Os limites à taylorização do trabalho na fase de concepção da produção de *software*

Cesar Ricardo Siqueira Bolano[1] e José Guilherme da C. C. Filho[2]

Sessões de Comunicações
Economia Política, Capitalismo e Socialismo
Capitalismo Contemporâneo

Resumo
Há um aparente paradoxo no atual processo de subsunção do trabalho intelectual, ligado ao fato de que o trabalho de concepção, etapa inicial que garante o que podemos chamar de produção de *software* por meio de *software*, elemento estratégico dos sistemas de inovação que estão no centro das cadeias produtivas que incorporam tecnologias de base micro-eletrônica, continua sendo de tipo essencialmente artesanal ou manufatureiro. O mesmo não ocorre nas demais fases da produção e uso de *softwares* nos processos produtivos, onde se observa, seja, no primeiro caso, uma importante taylorização, seja, no segundo, a mais ampla automatização, garantida, esta, justamente, pelo trabalho criativo, apenas formalmente subsumido, da concepção. Essa atividade, fortemente dependente do trabalho vivo e atrelada às linguagens de programação e metodologias de desenvolvimento de *software*, é requerida para a produção do programa de computador que assume o papel de ferramenta de concepção e desenvolvimento de um produto (que é também um *software*), cuja utilidade é garantir formas mais avançadas de subsunção do trabalho nas outras fases do processo produtivo.

Palavras-chave: trabalho intelectual, informática, economia política

Abstract
There is an apparent paradox in the current process of subsumption of labor Intellectual, connected to the fact that the design work, the initial step that for what we call software production by software, strategic element of innovation systems that are the backbone of the chains productive technologies that incorporate the basic micro-electronics, remains essentially craft or type of manufacturing. This does not occur in other phases of the production and use of software production processes, where notes, is the first case, an important Taylorism and second, broader automation, guaranteed, this precisely by creative work, only formal subordinate of the design. This activity, strongly dependent on the work alive and tied to programming languages and methodologies for software development, is required for the production of computer program that assumes the role of tool design and development of a product (which is also a software), whose usefulness is ensure more advanced forms of subsumption of labor in other phases of production process.

Keywords: intellectual work, computer science, political economy

[1] Universidade Federal de Segipe (UFS), Brasil.
[2] Universidade Federal de Segipe (UFS), Brasil.

1. Introdução

Os atuais processos produtivos, ligados à dinâmica da inovação no capitalismo da Terceira Revolução Industrial, se caracterizam pela combinação do desenvolvimento de técnicas e conhecimentos específicos aos diferentes setores da economia, com o de procedimentos e linguagens informacionais cada vez mais sofisticados, os quais possuem um caráter geral, podendo ser utilizados em um conjunto muito grande de setores produtivos. A essência da automação de base microeletrônica é uma extensa subsunção do trabalho intelectual (Bolaño, 1995, 2002), o que envolve especialização crescente deste, mas também, ao mesmo tempo, uma tendência ao apagamento das suas fronteiras em relação ao trabalho manual, submetido, por sua vez, a uma intelectualização geral da produção e do próprio consumo (idem).

Tal processo exige o estabelecimento de linguagens e métodos computacionais que garantem a existência de um amplo sistema de codificação do conhecimento, que depende crucialmente do desenvolvimento e produção de *softwares* com a finalidade específica de produzir *softwares*. O presente trabalho tem como objetivo abordar a tendência de codificação do trabalho no processo de produção de *software*, mais especificamente em sua fase de concepção, a partir da análise do controle do trabalho nesse processo, aliado às ferramentas de suporte à criação. Essas ferramentas permitem a produção de *softwares* por meio de *softwares*, segundo uma combinação contraditória entre o elevado nível de controle automatizado (conhecimento codificado), orquestrado pelos modelos de referência, e a alta dependência do trabalho vivo (conhecimento tácito).

Ao analisar o processo de criação de *software* é possível observar uma similaridade com o taylorismo, na medida em que se trata de simplificar e controlar um trabalho humano que não pode ser eliminado (substituído por máquinas) como nos processos automatizados. As ferramentas de *software* corroboram nesse sentido, como instrumentos de trabalho padronizados, sendo o elemento crucial do processo, a gerência e controle sobre um trabalho vivo que, neste caso, é extremamente qualificado, o que não elimina a necessidade do trabalhador adequar-se aos automatismos impostos pela própria forma que vai adquirindo o trabalho coletivo de concepção e produção desse tipo de *softwares*.

No entanto, o processo de produção de *softwares* é peculiar, pois o controle se dá através de ferramentas que devem ser constantemente aperfeiçoadas, em interação

com o trabalhador (desenvolvedor ou programador). Assim, a desqualificação do trabalho ligada à passagem da manufatura à indústria (inclusive no sistema taylorista-fordista) encontra limites muito importantes nesse setor, conservando o trabalho, características artesanais fundamentais.

