Engenharia de software e processos de produção: questões sobre quem produz e como se produz

Gilvan de Oliveira Vilarim[1], Alexandre do Nascimento[2]

[1] Instituto Federal de Educação, Ciência e Tecnologia do Rio de Janeiro (IFRJ)
Rua José Breves, 550 – 27197-000 – Pinheiral – RJ – Brasil

[2] Fundação de Apoio à Escola Técnica do Estado do Rio de Janeiro (FAETEC)
Rua Clarimundo de Melo, 847 – 21311-281 – Quintino – Rio de Janeiro – RJ

gilvan.vilarim@ifrj.edu.br, alex.nasc@uol.com.br

Abstract. *This article seeks to relativize the discussion about labor processes arising from software engineering, with comparisons between the forms of industrial production and the area of software. We note the limits of this approach in relation to the moment of contemporary capitalism, strongly based on collaborative production and network cooperation.*

Resumo. *Este artigo procura relativizar a discussão sobre os processos de trabalho advindos da engenharia de software, apresentando comparações entre as formas de trabalho da produção industrial e a área de software. Observamos os limites desse tipo de abordagem em relação ao momento do capitalismo atual, fortemente baseado em produção colaborativa e cooperação em rede.*

1. Introdução

No processo evolutivo da construção de software, Rezende (2008) observa que a arte de programar foi criando um rico ecossistema, com arquitetura e métodos de codificação estruturada, sobrepostos em camadas modulares. Estas camadas permitem um grau de interação entre os programas que abstrai detalhes de *hardware*; assim, *"... a produção de software pôde se desacoplar da indústria de hardware e explorar seus próprios modelos produtivos e negociais*" (*op.cit.*, p. 95).

Nosso destaque é que, ao longo das últimas décadas, forjou-se o que podemos denominar de disciplinamento para a produção de software, amparado em diversas técnicas, métricas e metodologias. Como é sabido, isso foi a base de uma nova área na Computação, a Engenharia de Software (ES); cabe lembrar que a ES, classicamente, é:

> (1) Aplicação de uma abordagem sistemática, **disciplinada** e quantificável para o desenvolvimento, operação e manutenção do software; isto é, a aplicação de engenharia ao software. (2) O estudo de abordagens como as de (1). (PRESSMAN, 2002, p. 18; grifo nosso)

É fato que a ES possui seu lado benéfico: como ela tem permitido a construção de softwares mais complexos e menos custosos, isto tem contribuído para a disseminação de diversas práticas e técnicas pertencentes ao seu rol de ferramentas. Nosso interesse, contudo, é relativizar esta discussão à luz do paradigma de produção

pós-industrial que transcende a computação[1]. Para as organizações, disciplinar um trabalho manual e já fragmentado por décadas de fordismo já era comum. Mas no caso do software, a sua produção representava, no início, um tipo de trabalho que corria à margem do espaço de gerenciamento corporativo, posto que não havia métricas nem formas de mensuração do que estava sendo criado.

2. A "fordização" da produção

Julgamos que, em sintonia mútua, o meio científico e o corporativo moldaram, em diferentes camadas de abstração, formas de disciplinamento do processo de construção dos softwares – é preciso observar que grandes empresas, em consonância com instituições científicas, detinham e detêm forte influência na consolidação de determinadas metodologias e padrões. Para Hard e Negri (2005), o controle do conhecimento científico é parte de um processo de luta biopolítica. O conhecimento científico se torna identificado com a produção, subsumido também a regras baseadas, em última instância, no lucro. Esta, portanto, não é uma discussão "pura" sobre os desdobramentos da ES sob o ponto de vista técnico, mas sim sobre uma dimensão biopolítica.

"Bons" softwares passaram a ser aqueles que utilizavam tais propostas de gerenciamento na sua construção. Mas, se por um lado tais camadas tinham uma intenção de facilitar a troca de conhecimentos entre os produtores, por outro elas foram uma tentativa de aumentar gradativamente um exercício de poder tal como no modelo fabril, alegando-se que o desenvolvimento estava se tornando extremamente custoso e não-gerenciável – incontrolável. A "crise do software" foi o horizonte sombrio apresentado para os anos seguintes (DIJKSTRA, 1972), mesmo que, naquele momento, houvesse divergências sobre a sua real existência (NAURR e RANDELL, 1968).

