Recebido em: 31/07/2018.
Aprovado condicionalmente em: 02/11/2018.
Aprovação final em: 10/12/2018.

TAYLORIZAÇÃO E AUTO-TAYLORIZAÇÃO DO TRABALHO: AS METODOLOGIAS ÁGEIS NA INDÚSTRIA DE SOFTWARE.*

TAYLORIZATION AND SELF-TAYLORIZATION OF LABOR: THE AGILE METHODOLOGIES IN THE SOFTWARE INDUSTRY.

TAYLORISATION ET D'AUTO-TAYLORISATION DU TRAVAIL: LES MÉTHODOLOGIES AGILES DANS L'INDUSTRIE DU LOGICIEL.

TAYLORIZACIÓN Y AUTO-TAYLORIZACIÓN DEL TRABAJO: LAS METODOLOGÍAS ÁGILES EN LA INDUSTRIA DEL SOFTWARE.

Henrique Amorim**
Maurício Reis Grazia***

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: Metodologias ágeis; Indústria de software; Taylor-fordismo; Toyotismo; Trabalho imaterial.

---

* Este artigo é fruto de pesquisa desenvolvida com o apoio de Bolsa de Produtividade em Pesquisa do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq – PQ) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). Agradecemos a revisão e as considerações do Grupo de Pesquisa Classe Social e Trabalho da Universidade Federal de São Paulo (GPCT – UNIFESP).
** Doutor em Ciências Sociais; Professor Associado de Sociologia do Departamento de Ciências Sociais e do Programa de Pós-Graduação em Ciências Sociais da Universidade Federal de São Paulo (UNIFESP – Campus Guarulhos), Guarulhos, SP, Brasil; E-mail: henriqueamorim@hotmail.com
*** Mestre em Ciências Sociais pelo Programa de Pós-Graduação em Ciências Sociais da Universidade Federal de São Paulo (UNIFESP – Campus Guarulhos), Guarulhos, SP, Brasil; E-mail: mauricio_reis_grazia@hotmail.com

***ABSTRACT**: The purpose of this article is to critically analyze the agile methodologies present in software production. To this end, we question if this production and forms of work organization reproduce old forms of work organization, especially the Taylorist-Fordists and the Toyotists. Contrary to the theses that present the emergence of immaterial labor as a paradigmatic moment of rupture with industrial production, weaim to discuss the if the "new", present in these forms of work organization in the twenty-first century, would be configured as adaptations of taylor-Fordism and toyotismo to a new productive frontier little or nothing exploited by capital in the nineteenth and twentieth centuries: the immaterial production.*

***Keywords**: Agile methodologies; Software industry; Taylor-fordism; Toyotismo; Immaterial labor.*

***RÉSUMÉ**: Le but de cet article est d'analyser de manière critique les méthodologies agiles présentes dans la production de logiciels. À cette fin, nous nous interrogeons sur la mesure dans laquelle cette production et ces formes d'organisation du travail reproduisent des formes typiques d'organisation du travail, notamment les tayloristes, fordistes et les toyotistes. Contrairement aux thèses qui présentent l'émergence du travail immatériel comme un moment paradigmatique de rupture avec la production industrielle, nous souhaitons discuter de la mesure dans laquelle le «nouveau», présent dans ces formes d'organisation du travail au XXIe siècle, serait configuré comme adaptations du taylor-fordisme et du toyotisme à une nouvelle frontière productive peu ou rien exploitée par le capital aux XIXème et XXème siècles, la production immatérielle.*

***Mots-Clés**: Méthodologies agiles, Industrie de logiciel; Taylor-fordisme; Toyotisme; Travail immatériel.*

***RESUMEN**: La propuesta de este artículo es analizar críticamente metodologías ágiles, presentes en la producción de software. Con este objetivo, nos preguntamos en qué medida tal producción y formas de organización del trabajo reproducen formas típicas de organización del trabajo, sobre todo, las taylor-fordistas y las toyotistas. Contrariamente a las tesis que presentan la emergencia del trabajo inmaterial como un momento paradigmático de ruptura con la producción industrial, tenemos por objetivo debatir en qué medida el «nuevo», presente en esas formas de organización del trabajo en el siglo XXI, se configuraría como adaptaciones del taylor-fordismo y el toyotismo a una nueva frontera productiva poco o nada explotada por el capital en los siglos XIX y XX, la producción inmaterial.*

***Palabras clave:** Metodologías ágiles; industria del software; Taylor-fordismo Toyotismo; Trabajo inmaterial.*

1 INTRODUÇÃO

As teorias da sociedade pós-industrial parecem balizar parte da sociologia do trabalho quando da análise de novas formas de trabalho e produção, como aquelas circunscritas ao setor de tecnologias da informação. Partindo da tese que viveríamos em uma sociedade estruturalmente distinta do industrialismo, projetam-se novos tipos de profissões, empregos e formas de organização do trabalho que dariam fundamento e razão à existência contemporânea de sociedades pós-industriais.[1]

Em sentido diverso dessas teorias, nos balizamos por dois argumentos que procuraremos debater neste artigo: em primeiro lugar, o trabalho imaterial e, de maneira estrita, a produção imaterial (aquela que tem como matérias-primas o conhecimento, a informação e a comunicação) não se configuram como indícios para a determinação de uma sociedade pós-industrial (ou mesmo de uma sociedade do conhecimento, da inteligência ou da informação); pois, em segundo lugar, em tais produção e trabalho imateriais, seja em suas formas mais precárias, como nos *call centers*, seja nos contextos em que se pressupõe a aquisição da informação como forma necessária da produção, como na programação e desenvolvimento de softwares, há mais indícios da presença da organização gerencial tipicamente capitalista, mesmo que em certos casos de *"novo tipo*", que de sua superação.

Em síntese, a proposta de nosso artigo é, em vez de ressaltarmos as novidades nas formas contemporâneas de organização da produção e do trabalho informacionais, como elementos de ruptura da organização industrial, demonstrar, com base na análise das chamadas *metodologias ágeis* no interior da produção de softwares, o que se reproduz das antigas formas de organização do trabalho na produção imaterial[2]. Trata-se, assim, de uma escolha metodológica

1 Grosso modo, as teorias da *"sociedade pós-industrial"* reivindicam, cada uma à sua maneira, modelos de sociedade surgidos da chamada "sociedade industrial". Entre estes modelos, podemos citar: a sociedade do conhecimento com Touraine (1970) e também com Bell (1973); a sociedade pós-materialista com Inglehart (1977); a sociedade da informação com Castells (1999) e Melucci (1980); a sociedade baseada nas atividades intelectuais com Gouldner (1979); e a sociedade de serviços com Offe (1989) e também com Touraine (1989). Para uma discussão mais detalhada destas teorias, ver: Amorim (2015).
2 Para uma discussão mais detalhada do setor de tecnologia da informação, sobretudo, para compreensão de novos perfis profissionais e de um panorama de funções e qualificações profissionais, ver, por exemplo: Bridi e Motim (2014); sobre o trabalho em *call centers*, suas formas de organização e adoecimento do trabalhador, ver: Dutra (2014); e sobre o mercado de trabalho em TI, ver: Duarte (2013).

que incorre, como todas as outras, em riscos. Não obstante, voltar os nossos olhos para o que se reproduz e se conserva nas formas de organização do trabalho em relação à produção taylor-fordista ou mais especificamente à toyotista, quando analisamos a produção de software como microcosmo da produção e do trabalho imaterial, nos permite debater, do ponto de vista do trabalho, as condições pelas quais os coletivos de trabalho em tecnologia da informação estão submetidos. Isto é, nos permite analisar quais seriam os condicionantes da organização industrial do trabalho imaterial no que tange, por exemplo, à intervenção e à participação dos trabalhadores nas decisões produtivas e na dinâmica de suas próprias atividades, sobretudo, em um quadro histórico em que os trabalhadores e trabalhadoras são designados como colaboradores e aparentemente chamados à participação criativa e inteligente nas mais variadas etapas e frentes deste tipo de produção.

Para além disso, nossa motivação encontra-se também influenciada pelo debate sobre a autonomia *do* e *no* trabalho. A bibliografia marxista, sobretudo, aquela preocupada com o desenvolvimento das forças produtivas (materiais e/ou imateriais) como motor da transformação social (como, por exemplo, Radovan Richta, André Gorz e Serge Mallet)[3] identificava, no surgimento do operário politécnico, ao final do anos 1960, o momento de retomada da autonomia política no interior dos processos de trabalho e produção. Sem entrar nessa polêmica, já discutida em outro momento[4], o fundamento central das análises destes autores caracterizava-se por um determinismo tecnológico, impulsionador e base para um devir social libertador. Castells (1999; 2004), de outra forma e mais contemporaneamente, observou diferenças claras na sociedade da informação que fundamentariam, pelo menos para parte dos trabalhadores envolvidos na produção informacional, certa autonomia, sobretudo, por conta da necessidade de adquirem um alto grau de educação formal e de, por conta da natureza deste tipo de trabalho e de produção cognitivos, serem forçados a aprender e reaprender cotidianamente.