Nossa hipótese é que os métodos de trabalho de codificação usados na fase de concepção controlada a partir de ferramentas de suporte à criação de *software*, embora apresentando similaridades, não se adéquam aos conceitos de taylorismo e fordismo, na interpretação aqui adotada, extraída de Benedito Moraes Neto (1991; 1998; 2003), tratando-se de uma situação muito mais próxima àquela do período manufatureiro, em que a divisão de trabalho se estabelece sobre a base de métodos e ferramentas herdados diretamente do artesanato, caracterizando uma "acumulação primitiva do conhecimento" que será o ponto de partida para a posterior industrialização (Bolaño, 2000).

Há uma diferença crucial, no entanto, que é o fato desse processo de concepção, ainda hoje apenas formalmente subsumido no capital, haver adquirido uma fundamental centralidade nos processos produtivos organizados, organizados em rede e em cadeias, sobre a base da Terceira Revolução Industrial, em que a inovação adquire uma posição absolutamente estratégica. Nesse sentido, ganha interesse retomar o conceito de "oficinas de progresso tecnológico" (Figueroa, 1986) para explicar, de forma mais adequada, a idéia corrente de "fábricas de *software*".

2. A produção de *software*

O processo de desenvolvimento de *software* estabelece métodos aliados a um conjunto de atividades, parcialmente ordenadas, com a finalidade de obter o produto final, ou seja, um *software* para atender a uma necessidade específica do cliente. Tal processo é estudado na área de Engenharia de *Software*, que visa assegurar o cumprimento das fases de desenvolvimento do *software* com o intuito de atingir os objetivos do projeto, ou seja, o produto de *software* demandado pelo cliente. Na década de 1970, como conseqüência do desenvolvimento das ferramentas de projeto assistido por computador (*Computer Aided Design* – CAD), surgiram as primeiras ferramentas conhecidas como ferramentas de engenharia de *software* assistida por computador (*Computer Aided Software Engineering* – CASE), que abrangem toda ferramenta baseada em computadores que auxiliam atividades de engenharia de *software*, desde a

concepção, passando pela análise de requisitos e modelagem, até a programação e testes finais. [3]

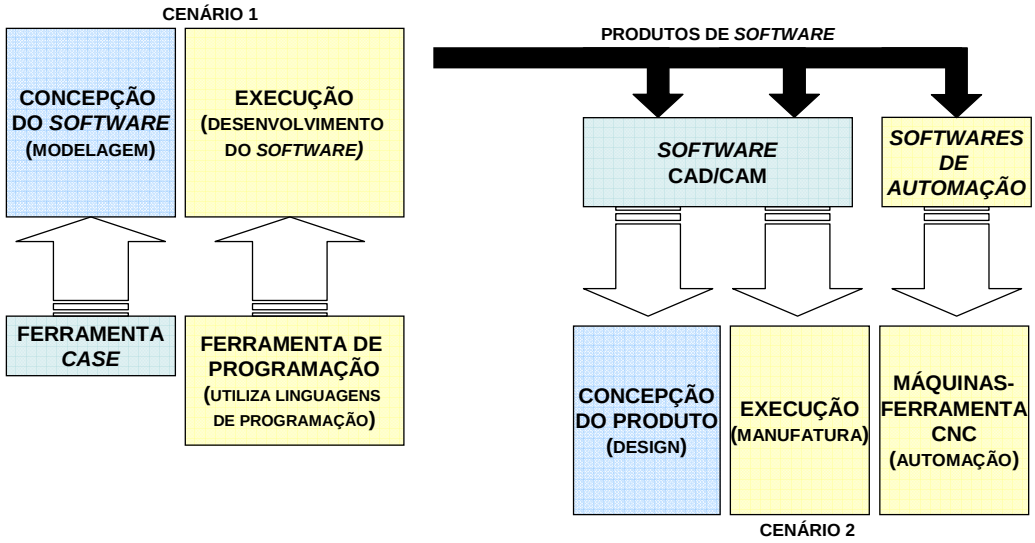
As ferramentas CASE, na prática, têm como objetivo automatizar atividades de pré-codificação ou concepção. Para isso, definem uma série de diagramas e regras, para moldar a forma como o indivíduo deve conceber o *software*. A engenharia de *software* é pensada de forma a capturar a concepção humana com o intuito de controlar o processo de produção. Pfleeger (2004, p. 10) sugere que