Paulatinamente, o programador funcional de antigamente foi sendo substituído por uma equipe de especialistas de software; cada um (PRESSMAN, 2002) se concentra em uma parte da tecnologia para produzir uma aplicação de software complexa. Para nós, isso trouxe reflexos no trabalho e nos processos de gestão; é um exercício de poder disciplinar que busca dessubjetivar o processo de criação de software em prol da racionalidade. Söderberg (2008) comenta que a codificação de software, nos últimos tempos, vem se tornando rotinizada justamente para que o controle seja passado dos programadores para os gerentes.

Podemos, historicamente, identificar dois momentos de tentativas de "fordização" (em alusão aos modelos de produção) ocorridas nesta criação. O primeiro tem como ponto de vista a organização do trabalho, quando o padrão da administração científica foi disseminado na produção de software. O processo de desenvolvimento foi fragmentado em tarefas e funções; uma verticalização funcional cristalizou a existência de analistas, projetistas e programadores (por vezes subdivididos) que, se em última instância ainda participavam da geração do produto, não necessariamente conseguiam agora visualizar a produção como um todo. É o momento de uma racionalização (GORZ, 2003) que precisava quantificar o trabalho de um novo tipo como uma grandeza material que descartasse a individualidade e as motivações do trabalhador.

[1] Esta discussão é parte da tese de doutorado "Trabalho, suas transformações e a questão da produção de software no capitalismo contemporâneo", defendida pelo primeiro autor na ESS-UFRJ em 2012.

A segunda tentativa de fordização foi o processo de componentização do próprio software como um construto, tal como ocorreu com artefatos industriais. Surgiram metodologias e técnicas (ou até mesmo uma capacidade de abstração diferente no pensamento) capazes de permitir a criação de partes de software preferencialmente reutilizáveis, padronizadas, independentes entre si, cujo maior exemplo atual é a Programação Orientada a Objetos (POO). Como o operário fordista, que inclui uma peça em um produto sem precisar saber como ela funciona, o programador passa a executar, em vários momentos, uma "bricolagem" de objetos (sem usar as mãos, mas sim o cérebro), perdendo a compreensão do que está sendo produzido como um todo.

Desta forma, embora deslocado temporalmente em relação ao que já estava ocorrendo há anos nos ambientes fabris com produtos materiais, o software passou a ser objeto de forte pressão de um paradigma de produção taylorista-fordista, cristalizando-se a partir do final dos anos 1960. Em diversas organizações, o maior exemplo da aplicação deste modelo é a existência, literalmente, das fábricas de software.

3. Limites da abordagem industrial

Consideramos que, no capitalismo cognitivo atual, tal como identificado por Moulier Boutang (2007), o próprio desenvolvimento do trabalho em rede numa dinâmica horizontal, colaborativa e subjetiva, aponta para uma ineficácia deste processo de fordização. Não é possível considerar um software como uma mercadoria, uma vez que o mesmo é uma concretização de conhecimentos, imateriais, embutidos no seu desenvolvimento. Na realidade, a tentativa de codificação total do conhecimento do software nunca se torna plena, pois o trabalho "vivo" (MOULIER BOUTANG, 2007) que é inerente às atividades do programador, nunca se deixou apropriar como um todo. Ao contrário, o trabalho vivo tem aumentado em função da horizontalidade trazida pelas redes e pela possibilidade dos desenvolvedores produzirem com meios que fogem da dialética capital-trabalho.

Por um lado, é estimulada essa aplicação do aparato da engenharia (industrial) de software no processo de produção, ainda que sejam discutidos modelos de desenvolvimento de caráter evolutivo (capazes de permitir a maior interação produtor-produtor e produtor-usuário). Por outro, e daí há um paradoxo, a criação de software nunca foi plenamente subordinada à relação capital-trabalho, pois criatividade implica em liberdade e ao mesmo tempo cooperação, que não cabem nos métodos industriais.