3 Nos referimos, entre outros livros, a Estratégia Operária e Neocapitatlaismo (Gorz, 1968); Socialismo Difícil (1968a); La Nouvelle Classe Ouvrière (Mallet, 1969); Le Pouvoir Ouvrier: Bureaucratie ou Democratie Ouvrière (Mallet, 1971) e a Economia Socialista e Revolução Tecnológica (Richta, 1972).
4 Ver, por exemplo: Amorim (2006 e 2009).

Esta perspectiva defendida por Castells é importante para qualificarmos nosso argumento, na medida em que, diferente dele, consideramos a formação, tanto educacional-formal, quanto técnica uma forma de adestramento do trabalhador à maneira como Gramsci (2011) opera tal argumento e que Braverman (1980) e depois Dias (1996)[5], o desenvolvem. Com isso, torna-se central precisarmos qual a natureza do conhecimento e das informações utilizadas nos processos de trabalho e produtivo. Isto é, se se parte do ponto de vista do capital, o conhecimento e a informação, que servem de matéria-prima para a produção de mercadorias imateriais, devem ser interpretadas e reinterpretadas, criadas e recriadas a todo momento como uma necessidade do próprio processo de produção que visa a aumentar sua produtividade. No entanto, se se parte do ponto de vista do trabalho deve-se, em primeiro lugar, questionar a natureza destes conhecimentos, informações e saberes. Seriam eles qualificações para o trabalho, isto é, algo que de fato se fundamentaria como positivo às condições de trabalho e de vida dos coletivos de trabalho ou se trataria de qualificações profissionais como formas de adestramento, a saber, de controle para o alcance de eficiência e eficácia produtivos que visam a redução da autonomia destes coletivos para a ampliação da produtividade? Em segundo, seria necessário analisar criticamente o que se expressa por meio da aparente novidade dos "novos tipos de organização do trabalho" que se caracterizam como criativos, inteligentes e que demandariam maior e sempre mais atualizados saberes técnicos, científicos e até emocionais, o que se apresenta, particularmente, como objetivo de nosso artigo.

Ao analisarmos criticamente o trabalho dos programadores e desenvolvedores de softwares, é possível ir ao centro destes problemas de pesquisa. Isto porque, além de terem sido situados como exemplos de trabalhadores com autonomia produtiva e de tomada de decisões em suas atividades de trabalho, por conta de seu perfil com alto grau de escolarização, com formação científica, com inteligência e criatividade, estes trabalhadores não teriam a rotinização como característica de seu trabalho.

5 Em síntese, Dias (1997, p. 96) destaca que: "Para nossa finalidade estaremos distinguindo adestramento de qualificação no preciso sentido de que sob o capitalismo a qualificação é (por mais sofisticada que apareça) uma forma de adestramento. Estaremos reservando a noção de qualificação para designar a qualificação para o Trabalho e não para o Capital".

Gorz (2003), por exemplo, conferiu a este tipo de produção a impossibilidade de rotinização, sobretudo, por que a matéria-prima e os meios de produção na produção imaterial seriam: o *"discernimento"*, a *"capacidade de enfrentar o imprevisto"* e de *"identificar e resolver os problemas"* (Gorz, 2005, p. 18)[6]. Assim, nos parece possível questionarmos, de um lado, de que tipo de conhecimento e saberes estamos falando quando se valora positivamente a autonomia no trabalho, a satisfação profissional, a liberdade produtiva de programadores e desenvolvedores de software, particularmente, e do trabalho na produção imaterial, em geral e, de outro, diagnosticarmos os fundamentos de um tipo de produção que tem, como neste caso específico, a mesma base da produção material[7], isto é, o toyotismo: suas práticas gerenciais e produtivas, e suas formas de cooptação ideológica baseadas no engajamento objetivo e subjetivo do trabalhador como parte do corpo social da empresa.

Feita essa breve introdução, reforçamos, portanto, que os objetivos de nosso artigo se encerram exclusivamente em analisar e problematizar, com base em uma opção metodológica que se restringe àquilo que permanece, àquilo que se conserva e se reproduz nas formas de organização do trabalho, algumas das chamadas *metodologias ágeis*. Como formas específicas de organização e de exploração do trabalho na produção de software, temos como objetivo

6 O que abriria, nos termos de Gorz, para a discussão, de uma *"economia do conhecimento"*, de um *"comunismo do saber"* ou mesmo de uma sociedade da inteligência, como nos lembra Gianinazzi (2016, p. 312). Como salientei em outro momento: para Gorz, *"(...) a produção imaterial abriria espaço para organizações societárias alternativas às capitalistas, já que em seu interior prevaleceriam atividades que não poderiam ser apreendidas, nem racionalizadas (ou externalizadas pelo capital)"* (Amorim, 2014, p. 118). Em termos semelhantes, Rosenfield (2011, p. 207) destaca que: *"O trabalho – como padrão, o que não significa a inexistência de trabalho taylorista, precário, penoso ou embrutecedor – tornou-se mais variado e mais complexo, seu conteúdo e sua natureza tornaram-se mais ricos, devido a uma demanda maior de investimento subjetivo e de mobilização da inteligência. O trabalho tornou-se mais instigante e, em muitos casos, imaterial. É possível, pois, supor que esse quadro represente ganhos para os trabalhadores, já que o trabalho tornou-se mais interessante e flexível."*

7 Tomamos aqui a distinção material versus imaterial apenas do ponto de vista do trabalho concreto, isto é, apenas para diferenciar trabalho e produção que tem como centro meios de produção e mercadorias físicas ou não físicas. Não obstante, trata-se aqui apenas de um recurso metodológico, haja vista que do ponto de vista aqui assumido trabalho material e imaterial devem, necessariamente, ser pensados com base no conceito de trabalho abstrato, o que faria dessa aparente dicotomia, uma relação contraditória, importando, com isso, a síntese de múltiplas determinações que uma produção encerra. A tangibilidade ou intangibilidade de um trabalho, produção, meios de produção e da própria mercadoria é fundamental apenas e tão somente para distinguirmos tipos de trabalho e atividades produtivas, entendendo que todos eles são conjuntos de relações sociais determinadas por uma materialidade histórica que envolve elementos que reproduzem um modo de produção e de vida. Discutimos mais profundamente este tema em: Amorim (2009 e 2014).

demonstrar que, contrariamente às teses que entendem a utilização do conhecimento como uma novidade paradigmática, o *novo* presente nas formas contemporâneas de gerenciamento do trabalho pode ser analisado em uma dupla dimensão: primeiro, como um *novo* que atualiza as formas de exploração e dominação, necessárias à própria dinâmica capitalista de renovação de suas forças produtivas, segundo, e ao mesmo tempo, como um *novo* que atualiza tais formas de exploração e dominação do trabalho à medida em que reproduz estruturalmente as dinâmicas de produção e trabalho, no entanto travestidas de inteligentes, criativas, participativas e empreendedoras. Assim, as *metodologias ágeis* se caracterizariam pela reprodução do caráter parcializado e seriado da produção taylor-fordista e, ao mesmo tempo, por um processo de auto-taylorização dos coletivos de trabalho na medida em que procura promover, com base em suas práticas gerenciais, uma forma atualizada de cooptação da subjetividade do trabalho.

2 O CAMINHO PARA AS METODOLOGIAS ÁGEIS

> Diferente das antigas formas de taylorização do trabalho, que davam aos trabalhadores os meios de produção necessários para cumprirem suas tarefas, hoje apesar de continuarmos a ver um processo de parcialização das tarefas e controle sobre o tempo de trabalho tipicamente taylorista, nota-se também um processo de auto-taylorização. No processo de taylorização tradicional não havia nenhum tipo de autonomia do trabalhador. Hoje, há, além das formas tradicionais de controle e pressão sobre o trabalho, uma aparência de autonomia que estabelece que o trabalhador deve encontrar os meios de organização e de qualificação para cumprir suas tarefas. Ou seja, o taylorismo ainda está presente, mas ao mesmo tempo há uma demanda para que os trabalhadores se auto-taylorizem, o que reforça duplamente o controle e a pressão sobre o trabalho. (...) assim, há, ao mesmo tempo, um quadro de pressão em relação ao tempo e ao cumprimento de tarefas que são impostas objetivamente, somado a um processo de auto-taylorização que exige que o traba-

> lhador encontre os meios de organização para realizar seu próprio trabalho (...), inclusive, entre os desenvolvedores e programadores de software. (...) se espera, assim, em adição aos princípios básicos do taylorismo, que o trabalhador seja seu próprio engenheiro de tempo e de método. A aparência, portanto, é de autonomia, mas, na verdade, trata-se de uma autonomia enjaulada que se fundamenta nos preceitos tradicionais do taylorismo, com seu conjunto de regras objetivas, mas que se soma a uma ideologia que estabelece um autogerenciamento que tem como objetivo sempre produzir mais. (...) pouco importando o tempo utilizado e os meios disponíveis para a produção. Na maioria das vezes, o trabalhador ultrapassa sua jornada de trabalho, trabalha nos fins de semana e à noite. Isso não configura uma autonomia. Na verdade, é uma falsa autonomia. (*Sindicalista da Union Syndicale Solidaires, maio de 2017*).