> assim como em outras áreas, em uma abordagem de engenharia de *software*, inicialmente o problema a ser tratado deve ser analisado e decomposto em partes menores, em uma abordagem "dividir para conquistar". Para cada uma dessas partes, uma solução deve ser elaborada. Solucionados os problemas isoladamente, é necessário integrar as soluções. Para tal, uma arquitetura deve ser estabelecida. Para apoiar a resolução de problemas, procedimentos (métodos, técnicas, roteiros etc.) devem ser utilizados, bem como ferramentas para parcialmente automatizar o trabalho. Neste cenário, muitas vezes não é possível conduzir o desenvolvimento de *software* de maneira individual. Pessoas têm de trabalhar em equipes, o esforço tem de ser planejado, coordenado e acompanhado, bem como a qualidade do que se está produzindo tem de ser sistematicamente avaliada.

É interessante verificarmos as etapas de produção de *software* a fim de compreendermos a dependência do trabalho vivo durante a etapa de concepção e a utilização das ferramentas CASE no intuito de minimizar tal dependência. A **figura 1** apresenta dois cenários. No primeiro, ocorre a idealização e criação do produto de *software* e, no segundo, a utilização efetiva do *software* criado no primeiro cenário, na produção industrial a partir de máquinas automatizadas por computador.

[3] CAD/CAM são sistemas de *software* que auxiliam a concepção (CAD, *Computer Aided Design*) e execução (CAM, *Computer Aided Manufacturing*) de peças piloto, as quais, uma vez construídas e testadas, serão produzidas continuamente, de forma automatizada, em máquinas de controle numérico computadorizado (CNC). As ferramentas CASE (*Computer Aided Software Engineering*) são uma evolução das ferramentas CAM especialmente desenvolvidas para auxiliar atividades produção de *softwares*.

Figura 2 – Processo de desenvolvimento de *softwares* CAD/CAM para utilização em máquinas automatizadas



Fonte: Elaboração própria, com base no processo de desenvolvimento de *software* descrito em SEI (2007) e no processo de produção industrial automatizado com uso de ferramentas de software CAD/CAM descrito por Castelltort (1988) e Feldens (2000).

O primeiro cenário é caracterizado pelo trabalho intelectual não automatizado e o segundo é parte fundamental da mediação e integração da automatização do trabalho industrial. No primeiro, artefatos de *software* compostos por ferramentas computacionais (CASE) baseadas em metodologias específicas (modelos de referência), permitem a elaboração (concepção) e produção (execução) de produtos de *software*. No segundo cenário, os produtos elaborados durante o processo de produção de *software* são usados como ferramentas que integrarão sistemas compostos por *hardware* (máquinas automatizadas) e *software*, que, por sua vez, permitem a concepção assistida por computador.

No **primeiro cenário**, operado geralmente em ambientes denominados **fábricas de *software***, o trabalho vivo é central, enquanto, no segundo, cede espaço para as máquinas automatizadas, controladas por *software*. A expressão "fábricas de *software*" é comum entre os informáticos. Tem sido usada desde os anos 1960, nos Estados Unidos e, nos anos 1970, no Japão, conforme Swanson et. alii.(1991). Numa perspectiva marxista, como a que embasa este trabalho, é possível aceitar esse senso comum, mas é preciso esclarecer que não se trata de um tipo de organização industrial próprio da grande indústria automatizada.

Vale, a esse respeito, recorrer a Victor Figueroa (1986). Falando sobre a passagem da subsunção formal à real, na transição da manufatura à grande indústria, lembra que "já no curso da manufatura, a oficina simples havia dado lugar à oficina mecanizada", quando o trabalhador opera a máquina-ferramenta que será responsável pela Revolução Industrial. Lembra, no entanto, que, na manufatura, "o uso da máquina não constitui um método predominante de produção", o que só ocorre na "grande indústria, no seio da fábrica" (Figueroa, 1986, p. 26). Em todo caso, esclarece:

> Marx, para designar esta situação [fábrica] também usou o nome de oficina automatizada, ou ainda oficina mecanizada, mas como aqui já não se trata no principal de um mecanismo cujos órgãos são homens preferimos seguir os desenvolvimentos de Marx em O Capital, onde utiliza a expressão 'fábrica' para designar aquilo que difere da 'oficina' (Figueroa, 1986, p. 27).

A situação a que estamos nos referindo – a fase inicial de concepção na produção de *softwares* – é caracteristicamente manufatureira (subsunção formal), ainda que totalmente articulada com um processo de automatização (Terceira Revolução Industrial) que levou a subsunção real a níveis extremos. Na verdade, o conceito do próprio Figueroa, de **oficina de progresso tecnológico** seria o mais adequado nesta situação. Sem entrar nos detalhes do texto, a questão se coloca, para o autor, justamente quando o processo direto da produção "se converteu em esfera de aplicação da ciência" (Figueroa, 1986, p. 41).

