención y actualización.

La necesidad de capital para mejorar el desarrollo de software, superando el alto costo y el bajo rendimiento del software artesanal, impulsó, ya en la década de 1970, a la adopción de métodos estructurados de desarrollo de software, acercándose así a los métodos Taylor- fordistas de gestión de producción [4]. La necesidad de aumentar el volumen de producción y su eficiencia llevó a la industria del software a reestructurar sus procesos de trabajo y, sobre todo, sus formas de organización del proceso laboral, centrándose en la racionalización de los procesos de trabajo bien como en establecer un depurado y definido conjunto de códigos. Fue así que el modelo de cascada se hizo popular en las compañías de software.

El modelo de producción en cascada fue difundido por Royce (1970) en su artículo Managing the Development of Large Software Systems, en el que presenta el desarrollo del software en cascada como un modelo secuencial de trabajo con un flujo constante "hacia adelante", es decir, para la próxima etapa de producción, verticalizada. En este modelo, la producción se divide por tareas especializadas, y en la que cada trabajador es responsable de sólo partes específicas del proyecto. La línea de comunicación es vertical (flujo único), hay una alta burocracia y una mayor cantidad de trabajadores.

Entre nuestros entrevistados, se destaca un supervisor de proyecto de una compañía de desarrollo de software que informa la importancia de implementar un modelo estructurado para la "profesionalización" de la compañía.

> "Al principio solo trabajaban los propietarios, no había forma de hacerlo de esa manera y crecer competitivamente en el mercado. Luego, poco a poco, los propietarios se dieron cuenta de que necesitaban contratar más personas, delegar tareas, en resumen, crear una cierta burocracia para satisfacer la demanda que estaba surgiendo. Cuando tienes más de tres personas involucradas en el desarrollo, no puedes tener esa organización espontánea desde el principio. Es necesario crear protocolos, estándares y reglas para garantizar que todos hablen el mismo idioma. Al principio, este proceso era más rígido, ahora estamos evolucionando y buscando flexibilizar estos procesos sin perder la eficiencia y competitividad que esta organización nos brindó." (ENTREVISTA 4, celebrada en mayo de 2016, nuestra traducción).

Para Royce, el desarrollo de software, al principio, podría dividirse entre Análisis y Codificación. Sin embargo, en cadenas de producción complejas en las que se producen grandes cantidades de bienes, sería necesaria una "compartimentación" más efectiva del desarrollo de software, lo que provocaría un segundo momento de expansión de las etapas de desarrollo de software. De acuerdo con Royce (1970):

> "The analysis and coding steps are still in the picture, but they are preceded by two levels of requirements analysis, are separated by a program design step, and followed by a testing step. These additions are treated separately from analysis and coding because they are distinctly different in the way they are executed. They must be planned and staffed differently for best utilization of program resources" (p. 1).

De este modo, las fases de desarrollo de software en un modelo cascada serían: Requisitos del sistema; Requisitos de software; Análisis; Diseño de programa; Codificación; Pruebas y operación (Royce, 1970, p. 2). Por lo tanto, el desarrollo de software de "cascada" tendría tres fases separadas en total, una de análisis y definición de requisitos, una segunda de diseño y codificación y una fase final de pruebas. La separación entre la definición de requisitos y el segmento de codificación generaría una relación jerárquica entre el diseño y la creación del software, lo que permitiría la rutina y la especialización de la segunda fase para aumentar la escala de producción.

> "The ordering of steps is based on the following concept: that as each step progresses and the design is further detailed, there is an iteration with the preceding and succeeding steps but rarely with the more remote steps in the sequence. The virtue of all of this is that as the design proceeds the change process is scoped down to manageable limits. At any point in the design process after the requirements analysis is completed there exists a firm and closeup moving baseline to which to return in the event of unforeseen design difficulties. What we have is an effective fallback position that tends to maximize the extent of early work that is salvageable and preserved" (Royce, 1970, p. 3).