Este trecho de uma entrevista que realizamos no âmbito de uma pesquisa sobre a as indústrias de software e os *call centers*[8], *nos permite traçar o caminho de nossa argumentação: o que estruturalmente permanece das formas taylor-fordista e toyotista na organização do trabalho imaterial, sobretudo, aquela voltada para a produção de softwares?*

Como sugerimos no início do artigo, temos como objetivo analisar criticamente as metodologias ágeis, pensando-as como formas de radicalização da exploração e organização da força de trabalho (i)material. Nesse sentido, refazer um histórico, mesmo que sumário, de como o trabalho vai, ao longo do desenvolvimento industrial, tendo sua subordinação social reproduzida, com base na imposição de variados e diferentes aparatos tecnológicos, formas de gerenciamento e de adestramento técnico e científico, pode nos conduzir às formas contemporâneas de organização do trabalho. Porém, nos conduzirá, não como uma novidade trazida por aparentes revoluções informacionais e/ou tecnológicas (baseadas nas novas tecnologias da informação e comunicação - NTICs) que fundamentariam

8 Realizamos onze entrevistas com teleoperadores de *Call Centers*, desenvolvedores e programadores de software e sindicalistas em Paris de fevereiro a julho de 2017. Além destas entrevistas, realizamos outras vinte e uma em uma grande empresa especializada em desenvolvimento e programação de software situada no Polo de alta Tecnologia de Campinas, entre 2015 e 2018.

sociedades informacionais, do conhecimento e/ou da inteligência, mas como um conjunto de práticas sociais que renovam as estruturas de dominação ao travestir o antigo, que Antonio Gramsci chamaria de revolução passiva (Gramsci, 2004)[9], na medida em que tudo é transformado para que nada seja alterado.

Ao refletir sobre esta subordinação do trabalho ao capital, Karl Marx (2013, p. 307) apresenta, de maneira metafórica, a sua compreensão sobre a importância do trabalho para a reprodução do capital como relação social. É nesse sentido, que o autor afirma: "*O capital é trabalho morto, que, como um vampiro, vive apenas da sucção de trabalho vivo, e vive tanto mais quanto mais trabalho vivo suga"*. Do ponto de vista da análise histórica, Marx nos oferece uma descrição da passagem da manufatura à maquinaria que tem como centro, não um determinismo da tecnologia, mas da determinação do social, do histórico. Preocupado em demonstrar[10] como os coletivos de trabalho se alteram em função das exigências históricas de ampliação do capital, Marx indica, para além de um processo regido pelo desenvolvimento das forças produtivas, que a passagem da manufatura para a maquinaria atende a um conjunto de interesses sociais em presença. A máquina ou o sistema de máquinas, ao mesmo tempo em que desqualifica a força de trabalho ainda artesanal e presente nas manufaturas, expressa como síntese os interesses das classes dominantes. Ao aprofundar o controle sobre as massas trabalhadoras, passando de uma relação de subordinação "externa", mediada pelo assalariamento e pelos limites impostos pelo Estado, a uma subordinação "interna" ao processo produtivo, reestrutura-se as formas de subordinação do trabalho em relação ao domínio do capital. A máquina, com isso, distante de ser apenas um aparato tecnológico que acelera o ritmo de produção, é também e conjuntamente uma forma de dominação, de controle e subordinação política dos coletivos de trabalho. A passagem da manufatura à maquinaria não

9 O conceito de Revolução Passiva de Antonio Gramsci pode ser relacionado a sua análise do Fordismo como uma forma de reestruturação da produção que teria como objetivo restaurar a dominação do capital sobre o trabalho.

10 Aqui nos ocuparemos apenas da descrição de como o desenvolvimento de "novas" tecnológicas, técnicas e formas de gestão se fundamentam como processos de radicalização da exploração e da dominação do trabalho em relação ao capital na medida em que cria um novo tipo de trabalhador adaptado às exigências de crescimento da produtividade.

rompe, assim, com as relações de produção capitalista, nem mesmo servem de base para a constituição de uma nova sociedade.

Do mesmo modo que se aprofundam as estratégicas de controle da força de trabalho na indústria dos século XVIII e XIX com a maquinaria, ao final do século XIX e início do XX, o taylorismo e o fordismo se apresentam com o mesmo objetivo. Não obstante, radicaliza-se este processo, seja pela introdução de preceitos racionalizantes da trazidos pela gerência taylorista, seja pela introdução da esteira mecânica e de um novo estilo de vida com o americanismo e o fordismo (Gramsci, 2011, pp. 237-282).

O taylorismo nos Estados Unidos e Europa Ocidental, assim como as *metodologias ágeis* contemporaneamente, se apresentava como um método científico que buscava organizar eficientemente a produção, procurando domesticar uma força de trabalho politicamente resistente. Além disso, seus preceitos de organização dos estoques, dos fluxos de peças e mercadorias, amparados por um processo de racionalização administrativa, dado pela relação entre tempos e movimentos, manifestou-se *"(...) como uma linguagem, como parte visível de um projeto maior, (...) como uma construção ideológica que deveria ser universalizada"* (Amorim, 2006, p.11).

O fordismo, por seu turno, se apresentou como uma extensão do taylorismo em pelo menos duas frentes: procurava renovar, em conjunto com a gerência taylorista, as formas de controle e produção de mercadorias, acionadas com o incremento da esteira mecânica e da produção e do consumo em massa e, ao mesmo tempo, se caracterizava muito mais como um modo de vida e de organização social que apenas um conjunto de práticas tecnológicas aplicadas à produção. Ao incrementar a produção, o taylor-fordismo carregava consigo a ideologia do progresso técnico e a ideia de que o desenvolvimento social, no sentido da repartição social da riqueza, estaria plasmado à produtividade industrial (Gramsci, 2004).

Como uma aparente substituição do taylor-fordismo, o toyotismo vem responder à diminuição das taxas de lucro e, por consequência, à crise do Estado de bem-estar social[11], ao recorrer a

11 Em uma tentativa de atender à demanda cada vez mais especializada e de poucos produtos, foi necessário que a produção mantivesse um processo produtivo flexível, permitindo ao trabalhador

estratégias enxutas, "científicas", cognitivas e que acionariam a subjetividade do trabalhador, sua participação ativa e um engajamento constante na resolução de problemas e na criação de alternativas criativas. Impõem-se, assim, com a introdução do toyotismo, um novo perfil do trabalhador, substituindo a produção em massa homogeneizada e trabalhadores especializados por uma produção "flexível" com trabalhadores multifuncionais. Como consequência, surge uma nova forma de adestramento que remete, entre outras questões, a aspectos vinculados ao uso seriado das capacidades cognitivas, à obtenção de novas informações e de recriação das antigas.

O método Toyota de produção, conhecido também como *Lean Manufacturing*, surgiu no Japão, na fábrica de automóveis da Toyota, logo após a Segunda Guerra Mundial. O elemento chave da chamada produção enxuta é a combinação de metodologias gerenciais com a automação da linha de produção, orientada por um princípio de produzir mais com o menor número possível de recursos. Assim, em sua estrutura básica, o toyotismo orienta-se pela flexibilização dos processos de trabalho, ao propor a diminuição da burocracia; a reposição imediata de materiais e peças; a capacidade de incorporar as variações de demanda; a redução dos estoques às necessidades dos clientes; a entregado produto no prazo, baseada no estímulo à comunicação constante entre os fornecedores, a equipe de trabalho e o cliente; além da eliminação dos desperdícios, baseada na diminuição contínua das etapas produtivas e do tempo de trabalho.[12] Nesse sentido, o método Toyota pressupõe uma força de trabalho *"engajada"* e *"proativa"*, disposta a intensificar seu ritmo de trabalho e a executar várias tarefas para garantir as necessidades diretas dos clientes. No depoimento de um programador, a relação entre o cliente e o trabalhador fica clara, sobretudo, por conta da intensidade do trabalho que lhe é imposta:

operar mais de uma máquina ao mesmo tempo, rompendo-se com a relação entre homem/máquina presente no taylor-fordismo (Gounet, 2002). A flexibilidade exigida pelo Método Toyota pressupõe, assim, uma flexibilidade do trabalhador, a saber, a flexibilidade sobre o tempo de trabalho e suas formas de contratação. Nesse sentido o toyotismo deve ser entendido como uma estratégia para ferir politicamente a classe operária, aumentando o controle do capital sobre a produção. Dessa forma, a passagem do fordismo para o toyotismo deve ser entendida como uma transformação das técnicas produtivas e gerenciais com a finalidade de permitir mais e melhor exploração do trabalho.
12 Ver sobre a questão da diminuição do tempo de trabalho: Cardoso (2009); Amorim (2013); Grazia (2017).