Nessas condições, "o trabalho imediato [operacional] guarda uma dependência estreita do trabalho geral [científico]", de modo que "o capital procura organizar este último, o que determina que junto da fábrica encontremos agora também o que chamaremos de oficina de progresso tecnológico" (idem), onde se processam as aplicações produtivas da ciência.[4] É nessa categoria que devemos definir a chamada fábrica de *software*.

Segundo Fernandes (2007), a fábrica de *software* constitui um processo estruturado, controlado e melhorado de forma contínua, considerando abordagens de engenharia industrial, orientado para atendimento a múltiplas demandas de natureza e escopo distintas, visando à geração de produtos de *software*, conforme os requerimentos documentados dos usuários ou demandantes de *software*, da forma mais produtiva e econômica possível.

Devido aos *softwares* serem constituídos de partes menores ou módulos que podem ser construídos e acoplados até a montagem do produto final, as partes

[4] Para maiores detalhes, vide também Bolaño (2010).

individuais podem ser desenvolvidas independentemente umas das outras, desde que planejadas considerando nesse processo de trabalho um alto grau de complexidade e controle (Oliveira e Neto, 2003). Conforme as diversas demandas que podem ser assumidas pela fábrica de *software*, além do grau de **complexidade e modularidade** dos componentes, a forma de organizar a produção pode ser estabelecida de maneiras distintas, a partir de **células de produção**, e é geralmente condicionada pela linha de serviço ou mais precisamente pela linguagem de programação. O arranjo de produção pode ainda estar geograficamente distribuído em uma ou mais células de produção, com a gestão da operação centralizada em relação a todas as células. Dessa forma, o processo de produção ocorre em cada local e as etapas de concepção e execução se distinguem acentuadamente, tanto em relação ao grau de importância que assumem no processo, quanto geograficamente.

A manufatura de *software* envolve pouca ou nenhuma produção tradicional: todo sistema é único; apenas partes individuais podem aparecer repetidamente em mais de um sistema e todo o processo é assistido por ferramentas computacionais específicas (CASE) que permitem o uso coordenado dos **componentes de *software* reutilizáveis**. Esses componentes de *software* são trechos de códigos reaproveitáveis que desempenham papéis definidos. Apesar de cada sistema ser único, é comum a ocorrência de componentes similares entre *softwares* em desenvolvimento. Na verdade, uma importante característica de uma fábrica de *software* é a importância dada à **informação acumulada** de vários projetos.

Essa informação pode adquirir várias formas, incluindo elementos reutilizáveis (código, projeto e documentação), medidas de *performance*, relatórios da aplicação de técnicas específicas e processos de desenvolvimento (Fernström et. allii. 1992). Nesse caso, seria mais adequado chamar esse tipo de manufatura de **manufatura assistida por computador**, pois as ferramentas CASE se apresentam como instrumentos de trabalho que devem ser utilizados na atividade de concepção do *software*, sendo que os trabalhadores intelectuais desempenham papel fundamental durante a etapa de concepção, cabendo às ferramentas um grau de reuso, padronização e controle.

No **cenário 2**, o grau de dependência entre trabalho vivo e *software* se inverte, com o elevado grau de automatização obtido a partir da aplicação do *software* gerado, por exemplo, na aplicação do *software* CAD/CAM em máquinas automatizadas na produção industrial. Nesse caso, durante a fase de produção industrial no chão de fábrica, são realizadas adaptações do *software* às máquinas automatizadas por meio de

ajustes através do *software* e não mais pela alteração interna (eixos e engrenagens) das mesmas.

Se observarmos mais detalhadamente o cenário 2, verificaremos que existe uma etapa de concepção (*design*) do produto e uma etapa de produção propriamente dita. O *software* assume papel distinto e fundamental, tanto na fase de concepção automatizada do produto, via CAD/CAM, quanto na fase de produção, controlando as máquinas automatizadas. Isto é diferente do primeiro cenário, em que o *software* assume o papel de ferramenta de concepção e desenvolvimento de um produto que é também um *software*.

Em outros termos, os sistemas CAM do cenário 2 são construídos a partir de ferramentas de *software* cujo processo de desenvolvimento é marcado por fases similares às da produção industrial automatizada (concepção-execução), enquanto que, os sistemas de produção de *software* por meio de *software* do cenário 1 são organizados de forma manufatureira, como nas oficinas de progresso tecnológico citadas. Num caso há um avanço crucial da subsunção do trabalho intelectual, permitido por uma nova onda, avassaladora, de automatização, fruto da revolução micro-eletrônica, mas esse avanço depende de um processo anterior (o outro caso) de criação em que a subsunção do trabalho intelectual enfrenta limites fundamentais. Essa dependência do segundo cenário em relação ao primeiro é a mesma que existem entre trabalho imediato e trabalho geral, que no limite determina, no concernente à divisão internacional do trabalho, a diferença entre desenvolvimento e subdesenvolvimento (Figueroa, 1986).