Como señala Cusumano (1989), los procesos de desarrollo de software comenzaron a estructurarse como un proceso de fabricación fordista, a medida en que presentaban una estandarización de control sobre el trabajo, una mecanización de los procesos de trabajo y su división parcial. Guiados por una línea de producción, los trabajadores especializados, aquellos con bajo grado de calificación técnica, operaban la producción a gran escala.

Las fábricas de software, utilizando métodos estructurados de desarrollo de software, que dividen y secuencian las etapas de desarrollo en una línea de montaje, se parecerían a las fábricas de Taylor-Ford, pero con algunas adaptaciones como el tipo de mercancía y el grado de tecnología disponible, dadas las

peculiaridades de la producción de software, como lo son el almacenamiento digital de códigos y programas para reemplazar las existencias físicas de piezas y la mercancía final [5].

La cinta transportadora en la producción de software adquiere un nuevo aspecto a través del software de gestión de producción que comenzó a permitir la existencia de esta estructura de forma virtual. En otras palabras, estarían presentes allí "(...) virtualmente (...) tanto la alfombra de producción como la figura del capataz". (Xavier, 2008, p. 31, nuestra traducción). En este sentido, las industrias de software reproducen, en gran medida, el mismo ritmo de producción secuencial, rutinario y parcial y las mismas formas de control sobre el trabajo presentes en la industria tradicional Taylor-Fordista, aunque el software presenta peculiaridades [6] en relación con la producción fordista entre los años 1930 y 1960.

Por lo tanto, en contraste con el papel "especial" desempeñado por los desarrolladores en los primeros días de las industrias de software, en las fábricas de software, el trabajo gana contornos precarios a medida que se masifica en detrimento de un rígido proceso de racionalización de tipo Taylor-Fordista, sobre todo, a partir de la década de 1980. La división de tareas fragmenta el proceso de desarrollo y programación, permitiendo que cualquier desarrollador reemplace al creador del programa, lo que socava cualquier necesidad de creatividad, presente en el período artesanal, transformando así al desarrollador en una pieza fácilmente reemplazable. Sin embargo, con un alto grado de complejidad en las actividades de software, los métodos de desarrollo estructurados comenzaron a presentar limitaciones a las necesidades de expansión de capital [7].

Metodologías ágiles

Los métodos estructurados de organización de la producción, como el método en cascada, presentaron un control estricto sobre el desarrollo de software, evitando que las empresas satisfagan una demanda cada vez más diversa y creciente. Por lo tanto, al igual que con la industria tradicional, la industria del software, bajo presión para entregar productos "personalizados" de alta calidad, buscaron en el Toyotismo una forma de hacer que la producción sea más flexible, permitiendo una reducción en los costos de producción basada en la intensificación del trabajo y la flexibilidad de los tiempos y las horas de trabajo. Por lo tanto, a principios del siglo XXI, varias compañías de software introdujeron elementos del método (material) de producción de Toyota a la producción inmaterial, a través de la implementación de metodologías de organización de producción ágiles.

Las metodologías ágiles del desarrollo de software se muestran como una evolución ejemplar del método toyota para el desarrollo de software, ya que su principal preocupación es reducir el costo de producción tanto como sea posible en función de la intensificación del trabajo. En este sentido, tales metodologías están estructuradas en principios, procesos y prácticas que buscan la realización de la producción ajustada propuesta por el toyotismo. Lean Digital, desde el punto de vista del capital, es un conjunto de principios de desarrollo de software que tiene como objetivo generar la mayor cantidad de "valor" posible para el cliente en función de la mejora continua en el uso de los recursos de producción. En otras palabras, Lean ofrece al cliente una solución rápida y de bajo costo, basada en un ciclo de "mejoras" continuas en los procesos de trabajo y los trabajadores.