> A gente está sempre conectado, não tem jeito. Primeiro, porque tem um pessoal que prefere trabalhar de noite. Então surge alguma questão ou problema e eles já compartilham com a gente. Segundo, porque o desenvolvimento de software é uma atividade muita dinâmica, os planos mudam o tempo inteiro, depende do que o cliente passa para gente, de uma dificuldade que alguém encontra e tem que alterar tudo, então a gente tem que estar sempre ligado (Programador de software, entrevista realizada em julho de 2016).

As transformações tecnológicas iniciadas no último quarto do século XX com as novas tecnologias da informação e comunicação fundamentam uma série de debates sobre o papel do trabalho imaterial no atual momento da produção capitalista. A produção imaterial, compreendida aqui como microcosmo da produção de software, é apontada muitas vezes como o epicentro de uma ruptura[13] entre as formas de produção industrial e "novas" formas de produção imaterial. Contrariamente, entendemos que a produção de conhecimento ou de informação, apesar de ter diferenças em relação à produção de mercadorias físicas, obedece aos mesmos princípios balizadores do ponto de vista organizacional. Ela pode, muitas vezes, como demonstraremos, utilizar padrões taylor-fordistas, sem deixar de abrir espaço, caso o grau de inserção da competição mercadológica aumente, para a utilização da base tecnológica, organizacional e ideológica empreendedorista[14] do toyotismo.

Nesse sentido, iremos agora analisar as formas de organização do trabalho na indústria de software, buscando destacar como o "novo" poderia ser interpretado como um "antigo" reposto.

13 Segundo autores como Rosenfield, o trabalho imaterial entendido como uma espécie de capitalismo cognitivo representaria uma superação das formas de organização do trabalho industrial, isto é: *"Quando está em jogo a produção de conhecimentos, como é o caso do capitalismo cognitivo, a cooperação não pode mais ocorrer nos marcos fordista-taylorista, ou seja, por meio da cooperação passiva, estática, garantida pelo encadeamento sequencial e pela adição de tarefas elementares e de funções. A cooperação para a produção de conhecimentos é consubstancial à atividade criativa, marcada pela comunicação horizontal não programada e por um trabalho coletivo, cooperativo e reticular, para além do controle hierárquico"* (Rosenfield, 2011, pp. 209-10).
14 Não é o objetivo deste artigo debater o conceito de empreendedorismo. Contudo, para uma leitura mais aprofundada sobre o tema ver, por exemplo, Harvey (2010); Dardot e Laval (2016); Colbari (2007); e Castro (2016).

3 DO SOFTWARE ARTESANAL À INDÚSTRIA DE SOFTWARE

Se levarmos em consideração o processo de subordinação do trabalho ao capital no que se refere ao controle e exploração dos processos de trabalho, podemos perceber que a indústria de software, apesar de ter surgido apenas na segunda metade do século XX, apresenta uma dinâmica similar à da indústria tradicional, sobretudo, no sentido em que também teve uma fase "artesanal", contudo superada pela produção fundada na subsunção real do trabalho.

Em seu início, nos anos 1960 até meados dos anos 1970, a indústria de software, em especial a estadunidense, apresentou, segundo Cusumano (1989), um método "artesanal" de desenvolvimento do software. A comparação do autor entre o desenvolvimento de software e produção artesanal se dá pela aparente ausência de racionalização da produção. Na produção artesanal, os softwares eram desenvolvidos em um processo de tentativa e erro que exigia, sobretudo, uma capacitação técnica dos desenvolvedores de software e intenso trabalho criativo. Algo que ajudou a construir uma mística de que o trabalhador do setor de tecnologia, especificamente os trabalhadores de software, seriam seres altamente criativos e diferentes que ocupariam um lugar diferenciado dos demais trabalhadores no interior do processo de produção (Xavier, 2008). Os desenvolvedores de software foram caracterizados como trabalhadores especiais, seja pelo seu grau de conhecimento técnico, sejam pelas competências criativas e emocionais tão necessárias em uma indústria em formação.

A falta de um método estruturado ou etapas formalizadas do processo de trabalho acarretou em processos caóticos de desenvolvimento, nos quais havia muito desperdício de recursos, pouca eficiência dos programas dando origem, em inúmeros casos, a programas que apenas o programador original conseguia manter ou atualizar.

A necessidade do capital em aperfeiçoar o desenvolvimento de software, superando o alto custo e o baixo desempenho dos softwares artesanais, impulsionou, já nos anos 1970, a adoção de *métodos estruturados* de desenvolvimento de software, aproximando-se, com isso, dos métodos taylor-fordista de gerenciamento da produ-

ção.[15] Assim, a necessidade em aumentar o volume da produção e sua eficácia conduziu a indústria de software a uma reestruturação dos processos de trabalho e, sobretudo, de suas formas de organização do trabalho, passando a se preocupar com a racionalização e burocratização dos processos de trabalho e a estabelecer um padrão definido e enxuto de códigos.

Royce (1970) apresentou um dos primeiros métodos estruturados para desenvolvimento do software, que ficou conhecido como método *"Waterfall"* ou desenvolvimento em cascata. Em seu artigo, Royce (1970) apresenta o método de produção estruturado em termos que se assemelhavam ao método fordista de produção. O desenvolvimento de software em cascata se configurava como um método sequencial de trabalho, com fluxo constante "para frente", ou seja, para a próxima etapa da produção. Assim, o controle da produção é estabelecido no planejamento e nas previsões de venda feitos pela empresa e não na demanda. Em outras palavras, trata-se de uma produção que se caracteriza pela produção contínua, como nos molde fordistas, na linha de montagem que nunca para e que independe da demanda. Nesse método, a produção é dividida em tarefas especializadas, sendo cada trabalhador responsável por apenas algumas das partes ou uma parte específica do projeto. A linha de comunicação é vertical (fluxo único), possuindo alta burocracia e necessitando de uma quantidade relativamente grande de trabalhadores.

Como aponta Cusumano (1989), os processos de desenvolvimento de software passaram a se estruturar como um processo fabril fordista, na medida em que apresentavam uma padronização do controle do trabalho, uma mecanização dos processos de trabalho e uma divisão do trabalho parcializada. Orientando-se por uma linha de produção, trabalhadores especializados, porém com baixo grau

15 A aproximação entre o método estruturado de desenvolvimento de software e o fordismo aparece na bibliografia sobre o tema com várias abordagens. Em Costa (2003), seria a existência de metas rígidas para evitar desperdício e maximizar o volume da produção que aproxima os dois métodos. Em Cusumano (1991, 1989 e 1987), as características mais marcantes dessa aproximação são a reutilização de códigos e divisão entre pesquisa e desenvolvimento de software. Em Fernandes e Teixeira (2004), a padronização de tarefas e os processos burocratizados. Já em Matsumoto (1981 e 1987), destacam-se a reutilização de códigos, padronização de tarefas e a produção em larga escala. Em Evans (1994), a linha de montagem e o suporte automatizado para desenvolvimento, e em Humphrey (1990) é a produção em larga escala que aparece como elemento central dessa comparação.

de qualificação técnica, operavam uma produção em larga escala. As fábricas de software, ao utilizarem métodos estruturados de desenvolvimento de software, que dividem e sequenciam as etapas de desenvolvimento em uma linha de montagem, assemelhavam-se às fábricas taylor-fordistas, porém com algumas adaptações ao tipo de mercadoria e ao grau de tecnologia disponível, haja vista as peculiaridades da produção de software, por exemplo, pela armazenagem digital dos códigos e programas em substituição aos estoques físicos de peças e da mercadoria final.[16]

A esteira na produção de software ganha uma nova roupagem através de softwares de gerenciamento de produção que passaram a permitir a existência dessa estrutura de modo virtual. Ou seja, estariam presentes ali *"(....) virtualmente (....) tanto a esteira de produção como a figura do capataz."* (Xavier, 2008, p. 31). Nesse sentido, as indústrias de software reproduzem, em grande medida, o mesmo ritmo sequencial, rotinizado de produção e as mesmas formas de controle sobre o trabalho presentes na indústria taylor-fordista tradicional, ainda que o software apresente peculiaridades[17] em relação à produção fordista entre os anos 1930 e 1960.