3. Linha de produção, controle e gerência

Durante a etapa de **modelagem (ou concepção)**, do primeiro cenário, o *software* é criado por trabalhadores especializados em engenharia de *software*, que utilizam ferramentas CASE, as quais, através de um ambiente gráfico virtual, provido de diagramas lógicos, oferecem aos engenheiros de *software* todas as facilidades para a concepção do sistema a ser codificado, produzindo como resultado, o **código inicial** (ou diagramas em alguns casos), pronto para ser detalhado pelo outro grupo de trabalhadores (programadores) durante a fase de execução ou desenvolvimento. Assim, o processo de concepção culmina com a geração, automaticamente, pela ferramenta CASE, do código estruturado, pronto para ser completado na fase seguinte.

A partir daí, os diversos pedaços (**módulos**[5]) do sistema são distribuídos entre os trabalhadores (programadores) da fábrica de *software*, que se ocupam em codificar as especificações oriundas da fase de concepção, a partir do diagrama ou código inicial gerado pelas ferramentas CASE, seguindo padrões e normas de **codificação (programação)**, previamente estabelecidos. Durante o processo de codificação de *software*, vários trechos de código são reutilizados, devido à constante repetição e rotinização da programação, que culmina em trechos de código que são completamente padronizados, criando assim uma **biblioteca de rotinas** e padrões de solução de código para **reuso**[6]. O meio comum que permite aos trabalhadores o compartilhamento das informações necessárias ao desenvolvimento das atividades de concepção são as **redes telemáticas** e o objeto que permite à gerência o controle de tais atividades é o ***software* de controle** aliado a um **repositório de dados** comum entre os envolvidos.

Durante a fase de concepção, os engenheiros de *software* se agrupam em estações de trabalho (computadores dotados das ferramentas CASE, conectados entre si e ao repositório de dados central através das redes telemáticas), a fim de desenvolverem um produto em conjunto, como numa manufatura. A **base de dados** da ferramenta CASE é a fonte de informações com a qual o trabalhador interage durante todo o processo de trabalho. Possui **bibliotecas de componentes e soluções de código**, todas disponíveis ao trabalhador virtualmente de maneira a ser possível projetar o produto de *software* em diferentes padrões conforme a sua aplicação final.

Esses padrões são definidos entre cliente e fábrica de *software* durante a fase de concepção. A ferramenta CASE permite, por exemplo, a concepção de elaborados controles de segurança e robustez do *software* em desenvolvimento, mas cabe ao cliente (demandante do *software*) definir qual o grau de segurança que o *software* deverá apresentar. Geralmente a maior complexidade do código estará ligada ao maior tempo de concepção e conseqüentemente ao seu desenvolvimento (codificação), ocasionando um produto de *software* de custo mais elevado devido ao **tempo de trabalho** necessário.

---

[5] Um módulo é uma parte do sistema que deve desempenhar uma função específica, de maneira que o seu funcionamento deve estar oculto a outros módulos e dedicado a uma função específica, assim como a interface de um módulo deve precisar somente das informações necessárias para que o módulo complete a tarefa (Rumbaugh, 2006).

[6] Reuso é o processo de implementação e atualização de sistemas de *software* utilizando-se *software* e trechos de códigos pré-existentes. O reuso de *software* geralmente ocorre entre sistemas similares ou entre sistemas não similares de forma genérica (Gerard, 2008).

O trabalho produzido por cada engenheiro de *software*, em cada estação de trabalho (computador), pode ser controlado por intermédio da **gerência** através de acesso ao repositório de dados. Durante todo o processo de trabalho, o controle das atividades pode ser monitorado pela gerência a partir da verificação do *status* de desenvolvimento dos diferentes arquivos dos trabalhadores envolvidos no projeto. Trata-se, portanto, de um processo típico dos sistemas conhecidos como "gestão do conhecimento" (Bolaño e Mattos, 2004). Através das consultas de verificação, efetuadas pela gerência, é possível tanto checar o status de desenvolvimento das tarefas de cada engenheiro de *software*, quanto fazer a verificação prévia dos erros técnicos nos arquivos gerados (Baldam & Costa, 2009).

Nesse sentido, as ferramentas disponibilizadas à gerência, além do acompanhamento das atividades de um grande número de trabalhadores de forma simultânea, permitem que todo o controle seja feito **à distância**, ou seja, independentemente de local de trabalho, bastando apenas que a base de dados esteja acessível à gerência pelas redes telemáticas, assim como seus respectivos relatórios, o que obviamente potencializa a capacidade de controle do trabalho durante todo o processo.