Como lo demuestran las entrevistas con gerentes y empleados de una empresa de desarrollo de software que utiliza Lean Digital, el control diario de los proyectos en este sistema sigue una estructura disciplinaria básica guiada por los objetivos de la empresa y la gerencia, las demandas de los clientes y el ritmo de trabajo necesario para su finalización del producto. A saber, esta estructura tiene cinco momentos: el comienzo de las reuniones por turnos (donde se realizan los balances del día anterior y la aclaración sobre las siguientes actividades); las tablas de resultados y métricas (tablas claras de resultados logrados y la evaluación métrica de estos resultados, junto con una indicación de lo que debe hacerse); Formularios de mejora continua (evaluación constante y paso a paso del proceso de producción e indicación de posibles soluciones); y Evaluación de resultados

(balance general al final del turno e indicación de lo que debe prepararse en el próximo). Para llevar a cabo este control diario sobre el proyecto, es necesario crear un equipo líder capaz de visualizar las tareas de cada grupo en cada turno y orientar su mejora, estos empleados deben estar en contacto constante con los grupos de trabajo y permitir informes de monitoreo, proporcionando así una actualización constante del progreso del proyecto.

En la práctica, Lean Digital, como otras metodologías ágiles, busca construir una especie de "ciclo de vida" de la programación de software. Un modelo circular en el que el cliente toma un papel central en el proceso de definición del sistema. Si tomamos como ejemplo la propuesta de estructuración de desarrollo de software de Royce en la que se encuentran las fases de Requisitos del sistema, Requisitos de software, Análisis, Diseño del programa, Codificación y Pruebas y operación, el ciclo de desarrollo de Lean Digital crearía una ruta de comunicación más flexible entre las etapas con un flujo de información no verticalizado en el que, en cada etapa de desarrollo, todas las etapas son realizadas y reevaluadas por el cliente para que el software sea lo más útil posible para él. Por lo tanto, la definición de los requisitos, el análisis y el diseño se llevan a cabo constantemente por los líderes del equipo y los clientes. Las fases de prueba las realiza el propio cliente que recibe pequeñas versiones del software solicitado.

Por lo tanto, observamos que, más que una autonomía de los desarrolladores, las metodologías ágiles ponen todo el foco en la idea de valor para el cliente, es decir, es el cliente quien determina lo que debe hacerse y evalúa el resultado mientras el software todavía está funcionando, lo que resulta en la necesidad de una comunicación intensa entre el mismo y el grupo responsable del proyecto. La eliminación de la distancia entre el cliente y la producción, según lo propuesto por Lean Digital, permite a la compañía entregar software personalizado a las necesidades del cliente. Si bien es el trabajador quien en la práctica realiza la programación y la codificación, no lo hace de manera autónoma, sino que está sujeto a las pautas del cliente y, sobre todo, de la empresa. En otras palabras, la creatividad y el conocimiento técnico del trabajador se someten a la evaluación gerencial del empleado. control de clientes y negocios.

La implementación de estos principios favoreció la satisfacción de las necesidades inmediatas de las empresas, ya que pudieron adaptarse continuamente a la provisión de soluciones específicas y dentro de los plazos establecidos por el cliente. Como nos dice un programador senior,

> A decir verdad, fue un período muy complicado, ya ves, hoy está más tranquilo, estamos acostumbrados, pero cuando estábamos poniendo en práctica Lean, fue difícil. Como es otro mundo, los procesos son diferentes, tenemos que estar más conectados, ser más proactivos, más creativos y dispuestos a colaborar. Fue un poco pesado de seguir, mucha presión. Pero creo que es así, si quieres tener una empresa competitiva necesitas empleados competitivos, así que aquí hay un alto rendimiento todo el tiempo (ENTREVISTA 3, gerente de TI, celebrada en mayo de 2016, nuestra traducción).