As facilidades trazidas pela colaboração em redes de comunicação só aumentaram o processo de fricção social entre seus atores, que contribuem com seus "cérebros e corpos" para a elaboração de produtos mais criativos e mais distantes de uma mercadoria, no sentido clássico. O que apresentamos anteriormente como tentativa de fordização, a componentização, acaba também por ter efeito inverso em rede, favorecendo a entrada de mais participantes no processo de desenvolvimento.

Um software muito granulado pode estimular a participação de muitas pessoas que desejam fazer pequenas contribuições (BENKLER, 2002). Componentes de tamanhos variados podem, no caso do software, representar lógicas mais ou menos complexas que captarão, por sua vez, pessoas mais motivadas e desafiadas à busca de soluções também mais ou menos complexas. E a integração na produção em rede pode ser estimulada em função da interação, por meio de plataformas de colaboração (ex:

wikis), normas de organização social e um nível bastante limitado de hierarquia (posto que, em excesso, a produção em rede não funciona).

De certo modo, as organizações já reconhecem a mudança de um paradigma fordista e a engenharia de software tenta – timidamente – captar esta mudança. Pressman (2002) lembra que: "*Apesar da indústria estar se movendo em direção a montagem baseada em componentes, a maior parte do software continua a ser construída sob encomenda*" (PRESSMAN, *op.cit.*, p. 7-8). Construído "sob encomenda", ou seja, customizado para a realidade de uma pessoa/organização/cliente, o software não pode ser encarado como um produto de manufatura clássico, aproximando-se mais de um serviço que é prestado (o "software como um serviço").

Considerações finais

Terranova (2004) observa que a administração tem se mostrado preocupada com a gestão de um trabalho mais baseado em conhecimento, onde a inteligência humana (coletiva) provê o valor; contudo, essa produção não pode ser gerenciada do mesmo jeito que os tipos mais tradicionais de trabalho, dado que ela necessita, no momento atual, de estruturas abertas para se gerar e deixar fluir o conhecimento.

Este é o conflito entre a chamada "economia da dádiva" e os mecanismos de gestão que ao mesmo tempo estimulam e tentam apropriar a produção de conhecimento. Os processos de trabalho em software precisam se adaptar aos novos tempos e contemplar essas novas formas de produção, embutindo subjetividades inerentes à criação dos softwares, em seus mais diversos escopos.

Referências

BENKLER, Y. (2002) Coase's penguin, or, Linux and the nature of the firm. *The Yale Law Journal*. v.112, n.3, dez. Disponível em: <http://www.benkler.org/CoasesPenguin.html>. Acesso em: 29 mar.2013.

DIJKSTRA, E.W. (1972) The humble programmer. *Communications of the ACM*, n.15. p.859–866.

GORZ, A. (2003) *Metamorfoses do trabalho*. São Paulo: Annablume.

HARDT, M.; NEGRI, A. (2005) *Multidão*. Rio de Janeiro: Record.

MOULIER BOUTANG, Y. (2007) *Le capitalism cognitif*. Paris: Éditions Amsterdam.

NAUR, P.; RANDELL, B. (1968) *Software Engineering: report on a conference sponsored by the NATO Science Committe*. Garmisch, Alemanha, out. Disponível em: <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>. Acesso em: 29 mar.2013.

PRESSMAN, R. (2002) *Engenharia de software*. Rio de Janeiro: McGraw-Hill, 5.ed.

REZENDE, P. A. D. (2008) Custo social: propriedade imaterial, software, cultura e natureza. In: PRETTO, N. L.; SILVEIRA, S. A. *Além das redes de colaboração: internet, diversidade cultural e tecnologias do poder*. Salvador: EDUFBA. p. 93-110.

SÖDERBERG, J. (2008) *Hacking capitalism: the free and open source software movement*. New York: Routledge.

TERRANOVA, T. (2004) *Network culture*. London: Pluto Press.