O que ocorre na fase de concepção de *software* é um movimento, num sentido, comparável ao observado no **taylorismo**. Este, ao invés de retirar a ferramenta das mãos do trabalhador, colocando-a num mecanismo autônomo, como no caso da automação, mantém aquela ferramenta e padroniza o seu uso. Conforme Moraes Neto (1998, p. 21)

> [...] em vez de se retirar a ferramenta das mãos do trabalhador e colocá-la em um mecanismo, ocorre o contrário: mantém-se a ferramenta nas mãos do trabalhador e vai-se, isto sim, dizer a ele como deve utilizar essa ferramenta; ou seja, ao mesmo tempo que se mantém o trabalho vivo como base do processo de trabalho, retira-se toda e qualquer autonomia do trabalhador que está utilizando a ferramenta. Essa é a idéia do taylorismo: é o controle de todos os tempos e movimentos do trabalhador, claro que de forma necessariamente despótica.

Dessa forma, o trabalho vivo é mantido como base do processo produtivo, e se retira a autonomia do trabalhador que faz uso da ferramenta. Tenta-se, assim, a transformação do homem em máquina e não propriamente substituição do homem pela máquina. Na **linha de montagem fordista**, essa mesma lógica se exacerba no sentido de objetivar o elemento subjetivo, ao fixar o trabalhador a seu posto de trabalho, automatizando o transporte do produto em elaboração, de modo que cada trabalhador só executa movimentos repetitivos ao longo do seu turno de trabalho. O fordismo se desenvolve, como o taylorismo, a partir da observação contínua dos movimentos dos

trabalhadores. As atividades são planejadas com antecedência e cada homem recebe instruções completas, pormenorizando a tarefa executada. O objeto de trabalho é transportado sem a interveniência do homem, fazendo com que cada ofício tenha participação específica no processo de produção.

No caso da **fase de concepção de *software*** que estamos analisando, subordinada a uma gerência científica com as características aqui mencionadas e amparada pelos *softwares* de controle, apesar das ferramentas de suporte à concepção desempenharem um papel importante no direcionamento do trabalho, o engenheiro de *software* ainda mantém o controle do processo de trabalho. Assim, a taylorização não ocorre na fase de concepção do *software*, apenas na execução. A subsunção do trabalho no capital, no caso dos processos de desenvolvimento das ferramentas de concepção, com a incorporação do conhecimento do engenheiro de *software* à ferramenta CASE, é apenas formal.

O que ocorre é ainda uma espécie de **acumulação primitiva de conhecimento**, similar ao ocorrido no período manufatureiro prévio à Primeira Revolução Industrial (Bolaño, 2000). Neste caso, por se tratar de um processo extremamente intelectualizado, talvez não seja tão trivial verificar o resultado dessa acumulação primitiva. Segundo Tauile (1984), o trabalho intelectual pode ser de concepção criativa (TICC) ou de execução de rotinas pré-programáveis (TIERPP)[7]. No processo de desenvolvimento de *software*, tal como conduzido em uma fábrica de *software*, devem ser considerados adicionalmente dois fatores importantes.

O primeiro são os diferentes tipos de **linguagem de programação**. Tal fator é estabelecido conforme diferentes arquiteturas de computadores e diferentes finalidades de programação. Em uma mesma arquitetura coexistem diversos tipos de linguagem de programação, com diferentes finalidades, conforme as características do sistema de *software* a ser produzido. Algumas linguagens, por exemplo, são voltadas para o desenvolvimento de *softwares* comerciais, com ênfase em controles financeiros, contábeis e gerência de estoques. Outros são voltados para aplicações mais específicas, como *softwares* de manipulação de imagens e vídeos. Nos primeiros, as linguagens de programação estão voltadas para prover facilidades de armazenamento, organização e acesso a bases de dados, assim como prover facilidades no desenvolvimento de interfaces de uso amigáveis. No segundo caso, as linguagens permitem maior facilidade

[7] Vale conferir também Tauille (2001).

de manipulação e processamento de cálculos complexos na memória do computador com o intuito de simplificar a execução, com o maior grau de consistência possível dos dados.

O segundo fator importante a ser considerado são os **modelos de referência** para o desenvolvimento e controle da organização do trabalho de desenvolvimento de *software*, que se propõem a eliminar as inconsistências, aumentar a clareza e entendimento, estabelecer terminologia comum e definir regras uniformes. [8] Os modelos de referência estabelecem uma padronização capaz de definir as regras segundo as quais o processo de desenvolvimento de *software* deve estar organizado, de maneira a prover os meios de controle do trabalho intelectual do programador e do engenheiro de *software*, na medida em que a fábrica de *software* não possui conhecimento específico em nenhuma área de negócio que os sistemas produzidos se propõem a informatizar.