Assim, em contraposição ao papel "especial" desempenhado pelos desenvolvedores nos primórdios das indústrias de software, nas fábricas de software, o trabalho ganha contornos precários ao se massificar com base em um rígido processo de racionalização fundamentado no taylor-fordismo. O parcelamento de tarefas fragmenta o processo de desenvolvimento e de programação, permitindo que qualquer desenvolvedor substitua o criador do programa, minando assim qualquer necessidade de criatividade, ressaltadas no período artesanal, transformando, com isso, o desenvolvedor em uma peça facilmente substituível. Contudo, como o alto grau de complexidade das atividades

16 No desenvolvimento de software, os ambientes virtuais de trabalho criam uma linha de produção virtual para o software, dando a impressão, para o observador que a compara a uma fábrica física, de que não existe uma linha de montagem, um fluxo vertical de informação, repetição de tarefas e supervisores. Como descreve Xavier (2008, p. 31): "Dificilmente se consegue identificar como estão estruturadas as equipes/competências, qual delas atua em que etapa da produção, quais seriam essas etapas, nem qual o papel de cada empregado no processo produtivo."

17 Como exemplo, temos a presença de uma linha de montagem virtual, na qual os trabalhadores não se encontram frente a uma esteira rolante, mas sim dispostos em baias em frente aos seus computadores, trabalhando na codificação de um programa através de uma integração vertical da produção que se fundamenta, à semelhança do fordismo, no controle centralizado de todos os fatores que influenciam a produção. Sobre o tema da flexibilização dos contratos de trabalho, ver: Castro (2016).

des de software, os métodos de desenvolvimento estruturados passaram a apresentar limitações às necessidades de expansão do capital.[18]

Os métodos estruturados de organização da produção, como o método em cascata, apresentavam um controle rígido sobre o desenvolvimento de software, não permitindo às empresas atender a uma demanda cada vez mais diversificada e crescente. Portanto, assim como ocorreu com a indústria tradicional, a indústria de software, pressionada a entregar produtos personalizados com alta qualidade, busca no toyotismo uma forma de flexibilizar a produção permitindo uma redução dos custos de produção com base na intensificação do trabalho e da flexibilização dos tempos e da jornada de trabalho.[19] Dessa forma, no início do século XXI, diversas empresas de software adaptaram elementos do método toyota de produção (material) à produção imaterial através da implementação de *metodologias ágeis* de organização da produção.

As metodologias ágeis de desenvolvimento de software, mostram-se como uma evolução exemplar do método toyota para o desenvolvimento de software, visto que sua principal preocupação é reduzir ao máximo o custo de produção com base na intensificação do trabalho. Nesse sentido, tais metodologias se estruturam em princípios, processos e práticas que buscam a efetivação da produção enxuta proposta pelo toyotismo. O *Lean Digital*, do ponto de vista do capital, é um conjunto de princípios de desenvolvimento de software que tem por objetivo gerar a maior quantidade de valor possível ao cliente através do aperfeiçoamento contínuo na utilização dos recursos de produção. Ou seja, o *Lean* oferece ao cliente uma solução rápida e de baixo custo, baseada em um ciclo de "melhorias" contínuas dos processos de trabalho e dos trabalhadores.

18 A mudança do desenvolvimento artesanal para um modelo fabril de desenvolvimento de software é fruto de uma determinação sócio-histórica. Tal mudança teve início já na metade dos anos 1970 e ganhou força nos anos 1980 com a adoção de certificados de controle de qualidade como o *Capability Maturity Model*. Na prática o CMM apresenta-se como um conjunto rígido de práticas a serem seguidas para conseguir obter um software de qualidade, ou seja, uma receita pronta para se seguir, com normas e práticas pré-estipuladas. É nesse contexto que os Estados Unidos começam a ganhar destaque no setor. Através do apoio do departamento de defesa, as empresas passaram a adotar sistemas de gestão de desenvolvimento de software inspirados no *Total Quality Management*. Este sistema parte da ideia de que a qualidade do produto depende de um controle rígido sobre o processo de produção, dos fornecedores de materiais até o fim do processo de trabalho. Nesse sentido, a partir dos anos 1980, torna-se cada vez mais comum as chamadas fábricas de software. Para mais ver: Cusumano (1989 e 1991) e Roselino (2006).

19 Sobre o tema da flexibilização dos contratos de trabalho, ver Castro (2016).

Como apresentado no Manifesto Ágil de 2001, as metodologias ágeis deveriam seguir 12 princípios:

> 1. Satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado. 2. Mudar os requisitos mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente. 3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo. 4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto. 5. Construir projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho. 6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face. 7. Software funcionando é a medida primária de progresso. 8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente. 9. Contínua atenção à excelência técnica e bom design aumenta a agilidade. 10. Simplicidade: a arte de maximizar a quantidade de trabalho não realizado é essencial. 11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis. 12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento (Manifesto Ágil, 2001).

A implementação desses princípios favoreceu o atendimento das necessidades imediatas das empresas na medida em que elas se tornaram capazes de se adaptar continuamente no fornecimento de soluções específicas e dentro dos prazos estabelecidos pelo cliente. Entretanto, do ponto de vista do trabalho, o *Lean Digital* é uma forma radicalização do controle e da intensificação do ritmo de trabalho na medida em que as chamadas melhorias continuas têm por objetivo aumentar "continuamente" a performance e a produtividade dos coletivos de trabalho, fazendo-os dispor de maior esforço mental, concentração e disposição em um tempo de trabalho cada vez menor.

Para tanto, a implementação do *Lean Digital* busca seguir o formato de uma produção puxada pela demanda, na qual são adotadas comunicações visuais simples para permitir fácil identificação aos desenvolvedores e programadores da tarefa que precisa ser realizada. Dessa forma, a produção deve se orientar, pelo ritmo das etapas e pela necessidade imediata da produção e do cliente, fundamentando-se, por exemplo, em processos como o Scrum, o Extreme Programming e o Kanban.

O Scrum foi originalmente utilizado para a fabricação de automóveis e suas práticas e princípios estão descritos no artigo *The new product development game* de Takeuchi e Nonaka (1986), no qual é apresentado um conjunto de processos e etapas para o desenvolvimento de software fundamentado, em preceitos do toyotismo, a saber com equipes de trabalhadores multidisciplinares em uma linha de produção customizável aos diferentes tipos de software.

O *Scrum* não apresenta uma definição sistemática das técnicas a serem utilizadas para "aprimorar"[20] a produção. Na verdade, ele se sintetiza em uma proposta de ciclo contínuo de melhorias que orienta os trabalhadores para um ambiente caracterizado como flexível e adaptado a mudanças. Em linhas gerais, o Scrum funciona como um jogo de teatro no qual os trabalhadores são orientados a se comportarem de acordo com o papel que devem desempenhar. A ideia central é a de que a produção possua variáveis diversas e com alto grau de transformação como, por exemplo, novos requisitos, prazos, funcionalidades e aprimoramento técnico. Este cenário se fundamenta na ideia de que o desenvolvimento do software seria uma tarefa complexa e difícil de ser previamente organizada. Dessa forma, o Scrum busca convencer ideologicamente o trabalhador a seguir um determinado tipo de conduta necessária à produção reafirmando a necessidade de um perfil de trabalhador flexível e proativo.

O Scrum funciona, portanto, como uma peça de teatro que poderia ser intitulada: como organizar a produção imaterial. Os papeis seriam contracenados pelo cliente, pelo gerente, pela equipe scrum e pela equipe master. O cliente é responsável por indicar as

20 Vale ressaltar que *aprimorar*, do ponto de vista da empresa, é acelerar a produção com vistas ao aumento de produtividade. Do ponto de vista do trabalho, trata-se de intensificar o ritmo de trabalho com base no aumento de performance do trabalhador.

principais funcionalidades do software. O gerente é o responsável por coordenar de forma global os grupos de trabalho, apresentar relatórios de desempenho e verificar se o projeto está dentro das especificações e no prazo. A Equipe Scrum é a equipe de trabalho. É dela a responsabilidade de definir quais parâmetros técnicos são necessários para a realização de determinado projeto. Seu trabalho é orientado pelos pareceres do cliente e pelos relatórios de desempenho fornecidos pela gerência. A equipe de trabalhadores deve ser capaz de indicar de forma clara uma estimativa do esforço necessário para a implementação de cada funcionalidade, indicando obstáculos que precisam ser eliminados do processo. Por fim, o Scrum master é um dos "líderes" do projeto, sendo sua responsabilidade garantir que todos os elementos da cadeia produtiva estejam funcionando de acordo com os princípios do Scrum e que o software esteja de acordo com as especificações demandadas pelo cliente.