Sin embargo, desde el punto de vista del trabajo, Lean Digital es una forma radical de control e intensificación del ritmo de trabajo, ya que las llamadas "mejoras continuas" apuntan a aumentar "continuamente" el rendimiento y la productividad de los grupos de trabajo haciendo que tengan un mayor esfuerzo mental, concentración y disposición en un tiempo de trabajo cada vez más corto. Como explica un desarrollador de software que utiliza Lean Digital:

> "Hubo cambios importantes en el personal. Llegué aquí un año o dos antes de comenzar este negocio Lean y había gente aquí. A medida que la compañía adoptó estas prácticas, la gente se fue, muchas personas porque no podían superar los estándares de calidad que la empresa quería e incluso hubo personas que comenzaron a volverse un poco locas, ya sabes, siguieron trabajando todo el tiempo, no descansaron, no tenían la familia y luego comenzó a generar problemas en los proyectos. Entonces, el único momento que vi aquí durante un gran intercambio de personas fue este, el resto está muy tranquilo aquí. La mayoría del personal ya no se va (ENTREVISTA 8, programador, celebrada en junio de 2016, nuestra traducción).

Por lo tanto, vemos que las industrias de software y los entornos virtuales permiten un intenso monitoreo, regulación y parcialización del trabajador colectivo. El reloj de tiempo, el vigilante, el entorno panóptico y tantos otros elementos de los procesos de trabajo son absorbidos por el software que controla y regula

la producción, esta absorción se radicaliza por las prácticas de gestión tayloristas y por la cooptación del trabajador subjetivo implementado como el americanismo y el fordismo.

Desde el punto de vista de la empresa, las metodologías ágiles optimizan la producción en la medida en que expanden su dominio sobre el trabajo al reconfigurar la capacidad de intervención colectiva de los trabajadores en los procesos de trabajo y producción. Sin embargo, desde el punto de vista del trabajo, son formas de expandir y radicalizar el control sobre el mismo, ya que además de reproducir los patrones de producción en serie del taylor-fordismo, se basan en una práctica organizacional que presupone un compromiso necesario del trabajador de ideología gerencial.

Por lo tanto, existe un doble movimiento de subordinación del trabajo en relación con el capital. Primero, a los preceptos, reglas, funciones y estrategias impuestas por el software de la máquina de producción y vigilancia y, segundo, por una nueva práctica de convencimiento subjetivo que termina creando una gestión internalizada en el trabajador, lo que llamamos aquí la auto-taylorización del trabajo.

La organización del trabajo se vuelve aún más radical en el sentido exacto de que la gestión, como un proceso de racionalización objetiva de las actividades laborales, se internaliza en un proceso de autoayuda que presiona al trabajador, para saber más, buscar más, incluso si esto ("lo máximo") no le ofrece autonomía en el trabajo o mejora en sus condiciones de trabajo y de vida.

Conclusión

Hay dos cuestiones que nos parecen centrales y que nuestro texto busca demostrar. El primero se refiere a cómo las industrias inmateriales y el trabajo inmaterial son expresiones renovadas de la misma forma organizativa que se adapta a las fronteras de la producción de información, conocimiento y comunicación. Este proceso puede observarse mediante la reanudación histórica de cómo el trabajo se subordinó al capital y mediante su comparación con las formas de organización del trabajo en el sector del software, basadas en metodologías ágiles, como Lean Digital. En base a esta comparación, vemos algunas similitudes entre cómo el trabajo está subordinado al capital en las industrias tradicionales y en las de software.

La trayectoria de pérdida de control sobre el proceso de trabajo que se ha profundizado desde la fabricación, con su producción aún artesanal, pasando por la maquinaria en la que aún se observa la presencia del trabajador como articulador del proceso de trabajo, profundizando en las prácticas de la racionalización del trabajo con el taylorismo y la producción en una serie fordista, también se puede observar en la trayectoria del desarrollo histórico de la producción de software.