Assim, na verdade, uma fábrica de *software* não precisa possuir, por exemplo, conhecimento acumulado sobre procedimentos médicos para produzir *softwares* de diagnóstico clínico, conforme a demanda de uma empresa da área de medicina. Em todo caso, a forma como será conduzido todo o processo de concepção e produção desse *software* particular, como de qualquer outro, pelos seus engenheiros de *software*, estará sempre de acordo com os modelos de referência com o suporte das ferramentas de concepção (CASE). É como se, no processo de produção de *software*, comparado com um processo taylorista, o trabalhador (engenheiro de *software*), a partir das ferramentas de concepção, fosse capaz de conceber, a cada solicitação de produção, um novo bem, diferindo do anterior em suas características particulares, mas ainda mantendo as características gerais que o definem como produto.

Assim, a taylorização não pode ser alcançada, devido ao fato de o produto de *software* ter características heterogêneas,[9] de modo que o capital continua, à diferença de outras áreas da produção social, essencialmente dependente do trabalho vivo dos trabalhadores criativos, responsáveis pela concepção dos *softwares* de produção de *softwares*. A idéia de uma acumulação primitiva de conhecimento, esta sim, aplica-se ao caso, pois o modelo produtivo de concepção através de ferramentas CASE, e o uso dos

[8] Um modelo de referência bastante difundido é o CMMI (*Capability Maturity Model Integration*), que fornece orientação para o desenvolvimento de *softwares* procurando assegurar compatibilidade com a ISO/IEC 15504 (SEI, 2007).

[9] Tais características são chamadas na área de informática de **requisitos de produto**, os quais caracterizam o *software* e determinam o uso das linguagens e modelos computacionais mais adequados.

modelos de referência, permitem o enquadramento do trabalho intelectual em exame e um processo recorrente, mas sempre inacabado, de codificação.

4. Conclusão

Segundo Papert, “programar significa, nada mais, nada menos do que se comunicar com o computador” (Papert, 1998, p. 34). Mas o ato de programar a máquina está inserido em um contexto de desenvolvimento e produção definido e pré-determinado, que extrapola o conceito de comunicação e interface entre homem e máquina, caracterizando-se como trabalho conceitual codificado a serviço do capital produtivo.

Nesse sentido, Sohn-Rethel (1989, p. 14) ressalta as características do processo de trabalho com base em quem concebe ou se apropria da concepção intelectual, pois deve-se distinguir, se o fim almejado de um processo de trabalho se encontra idealmente na cabeça daquele que leva adiante o trabalho, ou nas cabeças de vários, que realizam conjuntamente o trabalho, ou então em uma cabeça estranha, que envia aos trabalhadores só partes divididas do processo, as quais de forma alguma significam uma finalidade pretendida, porque para os executores elas são postas por outros. Esse processo tem uma história e a tendência é que, com a automação, o trabalhador seja excluído gradativamente do processo de trabalho, tornando o operário descartável.

A origem da bifurcação entre taylorismo-fordismo e automação pode ser observada a partir de um elemento comum a ambos os processos que, conforme Braverman, indica que os trabalhadores estão atados aos reais processos de trabalho, sinalizando uma forte interdependência entre trabalho vivo e trabalho morto, a partir da dissociação do processo de trabalho das especialidades dos trabalhadores, separação de concepção e execução e utilização do monopólio do conhecimento para controlar cada fase do processo de trabalho e seu modo de execução. Caracteriza-se o taylorismo, portanto, como controle do trabalho através do controle das decisões que são tomadas no curso do processo de trabalho.

Entretanto, durante a fase de concepção de *software*, o trabalho é ditado pelo ritmo do trabalhador e não pelas máquinas (*hardware*) ou ferramentas de criação (*software*). No momento presente, na fase de concepção da produção de *softwares* por meio de *softwares*, a participação do trabalho vivo dos engenheiros de *software* indica

um processo de subsunção formal do trabalho intelectual. A subjetividade envolvida na tarefa de codificação de programas a partir de modelos de referência e linguagens computacionais para a criação de *softwares* é, na verdade, elemento chave para que um processo de subsunção mais avançado possa ocorrer nas estações de trabalho que usam as máquinas automatizadas, reprogramáveis pelos operadores, no chão de fábrica.

Bibliografia

BALDAM, Roquemar; COSTA, Lourenço. **AUTOCAD 2009 - Utilizando totalmente.** Editora Érica, 2009.