Após a atribuição dos papéis, dá-se início ao ciclo Scrum que, grosso modo, pode ser descrito por em 5 etapas: 1. *Product Backlog*; 2. *Sprint Backlog*; 3. *Sprint*; 4. *Sprint Retrospective* e 5. *Working Software*. O Scrum funciona com um ciclo de interação constante entre a equipe desenvolvedora e o cliente através da realização de Sprints (ciclos de desenvolvimento) constantes que devem ser avaliados pelo cliente. Assim, na primeira etapa são definidos a "lista de desejos do cliente", ou seja, as funcionalidades que o software deviria ter. Em seguida são estipuladas as metas de cada Sprint a serem realizadas pela equipe, ou seja, são divididas as etapas de desenvolvimento em pequenos ciclos de desenvolvimento que devem produzir constantemente partes do software para a avaliação do cliente. Nas fases finais são realizadas reuniões para balanço do trabalho e entrega do software ao cliente. Dessa forma, o Scrum é, em última instância, um processo de gestão do desenvolvimento que se fundamenta em uma divisão das etapas de programação do software em pequenos ciclos de produção, com a entrega constante de etapas do software para a avaliação do cliente. Sendo assim, o ciclo Scrum busca produzir um software mais específico e útil com base em um processo de produção e gerenciamento de tarefas seriadas e sequen-

ciais do produto (software) que tem como um de seus mecanismos de avaliação os testes realizados pelo cliente.

No mesmo sentido, o Extreme Programming (XP), criado por Kent Beck e Ward Cunningha, em 1996, orienta-se para práticas semelhantes ao Scrum, utilizando-se de ciclos contínuos de trabalho e interação com o cliente para gerar um produto que possua a maior utilidade possível ao cliente. O ciclo XP é dividido em 2 fases cada qual com pelo menos 4 momentos. A primeira fase é denominada de práticas organizacionais e é composta pelas definições de funcionalidade do software, planejamento das equipes e metas para o desenvolvimento de cada grupo, entrega constante de pequenas versões para o cliente e o teste de aceitação final do cliente. Concomitantemente a esta fase, ocorre a fase 2 que são práticas a serem realizadas apenas pela equipe de trabalho, visando aprimorar o processo de desenvolvimento como, padronização do ritmo de trabalho, padronização dos códigos, edição do programa conforme especificação do cliente, testes e integração contínua das partes do software produzidas separadamente.

Já o Sistema Kanban[21], se utilizando da mesma lógica das indústrias de automóveis, consiste na existência de cartões de comunicação rápida e na compartimentação dos objetivos em pequenas tarefas e metas. Em cada reunião de planejamento é gerado um cartão tema que contenha o objetivo daquela interação. Em seguida são gerados cartões secundários que indicam o que precisa ser feito na sequência do processo de trabalho. Dessa forma, é possível controlar o ritmo e intensidade da produção, evitando desperdício e aumentando a intensidade do trabalho.[22] Usada originalmente como uma prática de controle de estoque, o Kanban permite à indústria de software produzir apenas o necessário e quando necessário. Nesse

21 Para mais esclarecimentos sobre o Kanban ver: Antunes (2010); Augusto Pinto (2010); Coriat (1994) e Ohno (1978)

22 Como sublinha Gounet (2002, p. 29) e Antunes (2010, p. 34), as formas de organização da produção como o Kanban do Método Toyota resultam diretamente em uma intensificação do processo de trabalho, através de um controle rígido sobre a performance do trabalhador. Segundo Antunes, "*Gounet nos mostra ainda que o sistema toyotista supõe uma intensificação da exploração do trabalho, quer pelo fato de que os operários atuam simultaneamente com várias máquinas diversificadas, quer através do sistema de luzes (verde = funcionamento normal; laranja = intensidade máxima, e vermelha = há problemas, deve-se reter a produção) que possibilitam ao capital intensificar — sem estrangular — o ritmo produtivo do trabalho. As luzes devem alternar sempre entre o verde e o laranja, de modo a atingir um ritmo intenso de trabalho e produção.*" (Antunes, 2010, p. 34).

sentido, cada grupo de trabalho deve orientar seu trabalho pelas necessidades dos outros grupos e pela demanda.

Assim, no desenvolvimento de software, o Kanban é utilizado como uma técnica de flexibilização do processo de desenvolvimento de software, ajustando-o às necessidades do cliente. Se na construção de automóveis o Kanban é utilizado como uma ferramenta visual para orientar o quando e quanto produzir em cada grupo de trabalho, no desenvolvimento de software ágil, o Kanban toma uma função diferente, sendo utilizado não como uma técnica de controle de estoque, mas como um ferramenta de gerenciamento do trabalho que organiza e controla os ciclos de produção e de entrega do software previstos pelo Scrum e pelo XP.[23] Dessa forma, a flexibilidade, como uma característica dos coletivos de programação e desenvolvimento de software, não significa em nada a diminuição sobre o controle sobre do trabalho. Na prática, as metodologias ágeis têm, portanto, como objetivo principal aprimorar os meios de controlar o trabalho e extrair mais mais-trabalho.

Sendo o Kanban um sistema de alta complexidade com fluxo de informação intensa e curto espaço de tempo para realização do trabalho, é necessária, por parte da gerência e da supervisão, uma leitura precisa e simples do todo e que garanta que todas as partes estejam desenvolvendo suas tarefas no *"tempo correto"* e com a *"qualidade esperada*"[24]. Nesse sentido, a flexibilidade, como forma de ajuste constante da produção à demanda na produção e desenvolvimento do software ágil, se apoia não apenas na flexibilização do trabalhador e na intensificação do trabalho, mas também em um processo de interiorização da gerência, sito é, de auto-taylorização. O sistema Kanban exige dos trabalhadores não apenas que observem a sua performance, garantindo que estejam dentro dos prazos e ritmo de trabalho desejados pela empresa, mas que vigiem o trabalho dos

23 Como observa um gerente de programação e inovação de uma grande empresa de produção de software: "A gente tem um sistema aqui para organizar esse trabalho disperso (home-office) dos funcionários. Hoje em dia é tudo online e compartilhado assim podemos saber em que pé anda cada, cada grupo e cada funcionário para saber se precisam corrigir algumas coisas, se vai entregar dentro do prazo. Não pense que porque existe essa flexibilização não tem cobrança, pelo contrário, nós exigimos o comprometimento de cada um e medimos a possibilidade de flexibilidade com base nisso também (Entrevista realizada em julho de 2016).

24 O que nos remete a algumas indicações de Taylor (1970), sobretudo, em relação às práticas relativas ao *"the Best oneway"* e ao *"ótimo dia de trabalho"*.

outros trabalhadores, haja vista a necessidade de que todas as partes da produção estejam sincronizadas e integradas.

Apesar de apresentar modificações conjunturais, adaptações à produção de software, as práticas gerenciais presentes no setor apresentam-se como uma radicalização dos pressupostos taylor-fordistas e toyotistas para o aumento de produtividade e, consequente, ampliação do capital. A busca por controle e vigilância da produção proposta por Taylor (1970) e aprimorada pelo fordismo encontra no setor de software um cenário fértil. Práticas como da "história do usuário", integração continuada, PDCA, caso de uso, desenvolvimento guiado por teste e *Time Box* representam algumas das formas de controle e organização do trabalho presentes no setor de software que tem como referência as práticas da gerência de Taylor, Ford e Ohno.

A noção de história do usuário, por exemplo, é uma prática que busca identificar os principais requisitos que o software deve ter. De fato, a noção de história do usuário nada mais é do que as demandas do cliente em relação ao software desejado. Como nos informou um desenvolvedor:

> (...) essa história de *User Stories* é assim: nos pedimos para o cliente resumir em algumas frases o que ele gostaria que o software fosse e como funcionasse. Depois transformamos isso em pequenos histórias, tipo pequenas listas de afazeres, e organizamos quem vai fazer o que e em que ordem (Desenvolvedor de software. Entrevista realizada em novembro de 2018).

O PDCA (*PLAN - DO - CHECK - ACT* ou Adjust*)* é uma técnica baseada na repetição dos processos de desenvolvimento, buscando a cada ciclo de desenvolvimento aprimoramentos possíveis.

> O PDCA é simples. Funciona assim: nós estabelecemos na empresa um ritmo e um tipo de processo de desenvolvimento. São coisas que vão desde estipular hierarquias, metas até a própria codificação do software. Depois disso, nós fazemos vários ciclos de desenvolvimento e a cada ciclo nós fazemos um balanço interno do que funcio-

> nou e do que está ruim e aprimoramos. Para as empresas, isso não tem fim, é um processo continuado, não importa se o resultado foi excelente. Deve ter um jeito de deixar isso mais barato, mais rápido e por ai vai (Desenvolvedor de software. Entrevista realizada em novembro de 2018).