Sin embargo, para satisfacer las demandas de capital, las industrias de software comienzan a racionalizar los procesos de trabajo a la manera del taylorismo y también incorporan las técnicas de control y producción en masa del fordismo, basadas en líneas de trabajo virtuales, en el flujo vertical. producción, trabajo especializado y altamente reemplazable y control sobre el ritmo de trabajo En este sentido, por un lado, los métodos estructurados agregados a los programas de control de calidad terminan sometiendo al desarrollador de software a condiciones similares a las del trabajador Taylor-Fordista y, por otro lado, toman el control directo sobre el proceso de desarrollo, eliminando los elementos que les dieron cierta autonomía productiva.

Las prácticas toyotistas han desarrollado la organización del trabajo, ya sea material o inmaterial. Sin embargo, no exceden los preceptos estructurales racionalizadores de la producción en serie. En la práctica, hay una doble subordinación: objetivo, en la medida en que el conocimiento, las técnicas y el conocimiento se internalizan mediante líneas de producción y control virtuales en un proceso típico de taylorización del trabajo, y subjetivo, en el sentido de que el trabajador colectivo e individual tiene la apariencia de autonomía, flexibilidad, participación y creación dentro de estos procesos. Sin embargo, esta apariencia necesaria termina obligándolos a auto-taylorizarse a través de la búsqueda de nuevos conocimientos, nuevas técnicas, nuevas formas de interacción con los clientes, con otros trabajadores y con sus superiores.

Entonces nos preguntamos: ¿qué aportan realmente la producción y la industria de software? En su discurso, las metodologías ágiles, presentes en los procesos de trabajo de programación de software, se presentan como una novedad aparente. Sin embargo, existe una brecha entre la apariencia innovadora y la práctica de la producción. Cuando analizamos el funcionamiento de metodologías ágiles, nos damos cuenta de que tales formas de organización del trabajo son adaptaciones de la lógica taylorista, de la producción en masa parcial y mecanizada del fordismo junto con la flexibilización del trabajo y los trabajadores presentes en el toyotismo.

Con esto, este "nuevo" presente en la producción de software se configura como un viejo conocido, como una adaptación de las formas de producción, ahora inmateriales, a un nuevo nivel tecnológico y social. Por lo tanto, la industria del software, en la práctica, preserva las relaciones de control y explotación del trabajo tradicional de la industria, a pesar de que se basan en cintas de correr y gestiones virtuales.

Referencias Bibliográficas

Braverman, H. (1980). " Trabalho e Capital Monopolista: A Degradação do Trabalho no Século XX." Rio de Janeiro: Zahar.

Castells, M. (1999). " A Sociedade em Rede". São Paulo: Paz e Terra.

Castells , M. (2004)." A Galáxia Internet: Reflexões sobre Internet, Negócios e Sociedade". Lisboa: Fundação Calouste Gulberkian.

Cusumano, M. (1898)." The Software Factory: A Historical Interpretation". New York: Oxford University Press.

Dias, E. A. (1997). "Liberdade (im)possível na ordem do capital: Reestruturação produtiva e passivização". Campinas: IFCH/Unicamp.

Fernandes, A. A. e Teixeira, D. D. S. (2004). "Fábrica de Software: implantação e gestão de operações". São Paulo: Atlas.

Gramsci, A. (2011). " Cadernos do Cárcere". Rio de Janeiro: Civilização Brasileira.11

Humprhey, W. (1990). "Managingthe Software Process." Boston: Addison-Wesley.

Marx, K. (2004)." Manuscritos econômico-filosóficos". São Paulo: Boitempo.

Marx, K. (2013). "O Capital". São Paulo: Boitempo.

Matsumoto, Y. (1987). "A Software Factory: An Overall Approach to Software Production", In P. Freeman (Ed.): "Software Reusability." IEEE.

Matsumoto, Y. (1981). "SWB System: A Software Factory". In H. Hunke (Ed.): "Software-Engineering Environments." Amsterdam: North-Holland.

Roselino, J.E. (2006). "A Indústria de Software: o "modelo brasileiro" em perspectiva comparada". Tese de doutorado – Instituto de Economia, Universidade Estadual de Campinas, Campinas, 216 p.