BOLAÑO, C. R. S. Economia política, comunicação e globalização. In: BOLAÑO, C. R. S. (org.). **Os Processos de Globalização e Mundialização: Tecnologias, Estratégias e Conteúdo**. Anais do III Colóquio Brasil-França de Pesquisadores. Aracaju: Editora UFS, 1995.

BOLAÑO, C. R. S. **Indústria cultural, informação e capitalismo**. São Paulo: Hucitec, 2000.

BOLAÑO, C. R. S. Trabalho Intelectual, Comunicação e Capitalismo. A re-configuração do fator subjetivo na atual reestruturação produtiva. **Revista da Sociedade Brasileira de Economia Politica** , Rio de Janeiro, 2002.

BOLAÑO, C. R. S. Productive restructuring, subsumption of intelectual labour, and the contraditory dynamics of development. In: ALBAGLI, S.; MACIEL, M. L. **Information, power and politics: Technological and institutional mediations**. New York: Lexington Books, 2010.

BOLAÑO, C. R. S.; MATTOS, F. M. Conhecimento e informação na atual reestruturação produtiva. Para uma crítica das teorias da gestão do conhecimento. In: **Datagramazero. Revista de Ciência da Informação** 5, n° 3 (junho de 2004). WWW.dgz.org.br (acesso em 10 de janeiro de 2010).

BRAVERMAN, Harry. **Trabalho e Capital Monopolista – A degradação do trabalho no século XX**. Rio de Janeiro, Guanabara, 1974.

CASTELLTORT, Xavier. **CAD/CAM: metodologias e aplicações práticas**. São Paulo. McGraw-Hill, 1988.

FELDENS, Luis Felipe. **Notas de Aula: Sistemas Integrados de Fabricação - Informatização Industrial CAD/CAM**. Disciplina de Sistemas de Fabricação. Pontifícia Universidade Católica do Rio Grande do Sul, PUC RS, Brasil. 2000.

FERNSTRÖM, C., NÄRFELT, K., OHLSSON, L. **Software Factory Principles, Architecture, and Experiments**. IEEE Software, 1992.

FERNANDES, Aguinaldo Aragon. **Fábrica de software: implantação e gestão de perações**. São Paulo, Atlas 2007.

FIGUEROA, Victor. **Reinterpretando el subdesarrollo**. México: Siglo XXI, 1986.

GERARD, R. (2008, 02 26). **Reuse Definitions - software reuse**. Earth Science Software Reuse. Disponível em <http://www.esdswg.org/softwarereuse/Resources/library/reusedefinitions> . Acesso em 06/11/2009.

MORAES NETO, B.R. **Marx, Taylor, Ford as forças produtivas em discussão**. Brasiliense. São Paulo. 1991.

MORAES NETO, B.R. **Fordismo e ohnoísmo: trabalho e tecnologia na produção em massa**. Estudos Econômicos, vol 28, n.2, abril-junho 1998, item 1: Fordismo: trabalho e tecnologia.

MORAES NETO, B.R. **Século XX e trabalho industrial: taylorismo/ Fordismo, ohnoísmo e automação em debate**. São Paulo. Xamã. 2003.

OLIVEIRA, D.; NETO, A. **Fábrica de Software: Promovendo a Criação de Empresas Competitivas em Tecnologia da Informação**. In: ENEGEP, Ouro Preto. Anais Eletrônicos. ABEPRO. 1 CD. Minas Gerais, 2003.

PAPERT, Seymour. **LOGO: computadores e educação**. São Paulo: Brasiliense, 1985.

PFLEEGER, S. L. **Engenharia de Software: Teoria e Prática**. São Paulo: Prentice Hall, 2ª edição, 2004.

RUMBAUGH, James. **Modelagem e Projetos baseados em objetos com uml2**. São Paulo: Elsevier, 2006.

SEI, Software Engenineering Institute. FAQ Disponível em <http://www.sei.cmu.edu/cmmi/faq/15504-faq.html>. Acesso em 06/09/2007.

SOHN-RETHEL, A. (1989). **Trabalho Espiritual e Corporal Para a Epistemologia da História Ocidental**. Rev. u. erg. Neuauflage. Weinheim, VCH, Acta Humaniora, 1989.

SWANSON, Kent; MCCOMB, Dave; SMITH, Jill; McCUBBREY, Don. **The application software factory: applying total quality techniques to systems development**. MIS Quarterly, 1991

TAUILE, José Ricardo. **Microeletronics, Automation and economic development**. Nova York, 1984. Tese de Doutorado, New School for social research.

TAUILE, José Ricardo. **Para (re)construir o Brasil contemporâneo:trabalho, tecnologia e acumulação**. Rio de Janeiro. 2001.