Os *Use Cases* são mecanismos próximos ao Kanban, sendo uma técnica de organização da informação produzida durante o desenvolvimento de um software. Contudo, nos *use cases* a informação é armazenada na forma de uma narrativa que permite o controle detalhado sobre como está cada ciclo de desenvolvimento. Segundo relato de outro desenvolvedor:

> Na última empresa em que eu trabalhei, eles usavam muito isso de *use case*, que no fundo nada mais é do que um mecanismo para gerenciar informação. Os desenvolvedores envolvidos nos projetos, os gerentes e o cliente precisam ter acesso ao que está sendo feito em cada equipe de desenvolvimento. No caso do *use case* é feito quase que uma história do que aconteceu com aquele programa. Eles chamam de narrativa e é bem parecido porque não são só detalhes técnicos, fica armazenado uma mini história do percurso que aquele programa fez, de ciclo para ciclo. Então, a qualquer momento você pode pegar e vai entender o que está acontecendo, quais as dificuldades, se vai sair no prazo ou não (Desenvolvedor de software. Entrevista realizada em novembro de 2018).

Já o *Time Box* é uma prática de gerenciamento do tempo. Após a definição dos requisitos no sistema, é construída uma time Box (caixa do tempo) no qual é determinado o tempo que cada equipe vai ter para a produção de cada ciclo do software preliminar e teste do cliente. Segundo um desenvolvedor:

> O Time Box nada mais é do que definição de prazos, os cara vão lá definem o que tem que fazer e quanto tempo a gente tem e chamam de Time Box. Na verdade, é só um nome que serve para indicar

> quanto tempo cada um tem para realizar as metas estabelecidas para aquele ciclo de desenvolvimento (Desenvolvedor de software. Entrevista realizada em novembro de 2018).

Vemos, assim, que as indústrias de software e os ambientes virtuais permitem monitoramento, regramento e parcialização intensos do trabalhador coletivo. O relógio de ponto, o vigia, o ambiente panóptico e tantos outros elementos dos processos de trabalho são absorvidos pelos softwares que controlam e regulam a produção,[25] na mesma medida em que a máquina, na passagem da manufatura à maquinaria, absorveu o saber-fazer diluído na experiência laboral dos artesãos das manufaturas dos séculos XVI, XVII e XVIII, absorção esta radicalizada pelas práticas gerenciais tayloristas e pela cooptação da subjetiva operária implementada pelo fordismo.

Do ponto de vista da empresa, as metodologias ágeis otimizam a produção na medida em que ampliam seu domínio sobre o trabalho ao reconfigurarem a capacidade de intervenção do trabalhador coletivo nos processos de trabalho e produção. Contudo, do ponto de vista do trabalho, elas são formas de ampliação e radicalização do controle sobre o trabalho, na medida em que além de reproduziriam os padrões de produção seriados do taylor-fordismo, se fundamentam numa prática organizacional que pressupõem um necessário engajamento do trabalhador à ideologia gerencial.

Portanto, observa-se um duplo movimento de subordinação do trabalho em relação ao capital, primeiro, aos preceitos, regras, funções e estratégias impostas pela máquina-software de produção e vigilância e, segundo, por uma prática de convencimento subjetivo que acaba por criar no trabalhador uma gerência internalizada, o que chamamos aqui de auto-taylorização do trabalho.

A organização do trabalho se torna, com isso, ainda mais radical no exato sentido em que a gerência, como um processo de racionalização objetivo das atividades laborais, é interiorizada em um processo de auto-taylorização que pressiona o trabalhador a saber mais, a conhecer mais, a buscar mais, mesmo que este "a mais" não

25 Sobre o tema, ver, por exemplo, Cingolani (2016).

ofereça a ele nenhuma autonomia no trabalho ou melhora das suas condições de trabalho e de vida.

4 CONCLUSÃO

Há duas questões que nos parecem centrais e que nosso artigo procura demonstrar. A primeira se refere a como as industrias imateriais e o trabalho imaterial são expressões renovadas de uma mesma forma organizacional que se adapta às fronteiras da produção da informação, do conhecimento, da comunicação. Este processo pode ser observado pela retomada histórica de como o trabalho foi se subordinando ao capital e pela sua comparação com as formas de organização do trabalho no setor de software, baseadas nas metodologias ágeis, como, por exemplo, o *Scrum* e o *Extreme Programming*. Com base nesta comparação, constatamos algumas semelhanças entre os processos de subordinação do trabalho em relação ao capital nas indústrias tradicionais e nas indústrias de software.

A trajetória de perda do controle sobre o processo de trabalho que foi se aprofundando desde a manufatura, com sua produção ainda artesanal, passando pela maquinaria na qual ainda se observa a presença do trabalhador como articulador do processo de trabalho, aprofundando-se nas práticas de racionalização do trabalho com o taylorismo e na produção em série fordista, pode ser observada também na trajetória histórica de constituição da produção de software.

Em seu início, o desenvolvimento da indústria de software caracterizou como uma atividade quase artesanal, assim como a manufatura, na medida em que não se estruturava com base em uma rotina burocratizada e submetida ao ritmo da linha seriada produção. Isto possibilitava certo controle dos desenvolvedores e programadores sobre o processo de trabalho, sobretudo, no *como fazer*, haja vista seu domínio sobre o saber-fazer técnico e científico. Ou seja, o desenvolvimento artesanal de software apesar de já submetido a uma subordinação externa ao capital, não possuía ainda uma subordinação interna que retirasse dos trabalhadores o controle sobre o trabalho de desenvolvimento e programação de softwares.

Não obstante, para atender às demandas do capital, as indústrias de software passaram a racionalizar os processos de trabalho à maneira do taylo-fordismo, baseadas nas linhas virtuais de trabalho, no fluxo vertical da produção, no trabalho especializado e altamente substituível e no controle sobre o ritmo de trabalho. Nesse sentido, por um lado, os métodos estruturados somados aos programas de controle de qualidade acabam por submeter o desenvolvedor de software a condições análogas ao do operariado taylor-fordista e, por outro lado, assumem diretamente o controle sobre o processo de desenvolvimento, acabando com os elementos que lhes rendiam certa autonomia produtiva.

As práticas toyotistas vem contemporaneamente radicalizar esta organização do trabalho. No entanto, elas não superam os preceitos estruturais racionalizantes da produção seriada. Na prática, constitui-se uma dupla subordinação. Objetiva, na medida em que os saberes, técnicas e conhecimentos são internalizados por linhas virtuais de produção e controle em um processo típico de taylorização do trabalho, e subjetiva, no sentido em que se confere ao trabalhador coletivo e individual a aparência de autonomia, flexibilidade, participação e criação no interior destes processos de trabalho e produção. Entretanto, essa aparência necessária acaba por forçá-los, os coletivos de trabalho, a se auto-taylorizar com base na busca de novos conhecimentos, novas técnicas, novos formas de interação com os clientes, com os outros trabalhadores e com seus superiores.

Portanto, nos cabe, por fim, questionar o que a produção de software e a indústria de software trazem realmente de novo? Em seu discurso, as metodologias ágeis do desenvolvimento de software se apresentam como algo novo, uma nova forma de trabalhar em um novo mundo. Contudo, existe um distanciamento entre a aparência inovadora e a prática real. Quando analisamos o funcionamento das metodologias ágeis, percebemos que tais formas de organização do trabalho apenas adaptam a racionalização taylorista, a produção em massa parcializada e mecanizada do fordismo e a flexibilização do trabalho e do trabalhador presentes no toyotismo à lógica da produção de software.

O novo presente na produção de software se configura como uma novidade conhecida, uma adaptação das formas de produzir a

um novo patamar tecnológico e social. Portanto, a indústria de software, na prática conserva as relações de controle e exploração do trabalho presentes na indústria tradicional ainda que estejam baseadas em na forma de esteiras e gerências digitais e virtuais.

REFERÊNCIAS

AMORIM, H. El "fin de las clases sociales" em la teoría social brasileña. *Revista Estudios Latinoamericanos*, nº. 35, pp. 15-37, 2015.

_____. As teorias do trabalho imaterial: uma reflexão crítica a partir de Marx. *Caderno CRH* (UFBA. Impresso), Vol. 27, nº. 70, pp.31-45, 2014.

_____. O tempo de trabalho: uma chave analítica. *Sociedade e Estado*, UnB, Vol. 28, nº. 03, pp. 503-518, 2013.

_____. *Trabalho imaterial*: Marx e o debate contemporâneo. São Paulo: Annablume, 2009.

ANTUNES, R. *Adeus ao trabalho*? Ensaio sobre as metamorfoses e a centralidade no mundo do trabalho. São Paulo: Cortez, 2010.