Royce, W. (1970). "Managing the development of large software system." Proceedings of IEEE Wescon", pp. 382-338.

Xavier, C. D. (1980). "Fábrica de software: até que ponto fordista?" Dissertação de Mestrado, Escola Brasileira de Administração Pública e de Empresas da Fundação Getúlio Vargas, Rio de Janeiro, 94 p.

Notas

1 Este artículo es una versión en profundidad del trabajo "A indústria de software e as metodologias ágeis como estratégia de taylorização e auto-taylorização do trabalho imaterial", publicado en Século XXI: Revista de Ciências Sociais, v. 8, n. 2, p. 747-776, dez. 2018.

2 Este artículo es el resultado de una investigación desarrollada con el apoyo de la Bolsa de Produtividade em Pesquisa (PQ) y del Auxílio Universal do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) y con el Auxílio Regular da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). Agradecemos la revisión y las consideraciones del Grupo de Investigación de Clase Social y Trabajo en la Universidad Federal de São Paulo (GPCT-UNIFESP) en el que se desarrolló este artículo. Agradecemos la revisión y las consideraciones del Henrique Oliveira.

3 Realizamos 11 entrevistas con teleoperadores de centros de llamadas, desarrolladores de software y programadores y miembros sindicales en París de febrero a julio de 2017. Además de estas entrevistas, realizamos otras 18 en una gran empresa especializada en desarrollo y programación de software ubicada en el centro de alta tecnología de Campinas, entre 2015 y 2016

4 El enfoque entre el método estructurado de desarrollo de software y el fordismo aparece en la bibliografía sobre el tema con varios enfoques. En Cusumano (1991, 1989 y 1987), las características más llamativas de este enfoque son la reutilización de códigos y la división entre investigación y desarrollo de software. En Fernandes y Teixeira (2004), la estandarización de tareas y procesos burocráticos. En Matsumoto (1981 y 1987), se destaca la reutilización de códigos, la estandarización de tareas y la producción a gran escala. Y en Humphrey (1990), la producción a gran escala aparece como el elemento central de esta comparación.

5 En el desarrollo de software, los entornos de trabajo virtuales crean una línea de producción virtual para el software, dando la impresión al observador que lo compara con una fábrica física, que no hay una línea de montaje, un flujo vertical de información, la repetición de tareas y supervisores. Como describe Xavier (2008, p. 31, nuestra traducción): "Es difícil identificar cómo se estructuran los equipos / competencias, cuál funciona en qué etapa de producción, cuáles serían estas etapas o cuál es el papel de cada empleado en el proceso de producción".

6 Como ejemplo, tenemos la presencia de una línea de ensamblaje virtual, en la cual los trabajadores no se encuentran frente a una cinta transportadora, sino que están dispuestos en puestos frente a sus computadoras, trabajando en la codificación de un programa a través de una integración vertical de producción que se basa, como el fordismo, en el control centralizado de todos los factores que influyen en la producción.

7 El cambio del desarrollo artesanal a un modelo de fábrica de desarrollo de software es el resultado de una determinación sociohistórica. Este cambio comenzó a mediados de la década de 1970 y ganó fuerza en la década de 1980 con la adopción de certificados de control de calidad como Capability Maturity Model. En la práctica, el CMM se presenta como un conjunto rígido de prácticas a seguir para obtener un software de calidad, es decir, una receta lista para seguir, con estándares y prácticas preestablecidos. Es en este contexto que Estados Unidos comienza a ganar protagonismo en el sector. Con el apoyo del departamento de defensa, las empresas comenzaron a adoptar sistemas de gestión de desarrollo de software inspirados en la Gestión de Calidad Total. Este sistema parte de la idea de que la calidad del producto depende del control estricto del proceso de producción, por parte de los proveedores de materiales. hasta el final del proceso de trabajo. En este sentido, desde la década de 1980, las llamadas fábricas de software se han vuelto cada vez más comunes. Para más información, Cusumano (1989 y 1991) y Roselino (2006).