BRAVERMAN, H. *Trabalho e capital monopolista*: a degradação do trabalho no século XX. Rio de Janeiro: Zahar, 1980.

BRIDI, M. A., MOTIM, B. L. Trabalho e trabalhadores na indústria de informática. *Contemporânea* – Revista de Sociologia da UFSCar. São Carlos, Vol. 4, nº. 2, jul-dez, pp. 351-380, 2014.

CARDOSO, A. C. M. *Tempos de trabalho, tempos de não trabalho*. Disputas em torno da jornada do trabalhador. São Paulo: Annablume/Fapesp, 2009.

CARVALHO, B. V. e MELLO, C. H. P. Aplicação do método ágil scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica. *Gestão & Produção*, Vol. 19, nº. 3, pp. 557-573, 2012.

CASTELLS, M. *A sociedade em rede*. São Paulo: Paz e Terra, 1999.

_____. *A galáxia internet*: Reflexões sobre internet, negócios e sociedade. Lisboa: Fundação Calouste Gulberkian, 2004.

CASTRO, B. *As armadilhas da flexibilidade*: trabalho e gênero no setor de tecnologia da informação. São Paulo: Annablume, 2016.

CINGOLANI, P. Capitalismo de plataforma: nuevas tecnologías de la comunicación e internacionalización del trabajo. *Boletín Onteaiken*, n°. 22, 2016. http://onteaiken.com.ar/ver/boletin22/onteaiken-22_Cingolani.pdf

CMMI® for SCAMPI. Class a Appraisal Results – 2011. In: http://www.sei.

cmu.edu/cmmi/casestudies/profiles/pdfs/upload/2011SeptCMMI-2.pdf Acesso em: 16 de janeiro de 2016.

COLBARI, A. L. A retórica do empreendedorismo e a formação para o trabalho na sociedade brasileira. *Sinais* - Revista Eletrônica - Ciências Sociais, Vol. 1, nº. 1, pp.75-111, 2007.

CUSUMANO, M. A. *Japan's software factories*. Oxford University Press. 1991.

_____. The 'factory' approach to large-scale software development: implications for strategy, technology, and structure. *MIT Sloan School of Management Working Paper* #1885-87, September 1987.

_____. *The software factory*: a historical interpretation. New York: Oxford University Press, 1989.

DARDOT, P. & LAVAL, C. *A nova razão do mundo*: ensaio sobre a sociedade neoliberal. São Paulo: Boitempo, 2016.

DIAS, E. *A liberdade (im)possível na ordem do capital*: reestruturação produtiva e passivização. Campinas: IFCH/Unicamp, 1997.

DIEGUES JUNIOR, A. C. Atividades de Software no Brasil: Dinâmica Concorrencial, Política Industrial e Desenvolvimento. 2010. *Tese* (Doutorado em Economia) Instituto de Economia, Universidade Estadual de Campinas, Campinas, 284 p.

DUARTE, V. Escassez de mão de obra em TI: uma abordagem qualitativa. In: *Mercado de trabalho e formação de mão de obra em TI*. SOFTEX, 2013.

DUTRA, R. Q. *Do outro lado da linha*: Poder Judiciário, regulação e adoecimento dos trabalhadores em call centers. São Paulo: LTR, 2014.

FADEL, A. C. SILVEIRA, H. M. Metodologias ágeis no contexto de desenvolvimiento de software: XP, Scrum e Lean. In: https://www.ft.unicamp.br/liag/Gerenciamento/monografias/Lean%20Agil_v8.pdf Acesso em: 17 de março de 2017.

FERNANDES, A. A. & TEIXEIRA, D. D. S. *Fábrica de software*: implantação e gestão de operações. São Paulo: Atlas, 2004.

GIANINAZZI, W. *André Gorz*: une vie. Paris: La Decouverte, 2016**.**

GORZ, A. *Estratégia operária e neocapitalismo*. Rio de Janeiro: Zahar, 1968.

_____. *Socialismo difícil*. Rio de Janeiro: Zahar, 1968a.

_____. *Adeus ao proletariado*. Para além do socialismo. Rio de Janeiro: Forense, 1987 [1981].

_____. *L'immatériel*. Paris: Galilée, 2003. (Edição Brasileira O imaterial: conhecimento, valor e capital. São Paulo: Annablume, 2005).

GOUDNER, A. *El futuro de los intelectuales y el ascenso de la nueva classe*. Madrid: Alianza Editorial, 1979.

GOUNET, T. *Fordismo e toyotismo na civilização do automóvel*. Coleção Mundo do Trabalho. Rio de Janeiro: Boitempo, 2002.

GRAMSCI, A. *Cadernos do cárcere*. Rio de Janeiro: Civilização Brasileira, 2004.

GRAZIA, M. R. O tempo de trabalho no contexto das novas tecnologias da informação e comunicação: uma análise sobre o setor de software. 2018. *Dissertação* (Mestrado em Ciências Sociais), Universidade Federal de São Paulo - Escola de Filosofia, Letras e Ciências Humanas - Programa de Pós-Graduação em Ciências Sociais, Guarulhos, 124 p.

HABERMAS, J. *Teoría de la acción comunicativa*. Madrid: Taurus, 1987.

HARVEY, D. *A condição pós-moderna*. São Paulo: Loyola, 2010.

HUMPRHEY, W. *Managing the software process*. Boston, MA: Addison-Wesley Longman, 1990.

INGLEHART, R. *The silent revolution*. Princeton, NJ: Princeton University Press, 1977.

LAZZARATO, M. Le cycle de la production immatériel. *Futur Antérieur*, [S.l.] nº. 16, pp. 111-120, 1993.

LEAN INSTITUTE BRASIL. In: http://www.Lean.org.br/ Acesso em: 20 jun 2016.

MAGALINE, A. D. *Luta de classes e desvalorização do capital*. Lisboa: Moraes, 1977.

MALAGUTI, M. L. A ideologia do modelo japonês de gestão. *Ensaios FEE*. Vol. 17, nº. 1, pp. 43-73, 1996.

MALLET, S. *La nouvelle classe ouvrière*. Paris: Éditions du Seuil, 1969.

_____. *Le pouvoir ouvrier*: bureaucratie ou democratie ouvrière. Paris: Anthropos, 1971.

MANIFESTO ÁGIL. *In*: http://agilemanifesto.org/iso/ptbr/manifesto.html Acesso em: 20 nov. 2018

MARX, K. *Manuscritos econômico-filosóficos*. São Paulo: Boitempo, 2004.

_____. *O Capital*. São Paulo: Boitempo, 2013.

MATSUMOTO, Y. A software factory: an overall approach to software production. *In*: FREEMAN, P. (Ed.). *Software reusability*. Washington: IEEE Computer Society Press, 1987.

MATSUMOTO, Y. SWB system: a software factory. *In*: HUNKE, H. (Ed.). *Software-Engineering Environments*. Amsterdam: North-Holland, 1981.

MELUCCI, A. The new social movements: a theoretical approach. *Social Science Information,* Vol. 19, nº 2, pp.199-226, 1980.

MOULIER-BOUTANG, Y. *Le capitalisme cognitif*: la nouvelle grande transformation. Paris: Éditons Amsterdam, 2007.

NEGRI, A. Valeur-travail: crise e problèmes de reconstruction dans le post-moderne. *Futur Antérieur*, n.º 10, pp. 20-36, 1992.

OFFE, C. Trabalho: a categoria-chave da sociologia? *Revista Brasileira de Ciências Sociais*, Vol. 4, nº. 10, pp. 6-20, 1989.

RICHTA, R. *Economia socialista e revolução tecnológica*. Rio de Janeiro: Paz & Terra, 1972.

ROSELINO, J. E.. A Indústria de software: o "modelo brasileiro" em perspectiva comparada. 2006. *Tese* (Doutorado em Economia) Instituto de Economia, Universidade Estadual de Campinas, Campinas, 216 p.

ROSENFIELD, C. L. & ALVES, D. A.. Autonomia e trabalho informacional: o teletrabalho. *DADOS – Revista de Ciências Sociais*, Vol. 54, nº. 1, pp. 207-233, 2011.

ROYCE, W. W. Managing the development of large software system. In: *Proceedings of IEEE Wescon*, nº26, pp. 382-338, 1970.

TAKEUCHI, H. & NONAKA, I. The new product development game. *Harvard Business Review*, Vol. 64, nº. 1, pp. 137-146, 1986.

TOURAINE, A. *Sociedade pós-industrial*. Lisboa: Moraes Editores, 1970.

XAVIER, C. D. Fábrica de software: até que ponto fordista? 2008. *Dissertação* (Mestrado em Gestão Empresarial), Escola Brasileira de Administração Pública e de Empresas da Fundação Getúlio Vargas, Rio de Janeiro, 94 p.